COPPE/UFRJ

ARCABOUÇO AUTONÔMICO DE PADRÕES PARA ELIMINAÇÃO DE DADOS

Wallace Anacleto Pinheiro

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientadores: Jano Moreira de Souza
Geraldo Bonorino Xexéo

Rio de Janeiro
Abril de 2010

ARCABOUÇO AUTONÔMICO DE PADRÕES PARA ELIMINAÇÃO DE DADOS

Wallace Anacleto Pinheiro

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

_____

Prof^{o} Jano Moreira de Souza, Ph.D.

_____

Prof^{a} Geraldo Bonorino Xexéo, D.Sc.

_____

Prof^{o} Alberto Henrique Frade Laender, Ph. D.

_____

Prof^{a}. Ana Maria de Carvalho Moura, Dr. Ing.

_____

Prof^{a} Marta Lima de Queirós Mattoso, D.Sc.

_____

Prof^{o} Geraldo Zimbrão, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

ABRIL DE 2010

Pinheiro, Wallace Anacleto

Arcabouço Autonômico de Padrões para Eliminação de Dados/ Wallace Anacleto Pinheiro. – Rio de Janeiro: UFRJ/COPPE, 2010.

XIX, 387 p.: il.; 29,7 cm.

Orientador(es): Jano Moreira de Souza

Geraldo Bonorino Xexéo

Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2010.

Referências Bibliográficas: p. 224-238.

1. Eliminação de Dados. 2. Redes de Petri de Alto Nível. 3. Computação Autonômica. I. Souza, Jano Moreira *et al*. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

A Deus, responsável por tudo.

À minha querida esposa, Ana Bárbara, e meus filhos, Rafael e Mateus, pelo amor, compreensão, por estarem sempre ao meu lado e serem a razão do meu viver.

A memória de meus pais, que me deram a vida, que me ensinaram a ter coragem e não desanimar, que me tornaram num homem digno, agradeço por cada momento que tivemos.

Agradecimentos

É difícil expressar em palavras o que sinto hoje. Certamente um Obrigado não será suficiente para agradecer a todos que me ajudaram nesta longa e árdua jornada, mas mesmo assim gostaria de deixar registrado o meu agradecimento.

À minha querida família, minha esposa Ana Bárbara e meus filhos Rafael e Mateus. Agradeço pelo amor e carinho que tiveram durante todo este tempo. Agradeço a compreensão pelos momentos em que não pude estar fisicamente presente, mas certamente em todos os momentos vocês estiveram no meu coração.

Aos meus pais que dedicaram momentos preciosos das suas vidas, ajudando a trilhar o caminho correto, abdicando de coisas importantes de suas próprias vidas, estando ao meu lado nos momentos difíceis e ensinando-me a viver.

À UFRJ, especialmente à COPPE/PESC, por oferecer a possibilidade de ter realizado um curso de excelência no nível de doutorado, avaliado com nota máxima na CAPES.

Ao Exército Brasileiro, em especial ao DCT e ao IME, por ter proporcionado a oportunidade de realizar o curso de doutorado, aperfeiçoando minha capacitação técnica e profissional voltada para a área de ciência e tecnologia.

Aos professores Jano e Xexéo, considerados por mim, além de orientadores, amigos. Suas ideias e sugestões contribuíram de forma decisiva para o meu crescimento acadêmico e profissional. Jano, sempre ofertando sugestões criativas, coerentes e que guiam o estado da arte da linha de pesquisa em que atua. Xexéo, com sua visão crítica e construtiva soube guiar com maestria o desenvolvimento dos trabalhos realizados.

À professora Ana Maria que, durante o curso de mestrado, sempre me incentivou a buscar o meu aperfeiçoamento profissional, tornando-se uma fonte de motivação para a realização do curso de Doutorado. Agradeço também por ter gentilmente aceitado o convite de participação para esta banca.

Aos professores Alberto Henrique Frade Laender, Marta Lima de Queirós Mattoso e Geraldo Zimbrão, que ao comporem a banca examinadora, compartilham seus conhecimentos para o aperfeiçoamento do trabalho desenvolvido.

Ao Ricardo, por ter contribuído no desenvolvimento deste trabalho através de sugestões e participações em artigos. Nossas discussões em busca de soluções para os problemas enfrentados serviram de base para várias das soluções propostas nesta tese.

Ao Marcelino, por ter colaborado ativamente na elaboração de artigos, discussão de ideias e desenvolvimento de experimentos realizados durante este trabalho.

Ao Thiago que colaborou na implementação de diversos componentes do arcabouço proposto e na realização dos experimentos com a versão alfa da ferramenta Feed Organizer.

Ao Zé, por ter auxiliado no aperfeiçoamento dos modelos desenvolvidos em UML e pela participação profícua na discussão de trechos relevantes que são apresentados.

À Jonice que serviu de modelo de um bom aluno de doutorado e, durante o início do meu curso, me apresentou o ritmo de trabalho que eu deveria seguir.

A Deus, responsável por todas as maravilhas que ocorrem na minha vida.

A todos: MUITO OBRIGADO!

“Os problemas significativos que enfrentamos não podem ser resolvidos no mesmo nível de pensamento em que estávamos quando os criamos”.
Albert Einstein

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

ARCABOUÇO AUTONÔMICO DE PADRÕES PARA ELIMINAÇÃO DE DADOS

Wallace Anacleto Pinheiro

Abril/2010

Orientadores: Jano Moreira de Souza
Geraldo Bonorino Xexéo

Programa: Engenharia de Sistemas e Computação

Atualmente, pessoas e organizações estão expostas a uma quantidade enorme de dados. Estes dados estão espalhados por diversas fontes, tais como bancos de dados, arquivos de sistemas e serviços *Web*. Quando pessoas e organizações têm que gerenciar estes dados, percebem que grande parte é duplicada, obsoleta, falsa ou não relevante. Este trabalho propõe o uso do Arcabouço Autonômico de Padrões para Eliminação de Dados que auxilia a lidar com essas situações. Os padrões autonômicos são descritos usando redes de Petri de alto nível que modelam de modo formal fluxos de controle e de dados. Com base nestes padrões, é fornecida uma álgebra de eliminação de dados e uma ferramenta para gerenciar *Web Feeds*, chamada *Feed Organizer*.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

PATTERN BASED AUTONOMIC FRAMEWORK FOR DATA KILLING

Wallace Anacleto Pinheiro

April/2010

Advisors: Jano Moreira de Souza
Geraldo Bonorino Xexéo

Department: Computer Science and Engineering

People and organizations today are exposed to a huge amount of data. This data is spread in heterogeneous and un-related data sources, such as the databases, file systems and Web Services. When people or organizations have to manage this data, they notice that large parts of it are found to be duplicated, outdated false or irrelevant. This work proposes the Autonomic Data Killing Pattern Framework to deal with these situations. The autonomic patterns are described using high-level Petri nets that model in a formal way control and data flows. Based on these patterns, is also provided a discarding data algebra and the tool to manage Web Feeds, called Feed Organizer.

Sumário

Índice de Figuras

Índice de Tabelas

Lista de Termos e Abreviações

Termo	*Descrição*
5W1H	*What, Why, Where, Who, When, How* (O que, Por que, Onde, Quem, Quando e Como)
AD ou ad	*Pattern or Operator Allowed Data* (Padrão ou Operador Dados Permitidos)
BI ou bi	*Pattern or Operator Blocking Irrelevant Data* (Padrão ou Operador Bloqueador de Dados Irrelevantes)
BP ou bp	*Pattern or Operator Blocking Prohibited Data* (Padrão ou Operador Bloqueador de Dados Proibidos)
BS ou bs	*Pattern or Operator Blocking SimilarData* (Padrão ou Operador Bloqueador de Dados Similares)
CA	Computação Autonômica
CAN-SPAM	*Controlling the Assault of Non-Solicited Pornography and Marketing Act* (Controle de Ataques referentes a Pornografia e Propaganda Não Autorizada).
CCPN	*Conditional Colored Petri Nets* (Redes de Petri Coloridas Condicionais)
CERN	*Conseil Européen pour la Recherche Nucléaire* (Conselho Europeu para Pesquisas Nucleares)
CPN ML	*Coloured Petri Net Meta-Language* (Meta-Linguagem para Redes de Petri Coloridas)
CPN *Tools*	*Coloured Petri Net Tools* (Ferramentas para Redes de Petri Coloridas)
CWM	*Common Warehouse Metamodel* (Metamodelo de Repositório Comum)
DELPHI	*DEtector with Lepton, Photon and Hadron Identification* (Detector com identificação de Leptón,Fotón e Hadrón)
DI ou di	*Pattern or Operator Discard Irrelevant Data* (Padrão ou Operador Eliminador de Dados Irrelevantes)
DO ou do	*Pattern or Operator Discard Obsolete Data* (Padrão ou Operador Eliminador de Dados Obsoletos)

DP ou dp	*Pattern or Operator Discard Prohibited Data* (Padrão ou Operador Eliminador de Dados Proibidos)
DS ou ds	*Pattern or Operator Discard Similar Data* (Padrão ou Operador Eliminador de Dados Similares)
EC	Evento Composto
ECA	Evento-Condição-Ação
ECA^2	Evento-Condição-Ação-Senão-Ação
EP	Evento Primitivo
ER	Entidade-Relacionamento
ETL	*Extract-Transform-Load* (Extração-Transformação-Carga)
FTC	*Federal Trade Commission* (Comissão Federal de Comércio)
GUI	*Graphic User Interface* (Interface Gráfica do Usuário)
HITS	*Hyperlink Induced Topic Search* (Busca de Tópicos Orientada a Hiperlinks)
IBM	*International Business Machines*
ISO	*International Standard Organization* (Organização de Padrões Internacional)
LC	Lista de Eventos Compostos
LE	Lista de Eventos
LEP	*Large Electron-Positron collider* (Grande Colisor de Elétrons-Prótons)
LIP	Laboratório de Instrumentação e Física Experimental de Lisboa
LOOPN	*Language for Object-Oriented* Petri *Nets* (Linguagem para Redes de Petri Orientadas a Objeto)
LP	Lista de Eventos Primitivos
MOF	*Meta Object Facility* (Arcabouço de Meta-Objetos)
OLAP	*Online Analytical Processing* (Processamento Analítico Online)
OLTP	*OnLine Transaction Processing* (Processamento de Transações Online)
OMG	*Object Management Group* (Grupo de Gerenciamento de Objetos)
PESC	Programa de Engenharia de Sistemas e Computação da Universidade Federal do Rio de Janeiro

PNML	*Petri Net Markup Language* (Linguagem de Marcação para Redes de Petri)
Renew	*Reference Net Workshop*
RSS	*Really Simple Syndication*
RPAN	Rede de Petri de Alto Nível
RPC	Redes de Petri Coloridas
SAS	*Statistical Analysis System* (Sistema de Análise Estatística)
SAP	Sistemas, Aplicações e Produtos
self-CHOP	*Self-configuration, self-healing, self-optimizing, self-protecting* (autoconfiguração, autocura, auto-otimização ou autoproteção)
SGBD	Sistema Gerenciador de Banco de Dados
SHORE	*Scalable Heterogeneous Object REpository* (Repositório de Objetos Heterogêneos Escalável)
SOS	*Second-Order Signature* (Assinatura de Segunda Ordem)
SPAM	*Sending and Posting Advertisement in Mass* (Enviar e Postar Propagandas em Massa)
TRASGO	Laboratório de Tratamento da Sobrecarga da Informação
URL	Universal Resource Location (Localizador Universal de Recursos)
XML	*eXtended Markup Language* (Linguagem de Marcação Estendida)
WfMC	*Workflow Management Coalition* (Coligação de Gerenciamento de Fluxos de Trabalho)

CAPÍTULO 1 INTRODUÇÃO

1.1 Motivação

Devido à criatividade e inteligência humana, uma quantidade de informação[1] cada vez maior está disponível às pessoas todos os dias, aumentando consideravelmente a complexa tarefa de análise. A *Web* na última década destacou-se neste sentido, pois popularizou o acesso e a publicação da informação. Atualmente, qualquer pessoa pode postar informações sobre sua vida pessoal ou profissional, contar suas histórias e experiências, relatar suas opiniões, etc. Isto, em contrapartida, aumentou de forma considerável a quantidade de dados disponíveis. Somam-se a isso, dados provenientes de empresas ou instituições na forma on-line de revistas, jornais, artigos, etc.

Nos últimos séculos mudanças radicais alavancaram a quantidade disponível de informação para as organizações. A revolução industrial no século XIX propiciou grande desenvolvimento em curto espaço de tempo. Essa época ficou conhecida como a época da produção em massa e da eficiência, o que gerou um acúmulo cada vez maior de bens de capital em torno das organizações. Com o passar do tempo, o conhecimento acumulado de forma cada vez mais acelerada levou a uma nova revolução, vivenciada hoje em dia, a revolução da informação (CAVALCANTI, 1995, DRUCKER, 2000).

Os governos também são obrigados a responder às mesmas questões, pois devem apoiar o desenvolvimento das empresas e negócios que envolvam os seus interesses. Um fator complicador é que existe a filosofia de dispersão dos dados, mantidos em diferentes escritórios, sem qualquer política de consistência ou manutenção (BERMAN, 2006). Além disso, nos últimos anos houve uma rápida expansão dos serviços públicos com o consequente incremento de dados a serem gerenciados. Neste cenário, muitas informações vitais para a gestão desses governos estão sendo perdidas. Por exemplo, muitas vezes procedimentos de sucesso realizados pelos funcionários não são capturados e acabam se perdendo quando os mesmos se aposentam ou param de trabalhar por algum motivo, ao mesmo tempo em que

[1] Apesar da diferença entre os termos “dado” (representação/notação) e “informação” (significado/denotação), neste trabalho eles são usados indistintamente.

procedimentos e normas desatualizados ou ineficientes são repassados aos novos funcionários que chegam.

Neste contexto, as organizações e governos vêm adotando diferentes estratégias para lidar com a quantidade de informação cada vez maior. Essas estratégias normalmente seguem modismos de época que normalmente passam pelas fases de introdução, crescimento, maturidade e declínio (SENGE, 2004). Um exemplo desses modismos foi a gestão pela qualidade total nas décadas de 1980 e 1990 (CORDEIRO, 2004), que levou a *International Standard Organization* (ISO) a publicar a série de normas ISO 9000, com o intuito de criar um padrão para a aplicação dos conteúdos de gestão da qualidade às empresas europeias e, posteriormente, do mundo todo. Mais recentemente, surgiram a excelência empresarial (STOLLENWERK, 1999) e a gestão do conhecimento (TARAPANOFF, 2001). A gestão do conhecimento vem ganhando enfoque nos últimos anos e tem sido divulgada como um dos pontos-chave para o sucesso das organizações. Essa gestão envolve diversas etapas que vão desde a identificação do que é importante a organização saber até o descarte do que é ou se tornou não relevante. Simultaneamente a essas estratégias mais gerais, muitas técnicas e ferramentas foram criadas para fornecer um acesso mais seletivo ao dado na tentativa de minimizar os problemas causados pela quantidade excessiva de informação, tais como os Armazéns de Dados (*Data Warehouses*) (INMON et al., 2001, KIMBALL, 2002), os sistemas para gerência eletrônica de documentos (DAVIS & STEVENS, 2006, MICROSOFT, 2006) e os sistemas de gestão de dados SAP (Sistemas, Aplicações e Produtos) (ANDERSON, 2003, BANCROFT et al., 1997). Estes sistemas objetivam gerenciar dados transacionais e gerenciais, e automatizar o processo que controla o ciclo de vida desses dados.

É importante observar que, para todas as estratégias adotadas, uma fase é essencial: a identificação do que é importante saber e, consequentemente, o que não é importante. MORESI & TARAPANOFF (2001) classificam a informação segundo o papel que ela desempenha nas atividades de uma organização, de acordo com a Figura 1.

Figura 1 - Classificação da informação (Fonte: MORESI & TARAPANOFF (2001))

Pode-se destacar na Figura 1 a importância da eliminação da informação sem interesse e, por conseguinte, dos dados associados a essa informação. Há casos em que a informação é aparentemente sem interesse para uma organização, mas muita valiosa para outra, como pode ser visto na Figura 2 (MORESI, 2000). Dessa forma, devem-se avaliar os diversos cenários da informação, ou seja, seus contextos, antes de considerá-la sem importância, pois, por exemplo, em alguns casos a importância de uma informação pode estar no seu valor de troca.

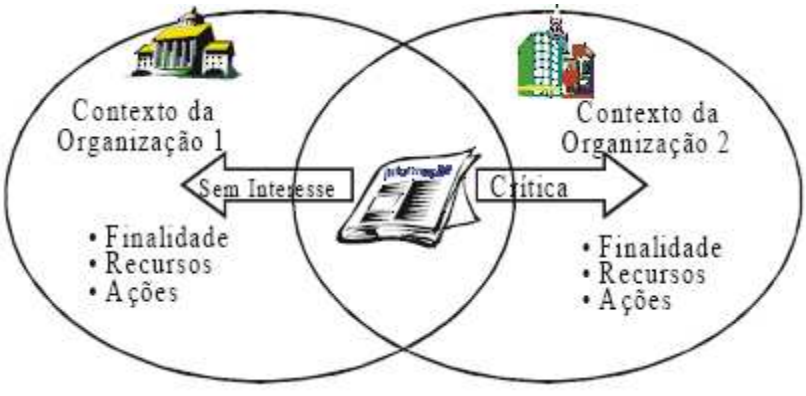

Figura 2 - O valor da informação (Fonte: MORESI (2000))

Apesar de o descarte de dados ser uma importante tarefa no processo de gestão de informações, de modo geral, isto tem sido negligenciado. As instituições enfrentam uma dicotomia, pois estão sempre buscando informações e, ao mesmo tempo, não

conseguem analisar a enorme quantidade de dados disponíveis. Por exemplo, seus empregados produzem, armazenam, compartilham e acessam muitos dados que não são efetivamente controlados. Enquanto isso, muitos dados armazenados por essas empresas encontram-se desatualizados ou apresentam pouca relevância para os negócios. Assim, não são raras as ocasiões em que informações relevantes são descartadas e informações sem importância continuam sendo armazenadas.

O setor financeiro é um exemplo de fonte de dados que gera preocupações no meio empresarial, pois produz um imenso volume de dados difícil de gerenciar. No século passado, a integração dos sistemas de gerenciamento financeiro e seus dados praticamente não existiu e as transações desses sistemas se tornaram tão fragmentadas que a integração efetiva desses dados praticamente se tornou impossível (BARATA et al., 2001). Dessa forma, a tarefa de seleção de dados relevantes e eliminação de dados irrelevantes é bastante complexa.

É importante considerar que, dentro desse universo de informações não gerenciadas, existem informações duplicadas, falsas, erradas, contraditórias, obsoletas, etc. Considerando a *Web* como exemplo, mesmo com as melhorias observadas nas áreas relacionadas à inteligência artificial e interação homem-máquina, nos últimos anos tem se tornado uma tarefa cada vez mais difícil para um ser humano ou sistema computacional processar os dados na mesma velocidade em que são produzidos (ACHENBACH, 1999).

No mundo de hoje, dados não podem ser considerados recursos perenes e intocáveis, sendo necessária a sua frequente reavaliação em busca da manutenção, melhoria da qualidade ou descarte, se for o caso. Para isto, é necessário que os mesmos sejam gerenciados durante todo o seu ciclo de vida, ou seja, desde a sua criação ou obtenção até a sua destruição. Infelizmente, esta não é uma tarefa fácil, pois normalmente existem muitas fontes de dados que fornecem informações variadas, similares e até mesmo conflitantes. Além disso, os dados percorrem caminhos diversos, por vezes, difíceis de gerenciar.

Para todos os cenários apresentados anteriormente, alguns questionamentos são comuns (5W1H), tais como: Dentro da imensa quantidade de informação existente, qual possui relevância no contexto em que está inserida (*What* – O que)? Se não possui relevância, essa informação deve ser descartada ou transferida para repositórios

secundários (*Why*, *Where* – Por que, Onde)? E finalmente: Se a informação for ser descartada ou transferida, por quem, como e quando devem ser realizados os descartes ou transferências (*Who, When*, *How* – Quem, Quando, Como)? Dados precisam ser gerenciados desde sua criação até sua destruição ou transferência para um repositório permanente. Infelizmente essa não é uma prática comum.

Boas práticas a serem aplicadas na gerência dos dados têm sido definidas em várias fontes, tais como em ISO 15489 (2001) e BARATA et al. (2001), o que reforça a importância da definição do destino final dos dados, seja a destruição ou transferência para um repositório permanente.

Como visto até agora, muitas iniciativas tentam organizar e priorizar as informações existentes, pois a quantidade de dados produzidos é muito maior que a quantidade de dados processados pelas aplicações e pelos usuários desses dados. Uma das possibilidades de controle dessa enorme quantidade de dados é a utilização de padrões de eliminação de dados. Uma vez que esses padrões de descarte de dados sejam identificados, diversas aplicações poderão se beneficiar do seu uso, propiciando uma melhor manutenção de seus repositórios de dados.

O desenvolvimento de padrões em sistemas complexos é uma prática comum hoje em dia. Estes padrões permitem nortear o desenvolvimento dos sistemas e a identificação de estruturas e comportamentos já existentes e que podem ser reutilizados. Um padrão normalmente não é algo inovador, mas, muito pelo contrário, algo que, em virtude da excelência na sua concepção e da ampla utilização, torna-se uma referência a ser seguida. A novidade no desenvolvimento de padrões está na sua classificação e documentação. COPLIEN (1996) descreve padrões (*patterns*) como elementos que apresentam um problema e uma solução geral aplicada a um contexto particular.

A abordagem de eliminação de dados proposta neste trabalho utiliza padrões de descarte de dados. Estes padrões, fruto da observação dos processos de eliminação de dados aplicados em diferentes contextos e domínios, combinam fluxos de dados e de controle de modo a fornecer uma visão integrada do processo de destruição dos dados. A modelagem do fluxo de dados mapeia os repositórios, seus relacionamentos e a estrutura dos dados para um modelo em redes de Petri de alto nível. A modelagem do fluxo de eventos utiliza regras ativas do tipo Evento-Condição-Ação que permitem monitorar e agir no ambiente com o mínimo de intervenção humana uma característica

muito importante na área da Computação Autonômica. Nesta tese, todos esses conceitos e ideias são combinados e propostos na forma de um de arcabouço extensível cujo objetivo é minimizar os problemas relacionados à sobrecarga, poluição e explosão de informação. Mais especificamente, dentro deste contexto, pretende-se abordar os problemas relacionados a dados duplicados, similares, obsoletos, irrelevantes e proibidos.

1.2 Hipótese

A sobrecarga de dados afeta os processos de busca e recuperação de informação, sendo que os resultados retornados podem incluir dados pouco relevantes, tais como dados repetidos, com alto grau de similaridade, obsoletos, irrelevantes e proibidos. Se forem fornecidos padrões de descarte de dados e ferramentas automatizadas que possam auxiliar aos usuários na eliminação de dados pouco relevantes, eles poderão adaptar estes padrões de acordo com as suas necessidades, bem como utilizar essas ferramentas de forma a eliminar os dados que considerarem pouco relevantes. Dessa forma, a eliminação de dados pouco relevantes pode auxiliar, de forma significativa, minimizando os efeitos da sobrecarga de dados para os usuários.

1.3 Objetivos do Trabalho

O objetivo geral deste trabalho é desenvolver um arcabouço computacional[2] autonômico[3] para ajudar no tratamento dos problemas relacionados à sobrecarga, poluição e explosão de informação, baseado em padrões de processamento de dados. São propostos como objetivos específicos:

- Desenvolver um método[4] para descrição e formalização de padrões computacionais autonômicos utilizando redes de Petri de alto nível e regras ativas;

[2]Adaptou-se a definição de GOVONI (1999) utilizada para *framework* de modo a definir um arcabouço computacional como uma coleção de classes, interfaces e padrões destinada a resolver uma classe de problemas através de uma arquitetura flexível e extensível. Neste caso, considera-se arquitetura um conjunto de componentes de software, suas propriedades externas e seus relacionamentos com outros softwares.

[3] Programa de computador voltado para o estudo de sistemas complexos com o uso de simulações que é executado com o mínimo de intervenção humana.

[4] Diante das diferentes definições encontradas em diversas fontes sobre metodologia e método, optou-se por utilizar a seguinte definição (HOUAISS, 2001):

- Metodologia - parte de uma ciência que estuda os métodos aos quais ela própria recorre;
- Método - o procedimento, técnica ou meio de se fazer alguma coisa, de acordo com um plano.

- Descrever os principais padrões de eliminação de dados voltados para informações consideradas proibidas, obsoletas, irrelevantes e duplicadas (ou com alto grau de similaridade). Estes padrões de processamento de dados deverão ser identificados, descritos, classificados e correlacionados com os casos de uso existentes de acordo com o método proposto;

- Propor uma álgebra baseada nos padrões de eliminação de dados propostos;

- Desenvolver uma arquitetura para apoiar o arcabouço computacional autonômico proposto;

- Descrever os principais padrões de detecção de eventos compostos utilizando RPAN (Rede de Petri de Alto Nível); e

- Avaliar o modelo por meio de experimentos.

A Figura 3 associa cada um dos objetivos específicos propostos anteriormente com os capítulos e seções relacionadas no texto do trabalho.

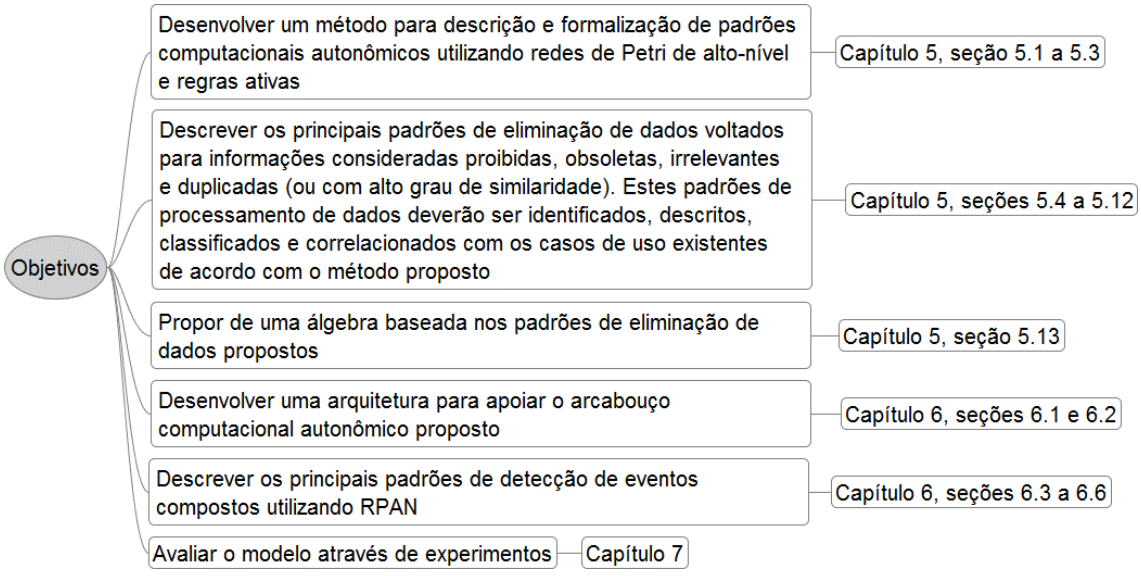

Figura 3 – Objetivos e seções

1.4 Justificativa

A Sociedade Brasileira de Computação recentemente reconheceu a necessidade de trabalhos para tratamento dos problemas associados à explosão, poluição e sobrecarga de informação quando definiu a “Gestão da Informação em grandes volumes de dados multimídia distribuídos” como um dos “Grandes Desafios da Computação no Brasil: 2006 – 2016” (DE CARVALHO et al., 2006). O estudo feito pela SBC discute o

planejamento e pesquisa de longo prazo em Computação no Brasil. Este estudo estabelece os “Grandes Desafios da Computação no Brasil: 2006 – 2016” e revela a preocupação com o crescimento explosivo de dados nos diversos setores empresariais, governamentais e não governamentais. O presente trabalho está alinhado com este objetivo e fornece estratégias para tratar um dos grandes problemas também abordado neste estudo: o problema da redução das massas de dados por meio de modelagem computacional, simulações e outros.

Além disso, outra importante característica deste trabalho é que os softwares propostos na solução desse problema seguem a diretriz do “Planejamento Estratégico para Implementação de Software Livre” (GOVERNO FEDERAL, 2003) que propõe “Priorizar soluções, programas e serviços baseados em software livre que promovam a otimização de recursos e investimentos em tecnologia da informação”.

1.5 Delimitação

É possível considerar a eliminação de dados como uma das possíveis soluções para os problemas associados à explosão, poluição e sobrecarga de dados. Dentro deste contexto, processos mais genéricos também poderiam ser apresentados. Por exemplo, seria possível visualizar a gestão de dados como um processo mais genérico, envolvendo também a produção e o consumo de dados. Da mesma forma, outros processos de eliminação de dados, não tratados neste trabalho, poderiam ser considerados, a exemplo da sumarização de dados. Entretanto, em função da grande complexidade desses problemas, esta tese concentra-se somente nos temas associados à eliminação autonômica de dados, mais especificamente a eliminação dos dados considerados proibidos (por apresentarem conteúdo impróprio), obsoletos, irrelevantes e similares (ou duplicados). Outros processos e propostas de solução não abordados neste estudo poderão ser temas de trabalhos futuros.

1.6 Metodologia de Pesquisa

Trata o presente de um estudo de natureza binária: qualitativa e quantitativa, cujo caráter é exploratório, entendida como mais apropriada às questões aqui abordadas, sendo que ao final os resultados obtidos são mensurados. As informações coletadas foram analisadas e discutidas, sendo dispostas de acordo com as vertentes de ideias apresentadas.

NEVES (1996) ressalta que os métodos qualitativo e quantitativo não se excluem, embora difiram em forma e ênfase. O autor acrescenta que combinar essas técnicas traz vantagens, pois enriquece a visão do pesquisador quanto ao contexto em que a pesquisa é realizada.

TRIVINOS (1994) classifica a pesquisa qualitativa como essencialmente descritiva e que sua atenção preferencial é pelo pressuposto que servem de fundamento à vida das pessoas. RUDIO (2001) entende que na pesquisa descritiva o pesquisador procura conhecer e interpretar a realidade, sem nela interferir para modificá-la. Estando este interessado em descobrir e observar fenômenos, procurando descrevê-los, classificá-los e interpretá-los. Para LÜDKE & ANDRÉ (1990) e SILVA & MENEZES (2001), este tipo de investigação abrange todo um contexto, cujo ambiente natural é a fonte direta de dados e a preocupação é maior com o processo que com o produto, onde o "significado" dado pelas pessoas às coisas torna-se o foco de atenção do pesquisador. Os Capítulos 2, 3, 4, 5 e 6 são exemplos deste tipo de abordagem.

TERENCE & ESCRIVÃO FILHO (2006) acreditam que a pesquisa quantitativa permite a mensuração de opiniões, reações, hábitos e atitudes em um universo, por meio de amostras que os representem estatisticamente. Dentre suas principais características, destacam-se: seguir um plano pré-estabelecido, com o objetivo de enumerar ou medir eventos; utilizar a teoria para desenvolver as hipóteses e as variáveis da pesquisa; examinar as relações entre as variáveis por métodos experimentais ou semi-experimentais, controlados com rigor; utilizar dados que representam uma amostra da população, a partir da qual os resultados podem ser generalizados; e usar, como instrumento para coleta de dados, questionários estruturados, aplicados em entrevistas individuais. O Capítulo 7 é um exemplo da abordagem quantitativa.

1.7 Contribuição e Originalidade

Entre as contribuições apresentadas neste trabalho incluem-se: a classificação e formalização dos padrões de eliminação de dados, a detecção de eventos compostos usando redes de Petri de alto nível com as ferramentas CPN Tools e Britney Suite, a proposta do arcabouço autonômico de eliminação de dados, a álgebra de eliminação de dados construída no SECONDO e a ferramenta Feed Organizer.

As inovações apresentadas nesta tese permitem o tratamento autonômico dos problemas relativos à sobrecarga de dados com o intuito de permitir que as pessoas possam concentrar seus esforços no trabalho sobre os dados relevantes. Apesar dos padrões de eliminação não serem algo novo, é inovadora a sua identificação, elicitação, classificação, bem como a construção de um arcabouço autonômico que permite a aplicação desses padrões. Não foi encontrada na literatura uma abordagem semelhante combinando as estratégias, conceitos e tecnologias propostas. Desta forma, as propostas são originais, não havendo trabalhos que trilhem os mesmos caminhos como uma forma de solucionar os problemas propostos.

1.8 Organização do Trabalho

Além do capítulo introdutório, este trabalho contém mais 7 capítulos. O Capítulo 2 discute os problemas e trabalhos relacionados e o Capítulo 3 faz a revisão da literatura. No Capítulo 4 é apresentada a modelagem de vários conceitos relacionados ou utilizados no presente trabalho. No Capítulo 5 são apresentados: o método para a descrição e formalização dos padrões baseados em RPAN, os padrões de eliminação de dados, bem como a álgebra que associa os operadores a estes padrões. No Capítulo 6 são descritos a arquitetura e o módulo de detecção de eventos compostos. O Capítulo 7 apresenta os experimentos realizados sobre o arcabouço proposto e os resultados obtidos. Finalmente, o Capítulo 8 apresenta a conclusão do trabalho proposto.

CAPÍTULO 2 O PROBLEMA DA EXPLOSÃO, POLUIÇÃO E SOBRECARGA DE DADOS

Este capítulo discute os três grandes problemas associados ao crescimento da quantidade de informação que atualmente apresentam poucas perspectivas de melhoria a curto e médio prazo. São eles:

- **A explosão da informação** – caracterizada pelo crescimento cada vez mais rápido da quantidade de informação, decorrente da facilidade em se criar, compartilhar e disponibilizar a mesma;
- **A poluição informacional** – caracterizada pela reduzida relevância, qualidade e organização da informação;
- **A sobrecarga de informação** – caracterizada pela dificuldade em se lidar com grandes volumes de informação.

Estes cenários já haviam sido previstos desde 1970 por Alvin Toffler em "Future Shock" (TOFFLER, 1984). Ele imaginava um aumento da informação disponível de forma muito rápida, causando o que é conhecido como sobrecarga de informação. Mais do que isso, considerava um espaço informacional poluído, isto é, contendo muita informação de baixa qualidade ou relevância.

O estudo "How much information" (LYMAN & VARIAN, 2003) estima que, entre 1999 e 2002, tenha sido dobrada a quantidade de informação armazenada em papel, filme, meio magnético e mídias óticas. Só em 2002 a quantidade de informação produzida nesses meios alcançou o valor de cinco hexabytes de informação (cinco bilhões de gigabytes), sendo que 92% dessa produção foram guardadas em meios magnéticos, proeminentemente em discos rígidos.

Um estudo mais recente (GANTZ et al., 2007) indica que em 2006 cerca de 161 hexabytes[5] de dados foram criados, capturados ou replicados. A previsão é de que este volume cresça mais de seis vezes até o final de 2010 (cerca de 988 hexabytes). É de se ressaltar que nem toda informação criada é necessariamente armazenada. Por exemplo, uma mensagem instantânea pode ser apagada quanto o software é desligado ou

[5] 1 hexabyte = 1024 petabytes; 1petabyte = 1024 terabytes; 1 terabyte = 1024 gigabytes

informações mais antigas podem ser descartadas para dar lugar a novos dados. Isto é apresentado na Figura 4.

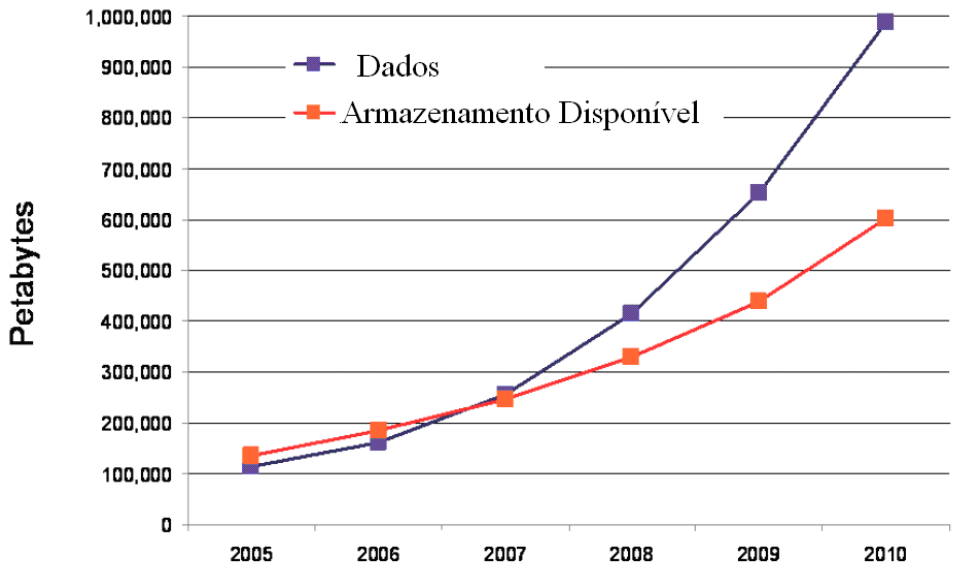

Figura 4 - Dados versus armazenamento (Fonte: GANTZ et al. (2007))

Este incrível crescimento nos leva a um indicador de explosão de dados numa escala sem precedentes na história do homem. Isto tem implicações diretas em diversos setores, quais sejam: privacidade, segurança, proteção de propriedade intelectual, gerenciamento de conteúdo e informação, entre outros.

Na realidade, todos os problemas apresentados anteriormente têm uma origem comum que é a grande quantidade de informação de qualidade diversificada, tratando de assuntos diversos e sem padronização de formato. Dessa forma, informações consideradas proibidas, obsoletas, irrelevantes e duplicadas (ou com alto grau de similaridade) estão presentes em diversos repositórios. Isto, normalmente fruto da explosão de dados, aumenta a poluição informacional e sobrecarga de informação para sistemas e pessoas.

2.1 Trabalhos Relacionados

2.1.1 Visão Geral

Dentro das organizações, muitas vezes, os dados se acumulam sem um controle eficiente, dificultando a sua análise e causando prejuízos às organizações. Dados provenientes de fontes heterogêneas dentro ou fora das empresas podem apresentar

formatos diferentes, conflitantes, duplicados e incompletos. Neste cenário, surgiram técnicas de limpeza de dados, bem como os armazéns de dados (data warehouses), como uma forma de organizar os dados relevantes das organizações (INMON, 1996, RAHM & DO, 2000).

Este problema também é comum no ambiente *Web*, onde, além das páginas, existe a troca de mensagens e documentos. É importante considerar que isto aumentou a possibilidade das pessoas receberem documentos contendo softwares maliciosos ou mensagens não solicitadas (propaganda, conteúdo criminoso, etc), pois cresceu enormente a interação entre pessoas que a princípio não se conhecem. Dessa forma, foram propostas novas formas de selecionar os dados relevantes e bloquear os dados não solicitados ou com conteúdo suspeito, a exemplo dos softwares antivírus, das ferramentas antispam, *antispyware* e de busca na *Web*.

No meio científico, o descarte de dados também é um assunto recorrente, muitas vezes encontrado nos sistemas que devem lidar com o problema da sobrecarga de informação. Em algumas áreas de pesquisa do Programa de Engenharia de Sistemas da UFRJ esta já é uma preocupação antiga. Um bom exemplo disso é a linguagem Fado 2.0 (WERNER, 1992), utilizada para identificar e selecionar eventos físicos interessantes ou promissores, separando-os dos demais eventos, através de suas características. Esta ferramenta lida com um fluxo de dados variável e permite a eliminação de informação irrelevante. Outra ferramenta que também trabalha com fluxo de dados, mais especificamente fluxo de notícias *Web*, é a Yahoo *Pipe*[6] que permite utilizar uma série de filtros encadeados para eliminar e selecionar notícias.

Todas as considerações feitas até agora, apesar de possuírem processos de tratamento dos dados profundamente diferentes, apresentam uma preocupação em comum: prover dados relevantes.

Pode-se fazer uma analogia das estratégias de eliminação de dados com uma fila de candidatos a certo número de vagas onde se deseja selecionar os candidatos mais aptos. Os candidatos passam por um processo de seleção, onde todos são avaliados e os melhores são selecionados. Neste momento podem-se impor estratégias que eliminem previamente os candidatos que não se encaixem no contexto requerido, sem a intenção de selecionar os candidados que efetivamente ocuparão as vagas. Assim, as estratégias

[6] http://pipes.yahoo.com

de eliminação ganham importância, pois permitem diminuir a sobrecarga dos sistemas seletivos, atuando na outra ponta da linha do processo. Ou seja, enquanto as estratégias de seleção têm como foco os dados mais importantes, as estratégias de eliminação têm como foco os dados com pouca ou nenhuma importância para o contexto requerido.

Em virtude da complexidade e generalidade do tema, este trabalho se concentra em cinco categorias de dados que podem se apresentar em diferentes cenários e estão relacionadas à poluição informacional, sobrecarga de informação e explosão da informação, quais sejam: dados obsoletos, irrelevantes, proibidos, similares ou duplicados, e permitidos. O estudo exploratório destes cenários conduziu a um estudo mais aprofundado das áreas apresentadas na Tabela 1. Assim, esta tabela relaciona os cenários problemáticos que serão tratados no decorrer desta tese com algumas das áreas de conhecimentos responsáveis atualmente em lidar com os mesmos.

Tabela 1: Dados Problemáticos e Algumas Áreas de Conhecimento Relacionadas

Cenário	Significado	Áreas
Dados Obsoletos	Dados que perderam a relevância em função da passagem do tempo.	Limpeza de Dados, Filtragem de Dados
Dados Irrelevantes	Dados que não possuem conteúdo de interesse.	Limpeza de Dados, Filtragem de Dados
Dados Proibidos	Dados que não devem estar presentes nos repositórios internos dos sistemas, podendo causar algum prejuízo caso sejam incluídos ou mantidos.	Filtragem de Dados
Dados Similares ou Duplicados	Dados que podem possuir mais de uma versão (similar ou igual).	Limpeza de Dados, Filtragem Dados, Agrupamento de Dados
Dados Permitidos	Dados que devem ser incorporados aos repositórios internos dos sistemas.	Limpeza de Dados, Filtragem de Dados

As áreas apresentadas na Tabela 1 tratam direta ou indiretamente dos processos relacionados à eliminação de dados de maneira diversa e com diferentes níveis de profundidade. Mesmo assim, seu estudo torna-se importante na medida em que se busca uma visão mais completa das diferentes formas de eliminação de dados. Portanto, a análise utilizada neste ponto da tese é uma análise investigativa dessas áreas objetivando-se adquirir uma visão que permite compreender os processos de eliminação de dados em diferentes cenários utilizando dados com diferentes granularidades. Ao

final deste capítulo é apresentado um resumo destas áreas na forma de lições aprendidas. Este resumo busca capturar a essência do conhecimento relacionado a estas áreas, destacando a sua contribuição nas técnicas de eliminação de dados.

As próximas seções apresentam em maiores detalhes as áreas relacionadas na Tabela 1, quais sejam:

- Seção 2.1.2 – Limpeza de Dados;
- Seção 2.1.3 – Filtragem de Dados;
- Seção 2.1.4 – Agrupamento de Dados.

2.1.2 Limpeza de Dados

A área relacionada à limpeza de dados (*Data Cleaning*, *Data Cleansing* ou *Data Scrubbing*) procura detectar e remover erros e inconsistências de dados de maneira a melhorar a sua qualidade (RAHM & DO, 2000). Ela objetiva resolver os problemas advindos da poluição informacional, permitindo tratar dados obsoletos, irrelevantes, similares ou duplicados, e permitidos.

Muitos dos avanços alcançados na área de limpeza de dados são provenientes dos estudos realizados na implementação dos Armazéns de Dados (CHAUDHURI & DAYAL, 1997, INMON, 2001, KIMBALL, 2002). Um armazém de dados (*data warehouse*) é um conjunto de dados orientado por assunto, integrado, variável com o tempo e não volátil, que fornece suporte ao processo de tomada de decisão do negócio (INMON, 1996).

O projeto de um armazém de dados ou de um bazar de dados favorece a criação de relatórios e análise de grandes volumes de dados, bem como a obtenção de informações estratégicas que podem facilitar a tomada de decisão. Eles possibilitam a análise de grandes volumes de dados, coletados dos sistemas transacionais ou OLTP (OnLine Transaction Processing). Por sua vez, os sistemas OLAP (Online Analytical Processing) permitem explorar os dados de um armazém ou bazar de dados.

A conversão dos dados usados nos sistemas OLTP para os sistemas OLAP é feita por ferramentas ETL (*Extract-Transform-Load*) (CHAUDHURI & DAYAL, 1997, INMON, 2001, KIMBALL, 2002). Este processo geralmente envolve: extrair os dados de fontes externas, transformá-los para que atendam às necessidades dos negócios e

carregá-los no armazém de dados. A Limpeza de dados assume um papel muito importante neste processo, pois é responsável por extrair a informação irrelevante capturada dos sistemas transacionais, mantendo somente a informação relevante.

A Figura 5 apresenta os diferentes processos que as ferramentas ETL executam sobre os dados. Todo o processamento ETL é feito em uma área separada, chamada área de transferência (*data staging*), antes de ser incorporado ao armazém de dados. Apesar do grande número de ferramentas ETL existentes, muitas atividades relacionadas à limpeza de dados são feitas manualmente ou por linguagens de baixo nível que são difíceis de escrever, manter e reutilizar.

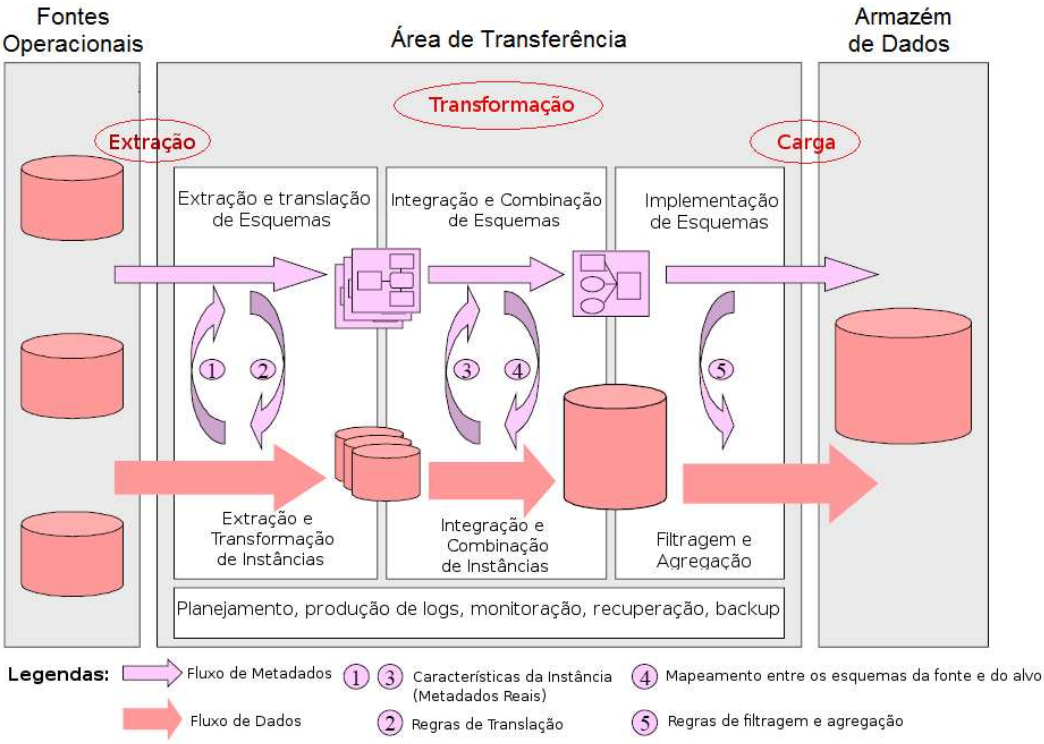

Figura 5 – Processo ETL (Fonte: adaptado de RAHM & DO (2000))

RAHM & DO (2000) utilizam os conceitos existentes nos SGBDs relacionais como apoio para classificar e organizar os problemas que podem ser solucionados pela área de limpeza de dados e utilizados, por exemplo, na implementação dos armazéns de dados. Esses autores classificam os problemas relacionados a esta área como:

- Problemas de uma fonte única de dados – são aqueles associados ao nível de controle dos valores que podem ser incluídos em um repositório. Se o controle não for eficientemente feito, valores errados e inconsistentes poderão ser adicionados. Estes

problemas podem existir no escopo de atributos, registros, tipos de registros e fonte desses dados. Conforme detalhado na Tabela 2, eles podem ser subdivididos em:

- Problemas no nível de esquema – ocorrem em virtude de uma especificação deficiente do esquema de dados ou aplicação das restrições de integridade.
- Problemas no nível de instância – correspondem a erros e inconsistências que não podem ser resolvidas no nível de esquema, tais como: erros de escrita, registros que representam os mesmos valores escritos de forma diferentes, etc.

- Problemas de fontes de dados múltiplas – são aqueles associados à heterogeneidade das fontes. As fontes, além de individualmente poderem apresentar os problemas de fonte única de dados, podem possuir dados em diferentes formatos, com sobreposição e contradição. Os maiores problemas estão relacionados à identificação de sobreposições e de registros que indicam a mesma entidade no mundo real. Conforme detalhado na Tabela 3, estes problemas também podem ser subdivididos em:

  - Problemas no nível de esquema – ocorrem em virtude das diferenças entre os esquemas de dados. Os problemas podem estar associados a nomes utilizados ou conflitos estruturais. No caso dos nomes, pode-se ter, por exemplo, o mesmo nome para entidades diferentes (homônimos) ou nomes diferentes para a mesma entidade (sinônimos). No caso de conflitos estruturais podem existir diferentes representações para a mesma entidade como, por exemplo, representação em atributo versus representação em tabela, tipos diferentes de dados, restrições de integridade diferentes, etc.
  - Problemas no nível de instância – todos os problemas que podem ocorrer no nível de instância para fonte única também podem existir neste caso. Além disso, atributos com mesmo nome e tipo podem ter representações de valores (Estado Civil: solteiro e casado ou SOLT. e CAS) ou diferentes interpretações dos valores (unidades: reais e dólares). Outrossim, a informação pode ser fornecida em diferentes granularidades, como vendas por produto ou vendas por grupo de produtos, e diferentes pontos no tempo, como vendas recentes sendo representadas pelas vendas de hoje ou de ontem.

Tabela 2: Problemas de fontes múltiplas de dados (Fonte: adaptado de RAHM & DO (2000))

	Escopo	Problema	Dado Sujo	Motivo
Problemas no nível de esquema	Atributo	Valores ilegais	Data: 31/02/2009	Valor fora do conjunto de valores permitidos.
	Registro	Violação da dependência entre atributos	Idade atual: 20, data nascimento: 01/01/1950	Idade deveria corresponder ao valor (data atual – data nascimento).
	Tipo de Registro	Violação de unicidade	$Nome_1$: Pedro L., CPF: 12345678910 $Nome_2$: Maria B., CPF: 12345678910	Unicidade do CPF violada.
	Fonte	Violação de integridade referencial	Nome: Maria B., ref: 10	Referência não está definida.
Problemas no nível de instância	Atributo	Valor em falta	Email: xxxx	Campos sem preenchimento ou com valores-padrão.
		Erros de grafia	Peça: caddeira	Erros na hora de digitação dos dados
		Abreviações, valor de significado indefinido	Experiência: A	Utilização de abreviaturas só conhecidas dentro de um determinado contexto.
		Atributos multivalorados	Pessoa= "Wallace, Rua 3, Rio, RJ"	Múltiplos valores em um só atributo.
		Valores incorretos	País: Rio de Janeiro	Valor do campo não corresponde à situação real.
	Registro	Violação da dependência entre atributos	País: Brasil Capital: Buenos Aires	País e capital devem estar de acordo.
	Tipo de Registro	Transposição de palavras	$Nome_1$: João P. $Nome_2$: B. Souza	Ocorre em campos com escrita livre.
		Registros duplicados	$Nome_1$: Jano M. $Nome_2$: J. Moreira	Mesma entidade no mundo real com duas representações diferentes devido a erros na entrada dos valores.
		Registros conflitantes	$Nome_1$: G. Xexéo, data nascimento: 21/09/73 $Nome_2$: G. Xexeó, data nascimento: 03/05/70	Mesma entidade no mundo real é descrita com valores diferentes.
	Fonte	Referências erradas	Nome: Ana Bárbara, ref: 10	Referência está definida, mas com valor errado.

Tabela 3: Problemas de fontes múltiplas de dados (Fonte: adaptado de RAHM & DO (2000))

	Problema	Dado Sujo	Motivo
Problemas no nível de esquema	Homônimos	$Fonte_1$: artigo – nome - manga $Fonte_2$: artigo – nome - manga	Expressões sintaticamente iguais, mas diferentes ao nível semântico em fontes diferentes. A primeira representa uma fruta e a segunda uma parte de uma peça de roupa.
	Sinônimos	$Fonte_1$: profissão - professor $Fonte_2$: profissão - docente	Expressões sintaticamente diferentes, mas iguais ao nível semântico em fontes diferentes.
	Conflito estrutural	$Fonte_1$: pessoa - endereço $Fonte_2$: pessoa - rua $Fonte_2$: pessoa - cidade $Fonte_2$: pessoa - estado $Fonte_2$: pessoa - CEP	Numa fonte a representação de endereço é feita em um único atributo e na outra fonte a representação de endereço é feita em vários atributos. Também se devem considerar tipos de dados e restrições de integridade diferentes.
Problemas no nível de instância	Mesmos problemas de uma fonte única		
	Diferente representação de valores	$Fonte_1$: pessoa - estado civil - casado $Fonte_2$: pessoa - estado civil - CAS.	O valor do atributo surge sob variados formatos de representação.
	Diferente interpretação de valores	$Fonte_1$: carro – preço - 30.000,00 (R$) $Fonte_2$: carro – preço - 30.000,00 (U$)	O valor do atributo surge sob variadas unidades de medida.
	Diferentes Granularidades	$Fonte_1$: vendas – produto - 1000 $Fonte_2$: vendas – produto - 10000	Produto numa fonte representa um grupo de produtos com características semelhantes e produto numa outra fonte representa só um tipo de produto.
	Diferentes pontos no tempo	$Fonte_1$: vendas – recente - 1000 $Fonte_2$: vendas – recente - 2000	Numa fonte, recente representa as vendas de hoje, enquanto na outra representa as vendas de ontem.

Outras áreas que também sofrem dos mesmos problemas apresentados na Tabela 2 e na Tabela 3 são as áreas que envolvem mediadores (*mediators*), encapsuladores (*wrappers*) e sistemas baseados na *Web* (WIEDERHOLD, 1992, ROTH & SCHWARZ, 1997, KUHLINS & STREDWELL, 2003), pois podem ser usados para integrar dados de fontes heterogêneas, que podem apresentar diferentes formatos e conteúdos conflitantes.

Como visto até agora, a área de limpeza de dados está preocupada diretamente com os problemas advindos da poluição informacional. Juntamente com os armazéns de dados, que facilitam a exploração de grandes quantidades e variedades de fluxo de informação, são também propostas soluções para os problemas associados à sobrecarga

de informação. Ademais, os armazéns de dados procuram documentar ao máximo todas as informações das organizações, sendo que, muitas vezes, o descarte das informações passa a ser uma preocupação secundária.

2.1.3 Filtragem de Dados

As técnicas relacionadas à filtragem de dados permitem lidar com os conjuntos de dados abordados nesta tese, quais sejam: dados obsoletos, irrelevantes, proibidos, similares ou duplicados, e permitidos. Assim, o estudo destas técnicas permite a análise de importantes características do processo de eliminação de dados. Neste contexto, as próximas subseções detalham os tópicos relacionados à filtragem de dados que contribuirão na consecução das técnicas propostas no decorrer desta tese, destacando-se os seguintes assuntos:

- Técnicas de Detecção e Proteção contra Softwares Maliciosos;
- Técnicas de Detecção e Proteção contra SPAMs;
- A Ferramenta de Filtragem de Notícias Yahoo *Pipe*;
- A Linguagem de Filtragem de Eventos Fado 2.0;
- Algoritmos de Busca e Recuperação de Páginas *Web*.

2.1.3.1 Técnicas de Detecção e Proteção contra Softwares Maliciosos

As técnicas de detecção e Proteção contra softwares maliciosos permitem lidar com conjuntos de instruções, entendidos neste trabalho como dados proibidos, permitindo que os mesmos sejam eliminados e evitando prejuízos às potenciais vítimas desses dados.

Os softwares maliciosos podem ser definidos como conjuntos de instruções que executam no seu computador e fazem com que o seu sistema execute ações que um atacante quer que ele execute (SKOUDIS & ZELTSER, 2003). Esses softwares podem ser divididos nas seguintes categorias: vírus, vermes, cavalos de troia, *rootkit*, programas espiões e *backdoors*.

O vírus altera um arquivo, geralmente um executável, anexando seu conjunto de instruções de modo a infectar outros arquivos e, possivelmente, causar algum tipo de dano à máquina (SKOUDIS & ZELTSER, 2003, LORIATO & LORIATO, 2008).

Quando executado, o vírus pode, por exemplo, infectar outros arquivos, formatar discos do computador ou prejudicar o seu desempenho, mandar mensagens e imagens para os dispositivos de saída. O vírus se propaga quando um arquivo infectado é transferido de um computador infectado para outro sem vírus.

Os vermes (*worms*) (SKOUDIS & ZELTSER, 2003, LORIATO & LORIATO, 2008) são semelhantes aos vírus, prejudicando o desempenho da máquina, apagando arquivos ou roubando informações dos usuários. No entanto, eles podem se disseminar sem precisar da permissão do usuário. Normalmente, eles usam de programas de email e falhas e/ou vulnerabilidades de redes para se replicar por outras máquinas.

Os cavalos de troia (SKOUDIS & ZELTSER, 2003, SPYMAN, 2004, LORIATO & LORIATO, 2008) vêm disfarçados em programas que parecem inofensivos, mas que podem abrir brechas nos computadores hospedeiros que são usadas para controlar estes computadores remotamente, coletar dados digitados pelos usuários e levar os usuários a navegarem em páginas *Web* falsas, onde outros programas podem ser instalados e usados, sem o conhecimento do usuário, para fins criminosos.

Rootkit é um conjunto maléfico de ferramentas administrativas modificadas para sistemas operacionais baseados em UNIX. Elas objetivam fundamentalmente tomar o controle *root* (raiz) de uma máquina, sem a autorização do administrador (SPYMAN, 2004). Eles usam as técnicas dos cavalos de troia para se esconder no sistema sem ser detectados. Eles podem cancelar e executar processos e esconder arquivos do sistema operacional.

Os programas espiões (THOMPSON, 2005) recolhem informações sobre o usuário da máquina infectada sem o seu conhecimento e transmite essa informação ao atacante. Seu objetivo não é danificar o computador, mas espionar as atividades executadas pelo usuário. Eles podem coletar informações pessoais do usuário como histórico na *Web* e senhas. Eles também podem alterar as configurações do sistema gerando trocas de páginas por propagandas.

Backdoors (ZHANG & PAXSON, 2000) são pequenos programas ou simples configurações que perpassam a segurança normal das redes e computadores, permitindo que o invasor tenha acesso privilegiado. Os ataques aos computadores são geralmente feitos usando-se outros computadores, conectados a outros computadores, e assim por diante, para que se torne difícil traçar a conexão ou ataque a sua origem verdadeira.

Nesses computadores intermediários, o invasor tem acesso privilegiado, possivelmente por já ter instalado um *backdoor* anteriormente.

A detecção de softwares maliciosos normalmente baseia-se em dois métodos, quais sejam: busca em listas contendo informações sobre as assinaturas de softwares maliciosos e análise de comportamentos conhecidos das ameaças (BAILEY et al., 2007, MESSMER, 2007, LORIATO & LORIATO, 2008). O primeiro método consiste em utilizar uma lista de definições de assinaturas de softwares maliciosos. Seu funcionamento baseia-se no exame do conteúdo da memória e dos arquivos armazenados no computador, comparando os dados obtidos com a base de definições de assinaturas de softwares maliciosos. Portanto, torna-se importante atualizar constantemente o software antivírus de forma a detectar as mais novas ameaças. O segundo método utiliza um algoritmo que analisa comportamentos conhecidos das ameaças (técnicas heurísticas). Este método pode detectar softwares maliciosos que ainda não possuem definições de assinaturas lançadas pelas empresas de segurança responsáveis pelos programas. Outras técnicas também têm sido estudadas na luta contra os softwares maliciosos, que se assemelham às técnicas utilizadas contra SPAM[7], tais como a utilização de listas branca aplicadas a programas, procura e bloqueio de páginas *Web* que podem conter softwares suspeitos, análise de padrões de tráfego, entre outros (BAILEY et al., 2007, MESSMER, 2007).

A defesa contra os diversos tipos de softwares maliciosos normalmente exige um conjunto de estratégias. Primeiramente os usuários devem possuir um software atualizado em sua máquina para a detecção e eliminação de vírus e programas espiões, e um *firewall* instalado. Além disso, entre outras medidas, eles não devem executar programas que não conheçam, abrir emails de origem duvidosa e certificar-se da autenticidade dos sites visitados na *Web*. Um dos problemas é que, depois de instalado, pode ser muito difícil removê-lo. Isto pode exigir inclusive que as máquinas infectadas sejam formatadas.

[7] Alguns autores classificam os softwares de distribuição de SPAM como softwares maliciosos, a exemplo de BAILEY et al. (2007).

2.1.3.2 Técnicas de Detecção e Proteção contra SPAMs

No contexto desta tese, as técnicas de detecção e proteção contra SPAMs são vistas como uma forma de eliminação de dados que visa proteger os usuários de dados que os mesmos não desejam receber.

SPAM pode ser caracterizado como uma mensagem enviada para um grande número de usuários sem que estes a tenham solicitado. Essas mensagens objetivam: disseminar propagandas de produtos e serviços, espalhar vírus, correntes de boatos e mensagens ofensivas, golpes, estelionatos e programas maliciosos.

Uma das possibilidades para o surgimento do nome SPAM é a contração das palavras "*SPiced hAM*" que foi utilizada como nome de um dos produtos da empresa Hormel Foods LLC[8]. O SPAM, um tipo de presunto enlatado, ficou conhecido no ano de 1937 por meio de uma campanha publicitária, sendo que também foi largamente utilizado durante a segunda guerra mundial pelo exército americano. Outras versões, menos populares, associam o termo SPAM a acrônimos (TEMPLETON, 2003). A primeira associa SPAM aos processos de enviar e postar publicidade em massa (*Sending and Posting Advertisement in Mass*), a segunda relaciona SPAM a uma mensagem única para todos os fóruns de discussão (*Single Post to All Messageboards*).

Existem várias razões para a proliferação de SPAMs na *Web*, destacando-se: o lucro obtido com este tipo de propaganda, a facilidade para se obter endereços de potenciais consumidores e o baixo custo para o envio das mensagens. A pesquisa feita por CUKIER et al. (2006) mostra que cerca de 13% dos usuários corporativos da *Web* e 11% dos usuários domésticos já adquiriram produtos por meio desse tipo de propaganda.

Os Estados Unidos, pioneiros nas leis antispam, desde 2003 tentam controlar e definir regras para o envio de SPAM através do o estatuto CAN-SPAM (*Controlling the Assault of Non-Solicited Pornography and Marketing Act*) (FTC, 2003). Este estatuto, definido pelo FTC (*Federal Trade Commission*)[9] órgão responsável pelas leis antispam nos EUA, estipula que SPAM é uma mensagem eletrônica com conteúdo comercial enviada para vários destinatários sem a requisição ou consentimento prévios (FTC, 2003, 2005). Porém, esta definição parece ser incompleta, pois não inclui mensagens

[8] http://www.spam.com/
[9] http://www.ftc.gov/spam/

fraudulentas e tentativas de golpe que também não são solicitadas. Outra definição também muito usada e também controversa é a de que um SPAM é simplesmente uma mensagem não desejada por um usuário. O problema é que uma mensagem que pode ser considerada como um SPAM para um usuário, pode não ser para outro. Então um dos primeiros problemas enfrentados por este estatuto foi encontrar uma definição adequada para SPAM. Além disso, de acordo com o estatuto:

- As informações acerca da origem da mensagem devem ser verdadeiras;
- O assunto deve estar relacionado com o conteúdo;
- Mensagens de propaganda devem indicar claramente o seu propósito;
- O endereço físico do remetente deve ser informado ao usuário;
- Deve estar disponível a opção de não receber mensagens semelhantes do mesmo remetente.

Apesar de prever punições para quem desrespeitar este estatuto, não houve redução no número de mensagens SPAM (NELSON, 2004, BERLIND, 2005). Um dos prováveis motivos é porque o estatuto tentou não interferir na liberdade de expressão, um ponto fundamental dentro da legislação americana. Isto, de certa forma, deixou brechas que permitiram aos responsáveis pelos envios de SPAM adaptar as suas mensagens ao que prevê a legislação, sem necessariamente interromper o envio das mesmas. Além disso, a identificação dos reais responsáveis pelo envio das mensagens é uma tarefa difícil, pois muitas vezes são utilizados falsos remetentes. Atualmente, muitos acreditam que novas leis não serão suficientes para reduzir o número de SPAMs em função do crescente avanço tecnológico. Assim, as ferramentas antispam se revestem de importância ainda maior, pois precisam cobrir as brechas deixadas pelas leis e as constantes evoluções nas técnicas de criação e envio de SPAMs.

As técnicas de combate aos SPAMs normalmente consideram as taxas de falso positivo como mais relevantes que as taxas de falso negativo, pois é preferível receber um SPAM a ter uma mensagem legítima bloqueada. Elas podem ser classificadas em (SAHAMI et al., 1998, ANDREOLINI et al., 2005, BOYKIN & ROYCHOWDHURY, 2005, LIEVEN et al., 2007, MADEIRA, 2007):

- Filtragem baseada em listas negras e listas brancas;
- Uso de pesos e regras;

- Filtros bayesianos;
- Sistemas com autoaprendizado.

As **listas negras** (LIEVEN et al., 2007) são listas das fontes de SPAM contendo os seus endereços de origem ou endereços IP. As **listas brancas** contêm os endereços de origem ou IP das pessoas e servidores confiáveis. Neste tipo de técnica, quando uma mensagem é recebida, primeiro o seu endereço é procurado na lista negra e, caso seja encontrado, a mensagem é classificada como SPAM e, então, descartada. Depois, os endereços são procurados na lista branca e, caso sejam encontrados a mensagem é aceita. Caso a mensagem não seja encontrada em nenhuma das listas então a mensagem deve ser classificada usando outros mecanismos antispam. Atualmente, servidores são usados para compartilhar as listas com outros usuários usando o protocolo DNS (Domain Name System).

Regras e pesos (MADEIRA, 2007) podem ser utilizados na identificação de SPAMs. As regras estipulam os testes a serem executados na mensagem. Os pesos de cada teste indicam a probabilidade de uma mensagem ser SPAM e podem ser positivos ou negativos. O resultado final é a soma ponderada dos resultados obtidos a partir das regras e dos pesos. Caso a soma seja alta, a mensagem é marcada como SPAM e então eliminada. A construção das regras baseia-se na extração de palavras ou frases características das mensagens de SPAM. Em seguida, é utilizada uma base de mensagens identificadas manualmente para avaliar a taxa de falsos positivos e negativos e, caso sejam detectadas taxas altas, a regra que gerou estes erros é refinada até que se alcancem taxas aceitáveis. A determinação do peso da regra geralmente é baseada na percentagem das mensagens de SPAM detectadas por cada regra nessa base de treinamento.

Os **Filtros Bayesianos** (SAHAMI et al., 1998) analisam o corpo inteiro das mensagens procurando palavras-chave ou frases que indicam ou não um SPAM. A representação da Rede Bayesiana é feita por meio de um grafo direcionado acíclico no qual os nós representam variáveis de um domínio e os arcos representam a dependência condicional ou informativa entre as variáveis. Para representar a força da dependência, são utilizadas probabilidades, associadas a cada grupo de nós pais-filhos na rede (SAHAMI et al., 1998). Nessa rede, cada símbolo é representado por um nó e uma aresta entre dois nós indica a probabilidade de influência do nó pai para o nó filho.

Além disso, cada nó da rede é associado a uma tabela de probabilidade condicional, que determina a distribuição de símbolo dados os valores dos seus pais. Um classificador bayesiano utiliza uma rede bayesiana onde existe um nó *Classe*, denotando uma das possíveis $classe_j$ (representando as possíveis classificações que o filtro pode realizar), e vários filhos *símbolo* para cada uma das características testadas. A equação a seguir apresenta a probabilidade de um vetor, formado por símbolos, ocorrer, dada que $classe_j$ ocorre.

$$P(Vetor = vetor \mid Classe = classe_j) = \prod P(Símbolo = símbolo \mid Classe = classe_j)$$

A equação a seguir fornece a probabilidade de uma classe ocorrer dado que um vetor de símbolos ocorre.

$$P(Classe = classe_j \mid Vetor = vetor) = \frac{P(Vetor = vetor \mid Classe = classe_j)P(Classe = classe_j)}{P(Vetor = vetor)}$$

O procedimento para utilização desse método envolve a criação de uma base de símbolos relacionados às palavras que indicam uma probabilidade alta da mensagem ser um SPAM e símbolos relacionados às palavras que indicam uma probabilidade muito baixa da mensagem ser um SPAM. Cada mensagem deve ser decomposta em um conjunto de símbolos correspondentes às suas palavras. A partir desse ponto, um vetor característico da mensagem, contendo estes símbolos, passa então a representá-la.

Atualmente, o processo de separação de símbolos é uma das tarefas mais importantes na detecção de SPAMs utilizando redes bayesianas, pois os responsáveis pela criação de SPAMs introduzem, intencionalmente, caracteres nas palavras tentando fazer a sua identificação mais difícil.

Sistemas com autoaprendizado aprendem com dados passados e presentes e não exigem pouca ou nenhuma interferência humana. Podem envolver:

- Análise de tráfego da rede – características como frequência e distribuição dos envios ao longo do tempo podem ser usadas para caracterizar os endereços que enviam SPAMs (LIEVEN et al., 2007).

- Listas brancas e cinzas – para otimizar o tempo de envio os geradores de SPAM normalmente evitam reenviar mensagens para servidores que comunicam erro na recepção. Dessa forma, ao receber uma mensagem, os servidores analisam se a mensagem pode ser enviada através da verificação do endereço do remetente na lista

branca. Caso isto não ocorra, o endereço do remetente e hora de recebimento da mensagem são armazenados em uma lista cinza. Servidores de mensagem legítimos devem seguir a RFC821 (POSTEL, 1982) e a RFC 2821 (NWG, 2001) e tentar reenviar a mensagem somente após certo tempo (LIEVEN et al., 2007). Assim, quando a segunda mensagem é recebida, os remetentes são incluídos nas listas brancas.

- Potes de mel – são mecanismos utilizados para diminuir a eficiência dos geradores de SPAM e, ao mesmo tempo, coletar informações que podem ser utilizadas para caracterizá-los. Os três principais componentes de um pote de mel são (ANDREOLINI et al., 2005):
    - Servidores falsos – geram um grande número de páginas com endereços eletrônicos falsos. Alguns endereços podem ser usados como recipientes de SPAMs e, assim, coletar características dos geradores e SPAMs.
    - Servidores de envio de mensagens – aparentemente podem enviar mensagens sem nenhuma restrição Este servidor é configurado para informar ao remetente que a mensagem foi corretamente enviada, quando na realidade não são. Estes servidores podem obter informações sobre as mensagens enviadas, origem dos SPAMs e características do tráfego.
    - Servidores que recebem mensagens – estes servidores recebem mensagens de endereços divulgados para serem incluídos nas listas de SPAMs. Eles só recebem mensagens de spam, podendo armazenar várias informações dos seus geradores e dos SPAMs gerados.
- Padrões de Redes Sociais – são construídos grafos com os endereços dos remetentes e destinatários, representando os nós, e os envios de mensagens, representado as arestas. Em redes sociais, os participantes trocam mensagens entre si, criando várias ligações entre os nós. No caso dos servidores que geram SPAM, eles normalmente enviam muitas mensagens para pessoas que não respondem e não se conhecem. Redes que apresentam alto grau de agrupamento podem ser consideradas redes sociais legítimas, enquanto redes com baixo grau de agrupamento são conhecidas como redes antisociais. A seguinte fórmula (BOYKIN & ROYCHOWDHURY, 2005) pode ser utilizada para fornecer o grau de agrupamento dessas redes e identificar os geradores de SPAM, onde N_2 é o número de nós com mais de dois vizinhos, k_i é o

número de vizinho do nó *i* e E_i é o número de arestas que existem entre os k_i vizinhos do nó *i*:

$$C = \frac{1}{N_2} \sum_i \frac{2E_i}{k_i(k_i - 1)}$$

2.1.3.3 A Ferramenta de Filtragem de Notícias Yahoo *Pipe*

Pessoas que recebem notícias *Web* por meio de feeds de notícias normalmente são inundadas diariamente por uma imensa quantidade de informação. Soma-se a isto a carência de ferramentas eficazes na redução da sobrecarga de informação gerada sobre essas pessoas. Dessa forma, o estudo das técnicas utlizadas para a filtragem de notícias *Web* reveste-se de importância no contexto da eliminação de dados.

A ferramenta de filtragem de notícias Yahoo *Pipe*[10] é, atualmente, uma das mais completas ferramentas para manipulação de feeds de notícas na *Web*, porque apresenta a possibilidade de configuração de diferentes filtros que podem atuar conjuntamente sobre um conjunto de feeds de notícias agregados de diferentes fontes, segundo regras definidas pelos seus usuários. Essas regras atuam sobre a maioria dos atributos disponibilizados pelos padrões de feeds *Web* existentes hoje em dia. Grande parte das ferramentas que lidam com feeds de notícias Web limita-se a agregar feeds de notícias de diferentes fontes e ordenar as notícias da mais recente para a mais antiga.

A ferramenta Yahoo *Pipe* filtra as notícias seguindo a ideia de *pipes* da plataforma Linux e Unix. Dessa forma, tomando por base as marcações XML, o Yahoo *Pipe* realiza a filtragem de notícias de acordo com parâmetros, fornecidos pelos seus usuários, aplicados às marcações XML disponíveis. Por exemplo, os usuários podem eliminar notícias de um determinado autor ou contendo certas palavras não desejadas existentes no título ou no corpo (descrição) da notícia. Também é possível combinar essas regras. Adicionalmente, a saída filtrada pode servir de entrada para outra filtragem, e assim sucessivamente, formando um sistema de *pipes* ou "canos" que direcionam somente as notícias desejadas para os usuários. A Figura 6 mostra a interface dessa ferramenta.

Os *feeds* de notícias *Web* são fornecidos por sites como listas e são escritas seguindo especificações baseadas em XML. Atualmente, existem três especificações

[10] http://pipes.yahoo.com

para a criação de arquivos *feed*: RSS 1.0 (*RDF*[11] *Site Summary 1.0*), RSS 2.0 (*Really Simple Syndication* – Distribuição muito simples *2.0*) e Atom. As notícias são apresentadas de acordo marcações XML que identificam informações, tais como: título, assunto, autor, categoria, etc.

Os arquivos XML, disponíveis através de uma URL, podem ser incluídos pelos usuários em um programa leitor de *feeds* (agregador). A partir desse momento, sem visitar o site, os usuários passam a receber informações sobre as atualizações dos arquivos de *feeds* contendo novas notícias. É importante considerar que um único site pode liberar por dia dezenas de notícias e os usuários podem incluir (assinar) diversos sites, gerando uma imensa quantidade de notícias a serem lidas diariamente.

Atualmente, as especificações RSS contêm outros parâmetros além de título, descrição, autor e categoria. A maioria destes parâmetros está disponível na ferramenta Yahoo *Pipe*, porém na prática são pouco usados, pois normalmente não são preenchidos pelos produtores de notícias.

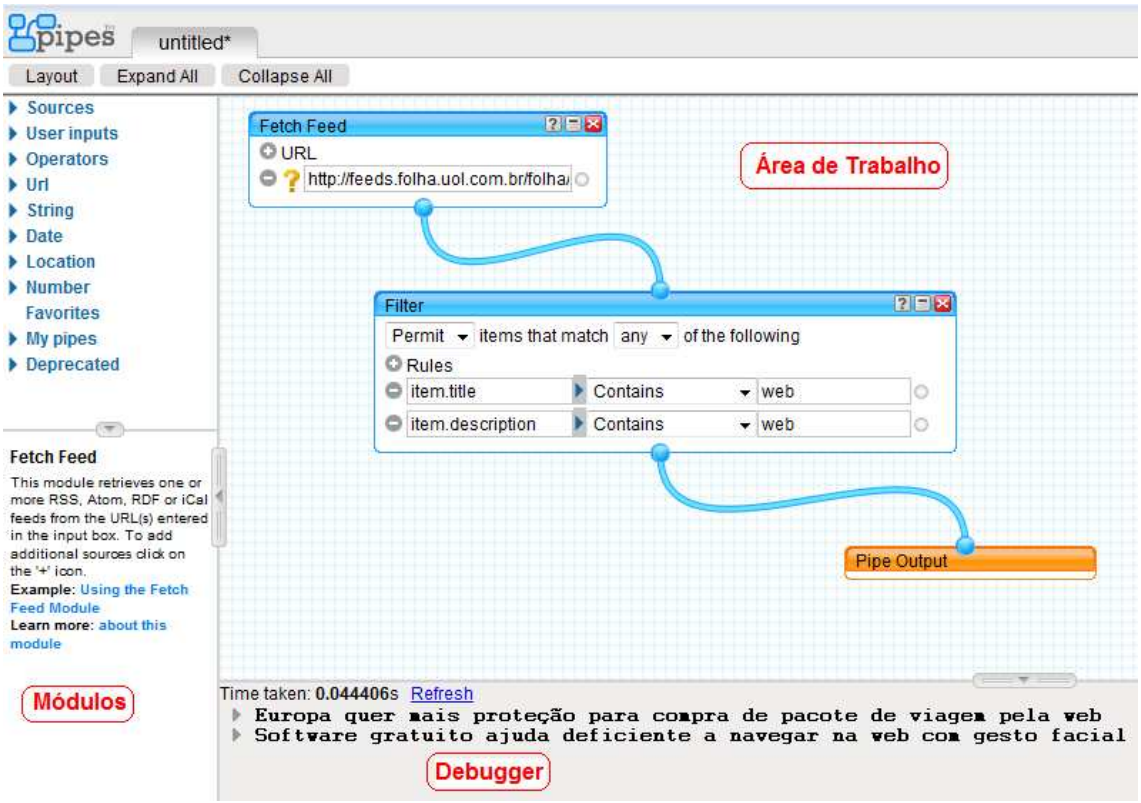

Figura 6 – Yahoo *Pipe*

[11] Resource Description Framework

2.1.3.4 A Linguagem de Filtragem de Eventos Fado 2.0

A linguagem Fado 2.0 é uma linguagem de quarta geração[12], desenvolvida no contexto do projeto DELPHI [13] (*DEtector with Lepton, Photon and Hadron Identification*) do CERN[14] (*Conseil Européen pour la Recherche Nucléaire*), utilizada para seleção, em tempo, de eventos de Física de Altas Energias (WERNER, 1992). Essa linguagem permite separar dados relevantes que ocorrem em um fluxo de dados variável resultante de um experimento de colisão de partículas, por meio da definição de atributos de interesse. Sendo assim, ela pode ser vista como um meio de realizar eliminação de dados considerados irrelevantes, facilitando o trabalho dos pesquisadores na análise de um grande volume de dados.

A linguagem Fado (WERNER, 1992) foi resultante da participação e colaboração da COPPE/UFRJ e do Laboratório de Instrumentação e Física Experimental de Lisboa (LIP) no projeto DELPHI. O projeto da linguagem foi baseado em conceitos e abstrações da lógica matemática e da teoria de conjuntos. Ela é acompanhada de um ambiente de suporte à programação que facilita a criação de programas em Fado, sendo executada no quarto nível de gatilho do experimento DELPHI. Ela é utilizada para identificar e selecionar eventos físicos interessantes ou promissores, separando-os dos demais eventos. A linguagem e o seu ambiente de programação permitem a implementação de um mecanismo de seleção de eventos a partir de critérios físicos estabelecidos. Um programa Fado da versão 2.0 é uma coleção de descrições de reações físicas descritas a partir de um conjunto de condições lógicas feitas sobre quantidades físicas. Por exemplo, selecionar uma reação onde o número de elétrons encontrados no evento deve ser dois e que a energia total deve ser maior que 75 GeVs, a seguinte descrição deve ser feita:

```
REAÇÃO exemplo
MULTIPLICADADE(ELÉTRON) = 2
ENERGIA_TOTAL(ELÉTRON) > 75 GeVs
FIM_REAÇÃO
```

[12] A exemplo da linguagem SQL, é uma linguagem de alto nível que especifica o que deve ser feito e não como deve ser feito, diferenciando-se das linguagens procedimentais de terceira geração.
[13] http://delphiwww.cern.ch/
[14] http://public.web.cern.ch/public/

Neste exemplo, ELÉTRON é a lista ou conjunto de partículas de elétrons encontradas no processo. Cada elemento de uma lista recebe o nome de objeto[15]. Cada objeto possui valores relacionados que descrevem valores físicos do objeto, tais como sua energia, sua massa e coordenadas no espaço. Os valores são chamados parâmetros do objeto. Uma lista pode também possuir valores associados, chamados atributos da lista, que descrevem características gerais da lista, a exemplo do número de elementos (MULTIPLICIDADE) e da energia total dos elementos (ENERGIA_TOTAL).

Uma das características dessa linguagem é que a mesma é específica para os problemas relacionados à seleção de eventos de Física de Altas Energias, sendo que o seu desenvolvimento foi feito em Fortran (TERRY, 1997). Ela não faz uso de modelos (*templates*) externos parametrizáveis que poderiam ampliar o seu potencial reuso. Assim, a adaptação da sua estrutura para aplicações em diferentes áreas fica prejudicada, limitando o uso dessa linguagem. Entretanto, ela fornece uma estratégia interessante para lidar com os problemas de fluxo de dados existentes em muitos cenários de aplicação.

2.1.3.5 Algoritmos de Busca e Recuperação de Páginas *Web*

Uma das primeiras tentativas para diminuir os efeitos da explosão de informação, poluição informacional e sobrecarga de informação veio da área de Recuperação da Informação (BAEZA-YATES & RIBEIRO-NETO, 1999). Os algoritmos de busca clássicos (a exemplo dos modelos booleano, vetorial e probabilístico) classificam documentos a partir de uma análise apenas dos termos que o compõem, não levando em consideração a análise de hipertextos dos documentos *Web* (*links*, tamanho da fonte, estrutura, etc), e nem do contexto no qual os documentos estão inseridos. Por esta razão, os algoritmos de busca clássicos mostraram-se insuficientes no processo de busca por páginas *Web*. A análise de *links* caracteriza uma tentativa de inferir o grau de relevância de uma página a partir da análise da estrutura de *links* da *Web*. Existem vários algoritmos na literatura que propõem a análise da estrutura de *links* de uma coleção de documentos *Web*, com o objetivo de extrair desta estrutura a opinião coletiva dos seus usuários. A abordagem simplista da análise da estrutura de *links* envolve contabilizar o número de páginas que apontam para ela. Quanto maior este número, maior a importância da página. Um problema surge a partir da análise dessa

[15] O termo objeto não está relacionado à definição de objeto do paradigma de orientação a objetos.

abordagem: é mais importante ser apontado por *sites* anônimos ou ser apontado por um *site* conhecido na *Web*? Para solucionar este problema, novos algoritmos foram propostos, tais como o HITS (*Hyperlink Induced Topic Search*) (KLEINBERG, 1999), que considera o conceito de concentradores (*hubs)* e autoridades (*authorities)*, e o *pagerank* (PAGE et al., 1999). A técnica normalmente empregada nos sistemas de busca na *Web*, por exemplo, é o emprego de palavras-chave digitadas pelo usuário para realizar buscas em bases de dados indexadas. São exemplos deste tipo de técnica, as máquinas de busca Google[16] e Yahoo[17]. Outros exemplos são as bibliotecas digitais científicas, tais como a ISI *Web of Knowledge*[18], LANL[19], DBLP[20], ACMDL[21] e a NZDL[22].

Mais tarde, como uma forma de melhorar ainda mais os trabalhos realizados por essas máquinas de busca, surgiram propostas envolvendo o uso de ontologias para o enriquecimento de buscas baseadas em palavras-chave (PINHEIRO, 2004, PINHEIRO & MOURA, 2004). Esses trabalhos revelam a importância da inclusão de regras de acordo com o perfil dos usuários e análise dos relacionamentos envolvendo os conceitos associados aos termos buscados, permitindo a expansão dos termos das buscas e uma melhor contextualização das pesquisas realizadas. Essas expansões, dependendo dos conectivos lógicos (AND ou OR) utilizados entre os termos originais e os incluídos, podem aumentar a abrangência das buscas realizadas pelos usuários, aumentando o número de páginas retornadas, ou especificar as buscas executadas, reduzindo o número de páginas retornadas. Neste sentido, estes trabalhos ganham importância no contexto da eliminação de dados.

É interessante pensar nessas aplicações como seletores de informação relevante, ou seja, elas privilegiam a informação mais importante, segundo algum critério escolhido, em detrimento da informação considerada menos importante. É neste ponto que estas ferramentas contribuem com a proposta apresentada neste trabalho, pois suas estratégias geram um contraponto, indicando, dentro do universo de informações disponíveis, quais informações não devem ser descartadas.

[16] www.google.com
[17] www.yahoo.com
[18] www.isiwebofknowledge.com
[19] arxiv.org
[20] http://www.informatik.uni-trier.de/~ley/db/
[21] http://portal.acm.org/dl.cfm
[22] nzdl.sadl.uleth.ca/cgi-bin/library

2.1.4 Agrupamento de Dados

Apesar de não objetivar a eliminação de dados, o agrupamento (*clustering*) de dados fornece importantes informações, permitindo diferenciar grupos de dados similares de outros grupos não similares. O processo de eliminação ocorre na medida em que um grupo de dados pode ser representado por um número relativamente menor de dados, extraído deste grupo. Por exemplo, uma área diretamente relacionada à eliminação de dados e que usa o agrupamento como uma importante técnica para auxiliar no descarte de dados é a área de deduplicação de dados (CARVALHO et al., 2006, SARAWAGI & BHAMIDIPATY, 2002).

O agrupamento pode ser definido como a técnica que permite agrupar dados de acordo com o grau de similaridade entre eles, sendo que podem ser usadas diferentes técnicas para medir a similaridade (medida de proximidade) entre os elementos. O objetivo é agrupar os dados de modo que os dados dentro de um grupo sejam similares entre si e pouco similares com os dados dos outros grupos (JAIN et al., 1999, BERKHIN, 2006). O agrupamento se baseia em uma classificação não supervisionada onde os grupos são criados sem o conhecimento prévio de qualquer classe de dados e os grupos formados possuam propriedades em comum. Para efeito de comparação, na classificação supervisionada é fornecida uma classificação prévia dos dados, sendo que novas classes de dados são descobertas com base na análise das classes de dados já fornecidas.

As técnicas de agrupamento são normalmente divididas em: hierárquica e particional. Estas técnicas estão intrinsecamente relacionadas a problemas de otimização, pois geralmente envolvem a minimização de uma função considerando certos critérios. Diferentes medidas de similaridade podem ser usadas em ambas as técnicas de agrupamento. Exemplos de medidas de similaridade são a medida do cosseno adotada no modelo vetorial e o coeficiente de Jaccard (BAEZA-YATES & RIBEIRO-NETO, 1999, JAIN et al., 1999).

Para calcular a similaridade entre dois textos utilizando o coeficiente de Jaccard, dois conjuntos de termos são extraídos dos dois textos a serem comparados. O coeficiente de Jaccard corresponde à divisão entre o número de termos comuns aos dois textos pelo total de termos dos textos. Nesta tese, os termos repetidos não são

considerados. Formalmente, sendo A e B dois conjuntos de termos e |conjunto| o número de termos de “conjunto”, tem-se:

$$J(A,B)=(|A \cap B|) / (|A \cup B|)$$

O agrupamento hieráquico utiliza árvores de grupos chamadas dendogramas, sendo que a construção dessas árvores pode ocorrer de forma aglomerativa (*bottom-up*) ou divisiva (*top-down*). Neste tipo de agrupamento, os algoritmos variam o número de grupos a cada iteração, ou seja, os grupos são construídos gradualmente, dividindo ou agregando grupos à medida que a árvore é construída. As medidas de proximidade deste tipo de agrupamento são genericamente conhecidas como *linkage metrics* e definem a proximidade ou distância entre os elementos dos grupos. Assim, o número de grupos resultante varia dependendo do nível de similaridade escolhido. Muitos algoritmos aglomerativos utilizam matrizes de similaridade para agrupar os elementos dos grupos. Essa matriz fornece o nível de similaridade entre todos os elementos de dados sobre o qual será feito o agrupamento. Os algoritmos divisivos normalmente são ineficientes, pois testam todas as combinações de grupo possíveis. Neste caso, seriam necessárias $2^{n-1}-1$ operações na primera iteração. Para efeitos comparativos, a complexidade dos algoritmos aglomerativos é quadrática, enquanto a complexidade dos algoritmos divisivos é exponencial (METZ & MONARD, 2006).

O agrupamento particional fornece uma divisão direta dos dados em grupos, em vez de uma estrutura hieráquica de grupos, como no caso dos dendogramas. Ela parte de uma função objetivo e tenta otimizá-la. Como o processo de análise de todos os possíveis grupos que possam otimizar a função ser computacionalmente custoso, a formação dos grupos utiliza algumas heurísticas, tais como fornecer para o algoritmo um conjunto de pontos iniciais representativo dos grupos. Após este passo, o algoritmo realiza as iterações, alterando a posição inicial dos grupos, dividindo-os ou agregando-os, até que não haja diferença significativa entre a alocação anterior e a atual. O desafio é fornecer um conjunto inicial que permita ao algoritmo convergir rapidamente e de forma correta.

2.2 Lições Aprendidas

Apesar da diversidade de assuntos discutidos nas áreas apresentadas anteriormente, pode-se notar que um fator comum a todas essas áreas é a necessidade de

técnicas que filtrem ou classifiquem as informações segundo fatores de relevância. Consequentemente, todas essas áreas também contribuem na formação de uma visão mais completa das técnicas de eliminação de dados. Dessa forma, observando essas abordagens no que concerne à classificação, ordenação, seleção ou eliminação de dados, algumas lições podem ser extraídas, destacando-se:

- Limpeza de dados – ressalta a importância da análise do conteúdo e dos metadados de forma a obter uma melhoria da qualidade dos dados. Em virtude das diferentes fontes de dados, sejam elas pessoas ou sistemas, muitos dados que representam a mesma entidade no mundo real podem apresentar estruturas e atributos diferentes. Raciocínio similar é válido para dados que aparentemente são iguais em termos de conteúdo e atributos, mas representam entidades diferentes no mundo real quando são analisados o seu contexto e estrutura. Dessa forma, apesar da análise de conteúdo ser uma ferramenta importante para validar e classificar o dado, ela se mostra insuficiente, sendo necessário considerar mais amplamente os metadados relacionados aos dados (seu contexto, sua estrutura e seus atributos). Estas considerações são importantes no momento da eliminação ou não de dados similares.

- Técnicas de Detecção e Proteção contra Softwares Maliciosos – através da análise dos padrões de comportamento dos dados (neste caso, vírus desconhecidos que apresente comportamento similar ao de um vírus conhecido), ressalta a importância da detecção de eventos do ambiente. Dessa forma, esses sistemas sugerem que o conceito da monitoração do ambiente é um importante fator na eliminação de softwares maliciosos, através da busca de padrões de comportamento característicos.

- Técnicas de Detecção e Proteção contra SPAMs – demonstra que a análise dos metadados é um importante fator na seleção de dados relevantes. Os sistemas de detecção de SPAMs permitem eliminar dados considerando, além do conteúdo, as características dos mesmos, tais como frequência, tamanho, remetente, tráfego da rede, entre outros. Outra ideia utilizada nos sistemas de detecção de SPAMs é a criação e compartilhamento de listas brancas e negras que identificam os domínios de servidores confiáveis e não confiáveis. Isto permite descartar ou selecionar grande quantidade de dados com base apenas na origem dos mesmos. Além disso, esses sistemas aplicam outros conceitos importantes no descarte de dados, tais como a definição de pesos e regras, utilização de filtros bayesianos e uso de iscas para selecionar ou descartar dados.

- A Ferramenta de Filtragem de Notícias Yahoo *Pipe* e a Linguagem de Filtragem de Eventos Fado 2.0 – demonstram a importância de criação de uma linguagem flexível para a descoberta de dados relevantes e a consequente eliminação dos dados irrelevantes. Dessa forma, cada usuário pode definir, usando uma série de parâmetros configuráveis, o que ele acredita ser relevante ou não. Isto dá um poder maior aos usuários desta estratégia, pois permite a criação de estratégias particulares de seleção ou descarte de dados.

- Algoritmos de Busca e Recuperação de Páginas *Web* – ressaltam a importância de se analisar não só o conteúdo, mas também o relacionamento entre os dados. Eles podem ser considerados filtros nos quais os dados mais relevantes são selecionados e os menos relevantes são eliminados. Para realizar este trabalho, eles utilizam a ligação (*links*) entre os dados como uma medida de relevância dos mesmos (ex.: algoritmos HITS e Pagerank). Foi essa ideia que no final da década de 1990 transformou o Google numa das mais importantes máquinas de busca existente nos dias atuais. Além disso, a introdução do enriquecimento automático das buscas considerando o perfil de cada usuário e a contextualização de termos da busca por meio da expansão dos mesmos através de ontologias destaca a relevância da contextualização de termos usados na recuperação de dados.

- Agrupamento de Dados – normalmente está associado à análise exploratória dos dados onde existem poucas informações a respeito dos dados (classificação não supervisionada). Neste contexto, o agrupamento fornece novas hipóteses sobre como os dados se relacionam, podendo prover importantes informações para o processo de eliminação de dado. Destaca-se, neste caso, a identificação de dados similares que podem ser descartados, através do uso de medidas de similaridade, a exemplo do a medida do cosseno adotada no modelo vetorial e o coeficiente de Jaccard.

De modo geral, todas as estratégias consideram que a análise somente do conteúdo do dado é insuficiente para definir a relevância ou não dos dados. Isto é um fator importante na generalização das técnicas de descarte de dados a serem propostas neste trabalho. Dessa forma, além das análises relacionadas ao **conteúdo**, devem ser consideradas as análises dos **metadados** (frequências, relacionamentos, contexto, similaridade, entre outros) e a **monitoração do ambiente**. Assim, esses fatores são relevantes nas estratégias de eliminação de dados propostas nos próximos capítulos.

CAPÍTULO 3 REVISÃO DA LITERATURA

A eliminação de dados nos dias atuais é um trabalho árduo, pois a quantidade de dados que as pessoas recebem é muito maior que a capacidade das mesmas em processá-los, sendo necessário o auxílio de novas técnicas e ferramentas suportadas por computador. Neste contexto, a **computação autonômica** surge como uma alternativa interessante, pois propõe que os computadores gerenciem a si mesmos, liberando os seres humanos para tarefas de mais alto nível. Para alcançar o seu objetivo, os sistemas autonômicos precisam ser capazes de monitorar o ambiente em que se encontram, analisar as informações que recebem e atuar de forma conveniente. Uma das técnicas disponíveis, que pode contribuir para realizar essas tarefas, envolve a modelagem de processos através do uso de **regras ativas**. No entanto, se o comportamento das regras ativas não for corretamente orquestrado, elas podem levar os sistemas para estados indesejáveis. De modo a evitar esses problemas, este trabalho propõe o uso de **redes de Petri de alto nível**, pois as mesmas oferecem o suporte necessário, eliminando os problemas relacionados ao uso das regras ativas.

Nesta tese, todos os assuntos mencionados anteriormente são combinados em um arcabouço único, chamado Arcabouço Autonômico de Eliminação de Dados, de forma a prover uma solução para os problemas relacionados à explosão, poluição e sobrecarga de dados. Serão apresentados neste Capítulo os trabalhos relativos à computação autonômica, às redes de Petri e às regras ativas.

3.1 Computação Autonômica

O termo computação autonômica (CA) foi introduzido pela IBM em 2001 (KEPHARD & CHESS, 2003). O objetivo inicial era criar sistemas que pudessem gerenciar por si próprios detalhes de operação e manutenção, liberando os administradores de sistemas para tarefas de alta complexidade. Pode ser entendida como a busca do autogerenciamento de sistemas computacionais utilizando pouca ou nenhuma interferência humana. Ela tem por objetivos:

- Diminuir a complexidade dos sistemas computacionais para os usuários;
- Aumentar a qualidade dos sistemas produzidos;

- Diminuir o tempo de instalação, ciclos de teste, manutenção;
- Antever problemas.

Ela foi inspirada no sistema nervoso autonômico, responsável por cuidar das funções vitais do corpo, enfrentando diferentes condições externas e mantendo equilibrado o estado interno. Ele realiza várias tarefas necessárias ao bom funcionamento do corpo, independente do estímulo consciente do cérebro.

STERRITT (2005) apresenta as seguintes características como potenciais componentes dos sistemas autonômicos:

- Autoconhecimento – os sistemas autonômicos devem "se conhecer" e ser formados por elementos que possuam uma identidade do sistema;
- Autoconfiguração e reconfiguração – os sistemas autonômicos podem ser capazes de se configurar e reconfigurar em condições variáveis e imprevisíveis, caso seja necessário;
- Auto-otimização – os sistemas autonômicos devem procurar formas de otimizar seus trabalhos;
- Autocura – os sistemas autonômicos podem descobrir, diagnosticar e reagir a falhas, sempre que possível, estando aptos a recuperar-se de situações que possam causar mau funcionamento;
- Autoproteção – os sistemas autonômicos devem se proteger contra ataques maliciosos e de problemas relacionados ao mau funcionamento que não foram corrigidos pelas medidas de autocura. Além disso, devem antecipar problemas baseados em relatórios de sensores, tentando evitá-los ou minimizá-los;
- Conhecimento do ambiente – um sistema autonômico conhece seu ambiente e o contexto que cerca suas atividades, agindo de acordo com eles;
- Heterogeneidade de Ambientes – um sistema computacional deve estar preparado para funcionar em ambientes heterogêneos;
- Abstração de complexidades – os sistemas computacionais devem abstrair ao máximo as suas complexidades dos usuários.

Dentro deste conjunto, quatro se destacam como objetivos básicos de um sistema: autoconfiguração, autocura, auto-otimização e autoproteção, também

conhecidos como *self-CHOP* (*self-configuring*, *self-healing*, *self-optimization* e *self-protection*) (MILLER, 2005). As outras quatro propriedades estão relacionadas à forma como um sistema autonômico irá alcançar os seus objetivos.

A IBM (2001) ao propor a CA também definiu cinco níveis de maturidade dos sistemas autonômicos, em ordem decrescente da necessidade de intervenção humana. São eles:

- Nível Básico (manual) – múltiplas fontes de dados do sistema são gerenciadas independentemente. O pessoal especializado agrupa os dados coletados das diferentes fontes, tomam as decisões e realizam as ações necessárias;

- Nível Gerenciado – tecnologia de gerenciamento de sistemas para consolidação dos dados. Os especialistas da área, baseados nos dados coletados, tomam as decisões e realizam as ações necessárias;

- Nível Preditivo – sistemas monitoram, correlacionam e recomendam as ações. Pessoal especializado gerencia o desempenho dos sistemas;

- Nível Adaptativo – sistemas monitoram, correlacionam e executam as ações. Pessoal especializado aprova e inicia as ações;

- Nível Totalmente Autonômico – permite a integração dinâmica de componentes, sendo o gerenciamento feito por regras/políticas de negócio. Usuários focam na criação e alteração das regras que descrevem as necessidades do negócio.

Apesar de a maioria dos sistemas atuais só alcançarem até o nível 3, muitas pesquisas propõem novas aplicações em áreas específicas, aumentando a complexidade dos sistemas autogerenciáveis em várias aplicações (OLIVEIRA et al., 2006, PINHEIRO et al., 2006).

Quanto à arquitetura, a IBM (2005) propõe que os sistemas autonômicos sejam formados por coleções de elementos autonômicos, os quais são sistemas individuais constituintes, contendo recursos e serviços para humanos ou outros sistemas autonômicos. Os elementos autonômicos gerenciarão seu comportamento externo e seus relacionamentos com outros elementos autonômicos de acordo com políticas e regras que humanos ou outros elementos tenham estabelecido. A estrutura interna dos elementos autonômicos retrata bem essa ideia, pois permite armazenar o conhecimento

dos especialistas sobre um determinado domínio. Este conhecimento pode ser utilizado continuamente, conforme pode ser visto na Figura 7, num laço que envolve:

- A monitoração do ambiente externo;
- A análise dos dados capturados;
- O planejamento e escolha das ações a serem tomadas;
- A execução das ações no ambiente.

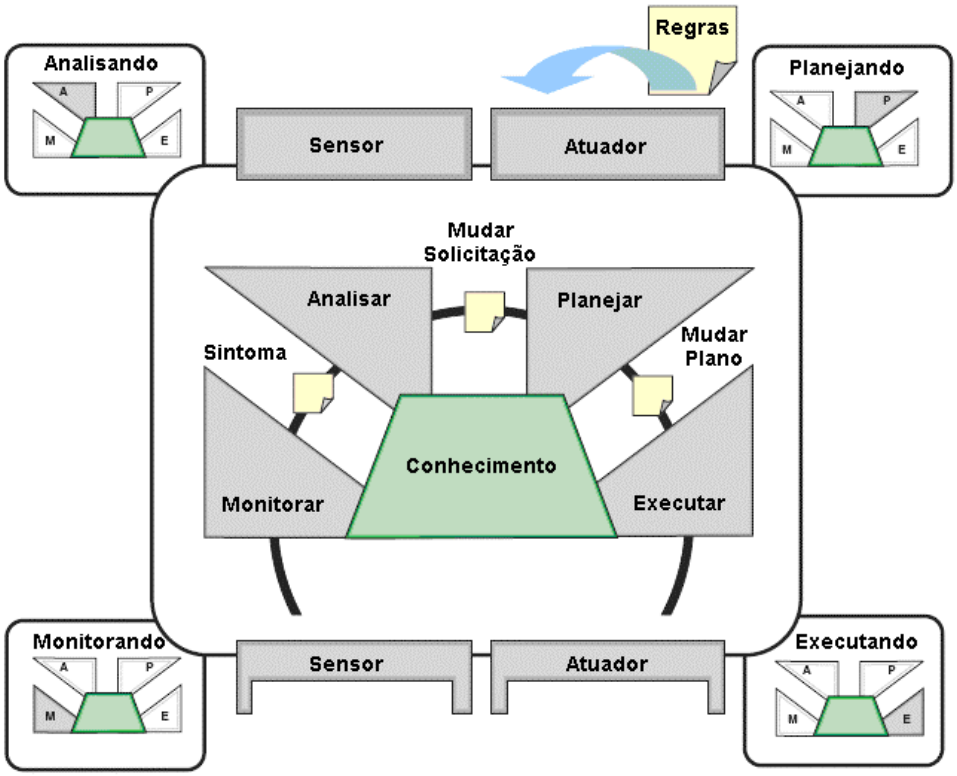

Figura 7 - Estrutura de um elemento autonômico (Fonte: IBM (2005))

Para PERAZOLO (2006) o paradigma da CA fornece conceitos, tais como eventos e sintomas, que facilitam o processamento automático e, consequentemente, a identificação, previsão e tratamento de problemas e incidentes. Evento, neste contexto, é uma ocorrência num instante de tempo que pode ser detectada por um elemento computacional. Um sintoma é a identificação que um possível problema ocorreu. Entre os grandes problemas dessa abordagem estão a detecção e identificação dos eventos e sintomas associados a problemas. Para resolver este problema, podem-se utilizar metodologias de verificação de sistemas e suas possíveis soluções utilizando o

raciocínio baseado em casos passados. Este tipo de solução, que é baseado na monitoração e inferência, necessita de uma representação comum e bem conhecida dos problemas e soluções de um domínio necessitando para isso o uso de ontologias.

Resumidamente, a computação autonômica permite a diminuição da sobrecarga de trabalho sobre os usuários através da maior autonomia dada aos sistemas computacionais. O presente trabalho utiliza esta importante característica no domínio da eliminação de dados. Como visto anteriormente, um dos problemas dessa área está relacionado à detecção e identificação de eventos, para isso propõe-se a utilização de regras ativas, detalhadas na próxima subseção.

3.2 Regras Ativas

Regras ativas podem ser definidas como regras *Evento-Condição-Ação*, onde o *Evento* especifica a ocorrência que dispara a regra, a *Condição* especifica a consulta a ser avaliada quando o Evento ocorre e a *Ação* especifica o ato a ser executado quando o Evento ocorre e a Condição é satisfeita. Elas foram originadas dos SGBDs Ativos (PATON & DÍAZ, 1999) na forma de gatilhos (*triggers)* e regras Evento-Condição-Ação (ECA) e variações, tais como as regras Evento-Condição-Ação-senão-Ação (ECA^2). As regras ativas também foram largamente discutidas na área de Gerenciamento Baseado em Políticas (LYMBEROPOULOS et al., 2003, MANSOURI-SAMANI & SLOMAN, 1997). Elas apresentam aplicabilidade em diversas outras áreas, tais como: sistemas de tempo real, sistemas militares, sistemas cooperativos, fluxos de trabalho (*workflows),* sistemas de monitoração, etc.

Apesar da grande gama de aplicações possíveis, as regras ativas têm sido pouco usadas na prática. Alguns autores acreditam que haja um apoio insuficiente das ferramentas nas fases de análise e projeto das aplicações que trabalhem com regras (BERNDTSSON & CALESTAM, 2003). Mas na realidade, o principal motivo é que o comportamento das regras ativas é difícil de ser corretamente orquestrado (garantido), podendo levar a estados indesejáveis dos sistemas que as utilizam. Por exemplo, caso seja desejável garantir o mesmo resultado final quando duas ou mais regras são disparadas simultaneamente, ou evitar que ocorra o disparo cíclico e infinito de regras.

Para sanar esses problemas, vêm sendo propostas combinações de diferentes linhas de pesquisa, tais como as que envolvem regras ativas e documentos XML

(ABITEBOUL et al., 1999, ABITEBOUL et al., 2002, ABITEBOUL et al., 2003) ou regras ativas e redes de Petri (LI et al., 2004, 2007), sendo esta última combinação também utilizada no presente trabalho. Este estratégia usa prioridade entre regras para garantir o mesmo resultado final quando múltiplas regras são disparadas ao mesmo tempo. Por sua vez, disparos cíclicos e infinitos entre regras podem ser identificados pela análise de grafos de alcançabilidade disponibilizados pelas redes de Petri.

3.2.1 Eventos, Condições e Ações

Estes são os componentes usados em regras ativas. O detalhamento de cada um desses componentes e sua semântica de execução é apresentado a seguir.

3.2.1.1 Evento

Evento é definido como uma ocorrência num ponto do tempo.

3.2.1.2 Condição

Uma vez que a regra é disparada, este elemento especifica uma condição a ser verificada, antes da execução da ação. Em regras ECA a condição é geralmente opcional. Quando a condição não é dada, sempre que um evento for executado, a ação associada também o será. As condições podem envolver os seguintes elementos:

- Predicados – a condição pode testar predicados simples ou compostos (conectivos lógicos), relativo ao estado do SGBD. Por exemplo, número de reclamações dos usuários maior que 10;

- Consultas – consultas podem ser realizadas para recuperar os dados que apresentam a condição verdadeira para a execução de uma ação. Por exemplo, idade das pessoas maior que 65 anos;

- Procedimentos de aplicações – a condição pode ser especificada como uma chamada a uma função, método ou procedimento de uma linguagem de programação de aplicação. Por exemplo, a função *possuiCredito( ).*

3.2.1.3 Ação

A ação é executada quando a regra é disparada e o resultado de sua condição é verdadeiro. As ações podem ser dos seguintes tipos:

- Modificação de dados – são ações que realizam a criação, alteração ou exclusão de dados;

- Recuperação de dados – são ações que realizam a busca de dados;

- Outras ações – são ações relacionadas à definição de dados (*create, alter, drop, rename, truncate*), controle de transações (*commit, rollback*) e controle de dados (*grant, revoke*), etc.;

- Procedimentos de aplicações – a ação da regra pode ser especificada como uma chamada a um método, função ou procedimento escrito em uma linguagem de programação de aplicação.

3.2.2 Modelagem de Regras Ativas

A previsão do comportamento ativo das regras é uma tarefa complexa e que exige técnicas que automatizem o processo. Para o desenvolvimento de aplicações ativas, devem existir ferramentas que permitam o projeto das regras (notação gráfica), sua análise (ferramentas de validação) e verificação (ferramentas de depuração) (PATON & DÍAZ, 1999). Além disso, essas ferramentas devem permitir a detecção de eventos compostos, ou seja, eventos disparados pela combinação de eventos primitivos.

As próximas seções mostram cada uma das etapas necessárias ao desenvolvimento de aplicações ativas.

3.2.2.1 Projeto de Regras Ativas: Modelagem Baseada em Dados ou Processos?

O projeto de regras ativas pode estar associado tanto à estrutura de armazenamento das informações, quanto aos processos que envolvem os dados. Atualmente há técnicas que enfatizam uma ou outra abordagem.

As ferramentas que ressaltam a descrição de aspectos estruturais de informação em um domínio normalmente estendem os modelos estruturais de informação para suportar o comportamento ativo das regras. Seguem este caminho, os trabalhos envolvendo o gerenciamento de expressões (YALAMANCHI et al., 2003), os trabalhos sobre metodologias orientadas a objeto (PATON & DÍAZ, 1999), o método Ross (ROSS, 1997) e o $(ER)^2$ (NAVATHE et al., 1995, TANAKA et al., 1991). Estas abordagens são centradas nos dados, o que na realidade propõe que as regras sejam vistas como uma extensão do modelo de dados.

O segundo conjunto de técnicas e ferramentas busca associar regras aos processos e tarefas dando destaque aos eventos e funções associados aos dados. São exemplos dessas abordagens as que utilizam modelos de transição de estados, gráficos de transição, diagramas de fluxo de dados, Modelo de Processamento Conceitual (*Conceptual Processing Model*), entre outros (BERNDTSSON & CALESTAM, 2003, CASATI et al., 1999, HERBST et al., 1994).

Atualmente, não existe um consenso sobre quais técnicas são mais adequadas para suportar as funcionalidades das regras ativas. Normalmente as ferramentas para projeto de regras são escolhidas em função da área das pessoas que produzem a aplicação. Sendo que as pessoas mais diretamente relacionadas à área de banco de dados preferem as técnicas e ferramentas ligadas à estrutura dos dados, enquanto as pessoas mais ligadas à área de negócios preferem as ferramentas voltadas a processos. Este trabalho tenta resolver este impasse na medida em que utiliza uma abordagem híbrida na qual tanto os dados quanto os processos têm a mesma importância, sendo modelados conjuntamente.

3.2.2.2 Análise de Regras Ativas

As ferramentas de projeto precisam definir formalmente a semântica das regras ativas utilizadas. Dessa forma, evitam-se ambiguidades e descrições incompletas que dificultam a tarefa de análise. Além disso, as ferramentas de análise precisam responder a algumas questões recorrentes (COMAI & TANCA., 2003):

- Ocorrerão disparos infinitos entre regras?
- Regras disparadas simultaneamente levam a um mesmo resultado final, independente da ordem em que são executadas?

A resposta à primeira questão é pesquisada nos trabalhos voltados para a análise de terminação de regras. Na análise de terminação de regras procura-se determinar se um disparo mútuo e infinito de regras irá ocorrer. Alguns autores (BABA-HAMED, 2006, ZIMMER et al., 1996a, ZIMMER et al., 1996b) realizam essa análise através da reescrita de regras (originalmente escritas com outras ferramentas de projeto) em redes de Petri coloridas estendidas e a análise dos grafos de alcançabilidade resultantes.

A segunda questão leva às pesquisas voltadas à análise de confluência das regras. Nestas pesquisas tenta-se determinar se, a partir de um estado inicial de um

SGBD ativo, o estado final não é influenciado pela ordem de execução das regras. Algumas pesquisas (COMAI & TANCA, 2003) resolvem este problema atribuindo diferentes prioridades às regras, o que também é feito nas pesquisas envolvendo redes de Petri de alto nível e confluência de regras (BABA-HAMED, 2006).

3.2.2.3 Depuração de Regras

É importante na criação de aplicações ativas a existência de ferramentas que permitam depurar as regras. Garantir terminação ou confluência a um conjunto de regras não é garantia de que a regra e sua semântica estejam corretas. O sistema deve permitir ao projetista acompanhar passo a passo a evolução das ações para diferentes estados iniciais, fornecendo ao usuário os diferentes fluxos de eventos possíveis e a consequente antecipação a problemas não previstos em tempo de projeto e análise. As redes de Petri com suas ferramentas de simulação são uma solução interessante, pois permitem acompanhar passo a passo a execução das regras e, dessa forma, detectar fluxo de eventos não previstos inicialmente no projeto.

3.2.2.4 Detecção de Eventos Compostos

A combinação de eventos primitivos para o disparo de outros eventos é uma parte natural do processo de modelagem de regras. Os eventos primitivos são detectados por sistemas detectores de eventos primitivos que enviam a informação para a ferramenta que suporta a modelagem de eventos compostos. Uma vez que uma combinação de eventos primitivos ocorra e corresponda a um evento composto modelado, o evento composto é disparado. Este disparo pode gerar o disparo de outros eventos compostos e/ou a sinalização para os sistemas responsáveis pela execução das ações. As redes de Petri podem ser utilizadas como uma poderosa ferramenta de modelagem e detecção de eventos compostos (GATZIU & DITTRICH, 1993, 1994, LI et al., 2004, 2007). Existem também outras abordagens que utilizam árvores compartilhadas (MORETO & ENDLER, 2001) e autômatos (PIETZUCH et al., 2004) na solução desses problemas.

3.2.3 Por que usar Regras Ativas e Redes de Petri?

Como visto anteriormente, a utilização de regras ativas pode levar a estados indesejáveis dos sistemas que as utilizam. Um dos problemas está ligado ao disparo

simultâneo de duas regras, o que pode levar a um resultado final inconsistente (diferente caso as regras sejam executadas em ordens diferentes). A solução deste problema é atribuir prioridades diferentes a cada uma das regras. Dessa forma, elas sempre serão executadas na mesma ordem, caso aconteça um disparo simultâneo. O outro problema das regras ativas é determinar se o processamento irá terminar, ou seja, se não acontecerá o disparo infinito entre regras. As redes de Petri oferecem o suporte necessário para determinar se isso irá ocorrer ou não, através do uso dos grafos de alcançabilidade.

Adicionalmente, as redes de Petri fornecem meios que permitem a detecção de eventos compostos (eventos formados pela combinação de eventos simples e/ou outros eventos compostos), o que pode ser uma necessidade em muitos sistemas que se baseiam em monitoração de eventos.

Dessa forma, os sistemas autonômicos podem usufruir da principal vantagem da utilização das regras ativas no contexto deste trabalho que é o monitoramento do ambiente com o mínimo de intervenção humana. Ao mesmo tempo, eventuais inconvenientes do uso dessas regras podem ser contornados em função da sua modelagem ser baseada em redes de Petri.

3.3 Redes de Petri

Como visto anteriormente, o desenvolvimento de aplicações ativas demanda ferramentas que realizem o projeto das regras, sua análise, verificação e detecção de eventos compostos. Neste sentido, as Redes de Petri fornecem uma abordagem interessante, pois permitem tratar todos estes aspectos. Dessa forma, elas podem ajudar a sanar importantes deficiências das ferramentas que trabalham com regras ativas.

Neste contexto, alguns trabalhos têm sido propostos no sentido de utilizar redes de Petri coloridas com extensões proprietárias para solucionar problemas relacionados à detecção de eventos compostos (GATZIU & DITTRICH, 1994) e terminação de regras (BABA-HAMED, 2006, ZIMMER et al., 1996a, ZIMMER et al., 1996b). Outro trabalho que tenta resolver esses problemas é o de LI et al. (2004, 2007) que propõe uma abordagem baseada na chamada *Conditional Colored Petri Nets* (CCPN), uma rede de Petri colorida estendida usada para a representação e execução de regras ECA. De modo geral, a deficiência desses trabalhos está no fato de utilizarem as redes como uma

ferramenta de análise complementar, não responsável pelo projeto das regras ativas. As regras ativas normalmente são criadas em outras ferramentas ou em arquivos textos que não utilizam a notação das redes de Petri. Isto dificulta o projeto, análise e depuração dessas regras, pois exige técnicas e ferramentas de mapeamento. Outro problema é que esses trabalhos somente apresentam a perspectiva que trata somente eventos e processos, não considerando a perspectiva sobre os dados (ROSS, 1997). Esta abordagem pode esconder detalhes semânticos importantes.

Esta tese procura fornecer uma análise mais abrangente das regras a serem modeladas, propondo, para isso, uma modelagem baseada em fluxo de dados e eventos integrados ao projeto de regras ativas com a utilização de redes de Petri de alto nível. Dessa forma, a seguir são apresentados os conceitos relacionados às redes de Petri e suas extensões.

3.3.1 Conceituação Formal e Gráfica

As Redes de Petri foram definidas por Carl Adam Petri na sua tese de doutorado intitulada "*Kommunikation mit Automaten* (*Communication with Automata*)" em 1962 (PETRI, 1962). Uma rede de Petri possui um conjunto de elementos capazes de descrever e modelar sistemas e/ou suas partes (TANEMBAUM, 2003a, 2003b). Pode ser aplicada na solução de problemas envolvendo, por exemplo, concorrência, controle, conflitos, sincronização e compartilhamento.

Uma rede de Petri é composta pelos seguintes elementos:

- Lugares (*places*) – representam onde estão localizados um evento, um dado, uma atividade ou um recurso;
- Fichas ou Marcas (*tokens*) – representam um evento, um dado, uma atividade ou um recurso;
- Transições (*transitions*) – representam uma condição;
- Arcos (*arcs*) – representam as conexões entre lugares e transições que permite o fluxo de marcas.

Pode-se definir uma rede de Petri como um grafo orientado bipartido definido por $R = \{P, T, A, PA, M_0\}$, onde:

- $P = \{P1, P2, ..., Pm\}$ é o conjunto de nós chamados lugares;

- $T = \{T1, T2, \ldots, Tn\}$ é o conjunto de nós chamados transições;
- $P \cap T = \emptyset \wedge P \cup T \neq \emptyset$ os conjuntos P e T são disjuntos e não vazios;
- $A: (P \times T) \cup (T \times P)$ é o conjunto dos arcos;
- $PA: A \rightarrow N$ são os pesos dos arcos;
- $M_0: P \rightarrow N_0$ é a marcação inicial.

A função de uma transição é representar um evento, ou seja, a transição é responsável por modificar a marcação de uma rede de Petri. Esta ocorrência só é válida quando existem marcas nos lugares de entrada de uma transição.

A Figura 8 e a Figura 9 representam uma rede de Petri com pesos para duas marcações iniciais diferentes. A Figura 8 permite o disparo da transição, pois possui o número de marcas nos lugares de entrada da transição *T1* iguais aos pesos dos arcos de entrada e, dessa forma, habilitam o disparo da transição. Neste caso, também é mostrado o estado da rede após o disparo. A Figura 9 não permite o disparo da transição, pois o número de marcas no lugar *P2* é menor que o peso estabelecido pelo arco de entrada que liga *P2* a *T1*.

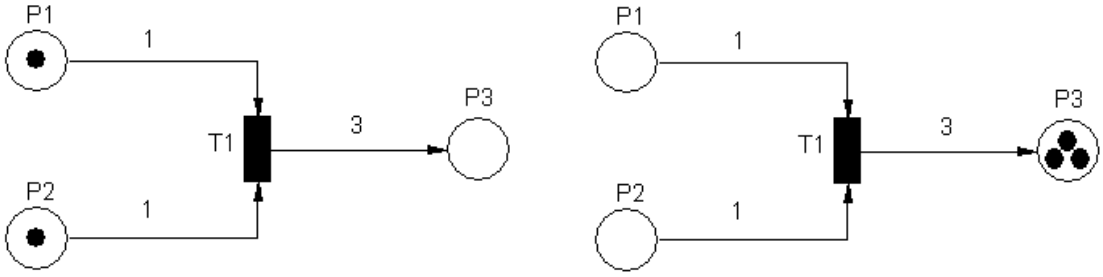

Figura 8 - Situação antes e após disparo da transição T1

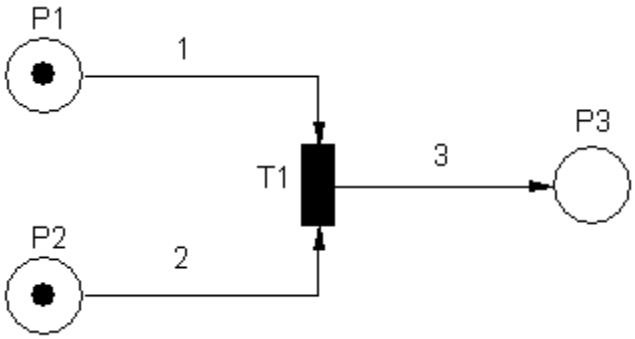

Figura 9 - T1 não pode disparar

Alternativamente, pode-se representar algebricamente a estrutura gráfica de uma Rede de Petri através das seguintes matrizes:

- Matriz de Entrada (*E*) – representa a quantidade de arcos de entrada em cada transição;
- Matriz de Saída (*S*) – representa a quantidade de arcos de saída em cada transição;
- Matriz de Incidência (*I*) – representa a subtração da matriz de saída pela matriz de entrada;
- Marcação inicial (M_0) – representa a marcação inicial das fichas na rede.

Arcos com pesos maiores que um são considerados como se fossem múltiplos arcos de peso 1.

Assim, a rede da Figura 8 tem as seguintes matrizes:

$$E_{TxP} = \begin{matrix} T1 \\ \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \end{matrix} \begin{matrix} \\ P1 \\ P2 \\ P3 \end{matrix}$$

$$S_{TxP} = \begin{matrix} T1 \\ \begin{bmatrix} 0 \\ 0 \\ 3 \end{bmatrix} \end{matrix} \begin{matrix} \\ P1 \\ P2 \\ P3 \end{matrix}$$

$$I_{TxP} = \begin{matrix} T1 \\ \begin{bmatrix} -1 \\ -1 \\ 3 \end{bmatrix} \end{matrix} \begin{matrix} \\ P1 \\ P2 \\ P3 \end{matrix}$$

$$M_0 = \begin{matrix} T1 \\ \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \end{matrix} \begin{matrix} \\ P1 \\ P2 \\ P3 \end{matrix}$$

A matriz de incidência associada a uma rede de Petri corresponde então a sua estrutura, independente da marcação. Na matriz de incidência, cada linha vai corresponder à modificação de uma marcação quando a transição associada é disparada. Por exemplo, a linha da matriz, correspondente ao disparo da transição *T1*, representa

que uma ficha de *P1* e uma ficha de *P2* são extraídas quando *T1* é disparada, enquanto três fichas são depositadas em *P3*.

A equação fundamental, ou equação de estados, possibilita a análise da acessibilidade das marcações, bem como o número de vezes que cada transição deve ser disparada para que se obtenha uma determinada referida marcação.

A Equação Fundamental das Redes de Petri é (MURATA, 1989):

$$M_k\,(p) = M_0\,(p) + I.\,s,\ \forall\, p \in P,$$

onde s é o vetor característico cujos componentes s_i são naturais e representam o número de vezes que cada transição T_i foi disparada para obter-se a marcação $M_k\,(P)$ a partir de $M_0\,(p)$, I é a matriz de incidência e P é o conjunto formado pelos lugares da rede. Assim, pode-se determinar se uma marcação M_k é acessível a partir de M_0. Por exemplo, para as matrizes calculadas anteriormente, após o disparo da transição *T1*, ter-se-ia:

$$M_k = \begin{bmatrix}1\\1\\0\end{bmatrix} + \begin{bmatrix}-1\\-1\\3\end{bmatrix}\ [1] = \begin{bmatrix}0\\0\\3\end{bmatrix}$$

Isto indica que, após o disparo da transição, tem-se três fichas em *P3* e nenhuma ficha nos outros lugares. Resultados negativos para M_k indicariam que este estado não é alcançável. Por exemplo, se a transição *T1* pudesse disparar duas vezes, obteriam-se valores negativos para M_k, indicando que este estado não é alcançável a partir das condições iniciais dadas, pois faltariam fichas nos lugares de entrada.

Um dos problemas na construção da equação fundamental é a obtenção da matriz de incidência para as redes chamadas impuras, ou seja, redes que contém arcos com o mesmo peso e que formam um ciclo fechado entre uma transição e um lugar (*self-loop*). Esses arcos, também conhecidos como arcos de leitura ou de teste (*read arcs*), permitem o disparo de uma transição sem que seja modificada a situação das fichas do lugar de entrada, sendo, algumas vezes, representados por uma única seta bidirecional ligando o lugar à transição. Nestes casos, a rede precisa ser refinada para eliminar esses ciclos, senão a matriz de incidência não mostrará por completo a estrutura da rede. Isto é feito por meio da inclusão de um par lugar-transição auxiliar (*dummy pair*) que elimina esses tipos de ciclo. A Figura 10 mostra uma rede impura e a Figura 11 mostra a inclusão do par lugar-transição que torna a rede pura.

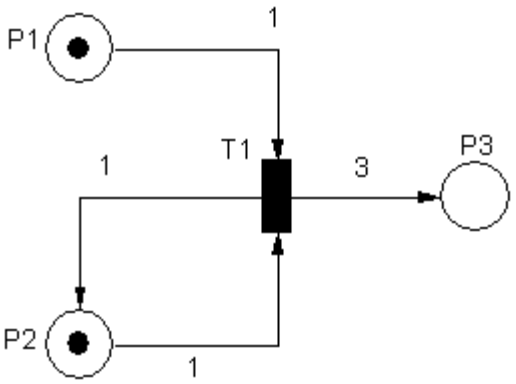

Figura 10 – Rede impura

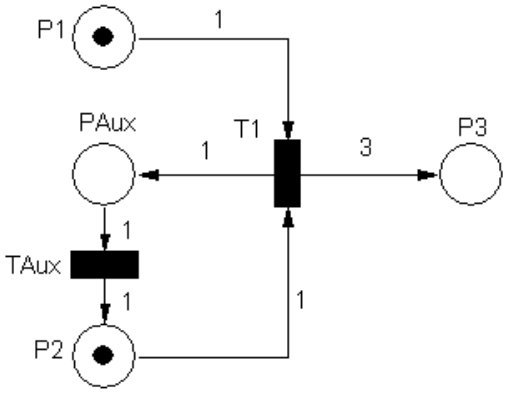

Figura 11 – Rede pura refinada

A rede da Figura 10 apresenta as seguintes matrizes de entrada, saída e incidência:

$$E_{TxP} = \begin{matrix} T1 \\ \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \end{matrix} \begin{matrix} \\ P1 \\ P2 \\ P3 \end{matrix}$$

$$S_{TxP} = \begin{matrix} T1 \\ \begin{bmatrix} 0 \\ 1 \\ 3 \end{bmatrix} \end{matrix} \begin{matrix} \\ P1 \\ P2 \\ P3 \end{matrix}$$

$$I_{TxP} = \begin{matrix} T1 \\ \begin{bmatrix} -1 \\ 0 \\ 3 \end{bmatrix} \end{matrix} \begin{matrix} \\ P1 \\ P2 \\ P3 \end{matrix}$$

A rede da Figura 11 tem as seguintes matrizes de entrada, saída e incidência:

$$E_{TxP} = \begin{matrix} T1 & TAux \\ \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \end{matrix} \begin{matrix} P1 \\ P2 \\ P3 \\ PAux \end{matrix}$$

$$S_{TxP} = \begin{matrix} T1 & TAux \\ \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 3 & 0 \\ 1 & 0 \end{bmatrix} & \begin{matrix} P1 \\ P2 \\ P3 \\ PAux \end{matrix} \end{matrix}$$

$$I_{TxP} = \begin{matrix} T1 & TAux \\ \begin{bmatrix} -1 & 0 \\ -1 & 1 \\ 3 & 0 \\ 1 & -1 \end{bmatrix} & \begin{matrix} P1 \\ P2 \\ P3 \\ PAux \end{matrix} \end{matrix}$$

Observa-se que, no caso da rede impura, o lugar *P2* e a transição *T1* apresentam valor zero, o que dá a falsa impressão de que não há conexão entre *P2* e *T1*, o que não ocorre no caso da rede pura. Por isso, as redes impuras não mostram por completo a estrutura da rede. Para a análise dos estados alcançáveis da rede pura refinada pela equação fundamental das redes de Petri, não podem existir fichas nos lugares auxiliares criados, sendo necessário o disparo das transições auxiliares correspondentes para que se chegue a um estado alcançável válido.

3.3.2 Propriedades das Redes de Petri

As redes de Petri possuem propriedades que refletem como será o seu funcionamento. Pela análise destas, é possível verificar algumas características desejáveis e indesejáveis do comportamento dos modelos e, assim, eliminar as propriedades indesejáveis Normalmente, para que um modelo esteja correto, a Rede de Petri deve apresentar um conjunto de propriedades, chamadas boas propriedades, tais como: limitabilidade, reiniciabilidade e vivacidade.

É importante considerar que determinadas propriedades podem ser estudadas independentemente da marcação inicial. Estas propriedades, associadas às redes de Petri não marcadas, são conhecidas como propriedades estruturais e podem ser analisadas a partir do estudo da matriz de incidência. Enquanto que as propriedades associadas às redes de Petri marcadas em função da utilização da marcação inicial são conhecidas como propriedades comportamentais ou dinâmicas.

A seguir são apresentadas as principais propriedades comportamentais das redes de Petri (MURATA, 1989):

- Alcançabilidade – a marcação inicial de uma rede é dada por M_0. Diz-se que uma marcação M_k é alcançável a partir de M_0 se existe uma sequência de disparos que transformam M_0 em M_k. A alcançabilidade da rede é o conjunto de todas as marcações alcançáveis da rede.

- Limitabilidade – uma rede é *k*-limitada ou apenas limitada se o número de marcas em cada lugar não excede k inteiro, considerando todos os estados alcançáveis a partir de M_0. Uma rede é dita segura se for 1-limitada.

- Vivacidade – uma rede é dita viva se, para uma marcação inicial M_0, existe uma sequência $M_1, M_2, ..., M_k$, tal que toda transição T_j pode disparar pelo menos uma vez. Mais especificamente, para uma determinada transição T_j tem-se a seguinte classificação:

    - Morta (ou L0-viva) – T_j nunca pode disparar a partir de M_0;
    - L1-viva – existe alguma sequência de disparos em que T_j dispara pelo menos uma vez a partir de M_0;
    - L2-viva – existe alguma sequência de disparos em que T_j dispara pelo menos *k* vezes a partir de M_0;
    - L3-viva – existe alguma sequência de disparos em que T_j aparece infinitas vezes na sequência;
    - Viva (ou L4-viva) – a transição T_j é L1-viva também para todos os estados alcançáveis a partir de M_0.

- Persistência – uma rede é persistente se, no caso de haver duas ou mais transições habilitadas, o disparo de uma não desabilita as outras.

- Reiniciabilidade – a rede é reiniciável caso todos os seus estados sejam reiniciáveis. Um estado é reiniciável se, partindo dele, há pelo menos uma sequência de disparos de transições que leve novamente ao estado inicial.

- Reversibilidade – uma rede é reversível se é possível retornar, a partir de uma sequência de disparos, a marcação inicial da rede, ou até mesmo a uma outra marcação acessível. O fato de a rede poder voltar a uma marcação (sem ser a marcação inicial) diferencia esta propriedade da repetividade.

- Justiça – O conceito de justiça está relacionado à quantidade de disparos entre duas transições. Diz-se que duas transições são justas quando a quantidade de disparos de uma transição está limitada pela quantidade de disparos da outra. Ou seja, uma transição não pode ficar disparando infinitamente, enquanto a outra não dispara.

A seguir são apresentadas as principais propriedades estruturais das redes de Petri (MURATA, 1989):

- Conservação – uma rede conservativa em relação a um vetor de pesos é aquela onde se pode encontrar um vetor *W* de inteiros positivos, tal que o somatório do produto dos pesos deste vetor pelas marcações dos lugares é constante para qualquer marcação alcançável. As redes parcialmente conservativas, por outro lado, são aquelas em que apenas alguns elementos da rede permanecem conservativos. Uma rede é estritamente conservativa se a soma total de fichas na rede é constante para qualquer marcação alcançável.

- Limitação Estrutural – uma rede é limitada estruturalmente se ela é limitada para qualquer marcação inicial.

- Repetividade – uma rede é repetitiva quando, a partir de uma marcação inicial e uma sequência de transições disparáveis para esta marcação, todas as transições serão disparadas indefinidamente.

- Consistência – uma rede é dita consistente se é possível retornar a marcação inicial, desde que todas as transições disparem pelo menos uma vez.

A análise das redes de Petri pode se basear tanto nas propriedades comportamentais, quanto nas propriedades estruturais. Através de cálculos realizados sobre a matriz de incidência, a análise estrutural, também conhecida como análise de invariantes, procura conjuntos de transições ou lugares que apresentem características especiais, como o funcionamento cíclico ou a conservação de fichas. A análise de invariantes de transição verifica e fornece componentes repetitivos e estacionários (ciclos) nos modelos. A análise de invariantes de lugar verifica e fornece componentes conservativos.

As árvores de alcançabilidade ou acessibilidade e cobertura realizam a análise comportamental e permitem dizer se a rede é viva, limitada, repetitiva, conservativa, reiniciável e se apresenta justiça para os disparos das transições e/ou ciclos infinitos.

Este processo de análise produz um grafo que apresenta todos os possíveis estados da rede e os disparos de transições que modificam esses estados.

Ambas as formas de análise permitem detectar a presença de bloqueios no modelo. A análise do grafo de acessibilidade mostra todos os estados da rede, inclusive eventuais bloqueios, enquanto a análise de invariantes indica bloqueios potenciais. Se um lugar da rede não pertence a um invariante de lugar, então pode haver um crescimento de fichas neste lugar, indicando um provável erro de modelagem. Outra situação que possivelmente indica um problema é quando uma transição não pertence a um invariante, o que indica que ela está envolvida em bloqueios ou ciclos infinitos.

A maioria das propriedades das redes de Petri possui uma verificação decidível[23], porém de alta complexidade. Geralmente, os problemas envolvendo árvores de alcançabilidade são resolvidos através de algoritmos combinatórios e, portanto, de ordem de grandeza exponencial. Outros problemas são redutíveis ao problema da alcançabilidade, como, por exemplo, verificar se uma rede é viva.

Em virtude do uso mais frequente das árvores de alcançabilidade e cobertura para análise das propriedades das redes de Petri, este assunto será mais bem detalhado na próxima seção.

3.3.3 Árvores de Alcançabilidade e Cobertura

A árvore de alcançabilidade ou acessibilidade é um grafo formado por nós que representam marcações da rede e arcos que representam transições (MURATA, 1989, FINKEL, 1993). Ela ilustra todos os possíveis estados que podem ser alcançados ou cobertos a partir de M_0. No grafo todos os estados e transições são representados individualmente, para o caso das redes limitadas em que o grafo é finito, permitindo verificar todas as propriedades de interesse por simples inspeção. A Figura 12 apresenta uma rede de Petri limitada e a Figura 13 mostra a sua árvore de alcançabilidade. Os nós da árvore seguem a ordem *M(P1)M(P2)M(P3)*. Por exemplo, o nó raiz, representado pela marcação *[100]*, indica que no lugar *P1* existe uma ficha e nos lugares *P2* e *P3* não há fichas.

[23] Se existir um algoritmo que receba como entrada um elemento x e retorne como saída o valor “sim”, caso x pertença ao conjunto A, ou “não”, caso contrário, então se diz que o problema de decisão para o conjunto A é decidível.

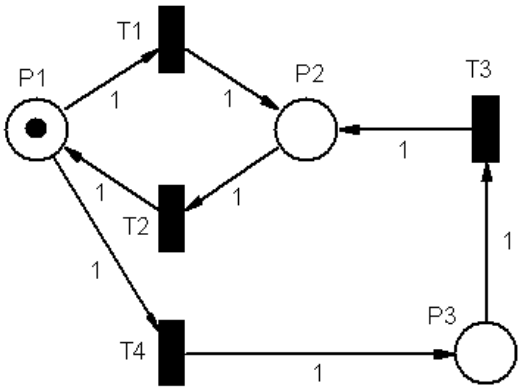

Figura 12 – Rede limitada

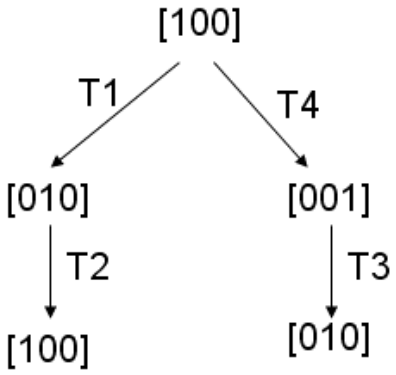

Figura 13 – Árvore de alcançabilidade

Para redes ilimitadas a árvore cresce indefinidamente. Neste caso, é mais conveniente utilizar a árvore de cobertura que é uma representação finita da árvore de alcançabilidade. Para isso, representa-se um número infinito de fichas, usando um símbolo especial *w*. Isto é o resultado da execução de múltiplos ciclos em que se verifica o acréscimo de fichas em determinados lugares da rede. Para os casos em que o número de fichas não ultrapassa um, as árvores de alcançabilidade e cobertura são idênticas. O algoritmo de Karp-Miller (FINKEL, 1993) é muito utilizado na construção da árvore de cobertura. Seu pseudocódigo é dado a seguir:

1. Colocar a marcação inicial M0 como raiz e marcá-la como nova

2. Enquanto existirem marcações novas faça:

3. Selecionar uma marcação nova M

4. Se M for igual a outra marcação presente na árvore, marcá-la como velha e vá para outra nova marcação

5. Se não há transições habilitadas em M, marcá-la com fim

6. Enquanto existirem transições habilitadas em M faça:

7. Obter a marcação M' que resulta do disparo de t em M

8. Se existe no caminho da raiz até M' uma marcação M'' tal que M'(p) ≥ M''(p) para cada lugar p e M' é diferente de M'' então substituir M'(p) por w para cada p em que M'(s) > M''(s)

9. Introduzir M' como um nó, desenhar um arco de M para M' com nome t e marcar M' como nova.

Para exemplificar a aplicação do algoritmo, a Figura 14 mostra uma rede de Petri ilimitada e a Figura 15 apresenta a árvore de cobertura correspondente.

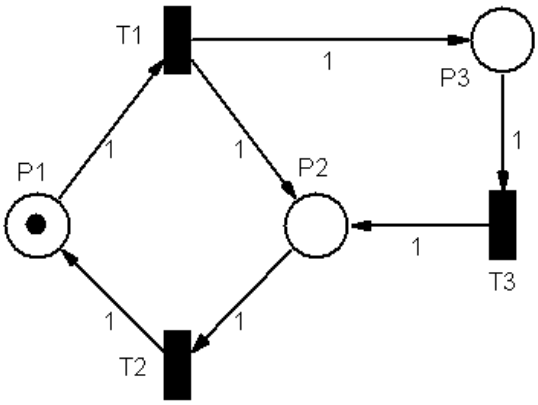

Figura 14 – Rede ilimitada

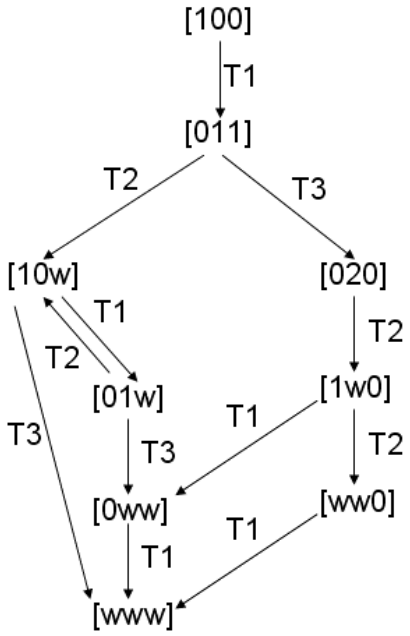

Figura 15 – Árvore de cobertura

3.3.4 Extensões às Redes de Petri

O modelo original das Redes de Petri não considera aspectos funcionais complexos, tais como, condições que determinam aspectos de temporização e fluxos de controle. Diante disso, outras propostas foram criadas, fornecendo extensões às redes de Petri. O trabalho de KUMAR & HAROUS (1990) associa tempo ao disparo da transição (Redes de Petri Temporais ou Temporizadas). O trabalho de CHIOLA (1991) associa tempo ao lugar, permitindo descrever o tempo consumido por uma atividade (Redes de Petri Temporizadas e Estocásticas). Os trabalhos de LAKOS & KEEN (1991) e LAKOS (1997) consideram marcas como objetos, introduzindo conceitos de orientação de objetos às redes (Redes de Petri Orientadas a Objeto) e propondo a linguagem para redes de Petri orientadas a objetos (*Language for Object-Oriented* Petri *Nets* - LOOPN). Outra extensão são as redes de Petri hierárquicas (FEHLING, 1993) que consideram sub-redes de lugares, transições e subsistemas, permitindo modelar sistemas mais complexos, usando estruturas que permitem abstrair detalhes de processos. Outro trabalho (JENSEN, 1987) versa sobre as redes de Petri Coloridas. Essas redes permitem que marcas individualizadas (coloridas) representem diferentes processos ou recursos em uma mesma sub-rede.

Uma extensão simples criada nas redes de Petri são os arcos inibidores. Eles invertem a lógica de um lugar de entrada, pois habilita a transição somente quando não existem fichas num lugar. Um arco inibidor tem origem num lugar Pi, destino numa transição Tj e um pequeno círculo, em vez da ponta da seta, na transição destino.

Outra extensão são as transições prioritárias nas quais transições de prioridade menor não podem ser disparadas se uma transição de prioridade maior estiver habilitada. As redes com transições prioritárias dividem as transições em grupos de prioridade, sendo que, dentro de um mesmo grupo, os disparos são não determinísticos.

TROMPEDELLER (1995) propôs, tomando por base o trabalho de BERNARDINELLO & CINDIO (1992), uma classificação das extensões das redes de Petri, levando em conta o nível de detalhes que um lugar e suas marcas podem representar. Foram apresentados três níveis de redes:

- Nível 1 – caracterizadas por lugares que podem representar valores booleanos;
- Nível 2 – caracterizadas por lugares que podem representar valores inteiros. Também são conhecidas como redes de Petri lugar/transição ou ordinárias;
- Nível 3 – caracterizadas por lugares que podem representar valores de alto nível, ou seja, multiconjuntos de valores estruturados. São também conhecidas como Redes de Petri de Alto Nível (*High-Level Petri Nets*). São exemplos desta categoria as redes de Petri orientadas a objetos, as redes de Petri temporizadas e as redes de Petri coloridas.

3.3.5 Redes de Petri de Alto Nível (RPANs)

Depois de diversas discussões sobre uma padronização para RPANs (JTC1, 2005), foi aprovado em 2002 o rascunho final do que viria a ser o padrão para redes de Petri de alto nível (ISO 15909-1 FD, 2002). Finalmente, em 2004, este padrão foi aprovado (ISO 15909-1, 2004). Há também tentativas de padronizar um formato baseado em XML, chamado PNML (*Petri Net Markup Language*), para transferência de modelos entre ferramentas diferentes (ISO 15909-2 WD, 2005, WEBER, M., 2006). Este trabalho ainda está na fase de levantamento das propostas que pretendem caracterizar essa linguagem.

O padrão aprovado para RPANs basicamente incorpora as características sintáticas e semânticas das Redes de Petri Coloridas, não considerando outras redes, também conhecidas como redes de Petri de alto nível, a exemplo das redes de Petri Temporizadas e redes de Petri Hierárquicas. Conforme apresentado na Figura 16, este trabalho está particularmente interessado nas extensões envolvendo esses três tipos de RPANs: as redes de Petri coloridas, temporizadas e hierárquicas; todas apoiadas pela ferramenta CPN *Tools* (RATZER et al., 2003).

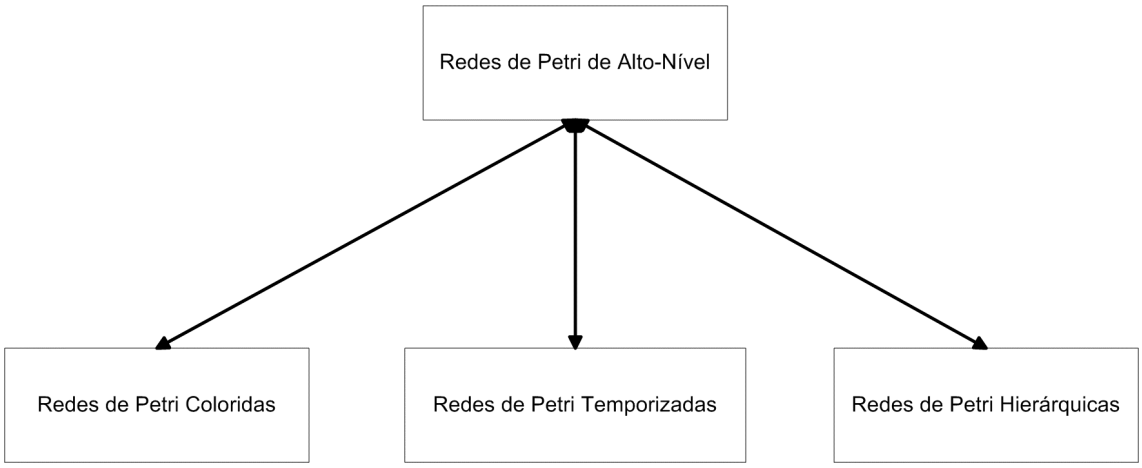

Figura 16 – Redes de Petri de alto nível

As redes de Petri Temporizadas permitem, por exemplo, estipular o tempo que uma transição leva para disparar. As redes de Petri Hierárquicas permitem subdividir uma rede em vários níveis, o que facilita o projeto das mesmas. As redes de Petri Coloridas, representante mais significativo das RPANs, são detalhadas na próxima seção.

3.3.6 Redes de Petri Coloridas (RPCs)

O principal objetivo das RPCs (GIRAULT & VALK, 2001) é a redução do tamanho do modelo. Inicialmente, as marcas das RPCs eram representadas por cores ou mesmo por padrões que possibilitam a distinção entre elas. Atualmente, são representados por estruturas de dados complexas não relacionando cores às marcas, a não ser pelo fato de que estas são distinguíveis. Deste modo, as marcas podem conter informações. Além disso, cada lugar armazena marcas de certo tipo, sendo que arcos realizam operações sobre as marcas.

Uma RPC é composta por três partes: estrutura, inscrições e declarações. A estrutura é um grafo direcionado, com dois tipos de nós (lugares e transições). As

inscrições são associadas aos lugares, transições e arcos. As declarações são tipos, funções, operações e variáveis. Quando a expressão do arco é avaliada, ela gera um multiconjunto de marcas coloridas. Os multiconjuntos são semelhantes aos conjuntos, porém permitem múltiplas ocorrências de um mesmo elemento, sendo também denominados de sacos ou coleções. Expressões podem conter variáveis, constantes, funções e operações definidas nas declarações, sendo que não produzem efeito colateral. Neste contexto, tem-se que:

- Cada lugar tem as seguintes inscrições: nome para identificação, conjunto de cores, especificando os tipos de marcas suportados no lugar e marcação inicial formada por um multiconjunto de marcas coloridas;

- Cada transição tem as seguintes inscrições: nome para identificação e expressão de guarda que possui resultado booleano e pode conter variáveis;

- Cada arco tem a seguinte inscrição: expressão do arco que pode conter variáveis.

Formalmente, uma rede de Petri colorida (JENSEN et al., 1994) é definida por $RPC = (\Sigma, P, T, A, N, C, G, E, I)$, onde:

- Σ é conjunto de tipos não vazios, também chamado de conjunto de cores;

- P é um conjunto finito de lugares;

- T é um conjunto finito de transições;

- A é um conjunto finito de arcos, tal que: $P \cap T = P \cap A = T \cap A = \emptyset$;

- N é uma função de nó. Ela é definida por $P \times T \cup T \times P$;

- C é função de cor. Ela é definida por $P \rightarrow \Sigma$;

- G é função de guarda. Ela é definida por $T \rightarrow exp$, onde exp é uma expressão tal que: $\forall\ t \in T$: [$Tipo(G(t)) = Boolean \wedge Tipo(Var(G(t))) \subseteq \Sigma$];

- E é uma função de expressões de arco. Ela é definida por $A \rightarrow exp$, onde exp é uma expressão tal que: $\forall\ a \in A$: [$Tipo\ (E(a)) = C(p) \wedge Tipo(Var(E(a))) \subseteq \Sigma$], onde p é o lugar de $N(a)$;

- I é uma função de inicialização. Ela é definida por $P \rightarrow exp$, onde exp é uma expressão tal que: $\forall\ p \in P$: [$Tipo(I(p)) = C(p)$].

As Figuras 18 a 21 mostram exemplos de RPCs antes e após o disparo da transição *t1*. Observe que os valores correntes dos dados contidos em cada lugar estão destacados por meio de um retângulo, sendo o somatório destas instâncias destacados por meio de um círculo junto ao lugar avaliado. Nas Figuras 18 e 20 o lugar *lugar1* possui 2 "*círculo*" e 1 "*quadrado*" (3 marcas divididas nas cores "*círculo*" e "*quadrado*" que são do tipo *Data=string*[24]), o lugar *lugar2* possui 1 "*círculo*" e 2 "*quadrado*", e o lugar *lugar3* não possui marcas. Nas Figuras 19 e 21, que descrevem a situação das redes após o disparo da transição *t1*, o lugar *lugar1* possui 1 "*quadrado*", o lugar *lugar2* possui 1 "*círculo*" e o lugar *lugar3* possui 2 "*círculo*" e 2 "*quadrado*".

A diferença entre as redes representadas pelas Figuras 18 a 21 é que as duas primeiras usam as próprias marcas nos arcos de entrada e saída da transição, enquanto as duas últimas usam variáveis relacionadas às marcas (variáveis *d* e *m*) e uma expressão de guarda na transição (a transição só dispara se *d* diferente de *m*, ou seja, *d<>m*).

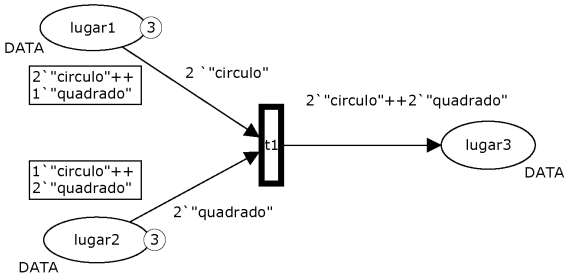

Figura 17 - RPC antes do disparo de t1

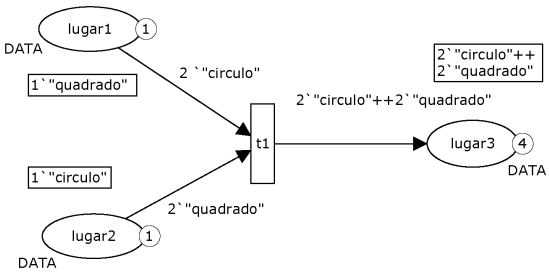

Figura 18 - RPC após disparo de t1

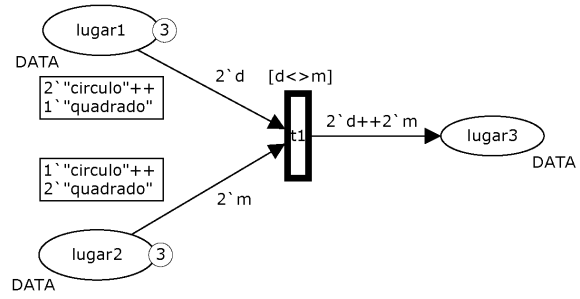

Figura 19 - RPC antes do disparo t1

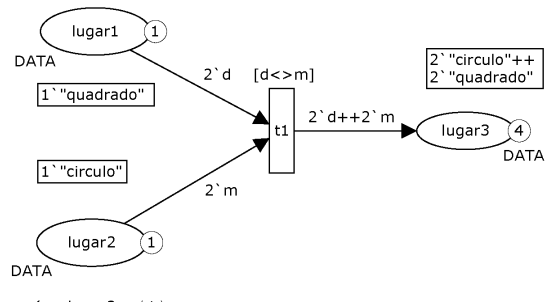

Figura 20 - RPC após disparo de t1

A Figura 21 ilustra o uso de expressões condicionais. Existem quatro lugares e uma transição. Os lugares *lugar1* e *lugar2* armazenam marcas coloridas do tipo *INTxDATA =int*string*, ou seja, cada marca é um par do tipo (*inteiro, string*). Os

[24] Cadeia de caracteres

lugares *lugar3* e *lugar4* armazenam marcas coloridas do tipo *DATAxDATA = string*string*, ou seja, cada marca é um par do tipo (*string, string*). As marcas nos lugares contendo valores complexos utilizam parênteses e somatórios de instâncias diferentes, sendo representadas por expressões do tipo $C_1`(a_{11}, a_{12}, .., a_{1n}) ++ C_2`(a_{21}, a_{22}, .., a_{2n}) ++ ..++ C_p(a_{p1}, a_{p2}, .., a_{pn})$, onde:

- C_x – número de instâncias de uma mesma marca, onde $x \in [1,p]$;
- p – quantidade de marcas distintas;
- a_{ij} – valor de um tipo específico;
- ++ – somatório de instâncias.

Por exemplo, a expressão *1`(1, "chave1") ++ 1`(2, "porta2")* representa duas instâncias distintas de marcas, contendo cada uma um par (*inteiro, string*).

As expressões dos arcos utilizam as variáveis *i* e *j* para os tipos inteiros e *d* e *m* para os tipos *string*. As expressões *(i, d)* e *(j, m)* correspondem a pares ordenados do tipo (*inteiro, string*) definidos pelos lugares do tipo *INTxDATA*. A expressão de guarda *[i=j]* da transição *t1* estipula que ela só pode ser disparada se os valores representados por *i* e *j* forem iguais. Finalmente, as expressões condicionais nos arcos de saída da transição estabelecem quais marcas serão passadas para os lugares de saída, em função das variáveis de entrada.

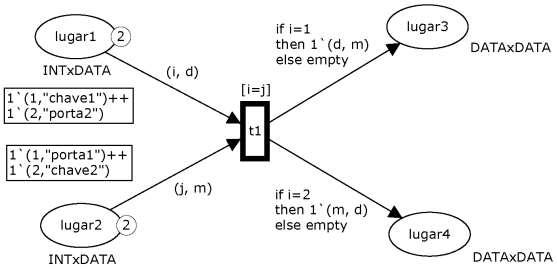

Figura 21 - RPC antes do disparo de t1

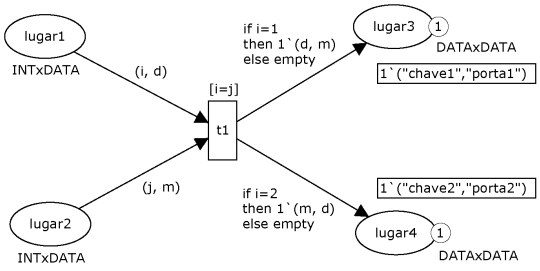

Figura 22 - RPC após disparo de t1

A Figura 22 mostra o estado final da RPC apresentada anteriormente, após disparos da transição *t1*. As marcas dos lugares de entrada da transição são transferidas para os lugares de saída desde que valor *i* da marca do lugar *lugar1* tenha o valor igual a *j* da marca do lugar *lugar2*. Além disso, as expressões condicionais alteram o formato das marcas de entrada, gerando uma nova marca composta pelas partes que contém o valor do tipo *string* dos lugares *lugar1* e *lugar2* no formato (*d*, *m*) ou (*m*, *d*).

Outros exemplos do uso de RDCs, bem como possíveis cenários de aplicação, podem ser encontrados em JENSEN et al. (1998).

3.3.7 Vantagens das Redes de Petri

Além das vantagens já citadas do uso combinado com regras ativas, a redes de Petri por si só oferecem várias vantagens em relação a outras abordagens de modelagem de processos, pois possuem representação gráfica, funcionam como linguagem de comunicação entre especialistas de diferentes áreas, são de fácil aprendizado, possibilitam descrever aspectos estáticos e dinâmicos do processo a ser representado, permitem a simulação e ainda possuem o formalismo matemático muitas vezes necessário na modelagem de processos. Podem ser destacadas outras qualidades dos modelos em redes de Petri e suas extensões, tais como:

- Permitir a detecção de ciclos infinitos na criação de fluxos de trabalho;
- Possuir semântica bem definida que denota de forma não ambígua os elementos modelados;
- Permitir representar concorrência verdadeira entre eventos, através do disparo de uma transição com múltiplos arcos de entrada. Normalmente, outras abordagens utilizam a alternância entre eventos;
- Realizar a simulação interativa e permitir a análise de comportamentos de interesse;
- Permitir descrições hierárquicas, tornando possível modelar sistemas complexos;
- Possibilitar a descrição do fluxo de controle sincronizado com a descrição da manipulação dos dados. Normalmente, outras abordagens tratam essas descrições separadamente, mas o funcionamento dos sistemas não separa esses fluxos no mundo real, sendo mais natural que a sua modelagem ocorra de forma integrada (MULYAR & VAN DER AALST & HOFSTEDE, 2005);
- Descrever e integrar processos com diferentes níveis de granularidade, simulando comportamentos complexos.

Uma solução alternativa às redes de Petri vem da área de fluxos de trabalho (*workflows*) (RUSSELL et al., 2004, 2006, VAN DER AALST et al., 2003). Segundo a

WfMC (*Workflow Management Coalition*) [25], os fluxos de trabalho permitem a automação do processo de negócio, na sua totalidade ou em partes, onde documentos, informações ou tarefas são passadas de um participante para o outro para execução de uma ação, de acordo com um conjunto de regras de procedimentos. Um dos problemas dessa abordagem é a falta de uma definição formal. Atualmente, não existe um consenso sobre o que constitui uma especificação de workflow (VAN DER AALST & HOFSTEDE, 2005, VAN DER STRAETEN et al., 2007). Inclusive, algumas abordagens propõem que a modelagem de fluxos de trabalho sejam baseadas em redes de Petri (VAN DER AALST, 1998, EHRIG et al., 2004).

3.3.8 Desvantagens das Redes de Petri

Apesar do grande número de vantagens, as redes de Petri e suas extensões também possuem desvantagens.

Um dos problemas relacionados ao uso das redes de Petri e suas extensões é que o usuário precisa dominar os seus conceitos e propriedades, o que pode demandar algum tempo. Assim, apesar de apresentar uma interface gráfica que facilita a modelagem e apresenta um comportamento bem definido, isto pode não ser suficiente ou apropriado em determinadas situações. Dessa forma, como ocorre nas linguagens de programação, pode ser necessária a criação de interfaces mais intuitivas para os usuários finais dos processos modelados de modo a facilitar o seu uso. Dessa forma, de modo a facilitar o uso dos processos modelados em redes de Petri, este trabalho propõe o emprego de uma álgebra implementada no SGBD SECONDO. Esta álgebra associa os processos aos operadores de eliminação de dados.

Outro problema é que, atualmente, no que se refere ao processamento de grandes volumes de dados, o desempenho proporcionado pelas ferramentas e linguagens que modelam e processam as redes de Petri não pode ser comparado ao de outras ferramentas e linguagens desenvolvidas especificamente para o eficiente processamento de dados.

Neste trabalho, o tempo de processamento não foi considerado o fator mais relevante, mas sim os conceitos relativos aos modelos dos processos de descarte de dados. Dessa forma, optou-se por utilizar redes de Petri de alto nível para executar esses

[25] http://www.wfmc.org/

processos. A ideia é só enviar para as redes os dados e referências estritamente necessários ao processamento, buscando-se minimizar os problemas relacionados ao desempenho.

3.4 Ferramentas de Apoio às Redes de Petri

Existem várias iniciativas que apoiam o uso de redes de Petri, seja por parte da comunidade científica ou das empresas. A seguir será apresentada uma visão geral de algumas ferramentas livres que permitem a criação, edição, análise e simulação dessas redes.

3.4.1 Visual Object Net++

A ferramenta Visual Object Net++[26] é mantida pelo Departamento de Controle Automático da Universidade de Tecnologia de Illemnau, Alemanha. Ela permite a construção de redes de Petri lugar/transição hierárquicas com transições discretas e contínuas. Ela suporta hierarquia de objetos e possui uma interface gráfica para edição e simulação das redes. Nas transições contínuas, o disparo ocorre na forma de um fluxo contínuo controlada na forma de uma constante ou função inserida pelo usuário. Pode ser utilizada para modelagem e visualização de sistemas dinâmicos, no entanto oferece poucas opções de análise, entre elas, a verificação de existência de transições conflitantes (*deadlocks*). Ela executa sobre o ambiente MS Windows. A Figura 23 seguir apresenta a interface dessa ferramenta.

[26] www.techfak.uni-bielefeld.de/~mchen/BioPNML/Intro/VON.html

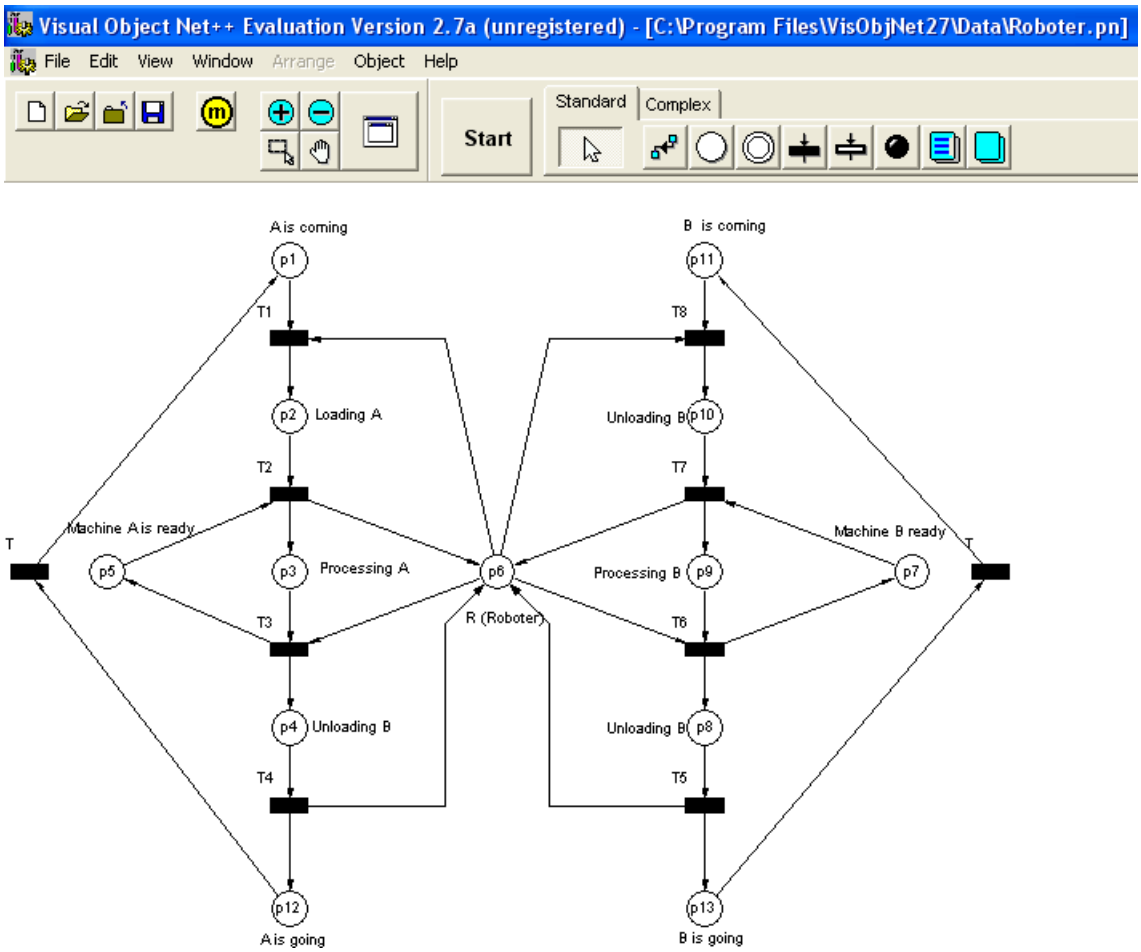

Figura 23 - Interface da ferramenta Visual Object Net++ (Fonte: exemplo disponibilizado na interface de ajuda da ferramenta)

3.4.2 Renew

Reference Net Workshop (Renew)[27] é mantida pelo Grupo de Informática da Universidade de Hamburgo, Alemanha. É uma ferramenta para edição, simulação e análise de RPCs orientadas a objetos, hierárquicas, temporizadas ou não. Entretanto, as ferramentas de análise ainda são rudimentares, sendo necessário usar de simulações para explorar as propriedades da rede. É uma ferramenta desenvolvida em Java, o que a torna portável para vários sistemas. Segundo KUMMER & MOLDT. (2004), a ferramenta Renew estende as RPCs, especificamente através da adição de novos tipos de arco, uma versão de alto nível da fusão de transições, trazendo a essas redes o formalismo das redes de Petri objeto. Sua interface é mostrada na Figura 24.

[27] http://www.renew.de

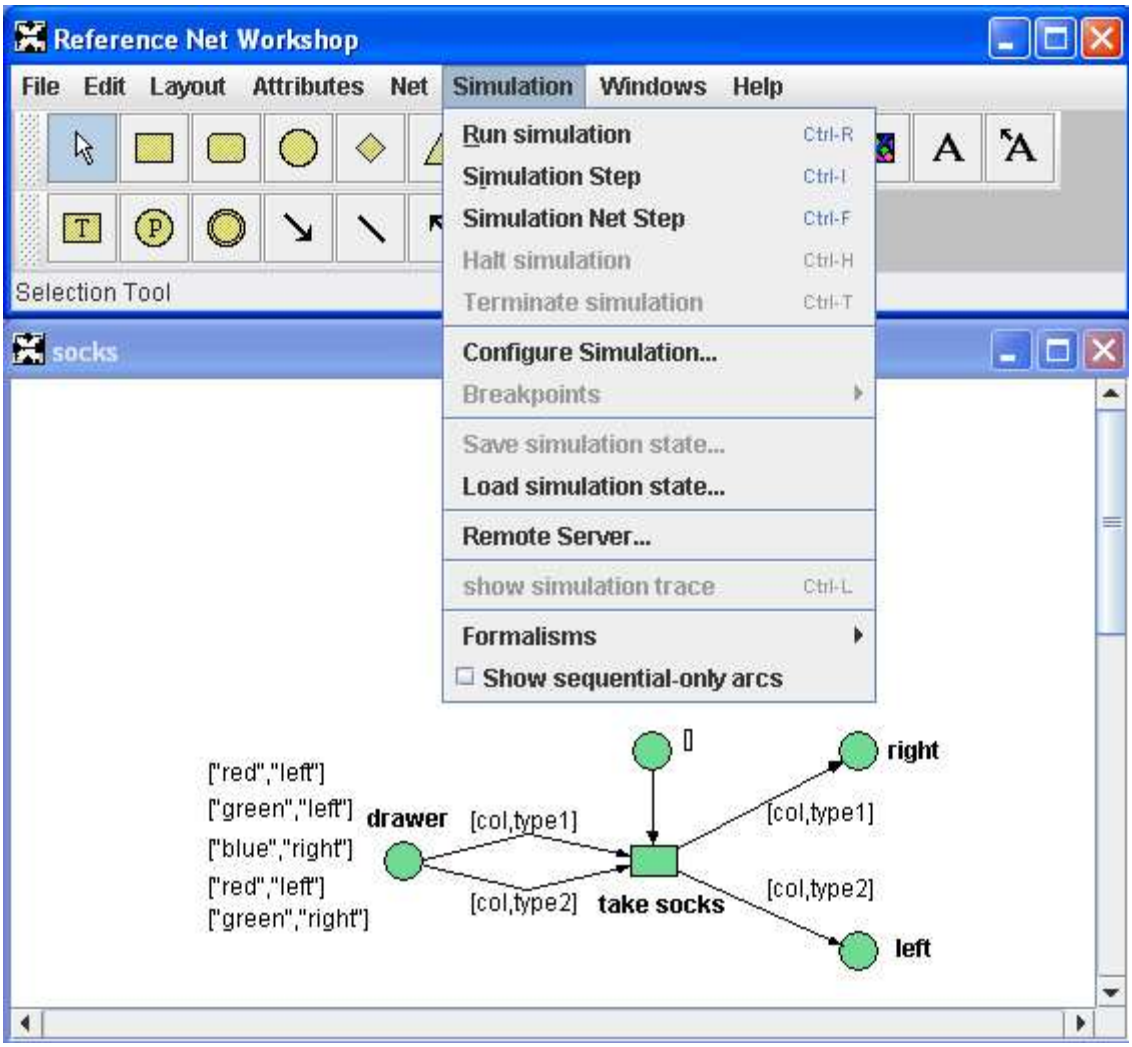

Figura 24 - Interface da ferramenta Renew (Fonte: exemplo disponibilizado na interface de ajuda da ferramenta)

3.4.3 CPN *Tools*

Coloured Petri *Nets Tools* (CPN *Tools*) [28] é uma ferramenta mantida pelo Grupo de Redes de Petri Coloridas da Universidade de Aarhus, Dinamarca. Ela permite a edição, simulação e análise de RPCs, hierárquicas, temporizadas ou não (RATZER et al., 2003). Dessa forma, o nome dessa ferramenta se mostra inapropriado, uma vez que ela suporta uma combinação de extensões da RPANs. CPN *Tools* foi criada para substituir a ferramenta Design/CPN[29]. Ela fornece um retorno (*feedback*) que facilita mensagens de erro contextuais e indica relações de dependência entre os elementos da rede. Além disso, fornece ferramentas de análise e simulação de redes temporizadas ou

[28] http://wiki.daimi.au.dk/cpntools/_home.wiki
[29] http://www.daimi.au.dk/designCPN/

não e uma interface interativa com técnicas avançadas, baseadas em menus circulares e contextualizados que aparecem quando se mantém o botão direito do mouse pressionado.

As inscrições e declarações dos lugares, arcos e transições dessa ferramenta são feitas utilizando a linguagem CPN ML[30], uma implementação da linguagem ML (HARPER, 2005). Ela permite a extensão de tipos e um armazenamento automático eficiente para estrutura de dados e funções, além de facilitar a programação com estrutura de dados recursiva e simbólica. A linguagem ML pode ser considerada um misto de linguagem funcional e imperativa. As linguagens puramente funcionais buscam a avaliação de funções matemáticas e evitam estados ou dados mutáveis. As linguagens imperativas se baseiam na execução de comandos e nas mudanças de estado do programa e dos dados em memória. Em uma linguagem funcional não existe explicitamente a declaração de variáveis ou alocação de memória, sendo estas operações executadas automaticamente quando as funções são chamadas e os espaços alocados são liberados, logo após a função receber o valor de retorno. Esta característica é conhecida como transparência referencial. Isso garante que o resultado da função será o mesmo para um determinado conjunto de parâmetros independente de quando a função seja avaliada, o que facilita a tarefa de verificar a correção do programa. A ML permite aproveitar as características dos dois paradigmas, encorajando a programação funcional, mas permitindo a programação imperativa onde for necessário.

A interface dessa ferramenta oferece diferentes contêineres na forma de paletas. Basta clicar sobre os elementos da rede que está sendo modelada para que as funções associadas a este elemento se tornem disponíveis. Ela executa sobre o ambiente Windows 2000, XP e Linux, além de apresentar uma documentação bastante completa. A Figura 25 apresenta a interface dessa ferramenta.

[30] http://wiki.daimi.au.dk/cpntools-help/cpn_ml.wiki

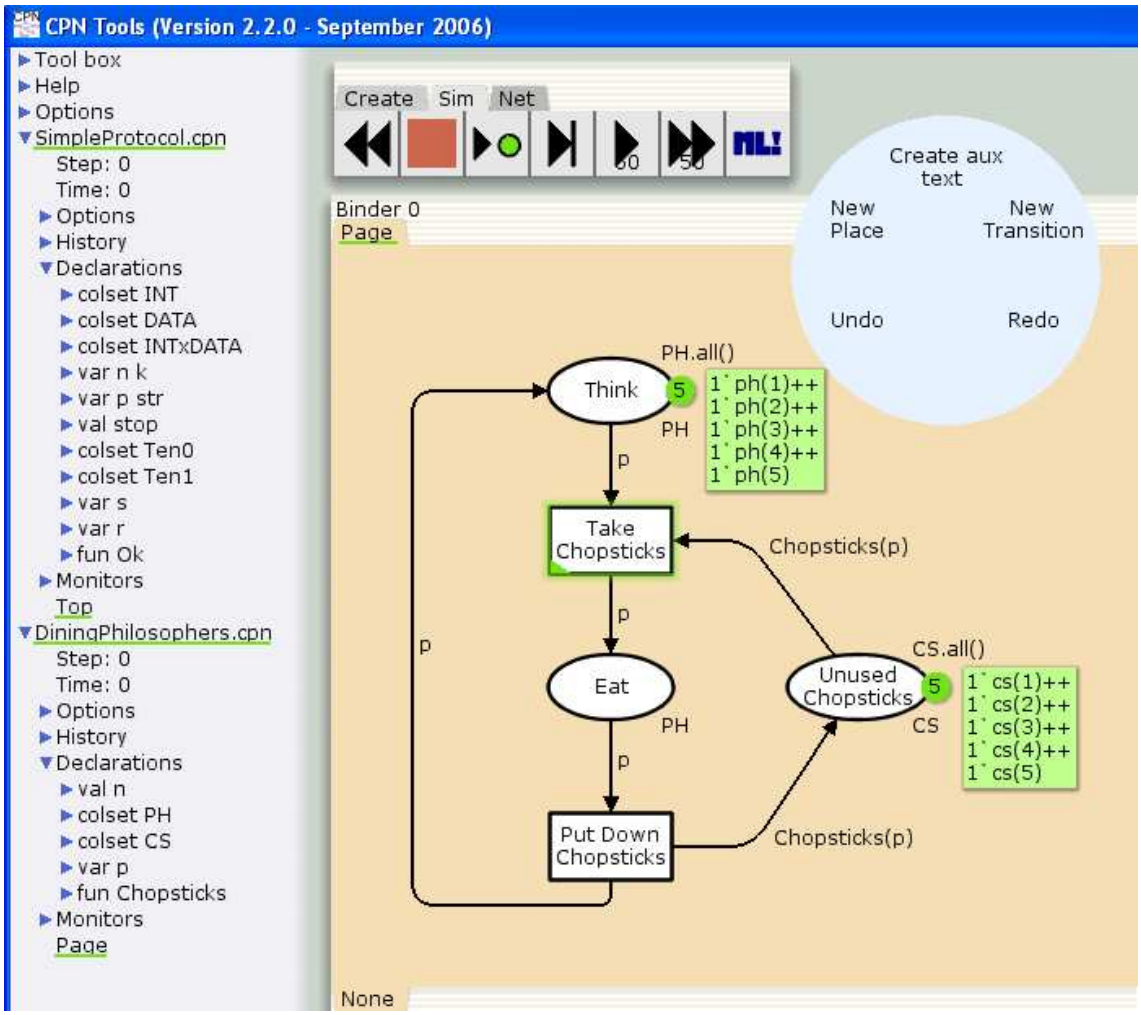

Figura 25 - Interface da ferramenta CPN *Tools* (Fonte: exemplo disponibilizado na interface de ajuda da ferramenta)

Outra característica dessa ferramenta é que ela fornece uma interface alternativa para experimentos chamada Britney *Suite* (WESTERGAARD, 2006). Esta interface fornece acesso à estrutura de dados e ao simulador da CPN *Tools*. Ela pode ser personalizada, diferentemente da interface principal da ferramenta. Ela permite personalizar simulações e visualizações de redes, além da possibilitar a criação de extensões e disponibiliza sua interface por meio de um applet que pode ser acessado via Web. Uma outra caracterísitca interessante dessa interface é que ela permite a programação de disparos infinitos das transições de uma rede por meio de seu arquivo de configuração, característica não disponível nas ferramentas estudadas neste trabalho e na própria interface principal da ferramenta CPN *Tools*, que permite até 999.999.999 disparos. A arquitetura da ferramenta CPN *Tools*, incluindo a plataforma alternativa Britney *Suíte* que é apresentada na Figura 26.

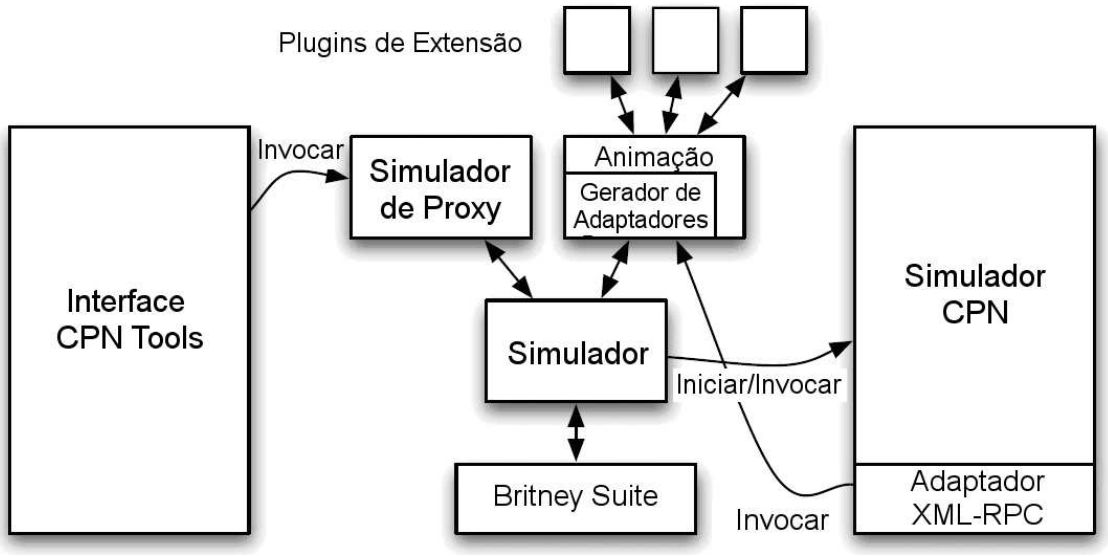

Figura 26 - Arquitetura da ferramenta CPN *Tools* (Fonte: WESTERGAARD (2006))

3.4.4 Comparação das Ferramentas

A Tabela 4 apresenta o resultado do estudo comparativo das principais características das ferramentas mostradas anteriormente. Observa-se que, apesar de não oferecer suporte a objetos, a ferramenta CPN *Tools* trabalha com os principais tipos de RPANs, incluindo as RPCs. De modo geral, ela se destaca das demais, à medida que permite um número maior de opções para análise das redes de Petri, fornece um maior suporte documental, trabalha com os três tipos de rede de Petri requeridos neste trabalho (as coloridas, temporizadas e hierárquicas) e permite personalizações, através da interface alternativa Britney *Suite*. Esta interface permite inclusive que a rede seja executada em segundo plano, disparando as transições automaticamente, e por um número desejado de vezes (inclusive infinitos disparos), atuando como um servidor para clientes na *Web*.

Como visto, a ferramenta CPN *Tools* permite a modelagem de RPAN e apresenta vantagens em relação às duas outras ferramentas também estudadas: a Visual Object Net++ e Renew. Em virtude disso, ela foi escolhida para a modelagem das RPANs utilizadas neste trabalho.

Tabela 4: Comparação das ferramentas

	Visual Object Net++	***Renew***	**CPN *Tools***
Tipos de Redes Suportadas	-Redes Lugar/Transição -Redes Temporais -Redes Híbridas -Redes Hierárquicas	-Redes Orientadas a Objeto -Redes Lugar/Transição -Redes Temporais -Redes Hierárquicas	-RPCs -Redes Temporais -Redes Hierárquicas
Ambiente	-MS Windows	Java	Linux, Win 2000 e XP
Componentes e Características	-Editor Gráfico -Simulação Rápida -Análise Estrutural -Análise de Desempenho Simplificada -Suporte a Hierarquia de Objetos	-Editor Gráfico -Simulação Rápida -Análise Estrutural -Análise da Sintaxe -Suporte a Hierarquia de Objetos	-Editor Gráfico -Simulação Rápida -Análise Estrutural -Análise de Desempenho -Análise da Sintaxe -Análise e Construção de Espaço de Estados[31] -Permite customizações e disparos infinitos das transições, através da interface alternativa Britney *Suite*
Principais Limitações	-Não oferece suporte a RPCs -Poucas Ferramentas de Análise -Parca Documentação -Não permite customizações	-Poucas Ferramentas de Análises -Parca Documentação -Não permite customizações	Não oferece suporte a Objetos

O próximo capítulo apresenta os conceitos relacionados à eliminação de dados. Estes conceitos serão utilizados, juntamente com os apresentados até agora, na elaboração do arcabouço proposto neste trabalho.

[31] A utilização espaço de estados permite reduzir o número de atributos (chamados de variáveis de estado) e manter o comportamento e as relações de interesse.

CAPÍTULO 4 MODELAGEM DOS CONCEITOS RELACIONADOS À ELIMINAÇÃO DE DADOS

O descarte de dados é apenas um dos processos que pode ser utilizado para reduzir os efeitos dos problemas relacionados à explosão, poluição e sobrecarga de dados. É importante ter-se uma visão geral dos outros processos que envolvem não somente o descarte, como também a gestão de dados, ou seja, é necessário analisar as diferentes situações e contextos que envolvem o ciclo de vida dos dados.

Com o objetivo de analisar a relação entre dados, eventos e gestores de dados no contexto da linha de pesquisa em banco de dados do PESC/UFRJ[32], foi proposta a modelagem desses elementos e suas interconexões relevantes através de diagramas de classes da UML [33], bem como a definição de todas as classes modeladas. Esta modelagem, além de prover uma visão mais ampla dos conceitos relacionados ao descarte de dados, permite que futuros trabalhos possam ser mais facilmente integrados aos elementos do arcabouço discutido nesta tese. Os diagramas, propostos pelo autor desta tese e apresentados na Figura 27, bem como as definições das classes modeladas, foram validados pelo grupo de pesquisa que compõe o Laboratório de Tratamento da Sobrecarga da Informação [34] (TRASGO) através de várias reuniões onde foram discutidos os conceitos relevantes a serem modelados e os seus relacionamentos. Obviamente, esta modelagem não descreve completamente a vasta relação de elementos tratados pelo TRASGO, entretanto, dá uma visão geral dos conceitos mais relevantes atualmente e seus relacionamentos, permitindo uma melhor compreensão dos problemas abordados. São esperadas futuras modificações e extensões que aprimorem esta proposta, de forma a torná-la mais completa e representante mais próxima dos temas abordados, ajudando a orientar os futuros trabalhos a serem realizados no laboratório.

[32] Programa de Engenharia de Sistemas e Computação da Universidade Federal do Rio de Janeiro.
[33] http://www.uml.org/
[34] O TRASGO – Laboratório de Tratamento da Sobrecarga da Informação é um projeto que está em fase de concepção, coordenado pelo Professor Geraldo Xexéo. Esta iniciativa busca caminhos de integração para as diversas pesquisas conduzidas na linha de BD, relacionadas à sobrecarga, avaliação de qualidade e busca e recuperação de informação. Atualmente, cerca de 20 pessoas, entre alunos de mestrado, doutorado e professores participantes, fazem parte deste projeto.

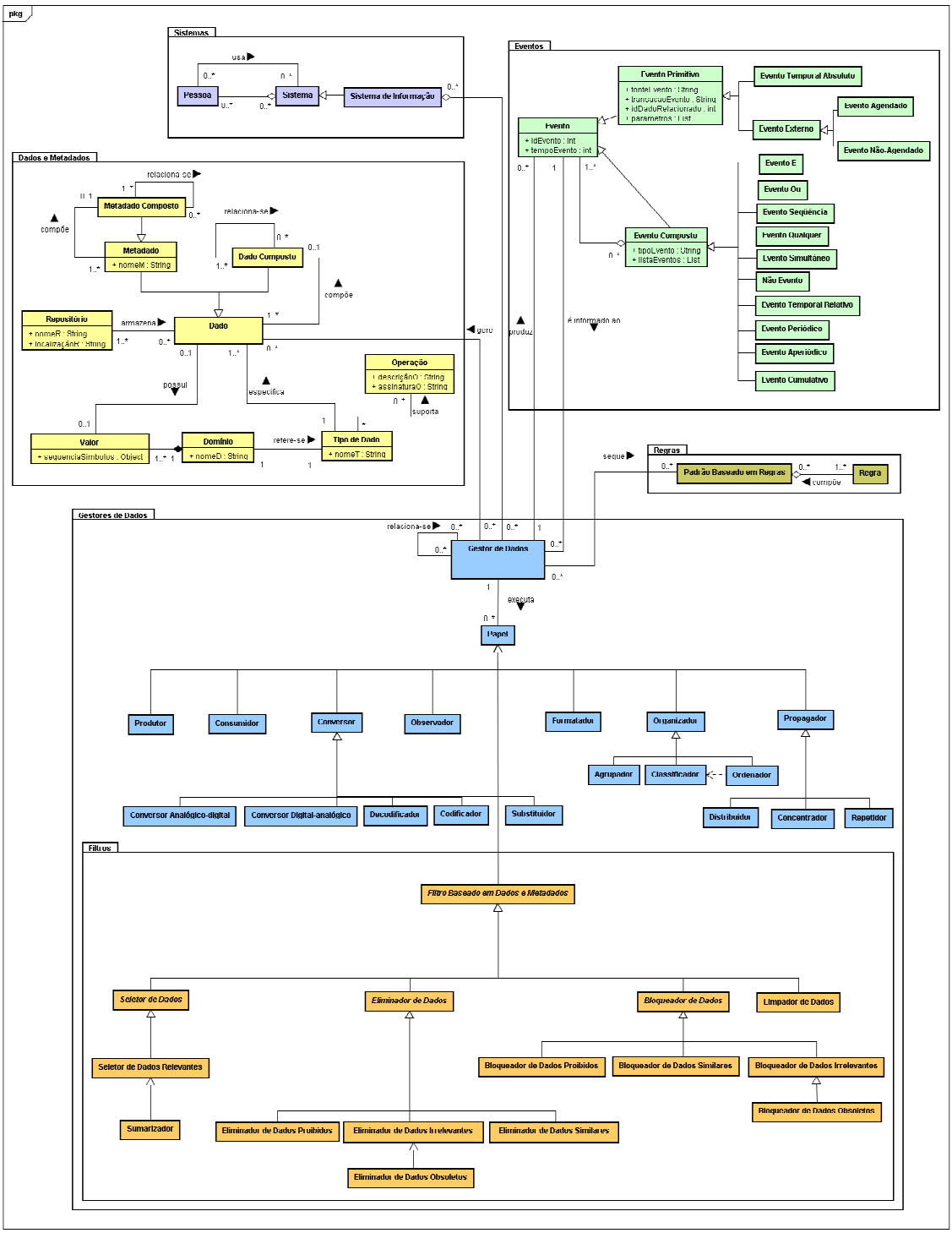

Figura 27 – Diagramas dos Conceitos de Interesse do TRASGO

Os diagramas e definições apresentados neste capítulo são utilizados como referência conceitual para os tópicos relacionados à sobrecarga de informação tratados no TRASGO. Dessa forma, ele representa é uma visão mais abrangente e de mais alto nível dos assuntos abordados no decorrer desta tese.

Com o intuito de facilitar a descrição e a leitura do diagrama apresentado na Figura 27, as classes deste diagrama serão novamente apresentadas, porém, desta vez, elas estarão divididas em diferentes pacotes, que agrupam classes que tratam de tópicos em comum, e acompanhadas das suas respectivas definições. O diagrama de pacotes da Figura 28 fornece a visão de alto nível destes tópicos, sendo que o detalhamento desses pacotes será feito nas próximas subseções.

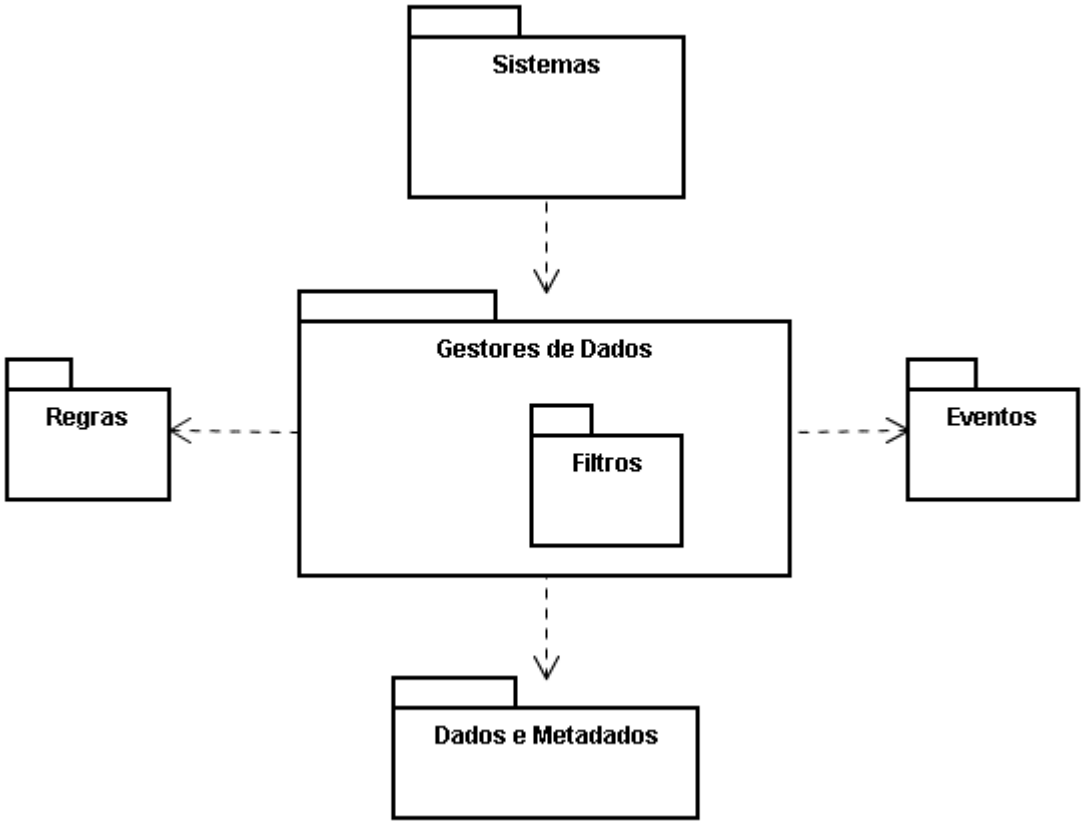

Figura 28 – Relacionamento entre Pacotes

4.1 Pacote Sistemas

O Pacote Sistemas permite modelar conceitos de alto nível, baseando-se em ideias provenientes da Teoria Geral de Sistemas (BERTALANFFY, 1977). Para isso, são propostas as classes: Sistema, Sistema de Informação e Pessoa. Estes conceitos mais gerais fornecem uma contextualização de mais alto nível dos problemas tratados pelos

projetos desenvolvidos no TRASGO. A modelagem dessas classes é apresentada na Figura 29.

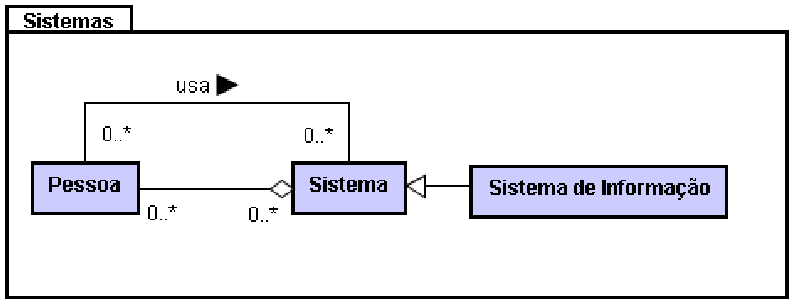

Figura 29 – Pacote Sistemas

As definições adotadas para as classes que fazem parte do Pacote Sistemas são as seguintes:

- Sistema – um conjunto de elementos dinamicamente relacionados formando uma atividade para atingir um objetivo, operando sobre dados/energia/matéria para fornecer informação/energia/matéria (BERTALANFFY, 1977);

- Sistema de Informação – é um sistema que troca informação com seu exterior e internamente é constituído exclusivamente de mecanismos capazes de gerir informação;

- Pessoa – representa pessoas que participam ou usam sistemas, ou seja, pessoas podem ser usuários ou componentes dos sistemas.

Obviamente, essas classes não modelam todos os componentes da classe Sistema, nem é este o objetivo dessa modelagem. As classes do Pacote Sistemas servem apenas como ponto de partida para as classes modeladas no Pacote Gestores de Dados que são de interesse do grupo de pesquisa do TRASGO. Este diagrama é extensível, portanto, ao longo do tempo, novas classes podem ser adicionadas de forma a torná-lo mais completo.

4.2 Pacote Gestores de Dados

As classes do Pacote Gestores de Dados, apresentadas na Figura 30, permitem organizar e agrupar diversos dos trabalhos desenvolvidos atualmente na linha de Banco de Dados da UFRJ, existentes no TRASGO.

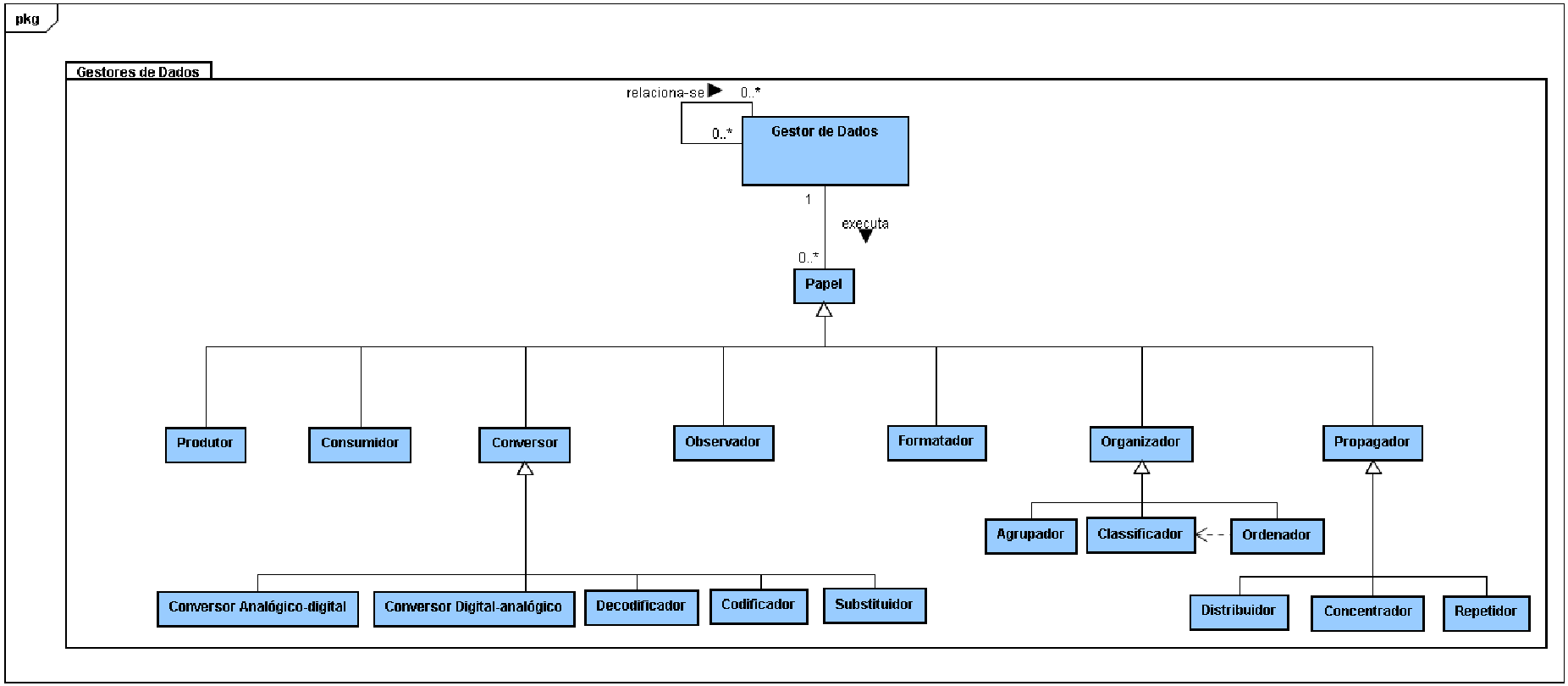

Figura 30 – Pacote Gestores de Dados

A classe Gestor de Dados e suas especializações apontam conceitos que influem no ciclo de vida dos dados, partindo da sua produção (Classe Produtor), passando pela sua filtragem (Pacote Filtros que é parte integrante do Pacote Gestores de Dados) e chegando até o consumo do dado (Classe Consumidor). Estas classes podem produzir e receber informações sobre eventos modelados no Pacote Eventos. Adicionalmente, estas classes podem ter seu comportamento orientado segundo certos critérios (regras), através de padrões baseados em regras. Finalmente, a matéria-prima e produto final dos gestores de dados são modelados no Pacote Dados e Metadados.

As seguintes definições foram adotadas para as classes do Pacote Gestor de Dados:

- Gestor de Dados – elemento que gere os dados, usando diferentes processos;
- Produtor – gera dados de forma contínua ou discreta;
- Consumidor – utiliza dados para uso próprio (interno) e não os repassa a outras entidades;
- Conversor – escreve dados de saída em função dos dados de entrada, após alguma transformação. Possui as seguintes especializações:
    - Conversor Analógico-digital – converte dados analógicos para o formato digital;
    - Conversor Digital-analógico – converte dados discretos para o formato analógico;
    - Codificador – transforma dados de acordo com um determinado código;
    - Decodificador – transforma dados codificados de forma o obter o seu formato original;
    - Substituidor – substitui dados por outros dados de acordo com algum critério;
- Formatador – modifica o formato de apresentação do dado;
- Observador – utiliza dados para uso próprio (interno), podendo repassá-los a outras entidades sem modificá-los;

- Organizador – organiza os dados segundo certos critérios. Possui as seguintes especializações:
  - Agrupador – sem necessariamente ter um conjunto de categorias pré-definidas, agrupa dados de acordo com o grau de similaridade entre eles, sendo que diferentes critérios de similaridade podem ser usados;
  - Classificador – dispõe os dados em categorias pré-definidas;
  - Ordenador – ordena os dados pertencentes a uma mesma categoria;
- Propagador – dissemina, espalha e transmite dados. Possui as seguintes especializações:
  - Distribuidor – realiza a distribuição de dados de uma fonte, direcionando os dados para os seus destinatários;
  - Concentrador – recebe dados transmitidos de diferentes fontes e/ou tempos, os reagrupa e os retransmite;
  - Repetidor – recebe os dados do remetente e os envia ao destinatário.

4.3 Pacote Filtros

O Pacote Filtros, Figura 31, contém as classes de maior interesse deste trabalho, pois incluem as classes relacionadas aos padrões de eliminação de dados. Ele permite modelar a geração de dados de saída a partir de dados e metadados de entrada. As classes deste pacote são subclasses da Classe Papel do Pacote Gestores de Dados.

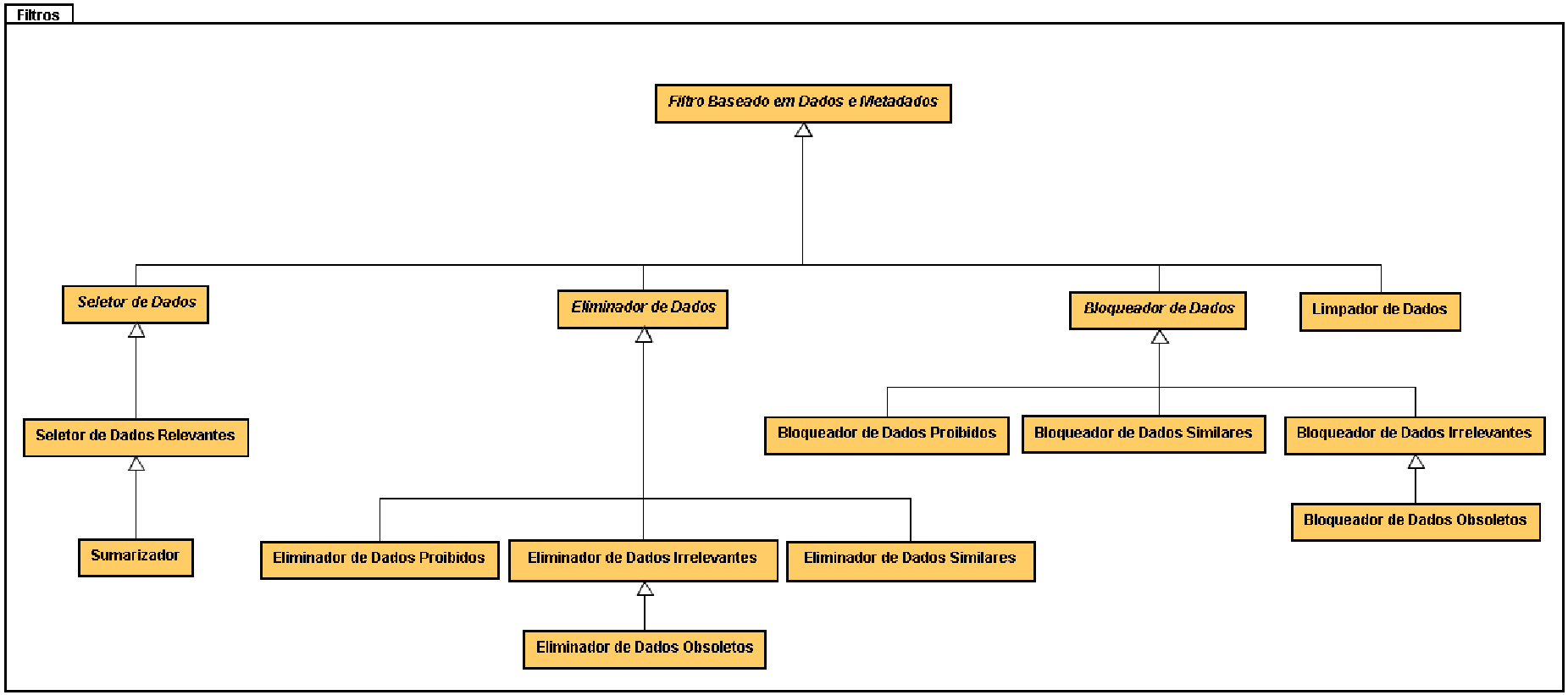

Figura 31 – Pacote Filtros

As seguintes definições foram adotadas para as classes pertencentes a este pacote:

- Filtro Baseado em Dados e Metadados – gera dados de saída baseados em dados e metadados de entrada, retendo o essencial ou o desejado de acordo com determinadas regras e/ou critérios. Possui as seguintes especializações:
  - Seletor de Dados – permite a entrada em repositórios de dados com determinadas características. Possui as seguintes especializações:
    - Seletor de Dados Relevantes (Permitidos) – permite a passagem de dados identificados como relevantes, tomando por base os seus valores e/ou metadados comparados a parâmetros que caracterizem dados relevantes. Possui a seguinte especialização:
      - Sumarizador – realiza a redução do tamanho do dado, enfocando seus pontos principais de acordo com critérios previamente definidos;
  - Eliminador de Dados – elimina dados com determinadas características contidos em repositórios. Possui as seguintes especializações:
    - Eliminador de Dados Proibidos – elimina dados identificados como proibidos, tomando por base os seus valores e/ou metadados comparados a parâmetros que caracterizem dados proibidos. Estes dados, caso sejam mantidos nos repositórios internos, podem causar algum tipo de prejuízo aos usuários dos mesmos;
    - Eliminador de Dados Similares – elimina dados identificados como similares, tomando por base os dados a serem comparados e parâmetros que identificam as funções de similaridade a serem utilizadas e os limites definidos para identificar a similaridade entre dados. Diferentes critérios podem ser usados para eliminar os dados similares, tais como: tamanho, dado mais recente, etc;
    - Eliminador de Dados Irrelevantes – elimina dados identificados como irrelevantes, tomando por base os seus valores e/ou metadados comparados a parâmetros que caracterizem dados

irrelevantes. Estes dados podem ter sido considerados relevantes em outro momento ou situação. Possui a seguinte especialização:

- Eliminador de Dados Obsoletos – elimina dados identificados como obsoletos, tomando por base metadados relativos à data de criação, atualização e/ou leitura comparados a parâmetros que caracterizem o tempo de validade dos dados;

- Bloqueador de Dados – bloqueia a entrada em repositórios de dados com determinadas características. Possui as seguintes especializações:
  - Bloqueador de Dados Proibidos – não permite a passagem de dados identificados como proibidos, tomando por base os seus valores e/ou metadados comparados a parâmetros que caracterizem dados proibidos. Estes dados, caso sejam incorporados aos repositórios internos, podem causar algum tipo de prejuízo aos usuários dos mesmos;
  - Bloqueador de Dados Similares – não permite a passagem de dados identificados como similares, tomando por base os dados a serem comparados com os dados do repositório interno e parâmetros que identificam as funções de similaridade a serem utilizadas e os limites definidos para identificar a similaridade entre dados. Diferentes critérios podem ser usados para eliminar os dados similares existentes nos repositórios externos, tais como: tamanho, dado mais recente, etc;
  - Bloqueador de Dados Irrelevantes – não permite a passagem de dados identificados como irrelevantes, tomando por base os seus valores e/ou metadados comparados a parâmetros que caracterizem dados irrelevantes. Estes dados podem ser relevantes em outros contextos. Possui a seguinte especialização:
    - Bloqueador de Dados Obsoletos – não permite a passagem de dados identificados como obsoletos, tomando por base metadados relativos à data de criação,

atualização e/ou leitura comparados a parâmetros que caracterizem o tempo de validade dos dados;

- Limpador de dados – procura detectar e remover erros e inconsistências de dados de maneira a melhorar a sua qualidade.

Os padrões de eliminação de dados associados as classes deste pacote serão estudas em maior profundidade nos próximos capítulos.

4.4 Pacote Regras

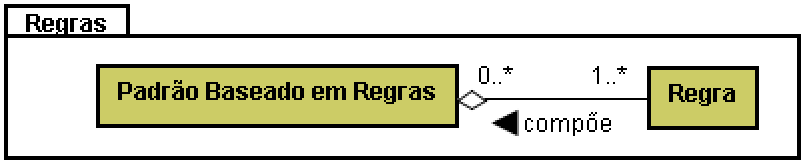

Figura 32 – Pacote Regras

Este trabalho considera que os gestores de dados podem seguir padrões baseados em regras de acordo com o apresentado na Figura 32. Portanto, este pacote pemite representar regras que guiam o comportamento dos conceitos modelados pelo Pacote Gestores de Dados. As seguintes definições são adotadas neste pacote:

- Padrão Baseado em Regras – define padrões de comportamento baseado em regras;
- Regra – proposição condicional com um ou mais termos.

4.5 Pacote Dados e Metadados

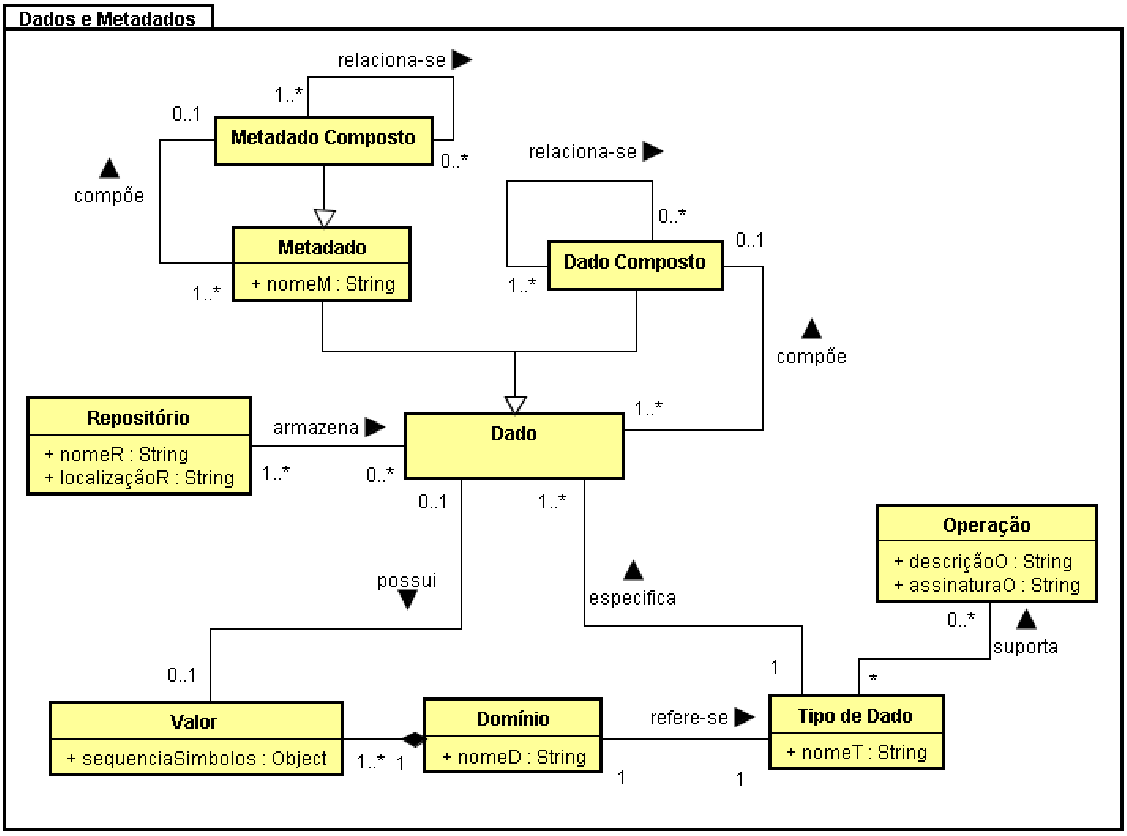

Figura 33 – Pacote Dados e Metadados

O Pacote Dados e Metadados, Figura 33, contém as classes que são efetivamente tratadas pelos gestores de dados. A representação de dados e metadados utilizados neste diagrama permite construir objetos complexos a partir de objetos simples, além de tratar esses objetos de maneira uniforme. Isto flexibiliza o uso desses elementos, permitindo o seu emprego em diferentes cenários. Os metadados armazenados podem fornecer características importantes que descrevem os dados, tais como: contexto, frequência de acesso, data de atualização e criação, entre outros. Vale ressaltar que, apesar de genericamente serem considerados dados, os metadados aparecem na hieraquia de objetos separados dos nós que descrevem os dados compostos. Isto mantém a separação entre os dados (compostos) e metadados, estando, neste sentido, de acordo com as ideias adotadas pelo arcabouço MOF[35] (*Meta Object Facility*) e suas aplicações, a exemplo do CWM (*Common Warehouse Metamodel*)[36] (OMG, 2003).

[35] http://www.omg.org/mof/
[36] http://www.omg.org/technology/documents/modeling_spec_catalog.htm

As seguintes definições foram adotadas para as classes do Pacote Dados e Metadados:

- Repositório – local onde ficam fisicamente armazenados os dados e metadados;
- Dado – é a representação/denotação de um objeto de interesse, podendo ser usado para representar objetos simples de um determinado tipo ou a combinação de objetos simples;
- Metadado – conjunto de características e atributos que representam ou descrevem um dado. Os metadados geralmente têm a finalidade de localizar, avaliar, descobrir, analisar ou citar o dado a que se referem, não se limitando a essas funções;
- Valor – é um conjunto de elementos contendo uma sequência de símbolos;
- Domínio – define um conjunto de valores limitados por um tipo de dado (RAPID INTELLINGENCE, 2009);
- Tipo de dado – especifica um conjunto de dados e associa as operações suportadas pelos elementos deste conjunto (IEEE et al., 2001);
- Operação – define o conjunto de procedimentos que podem ser realizados sobre certa quantidade de elementos.

4.6 Pacote Eventos

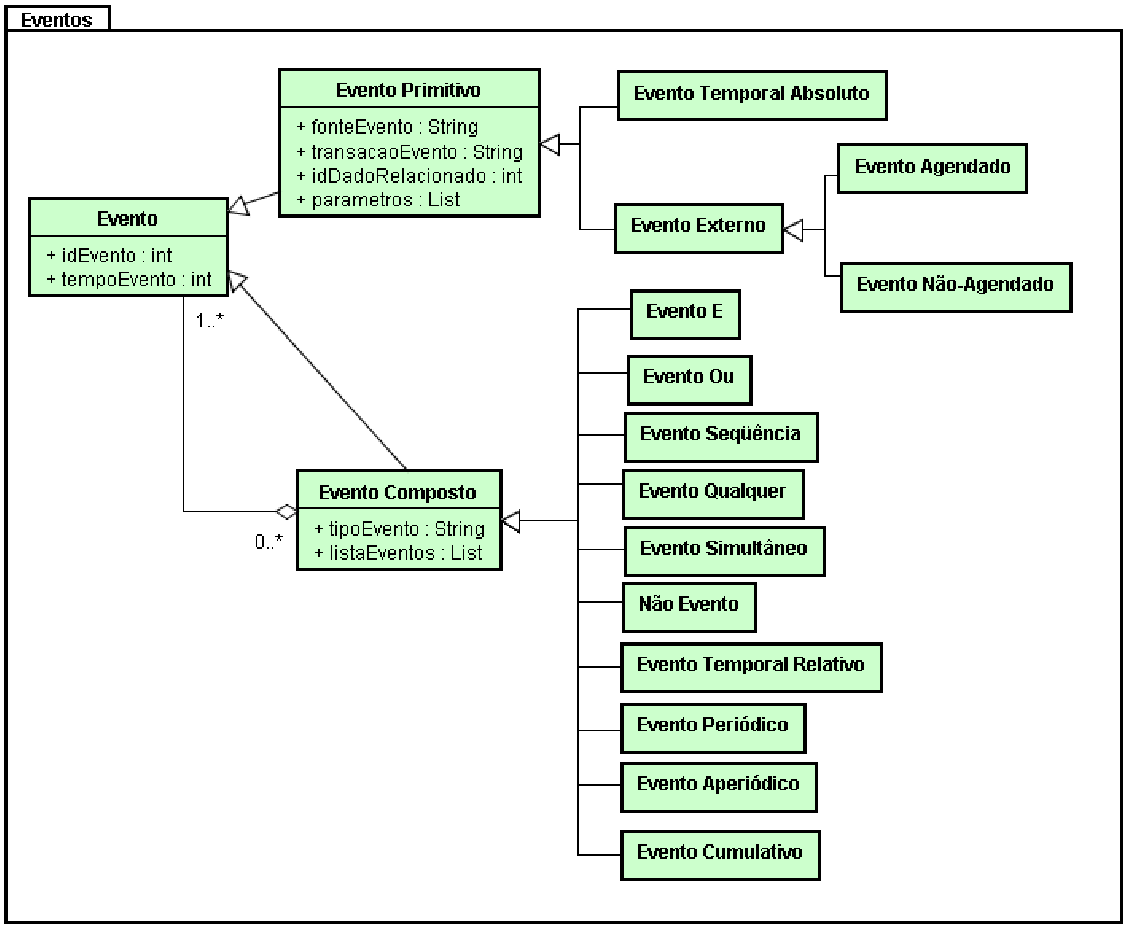

Figura 34 – Pacote Eventos

É importante considerar que para controlar o comportamento de um processo, devem existir mecanismos capazes de identificar mudanças e que comuniquem qualquer sinal de alteração expressiva. Com isso, segundo PARASHAR & HARIRI (2005), um elemento autonômico requer canais de sensores que detectem mudanças externas e internas no ambiente, e um canal que reaja a efeitos de mudanças no ambiente, alterando-o e mantendo o equilíbrio. Neste cenário, a análise dos eventos envolvidos é muito relevante.

O Pacote Eventos, Figura 34, permite modelar todas as ocorrências geradas e utilizadas pelos gestores de dados. A **Classe Evento** é definida como **uma ocorrência num ponto do tempo**.

Apesar de a literatura oferecer diferentes classificações para eventos, este trabalho está interessado particularmente nas características associadas à fonte geradora e à granularidade dos eventos.

Em termos de **fonte geradora** do evento, dependendo da literatura e da área adotada, eventos podem ser classificados de diferentes formas. Na área de Banco de Dados (CHAKRAVARTHY et al., 1994, CHAKRAVARTHY & MISHRA, 1994) e Gerenciamento Baseado em Políticas (LYMBEROPOULOS et al., 2003, MANSOURI-SAMANI & SLOMAN, 1997) podem ser classificados em internos e externos. Na área de Análise Essencial, os mesmos são divididos em dois grandes grupos, essenciais externos e essenciais temporais, não existindo os eventos essenciais internos (XEXÉO, 2007, SATZINGER, 1992, MCMENAMIN & PALMER, 1984). Esta última abordagem é a utilizada nesta tese.

No que concerne a **granularidade**, a literatura indica que os eventos podem ser divididos em: primitivos e compostos (CHAKRAVARTHY et al., 1994, CHAKRAVARTHY & MISHRA, 1994, PATON & DÍAZ, 1999). Considera-se esta divisão como ponto de partida da modelagem de eventos utilizados neste trabalho.

O primeiro grupo, de **eventos primitivos**, contém eventos simples, não compostos de outros eventos. As classes especializadas desses eventos podem ser definidas como:

- **Evento temporal absoluto** – corresponde a um ponto único no tempo especificado por valores absolutos de tempo, por exemplo, um evento que acontece na expressão de tempo usando uma instância no formato dia:mês:ano-hora:minuto:segundo;
- **Evento Externo** – corresponde a um evento gerado por uma aplicação externa. As classes especializadas dos eventos externos podem ser definidas como:
    - **Evento agendado** – corresponde a um evento que possui um momento certo para acontecer, ou que possui um limite de prazo para ocorrer;
    - **Evento não agendado** – corresponde a um evento no qual não se pode determinar o momento ou limite do seu acontecimento.

O segundo grupo, de **eventos compostos**, contém eventos formados pela combinação de eventos primitivos ou compostos (CHAKRAVARTHY et al., 1994,

CHAKRAVARTHY & MISHRA, 1994, GATZIU & DITTRICH, 1994, BABA-HAMED, 2006). Para que sua detecção seja possível, é necessário que o sistema detector de eventos possa armazená-los, permitindo a detecção das combinações desejadas. As classes especializadas dos eventos compostos podem ser definidas como:

- **E** (*And*) – indica a ocorrência de dois eventos *E1* e *E2*, independentemente da ordem (o ponto no tempo do evento composto corresponde à ocorrência do último evento);

- **Ou** (*Or*) – indica a ocorrência de um de dois eventos *E1* ou *E2*;

- **Sequência** (*Sequence*) – indica a ocorrência de dois eventos *E1* e *E2* em sequência (*E2* depois da ocorrência de *E1*, por exemplo);

- **Simultâneos** (*Simultaneous*) – indica que dois ou mais eventos ocorreram ao mesmo tempo;

- **Não Evento** (*Not*) – indica a não ocorrência de um evento em um determinado intervalo de tempo;

- **Cumulativo** (*History*) – indica a *n-ésima* ocorrência de um evento *E*. O tempo do evento composto corresponde ao ponto no tempo da *n-ésima* ocorrência do evento *E*;

- **Qualquer** (*Any*) – denotado por *ANY(m, E1, E2, ..., En)*, onde $m \leq n$, indica a ocorrência de *m* eventos de *n* eventos distintos, independentemente da ordem;

- **Evento Temporal Relativo** – corresponde a um ponto único no tempo especificado em relação ao tempo absoluto de detecção de outro evento, por exemplo, um evento que deve ocorrer 50 segundos após a notificação de um evento de escrita de um dado;

- **Periódico** (*Periodic*) – indica ocorrências de eventos que ocorrem em intervalos de tempo regulares;

- **Aperiódico** (*Aperiodic*) – indica a ocorrência de um evento dentro de um intervalo de tempo que pode ser denotado por *[tempo inicial, evento, tempo final]*. Duas considerações são possíveis: 1) o tempo do evento composto pode corresponder à primeira ocorrência do evento dentro do intervalo (chamado evento aperiódico não cumulativo); 2) o tempo do evento composto pode corresponder ao tempo final

estipulado de detecção de evento, desde que o evento ocorra pelo menos uma vez, sendo possível mais de uma ocorrência (chamado evento aperiódico cumulativo).

Adicionalmente, os seguintes atributos foram incluídos de modo a permitir a identificação e detecção de eventos primitivos e compostos:

- *idEvento* (*idEvent*) – identificador único do evento;
- *tempoEvento* (*eventTime*) – o instante no tempo em que o evento ocorreu (*timestamp*) baseado em um relógio global sincronizado. Se a fonte do evento não fornece o tempo, então o detector do evento usa o tempo de detecção como o instante (aproximado) em que o evento ocorreu;
- *listaEventos* (*eventList*) – lista de eventos que compõem um evento composto;
- *tipoEvento* (*eventType*) – é o tipo de evento composto. Por exemplo, *E*, *Ou*, *Não Evento*, etc;
- *fonteEvento* (*eventSource*) – identificação da entidade responsável pelo disparo do evento. Ex.: um usuário, o próprio sistema, etc;
- *transacaoEvento* (*eventTransaction*) – transação em que o evento ocorreu. Por exemplo, "*ler*" representa uma transação de leitura;
- *idDadoRelacionado* (*dataId*) – é o identificador do elemento de dado afetado pelo evento;
- *parametros* (*parameters*) – parâmetros adicionais no caso de eventos primitivos definidos por usuários, tais como o nome da classe e o nome do método em ambientes orientados a objetos.

É importante observar que, considerando os atributos dados anteriormente, não existem dois eventos iguais, sejam eles primitivos ou compostos.

O próximo capítulo utiliza os conceitos apresentados até agora e introduz os padrões de eliminação de dados, a metodologia utilizada na sua construção e a álgebra do SECONDO que associa cada padrão a um operador.

CAPÍTULO 5 PADRÕES AUTONÔMICOS DE ELIMINAÇÃO DE DADOS

Este capítulo apresenta as técnicas utlizadas na descrição dos padrões, bem como os padrões de eliminação de dados propostos. Atualmente, várias fontes de dados fornecem uma enorme quantidade de informações. Estas diferentes fontes de dados, de modo geral, não se relacionam, são heterogêneas e apresentam características diferentes em relação à qualidade dos dados. São exemplos disso as bases de dados heterogêneas de uma organização, alimentadas pelos seus diferentes setores e funcionários, ou as notícias Web fornecidas atualmente pelos diferentes canais de comunicação.

Diversas abordagens têm sido propostas no sentido de organizar e priorizar as informações existentes, tentando reduzir os efeitos do excesso de informação. Uma das possibilidades ainda não explorada neste contexto é o uso de padrões de eliminação de dados.

No caso dos padrões autonômicos para eliminação de dados propostos nesta tese, o objetivo é identificar os padrões de descarte existentes, classificá-los, descrevê-los e correlacioná-los com os casos de uso existentes. Uma vez que esses padrões de descarte de dados sejam conhecidos, diversas aplicações poderão se beneficiar do seu uso, propiciando uma melhor manutenção de seus repositórios de dados.

É importante considerar que a destruição de dados é um processo que deve ser realizado de maneira controlada e segura. Uma boa prática, principalmente para as organizações, é ter um documento que autorize a realização do descarte, fornecido pelo responsável pelo dado, bem como instruções de como o descarte deve ser realizado. A destruição de dados precisa ser claramente documentada para fornecer evidência da destruição de acordo com um programa de concordância. A utilização de padrões de eliminação bem documentados torna-se essencial nestas situações. Além disso, em alguns casos, pode ser essencial ter o planejamento para a recuperação emergencial de dados excluídos dos repositórios principais. Estes dados podem estar armazenados em repositórios secundários e permanentes. Nestes casos, também pode ser necessário prever estratégias que facilitem a recuperação desses dados, o que pode levar à criação

dos padrões inversos de eliminação de dados, ou seja, padrões que recuperem dados apagados pelos padrões de eliminação de dados.

5.1 Algumas Convenções Adotadas

Na modelagem dos padrões foram adotadas algumas convenções na construção dos modelos representados em RPAN. Estas convenções são adotadas nas inscrições dos lugares, transições e arcos, sendo detalhadas a seguir:

- Língua – em virtude da maioria dos padrões internacionais, a exemplo dos padrões propostos pelo W3C[37], adotarem o inglês como língua oficial, optou-se pela modelagem dos mesmos usando esta língua. Contudo, destaca-se o fato de a ferramenta utilizada só oferecer suporte ao inglês. Portanto, para suportar o português, seria necessário um trabalho extra de adaptação da ferramenta que foge ao escopo desta tese.

- Funções (dos arcos, de guarda) ou Variáveis – devem usar uma palavra ou uma composição de palavras, formada por um verbo seguido de um ou mais substantivos. A primeira palavra deve possuir todas as letras minúsculas, a primeira letra das palavras seguintes deve ser maiúscula e as demais letras devem ser minúsculas. Exemplo: *putData( )*.

- *read*, get*, put** e *check** – correspondem às variáveis e funções usadas para ler (*read**), extrair (*get**), colocar (*put**) e testar (*check**) dados dos repositórios. Estas variáveis e funções são escritas usando a linguagem CPN ML, uma implementação do padrão de linguagem ML, que é adotada pela ferramenta CPN *Tools*. As funções CPN ML podem chamar funções externas (ex.: Java, C) que realizam os processamentos nos repositórios reais e retornam ponteiros que indicam o posicionamento dos dados processados. São os valores retornados por essas funções que atualizam os ponteiros dos dados nos repositórios. O "*" indica que o nome dessas variáveis e funções pode ser estendido, a exemplo de *readData*.

- Tipo das Marcas – devem utilizar um nome com todas as letras maiúsculas. Exemplos: *INTEGER (INT), STRING, EVENT, DATA.*

[37] http://www.w3.org/

- Lixeira (*Trash*) – pode representar o descarte definitivo dos dados (não é possível a posterior recuperação dos dados) ou um repositório temporário onde os dados permanecem algum tempo antes de serem efetivamente descartados.

- Tipo *DATA* – corresponde aos tipos das marcas permitidas nos lugares que representam dados de repositórios diversos. O tipo *DATA* pode conter ponteiros para os dados dos repositórios ou os próprios dados dos repositórios de forma completa ou parcial. Outrossim, ele é utilizado para representar parâmetros necessários aos padrões no processamento dos dados.

- Tipo *EVENT* – corresponde aos tipos de marcas permitidos nos lugares que contêm eventos. Pode conter listas de eventos primitivos e/ou compostos. O tipo *EVENT* permite que sejam representados todos os atributos das classes Evento, Evento Primitivo e Evento Composto da Figura 34.

- Funções de Similaridade – correspondem às funções usadas na comparação de textos. Como os textos tratados nesta tese são relativamente curtos, foi escolhido o coeficiente de Jaccard na comparação dos mesmos, devida a sua facilidade de entendimento e implementação[38]. O coeficiente de Jaccard fornece valores entre 0 e 1, sendo que, nesta tese, estes valores foram convertidos para seus valores percentuais entre 0 e 100.

5.2 Formato dos Dados Aceitos como Marcas

Para fins de implementação e de modo a permitir que os modelos em RPAN possam trabalhar com diferentes tipos de dados e metadados, é proposta uma representação para os mesmos. Esta representação, inspirada no formato RDF [39] simplificado, permite descrever diferentes tipos de dados ou metadados, como os apresentados nos Pacotes Dados e Metadados do Capítulo 4, possuindo o formato:

"<identificador do dado ou metadado>

<nome do metadado ou atributo associado ao dado>

<valor do metadado ou valor do atributo associado ao dado>"

[38] Caso seja necessário fazer a comparação de textos longos em uma coleção de documentos, é mais recomendável utilizar tf-idf (*term frequency–inverse document frequency*) que considera a frequência do termo no documento e o inverso da frequência do termo entre os documentos da coleção (BAEZA-YATES et al., 1999).
[39] http://www.w3.org/RDF/

Esta tripla é representada em RPAN através do tipo *DATA* [40], criado especificamente para este fim. Isto permite o casamento de diferentes tipos de dados com os formatos das marcas suportadas pelos modelos em RPAN através da ferramenta CPN *Tools*. Na representação das marcas, cada um dos campos da tripla é representado por um par campo-valor, sendo o primeiro campo nomeado *idData* com valor do tipo inteiro, o segundo campo nomeado *metadata* com valor do tipo caracteres (*string*) e o terceiro campo nomeado *value* com valor do tipo caracteres (*string*). Supondo que se deseja converter os dados de duas pessoas: Brian com 33 anos e Joe com 40 anos, conforme apresentado na Tabela 5.

Tabela 5: Dados sobre nome e idade de duas pessoas usando a tripla (idData, metadata, value)

Dados	***idData***	***metadata***	***value***
Primeira Pessoa	1	nome	Brian
	1	idade	33
Segunda Pessoa	2	nome	Joe
	2	idade	40

A seguinte expressão representa as marcas representativas destes dados em CPN ML:

```
1`[{idData=1,metadata="nome",value="Brian"},
{idData=1,metadata="idade",value="33"},
{idData=2,metadata="nome",value="Joe"},
{idData=2,metadata="idade",value="40"}]
```

Pode-se notar que todos os valores dos dados devem ser convertidos em string para que possam ser processados pelos modelos descritos em RPAN na ferramenta CPN *Tools*.

Suponha agora que uma empresa chamada Sobremesas possua dois empregados John e Bill. Uma representação possível destes dados é apresentada na Tabela 6.

[40] A especificação dos tipos PRIMITIVE e COMPOSITE, que representam os eventos, é feita no próximo capítulo.

Tabela 6: Dados sobre uma empresa e seus empregados usando a tripla (idData, metadata, value)

idData	*metadata*	*value*
1	tipo	Empresa
1	nome	Sobremesas
1	componente	2
1	componente	3
2	tipo	Empregado
2	nome	John
3	tipo	Empregado
3	nome	Bill

Em CPN ML, ter-se-ia a seguinte expressão:

1`[{idData=1,metadata="tipo",value="Empresa"},
{idData=1,metadata="nome",value="Sobremesas"},
{idData=1,metadata="componente",value="2"},
{idData=1,metadata="componente",value="3"},
{idData=2,metadata="tipo",value="Empregado"},
{idData=2,metadata="nome",value="John"},
{idData=3,metadata="tipo",value="Empregado"}
{idData=3,metadata="nome",value="Bill"}]

Como a palavra *metadata* foi escolhida nesta implementação para representar não somente metadados, mas também atributos (incluindo relacionamentos), a relação de associação entre a empresa Sobremesas e os seus empregados é designada pela palavra *"componente"*. Outrossim, *value* representa o identificador do elemento associado. O relacionamento entre elementos também pode ser utilizado para indicar uma relação de dependência. Neste caso, a identificação do nome do relacionamento e se há ou não dependência entre os dados é feita por meio de parâmetros adicionais que utilizam no campo *metadata* as palavras-chave *dependentDataRelationship* e *dependentDataStrategy*.

Além disso, caso ocorra em uma operação de descarte de dados que possuem dependentes, parâmetros relativos à estratégia de eliminação dos dados com dependentes também podem ser fornecidas, indicando que:

- Os dados dependentes devem ser apagados em cascata, utilizando a palavra-chave *discard all*;

- Os dados dependentes devem ser deixados como dados órfãos, utilizando a palavra-chave *keep dependents*; ou

- Nem os dados dependentes, nem os dados que possuem dependentes devem ser apagados, utilizando a palavra-chave *keep all*.

A escolha da estratégia a ser adotada é uma decisão do usuário do arcabouço. Por exemplo, a Tabela 7 mostra os parâmetros que deveriam ser fornecidos de modo a indicar que todos os dados dependentes devem ser apagados e que o relacionamento indicador de dependência entre os dados possui o nome componente.

Tabela 7: Parâmetros definindo se os dados dependentes também serão descartados, caso os dados de que dependam sejam descartados

idData	***metadata***	***value***
1	*dependentDataStrategy*	*discard all*
2	*dependentDataRelationship*	*componente*

Em CPN ML, ter-se-ia a seguinte expressão correspondente:

1`[{idData=1,metadata="dependentDataStrategy",value="discard all"}

{idData=2,metadata="dependentDataRelationship",value="componente"}]

Com o objetivo de facilitar a visualização dos dados e parâmetros, todas as marcas apresentadas anteriormente serão, a partir deste ponto, representadas somente na forma tabular.

5.3 Método Proposto para Descrição e Formalização de Padrões usando Redes de Petri de Alto Nível

Este trabalho propõe um método para a descrição e formalização de padrões modelados em RPANs. Considera-se a descrição de fluxos de dados e de controle como elementos essenciais. Assim, apesar de serem usadas especificamente para modelar padrões de processos que realizam a eliminação de dados, as ideias a seguir descritas também podem ser usadas para modelar outros padrões de processos que lidem com fluxo de dados e controle. O método proposto para descrever e formalizar os padrões computacionais autonômicos perfaz os seguintes passos:

Passo 1) Identifica as tarefas a serem executadas sobre os dados (detectar dado, eliminar dado, etc). Convenciona-se que os nomes das tarefas possuem a forma *<verbo><objeto>*.

Passo 2) Identifica os eventos e ações que disparam as tarefas ou que resultam da execução das tarefas. Convenciona-se que os nomes dos eventos e ações devem ser dados por um ou mais substantivos.

Passo 3) Cria o fluxo de controle em redes de Petri de alto-nível a partir das tarefas, eventos e ações identificados. Eventos e ações iniciam tarefas, enquanto tarefas geram ações ou eventos. Este fluxo se assemelha a uma Cadeia de Processos Dirigida por Eventos associado à implantação de sistemas de ERP SAP/R3 (XEXÉO, 2007). Os eventos/ações são representados por lugares (do tipo *EVENT*) e as tarefas são representadas por transições. Na representação em redes de Petri, as transições, além de representarem uma tarefa, testam se a mesma pode ser executada através da análise de expressões de guarda. Dessa forma, este fluxo pode ser visto como uma combinação de regras ativas ECA. Em regras ECA, um evento que dispara uma condição gera uma ação, sendo que esta ação pode corresponder a um evento em outra regra. A Figura 35 exemplifica a modelagem proposta de uma estrutura básica de fluxo de controle evento-condição-ação (ECA) com duas regras encadeadas, utilizando a ferramenta CPN *Tools*[41]. Dessa forma, para o fluxo de controle, associam-se os seguintes significados aos elementos da rede de Petri:

- **Lugar** – Um Lugar é um conjunto assim definido:

$Lugar = \{lugar_1, lugar_2, ...lugar_n\}$, onde cada $lugar_i, 1 \leq i \leq n$, é um evento ou ação do tipo *EVENT*.

- **Transição** – representa a condição, através de expressões de guarda, para a ocorrência de uma ação ou evento. O nome dado à transição (que representa uma tarefa a ser executada caso as condições estabelecidas sejam satisfeitas) sumariza o objetivo da regra e, neste trabalho, também é utilizado para nomear a regra.

- **Arco de Entrada** – representa a conexão entre um lugar (evento ou ação) e uma transição. Especifica o conteúdo das marcas de entrada.

[41] A modelagem dos padrões foi feita em inglês em virtude do suporte oferecido pela ferramenta CPN Tools.

- **Arco de Saída** – representa a conexão entre uma transição e um lugar (evento ou ação). Especifica o conteúdo das marcas de saída, através de variáveis ou funções.

- **Marca** – Uma marca é um conjunto assim definido:

$Marca = \{marca_1, marca_2, ...marca_n\}$, onde cada $marca_i, 1 \leq i \leq n$, é a instância (ocorrência) de um evento ou ação.

Figura 35 - Modelagem do fluxo de eventos/ações se-então

Passo 4) Identifica os repositórios envolvidos no processo. Convenciona-se que os nomes dos repositórios são formados por um ou mais substantivos. A primeira letra de cada palavra deve ser maiúscula e as demais minúsculas. Exemplo: *Internal Repository*.

Passo 5) Tomando por base o fluxo de controle e os repositórios envolvidos no processo, cria o fluxo de dados. A interconexão entre os fluxos de dados e de controle deve ocorrer por meio de transições comuns. As estruturas dos dados presentes nos repositórios podem ser modeladas como estruturas de listas e registros, sendo que as instâncias dos dados são representadas por marcas coloridas, de acordo com a Seção 5.2. Propõe-se que só sejam importados os atributos dos dados necessários a sua identificação na base real. Para o fluxo de dados, associam-se os seguintes significados aos elementos da rede de Petri:

- **Lugar** – corresponde a um local de armazenamento temporário ou persistente dos dados. Nos lugares são colocados ponteiros para os dados nos repositórios reais. Deve ser observado que neste passo deseja-se modelar os dados a serem manipulados e não mais os eventos como ocorria no Passo 3 (os tipos dos lugares que modelam o fluxo de dado são tipo *DATA* e não tipo *EVENT*). Formalmente:

$Lugar = \{lugar_1, lugar_2, ...lugar_n\}$, onde cada $lugar_i, 1 \leq i \leq n$, é a instância de um repositório do tipo *DATA*.

- **Transição** – possui a condição que permite a passagem de dados entre repositórios.

- **Arco de Entrada** – representa a conexão entre um lugar (repositório) e uma transição. Especifica o conteúdo das marcas de entrada através de variáveis.

- **Arco de Saída** – representa a conexão entre uma transição e um lugar (repositório). Especifica o conteúdo das marcas de saída através de variáveis (regras ECA).

- **Marca** – Uma marca é um conjunto assim definido:

$Marca = \{marca_1, marca_2, ...marca_n\}$, onde cada $marca_i, 1 \leq i \leq n$, representa a instância de um dado que identifica por meio de um ponteiro o dado no repositório real.

A Figura 36 exemplifica a modelagem proposta de uma estrutura de fluxo de dados e de controle. Os repositórios que podem conter dados foram destacados (em cinza). A transição *DETECT DATA* apenas verifica a existência de dados proibidos no repositório interno, portanto é usado um arco de leitura para conectar estes elementos. A transição *DISCARD PROHIBITED DATA* elimina efetivamente os dados, portanto são usados dois arcos, um que obtém todos os dados do repositório Internal Repository e outro que retorna os dados considerados não proibidos para o mesmo repositório. Existe também outro arco que leva os dados proibidos para o repositório Trash. As variáveis *readData1*, *getData* and *event* tornam possível o fluxo de marcas nos arcos.

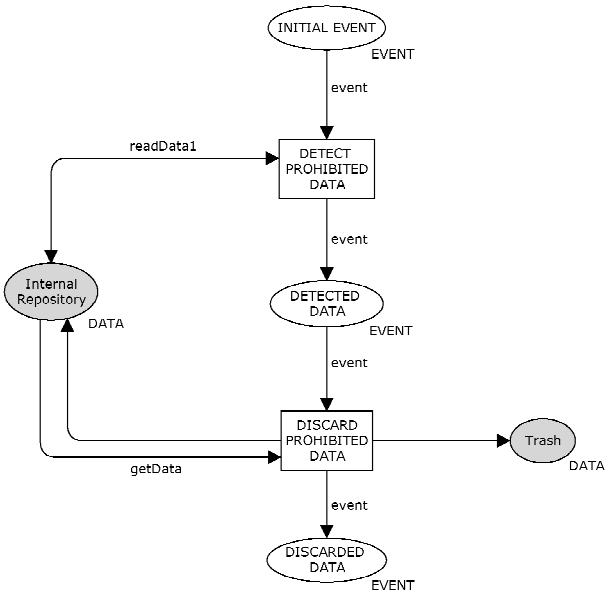

Figura 36 - Modelagem do fluxo de dados e controle

Passo 6) Cria um ou mais lugares para armazenamento dos parâmetros e os conectar às transições de interesse. O conjunto de parâmetros de cada padrão pode variar, portanto o uso de lugares para a entrada de parâmetros permite maior

flexibilidade aos padrões, pois os mesmos podem ser configurados para diferentes cenários de aplicação. São exemplos de parâmetros: a estratégia de similaridade a ser usada (Jaccard, Dice, palavras-chave, etc.) (KOUDAS et al., 2006), os atributos a serem comparados (nome, CPF, etc.), o idioma dos dados a serem processados (inglês, português, etc.), o tipo de pré-processamento dos dados (*stemming*, *stopwords*, *etc.*), a ordem de saída dos dados (ordem da entrada, ordem por data, ordem por tamanho, ordem por similaridade com os termos comparados, etc), entre outros.

Passo 7) Cria as funções dos arcos. A Figura 37 mostra duas funções criadas: *putProhibitedData ( )* e *putSub( ).* A primeira função seleciona os dados proibidos, considerando os dados existentes nos lugares Internal Repository e Parameters. A segunda função obtém os dados que não são proibidos a partir dos dados obtidos pela primeira função. Como fruto do presente trabalho, foi criada uma biblioteca de funções, utilizando a linguagem CPN ML, que permite a manipulação e eliminação de dados. A descrição dessa biblioteca pode ser encontrada no Anexo I.

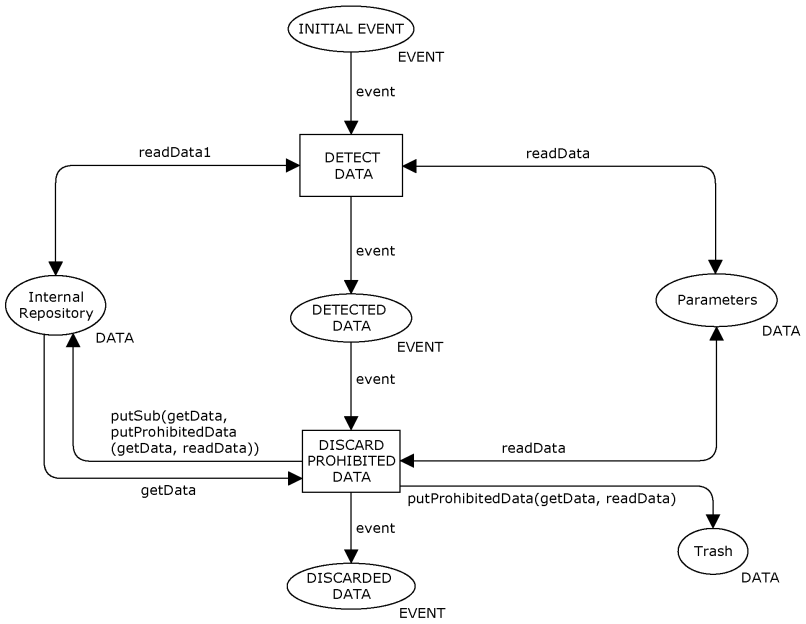

Figura 37 - Modelagem do fluxo de dados e controle considerando parâmetros e funções

Depois de modelado um padrão, ele passa a receber eventos e dados dos sistemas, monitorando estes ambientes e executando ações de acordo com suas regras comportamentais. Isto permite que as regras atuem autonomicamente, diminuindo a sobrecarga de trabalho sobre as pessoas. Com relação às propriedades das redes de Petri

discutadas na Seção 3.3.2, tem-se que a rede deve ser: limitada, l1-viva, justa e com a quantidade de dados conservada durante a execução da mesma.

O modelo da Figura 37 corresponde ao padrão Eliminador de Dados Proibidos que será discutido em maiores detalhes mais adiante.

5.4 Padrões de Eliminação de Dados

Seguindo o método detalhado anteriormente, foram elicitados oito padrões, quais sejam (PINHEIRO et al., 2008b, PINHEIRO et al., 2008c, PINHEIRO et al., 2009a, PINHEIRO et al., 2009b, PINHEIRO et al., 2010): Padrão Dados Permitidos (*Allowed Data* ou *AD*), Padrão Bloqueador de Dados Proibidos (*Blocking Prohibited Data* ou *BP*), Padrão, Padrão Bloqueador de Dados Similares (*Blocking Similar Data* ou *BS*), Padrão Bloqueador de Dados Irrelevantes (*Blocking Irrelevant Data* ou *BI*), Padrão Eliminador de Dados Proibidos (*Discarding Prohibited Data* ou *DP*), Padrão Eliminador de Dados Similares (*Discarding Similar Data* ou *DS)*, Padrão Eliminador de Dados Obsoletos (*Discarding Obsolete Data* ou *DO*) e Padrão Eliminador de Dados Irrelevantes (*Discarding Irrelevant Data* ou *DI*).

A ideia geral que norteou a criação dos padrões citados anteriormente partiu da premissa de que, do ponto de vista do usuário, os dados indesejados podem ser descartados de duas formas:

- Impedindo a sua entrada nos repositórios de interesse, ou seja, descartando-os antes mesmos que sejam incorporados às bases de dados dos usuários;

- Eliminando esses dados dos repositórios de interesse, ou seja, descartando-os depois que os mesmos estejam incorporados às bases de dados dos usuários.

De modo geral, os padrões elicitados neste trabalho podem ser divididos em duas grandes categorias: padrões bloqueadores de dados indesejados e padrões eliminadores de dados indesejados.

Além disso, cada padrão apresenta características autonômicas dominantes. Assim, procurou-se relacionar os padrões criados com essas características. A Tabela 8 mostra a classificação dos padrões considerando os fatores mencionados anteriormente.

Tabela 8: Classificação dos padrões de eliminação de dados segundo suas características autonômicas e modos de descarte

Padrões		Características			
		Autoproteção	Autocura	Autoconfiguração	Auto-otimização
Bloqueadores	Dados Permitidos			X	X
	Bloqueador de Dados Proibidos	X			
	Bloqueador de Dados Similares	X			
	Bloqueador de Dados Irrelevantes	X			
Eliminadores	Eliminador de Dados Proibidos		X		
	Eliminador de Dados Similares				X
	Eliminador de Dados Obsoletos		X		X
	Eliminador de Dados Irrelevantes		X		X

As próximas seções detalham os padrões apresentados anteriormente. A estrutura documental dos padrões usa uma adaptação simplificada do formato GoF, proposto por GAMMA (1994), contendo: motivação, aplicabilidade, modelo e exemplo.

5.5 Padrão Dados Permitidos

5.5.1 Motivação

Existem situações onde é necessário autorizar a inclusão de dados externos que possuem certas características nos repositórios internos de sistemas, redes ou organizações. Isto é muito comum em sistemas que utilizam dados disponíveis na *Web*, a exemplo dos sistemas que trabalham com notícias *Web* (*feeds* de notícias) ou com emails (sistemas antispam quando trabalham com listas brancas). Outro cenário onde este tipo de situação pode ser encontrado é na construção dos repositórios de armazéns

de dados (*data warehouses*), responsáveis por fornecer uma visão gerencial dos dados produzidos por sistemas transacionais.

Normalmente, os sistemas mencionados anteriormente buscam dados usando algum critério (por exemplo, busca de atributos, palavras-chave, assinaturas ou certos textos), podendo haver o uso de funções de similaridade para auxiliar na seleção. Neste contexto, os dados não selecionados (por exemplo, proveniente da *Web* ou de aplicações transacionais) podem ser descartados, enquanto os outros são incorporados aos repositórios internos.

5.5.2 Aplicabilidade

Autorizar a entrada de dados de interesse para uma organização, sistema ou certas pessoas. Os dados não selecionados são considerados dados descartados e podem ser eliminados. Dessa forma, podem ser atenuados os efeitos da explosão de dados nos repositórios internos.

5.5.3 Modelo

A Figura 38 detalha este padrão. Em termos de repositórios, ela apresenta o *External Repository* que representa um repositório externo à aplicação, contendo os dados a serem processados; o *Parameters* que representa um repositório de parâmetros, contendo uma lista de dados de interesse e informações adicionais, tais como o nome da função de similaridade a ser usada na identificação dos dados permitidos; o *Internal Repository* que representa o repositório interno à aplicação e armazena os dados selecionados; e a lixeira (*Trash*).

Ao receber um evento iniciador do processo (*INITIAL EVENT*), a primeira regra (*DETECT DATA*) verifica a existência de dados no repositório externo, através do arco de leitura e da variável *readData*. Se existirem dados novos, o evento *DETECTED DATA* é disparado, o que permite o processamento da segunda regra (*ALLOW DATA*). Nessa regra, a variável *getData* extrai os dados do repositório externo, enquanto que a variável *readData* lê os dados do repositório de parâmetros. Os dados do repositório externo são comparados com os parâmetros e, se existirem similaridades de acordo com a função de similaridade escolhida, os dados são copiados para o repositório interno, através do arco que contém a função *putAllowedData( ).* A função *putSub( )* transfere os dados sem relevância para a lixeira.

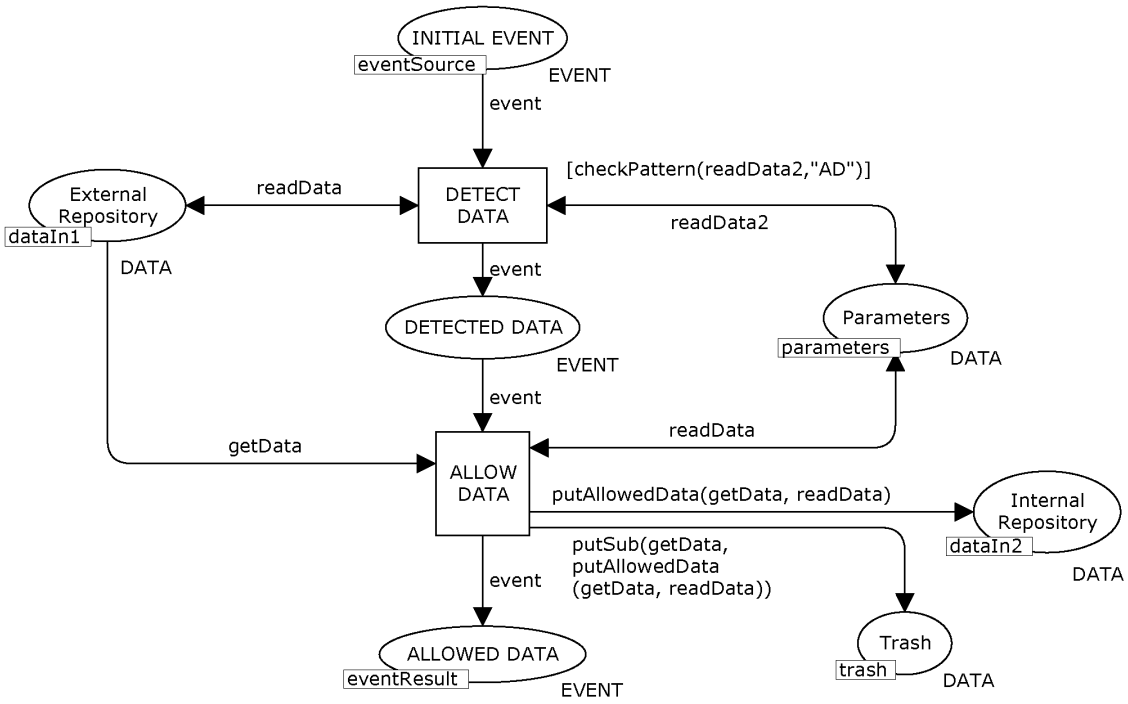

Figura 38 - Dados Permitidos

5.5.4 Exemplo

Suponha que o repositório externo contenha marcas que indicam quatro instâncias de dados, cada uma composta por dois metadados e valores associados, representando o tópico do dado (*topic*) e o texto associado a este tópico (*text*). Todos estes dados podem conter tópicos de interesse ou não. A Tabela 9 fornece um exemplo para esta situação.

Tabela 9: Dados a serem tratados pelo padrão AD

idData	***metadata***	***value***
1	*topic*	*Advertising*
1	*text*	*Information about ads*
2	*topic*	*Database*
2	*text*	*Information about DB*
3	*topic*	*Workflow*
3	*text*	*Information about WF*
4	*topic*	*Database and Applications*
4	*text*	*Information about DB*

Suponha agora que os parâmetros apresentados na Tabela 10 foram declarados, definindo: o metadado que será comparado (*topic*), os valores de interesse para este metadado (*database* e *workflow*), a função de similaridade a ser usada na comparação desses valores (Jaccard) e o nível de similaridade mínimo para que o valor seja considerado de interesse (30%).

Tabela 10: Parâmetros a serem utilizados pelo padrão AD

idData	***metadata***	***value***
1	*comparedMetadata*	*topic*
2	*topic*	*Database*
3	*topic*	*Workflow*
4	*similarityStrategy*	*Jaccard*
5	*similarityCoeficient*	30

Dessa forma, o dado apresentado na Tabela 11 seria descartado, pois não apresenta nenhuma similaridade com os assuntos de interesse.

Tabela 11: Dado descartado

idData	***metadata***	***value***
1	*topic*	*Advertising*
1	*text*	*Information about ads*

Depois de realizado o processamento, os dados mostrados na Tabela 12 seriam transferidos para o repositório interno, pois apresentam nível de similaridade maior ou igual a 30% com qualquer dos tópicos de interesse.

Tabela 12: Dados transferidos para o repositório interno

idData	***metadata***	***value***
2	*topic*	*Database*
2	*text*	*Information about DB*
3	*topic*	*Workflow*
3	*text*	*Information about WF*
4	*topic*	*Database and Applications*
4	*text*	*Information about DB*

5.6 Padrão Bloqueador de Dados Proibidos

5.6.1 Motivação

Muitos dados podem causar prejuízos às redes, sistemas, repositórios e, consequentemente, aos seus usuários. Normalmente, estes dados são provenientes do ambiente externo. Dessa forma, barreiras devem ser implementadas de forma a impedir que esses dados sejam incorporados aos repositórios internos. Têm-se como exemplos os sistemas antispam e os sistemas de detecção de softwares maliciosos, nos quais os dados, antes de serem incorporados aos repositórios internos, devem passar por um processo de verificação e análise que pode envolver tanto a análise dos metadados (fonte do dado, proprietário, frequência de envio, etc) como a análise do conteúdo dos dados (busca de textos específicos, certos tipos de assinatura, etc). Em ambos os casos, pode haver o uso de funções de similaridade para auxiliar a identificar os dados proibidos.

É importante também considerar que no ambiente *Web*, em função da liberdade e falta de controle intrínsicos, algumas pessoas geram sites com conteúdo impróprio ou criminoso, sendo necessário bloqueá-los de determinadas audiências, a exemplo dos sites que promovem terrorismo, racismo, pirataria, etc.

5.6.2 Aplicabilidade

Impedir a entrada de dados proibidos ou que possam ser prejudiciais a um repositório, um sistema, seus usuários ou a uma organização. Estes dados são comparados a certos termos antes de serem incorporados aos repositórios internos. É necessária uma fonte de dados contendo as características dos dados a serem eliminados e/ou as regras a serem aplicadas na obtenção desses dados. Dessa forma, podem ser minimizados os efeitos da explosão e poluição de dados nos repositórios internos.

5.6.3 Modelo

A Figura 39 detalha este padrão. Em termos de repositórios, ela apresenta o *External Repository* que representa um repositório externo à aplicação, contendo os dados a serem processados; o *Parameters* que representa um repositório de parâmetros, contendo uma lista de dados proibidos e informações adicionais, tais como o nome da função de similaridade a ser usada na identificação dos dados proibidos; o *Internal*

Repository que representa o repositório interno à aplicação e armazena os dados verificados/autorizados; e a lixeira (*Trash*).

Ao receber um evento iniciador do processo (*INITIAL EVENT*), a primeira regra (*DETECT DATA*) verifica a existência de dados no repositório externo, através do arco de leitura e da variável *readData*. Se existirem dados novos, o evento *DETECTED DATA* é disparado, o que permite o processamento da segunda regra (*BLOCK PROHIBITED DATA*). Nessa regra, a variável *getData* extrai os dados do repositório externo, enquanto que a variável *readData* lê os dados do repositório de parâmetros. Os dados do repositório externo são comparados com os parâmetros e, se não existirem similaridades, os dados são copiados para o repositório interno, através do arco que contém a função *putNotProhibitedData( )*. A função *putSub( )* transfere os dados proibidos para a lixeira.

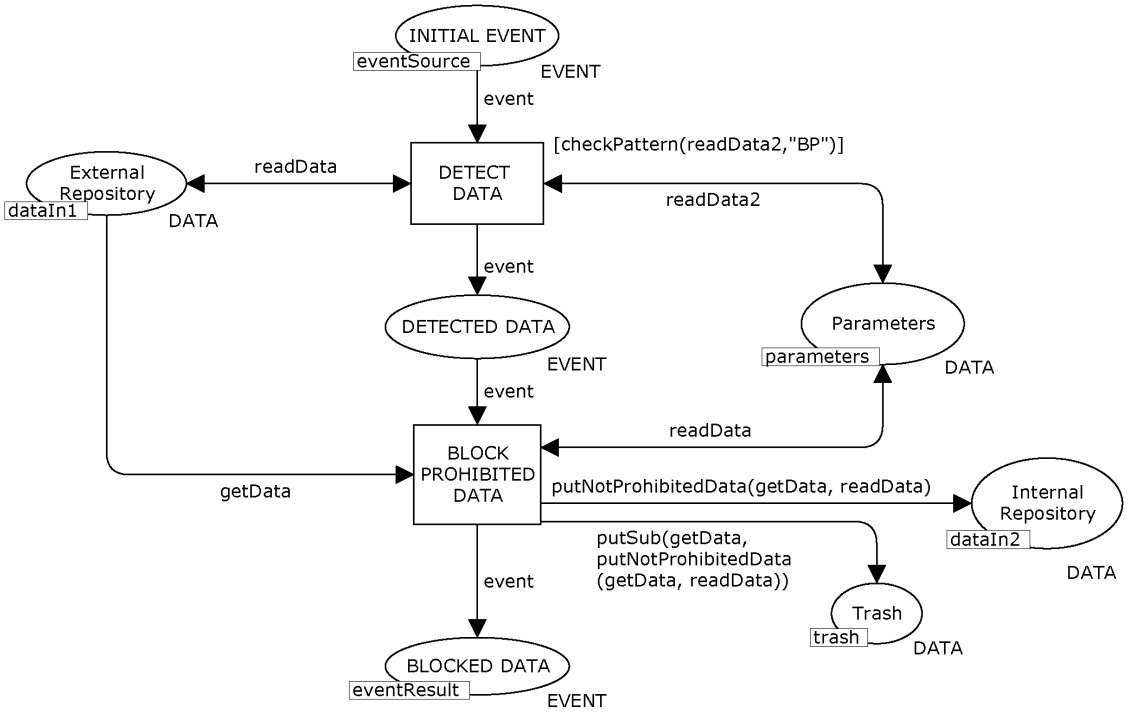

Figura 39 - Bloqueador de Dados Proibidos

5.6.4 Exemplo

Suponha que o repositório externo contenha marcas que indicam seis instâncias de dados, cada uma composta por dois metadados e valores associados, representando o tópico do dado (*topic*) e o texto associado a este tópico (*text*). Todos esses dados podem

conter tópicos de proibidos ou não. Adicionalmente, o *dado 1* relaciona-se com o *dado 4* e o *dado 5*, e o *dado 5* relaciona-se com o *dado 6*. A Tabela 13 retrata essa situação.

Tabela 13: Dados a serem tratados pelo padrão BP

idData	***metadata***	***value***
1	*topic*	*Advertising*
1	*text*	*Information about ads*
1	*contains*	4
1	*contains*	5
2	*topic*	*Database*
2	*text*	*Information about DB*
3	*topic*	*Workflow*
3	*text*	*Information about WF*
4	*topic*	*Ads about Databases*
4	*text*	*Information about databases ads*
5	*topic*	*Ads about Petri*
5	*text*	*Information about Petri ads*
5	*incorporates*	6
6	*topic*	*Ads about Workflow*
6	*text*	*Information about workflow ads*

Suponha agora que os parâmetros descritos na Tabela 14 foram declarados, definindo: o metadado que será comparado (*topic*), os valores de interesse para este metadado (*database* e *workflow*), a função de similaridade a ser usada na comparação desses valores (Jaccard), o nível de similaridade mínimo para que o valor seja considerado proibido (20%) e que os dados dependentes (identificados pelos metadados *contains* e *incorporates*) sejam descartados juntamente com os dados principais através da definição da estratégia de descarte para dados dependentes (*discard all*).

Tabela 14: Parâmetros a serem utilizados pelo padrão BP

idData	***metadata***	***value***
1	*comparedMetadata*	*topic*
2	*topic*	*Advertising*
3	*similarityStrategy*	*Jaccard*
4	*similarityCoeficient*	20
5	*dependentDataStrategy*	*discard all*
6	*dependentDataRelationship*	*contains*
6	*dependentDataRelationship*	*incorporates*

Dessa forma, os dados apresentados na Tabela 15 seriam descartados, pois apresentam similaridade superior a 20% com o assunto proibido (100% de similaridade).

Tabela 15: Dados descartados

idData	***metadata***	***value***
1	*topic*	*Advertising*
1	*text*	*Information about ads*
1	*contains*	4
1	*contains*	5
4	*topic*	*Ads about Databases*
4	*text*	*Information about databases ads*
5	*topic*	*Ads about Petri*
5	*text*	*Information about Petri ads*
5	*incorporates*	6
6	*topic*	*Ads about Workflow*
6	*text*	*Information about workflow ads*

Depois de realizado o processamento, os dados da Tabela 16 seriam transferidos para o repositório interno, pois apresentam nível de similaridade menor que 20% com qualquer dos tópicos de interesse.

Tabela 16: Dados transferidos para o repositório interno

idData	*metadata*	*value*
2	*topic*	*Database*
2	*text*	*Information about DB*
3	*topic*	*Workflow*
3	*text*	*Information about WF*

5.7 Padrão Bloqueador de Dados Similares

5.7.1 Motivação

Nos dias atuais, é comum lidar com fontes de dados externas e heterogêneas, que não são eficientemente controladas e muitas vezes possuem dados iguais ou muito similares sobre um mesmo assunto. Um dos maiores problemas nestes casos é identificar os dados que representem o mesmo objeto no mundo real. Por exemplo, podem existir dois registros, um com o nome de um estado da federação sem abreviações e o outro com o nome do estado abreviado. Ou seja, duas representações diferentes do mesmo objeto no mundo real. O processo de identificação de dados que se referem à mesma entidade ou objeto no mundo real, mesmo que com palavras escritas erroneamente, abreviações e tipos ou estilos de escritas diferentes possui diversos nomes na literatura, tais como: deduplicação (*deduplication*) (CARVALHO et al., 2006, SARAWAGI & BHAMIDIPATY, 2002), ligação de registros (*record linkage*) (CHRISTEN & CHURCHES, 2005, CHRISTEN et al., 2004, FELLEGI & SUNTER, 1969), semelhança de objetos (*object isomerism*) (CHEN et al., 1996), reconciliação de entidades (*entity reconciliation*) (DEY et al., 2002), heterogeneidade de entidades (*entity heterogeneity*) (DEY et al., 1998), junção/purificação (*merge/purge*) (HERNÁNDEZ & STOLFO, 1995), identificação de instâncias (*instance identification*) (WANG & MADNICK, 1989), entre outros.

5.7.2 Aplicabilidade

Impedir a incorporação de dados similares ou duplicados em repositórios internos segundo certas funções e níveis de similaridade entre os dados. Os dados não similares aos dados internos são incorporados normalmente, enquanto os demais são eliminados. Caso dois dados externos sejam similares entre si, o dado mais relevante,

segundo um determinado critério de escolha (dado mais recente, dado contendo o texto mais longo, etc), deve ser escolhido. Dessa forma, este padrão possibilita minimizar os efeitos da poluição de dados nos repositórios internos.

5.7.3 Modelo

A Figura 40 detalha este padrão. Em termos de repositórios, ela apresenta o *External Repository* que representa um repositório externo à aplicação, contendo os dados a serem processados; o *Parameters* que representa um repositório de parâmetros, contendo informações adicionais, tais como: o nome da função de similaridade a ser usada na identificação dos dados similares, o critério de escolha para dois dados similares, entre outros; o *Internal Repository* que representa o repositório interno à aplicação e armazena os dados verificados/autorizados; e a lixeira (*Trash*).

Ao receber um evento iniciador do processo (*INITIAL EVENT*), a primeira regra (*DETECT DATA*) verifica a existência de dados no repositório externo, através do arco de leitura e da variável *readData*. Se existirem dados novos, o evento *DETECTED DATA* é disparado, o que permite o processamento da segunda regra (*BLOCK SIMILAR DATA*). Nessa regra, a variável *getData1* extrai os dados do repositório externo, a variável *getData2* extrai os dados do repositório interno, enquanto que a variável *readData* lê os dados do repositório de parâmetros. Primeiramente, os dados do repositório externo são comparados entre si. Se existirem similaridade entre eles, o critério de escolha (ex.: dado, tamanho, etc) determina qual dado deve permanecer. Depois, os dados do repositório externo (já sem similaridades) são comparados com os dados do repositório interno e, se não existirem similaridades de acordo com a função de similaridade escolhida, os dados são copiados para este último, sendo acrescidos aos dados já existentes, através do arco que contém a inscrição *getData2^^putNotSimilarData( )*. A função *putSub( )* transfere os dados proibidos para a lixeira.

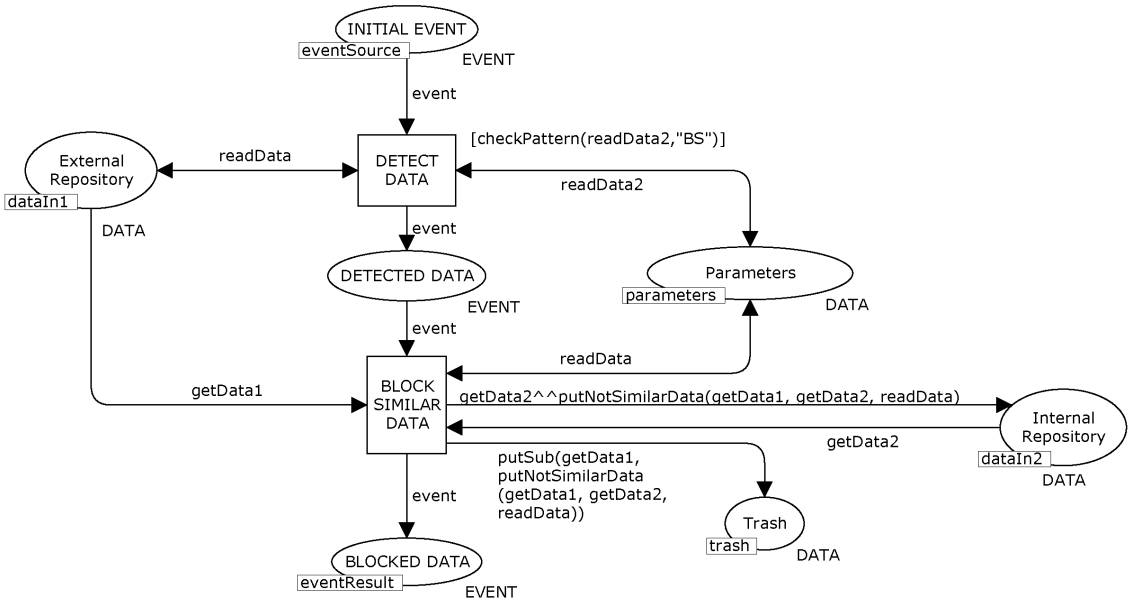

Figura 40 - Bloqueador de Dados Similares

5.7.4 Exemplo

Suponha que o repositório externo contenha marcas que indicam quatro instâncias de dados, cada uma composta por três metadados e valores associados, representando o tópico do dado (*topic*), o texto associado a este tópico (*text*) e a sua data de publicação (*publication date*). Todos estes dados podem ser similares entre si ou não. A Tabela 17 retrata essa situação.

Tabela 17: Dados a serem tratados pelo padrão BS

idData	***metadata***	***value***
1	*topic*	*Advertising*
1	*text*	*Buy our new software*
1	*publication date*	*01/01/2005*
2	*topic*	*Database*
2	*text*	*Information about DB*
2	*publication date*	*01/03/2006*
3	*topic*	*Workflow*
3	*text*	*Standards in workflows*
3	*publication date*	*12/12/22008*
4	*topic*	*Advertising*
4	*text*	*Buy our new software*
4	*publication date*	*01/01/2009*

Considere também que o repositório interno contém o dado apresentado na Tabela 18.

Tabela 18: Dado contido no repositório interno

idData	***metadata***	***value***
5	*topic*	*Database*
5	*text*	*Relational database is important*
5	*publication date*	*01/01/2007*

Suponha agora que os parâmetros da Tabela 19 foram declarados, definindo: o metadado a ser comparado (*text*), a função de similaridade a ser usada (Jaccard), o nível de similaridade (50%), a opção de escolha de prioridade para dois dados similares (*moreRecentDate*, sendo que, atualmente, as seguintes opções estão disponíveis: *moreRecentDate*, *olderDate*, *higherLength*, *lowerLength*), o metadado que será utilizado para a escolha de prioridade (*date*), a língua a ser utilizada (*english*), a necessidade de extração dos radicais das palavras que compõem o texto dos dados (técnica de *stemming*) e a necessidade de eliminação de *stopwords*.

Tabela 19: Parâmetros a serem utilizados pelo padrão BS

idData	***metadata***	***value***
1	*comparedMetadata*	*text*
2	*similarityStrategy*	*Jaccard*
3	*similarityCoeficient*	50
4	*moreRecentDate*	*publication date*
5	*language*	*english*
6	*stemming*	*true*
7	*stopwords*	*true*

Assim, antes do processamento das regras, os dados do repositório externo e interno serão pré-processados com as técnicas de *stemming* e *stopwords*, assumindo os formatos apresentados na Tabela 20 e na Tabela 21.

Tabela 20: Dados pré-processados do repositório externo

idData	*metadata*	*value*
1	*topic*	*advertis*
1	*text*	*buy new softwar*
1	*publication date*	*01/01/2005*
2	*topic*	*databas*
2	*text*	*inform db*
2	*publication date*	*01/03/2006*
3	*topic*	*workflow*
3	*text*	*standard workflow*
3	*publication date*	*12/12/22008*
4	*topic*	*advertis*
4	*text*	*buy new softwar*
4	*publication date*	*01/01/2009*

Tabela 21: Dado pré-processado do repositório interno

idData	*metadata*	*value*
5	*topic*	*databas*
5	*text*	*relat databas import*
5	*publication date*	*01/01/2007*

Este formato serve apenas para a aplicação das regras, sendo que o formato de entrada original dos dados dever ser restabelecido, após o processamento. Dessa forma, depois de realizado o processamento, os dados da Tabela 22 serão descartados, pois apresentam níveis de similaridade igual ou superior a 50% com dados existentes no repositório interno ou com dados existentes no respositório externo.

Tabela 22: Dados descartados

idData	*metadata*	*value*
1	*topic*	*Advertising*
1	*text*	*Buy our new software*
1	*publication date*	*01/01/2005*
2	*topic*	*Database*
2	*text*	*Information about DB*
2	*publication date*	*01/03/2006*

Em contrapartida, os dados mostrados na Tabela 23 serão incorporados/mantidos no repositório interno.

Tabela 23: Dados do repositório interno após o processamento

idData	*metadata*	*value*
3	*topic*	*Workflow*
3	*text*	*Standards in workflows*
3	*publication date*	*12/12/22008*
4	*topic*	*Advertising*
4	*text*	*Buy our new software*
4	*publication date*	*01/01/2009*
5	*topic*	*Database*
5	*text*	*Relational database are important*
5	*publication date*	*01/01/2007*

5.8 Padrão Bloqueador de Dados Irrelevantes

5.8.1 Motivação

Muitas vezes pessoas e organizações têm que lidar com um grande volume de dados, sendo que apenas uma parte deles é relevante. Elas devem selecionar os dados de interesse e descartar os demais. O padrão originado dessas aplicações busca obter, a partir de repositórios externos, os dados mais relevantes às mesmas, através da seleção de valores de interesse para certos metadados. Essa seleção funciona como um filtro interposto entre os repositórios externos e os internos de modo a permitir somente a inclusão de dados relevantes, minimizando o problema original do grande volume de dados. Os filtros de dados (Ex.: Yahoo *Pipe*) e os sistemas que manuseiam grande

volume de dados provenientes de experiências científicas (Ex.: sistemas que usam a linguagem Fado 2.0 (WERNER, 1992) no Centro Europeu de Pesquisas) são exemplos de abordagem que lidam com dados irrelevantes.

5.8.2 Aplicabilidade

Bloquear a entrada de dados irrelevantes segundo as necessidades de um sistema, organização ou das pessoas que o utilizam e que, dessa forma, possam poluir as informações existentes nos repositórios internos e sobrecarregar os sistemas que utilizam estes dados. Estes dados são selecionados em função dos seus metadados e conteúdo associados que, por sua vez, são comparados a termos de interesse. É necessária uma fonte de dados que contenha os dados, metadados e/ou regras para obtenção dos dados de interesse. Dessa forma, este padrão possibilita minimizar os efeitos da explosão e poluição de dados nos repositórios internos.

5.8.3 Modelo

A Figura 41 detalha este padrão. Em termos de repositórios, ela apresenta o *External Repository* que representa um repositório externo à aplicação, contendo os dados a serem processados; o *Parameters* que representa um repositório de parâmetros, contendo informações adicionais, tais como as regras de seleção dos dados relevantes; o *Internal Repository* que representa o repositório interno à aplicação e armazena os dados verificados/autorizados; e a lixeira (*Trash*).

Ao receber um evento iniciador do processo (*INITIAL EVENT*), a primeira regra (*DETECT DATA*) verifica a existência de dados no repositório externo, através do arco de leitura e da variável *readData*. Se existirem dados novos, o evento *DETECTED DATA* é disparado, o que permite o processamento da segunda regra (*BLOCK IRRELEVANT DATA*). Nessa regra, a variável *getData* extrai os dados do repositório externo, enquanto que a variável *readData* lê os dados do repositório de parâmetros. Os dados do repositório externo são verificados a partir dos parâmetros e, se forem selecionados, os dados são copiados para o repositório interno, através do arco que contém a função *putRelevantData( )*. A função *putSub( )* transfere os dados irrelevantes para a lixeira.

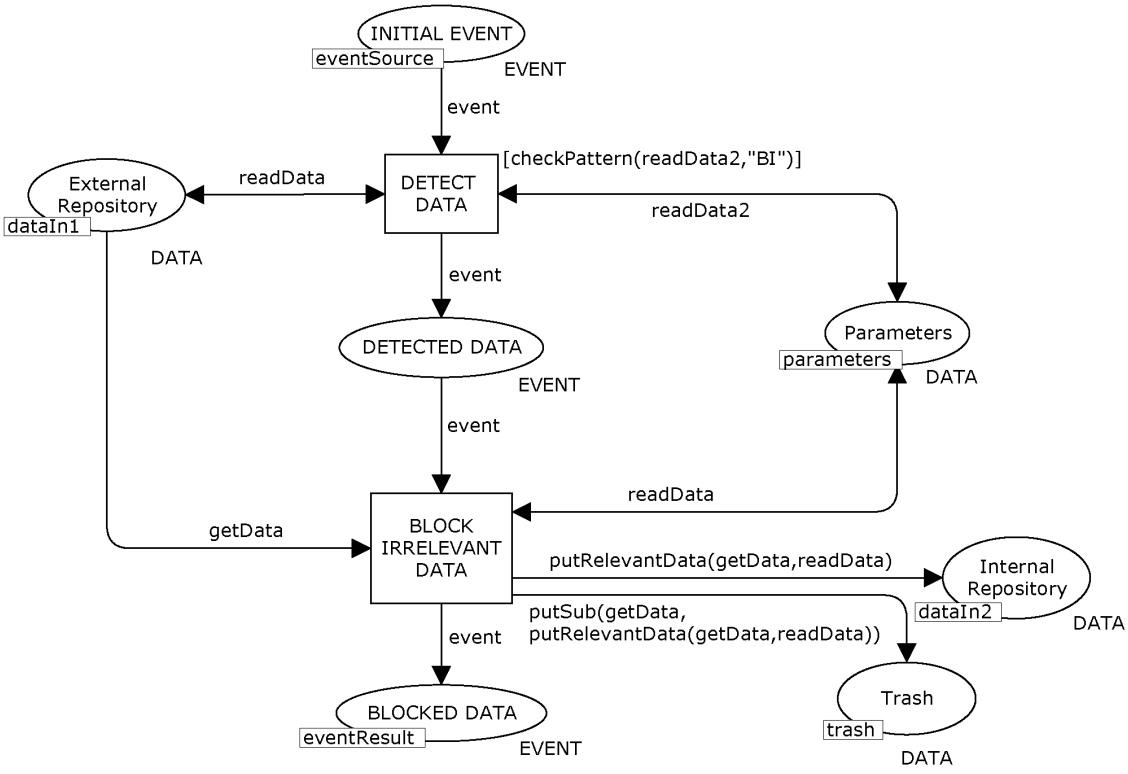

Figura 41 - Bloqueador de Dados Irrelevantes

5.8.4 Exemplo

Suponha que o repositório externo contenha marcas que indicam quatro instâncias de dados, cada uma composta por três metadados e valores associados. A primeira instância representa o tópico do dado (*topic=music*), a segunda instância representa o número de segundos relacionados a este tópico (*seconds*) e a terceira instância representa o estilo associado a este tópico (*style*). Todos estes dados podem ser relevantes ou não. A Tabela 24 retrata essa situação.

Tabela 24: Dados a serem tratados pelo padrão BI

idData	***metadata***	***value***
1	*topic*	*music*
1	*seconds*	180
1	*style*	*classic*
2	*topic*	*music*
2	*seconds*	*200*
2	*style*	*hip-hop*
3	*topic*	*music*
3	*seconds*	*300*
3	*style*	*rock*
4	*topic*	*music*
4	*seconds*	*400*
4	*style*	*classic*

Suponha agora que os parâmetros da Tabela 25 foram declarados, definindo: os metadados que serão comparados (*seconds* e *style*), os tipos de comparação que serão realizados (“>=” e “=”) e os valores de interesse para estes metadados (300 e *classic*).

Tabela 25: Parâmetros a serem utilizados pelo padrão BI

idData	***metadata***	***value***
1	*comparedMetadata*	*seconds*
1	*comparingSignal*	*>=*
1	*comparedValue*	*300*
2	*comparedMetadata*	*style*
2	*comparingSignal*	*=*
2	*comparedValue*	*classic*

Dessa forma, os dados da Tabela 26 seriam descartados, pois não estão de acordo com os metadados e valores definidos.

Tabela 26: Dados descartados

idData	*metadata*	*value*
1	*topic*	*music*
1	*seconds*	180
1	*style*	*classic*
2	*topic*	*music*
2	*seconds*	*200*
2	*style*	*hip-hop*
3	*topic*	*music*
3	*seconds*	*300*
3	*style*	*rock*

Depois de realizado o processamento, o dado da Tabela 27 seria transferido para o repositório interno, pois está de acordo com os critérios de seleção definidos.

Tabela 27: Dados transferidos para o repositório interno

idData	*metadata*	*value*
4	*topic*	*music*
4	*seconds*	*400*
4	*style*	*classic*

5.9 Padrão Eliminador de Dados Proibidos

5.9.1 Motivação

Muitas vezes é necessário eliminar dados já incorporados aos repositórios internos e que mesmo assim possam causar prejuízo a sistemas e redes que os suportam. Dentro dessa categoria existem: os vírus, os cavalos de troia, os dados proibidos (que não possam ou devam ser acessados), os *scripts* maliciosos, etc. Normalmente, na eliminação de dados proibidos procura-se observar as características dos dados que devem ser eliminados, tais como uma assinatura específica escondida num arquivo e que denuncie o software a ser descartado. Este comportamento pode estar presente nos sistemas antivírus, mediação de fóruns, etc.

5.9.2 Aplicabilidade

Eliminar dados considerados proibidos ou que possam causar algum prejuízo a um repositório, um sistema, seus usuários ou a uma organização. Estes dados podem ser eliminados definitivamente ou armazenados em um repositório de descarte, a exemplo da pasta quarentena de alguns sistemas antivírus. É necessária uma fonte de dados contendo textos, assinaturas, palavras-chave e/ou regras que permitem obter dados a serem eliminados. Dessa forma, este padrão possibilita minimizar os efeitos da sobrecarga e poluição de dados nos repositórios internos.

5.9.3 Modelo

A Figura 42 detalha este padrão. Em termos de repositórios, ela apresenta o *Parameters* que representa um repositório de parâmetros, contendo uma lista de dados considerados proibidos e informações adicionais, tais como o nome da função de similaridade a ser usada na identificação dos dados proibidos; o *Internal Repository* que representa o repositório interno à aplicação, podendo conter dados proibidos; e a lixeira (*Trash*). O repositório interno, ao final do processo, conterá somente os dados verificados/autorizados.

Ao receber um evento iniciador do processo (*INITIAL EVENT*), a primeira regra (*DETECT DATA*) verifica a existência de dados no repositório interno, através do arco de leitura e da variável *readData*. Se existirem dados considerados proibidos, o evento *DETECTED DATA* é disparado, o que permite o processamento da segunda regra (*DISCARD PROHIBITED DATA*). Nessa regra, a variável *getData* extrai os dados do repositório interno, enquanto que a variável *readData* lê os dados do repositório de parâmetros. Os dados do repositório interno são comparados com os parâmetros que indicam os dados proibidos e, se existirem similaridades de acordo com a função de similaridade escolhida, os dados proibidos são enviados para a lixeira, através do arco que contém a função *putProhibitedData( ).* A função *putSub( )* transfere os dados não proibidos de volta para o repositório interno.

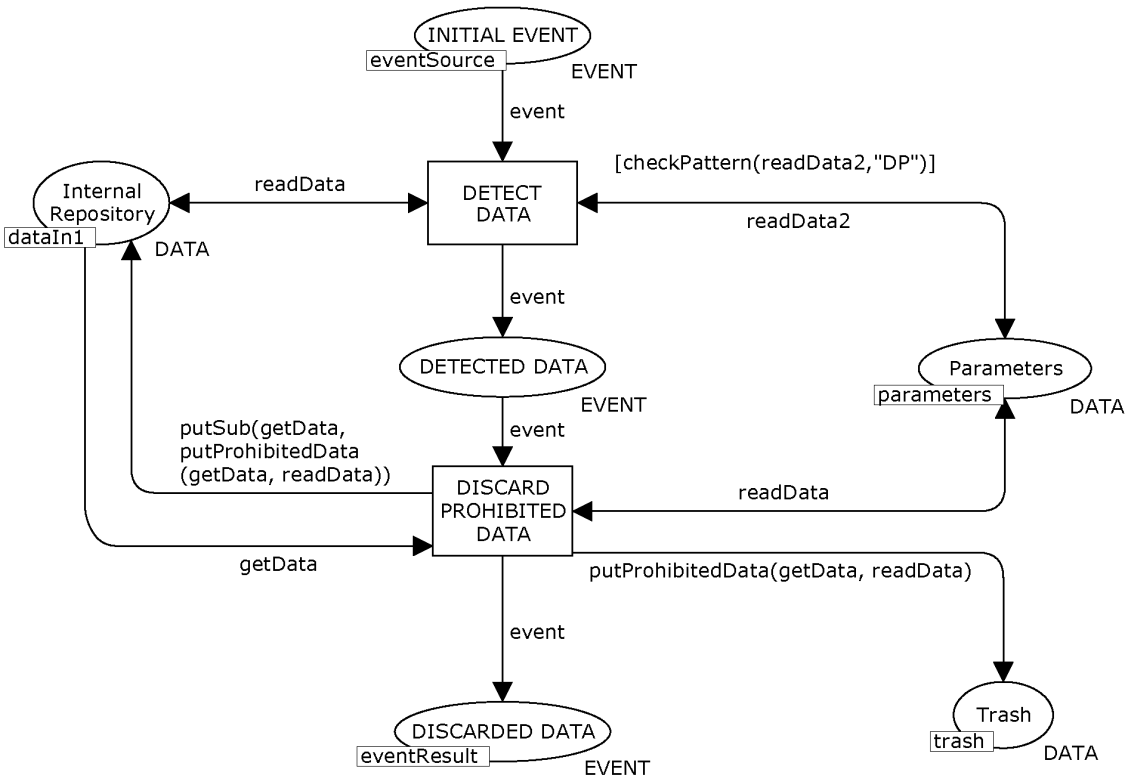

Figura 42 - Eliminador de Dados Proibidos

5.9.4 Exemplo

Suponha que o repositório interno contenha marcas que indicam três instâncias de dados, cada uma composta por dois metadados e valores associados, representando o tópico do dado (*topic*) e o texto associado a este tópico (*text*). Todos estes dados podem conter tópicos proibidos ou não. A Tabela 28 retrata essa situação.

Tabela 28: Dados a serem tratados pelo padrão DP

idData	***metadata***	***value***
1	*topic*	*Terror Movies*
1	*text*	*script of various terror movies*
2	*topic*	*Drama Movies*
2	*text*	*information about drama movies*
3	*topic*	*Sci-fi Movies*
3	*text*	*Top ten sci-fi movies*

Uma opção de função para comparação de textos que também está disponível é a baseada em palavras-chave. Essa função permite, além da escolha das palavras-chave, que seja configurado o número mínimo de palavras-chave a ser encontrado. Por

exemplo, se forem escolhidas cinco palavras-chave e o número mínimo for dois, então todo o texto com no mínimo quaisquer duas das palavras-chave escolhidas será selecionado.

Suponha agora que os parâmetros da Tabela 29 foram declarados, definindo: o metadado que será comparado (*text*), os valores proibidos para este metadado (*terror movies*), a função de comparação a ser usada (Palavras-chave ou *Keyword*) e o número mínimo de palavras-chave que deve ser encontrado para que o valor seja considerado proibido (igual a 2).

Tabela 29: Parâmetros a serem utilizados pelo padrão DP

idData	***metadata***	***value***
1	*comparedMetadata*	*text*
2	*text*	*terror movies*
3	*similarityStrategy*	*Keyword*
4	*similarityCoeficient*	*2*

Dessa forma, o dado da Tabela 30 seria descartado, pois apresenta as duas palavras-chave que definem os dados proibidos.

Tabela 30: Dado descartado

idData	***metadata***	***value***
1	*topic*	*Terror Movies*
1	*text*	*script of various terror movies*

Depois de realizado o processamento, os dados da Tabela 31 permaneceriam no repositório interno:

Tabela 31: Dados mantidos no repositório interno

idData	***metadata***	***value***
2	*topic*	*Drama Movies*
2	*text*	*information about drama movies*
3	*topic*	*Sci-fi Movies*
3	*text*	*Top ten sci-fi movies*

5.10 Padrão Eliminador de Dados Similares

5.10.1 Motivação

Os sistemas de uma organização podem possuir processos de obtenção de dados diferentes. Quando isto ocorre, se não for feita uma verificação cuidadosa, muitos dados similares ou duplicados são armazenados nos repositórios internos dessas organizações. Um dos grandes problemas é identificar os dados que representem o mesmo objeto no mundo real, apesar de apresentar pequenas variações. Algumas vezes, estes dados não podem ser descartados, por causa da arquitetura dos dados ou de regras de negócio da empresa. Nos casos em que o descarte pode ocorrer, sua motivação ocorre por diversos fatores, entre eles: facilitar a manutenção da consistência da base de dados, evitar redundâncias, não permitir o aumento do tamanho do repositório com dados duplicados ou diminuir o tempo de processamento de consultas.

5.10.2 Aplicabilidade

Eliminar dados similares ou duplicados segundo determinadas funções e níveis de similaridade entre estes dados. Estes dados podem ser eliminados definitivamente ou armazenados em um repositório de descarte. A eliminação das duplicações facilita a manutenção da consistência do repositório, pode aumentar a velocidade de processamento dos dados e diminuir o tempo de processamento de consultas. Dessa forma, este padrão possibilita minimizar os efeitos da poluição de dados nos repositórios internos.

5.10.3 Modelo

A Figura 43 detalha este padrão. Em termos de repositórios, ela apresenta o *Parameters* que representa um repositório de parâmetros, contendo informações adicionais, tais como o nome da função de similaridade a ser usada na identificação dos dados similares; o *Internal Repository* que representa o repositório interno à aplicação, podendo conter dados similares; e a lixeira (*Trash*). O repositório interno, ao final do processo, conterá somente os dados verificados/autorizados

Ao receber um evento iniciador do processo (*INITIAL EVENT*), a primeira regra (*DETECT DATA*) verifica a existência de dados no repositório interno, através do arco de leitura e da variável *readData*. Se existirem dados considerados similares, o evento

DETECTED DATA é disparado, o que permite o processamento da segunda regra (*DISCARD SIMILAR DATA*). Nessa regra, a variável *getData* extrai os dados do repositório interno, enquanto que a variável *readData* lê os dados do repositório de parâmetros. Os dados do repositório interno são comparados entre si e, se existirem similaridades de acordo com a função de similaridade escolhida, os dados mais antigos são enviados para a lixeira, através do arco que contém a função *putSimilarData( )*. A função *putSub( )* transfere os dados não similares e mais atuais de volta para o repositório interno.

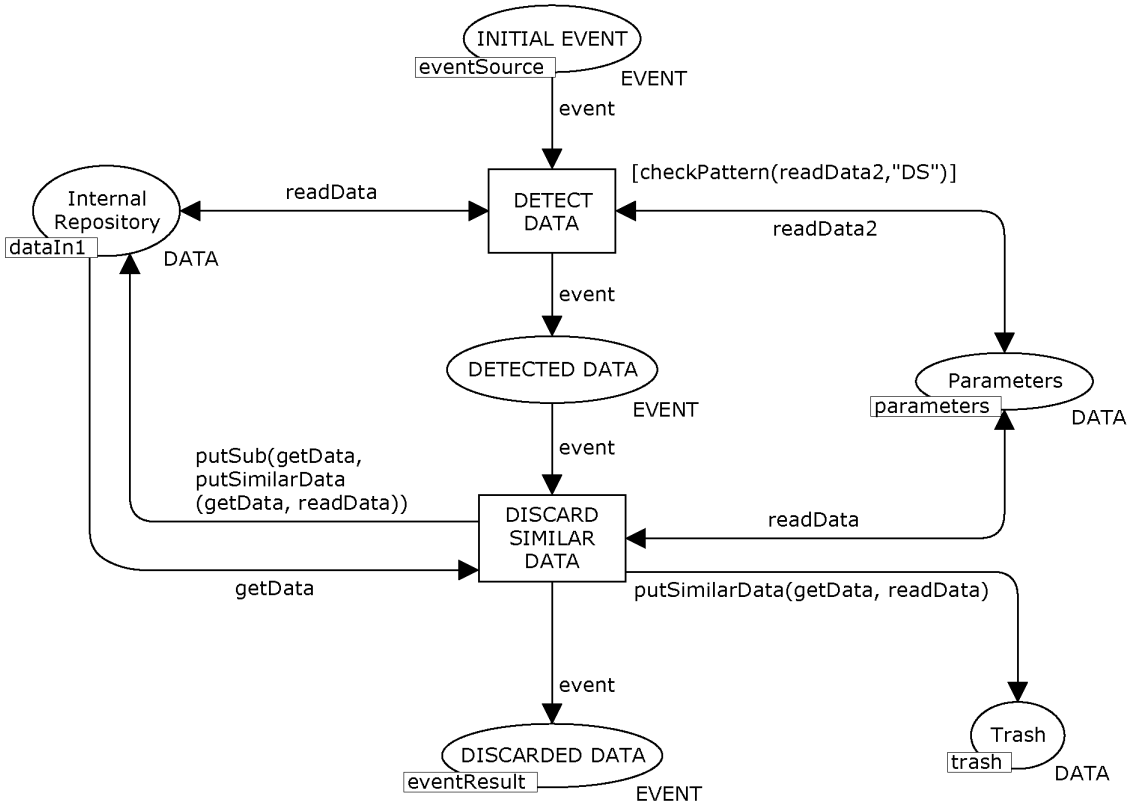

Figura 43 - Eliminador de Dados Similares

5.10.4 Exemplo

Suponha que o repositório interno contenha marcas que indicam três instâncias de dados, cada uma composta por dois metadados e valores associados, representando o código identificador do texto associado ao dado (ISBN) e o sumário associado a esse dado (*summary*). Todos esses dados podem conter códigos e sumários similares ou não. A Tabela 32 retrata essa situação.

Tabela 32: Dados a serem tratados pelo padrão DS

idData	*metadata*	*value*
1	*ISBN*	*01234567*
1	*summary*	*This article discusses Petri Nets and its applications.*
2	*ISBN*	*01234567*
2	*summary*	*This article provides a vision of Petri Nets and its applications.*
3	*ISBN*	*77777777*
3	*summary*	*High-Level Petri Nets are interesting mechanisms to model business process.*

Suponha agora que os parâmetros da Tabela 33 foram declarados, definindo: o metadado que será comparado (*any*, ou seja, todos os metadados serão comparados com os seus correspondentes), a função de comparação a ser usada (Jaccard), o coeficiente de similaridade para que o dado seja considerado similar (50%), a opção de escolha de prioridade para dois dados similares (*higherLength*, sendo que, atualmente, as seguintes outras opções também estão disponíveis: *lowerLength*, *moreRecentDate* e *olderDate*) e o metadado que será utilizado para a escolha de prioridade (*summary*).

Tabela 33: Parâmetros a serem utilizados pelo padrão DS

idData	*metadata*	*value*
1	*comparedMetadata*	*any*
2	*similarityStrategy*	*Jaccard*
3	*similarityCoeficient*	*50*
4	*higherLength*	*summary*

Dessa forma, o dado da Tabela 34 seria descartado, pois apresenta o valor de um dos seus metadados mais de 50% similar ao valor encontrado em outro dado, porém possui menor tamanho do atributo referente ao sumário, sendo que o critério de escolha do dado preferido, em caso de similaridade, privilegia o dado de maior tamanho.

Tabela 34: Dados descartados

idData	*metadata*	*value*
1	*ISBN*	*01234567*
1	*summary*	*This article discusses Petri Nets and its applications.*

Depois de realizado o processamento, o dado da Tabela 35 permaneceria no repositório interno.

Tabela 35: Dado mantido no repositório interno

idData	*metadata*	*value*
2	*ISBN*	*01234567*
2	*summary*	*This article provides a vision of Petri Nets and its applications.*
3	*ISBN*	*77777777*
3	*summary*	*High-Level Petri Nets are interesting mechanisms to model business process.*

5.11 Padrão Eliminador de Dados Obsoletos

5.11.1 Motivação

Alguns dados perdem a sua importância com o passar do tempo. Esses dados sobrecarregam sistemas se não forem eliminados dos repositórios principais. Os dados considerados obsoletos podem ser armazenados em repositórios secundários, tais como os repositórios de dados históricos de armazéns de dados, ou serem simplesmente eliminados. São exemplos desses dados os referentes a registros financeiros para fins legais e de auditoria que, depois de certo tempo, perdem a relevância em função de limite de tempos estabelecidos em leis, regulamentos ou normas. É importante considerar que alguns dados, mesmo com o passar do tempo, não perdem a importância, devendo ser mantidos nos repositórios.

5.11.2 Aplicabilidade

Eliminar dados que não tenham mais valor em função da passagem do tempo e do seu período de validade intrínseco. Esses dados, conhecidos como dados obsoletos, podem ser descartados definitivamente ou armazenados em um repositório secundário. Podem ser usadas regras adicionais que estabeleçam os critérios de tempo para identificação dos dados obsoletos por determinada área ou assunto. Dessa forma, os efeitos da sobrecarga e poluição de dados nos repositórios internos podem ser minimizados.

5.11.3 Modelo

A Figura 44 detalha este padrão. Em termos de repositórios, ela apresenta o *Parameters* que representa um repositório de parâmetros, contendo informações adicionais, tais como o limite de tempo que caracteriza um dado como obsoleto; o *Internal Repository* que representa o repositório interno à aplicação, podendo conter dados obsoletos; e a lixeira (*Trash*). O repositório interno, ao final do processo, conterá somente os dados verificados/autorizados.

Ao receber um evento iniciador do processo (*INITIAL EVENT*), a primeira regra (*DETECT DATA*) verifica a existência de dados no repositório interno, através do arco de leitura e da variável *readData*. Se existirem dados considerados obsoletos, o evento *DETECTED DATA* é disparado, o que permite o processamento da segunda regra (*DISCARD OBSOLETE DATA*). Nessa regra, a variável *getData* extrai os dados do repositório interno, enquanto que a variável *readData* lê os dados do repositório de parâmetros. Os dados do repositório interno são processados de acordo com os parâmetros e, se forem considerados obsoletos, são enviados para a lixeira, através do arco que contém a função *putOldData( ).* A função *putSub( )* transfere os dados não obsoletos de volta para o repositório interno.

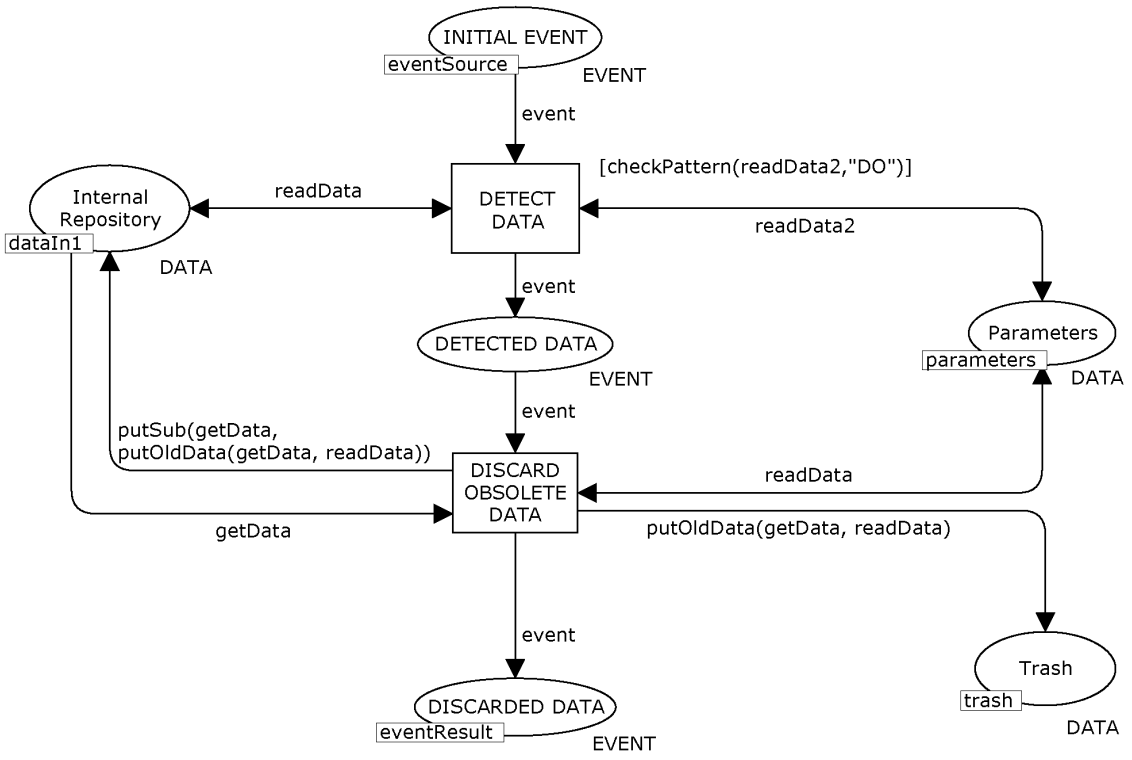

Figura 44 - Eliminador de Dados Obsoletos

5.11.4 Exemplo

Suponha que o repositório interno contenha marcas que indicam três instâncias de dados, cada uma composta por dois metadados e valores associados, representando o tópico do dado (*topic*) e a etiqueta de tempo associada a este dado (*dataTimestamp*). A Tabela 36 retrata essa situação.

Tabela 36: Dados a serem tratados pelo padrão DO

idData	***metadata***	***value***
1	*topic*	*Database*
1	*dataTimestamp*	*20*
2	*topic*	*Advertising*
2	*dataTimestamp*	*10*
3	*topic*	*Petri Nets*
3	*dataTimestamp*	*40*
4	*topic*	*Workflow*
4	*dataTimestamp*	*30*

Suponha agora que os parâmetros da Tabela 37 foram declarados, definindo: o metadado utilizado para ordenar o resultado de saída (ex.: *orderMetadata* com *value* = *dataTimestamp*), a ordem em que os resultados serão apresentados de acordo com o metadado de ordenação (ex.: *sort* com *value* = *ascendent* – do menor para o maior, ou *descendent* – do maior para o menor), o metadado que será comparado (*dataTimestamp*), a etiqueta de tempo que representa o instante atual (*currentTimestamp*) e o período de tempo que define um dado como obsoleto se contado a partir do instante atual (*periodTimestamp*).

Tabela 37: Parâmetros a serem utilizados pelo padrão DO

idData	***metadata***	***value***
1	*orderMetadata*	*dataTimestamp*
2	*sort*	*ascendent*
3	*comparedMetadata*	*dataTimestamp*
4	*currentTimestamp*	*40*
5	*periodTimestamp*	*10*

Dessa forma, os dados da Tabela 38 seriam descartados, pois possuem uma etiqueta de tempo mais antiga que a etiqueta do tempo atual menos o período estipulado.

Tabela 38: Dados descartados

idData	***metadata***	***value***
2	*topic*	*Advertising*
2	*dataTimestamp*	*10*
1	*topic*	*Database*
1	*dataTimestamp*	*20*

Depois de realizado o processamento, os dados da Tabela 39 permaneceriam no repositório interno.

Tabela 39: Dados mantidos no repositório interno

idData	***metadata***	***value***
4	*topic*	*Workflow*
4	*dataTimestamp*	*30*
3	*topic*	*Petri Nets*
3	*dataTimestamp*	*40*

5.12 Padrão Eliminador de Dados Irrelevantes

5.12.1 Motivação

Atualmente, é comum que uma grande quantidade de informação deva ser armazenada, sendo que apenas uma pequena quantidade é realmente relevante. Neste caso, é interessante eliminar os dados irrelevantes já armazenados, melhorando a qualidade desses repositórios. A seleção dos dados relevantes pode ser feita por meio da identificação dos valores de interesse dos metadados associados aos dados (por exemplo, frequência de acesso maior que um determinado número ou especificar o autor do dado). Neste contexto, existem muitos dados, mas poucos são relevantes, sendo que a maior parte pode ser descartada. É importante considerar que existem situações onde o espaço de armazenamento é uma função crítica em virtude da escassez de recursos para aquisição de repositórios de armazenamento e/ou da imensa quantidade de dados armazenados rotineiramente.

5.12.2 Aplicabilidade

Eliminar dados que não tenham valor em função da necessidade de um sistema, organização ou das pessoas que o utilizam. Estes dados podem ser descartados definitivamente ou armazenados em um repositório de armazenamento secundário. Adicionalmente, podem ser usadas regras que estabeleçam os critérios de usabilidade por área ou assunto de modo a permitir a identificação desses dados. Estas regras são baseadas, normalmente, na seleção de valores e intervalos que indicam os dados de interesse. Dessa forma, este padrão possibilita minimizar os efeitos da sobrecarga e poluição de dados nos repositórios internos.

5.12.3 Modelo

A Figura 45 detalha este padrão. Em termos de repositórios, ela apresenta o *Parameters* que representa um repositório de parâmetros, contendo informações adicionais, tais como as regras que definem a frequência mínima de uso que caracteriza um dado como não relevante em função do pouco uso; o *Internal Repository* que representa o repositório interno à aplicação, podendo conter dados não relevantes; e a lixeira (*Trash*). O repositório interno, ao final do processo, conterá somente os dados verificados/autorizados.

Ao receber um evento iniciador do processo (*INITIAL EVENT*), a primeira regra (*DETECT DATA*) verifica a existência de dados no repositório interno, através do arco de leitura e da variável *readData*. Se existirem dados considerados não relevantes, o evento *DETECTED DATA* é disparado, o que permite o processamento da segunda regra (*DISCARD IRRELEVANT DATA*). Nessa regra, a variável *getData* extrai os dados do repositório interno, enquanto que a variável *readData* lê os dados do repositório de parâmetros. Os dados do repositório interno são processados de acordo com os parâmetros e, se forem considerados relevantes, são enviados para o repositório interno, através do arco que contém a função *putRelevantData( ).* A função *putSub( )* transfere os dados não relevantes para a lixeira (*Trash*).

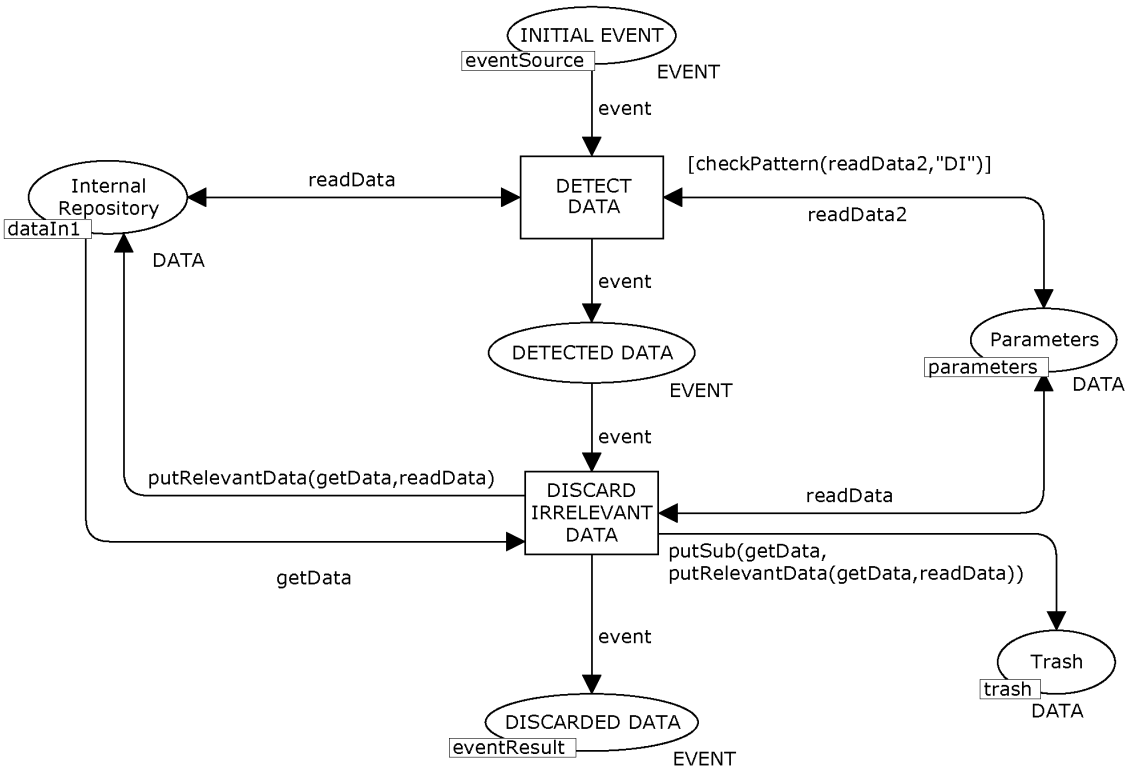

Figura 45 - Eliminador de Dados Irrelevantes

5.12.4 Exemplo

Suponha que o repositório interno contenha marcas que indicam quatro instâncias de dados, cada uma composta por três metadados e valores associados, representando o tópico do dado (*topic*), a frequência de acesso associada a este dado (*accessFrequency*) e o autor do dado (*author*). A Tabela 40 retrata essa situação.

Tabela 40: Dados a serem tratados pelo padrão DI

idData	***metadata***	***value***
1	*topic*	*Advertising*
1	*accessFrequency*	*10*
1	*author*	*John*
2	*topic*	*Database*
2	*accessFrequency*	*20*
2	*author*	*Bill*
3	*topic*	*Workflow*
3	*accessFrequency*	*30*
3	*author*	*Bill*
4	*topic*	*Petri Nets*
4	*accessFrequency*	*40*
4	*author*	*John*
4	*author*	*Bill*

Suponha agora que os parâmetros da Tabela 41 foram declarados, definindo as características dos metadados: frequência de acesso (*accessFrequency*) e autor (*author*). Os valores desses metadados (30 e Bill) são comparados por meio de operadores lógicos (“>=” e “=”) e a necessidade de ocorrência simultânea é definida pela expressão “*AND*” (ou “*OR*” quando a simultaneidade não for necessária) associada ao metadado *connective*.

Tabela 41: Parâmetros a serem utilizados pelo padrão DI

idData	***metadata***	***value***
1	*connective*	*AND*
2	*comparedMetadata*	*accessFrequency*
2	*comparingSignal*	*>=*
2	*comparedValue*	30
3	*comparedMetadata*	*author*
3	*comparingSignal*	*=*
3	*comparedValue*	*Bill*

Dessa forma, os dados da Tabela 42 seriam descartados, pois possuem uma frequência de acesso menor que a frequência de acesso mínima estipulada ou autor diferente do autor determinado nos parâmetros.

Tabela 42: Dados descartados

idData	***metadata***	***value***
1	*topic*	*Advertising*
1	*accessFrequency*	*10*
1	*author*	*John*
2	*topic*	*Database*
2	*accessFrequency*	*20*
2	*author*	*Bill*

Depois de realizado o processamento, os dados da Tabela 43 permaneceriam no repositório interno.

Tabela 43: Dados mantidos no repositório interno

idData	***metadata***	***value***
3	*topic*	*Workflow*
3	*accessFrequency*	*30*
3	*author*	*Bill*
4	*topic*	*Petri Nets*
4	*accessFrequency*	*40*
4	*author*	*John*
4	*author*	*Bill*

5.13 Álgebra de Eliminação de Dados

Os padrões apresentados anteriormente podem ser utilizados isoladamente ou de forma combinada de modo a atender uma vasta gama de aplicações. Entretanto, a utilização direta dos padrões modelados em redes de Petri exige o domínio de seus conceitos e propriedades, o que pode demandar algum tempo.

No sentido de facilitar o uso dos processos modelados em redes de Petri, propõe-se uma interface alternativa para esses processos por meio do uso do **SGBD SECONDO**. Este SGBD permite a criação de álgebras não-convencionais, permitindo associar operadores a processos por meio da criação de uma álgebra. Nesta álgebra,

cada operador comunica-se com um padrão em RPAN, modelado pela ferramenta CPN Tools. Dessa forma, cada operador da álgebra corresponde a uma interface para um padrão.

Assim, podem ser destacadas como vantagens dessa estratégia:

- A criação de uma interface alternativa para acesso aos padrões de eliminação de dados, através de uma álgebra customizada que representa cada um dos padrões como um operador;
- A possibilidade de armazenar o histórico dos eventos ocorridos no próprio SGBD;
- A possibilidade de armazenar os dados descartados para análises futuras;
- Facilidade na combinação dos operadores de modo a fornecer operações mais complexas.

Um diferencial importante dos operadores criados neste trabalho em relação aos operadores tradicionais da álgebra relacional é a possibilidade de se utilizar funções de similaridade para comparação com os atributos ou metadados dos dados, um recurso considerado importante principalmente para domínios de dados complexos (ARANTES et al., 2003). A Tabela 44 apresenta a álgebra proposta usando uma sintaxe genérica em BNF.

Para manter a coerência com a convenção de nomes utilizada nos padrões, optou-se por descrever os símbolos dessa álgebra também em inglês. Na Tabela 44, os símbolos não terminais são envoltos por "<" e ">", os símbolos terminais não estão envolvidos por estes sinais. Excetuam-se os símbolos <*integer*> e <*text*> que expressam seus próprios significados e, por conta disso, correspondem a símbolos terminais nesta representação.

Diversos parâmetros, definidos na álgebra, podem ser fornecidos, dentre eles:

- Definição da função de similaridade e do nível de similaridade. Atualmente, é possível usar um conjunto de palavras-chave ou as funções de similaridade de Jaccard e Dice (BAEZA-YATES & RIBEIRO-NETO, 1999). Outras medidas de similaridade podem ser incluídas;
- Comparação das palavras originais ou dos radicais da palavra (*stemming*) (PORTER, 1997);

- Retirada de *stopwords*;
- Escolha da ordenação dos resultados (ex.: por data, por tamanho, por nível de similaridade com as palavras fornecidas, etc);
- Escolha do idioma utilizado (ex.: inglês, português, etc.).

Na Tabela 44, os símbolos terminais dos operadores são identificados por:

- *ad* – correspondente ao padrão Dados Permitidos;
- *bp* – correspondente ao padrão Bloqueador de Dados Proibidos;
- *bs* – correspondente ao padrão Bloqueador de Dados Similares;
- *bi* – correspondente ao padrão Bloqueador de Dados Irrelevantes;
- *dp* – correspondente ao padrão Eliminador de Dados Proibidos;
- *ds* – correspondente ao padrão Eliminador de Dados Similares;
- *do* – correspondente ao padrão Eliminador de Dados Obsoletos;
- *di* – correspondente ao padrão Eliminador de Dados Irrelevantes.

Tabela 44: Álgebra de eliminação de dados

Sintaxe e símbolos dos operadores em BNF
<command>::=(<operator><command>)|(<operator>(<data>))
<data>::=(<integer><metadata><text>)<data>
|(<integer><metadata><text>)
<metadata>::=<text>
<bool>::=true|false
<operator>::=bp(<parametersAllowedProhibited>)|dp(<parametersAllowedProhibited>)
|ad(<parametersAllowedProhibited
|bs(<parametersSimilar><data>)|ds(<parametersSimilar>)
|bi(<parametersIrrelevant>)|di(<parametersIrrelevant>)|do(<parametersObsolete>)
<parametersAllowedProhibited>::=(order=dateTimestamp|similarity|length/none)
(sort=ascendent|descendent)
(comparedMetadata=<metadata>)
<metadataValue><similarityParameters><dependenceDataStrategy>
<parametersSimilar>::=(order= dateTimestamp|length|none)
(sort=ascendent|descendent)
(comparedMetadata=<metadata>)
<similarityParameters><dependenceDataStrategy>
<parametersIrrelevant>::=(order= dateTimestamp|length|none)
(sort=ascendent|descendent)
(connective=OR|AND)
<conditions><dependenceDataStrategy>
<parametersObsolete>::=(order=dateTimestamp|length|none)
(sort=ascendent|descendent)
(comparedMetadata=<metadata>)
(currentTimestamp=<text>)
(periodTimestamp=<text>)<dependenceDataStrategy>
<metadataValue>::=(metadata=<metadata>)<metadataValue>
|(metadata=<metadata>)
<similarityParameters>::=(similarityStrategy=Jaccard|Dice\Keyword)
(similarityCoeficient=<integer>)
(stemming=<bool>)(stopwords=<bool>)
(language=english|portuguese)
<conditions>::=(comparedMetadata=<metadata>)(comparingSignal=<signal>)
(comparedValue=<text>)<conditions>
|(comparedMetadata=<metadata>)(comparingSignal=<signal>)
(comparedValue=<text>)
<dependenceDataStrategy>::=(dependentDataStrategy=discard all|keep dependents|keep all)
<dependentDataRelationship>
<dependentDataRelationship>::=(dependentDataRelationship=<metadata>)
|(dependentDataRelationship=<metadata>) <dependentDataRelationship>
<signal>::=|<|>|<=|>=

5.13.1 SGBD SECONDO

O SECONDO[42] é um SGBD extensível capaz de suportar aplicações não convencionais. Está sendo desenvolvido na Universidade de Fern, em Hagen, Alemanha. Ele permite criar ou modificar álgebras existentes e também fornece um modelo de SGBD genérico que pode ser estendido com implementações de vários tipos de dados.

[42] http://dna.fernuni-hagen.de/Secondo.html/

A ideia original do SECONDO foi descrita por GUTING (1993). A primeira versão (DIEKER & GUTTING, 2000) foi construída entre os anos de 1995 e 2001 e utilizava o sistema SHORE (*Scalable Heterogeneous Object REpository*) (CAREY et al., 1994) como gerenciador de armazenamento, sendo compatível com o sistema operacional Solaris[43]. A versão atual do sistema (GÜTING et al., 2004a, 2004b, GÜTING et al., 2005) usa o Berkeley DB[44] como gerenciador de armazenamento e é compatível com os sistemas operacionais Windows[45], Linux[46], Solaris e Mac OS X[47].

A definição das álgebras está baseada no conceito de assinaturas de segunda ordem (*Second-Order Signature* – SOS) (GÜTING et al., 1993). A ideia é usar duas assinaturas acopladas. A primeira assinatura descreve os tipos construtores. A segunda assinatura descreve as operações que são aplicadas aos tipos da primeira assinatura. Dessa forma, um módulo da álgebra fornece uma coleção de tipos construtores, assim como operadores que atuam sobre os tipos criados.

O SECONDO possui três componentes principais, que podem ser usados juntos ou separados. São eles:

- Núcleo (*Kernel*) – escrito em C++, implementa modelos de dados específicos, é extensível com o uso de módulos de álgebra e possibilita o processamento de consultas sobre as álgebras implementadas;

- Otimizador – escrito em PROLOG[48], tem como função principal a otimização das consultas. O algoritmo usa estimativas de custo para produzir bons planos. Adicionalmente, otimiza a parte essencial da linguagem SQL usando uma notação adaptada ao PROLOG;

- Javagui – escrita em Java, a Interface Gráfica do Usuário (*Graphic User Interface* - GUI), é uma interface extensível a novos tipos de dados e modelos, perfeito para Sistemas de Gerencia de Bancos de Dados extensíveis como o SECONDO. Além disso, há um visualizador especial disponível na interface para tipos espaciais e objetos em movimento, possibilitando uma interface para bancos de dados genérica e sofisticada, incluindo animação para objetos em movimento.

[43] http://www.sun.com/software/solaris/
[44] http://www.oracle.com/technology/products/berkeley-db/index.html
[45] http://www.microsoft.com/en/us/default.aspx?pf=true&group=Windows
[46] http://www.linux.org/
[47] http://www.apple.com/macosx/
[48] http://www.swi-prolog.org/

O SECONDO fornece, além da Javagui, quatro diferentes interfaces com o usuário, a saber:

- SecondoTTYBDB – Interface textual para modo monousuário;
- SecondoTTYCS – Versão multiusuário do SecondoTTYBDB;
- SecondoPL – Versão monousuário do otimizador;
- SecondoPLCS – Versão multiusuário do otimizador;

Atualmente, o SECONDO disponibiliza um conjunto de trinta álgebras, dentro das quais se destacam:

- Álgebra Padrão – fornece tipos simples como inteiros, reais, booleanos e strings junto com suas operações;
- Álgebra Relacional – fornece tipos e operações para realizar relacionamentos clássicos em SGBDs;
- Álgebra Espacial – representa objetos geométricos como pontos, linhas e regiões junto com operações espaciais;
- Álgebra Temporal – representa objetos em movimento no BD. Ex: Pessoas ou veículos. Possibilita que sejam formuladas consultas sobre os movimentos desses objetos.

Neste trabalho, é definida uma nova álgebra especificamente para eliminação de dados. Esta álgebra é implementada no SECONDO e cada um dos seus operadores se comunica com um padrão modelado em RPAN. Por conta disso, um operador no SECONDO pode ser considerado uma interface para um padrão. O uso do SECONDO como interface oferece algumas vantagens, dentre elas:

- Armazenamento temporário do fluxo de dados;
- Armazenamento de dados descartados para posterior análise;
- Armazenamento do histórico dos eventos ocorridos;
- Criação de uma linguagem formal através de uma álgebra personalizada que representa os padrões (operadores) de eliminação de dados.

Assim, o SECONDO fornece uma forma alternativa de acesso aos padrões modelados em RPAN através de uma álgebra especialmente criada para esse fim. Ao

mesmo tempo, procura-se aproveitar algumas das vantagens das duas ferramentas, buscando facilitar o processamento de grande volume de dados. A ideia é que o SGBD receba o conjunto de dados e só envie para as redes de Petri os dados necessários ao processamento dos padrões.

Para serem processados pelo SECONDO, os dados e parâmetros dos operadores devem ser transformados para o formato da tripla (*<identificador><metadado><valor>)*, a exemplo do que ocorre com os dados e parâmetros passados para as RPAN da ferramenta CPN *Tools*, como descrito anteriormente.

Considerando este formato, caso seja necessário representar os dados de duas pessoas: Brian com 33 anos e Joe com 40 anos, obtém-se:

<01><name><Brian>
<01><age><33>
<02><name><Joe>
<02><age><40>

Internamente no Secondo, essas triplas podem ser representadas de duas formas:

- Como uma relação contendo os tipos inteiro, text e texto, no formato *rel (tuple ((idData int)(metadata text)(value text)))*, sendo que o tipo *text*, nativo do SECONDO, é limitado em relações ao tamanho de 128 bytes. Por exemplo, os dados apresentados anteriormente seriam representados, usando esta estratégia, pela seguinte expressão:

rel (tuple ((idData int) (metadata text) (value text)))
((1 <text>name</text---> <text>Brian</text--->)
(1 <text>age</text---> <text>33</text--->)
(2 <text>name</text---> <text>Joe</text--->)
(2 <text>age</text---> <text>44</text--->))

- Como um novo tipo de dado, chamado *XDATA*, criado para suportar a tripla *<identificador><metadado><valor>* e representado simplesmente por *<xdata>*. Este novo tipo fornece uma representação mais compacta da tripla e não possui limite de tamanho para os atributos metadado e valor. Por exemplo, os dados apresentados previamente seriam representados, usando esta estratégia, pela seguinte expressão:

rel (tuple ((dat xdata)))
(((1 <text>name</text---> <text>Brian</text--->))
((1 <text>age</text---> <text>33</text--->))
((2 <text>name</text---> <text>Joe</text--->))
((2 <text>age</text---> <text>44</text--->)))

Todos os operadores de eliminação de dados discutidos nesta tese aceitam os dois formatos de dados apresentados anteriormente.

Considerando os padrões de eliminação de dados já apresentados, os tipos de dados aceitos pelo SECONDO e a álgebra genérica apresentada na Tabela 44, foi construída uma álgebra utilizando a sintaxe do SECONDO. Esta álgebra é apresentada na Tabela 45 e a sua implementação, feita no SGBD SECONDO, pode ser encontrada no Anexo II. Os símbolos não terminais são envoltos por "<" e ">", os símbolos terminais não estão envolvidos por estes sinais. Excetuam-se os símbolos <*integer*>, <*string*> que expressam seus próprios significados e, por conta disso, correspondem a símbolos terminais nesta representação. Além disso, os símbolos <*text*> e <*/text*> também são não terminais e representam as sequências de caracteres "<*text*>" e "<*/text*>", respectivamente.

A álgebra apresentada na Tabela 45 segue a notação com listas aninhadas do SECONDO (GÜTING et al., 2004a, 2004b). A representação textual de uma lista aninhada consiste de um número arbitrário de elementos delimitados por parênteses. Os elementos de uma lista podem ser outras listas aninhadas ou elementos atômicos como números, símbolos, etc. A expressão *(a b ((c) (d e)))* representa uma lista de 3 elementos, sendo a o primeiro, *b* o segundo e *((c) (d e))* o terceiro. Este elemento é também uma lista aninhada composta de duas listas, cujo primeiro elemento é um elemento atômico *c* e o segundo elemento é uma lista composta por dois elementos *d, e*. Alternativamente, o SECONDO oferece uma sintaxe, chamada sintaxe textual, que, dependendo da álgebra implementada, pode facilitar a escrita dos operadores, pois não exige o uso de parênteses. Entretanto, esta sintaxe alternativa é internamente transformada na sintaxe com listas aninhadas, sendo que os erros informados pelo SECONDO são relativos a sintaxe com listas aninhadas.

Adicionalmente, de modo a permitir a consulta do tipo <xdata> e prover flexibilidade no uso dos operadores, foram definidos dois operadores auxiliares: *showxdata* (sintaxe: *showxdata* <*xdataTuple*>) e *conc* (sintaxe: *conc* <*data1*> <*data2*>). O primeiro operador simplesmente lê uma tupla do dado representado no formato <*xdataTuple*> (Ex.: *(xdata (1 <text>test</text---> <text>test</text--->))*) e o mostra na interface do SECONDO. O segundo operador une triplas de dados em

qualquer dos formatos apresentados anteriormente, eliminando duplicatas, ou seja, realiza a disjunção de dois conjuntos de dados.

Tabela 45: Álgebra de eliminação de dados adaptada ao SECONDO

Sintaxe e símbolos dos operadores em BNF seguindo a sintaxe do SECONDO
<command>::=query <operators> \n *<operators>::=(<operator><operators>)|(<operator>(<xdata>))* *<xdata>::= <xdataType><xdataInstance>* *<xdataType>::=(rel(tuple(<string> xdata)))* *<xdataInstance>::=(<integer><text><string></text><text><string></text>)<xdataInstance>|* *(<integer><text><string></text><text><string></text>)* *<bool>::=true|false* *<operator>::=bp(<parametersAllowedProhibited>)|dp(<parametersAllowedProhibited>)* *|ad(<parametersAllowedProhibited>)* *|bs(<parametersSimilar><data>)|ds(<parametersSimilar>)* *|bi(<parametersIrrelevant>)|di(<parametersIrrelevant>)|do(<parametersObsolete>)* *<parametersAllowedProhibited>::=<xdataType>* *(<integer><text>order</text><text>dateTimespamp|similarity|length/none</text>)* *(<integer><text>sort</text><text>ascendent|descendent </text>)* *(<integer><text>comparedMetadata</text><text><string> </text>)* *<metadataValue><similarityParameters><dependenceDataStrategy>* *<parametersSimilar>::=<xdataType>* *(<integer><text>order</text><text>dateTimestamp|length/none</text>)* *(<integer><text>sort</text><text>ascendent|descendent</text>)* *(<integer><text>comparedMetadata</text><text><string> </text>)* *<similarityParameters><dependenceDataStrategy>* *<parametersIrrelevant>::=<xdataType>* *(<integer><text>order</text><text>dateTimestamp|length/none</text>)* *(<integer><text>sort</text><text>ascendent|descendent</text>)* *(<integer><text>connective</text><text>OR|AND</text>)* *<conditions><dependenceDataStrategy>* *<parametersObsolete>::=<xdataType>* *(<integer><text>order</text><text>dateTimestamp|length/none</text>)* *(<integer><text>sort</text><text>ascendent|descendent </text>)* *(<integer><text>comparedMetadata</text><text><string></text>)* *(<integer><text>currentTimestamp</text><text><string></text>)* *(<integer><text>periodTimestamp</text><text><string></text>)* *<dependenceDataStrategy>* *<metadataValue>::=(<integer><text>metadata</text><text><string></text>)<metadataValue>* *|(<integer><text>metadata</text><text><string></text>)* *< similarityParameters>::=* *(<integer><text>similarityStrategy</text><text>Jaccard|Dice\Keyword</text>)* *(<integer><text>similarityCoeficient</text><text><integer></text>)* *(<integer><text>stemming</text><text><bool></text>)* *(<integer><text>stopwords</text><text><bool></text>)* *(<integer><text>language</text><text>english|portuguese</text>)* *<conditions>::=(<integer><text>comparedMetadata</text><text><string></text>)* *(<integer><text>comparingSignal</text><text>=|<|>|<=|>=</text>)* *(<integer><text>comparedValue</text><text><string></text>)<conditions>* *|(<integer><text>comparedMetadata</text><text><string></text>)* *(<integer><text>comparingSignal</text><text>=|<|>|<=|>=</text>)* *(<integer><text>comparedValue</text><text><string></text>)* *<dependenceDataStrategy>::=(<integer><text>dependentDataStrategy</text>* *<text>discard all|keep dependents|keep all</text>)* *<dependentDataRelationship>* *<dependentDataRelationship>::=<dependentDataRelationship>* *(<integer><text>dependentDataRelationship</text><text><string></text>)* *|(<integer><text>dependentDataRelationship</text><text><string></text>)*

É importante considerar que a conjunção dos resultados dos operadores sobre um mesmo conjunto de dados pode ser realizada pela aplicação em cascata dos operadores, conforme mostrado na álgebra. Um exemplo de aplicação desta álgebra é dado a seguir.

Considerando a expressão **<*dataSecondo*>** e os parâmetros indicados pelas expressões **<*parametersAD*>** e **<*parametersDP*>**, todos no formato XDATA, representados a seguir:

- **<*dataSecondo*>**:

(rel(tuple(dat xdata)))
((1<text>topic</text---><text>Terror Movies</text--->)
(1<text>text</text---><text>script of various terror movies</text--->)
(2<text>topic</text---><text>Drama Movies</text--->)
(2<text>text</text---><text>information about drama movies</text--->)
(3<text>topic</text---><text>Sci-fi Series</text--->)
(3<text>text</text---><text>Top ten sci-fi series</text--->)
(4<text>topic</text---><text>Soap opera</text--->)
(4<text>text</text---><text>Ten Lifes</text--->))

- **<*parametersAD*>**:

(rel(tuple(dat xdata)))
((1<text>comparedMetadata</text---><text>topic</text--->)
(2<text>topic</text---><text>Movies and Series</text--->)
(3<text>similarityStrategy</text---><text>Jaccard</text--->)
(4<text>similarityCoeficient</text---><text>10</text--->)
(5<text>stemming</text---><text>true</text--->)
(6<text>stopwords</text---><text>true</text--->)
(7<text>language</text---><text>english</text--->))

- **<*parametersDP*>**:

(rel(tuple(dat xdata)))
((1 <text>comparedMetadata</text---><text>text</text--->)
(2 <text>text</text---> <text>script of terror movies</text--->)
(3 <text>similarityStrategy</text---><text>Jaccard</text--->)
(4 <text>similarityCoeficient</text---><text>50</text--->)
(5 <text>stemming</text---><text>true</text--->)
(6 <text>stopwords</text---> <text>true</text--->)
(7 <text>language</text---><text>english</text--->))

<parametersAD> representa os parâmetros do operador AD. Eles detalham que esse operador buscará nos valores do metadado "*topic*" textos similares a "*Movies and*

Series” usando a função de Jaccard a 10%. Esta expressão também detalha que o texto será analisado em inglês e deve ser pré-processado, realizando *stemming* e retirada de *stopwords*. Pode-se observar que, considerando os parâmetros fornecidos, somente o dado referente à novela (*soap opera*) será eliminado.

<parametersDP> representa os parâmetros do operador DP. Eles detalham que esse operador buscará nos valores do metadado *text* textos similares a “s*cript of terror movies*” usando a função de Jaccard a 50%. Esta expressão também detalha que o texto será analisado em inglês e deve ser pré-processado, realizando *stemming* e retirada de *stopwords*. Pode-se observar que, considerando os parâmetros fornecidos, somente o dado referente a filmes de terror (*script of terror movies)* será eliminado.

A seguinte expressão representa a aplicação em cascata dos operadores AD e DP, descritos anteriormente:

```
query(
   dp<parametersDP>(
      ad<parametersAD>(<dataSecondo>)))
```

O resultado desta operação retorna o resultado dado pela expressão:

```
(rel(tuple(dat xdata)))
((2<text>topic</text---><text>Drama Movies</text--->)
(2<text>text</text---><text>information about drama movies</text--->)
(3<text>topic</text---><text>Sci-fi Series</text--->)
(3<text>text</text---><text>Top ten sci-fi series</text--->))
```

Vale fazer neste ponto um comentário sobre a ordenação baseada em similaridade apresentada como uma das opções de ordenação da álgebra definida anteriormente. Esta ordenação utiliza o nível de similaridade obtido a partir da comparação dos dados com os termos existentes no parâmetro *<metadataValue>*. Este tipo de ordenação só é possível para os operadores *ad*, *dp* e *bp*. Nestes três operadores, a ordenação pode ser feita do dado mais similar para o menos similar ou vice-versa. Entretanto, em sua operação normal, o operador *ad* descarta os dados menos similares e mantém os mais similares de acordo com um limite de similaridade previamente escolhido. Seguindo este raciocínio, é intuitivo dizer que, quando for usado este tipo de ordenação, o operador *ad* demanda a ordenação do mais similar para o menos similar. Raciocínio inverso pode ser aplicado aos operadores *bp* e *dp*. Dessa forma, é intuitivo dizer que estes operadores demandam a ordenação do menos similar para o mais similar.

5.13.2 Considerações sobre o Comportamento dos Operadores

Neste ponto, vale fazer algumas considerações sobre o comportamento dos operadores: suas similaridades e diferenças.

Em relação à diferença entre os operadores que atuam sobre dados proibidos (BP e DP) e os que atuam sobre dados irrelevantes (BI e DI), pode-se observar que os primeiros utilizam funções de similaridade como uma das opções de seus parâmetros. Isto fornece certa flexibilidade a estes operadores, permitindo a identificação dos dados proibidos pela proximidade com os parâmetros fornecidos. Os padrões sobre dados proibidos, origem dos operadores sobre dados proibidos, partem do princípio de que são conhecidas algumas das características desses dados, porém as mesmas podem variar, mantendo alguma similaridade com as características buscadas (ex.: assinaturas de softwares maliciosos, textos que caracterizam dados proibidos, etc.). Por isso, as funções de similaridade se revestem de grande importância nestes casos. No caso dos operadores que atuam sobre dados irrelevantes, não há uso de funções de similaridade, sendo que a identificação dos dados relevantes é feita por meio da comparação de valores estabelecidos nos parâmetros e nos dados por meio de operadores lógicos (>,<,>=,<= e =). Os padrões sobre dados irrelevantes, origem dos operadores sobre dados irrelevantes, partem do princípio que são conhecidas exatamente as características desses dados, quais sejam: valores, faixa de valores e/ou atributos. Por isso não fazem uso de funções de similaridade.

Outra diferença que merece destaque é a existente entre os operadores baseados nos padrões BS e DS. O primeiro recebe dois conjuntos de dados referentes aos dados dos repositórios externo e interno, buscando eliminar as similaridades entre esses dados. O segundo atua somente sobre o conjunto de dados referentes ao repositório interno, buscando por similaridades entre os dados deste repositório.

Apesar de semanticamente diferentes por atuarem sobre repositórios diferentes, os padrões BP e DP geram operadores que possuem comportamento bastante similar, se for desconsiderada a diferença entre repositório externo e interno. Ou seja, quando se considera os mesmos parâmetros e conjunto de dados de entrada (não diferenciando os repositórios internos e externos) o mesmo conjunto de dados proibidos será eliminado, independentemente do operador utilizado. O mesmo raciocínio é válido para os padrões

BI e DI, que atuam sobre dados irrelevantes. Por sua vez, o padrão DO pode ser visto como uma especialização do padrão DI.

Outra consideração é que o operador AD possui comportamento complementar ao dos operadores BP e DP. Isto significa que, enquanto AD aceita dados que possuem alguma similaridade com os parâmetros de entrada, BP e DP buscam eliminar dados que possuem alguma similaridade com estes parâmetros. Isto pode ser observado também pelo uso de funções diferentes na construção destes padrões.

Resumidamente, foram apresentados neste Capítulo os padrões de eliminação de dados e uma álgebra baseada nos mesmos. No entanto, não foram discutidos em mais detalhes como os eventos que os disparam são tratados. Principalmente se estes eventos irão disparar os padrões diretamente ou se é possível processá-los e combiná-los de forma a tornar o disparo dos padrões uma tarefa que possa ser configurada em funções de diferentes eventos detectados. O próximo capítulo, além de apresentar a arquitetura de suporte aos padrões, discute esses temas a apresenta técnicas que permitem tornar o disparo dos padrões, que é dependente de eventos, uma atividade que pode ser refinada.

CAPÍTULO 6 ARQUITETURA PROPOSTA E MODELAGEM DE EVENTOS COMPOSTOS

Este Capítulo apresenta a arquitetura que apoia o arcabouço autonômico de eliminação de dados. Ela combina diferentes ferramentas para que os padrões propostos possam atuar de forma autonômica. Adicionalmente, é apresentado o método para Descrição e Formalização de Padrões usando redes de Petri de alto nível. Os padrões modelados por este método estão no núcleo do arcabouço, pois contêm as regras básicas que orientam o funcionamento de toda a arquitetura.

6.1 Arquitetura Conceitual

A arquitetura conceitual a ser apresentada utiliza diversos dos conceitos apresentados no Capítulo 4. As camadas que compõem a arquitetura são definidas como:

- Camada de Modelos, Padrões e Regras – responsável por armazenar os modelos, padrões e regras, recuperando-os e atualizando-os quando necessário. Ela pode conter os modelos correspondentes aos padrões de eliminação de dados, detecção de eventos compostos e outros modelos e padrões de manipulação de dados.

- Camada de Projeto e Processamento – responsável pela criação, manutenção e processo dos modelos e padrões. Ela permite que especialistas modelem os comportamentos dos gestores de dados. Ao mesmo tempo, essa camada deve permitir que estes modelos sejam processados através de ferramentas que interpretem e executem diretamente os modelos e padrões ou de outras implementações extraídas destes.

- Camada de Comunicação – responsável por gerenciar a comunicação com os modelos, fornecendo uma interface de mais alto nível. Ela é composta de três componentes: Leitor de Dados, Leitor de Eventos e Gerador de Ações. O Leitor de Dados e o Leitor de Eventos são responsáveis por receber os dados e eventos, respectivamente, advindos das interfaces criadas para as bases de dados, aplicações e agentes. Estes leitores repassam os dados e eventos para a camada de Projeto e

Processamento. O Gerador de Ações é responsável por repassar as decisões advindas da Camada de Projeto e Processamento para a Camada de Interface.

- Camada de Interface – fornece as diferentes formas de comunicação do ambiente externo com os componentes do arcabouço, por meio de interfaces apropriadas.

- Camada de Aplicação – contém as diferentes aplicações e agentes que manipulam dados de forma autonômica.

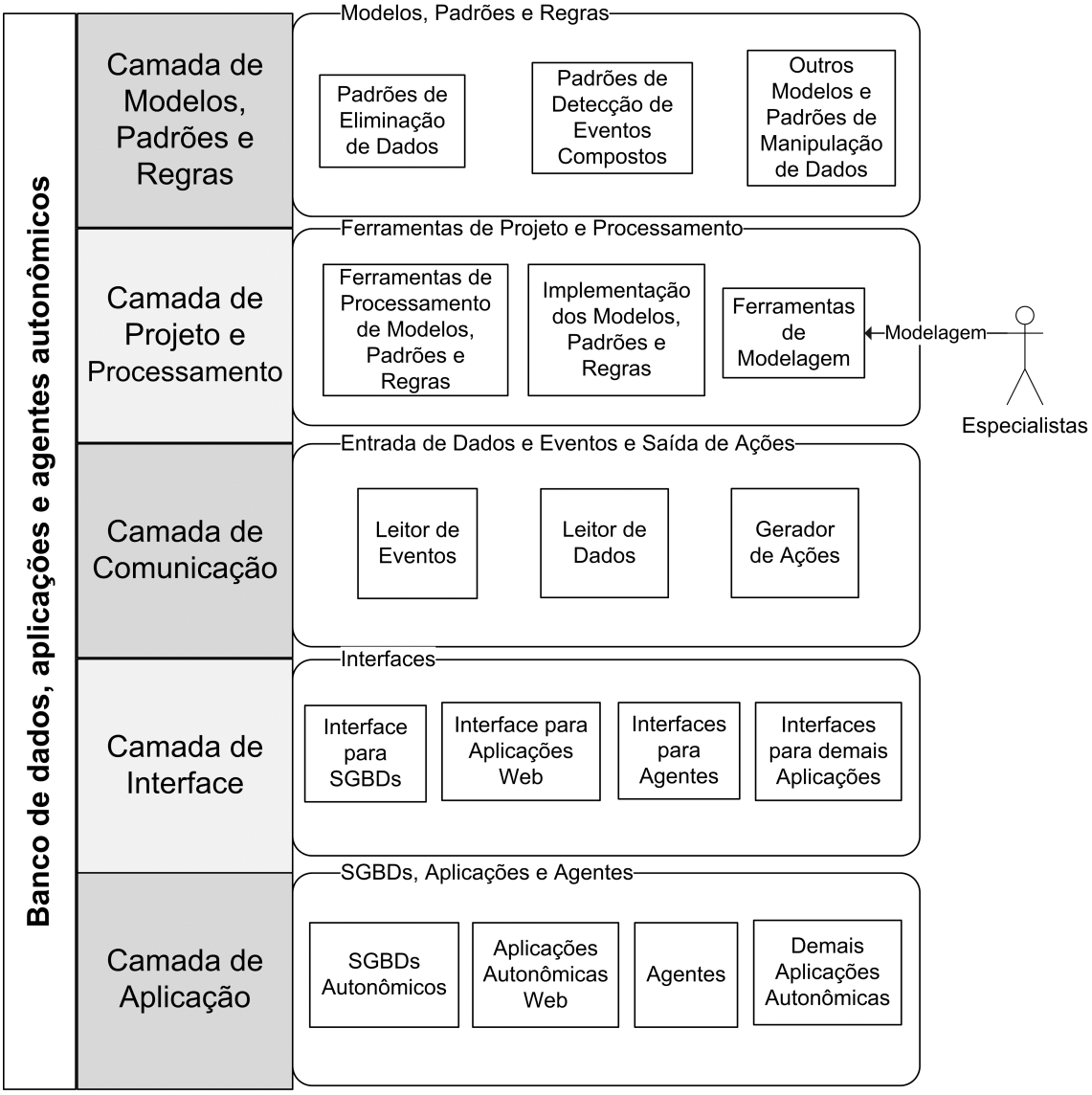

Figura 46 – Arquitetura Autonômica de Manipulação de Dados

Podem-se relacionar as diferentes camadas propostas na arquitetura e as funções de um elemento autonômico (IBM, 2005), como apresentado na Figura 47. As setas contínuas indicam os fluxos de dados e eventos. A seta tracejada indica que o processo

EXECUTAR pode influenciar o processo MONITORAR. Isto pode ocorrer, por exemplo, quando os dados modificados na execução disparam eventos que estão sendo monitorados.

Assim, as camadas responsáveis pela monitoração detectam eventos e coletam dados dos ambientes. As camadas responsáveis pela análise recebem e analisam os eventos e dados, buscando realizar a detecção de eventos compostos e identificar assinaturas de interesse. As camadas responsáveis pelo planejamento processam as regras armazenadas para realizar o tratamento dos dados recebidos. Finalmente, as camadas responsáveis pela execução processam no ambiente as ações decididas na fase de planejamento.

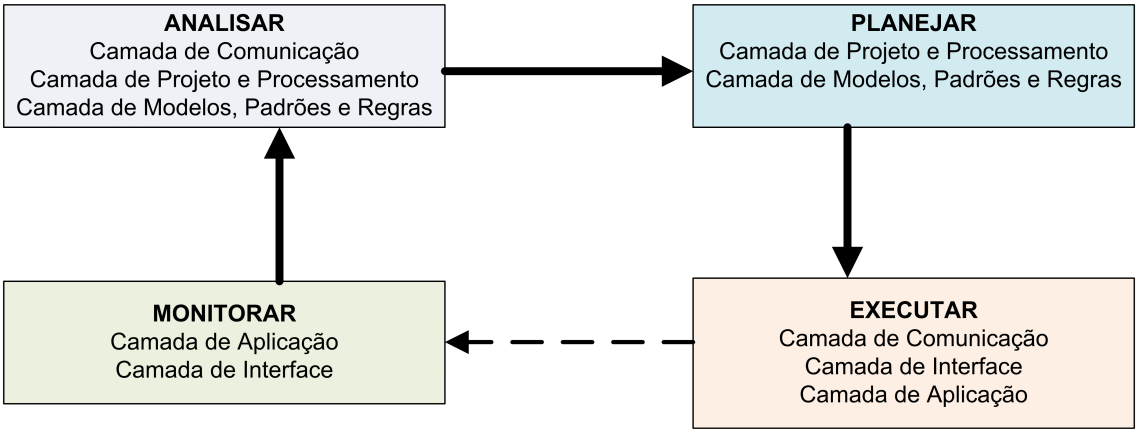

Figura 47 – Relacionamento entre as camadas da arquitetura e as funções de um elemento autonômico

6.2 Arquitetura da Implementação

O arcabouço proposto permite a extensão das aplicações atuais com características autonômicas e demanda que diversos componentes diferentes sejam combinados de forma a prover a funcionalidade desejada. A Figura 48 apresenta a arquitetura com os componentes utilizados que permitem a consecução dos objetivos propostos nesta tese e estão de acordo com os componentes descritos na Figura 46. As camadas apresentam módulos na cor cinza, representando os componentes desenvolvidos pelo autor desta tese; os módulos sem cor de preenchimento representam

módulos a serem desenvolvidos em trabalhos futuros, o que torna a arquitetura expansível; e os módulos em degradê com tons de cinza correspondem a ferramentas de suporte aos outros módulos que já foram desenvolvidos por outros trabalhos e, atualmente, podem ser baixados na *Web*. Os componentes apresentados na arquitetura são discutidos no contexto de cada camada, como a seguir:

- Camada de Modelos, Padrões e Regras – contém os modelos correspondentes aos padrões de eliminação de dados e detecção de eventos compostos desenvolvidos em RPAN, podendo conter outros padrões e modelos desenvolvidos utilizando RPAN ou outras ferramentas.

- Camada de Projeto e Processamento – permite criar, editar e processar modelos. Atualmente comporta: o otimizador e núcleo do SECONDO; a álgebra de Eliminação de Dados; as ferramentas CPN *Tools*, Britney *Suite* e o Simulador de RPAN. Os modelos em RPAN podem manipular marcas representando dados, eventos e parâmetros via *sockets*. Foi estipulado um formato para troca de dados, metadados e eventos entre as RPANs e outras aplicações. A estratégia utilizada é a serialização desses elementos como uma cadeia de caracteres (*strings*), obedecendo a um formato XML-like como a seguir:

"<tu>
<at1>identificador do dado, metadado ou evento</at1>
<at2>nome do metadado ou atributo do dado ou evento</at2>
<at3>valor do metadado ou valor do atributo do dado ou evento</at3>
</tu>"

- Camada de Comunicação – interliga as camadas de interface e as ferramentas que executam os modelos em RPAN. Atualmente, a comunicação é feita pela interface Javagui, adaptada do SECONDO. Este componente permite a leitura de dados e eventos, possibilitando repassar as decisões relacionadas às ações a serem executadas.

- Camada de Interface – responsável por gerenciar a entrada de eventos e dados e convertê-la para chamadas a outros componentes do arcabouço. Servidores *Web* (ex.: JBOSS[49]) podem fornecer suporte a algumas das interfaces do arcabouço proposto, a exemplo da interface correspondente ao módulo FeedOrganizer.war. Este módulo

[49] http://www.jboss.com/products/platforms/application

comunica-se com os outros elementos do arcabouço de modo a eliminar notícias *Web* indesejadas de acordo com os critérios escolhidos pelos usuários da ferramenta *Feed Organizer*, a ser mais bem detalhada no Capítulo 7.

- Camada de Aplicação – contém os componentes que utilizam o arcabouço. Atualmente, é composta do *plugin Feed Organizer* (componente da ferramenta *Feed Organizer*) para o navegador Mozilla Firefox[50], além de futuras interfaces que permitam o desenvolvimento de SGBDs, agentes e aplicações autonômicas.

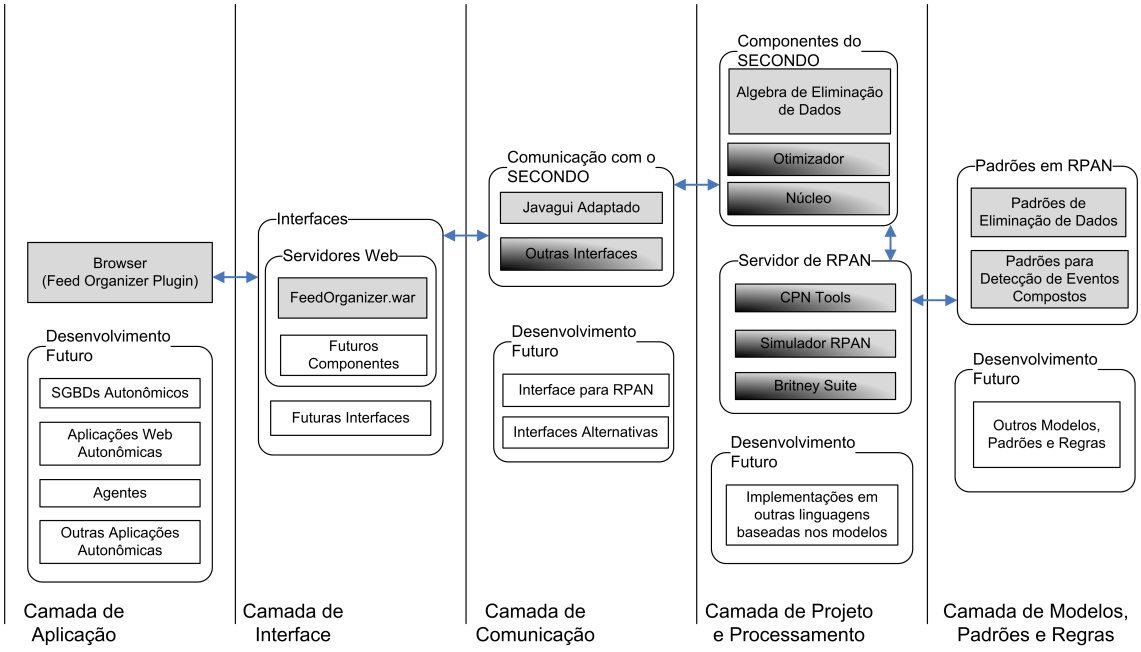

Figura 48 - Arquitetura proposta

Os módulos que representam trabalhos futuros, identificados na arquitetura, necessitam de um conjunto de especialistas de diferentes áreas, dependendo do trabalho a ser executado. A Figura 49 mostra estes especialistas, cujas características são detalhadas a seguir:

- Projetista de Modelos, Padrões e Regras – trabalha na Camada de Projeto e Processamento e é responsável por criar os modelos de processos e manipulação de dados usando as ferramentas de redes de Petri e as técnicas previstas neste trabalho.

- Projetista de Álgebras do SECONDO – trabalha na Camada de Comunicação e é responsável por desenvolver e adaptar as álgebras para suportar os novos padrões de processos como operadores do SGBD SECONDO.

[50] http://br.mozdev.org/

- Projetista de Interfaces – trabalha na Camada de Interface e é responsável por criar as interfaces que adaptam e permitem que outros programas que utilizem os conceitos incorporados nos componentes da arquitetura.

- Projetista de BDs, Aplicações e Agentes – trabalha na Camada de Aplicação e é responsável por desenvolver novas ferramentas ou programas que utilizem os demais componentes propostos.

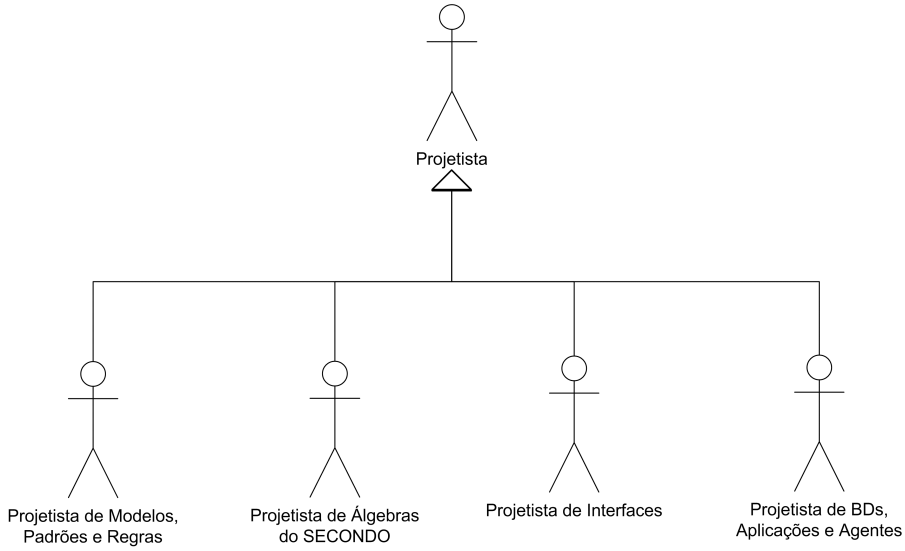

Figura 49 – Especialistas necessários

6.3 Modelagem e Detecção de Eventos Compostos

Para a modelagem de eventos, um requisito importante é que o arcabouço proposto suporte a detecção de eventos compostos. Os eventos compostos tratados nesta tese requerem uma estratégia específica de detecção, pois podem ser gerados de forma não determinística a partir da detecção de combinações envolvendo eventos primitivos ou outros eventos compostos. Dessa forma, são necessárias ferramentas ou ambientes de detecção que apoiem essa modelagem. Consequentemente, as aplicações baseadas nas técnicas propostas pela computação autonômica poderão, além de realizar a monitoração de eventos primitivos em diversos ambientes, detectar eventos compostos gerados pela combinação desses eventos primitivos, ou mesmo outros eventos compostos (PINHEIRO et al., 2008a).

Os eventos compostos modelados por redes de Petri presentes na literatura utilizam extensões complementares às definições propostas de RPCs (GATZIU &

DITTRICH, 1994, LI et al., 2004, BABA-HAMED, 2006). Nenhum desses trabalhos propôs a modelagem desses eventos utilizando os conceitos já existentes de RPAN como os disponíveis na ferramenta CPN *Tools*. Além disso, esses trabalhos propõem somente a detecção desses eventos através da alteração da modelagem da estrutura das redes. Assim, para cada evento composto diferente que se deseje modelar, uma nova estrutura deve ser implementada. Considerando isto, propõe-se a modelagem desses eventos utilizando tanto a abordagem tradicional, baseada na modificação da estrutura das redes, quanto uma abordagem alternativa, baseada no uso de marcas-modelo. Dessa forma, através de uma estrutura fixa, as combinações de eventos primitivos detectados podem ser comparadas com as marcas-modelo através de funções CPN ML, ou seja, sempre que uma combinação conveniente for encontrada, um evento composto é disparado.

Neste trabalho, podem existir lugares do tipo primitivo (*PRIMITIVE*), composto (*COMPOSITE*) e lista de eventos (*EVENT_LIST*), independente do tipo de modelagem escolhida. Este último tipo é utilizado quando há situações em que se faz necessário armazenar eventos de tipos diferentes: primitivos e\ou compostos.

Os atributos apresentados no Pacote de Eventos da Figura 34 são utilizados para compor as ocorrências dos eventos primitivos e compostos a serem modelados. As mesmas convenções utilizadas na Seção 5.1 são utilizadas, quando aplicáveis. Por exemplo, uma marca que representa uma ocorrência de evento primitivo é dada pela expressão:

{eventTime=1,
eventSource=“user John”,
eventTransaction=“write”,
dataId =3121,
parameters =[]
}

Para manter a generalidade do formato do evento, a interligação entre um evento e um ou mais padrões de eliminação de dados ou os operadores derivados dos mesmos é feita por meio do atributo *parameters* que aceita dados da forma *<identificador><metadado><valor>*. Por exemplo, o evento primitivo a seguir aciona o padrão Eliminador de Dados Proibidos por meio do operador DP:

{eventTime=2,
eventSource=“SGBDs/SECONDO/TestDB/Table1”,
eventTransaction=“write”,
dataId=1001,
parameters=
[{identifier=1,
attribute="operator1",
value="query dp <parameters1> <data>"
}]
}

onde *"query dp <parameters1> <data>"* está de acordo com o especificado na Tabela 45.

Uma marca que representa uma ocorrência de evento composto formada pelos eventos primitivos mostrados anteriormente, pode ser dada pela expressão:

{eventTime=2,
eventType=“E”,
parameters=[],
eventList=
[
{eventTime=1,
eventSource=“user John”,
eventTransaction=“write”,
dataId=3121,parameters =[]
},
{eventTime=2,
eventSource=“SGBDs/SECONDO/TestDB/Table1”,
eventTransaction=“write”,
dataId=1001,
parameters=
[{identifier=1,
attribute="operator1",
value="query dp <parameters1> <data>"
}]
}
]
}

Finalmente, uma lista de eventos formada por um evento primitivo e um evento composto correspondentes aos apresentados anteriormente é representada pela expressão:

```
[primitive
        {eventTime=1,
        eventSource=“user John”,
        eventTransaction=“write”,
        dataId =3121,
        parameters =[]
        },
composite
        ({eventTime=2,
        eventType=“E”,
        parameters=[],
        eventList=
                [
                        {eventTime=1,
                        eventSource=“user John”,
                        eventTransaction=“write”,
                        dataId=3121,
                        parameters =[]
                        },
                        {eventTime=2,
                        eventSource=“SGBDs/SECONDO/TestDB/Table1”,
                        eventTransaction=“write”,
                        dataId=1001,
                        parameters=
                                [{identifier=1,
                                attribute="operator1",
                                value="query dp <parameters1> <data>"
                                }]
                        }
                ]
        })
]
```

6.4 Detecção de Eventos Compostos Baseados na Modelagem Estrutural para RPANs

Na modelagem estrutural, cada evento composto básico é detectado através de uma representação de RPAN diferente. Para a detecção de eventos compostos mais complexos, os eventos compostos básicos podem ser combinados.

Caso seja necessário combinar os padrões em paralelo, torna-se então oportuna a criação de cópias de eventos. Isto é feito transferindo o evento através de uma transição auxiliar para os lugares que representam as cópias, como mostrado na Figura 50.

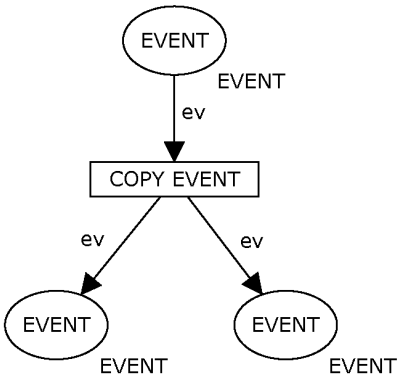

Figura 50 - Cópia de eventos

Caso seja necessário combinar os padrões em sequência, pode se considerar que a saída de um padrão (uma lista de eventos compostos, por exemplo) deva ser convertida num evento primitivo de outro padrão. Assim os padrões poderão ser combinados seqüencialmente, formando cadeias cada vez mais complexas.

As próximas subseções descrevem os eventos compostos básicos através da modelagem estrutural para RPANs.

6.4.1 Evento Composto do Tipo E (And)

A Figura 51 apresenta a RPAN que detecta o evento composto do tipo E. Ela é formada por dois tipos de evento primitivo, representados pelos lugares *PRIMITIVE EVENT1* e *PRIMITIVE EVENT2*. Estes lugares são conectados à transição *DETECT COMPOSITE EVENT* através de dois arcos de leitura que contêm as variáveis *ep1* e *ep2*. A expressão de guarda *[fireAND(ep1, ep2, lec)]* permite que a transição dispare somente se o evento composto gerado por essas duas marcas já não estiver armazenado no lugar *COMPOSITE EVENT*. Este lugar armazena todos os eventos compostos detectados, sendo que a inscrição contida no seu arco de entrada determina como os eventos compostos são formados, o que é visto a seguir:

1. [composite{eventTime = max(#eventTime ep1, #eventTime ep2),

2. eventType="AND", parameters=[],

3. eventList = [ep1, ep2]}]^^lec

A linha 1 indica que uma lista contém um evento composto cujo tempo é dado pelo tempo da ocorrência do evento componente mais recente. A linha 2 indica que é um evento do tipo E e que não possui parâmetros adicionais. A linha 3 indica que a lista de eventos componentes é formada pelas ocorrências dos eventos contidos nas variáveis

ep1 e *ep2*. Além disso, ela combina a lista que contém o evento recém detectado com a lista de eventos compostos, dada pela variável *lec*, através de uma operação de concatenação ^^.

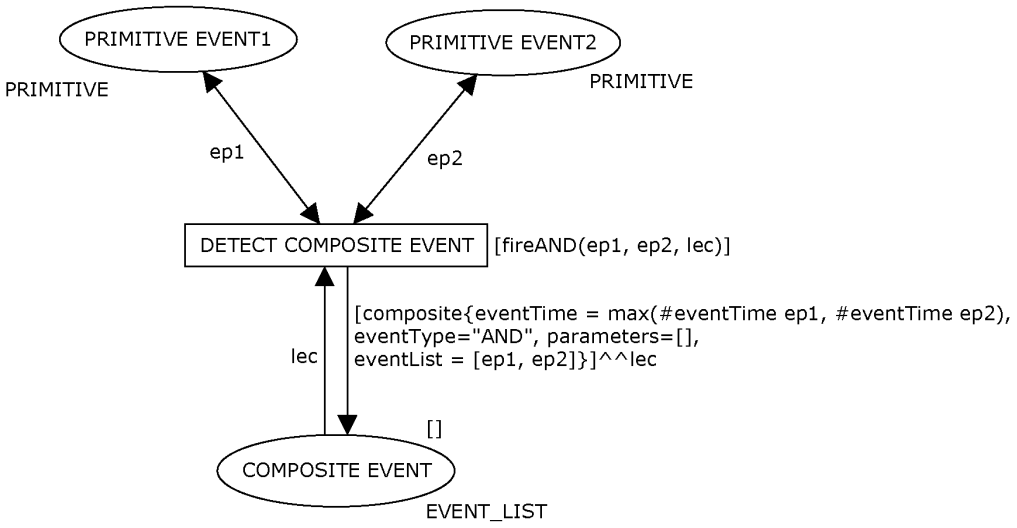

Figura 51 - Evento composto do tipo E

Por exemplo, as seguintes marcas podem corresponder a ocorrências de eventos nos lugares:

- *PRIMITIVE EVENT1* (duas marcas, sendo que *1`* indica um elemento, *++* indica a soma de elementos e o valor zero para *dataId* indica que não há dados associados ao evento):

1`{eventTime=1,
eventSource="Pedro",
eventTransaction="read",
dataId=0,
parameters=[]
}++
1`{eventTime=2,
eventSource="Pedro",
eventTransaction="read",
dataId=0,
parameters=[]
}

- *PRIMITIVE EVENT2* (uma marca):

1`{eventTime=2,
eventSource="Souza",
eventTransaction="write",
dataId=1,
parameters=[]
}

Neste caso, será obtido como resultado no lugar *COMPOSITE EVENT* uma lista formada por dois eventos compostos diferentes, formados por cada um dos eventos primitivos do lugar *PRIMITIVE EVENT1* com o evento primitivo do lugar *PRIMITIVE EVENT2*, como visto a seguir:

```
[composite
        ({eventTime=2,
        eventType="AND",
        parameters=[],
        eventList=
                [
                        {eventTime=1,
                        eventSource="Pedro",
                        eventTransaction="read",
                        dataId=0,
                        parameters=[]
                        },
                        {eventTime=2,
                        eventSource="Souza",
                        eventTransaction="write",
                        dataId=1,
                        parameters=[]
                        }
                ]
        }),
composite
        ({eventTime=2,
        eventType="AND",
        parameters=[],
        eventList=
                [
                        {eventTime=2,
                        eventSource="Pedro",
                        eventTransaction="read",
                        dataId=0,
                        parameters=[]
                        },
                        {eventTime=2,
                        eventSource="Souza",
                        eventTransaction="write",
                        dataId=1,
                        parameters=[]
                        }
                ]
        })
]
```

6.4.2 Evento Composto do Tipo Ou (Or)

A Figura 52 apresenta a RPAN que detecta o evento composto do tipo Ou. Cada ocorrência dos eventos primitivos pode disparar independentemente a sua transição e gerar o evento composto associado. A RPAN é formada por dois tipos de evento primitivo, representados pelos lugares *PRIMITIVE EVENT1* e *PRIMITIVE EVENT2*. O primeiro lugar é conectado à transição *DETECT COMPOSITE EVENT1* e o segundo lugar é conectado a transição *DETECT COMPOSITE EVENT2*. Ambas as ligações são feitas através de um arco de leitura que contém a variável *ep*. As expressões de guarda são iguais e representadas por *[fireAND(ep1, ep2, lec)]*. Elas permitem que as transições disparem somente se o evento composto gerado por qualquer das marcas já não estiver armazenado no lugar *COMPOSITE EVENT*. Este lugar armazena todos os eventos compostos detectados, sendo que as inscrições contida nos seus arcos de entrada determinam como os eventos compostos são formados, o que é visto a seguir:

1. [composite{eventTime= #eventTime ep,

2. eventType="OR", parameters=[],

3. eventList=[ep]}]^^lec

A linha 1 indica que uma lista contém um evento composto cujo tempo é dado pelo tempo da ocorrência do evento componente. A linha 2 indica que é um evento do tipo Ou e que não possui parâmetros adicionais. A linha 3 indica que a lista de eventos é formada pela ocorrência dos eventos contidos na variável *ep*. Além disso, ela combina a lista que contém o evento recém detectado com a lista de eventos compostos, dada pela variável *lec*, através de uma operação de concatenação ^^.

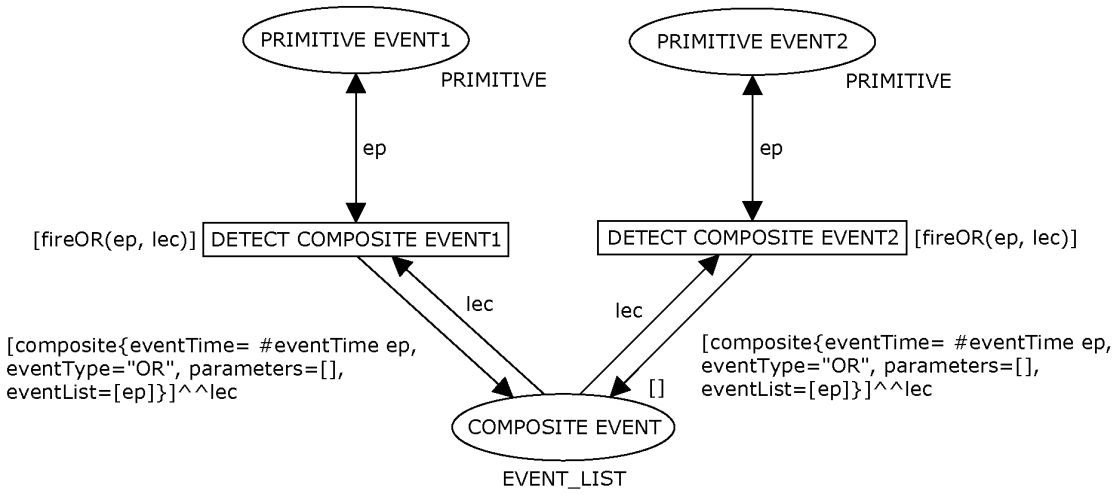

Figura 52 - Evento composto do tipo Ou

Por exemplo, as seguintes marcas podem corresponder a ocorrências de eventos nos lugares:

- *PRIMITIVE EVENT1* (uma marca):

```
1`{eventTime=1,
eventSource="Pedro",
eventTransaction="read",
dataId=0,
parameters=[]}
```

- *PRIMITIVE EVENT2* (uma marca):

```
1`{eventTime=2,
eventSource="Souza",
eventTransaction="write",
dataId=1,
parameters=[]}
```

Neste caso, será obtido como resultado no lugar *COMPOSITE EVENT* uma lista formada por dois eventos compostos diferentes, formados pelo evento primitivo do lugar *PRIMITIVE EVENT1* e pelo evento primitivo do lugar *PRIMITIVE EVENT2*, como visto a seguir:

```
[composite
        ({eventTime=2,
        eventType="OR",
        parameters=[],
        eventList=
                [{eventTime=2,
                eventSource="Souza",
                eventTransaction="write",
                dataId=1,
                parameters=[]
                }]
        }),
composite
        ({eventTime=1,
        eventType="OR",
        parameters=[],
        eventList=
                [{eventTime=1,
                eventSource="Pedro",
                eventTransaction="read",
                dataId=0,
                parameters=[]
                }]
        })
]
```

6.4.3 Evento Composto do Tipo Sequência (Sequence)

A Figura 53 apresenta a RPAN que detecta o evento composto do tipo Sequência em que a ordem de ocorrência dos eventos é determinante para sua detecção. A RPAN é formada por dois tipos de evento primitivo, representados pelos lugares *PRIMITIVE EVENT1* e *PRIMITIVE EVENT2*. Estes lugares são conectados à transição *DETECT COMPOSITE EVENT* através de dois arcos de leitura que contém as variáveis *ep1* e *ep2*. A expressão de guarda *[#eventTime ep1 < #eventTime ep2 andalso fireSequence(ep1, ep2, lec)]* permite que a transição dispare somente se o tempo de ocorrência do evento, dado por *#eventTime ep1*, for menor que o tempo de ocorrência do evento, dado por *#eventTime ep1*, e se o evento composto gerado por essas duas marcas já não estiver armazenado no lugar *COMPOSITE EVENT*, o que é verificado pela função *fireSequence(ep1, ep2, lec)*. Este lugar armazena todos os eventos compostos detectados, sendo que a inscrição contida no seu arco de entrada determina como os eventos compostos são formados, o que é visto a seguir:

1. [composite{eventTime = max(#eventTime ep1, #eventTime ep2),

2. eventType="Sequence", parameters=[],

3. eventList = [ep1, ep2]}]^^lec

A linha 1 indica que uma lista contém um evento composto cujo tempo é dado pelo tempo da ocorrência do evento componente mais recente. A linha 2 indica que é um evento do tipo Sequência e que não possui parâmetros adicionais. A linha 3 indica que a lista de eventos é formada pelas ocorrências dos eventos contidos nas variáveis *ep1* e *ep2*. Além disso, ela combina a lista que contém o evento recém detectado com a lista de eventos compostos, dada pela variável *lec*, através de uma operação de concatenação ^^.

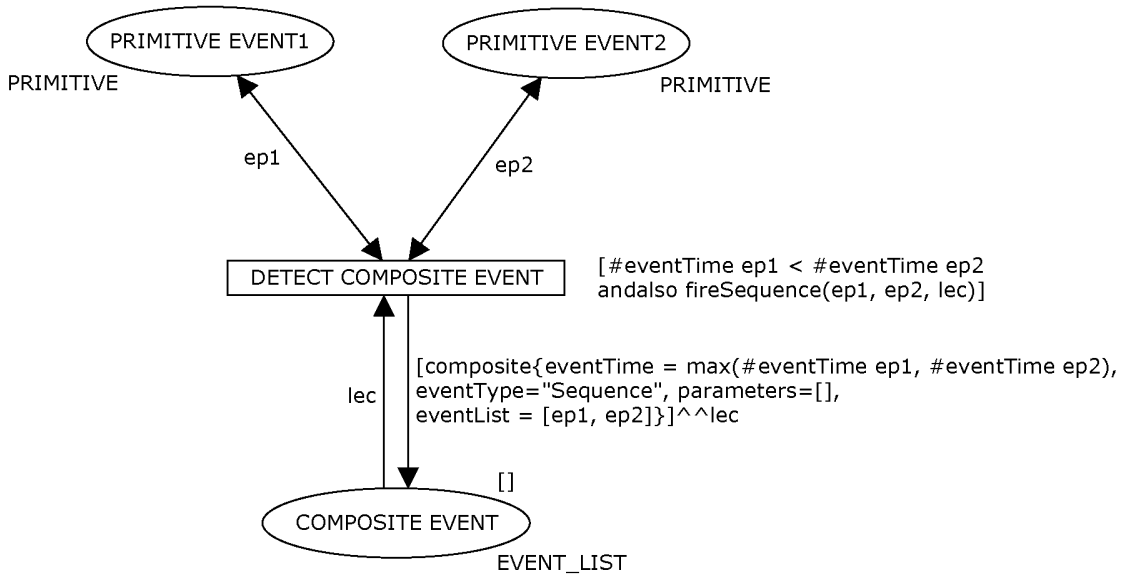

Figura 53 - Evento composto do tipo Sequência

Por exemplo, as seguintes marcas podem corresponder a ocorrências de eventos nos lugares:

- *PRIMITIVE EVENT1* (duas marcas):

1`{eventTime=1,
eventSource="Pedro",
eventTransaction="read",
dataId=0,
parameters=[]
}++
1`{eventTime=2,
eventSource="Pedro",
eventTransaction="read",
dataId=0,
parameters=[]
}

- *PRIMITIVE EVENT2* (uma marca):

1`{eventTime=2,
eventSource="Souza",
eventTransaction="write",
dataId=1,
parameters=[]
}

Neste caso, será obtido como resultado no lugar *COMPOSITE EVENT* uma lista formada pelo evento do lugar *PRIMITIVE EVENT1* que possui tempo de ocorrência menor que o evento do lugar *PRIMITIVE EVENT2*, como visto a seguir:

```
[composite
        ({eventTime=2,
        eventType="Sequence"
        ,parameters=[],
        eventList=
                [
                        {eventTime=1,
                        eventSource="Pedro",
                        eventTransaction="read",
                        dataId=0,
                        parameters=[]
                        },
                        {eventTime=2,
                        eventSource="Souza",
                        eventTransaction="write",
                        dataId=1,
                        parameters=[]
                        }
                ]
        })
]
```

6.4.4 Evento Composto do Tipo Simultâneo (Simultaneous)

A Figura 54 apresenta a RPAN que detecta o evento composto do tipo Simultâneo. Ela é formada por dois tipos de eventos primitivos, representados pelos lugares *PRIMITIVE EVENT1* e *PRIMITIVE EVENT2*. Estes lugares são conectados à transição *DETECT COMPOSITE EVENT* através de dois arcos de leitura que contém as variáveis *ep1* e *ep2*. A expressão de guarda *[(#eventTime ep1) = (#eventTime ep2) andalso fireSimultaneous(ep1, ep2, lec)]* permite que a transição dispare somente se o tempo de ocorrência do evento, dado por *#eventTime ep1*, for igual ao tempo de ocorrência do evento, dado por *#eventTime ep1*, e se o evento composto gerado por essas duas marcas já não estiver armazenado no lugar *COMPOSITE EVENT*, o que é verificado pela função *fireSimultaneous(ep1, ep2, lec).* Este lugar armazena todos os eventos compostos detectados, sendo que a inscrição contida no seu arco de entrada determina como os eventos compostos são formados, o que é visto a seguir:

1. [composite{eventTime = #eventTime ep1,

2. eventType="Simultaneous", parameters = [],

3. eventList = [ep1, ep2]}]^^lec

A linha 1 indica que uma lista contém um evento composto cujo tempo é dado pelo tempo da ocorrência do evento *ep1* que deve ser igual ao de *ep2*. A linha 2 indica

que é um evento do tipo Simultâneo e que não possui parâmetros adicionais. A linha 3 indica que a lista de eventos é formada pelas ocorrências dos eventos contidos nas variáveis *ep1* e *ep2*. Além disso, ela combina a lista que contém o evento recém detectado com a lista de eventos compostos, dada pela variável *lec*, através de uma operação de concatenação ^^.

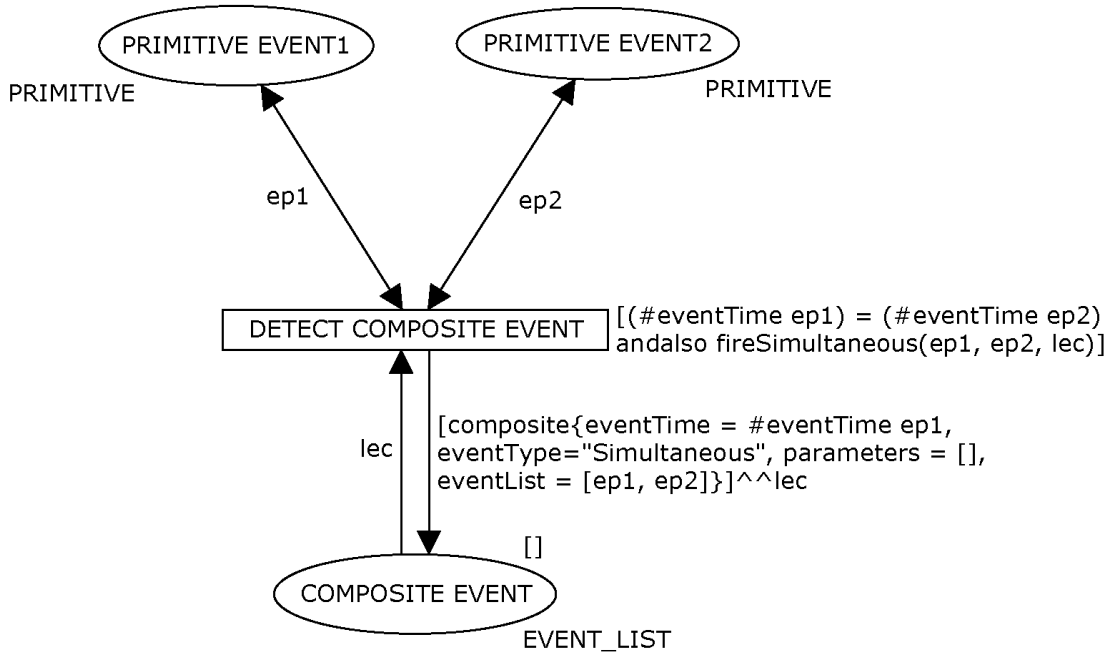

Figura 54 - Evento composto do tipo Simultâneo

Por exemplo, as seguintes marcas podem corresponder a ocorrências de eventos nos lugares:

- *PRIMITIVE EVENT1* (duas marcas):

1`{eventTime=1,
eventSource="Pedro",
eventTransaction="read",
dataId=0,
parameters=[]
}++
1`{eventTime=2,
eventSource="Pedro",
eventTransaction="read",
dataId=0,
parameters=[]
}

- *PRIMITIVE EVENT2* (uma marca):

```
1`{eventTime=2,
eventSource="Souza",
eventTransaction="write",
dataId=1,
parameters=[]
}
```

Neste caso, será obtido como resultado no lugar *COMPOSITE EVENT* uma lista formada pelo evento do lugar *PRIMITIVE EVENT1* que possui tempo de ocorrência igual ao do evento do lugar *PRIMITIVE EVENT2*, como visto a seguir:

```
[composite
        ({eventTime=2,
        eventType="Simultaneous",
        parameters=[],
        eventList=
                [
                        {eventTime=2,
                        eventSource="Pedro",
                        eventTransaction="read",
                        dataId=0,
                        parameters=[]
                        },
                        {eventTime=2,
                        eventSource="Souza",
                        eventTransaction="write",
                        dataId=1,
                        parameters=[]
                        }
                ]
        })
]
```

6.4.5 Evento Composto do Tipo Não Evento (Not Event)

A Figura 55 apresenta a RPAN que detecta o evento composto do tipo Não Evento e utiliza propriedades relacionadas às redes de Petri temporizadas. Ela é formada por dois tipos de evento primitivo, representados pelos lugares *PRIMITIVE EVENT1* e *NOT EVENT*. O primeiro é ligado à transição *DESTROY EVENT* através de um arco de leitura que contém a variável *ep1*. O segundo é ligado às transições *DETECT COMPOSITE EVENT* e *DESTROY EVENT* através de um arco que contém a variável *ep*. Além disso, este lugar é ligado à transição *CLOCK* por meio de um arco de saída que contém a variável *ep* e de um arco de entrada que contém a inscrição *ep@+1*. Toda

vez que a transição *CLOCK* dispara, esta inscrição soma uma unidade de tempo à marca representada pela variável *ep*.

A transição *DESTROY EVENT* dispara quando há uma ocorrência de evento nos lugares *PRIMITIVE EVENT1* e *NOT EVENT* e a sua expressão de guarda é satisfeita, ou seja, o tempo da ocorrência representada pela variável *ep1* for menor que o tempo da ocorrência representada por *ep*. Isto é representado pela expressão *[(#eventTime ep1)<(#eventTime ep)]*.

Considerando que o tempo atual é representado por *intTime()*, a transição *CLOCK* dispara enquanto o tempo atual for menor do que a ocorrência representada por *ep*, segundo a expressão de guarda [*intTime()<(#eventTime ep)]*. Por sua vez, a transição *DETECT COMPOSITE EVENT* só dispara quando o tempo atual é maior ou igual do que o tempo da ocorrência do evento representado por *ep*, o que está de acordo com a expressão de guarda *[intTime()>=(#eventTime ep)]*.

O funcionamento do padrão está baseado no tempo definido para o evento depositado no lugar *NOT EVENT*. Este tempo, não é o tempo de ocorrência desse evento, mas sim o tempo estipulado para o disparo da transição *DETECT COMPOSITE EVENT*. Ou seja, caso seja depositado um evento no lugar *NOT EVENT*, a transição *DETECT COMPOSITE EVENT* só irá disparar se nenhum evento ocorrer no lugar *PRIMITIVE EVENT1* num tempo menor do que o tempo estipulado no evento do lugar *NOT EVENT*. Caso contrário, a transição *DESTROY EVENT*, elimina o evento do lugar *NOT EVENT* e a transição *DETECT COMPOSITE EVENT* não pode mais disparar.

O lugar *COMPOSITE EVENT* armazena todos os eventos compostos detectados, sendo que a inscrição contida no seu arco de entrada determina como os eventos compostos são formados, o que é visto a seguir:

1. [composite{eventTime=(#eventTime ep),

2. eventType="NOT",parameters = [],

3. eventList=[]}]^^lec

A linha 1 indica que uma lista contém um evento composto cujo tempo de ocorrência é dado pelo tempo da ocorrência do evento *ep*. A linha 2 indica que é um evento do tipo Não Evento e que não possui parâmetros adicionais. A linha 3 indica que a lista de eventos é formada pela ocorrência do evento contido na variável *ep*. Além

disso, ela combina a lista que contém o evento recém detectado com a lista de eventos compostos, dada pela variável *lec*, através de uma operação de concatenação ^^.

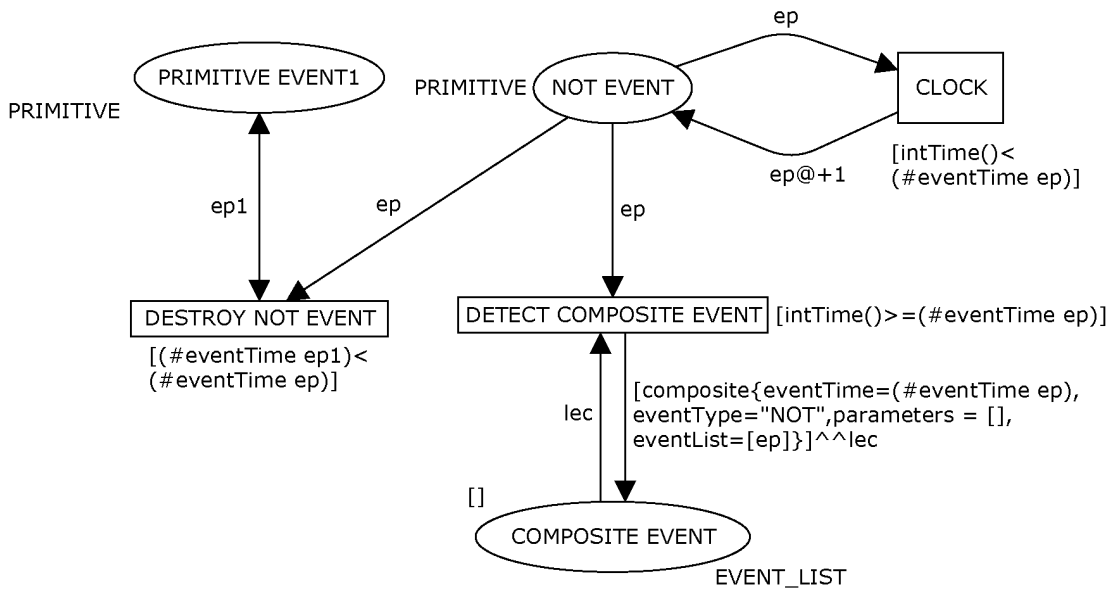

Figura 55 - Evento composto do tipo Não Evento

Por exemplo, as seguintes marcas podem corresponder a ocorrências de eventos nos lugares:

- *PRIMITIVE EVENT1* (uma marca):

1`{eventTime=3,
eventSource="Maria",
eventTransaction="write",
dataId=1,
parameters=[]
}

- *NOT EVENT* (uma marca):

1`{eventTime=2,
eventSource="Souza",
eventTransaction="write",
dataId=1,
parameters=[]
}

Neste caso, será obtido como resultado no lugar *COMPOSITE EVENT* uma lista formada pelo evento do lugar *NOT EVENT*, pois o evento do lugar *PRIMITIVE EVENT1* ocorreu depois do tempo determinado, como visto a seguir:

```
[composite
        ({eventTime=2,
        eventType="NOT",
        parameters=[],
        eventList=
                [{eventTime=2,
                eventSource="",
                eventTransaction="",
                dataId=0,
                parameters=[]
                }]
        })
]
```

6.4.6 Evento Composto do Tipo Cumulativo (History)

Esse evento composto é detectado cada vez que *m* ocorrências de um determinado evento ocorrem dentro de um intervalo de tempo pré-estabelecido. Em outras palavras, todos os atributos dessas *m* ocorrências devem ser os mesmos, a exceção do tempo. A Figura 56 apresenta a RPAN que detecta um evento composto do tipo Cumulativo. Ela é formada por um tipo de evento primitivo, representado pelo lugar *PRIMITIVE EVENT1*. Este lugar é conectado à transição *PUT ON LIST* através de um arco de leitura que contém a variável *ep*. Os valores *startPeriod* e *endPeriod* determinam o período de tempo no qual os eventos serão considerados na formação do evento composto. A expressão de guarda *[#eventTime ep >= startPeriod andalso #eventTime ep < endPeriod andalso (not (contains lp [primitive(ep)]))]* permite que a transição dispare somente se o tempo de ocorrência do evento, dado por *#eventTime ep*, for maior ou igual a *startPeriod* e menor ou igual a *endPeriod*, e se o evento já não estiver armazenado no lugar *EVENT LIST*, o que é verificado por *(not (contains lp [primitive(ep)])*. Este lugar armazena todos os eventos primitivos em uma lista, sendo que eles são incluídos na mesma através da inscrição no seu arco de saída, dada por *primitive(ep)::lp*.

O lugar *EVENT LIST* é conectado à transição *DETECT COMPOSITE EVENT* através de um arco de leitura que contém a variável *lp*. A expressão de guarda *[length lp >=m andalso fireHistory(lp, m, "History", [], lec)<>[]]* permite que a transição dispare somente se o tamanho da lista, dada por *length lp*, for maior ou igual ao número de ocorrências desejadas, dado por *m*, e se o evento composto gerado por essas marcas for diferente de vazio. O lugar *COMPOSITE EVENT* armazena todos os eventos compostos detectados, sendo que a função contida no seu arco de entrada, dada por *fireHistory(lp,*

m, "History", [], lec), determina como os eventos compostos são formados. Essa função gera uma lista de eventos compostos ainda não detectados a partir da combinação de *m* eventos obtidos dos eventos contidos na lista representada por *lp*. O tempo da ocorrência de cada evento composto detectado corresponde ao tempo da ocorrência do evento componente mais recente. Depois, é feita a concatenação deste resultado com a lista representada por *lec* através do operador ^^.

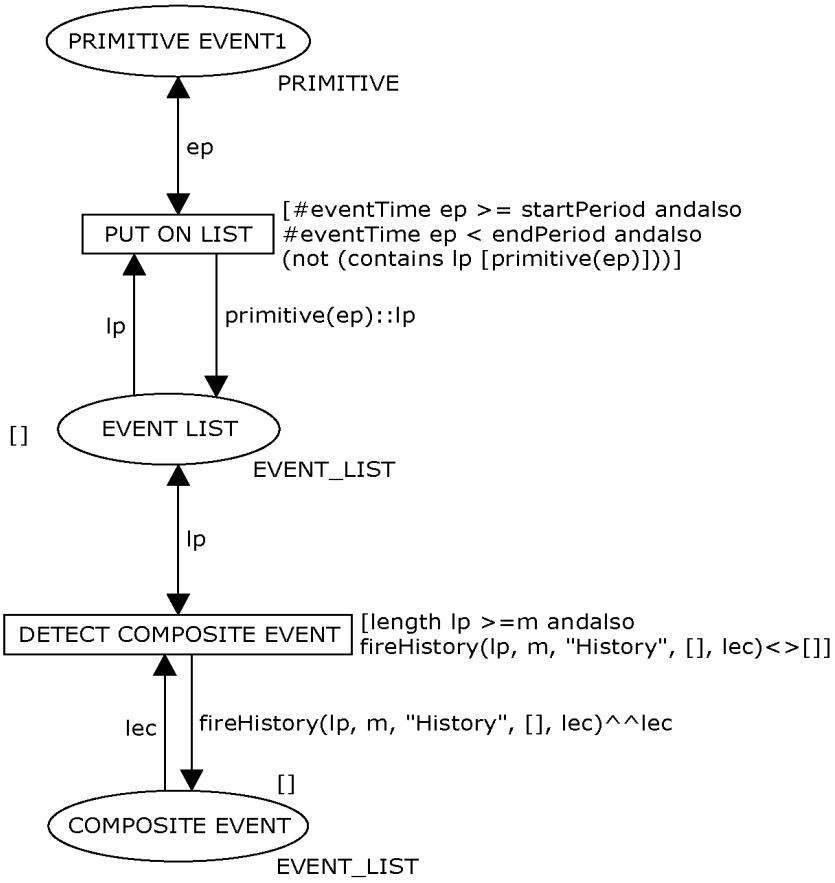

Figura 56 - Evento composto do tipo Cumulativo

Por exemplo, as seguintes marcas podem corresponder a ocorrências de eventos no lugar *PRIMITIVE EVENT1* (três marcas):

1`{eventTime=1,
eventSource="Pedro",
eventTransaction="read",
dataId=0,
parameters=[]}++
1`{eventTime=2,
eventSource="Pedro",
eventTransaction="read",
dataId=0,
parameters=[]}++
1`{eventTime=3,
eventSource="Pedro",
eventTransaction="read",
dataId=0,
parameters=[]}

Supondo que *m* seja dois, será obtido como resultado no lugar *COMPOSITE EVENT* uma lista formada pelos três eventos do lugar *PRIMITIVE EVENT1* mostrados anteriormente, combinados dois a dois, como visto a seguir:

```
[composite
        ({eventTime=3,
        eventType="History",
        parameters=[],
        eventList=
                [
                        {eventTime=2,
                        eventSource="Pedro",
                        eventTransaction="read",
                        dataId=0,
                        parameters=[]
                        },
                        {eventTime=3,
                        eventSource="Pedro",
                        eventTransaction="read",
                        dataId=0,
                        parameters=[]
                        }
                ]
        }),
composite
        ({eventTime=3,
        eventType="History",
        parameters=[],
        eventList=
                [
                        {eventTime=1,
                        eventSource="Pedro",
                        eventTransaction="read",
                        dataId=0,
                        parameters=[]
                        },
                        {eventTime=3,
                        eventSource="Pedro",
                        eventTransaction="read",
                        dataId=0,
                        parameters=[]
                        }
                ]
        }),
composite
        ({eventTime=2,
        eventType="History",
        parameters=[],
        eventList=
                [
                        {eventTime=1,
                        eventSource="Pedro",
                        eventTransaction="read",
                        dataId=0,
                        parameters=[]
                        },
                        {eventTime=2,
                        eventSource="Pedro",
                        eventTransaction="read",
                        dataId=0,
                        parameters=[]}
                ]
        })
]
```

6.4.7 Evento Composto do Tipo Qualquer (Any)

Esse evento composto é detectado quando existem *m* ocorrências *n* eventos distintos ($m \leq n$), independentemente da ordem. Este trabalho considera que se o atributo *Tempo do Evento* possuir o mesmo valor para um par de eventos, pelo menos um dos outros pares de atributos equivalentes desses eventos deve ser diferente entre si. Além disso, considera também que se atributo *Tempo do Evento* possuir valor diferente para um par de eventos, todos os outros pares de atributos equivalentes desses eventos podem ser iguais entre si. Dessa forma, não existem ocorrências iguais de eventos, pois pelo menos um dos atributos será diferente. Assim, quando se diz **evento distinto**, se quer dizer que o evento possui necessariamente um atributo diferente, desconsiderando o atributo *Tempo de Evento*.

A Figura 57 apresenta a RPAN que detecta um evento composto do tipo Qualquer. Ela é formada por n tipos de eventos primitivos, representado pelos lugares *PRIMITIVE EVENT1*, *PRIMITIVE EVENT2* e *PRIMITIVE EVENTN*. O primeiro lugar é conectado à transição *PUT ON LIST1*, o segundo lugar é conectado a transição *PUT ON LIST2* e assim sucessivamente, até que o enésimo lugar que é conectado à transição *PUT ON LISTN*. Todas estas conexões são feitas através de arcos que contém a variável *ep*. As transições mencionadas anteriormente possuem expressões de guarda, dadas por *[not (contains lp [primitive(ep)])]*, que disparam somente se o lugar *EVENT LIST* não possuir o evento representado por *ep*. Este lugar armazena todos os eventos primitivos em uma lista, sendo que eles são incluídos na mesma através da inscrição no seu arco de saída, dada por *primitive(ep)::lp*.

O lugar *EVENT LIST* é conectado à transição *DETECT COMPOSITE EVENT* através de um arco de leitura que contém a variável *lp*. A expressão de guarda *[length lp >=m andalso fireAny(lp, m, "Any", [], lec)<>[]]* permite que a transição dispare somente se o tamanho da lista, dada por *length lp*, for maior ou igual ao número de ocorrências desejadas, dado por *m*, e se o evento composto gerado por essas marcas for diferente de vazio. O lugar *COMPOSITE EVENT* armazena todos os eventos compostos detectados, sendo que a função contida no seu arco de entrada, dada por *fireAny(lp, m, "Any", [], lec)*, determina como os eventos compostos são formados. Essa função gera uma lista de eventos compostos ainda não detectados a partir da combinação de *m* eventos distintos obtidos dos eventos contidos na lista representada por *lp*. O tempo da

ocorrência de cada evento composto detectado corresponde ao tempo da ocorrência do evento componente mais recente. Depois é feita a concatenação deste resultado com a lista representada por *lec* através do operador ^^.

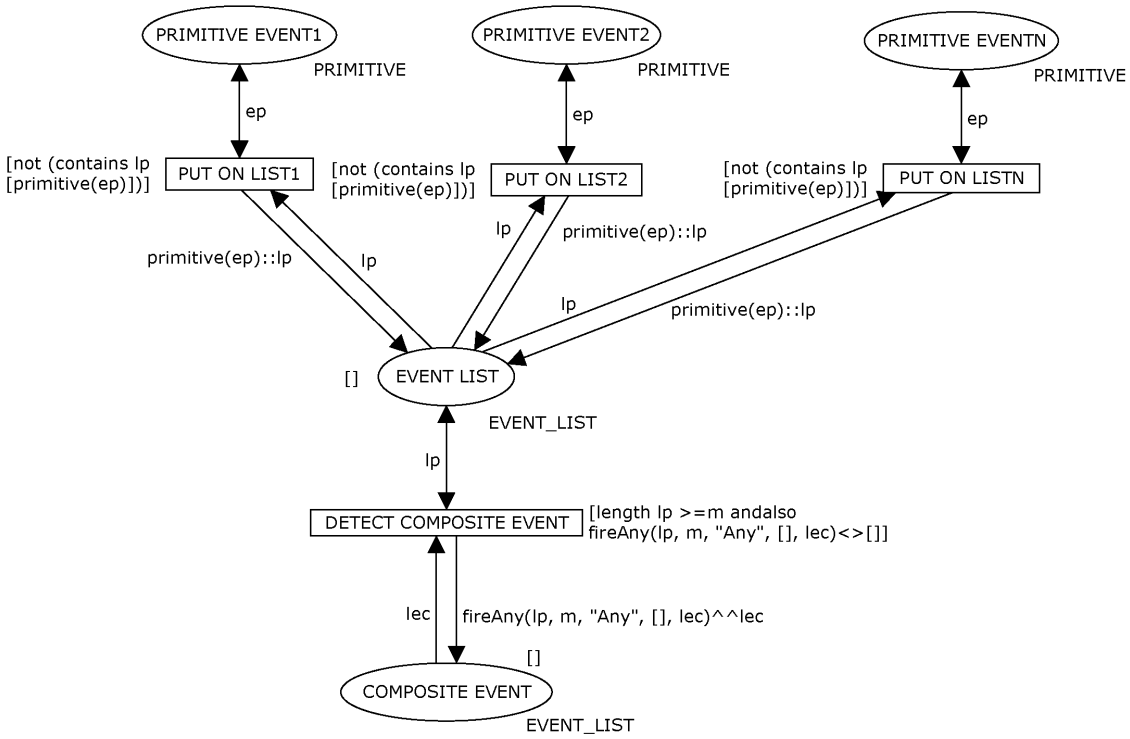

Figura 57 - Evento composto do tipo Qualquer

Por exemplo, as seguintes marcas podem corresponder a ocorrências de eventos nos lugares:

- *PRIMITIVE EVENT1* (uma marca):

1`{eventTime=1,
eventSource="Pedro",
eventTransaction="read",
dataId=0,
parameters=[]
}

- *PRIMITIVE EVENT2* (uma marca):

1`{eventTime=2,
eventSource="Souza",
eventTransaction="write",
dataId=1,
parameters=[]
}

- *PRIMITIVE EVENTN* (uma marca):

1`{eventTime=3,
eventSource="Maria",
eventTransaction="write",
dataId=1,
parameters=[]
}

Supondo que *m* seja dois, será obtido como resultado no lugar *COMPOSITE EVENT* uma lista formada pelos três eventos mostrados anteriormente, combinados dois a dois, como visto a seguir:

```
[composite
        ({eventTime=3,
        eventType="any",
        parameters=[],
        eventList=
                [
                        {eventTime=2,
                        eventUser="Souza",
                        eventTransaction="write",
                        dataId=1,
                        parameters=[]
                        },
                        {eventTime=3,
                        eventUser="Maria",
                        eventTransaction="write",
                        dataId=1,
                        parameters=[]
                        }
                ]
        }),
composite
        ({eventTime=3,
        eventType="any",
        parameters=[],
        eventList=
                [
                        {eventTime=1,
                        eventUser="Pedro",
                        eventTransaction="read",
                        dataId=0,
                        parameters=[]
                        },
                        {eventTime=3,
                        eventUser="Maria",
                        eventTransaction="write",
                        dataId=1,
                        parameters=[]
                        }
                ]
        }),
composite
        ({eventTime=2,
        eventType="any",
        parameters=[],
        eventList=
                [
                        {eventTime=1,
                        eventUser="Pedro",
                        eventTransaction="read",
                        dataId=0,
                        parameters=[]
                        },
                        {eventTime=2,
                        eventUser="Souza",
                        eventTransaction="write",
                        dataId=1,
                        parameters=[]}
                ]
        })
]
```

6.5 Detecção de Eventos Compostos Baseados em Marcas-Modelo para RPANs

Na abordagem de detecção de eventos compostos baseada em marcas-modelo, a estrutura da rede não muda, mas sim as marcas que determinam os modelos de eventos compostos que devem ser detectados. Neste sentido, são usadas funções que comparam as marcas-modelo, representantes dos eventos compostos a serem detectados, com as possíveis combinações de eventos primitivos detectados por sensores, ou mesmo eventos compostos detectados por outras marcas-modelo. A grande vantagem no uso dessas funções é que a estrutura da rede não precisa ser alterada, mesmo que os tipos de eventos compostos a serem detectados sejam diferentes.

A detecção de eventos compostos demanda que sejam configurados o tipo do evento composto a ser detectado juntamente com o período de detecção, prioridade e lista de eventos componentes. Em virtude disso, é proposto o seguinte conjunto de atributos que caracterizam uma marca-modelo:

- *tempoInicialEvento* – indica o limite de tempo (*timestamp*) mínimo considerado para a ocorrência dos eventos componentes.

- *tempoFinalEvento* – indica o limite de tempo (*timestamp*) máximo considerado para a ocorrência dos eventos componentes. Caso seja desejável considerar um tempo infinito para a detecção dos eventos, convenciona-se utilizar o valor zero.

- *nivelPrioridade* – indica o nível de prioridade, caso sejam detectados dois eventos compostos simultaneamente. Convencionou-se que o maior nível de prioridade é dado pelo valor um, o segundo maior nível de prioridade é dado pelo valor dois e assim sucessivamente. O valor zero indica que a detecção do evento composto não tem nenhuma prioridade. Eventos com a mesma prioridade e detectados ao mesmo tempo são não determinísticos.

- *tipoEvento* – indica o tipo de evento composto que é detectado, quais sejam: E, Ou, Sequência, Simultâneos, Não Evento, Cumulativo, Qualquer.

- *parametrosEventoComposto* – indica uma lista de parâmetros adicionais que podem ser cadastrados pelos usuários. Por exemplo, uma chamada a um operador de eliminação de dados.

- *listaEventos* – contém a lista de modelos de eventos que compõem o evento composto que é detectado. Cada evento dessa lista é composto pelos seguintes atributos:
    - *tempoEvento* – indica o limite de tempo (*timestamp*) mínimo considerado para a ocorrência do evento.
    - *fonteEvento* – indica a entidade responsável pelo disparo do evento. Caso a fonte não seja importante, o valor "*any*" deve ser utilizado.
    - *transacaoEvento* – indica a transação em que o evento ocorreu. Caso a transação não seja importante, o valor "*any*" deve ser utilizado.
    - *idDadoRelacionado* – é o identificador do elemento de dado afetado pelo evento. Caso o dado afetado não seja importante, o valor zero deve ser utilizado.
    - *parametros* – parâmetros adicionais definidos por usuários. Por exemplo, o número de notícias retornadas de um site de notícias, o endereço de certo documento, etc. Caso os parâmetros não sejam importantes, uma lista vazia deve ser fornecida. Caso os parâmetros devam ser nulos, então uma lista com o valor *null* (nulo) deve ser fornecida.

Com base nos atributos definidos anteriormente, é possível configurar a detecção de eventos compostos usando a arquitetura proposta. Considerando que numa marca-modelo os eventos são envoltos por chaves, as listas são envoltas por colchetes e os atributos são separador por ponto e vírgula, a seguinte configuração poderia ser feita:

1. {tempoInicialEvento = 0, tempoFinalEvento = 10,

2. nivelPrioridade = 1,

3. tipoEvento = E,

4. parametrosEventoComposto = [],

5. listaEventos = [

6. {tempoEvento = 0, fonteEvento = “any”, transacaoEvento = “any”,

7. idDadoRelacionado = 0, parâmetros = []},

8. {tempoEvento = 5, fonteEvento = “Sistema X”, transacaoEvento = “read”,

9. idDadoRelacionado = 0, parametros = []}]}

Neste exemplo, a linha 1 determina o período em que o evento composto será detectado. A linha 2 define a prioridade para detecção deste evento composto como 1. A linha 3 classifica este evento composto como do tipo E. A linha 4 diz que não há parâmetros adicionais. A linha 4 inicia a relação de eventos primitivos que formam este evento composto. As linhas 6 e 7 apresentam a assinatura do primeiro evento primitivo componente. Este evento deve ter tempo de ocorrência maior ou igual a zero unidade de tempo, sendo que a fonte, transação, identificação do dado relacionado e parâmetros podem assumir qualquer valor. As linhas 8 e 9 apresentam o segundo evento componente. Este evento deve ter tempo de ocorrência maior ou igual a 5 unidades de tempo, ter como sua fonte o *Sistema X*, ter como transação de origem a transação de leitura (*read*) e, finalmente, permitir qualquer valor para a identificação dos dados e parâmetros relacionados.

As marcas-modelo podem ser usadas como parâmetros de entrada em expressões de guarda da transição de uma RPAN. Estas expressões devem combinar os eventos capturados e checar se estão de acordo com as marcas-modelo existentes.

A Figura 58 mostra uma rede que pode ser utilizada na detecção de eventos compostos utilizando marcas-modelo. O lugar *CAPTURED EVENT* armazena todos os eventos primitivos capturados em uma lista de eventos, servindo como um concentrador de eventos. O lugar *EVENT TEMPLATE* armazena as marcas modelos que representam as assinaturas dos eventos compostos a serem detectados. O lugar *DETECTED COMPOSITE EVENT* armazena os eventos compostos detectados a partir dos eventos capturados. Por sua vez, a transição *DETECT COMPOSITE EVENT* detecta os eventos compostos através de uma expressão de guarda.

O arco que possui a variável *etl* é um arco de leitura e dá acesso à lista de marcas-modelo registradas. O arco que possui a variável *le* também é um arco de leitura e dá acesso à lista de eventos primitivos capturados. O arco que possui a variável *lec* obtém a lista de eventos compostos antiga, enquanto que o arco que contém a função *fireComposite(etl, le, lec)* produz a nova lista de eventos compostos detectados. A expressão de guarda permite o disparo da transição somente se o valor da função *fireComposite(etl, le, lec)* for diferente da variável *lec*. A função combina os eventos capturados e os compara com as marcas-modelo. O valor retornado pela função é dado por uma lista que contém todas as combinações de eventos que possuam marcas-modelo correspondentes. Toda vez que um evento é adicionado ao lugar *CAPTURED EVENT*, o

resultado da função pode ser alterado, atualizando a lista de eventos compostos detectados.

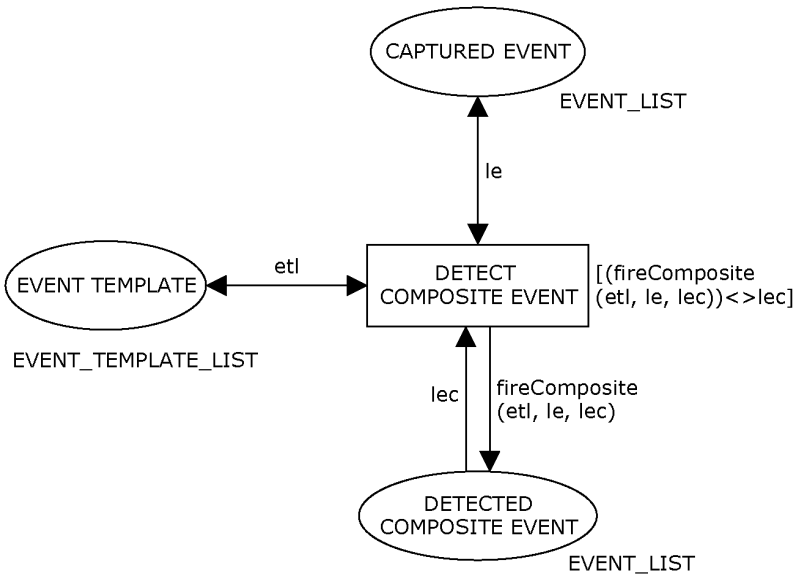

Figura 58 – Rede para detecção de eventos compostos

Os resultados obtidos por essa abordagem são os mesmos obtidos pela modelagem estrutural dos eventos compostos. Uma diferença é que essa estratégia utiliza como entrada para eventos o lugar *CAPTURED EVENT* que é um concentrador de eventos na forma de lista. Este concentrador pode receber eventos de diferentes tipos, distintos ou não. Além disso, o lugar *EVENT TEMPLATE* atua como um concentrador de modelos de eventos compostos a serem detectados, dispensando a necessidade de construir diferentes estruturas para detectar diferentes tipos de eventos compostos. A última diferença relevante é que a função que determina o formato dos eventos detectados é mais complexa que as funções utilizadas na outra abordagem, pois incorpora informações que eram representadas pela estrutura da rede.

Adicionalmente, de modo a permitir a manipulação de eventos utilizando a interface do SECONDO, foram definidos três operadores que utilizam eventos e marcas-modelo, listados a seguir:

- *re* (*register event*) – este operador registra um evento na rede de Petri;
- *rt* (*register template*) – este operador registra uma marca-modelo na rede de Petri;
- *ut* (*unregister template*) – este operador apaga uma marca-modelo previamente registrada na rede de Petri.

A resposta obtida na utilização desses operadores é a simples confirmação de sucesso no recebimento dos eventos ou marcas-modelo. A detecção de eventos compostos é feita de forma não-determinística e enviada paralelamente para a aplicacação cliente via *sockets*, utilizando uma porta de comunicação diferente (vide Item 5 do Anexo III).

6.6 O Problema da Complexidade Exponencial

Uma das complicações que surge na detecção de todos os eventos compostos é a possibilidade de combinação das várias ocorrências dos eventos primitivos, qualquer que seja o tipo de modelagem, baseada em estrutura ou marcas-modelo. O problema é que a armazenagem continuada e prolongada de eventos primitivos gera uma dificuldade cada vez maior para a detecção de eventos compostos, sendo que o algoritmo necessário para descoberta de eventos compostos é combinatório, possuindo, portanto, complexidade exponencial, ou seja, $O(2^n)$. Dessa forma, para um número de eventos grande, o algoritmo da RPAN não é executável na prática.

A maioria das abordagens que tratam a detecção de eventos compostos (GATZIU & DITTRICH, 1994, LI et al., 2004, BABA-HAMED, 2006) resolvem este problema através do descarte dos eventos primitivos usados na detecção de um evento composto. Assim, a combinação de eventos primitivos fica limitada.

Os modelos apresentados neste capítulo também podem adotar este tipo de estratégia. Se for necessário eliminar os eventos primitivos, utilizados na detecção de um evento composto, basta substituir os arcos de leitura de todos os modelos por arcos unidirecionais que levem os eventos na direção do lugar de detecção do evento. Assim, toda vez que um evento composto for detectado, os eventos primitivos associados serão descartados. Obviamente, neste caso, várias combinações de eventos deixarão de ser detectadas.

Existem duas outras possibilidades que não eliminam por completo o problema da complexidade exponencial, mas reduzem os seus efeitos, eliminando os eventos primitivos mais antigos com o passar do tempo. A primeira, que é adotada neste trabalho, estipula um período de validade para a detecção de cada evento composto e só mantém eventos primitivos que estejam dentro dos períodos válidos existentes. A

segunda utiliza períodos de validade associados aos eventos primitivos. Assim, depois de certo tempo, os eventos primitivos são descartados.

As técnicas propostas tornam possível a construção de uma ferramenta que execute o descarte dos dados utilizando a monitoração de eventos e detecção de eventos compostos. Assim, por exemplo, é possível disparar os padrões de eliminação de dados ou outros processos desejados.

CAPÍTULO 7 A FERRAMENTA *FEED ORGANIZER* E OS EXPERIMENTOS RELACIONADOS

7.1 Introdução

Este Capítulo apresenta a ferramenta Feed Organizer e os experimentos realizados. Esta ferramenta utiliza alguns padrões de eliminação de dados através da interface do SGBD SECONDO, eliminando notícias consideradas irrelevantes, segundo determinados critérios. Os experimentos foram baseados em análises comparativas dos resultados obtidos através do uso da ferramenta proposta, chamada Feed Organizer, com as opiniões dos participantes do experimento e resultados obtidos com o uso da ferramenta Yahoo *Pipe*, detalhada no Capítulo 2. Os procedimentos de instalação da ferramenta *Feed Organizer*, bem como de todas as implementações feitas no arcabouço de eliminação de dados, são apresentados no Anexo III.

7.2 A Ferramenta *Feed Organizer*

Uma das ferramentas resultantes do trabalho proposto é a *Feed Organizer*. Esta ferramenta permite avaliar vários dos conceitos apresentados nos capítulos anteriores, tais como a aplicação dos padrões de eliminação de dados e o uso de eventos compostos.

O objetivo desta ferramenta é eliminar notícias que apresentem pouca relevância para os seus usuários através da configuração de diversos filtros, similarmente ao que acontece com a ferramenta Yahoo *Pipe*, introduzida no Capítulo 2.

Neste trabalho, foram construídas duas versões da ferramenta *Feed Organizer*. A primeira versão, chamada versão Alfa, foi utilizada na primeira parte dos experimentos e permitiu avaliar a viabilidade de uso do arcabouço. Ela permite basicamente configurar três dos padrões propostos (os padrões: Dados Permitidos, Eliminador de Dados Proibidos e Eliminador de Dados Similares), apresentando os resultados obtidos via *Web*.

A segunda versão, chamada versão Beta, foi utilizada na segunda parte dos experimentos e possibilitou a avaliação do resultado da aplicação combinada dos

operadores de eliminação de dados comparado aos resultados obtidos com o uso da ferramenta Yahoo *Pipe*. Essa versão permite configurar cinco dos padrões propostos (os padrões: Dados Permitidos, Eliminador de Dados Proibidos, Eliminador de Dados Similares, Eliminador de Dados Irrelevantes e Eliminador de Dados Obsoletos), além de prover algumas características autonômicas ao arcabouço, tais como: atualização dos níveis de similaridade de alguns padrões para atender a requisitos da consulta dos usuários e bloqueio de endereços de notícias utilizando eventos compostos. Esta última característica da ferramenta permitiu avaliar a viabilidade de detecção do evento composto cumulativo apresentado na Seção 6.4.6.

7.2.1 *Feed Organizer*: Versão Alfa

A versão Alfa desta ferramenta (PINHEIRO et al., 2009a, PINHEIRO et al., 2009b, PINHEIRO et al., 2010) é utilizada na primeira fase dos experimentos realizados e tem sua interface apresentada na Figura 59 Para utilizá-la, basta que se instale o *plugin Feed Organizer* no navegador Mozilla Firefox. Depois, ao clicar no botão *Feed Organizer*, que se torna visível, uma nova aba será aberta no navegador com algumas opções de configuração. Ao selecionar as opções desejadas e clicar no botão Filter, a ferramenta grava essas configurações e filtra os arquivos existentes. A partir desse momento, este processo de filtragem pode ser iniciado automaticamente toda a vez que um novo arquivo feed é baixado.

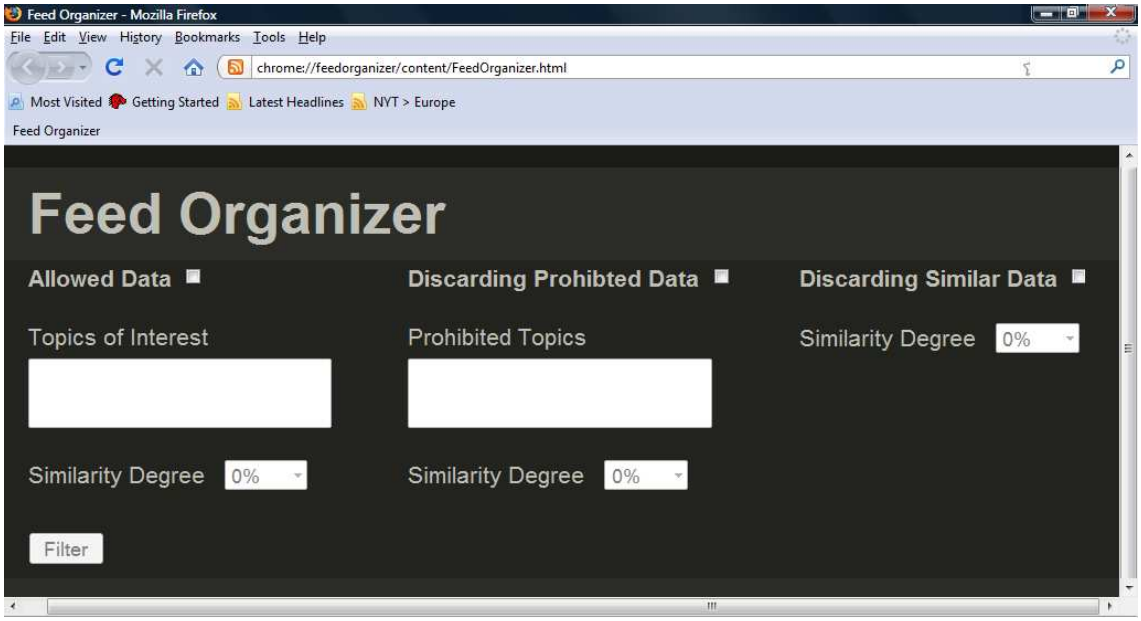

Figura 59 – Interface da Ferramenta *Feed Organizer*

Realizando uma análise da tela de configuração da ferramenta, podem ser descritas algumas funcionalidades. Entre elas, destaca-se a seleção dos padrões que serão utilizados no processo de filtragem das notícias. O primeiro padrão listado é o padrão Dados Permitidos, que permite ao usuário definir tópicos de interesse para a filtragem e o grau de similaridade exigido com relação a estes tópicos. Em seguida, tem-se o padrão Eliminador de Dados Proibidos, que possui um campo para os tópicos proibidos e, assim como ocorre para o padrão Dados Permitidos, permite definir o grau de similaridade com relação aos tópicos fornecidos. Por fim, tem-se o padrão Eliminador de Dados Similares, que possui apenas o campo para definição do grau de similaridade, comparando e descartando as notícias mais antigas consideradas similares com base neste valor.

A ferramenta permite a combinação de termos através do uso de aspas ou vírgulas separadoras. Por exemplo, “McCain Obama” irá procurar a ocorrência dos dois termos em cada notícia, dependendo do nível de similaridade. No caso de termos isolados, ela sempre procurará a existência de cada um dos termos na notícia e não dos dois simultaneamente.

7.2.2 *Feed Organizer*: Versão Beta

A versão Beta surgiu como uma evolução natural da ferramenta *Feed Organizer*, agregando outros conceitos apresentados no decorrer desta tese e não tratados na versão anterior. De modo geral, esta versão apresenta as seguintes inovações:

- Possibilidade de criação de múltiplos perfis para um usuário. Assim, um mesmo usuário pode especificar várias configurações diferentes para os operadores e usar a que mais preferir.

- Cálculo inicial e ajuste autonômico do valor de similaridade para o operador Eliminador de Dados Similares. Com relação ao cálculo inicial, basicamente, cada operador testado no experimento com a versão Alfa da ferramenta fornece uma relação experimental entre similaridade e número de notícias retornadas. Esta relação, modelada por uma função, permitiu o cálculo inicial do valor de similaridade. Com relação ao ajuste deste valor, caso o número ou porcentagem do número de notícias retornadas seja diferente do solicitado pelo usuário, o valor de similaridade deste operador é ajustado, somando ou reduzindo por um determinado valor. Este processo ocorre até que o

número ou porcentagem de notícias atenda ao que foi solicitado pelo usuário. Para isto, foram utilizados os resultados obtidos com a versão Alfa da ferramenta.

- Bloqueio autonômico de URLs. Caso uma determinada URL gere um grande número de notícias descartadas em determinado tempo, esta URL é colocada autonomicamente na lista de URLs que terão suas notícias descartadas. Este processo utiliza os conceitos relacionados à detecção de eventos compostos, mais especificamente o Evento Composto Cumulativo. A especificação padrão do número de notícias e tempo de captura pode ser alterada pelo usuário.

- Adição dos operadores: Eliminador de Dados Irrelevantes e Eliminador de Dados Obsoletos. O primeiro operador é acessado quando uma URL, categoria ou lista negra é escolhida para ser descartada. O segundo operador é utilizado no descarte de notícias já lidas ou mais antigas que o número de dias especificado pelo usuário. Neste caso, as datas com diferentes fusos horários das notícias são convertidas para a data da máquina onde a ferramenta está sendo executada.

- Possibilidade de especificar detalhes adicionais dos operadores, tais como: se deverá ser realizado *stemming* e *stopwords* e se o título e/ou descrição da notícia serão considerados na operação.

- Inclusão de listas negras pré-configuradas, quais sejam: violência, pornografia, religião e política. As listas foram criadas com base no site www.thesaurus.com.

- Possibilidade de ordenar as notícias por data de publicação e por nível de similaridade com as palavras-chave escolhidas pelos usuários na interface de configuração dos operadores.

A Figura 60 apresenta a tela inicial da versão Beta da ferramenta Feed Organizer. Nesta tela o usuário pode escolher o perfil a ser utilizado, o tipo de configuração que deseja fazer (básica ou avançada) ou se deseja fazer a filtragem direta baseada em um perfil previamente configurado. Também é possível visualizar e alterar os endereços dos feeds que serão baixados. Estes endereços são capturados da interface do Firefox.

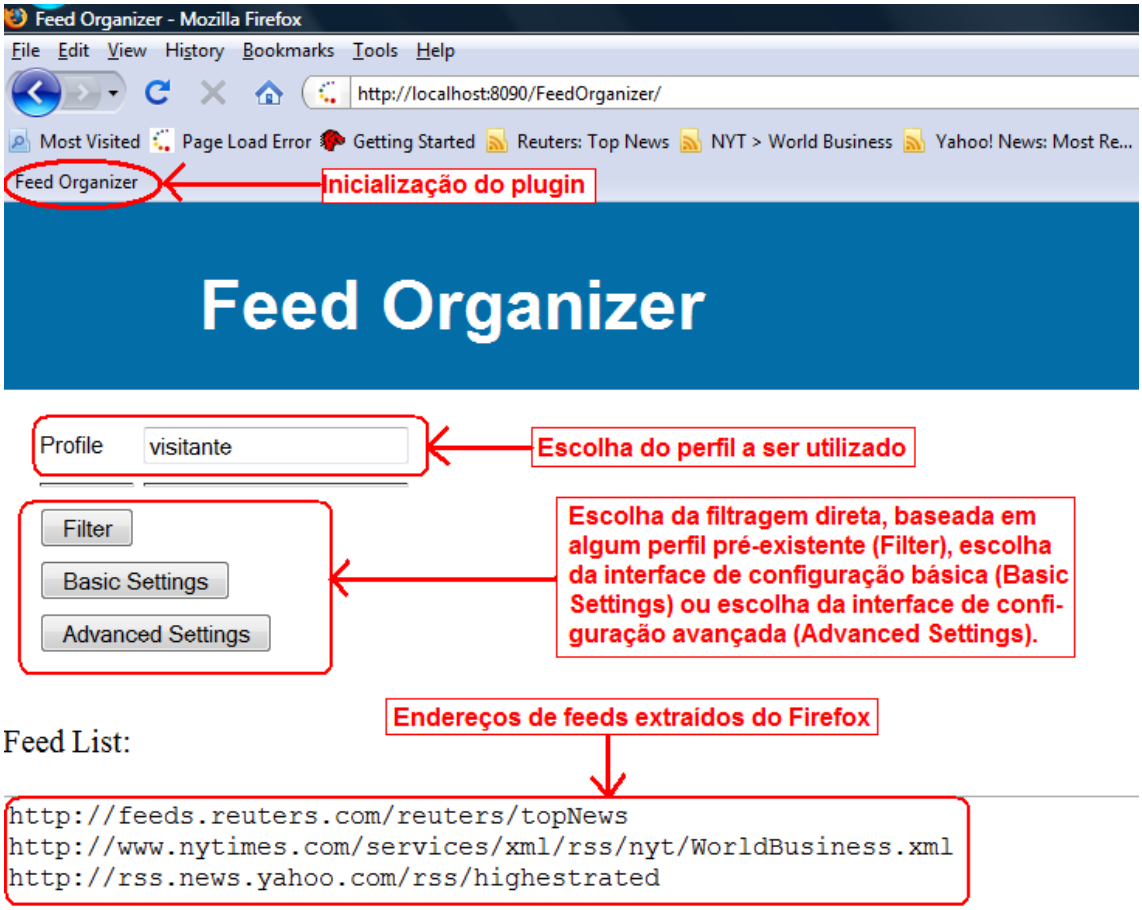

Figura 60 – Interface da Ferramenta *Feed Organizer*: Página Inicial

A Figura 61 mostra a interface de configuração básica desta ferramenta. Inicialmente, o usuário pode especificar se deseja processar notícias já lidas anteriormente e determinar a quantidade de notícias que deseja ler (em porcentagem ou valores absolutos). Em seguida, ele pode selecionar as listas negras que serão utilizadas na seleção das notícias. O usuário também pode digitar palavras-chave que contenham tópicos de interesse ou proibidos. Adicionalmente, ele pode estipular o tempo máximo (em dias), contados a partir da data atual, a partir do qual as notícias serão selecionadas, além de fornecer as URLs e categorias das notícias que não deseja ler. Finalmente, ele dever escolher a linguagem das notícias que serão filtradas e o tipo de ordenação que será usada na apresentação dos resultados. É importante ressaltar que vários detalhes de configuração dos operadores não são mostrados na interface de configuração básica. Alguns destes dados são configurados previamente e possuem valores fixos, outros são

fornecidos autonomicamente, a exemplo do cálculo do valor de similaridade dos operadores e definição endereços das notícias que serão bloqueados.

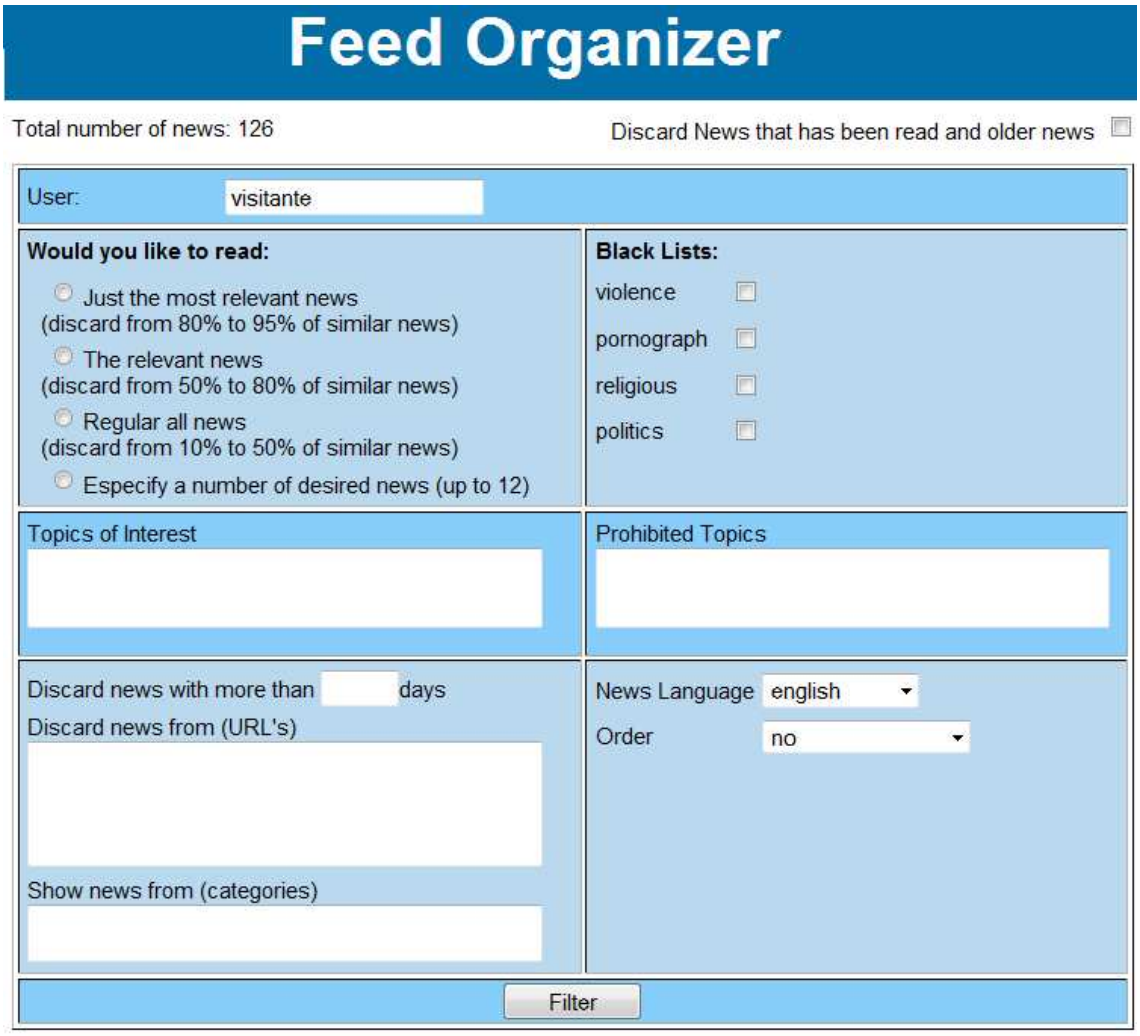

Figura 61 – Ferramenta *Feed Organizer*: Configuração Básica

A Figura 62 mostra a interface de configuração avançada da versão Beta. Nesta interface todos os detalhes de configuração são mostrados. De modo geral, ela permite configurar os mesmos filtros da interface de configuração básica, mas neste caso, o usuário pode especificar todos os parâmetros dos operadores.

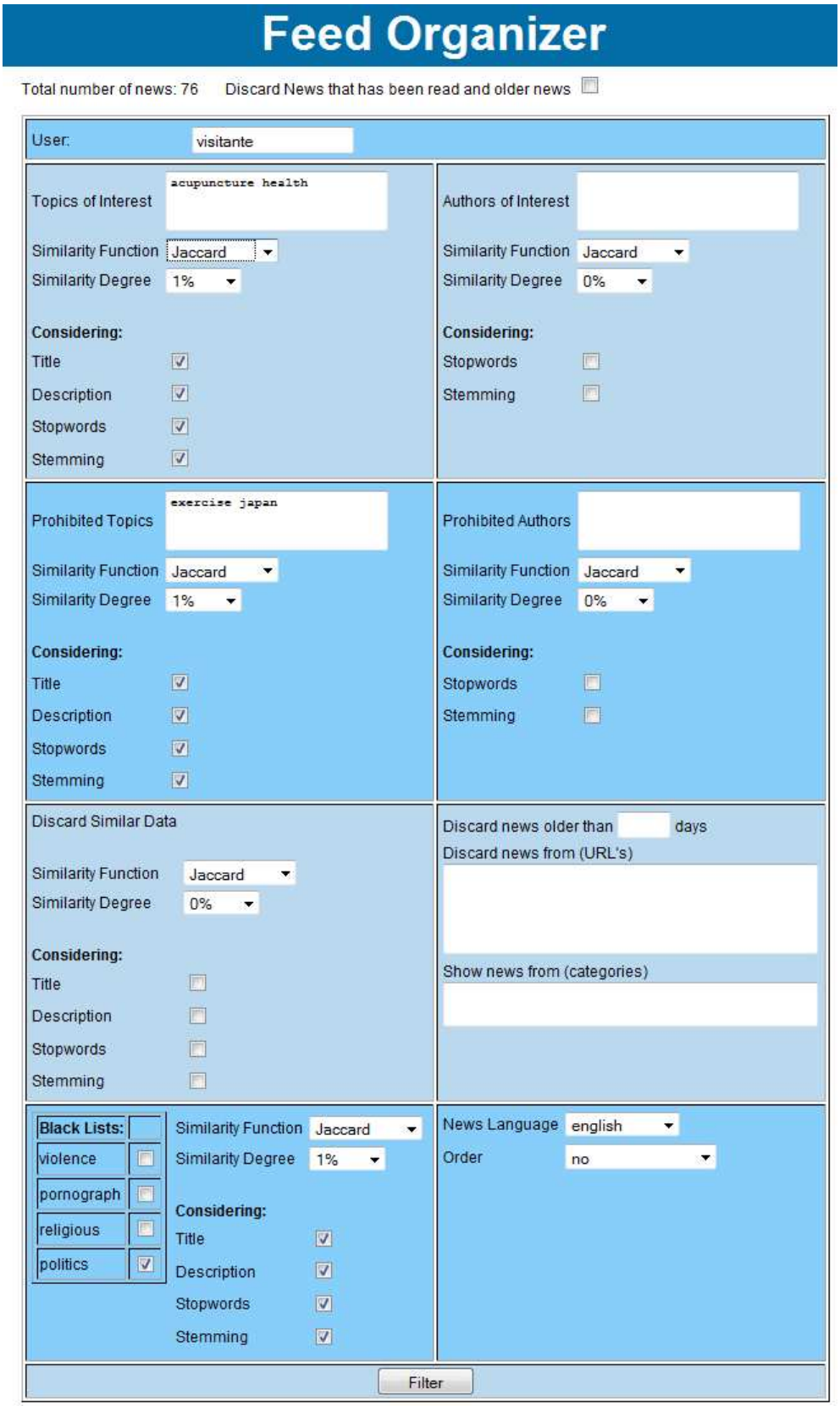

Figura 62 – Ferramenta *Feed Organizer*: Configuração Avançada

7.2.3 *Feed Organizer*: Funcionamento das Versões

O agregador de *feeds* do Firefox trata cada *feed* adicionado pelo usuário como uma entrada em seu conjunto de sites favoritos. A Figura 63 ilustra este cenário.

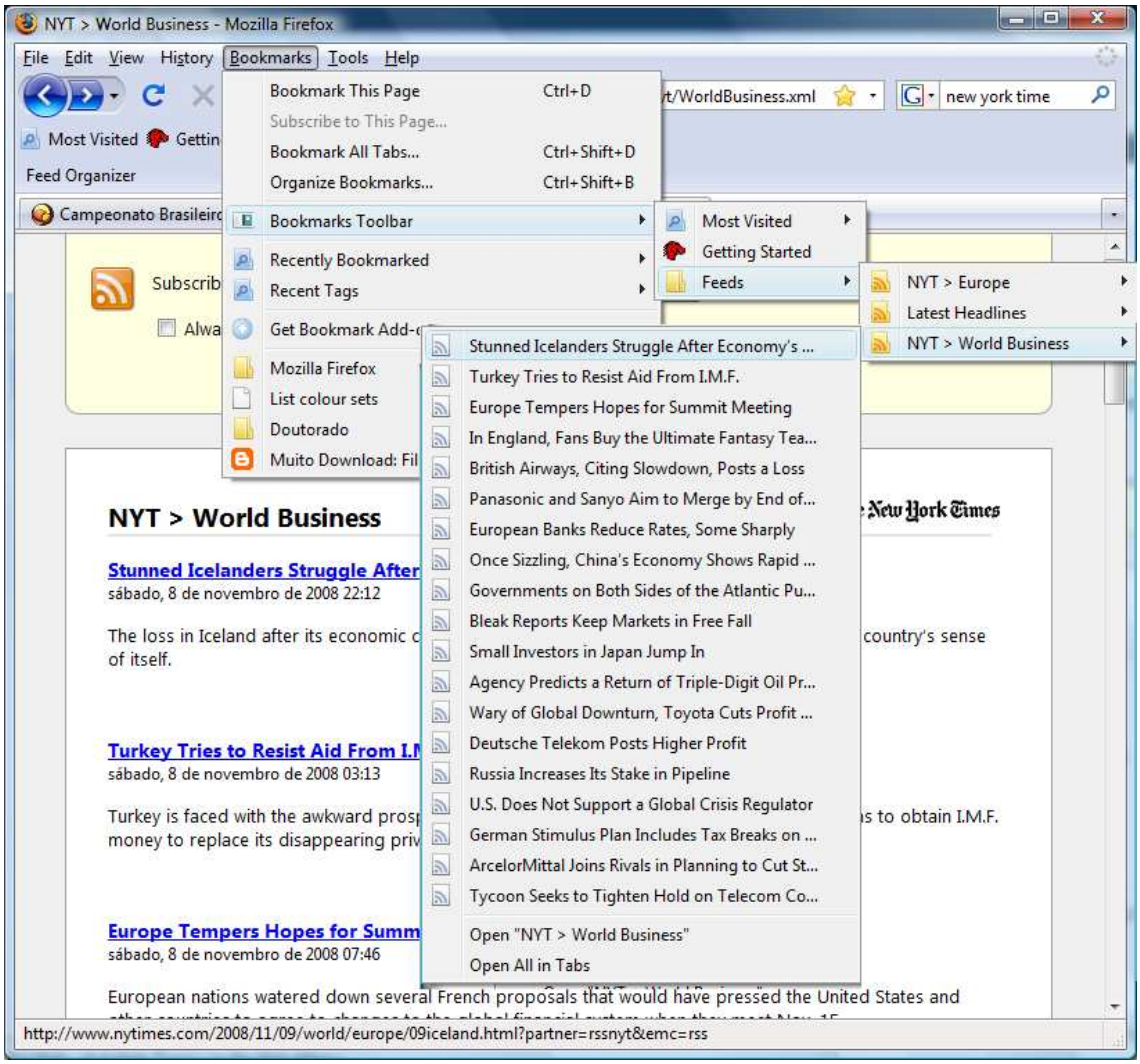

Figura 63 - Conjunto de *feeds* no Firefox

Dessa forma, quando o processo de filtragem é iniciado, o plugin *Feed Organizer* busca os endereços de todos os *feeds* existentes na aba Favoritos do usuário e envia estes dados para o servidor JBOSS que contém o aplicativo *FeedOrganizer.war* juntamente com as opções de filtragem selecionadas. Em seguida, este aplicativo, através dos endereços recebidos, faz o download de todos os arquivos XML, que correspondem aos *feeds* do usuário. A partir destes XMLs, ele monta um único arquivo XML com as informações de todos os *feeds* e as opções de filtragem definidas pelo usuário.

Antes da filtragem das notícias, o texto pode ser pré-processado. Neste caso, são eliminadas *stopwords* e é feita a extração dos radicais das palavras (*stemming*) baseadas no algoritmo de Porter (PORTER, 1997). Nesta tese, utiliza-se o algoritmo Snowball Stemmer e consideram-se apenas textos de notícias em inglês. Nos experimentos seguintes, é utilizada como medida de similaridade o coeficiente de Jaccard ou a busca por palavras-chave (BAEZA-YATES & RIBEIRO-NETO, 1999). Entretanto, outras estratégias poderiam ser facilmente incorporadas ao arcabouço.

Depois disso, o aplicativo realiza uma transformação (*parser*) do arquivo XML montado para o formato de entrada de dados do SGBD SECONDO de acordo com os padrões (operadores) utilizados. Assim, um novo arquivo é gerado, de acordo com este formato. Outro papel importante desempenhado por este aplicativo é a geração das consultas que deverão ser executadas no SECONDO com base nas opções de filtro selecionadas pelo usuário. Estas consultas, em conjunto com os dados gerados a partir do arquivo XML, são enviadas para o SECONDO via *sockets*. De modo a suportar tal funcionalidade, a interface gráfica do SECONDO foi alterada.

A sequência seguinte é o envio dessas informações para o servidor de RPAN. É neste servidor que as consultas são processadas de acordo com os padrões em RPAN armazenados previamente. Depois de processadas, essas informações são repassadas de volta para o SECONDO.

Por fim, o SECONDO recebe o resultado das consultas e os envia para o *FeedOrganizer.war*, que realiza uma nova transformação (*parser*) sobre o resultado que está no formato do SECONDO para o formato XML dos *feeds*. Este arquivo é enviado para o *plugin* (cliente) que exibe o resultado. A Figura 64 e a Figura 65 mostram exemplos de resultados deste processo.

Figura 64 – Resultado da filtragem: Versão Alfa

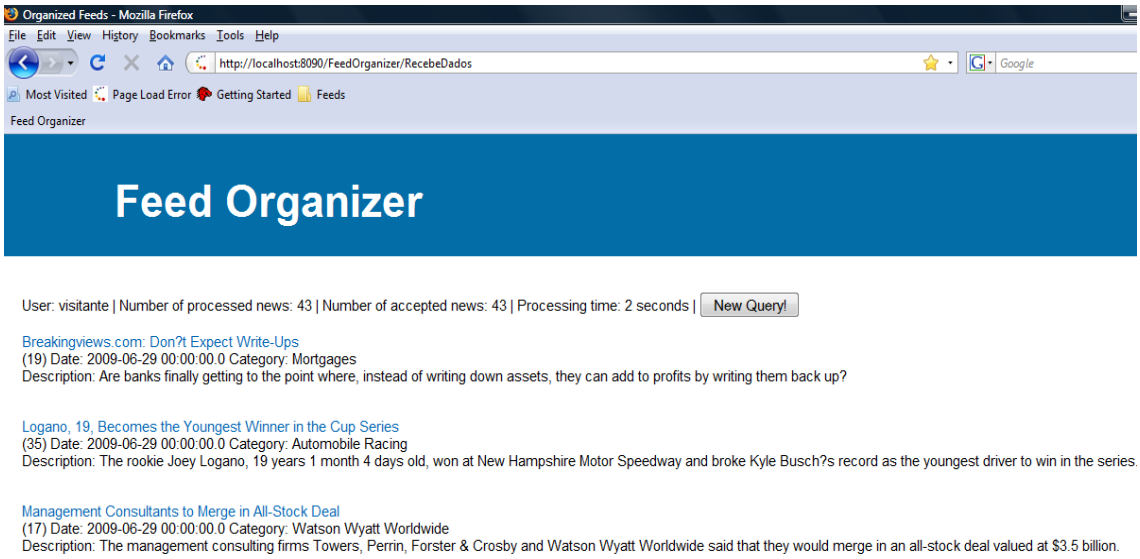

Figura 65 – Resultado da filtragem: Versão Beta

7.3 Experimentos

7.3.1 Visão Geral

Com o objetivo de avaliar a ferramenta *Feed Organizer* e, consequentemente, os padrões que esta ferramenta contempla, foram realizados dois conjuntos de experimentos, envolvendo a ferramenta *Feed Organizer*, versões Alfa e Beta. O primeiro conjunto de experimentos foi realizado no decorrer do mês de outubro de 2008 e o segundo conjunto de experimentos foi realizado no decorrer do mês de outubro de 2009. Os questionários, as respostas dos participantes e os dados consolidados, que foram utilizados nos experimentos, encontram-se nos anexos IV, V, VI, VII, VIII e IX.

Os planos dos experimentos foram adaptados da pesquisa de BARROS et al. (2002). De acordo com aquele trabalho, a realização de um estudo experimental geralmente pode ser dividida em cinco fases: Definição, Planejamento, Execução, Análise e Empacotamento do Estudo. A definição de cada uma dessas fases é contemplada nas próximas seções, bem como os procedimentos relativos aos experimentos realizados.

7.3.2 Definição

Consiste em resumir os objetivos do estudo, seu foco de qualidade e os objetos que serão analisados. Assim, tem-se:

- Objeto de Estudo – a ferramenta *Feed Organizer*, protótipo funcional que perpassa todas as camadas da arquitetura proposta para o Arcabouço Autonômico de Eliminação de Dados e representa uma das possíveis aplicações deste arcabouço.

- Objetivo – identificar a viabilidade da utilização da ferramenta desenvolvida como uma possível aplicação do arcabouço de eliminação de dados e comparar os resultados obtidos com as opiniões dos participantes e os resultados obtidos por outra ferramenta que possua características similares (Yahoo *Pipe*).

- Foco de Qualidade – os ganhos obtidos, medidos através da comparação dos resultados retornados pela ferramenta *Feed Organizer* com os resultados retornados pela ferramenta Yahoo *Pipe* e os resultados que os participantes consideram relevantes. Portanto, este estudo não está diretamente interessado no tempo necessário para o

processamento das consultas. Supõe-se que o processamento deve ocorrer em segundo plano, de modo a causar o mínimo de interferência nas ações rotineiras dos usuários (processamento autonômico da consultas).

- Perspectiva – o estudo será desenvolvido sob a ótica do pesquisador. É avaliada a viabilidade de utilização das técnicas de eliminação de dados, tendo em vista a continuidade do desenvolvimento das pesquisas relacionadas com estas técnicas.

- Contexto – demonstração da ferramenta *Feed Organizer*, segundo os preceitos definidos para o Arcabouço Autonômico de Eliminação de Dados, com o uso de dados provenientes da *Web* representando notícias.

7.3.3 Planejamento

O planejamento engloba a descrição do perfil dos participantes, a descrição dos instrumentos, do processo de execução e uma avaliação crítica dos problemas que podem ser encontrados ao longo desta execução. Assim, tem-se:

- Contexto global – as técnicas utilizadas no arcabouço autonômico de eliminação de dados formam uma das possíveis soluções para os problemas relacionados ao cenário de explosão de dados, combinando os conceitos de vários domínios, quais sejam: eventos, regras ativas, redes de Petri de alto nível e computação autonômica.

- Contexto Local – avaliação das técnicas de eliminação de dados propostas. Os resultados retornados pela ferramenta *Feed Organizer* que emprega essas técnicas serão comparados com as respostas da ferramenta Yahoo Pipe e as respostas que os participantes consideram relevantes.

- Treinamento – para ambos os conjuntos de experimento, o treinamento foi realizado em sala de aula ou laboratório, com duração aproximada de quarenta minutos. O treinamento foi composto de duas fases, que puderam ser interrompidas a qualquer momento pelos participantes para perguntas. Na primeira fase, foi realizada uma exposição sobre as técnicas propostas e a ferramenta Feed Organizer, utilizando um conjunto de transparências. Na segunda fase, foram apresentados exemplos de uso das técnicas de eliminação de dados, através dos operadores que executem estas funções. Os participantes precisaram fornecer dados (palavras-chave, funções de similaridade, níveis

de similaridade, entre outros), de modo a permitir que a ferramenta processasse as notícias.

- Participantes – para os experimentos com a versão Alfa da ferramenta, os participantes foram dezoito alunos do curso de graduação (oitavo período) da área de informática da COPPE/UFRJ, identificados no Anexo V. Para os experimentos com a versão Beta da ferramenta, os participantes foram trinta e três pessoas oriundas de diversas instituições e empresas, possuindo diferentes formações e níveis, conforme pode ser verificado no Anexo VIII. Os estudos foram realizados em dois ambientes acadêmicos (sala de aula e laboratório respectivamente) existentes na UFRJ, no IME e na Chemtech[51]. A capacidade de generalização deste estudo é discutida adiante, quando se avaliam as limitações e problemas que podem ser encontrados durante sua execução.

- Instrumentos – os participantes treinados na utilização das técnicas receberam questionários contendo perguntas para caracterização da sua formação e experiência, além da especificação das configurações necessárias pela ferramenta para o processamento das notícias. Para os experimentos com a versão Alfa da ferramenta, foi utilizado um conjunto de 90 notícias extraídas de três canais de comunicação: Herald Tribune, New York Times e Washington Post. Os arquivos de *feeds* de notícias foram extraídos nos dias 01 e 02 do mês de outubro de 2008. Foram consideradas as notícias de três domínios diferentes: negócios (*business*), esportes (*sports*) e notícias gerais dos EUA (US *news*). Os questionários relacionados a estes experimentos estão no Anexo IV. Para os experimentos com a versão Beta da ferramenta, foi utilizado um conjunto 400 notícias extraídas de seis veículos de comunicação diferentes: Herald Tribune, New York Times, The Guardian, Washington Post, The Sun e Financial Times. Os arquivos de *feeds* de notícias foram extraídos em dias e meses variados do ano de 2009. Foram consideradas as notícias de oito domínios diferentes: negócios (*business*), esportes (*sports*), cultura (*culture*), meio ambiente (*environment*), saúde (*health*), internet, política (*politic*) e tecnolologia (*technology*). Os questionários relacionados a estes experimentos são apresentados no Anexo VII.

- Critérios – o foco de qualidade do estudo exige critérios que avaliem os ganhos proporcionados pela utilização das técnicas de eliminação de dados. Os ganhos obtidos pela utilização das técnicas serão avaliados quantitativamente. Esta análise tem

[51] http://www.chemtech.com.br/lportal/web/guest/home

o objetivo de avaliar a ferramenta, seus pontos fontes, fracos e possibilidades de melhorias futuras.

- Variáveis Independentes – os dados pessoais, experiência com uso de *feeds* e os tipos de agregadores de *feeds* utilizados são informações independentes coletadas durante o estudo.

- Variáveis Dependentes – todas as demais variáveis (por exemplo, notícias e palavras-chave relevantes) são dependentes.

- Capacidade Aleatória – pode ser exercida na seleção dos veículos de comunicação de onde foram extraídas as notícias, na seleção das próprias notícias, na escolha dos participantes do estudo e na distribuição dos objetos de análise entre eles. Foram dois objetos de análise diferentes, relativos à versão Alfa e a versão Beta da ferramenta, bem como foram dois grupos que realizaram a análise destes objetos. O objeto da análise do primeiro grupo foi o mesmo para todos os participantes deste grupo, ou seja, a versão Alfa da ferramenta. O mesmo ocorreu para o objeto de análise do segundo grupo. Os participantes do primeiro grupo foram alunos da disciplina de Banco de Dados II oferecida pela linha de Banco de Dados do PESC/UFRJ. O segundo grupo foi composto de pessoas de diferentes áreas e formações. Isto indica uma amostra aleatória da população descrita no item “Participantes”. Outro fator relevante é que os canais de comunicação de onde foram extraídas as notícias deveriam possuir uma característica; disponibilizar *feeds* em inglês, sendo que qualquer dos canais de comunicação disponível na *Web* que apresentasse esta característica poderia ter sido escolhido. Além disso, as notícias destes canais foram extraídas em dias e meses escolhidos aleatoriamente. Isto indica uma amostra aleatória dos canais e notícias descritos no item “Instrumentos”. Dessa forma, considera-se que tanto as amostras dos participantes dos experimentos quanto às amostras das notícias analisadas pelos participantes são amostras aleatórias, sendo, portanto, representações realistas e não tendenciosas de suas populações completas (SALVATORE & REAGLE, 2002, NETO et al., 2007).

- Classificação em Bloco – os participantes dos experimentos foram divididos em dois blocos: participantes que avaliaram a versão Alfa e participantes que avaliaram a versão Beta da ferramenta. Entretanto, a coleta de dados sobre experiência dos

participantes com o uso de *feeds* de notícias e os tipos de agregadores utilizados permite que estes blocos sejam subdivididos e organizados durante a análise dos dados.

- Balanceamento – não houve necessidade de balanceamento, pois os objetos de análise (formulários e notícias) foram diferentes para cada um dos grupos e, para cada objeto de análise, os participantes formaram um único bloco de avaliação, sendo que as análises das ferramentas foram feitas em separado.

- Mecanismo de Análise – as variáveis dependentes serão apresentadas utilizando-se as escalas próprias de cada variável. Além disto, os resultados serão discutidos utilizando-se por base a classificação de relevância das notícias, obtida a partir da opinião dos participantes, e a comparação dos resultados obtidos com a ferramenta *Feed Organizer* e Yahoo *Pipe*.

- Validade Interna do Estudo – é a capacidade de um novo estudo repetir o comportamento do estudo atual com os mesmos participantes e objetos com que ele foi realizado. Logo, os procedimentos utilizados, os dados de identificação dos participantes e questionários utilizados foram devidamente registrados e organizados. Vale ressaltar que a validade interna do estudo é dependente do número de participantes executando o mesmo. 18 participaram do experimento com a versão Alfa da ferramenta e 33 pessoas participaram dos experimentos com a versão Beta da Ferramenta, o que garante um bom nível de validação interna. Certamente, um número maior de participantes melhoraria esta validação. Outro ponto que pode influenciar o resultado do estudo é a troca de informações entre os participantes que já realizaram o estudo e os que não o realizaram. Para evitar este problema, foi requisitado explicitamente que os participantes não trocassem informações a respeito da ferramenta e dos questionários.

- Validade Externa do Estudo – mede sua capacidade de refletir o mesmo comportamento em outros grupos de participantes e profissionais, ou seja, em outros grupos além daquele em que o estudo foi aplicado. Um dos problemas enfrentados é que alguns participantes apresentaram pouco interesse em participar do estudo, não respondendo a todas as questões ou respondendo de forma desleixada. Entretanto, como a maioria, aparentemente, demonstrou-se motivada e respondeu a todos os questionários, a validade externa do estudo foi considerada suficiente.

- Validade de Construção do Estudo – refere-se à relação entre os instrumentos e participantes do estudo e a teoria que está sendo provada por este. É

importante ressaltar que a aplicação foi feita sobre uma área amplamente conhecida, que não exigiu conhecimento prévio específico, ou seja, a área de notícias *Web*. Isto, de certa forma, reduz o efeito da experiência dos participantes e evita que experiências anteriores gerem uma interpretação incorreta do procedimento a ser adotado.

- Validade de Conclusão do Estudo – mede a relação entre os tratamentos e os resultados, determinando a capacidade do estudo em gerar alguma conclusão. O estudo utilizará medidas objetivas, o que neutraliza a influência humana sobre os dados apurados e analisados. Isto não gera grandes dificuldades em relação à capacidade de conclusão do estudo, visto que os seus resultados são diretos.

7.3.4 Execução

A execução permite a realização do estudo experimental pelos participantes, utilizando os instrumentos e os processos definidos no planejamento. As seguintes etapas foram executadas:

- Seleção dos Participantes – os participantes foram selecionados, conforme descrito no item “Participantes” da seção anterior. Os participantes do experimento são apresentados nos Anexos V e VIII.

- Instrumentação – foram distribuídos questionários, conforme descrito no item “Instrumentos” da seção anterior.

- Procedimentos de Participação – o estudo investigativo foi dividido em duas etapas. Na primeira parte, os participantes receberam um treinamento sobre as técnicas utilizadas na ferramenta *Feed Organizer*. Na segunda parte, eles receberam questionários em papel ou meio digital, devolvendo-os preenchidos.

- Execução – O treinamento dos participantes durou cerca de quarenta minutos. Depois os formulários foram distribuídos e os participantes puderam preenchê-los. O prazo máximo para a entrega dos formulários preenchidos foi de sete dias. As respostas fornecidas pelos participantes, bem como os resultados obtidos são fornecidos nos Anexos V, VI, VIII e IX.

7.3.5 Análise dos Resultados

A fase de análise dos resultados realiza a organização dos resultados gerados pelos participantes durante a execução e discute as inferências feitas sobre estes resultados.

Em virtude das diferentes características de filtragem e padrões de descarte de dados utilizados, a análise dos resultados foi dividida segundo os seguintes critérios:

1) Análise dos Dados Genéricos – realiza a análise dos dados gerais sobre a experiência dos participantes com *feeds* e seus agregadores (Seção 7.3.5.1).

2) Análise dos padrões Dados Permitidos e Eliminador de Dados Proibidos – executa a análise dos resultados obtidos a partir da comparação das respostas dos participantes com os resultados obtidos pela ferramenta *Feed Organizer* versão Alfa para os padrões Dados Permitidos e Eliminador de Dados Proibidos (Seção 7.3.5.2).

3) Análise do padrão Eliminador de Dados Similares – perfaz a análise dos resultados obtidos a partir da comparação das respostas dos participantes com os resultados obtidos pela ferramenta *Feed Organizer* versão Alfa para o padrão Eliminador de Dados Similares (Seção 7.3.5.3).

4) Análise Comparativa da Ferramenta *Feed Organizer* Versão Beta e da Ferramenta Yahoo *Pipe* – realiza a análise dos resultados obtidos a partir da comparação das respostas dos participantes com os resultados obtidos pela ferramenta Yahoo *Pipe* e pela ferramenta *Feed Organizer* versão Beta para os padrões Dados Permitidos e Eliminador de Dados Proibidos (Seção 7.3.5.4).

5) Análise das Características Autonômicas da Ferramenta – procede a análise de duas características autonômicas da ferramenta Feed Organizer: o cálculo inicial e ajuste autonômico do valor de similaridade, e o bloqueio autonômico de URLs (seções 7.3.5.5 e 7.3.5.6).

7.3.5.1 Análise dos Dados Genéricos

Esta análise considera os dados genéricos sobre o universo dos *feeds* de notícias coletados com os participantes dos experimentos. No total, considerando os dois experimentos, cinqüenta e uma pessoas foram ouvidas.

Com relação à experiência dos participantes do experimento com a versão Alfa da ferramenta, dos dezoito participantes, nove declararam que não conheciam, utilizavam ou tinham experiência com *feeds* de notícias, os outros nove declararam possuir alguma experiência. Dos nove participantes que disseram ter alguma experiência, quatro declararam preferir a leitura de notícias de interesse geral e cinco declararam preferir a leitura de notícias específicas por assunto. Adicionalmente, sete participantes disseram utilizar o navegador Mozilla Firefox como agregador de *feeds* e os outros dois declararam utilizar o igoogle, sendo a nota média desses agregadores igual a 5,4 (cinco vírgula quatro), sendo o máximo possível igual a 10. Além disso, estes alunos declararam que no último período de quinze dias receberam em média 3,4 notícias repetidas/parecidas.

Com relação à experiência dos participantes do experimento com a versão Beta da ferramenta, dos trinta e três participantes, doze participantes relataram que não conheciam, utilizavam ou tinham experiência com *feeds* de notícias, dezessete participantes declararam possuir alguma experiência e quatro não responderam. Dos dezessete participantes que disseram ter alguma experiência, nove declararam preferir a leitura de notícias de interesse geral, cinco declararam preferir a leitura de notícias específicas por assunto e dois declararam assinar sites de *feeds* indiscriminadamente. Dez participantes disseram utilizar o navegador Mozilla Firefox como agregador de *feeds* e quatro disseram utilizar o Google Reader. Os agregadores Yahoo *Pipe*, Akregator, Internet Explorer, Mail da Apple, Safari e Thunderbird foram escolhidos, cada um, por somente um participante. A nota média desses agregadores foi igual a 6,7 (seis vírgula sete), sendo o máximo possível igual a 10. Os participantes também informaram que, no último período de quinze dias, receberam em média 5,9 notícias repetidas/parecidas.

Como os experimentos foram realizados com uma diferença de praticamente um ano, pode-se realizar algumas análises comparativas temporais entre os experimentos:

- Considerando as pessoas que responderam a pergunta relativa ao uso ou experiência prévia com feeds, o índice aumentou de 50% para 58,6%. Isto indica um leve aumento no número de pessoas que utilizam feeds.

- Os números indicam que o Mozilla Firefox é o agregador de *feeds* preferido pela maioria dos participantes em ambas as pesquisas, alcançando os níveis de 38,9% e 33,3% respectivamente.

- A nota média dada aos agregadores de *feeds* melhorou de 5,4 para 6,7, o que pode indicar uma evolução dessas ferramentas.

- O número médio de notícias repetidas no período de quinze dias que antecedeu às pesquisas aumentou de 3,4 para 5,9. Isto indica que a distribuição de notícias repetidas/similares feita pelos diferentes distribuidores de *feeds* tem aumentado. Isto provavelmente ocorreu porque o número de distribuidores de *feeds* tem aumentado, sendo que muitos desses sites simplesmente redistribuem as notícias dos sites mais renomados.

De modo geral, pode-se considerar que os *feeds* de notícias *Web* estão, pouco a pouco, sendo mais utilizados pelas pessoas, principalmente através do agregador Mozilla Firefox. Outro ponto importante é que a percepção do número de notícias repetidas ou similares tem aumentado, o que reveste as pesquisas referentes às ferramentas de filtragem de *feeds* de uma importância ainda maior. Obviamente, estas tendências são baseadas em apenas duas amostras temporais, sendo que, para obter um resultado mais preciso, novas amostras serão necessárias.

7.3.5.2 Análise dos padrões Dados Permitidos e Eliminador de Dados Proibidos

Esta análise (PINHEIRO et al., 2009a, PINHEIRO et al., 2009b) considera os resultados retornados de acordo com diferentes níveis de similaridade, além de se verificar a concordância das palavras retornadas pela ferramenta versão Alfa e as respostas esperadas pelos participantes do experimento.

Os anexos IV e V apresentam os questionários contendo o conjunto de 90 notícias utilizadas neste experimento. Foi pedido aos participantes do experimento que realizassem buscas de notícias usando os padrões Dados Permitidos e Eliminador de Dados Proibidos. Para o primeiro, o participante deveria selecionar palavras que ele

gostaria de encontrar nas notícias. Para o segundo, ele deveria escolher palavras que ele não desejaria encontrar nas notícias.

A Figura 66 e a Figura 67 mostram as curvas relativas à média do número de notícias retornadas pela ferramenta Feed Organizer por nível de similaridade [52], utilizando por base as palavras-chave escolhidas pelos participantes. Essas são as curvas características dos padrões Dados Permitidos e Eliminador de Dados Proibidos e estão de acordo com os resultados esperados, visto que na medida em que o nível de similaridade aumenta para o padrão Dados Permitidos o número de notícias retornadas diminui, enquanto que para o padrão Eliminador de Dados Proibidos acontece o oposto. Estes padrões podem ser vistos como complementares, pois um padrão retorna os resultados que são descartados pelo outro, caso seja usado o mesmo conjunto de termos e parâmetros.

Pode-se verificar que o número de respostas retornadas pelo padrão Dados Permitidos é relativamente baixo (menos de 20% de respostas retornadas), indicando que dentro do conjunto de notícias, poucas eram de interesse para os participantes, pois não coincidiam com o conjunto de palavras-chave escolhidas. O domínio Esportes foi o que, a princípio, apresentou menor número de notícias de interesse, provavelmente, porque as notícias (em inglês) do experimento retratam esportes pouco conhecidos ou que despertam pouco interesse aos participantes do experimento.

Para o padrão Eliminador de Dados Proibidos, é possível verificar que a seleção é pouco restrita (o percentual máximo é de 100% - 13% = 87%), pois a intenção é apenas filtrar assuntos proibidos, permitindo todo o resto. Observa-se para este padrão que o domínio Esportes apresentou o maior nível de rejeição por parte dos participantes, o que comprova a falta de interesse pelos esportes noticiados pelos canais escolhidos (esportes tipicamente relacionados à Inglaterra e aos Estados Unidos).

[52] Nível de similaridade considera o nível mínimo de similaridade entre textos para o valor definido,ou seja, o ponto de corte para textos similares. Dessa forma, valores maiores também são retornados, por exemplo se o nível de similaridade for 5%, os textos que forem 10% similares também serão retornados.

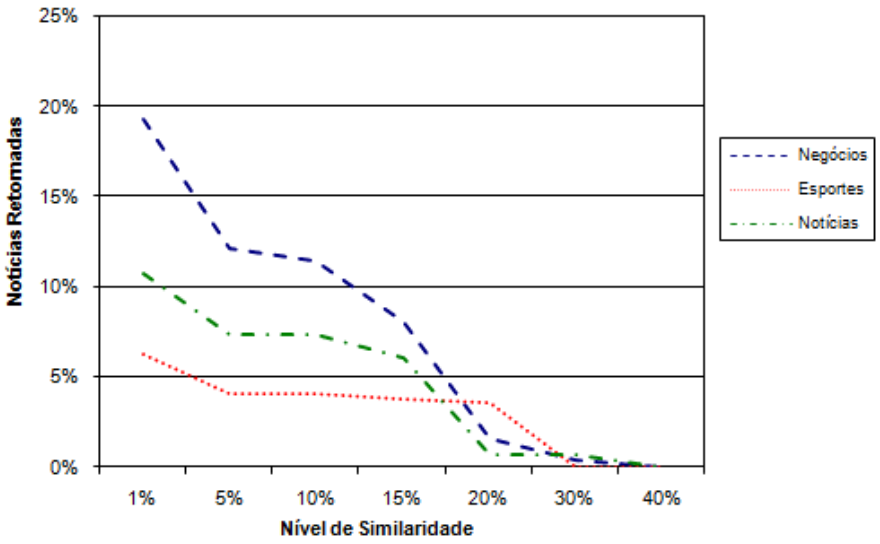

Figura 66 – Curva característica do padrão Dados Permitidos

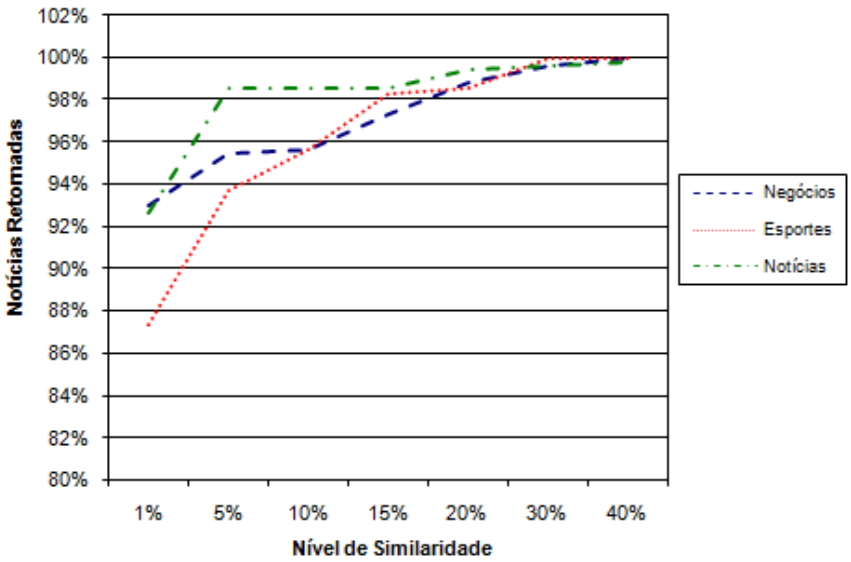

Figura 67 – Curva característica do padrão Eliminador de Dados Proibidos

A Figura 68 apresenta a média das precisões obtidas pelas pesquisas de todos os participantes do experimento por nível de similaridade para o padrão Eliminador de Dados Proibidos. A precisão pode ser definida pela quantidade de respostas retornadas que são relevantes sobre o número de respostas retornadas (BAEZA-YATES & RIBEIRO-NETO, 1999), ou seja, no caso das notícias, é dada pela quantidade de notícias relevantes retornadas sobre o número de notícias retornadas. Pode-se observar que para obter altos valores de precisão para o padrão Eliminador de Dados Proibidos devem ser escolhidos baixos valores de similaridade, entretanto obtém-se um número

maior de notícias a serem lidas. Aumentando o nível de similaridade fica mais difícil que um conjunto de palavras dentro da notícia coincida com os termos escolhidos pelos participantes, pois todos os termos devem estar presentes, ao passo que, com baixo nível de similaridade, são necessários que menos termos sejam coincidentes.

No caso do padrão Eliminador de Dados Proibidos, a precisão não é menor que 91% o que indica que os participantes usaram termos isolados ou poucas combinações de termos, comparado ao universo de palavras existentes nas notícias. De fato, a média de palavras-chave selecionadas pelos participantes foi cinco. Comparando os três domínios analisados, o domínio Notícias foi o que retornou mais dados de interesse dos participantes do experimento. Os resultados revelam que, para este domínio, houve um melhor casamento entre as notícias existentes e as palavras escolhidas pelos participantes.

No caso do padrão Dados Permitidos, a precisão é sempre 100% para qualquer nível de similaridade. Isto acontece porque o padrão recupera somente notícias relevantes, pois estão de acordo com as palavras-chave escolhidas pelos participantes[53].

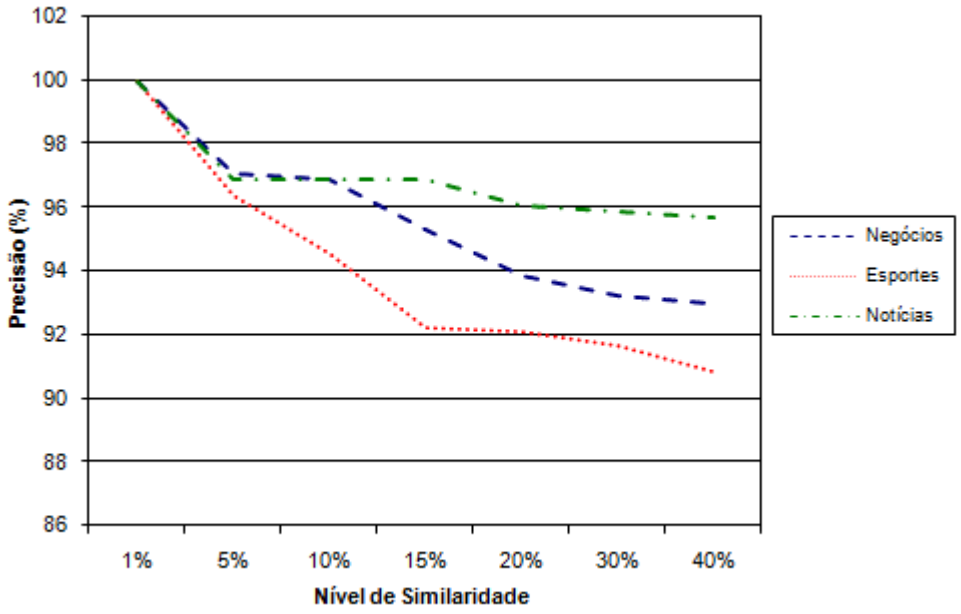

Figura 68 – Precisão do padrão Eliminador de Dados Proibidos para os diversos níveis de similaridade

A Figura 69 apresenta a média das coberturas de todos os participantes do experimento por nível de similaridade para o padrão Dados Permitidos. A cobertura[54]

[53] Neste experimento a ocorrência de uma palavra-chave está relacionada à relevância da notícia.

pode ser definida pela quantidade de respostas retornadas que são relevantes sobre o número de respostas relevantes (BAEZA-YATES & RIBEIRO-NETO, 1999), ou seja, é dada pela quantidade de notícias relevantes retornadas sobre o número de notícias relevantes. Pode-se observar que para obter altos valores de cobertura para o padrão Dados Permitidos devem ser escolhidos baixos valores de similaridade, entretanto, obtém-se um número maior de notícias a ser lido.

Comparando os três domínios, observa-se que o domínio Notícias apresentou, de modo geral, a pior cobertura. Isto pode ser explicado porque os participantes escolheram assuntos mais diversificados que cobriram um maior número de notícias deste domínio, sendo que estas notícias não foram efetivamente selecionadas pelo operador.

No caso do operador Eliminador de Dados Proibidos a cobertura será sempre 100% para qualquer nível de similaridade, pois este operador visa remover as notícias não relevantes, segundo a busca do usuário[55]. O caso mais restrito, com o menor número de respostas retornadas, já contém todas as respostas relevantes. O que acontece quando se aumenta o nível de similaridade é que mais notícias irrelevantes são incluídas.

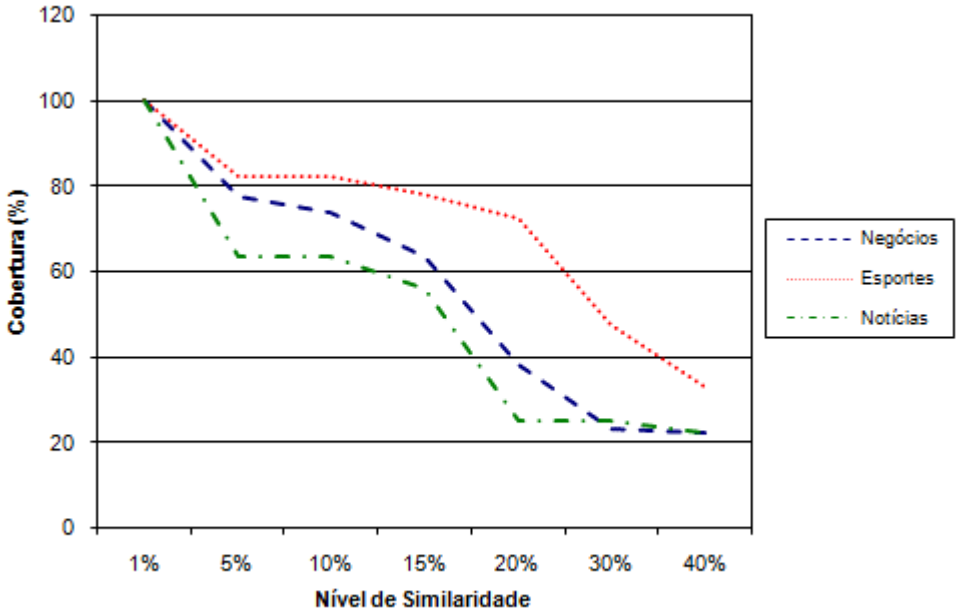

Figura 69 – Cobertura do padrão Dados Permitidos para os diversos níveis de similaridade

[54] Revocação é um termo sinônimo também encontrado na literatura.
[55] Neste experimento a não ocorrência de todas as palavras-chave selecionadas pelo usuário indica que a notícia é relevante.

7.3.5.3 Análise do padrão Eliminador de Dados Similares

Esta análise (PINHEIRO et al., 2010) considera o experimento realizado com a versão Alfa da ferramenta. O padrão Eliminador de Dados Similares compara as notícias uma a uma usando a medida de similaridade de Jaccard. Se as notícias forem consideradas similares para o nível de similaridade escolhido, então a notícia mais antiga é eliminada. Isto pode ser visto como um agrupamento[56] baseado no método de Jaccard das notícias para um determinado nível de similaridade em que somente a notícia mais recente de cada grupo permanece. À medida que o nível de similaridade aumenta, o número de grupos também aumenta. Por exemplo, se for considerado o nível de similaridade de 0%, restará apenas um grupo, pois todas as notícias são no mínimo 0% similares. Já para um nível de similaridade de 100%, se não houver notícias idênticas, existirão tantos grupos quanto o número de notícias.

Os anexos IV e VI apresentam os questionários contendo o conjunto de 90 notícias, bem como o resumo dos dados utilizados neste experimento. Para avaliar o operador Eliminador de Dados Similares, o experimento foi dividido em três etapas.

A primeira etapa do experimento envolveu os usuários da ferramenta. Foi solicitado aos participantes do experimento que determinassem para cada notícia as outras notícias que eles considerassem similares dentro de cada domínio. Com isso, obtiveram-se as notícias consideradas similares para cada usuário. Estas medidas são utilizadas para gerar o agrupamento das notícias com base na opinião dos participantes.

A segunda etapa consistiu em executar filtragens de notícias utilizando a ferramenta *Feed Organizer* com base no padrão Eliminador de Dados Similares. Foi observado que o número de notícias variava muito na faixa de 1% a 10% de similaridade. Portanto, para estes valores, a variação de similaridade foi feita a intervalos de 1%, sendo adotados os seguintes níveis de similaridade (%): 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 30, 40 e 50. Para cada nível foi registrado o conjunto de notícias retornado. Isto é mostrado no Anexo VI. É importante considerar que o padrão Eliminador de Dados Similares somente mantém a notícia mais recente de cada grupo criado para um determinado nível de similaridade.

A Figura 70 mostra a curva característica do padrão Eliminador de Dados Similares. Essas curvas mostram o percentual de notícias descartadas (notícias

[56] *Clustering* é o termo correspondente a agrupamento em inglês.

consideradas similares) por nível de similaridade. Os resultados estão de acordo com o comportamento esperado para o padrão, tendo em vista que na medida em que o nível de similaridade aumenta, o número de notícias descartadas diminui, pois o critério de comparação das notícias torna-se mais rigoroso.

O domínio Notícias apresenta, de modo geral, menor número de notícias descartadas para os diferentes níveis de similidade. Isto se deve ao fato deste domínio apresentar um menor número de notícias similares que os outros dois domínios, considerando o universo de notícias do experimento.

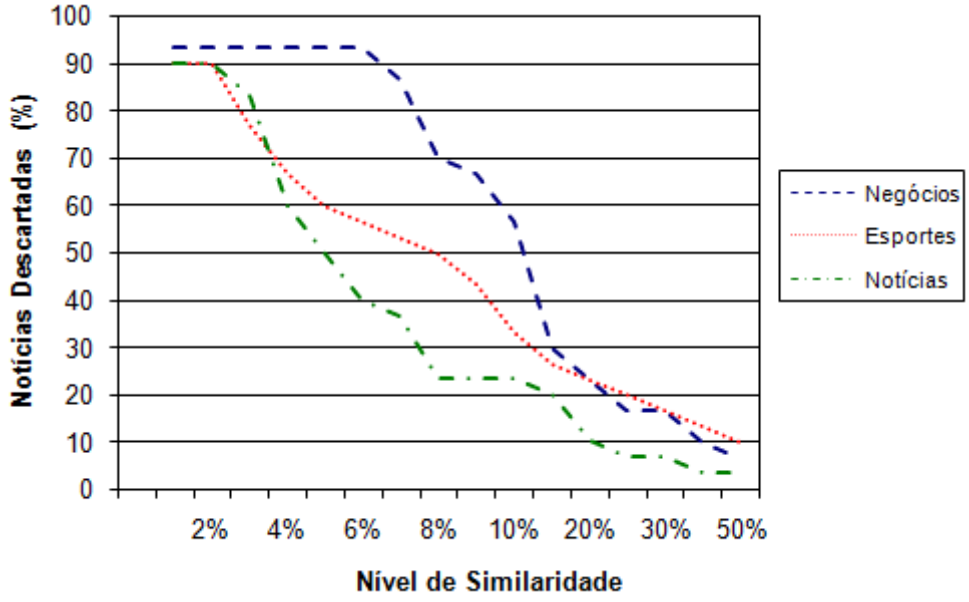

Figura 70 – Curva característica do padrão Eliminador de Dados Similares

O padrão mantém somente a notícia mais recente de cada grupo, sendo assim, baixos valores de similaridade por possuírem, normalmente, grupo com mais elementos terão menos notícias retornadas e mais notícias descartadas.

A terceira etapa consistiu na comparação dos resultados obtidos pela ferramenta com os resultados obtidos a partir da opinião dos participantes. Isto foi feito a partir do agrupamento das informações fornecidas pelos participantes, usando matrizes de similaridade, comparado com os resultados obtidos automaticamente pela ferramenta. Nesta comparação, três matrizes de similaridade (30 x 30) foram montadas, correspondendo a cada um dos domínios definidos. Isto é mostrado no Anexo VI. Cada elemento dessas matrizes corresponde ao somatório de vezes que os alunos consideraram similar o par de notícias que este elemento conecta. Por exemplo, se o

elemento que liga a *coluna1* a *linha2* tem valor dois, então isso indica que dois participantes acharam que as notícias 1 e 2 são similares.

Para determinar os conjuntos de notícias similares (grupos), segundo a opinião dos participantes, escolheu-se o método de agrupamento pelo vizinho mais próximo (*nearest neighbor clustering*) (BEYGELZIMER et al., 2006). Para cada um dos domínios, foi utilizado este algoritmo sobre as matrizes normalizadas e os resultados foram convertidos em dendogramas que são mostrados a seguir. Bons resultados são caracterizados por notícias retornadas pela ferramenta (notícias não similares) ocupando grupos distintos (*UCG*) e poucos grupos não ocupados (*AE*).

Para determinar a melhor combinação entre a quantidade de grupos e o nível de similaridade, foram criadas duas medidas que apoiam este processo e que são mostradas a seguir:

UCG = RCLUSTERS / CLUSTERS;

AE = 1 – (RCLUSTERS / NNR).

Onde:

- *UCG* – uso dos grupos (*clusters*) gerados;
- *RCLUSTERS* – número de grupos gerados com pelo menos uma notícia retornada;
- *CLUSTERS* – número total de grupos (*clusters*) gerados;
- *AE* – alocações erradas, ou seja, número de grupos que têm mais de uma notícia;
- *NNR* – número de notícias retornadas para um determinado nível de similaridade.

Analisando as medidas mostradas anteriormente, fica claro que devem ser buscados altos valores de *UCG* e baixos valores de *AE*. Intuitivamente, a melhor situação é aquela em que todos os grupos de um determinado nível de similaridade devem ser retornados com apenas um elemento em cada grupo. Pontos ótimos da curva são aqueles em que *(UCG – AE)* é o maior valor possível. Estes pontos ótimos devem ser calculados para cada medida de similaridade considerada.

Por exemplo, na Figura 71, que apresenta o dendograma relativo ao domínio Negócios, a linha vertical mais escura indica o ponto ótimo de divisão dos grupos (23

grupos) para o nível de similaridade de 20%. Os números em negrito representam as notícias retornadas pela ferramenta este nível de similaridade, sendo seu total igual a 23 notícias. Para este caso, o *UCG* foi igual a 96%, pois um grupo não foi retornado e o *AE* foi 4%, pois um grupo teve uma alocação indevida (mais de uma notícia no mesmo grupo).

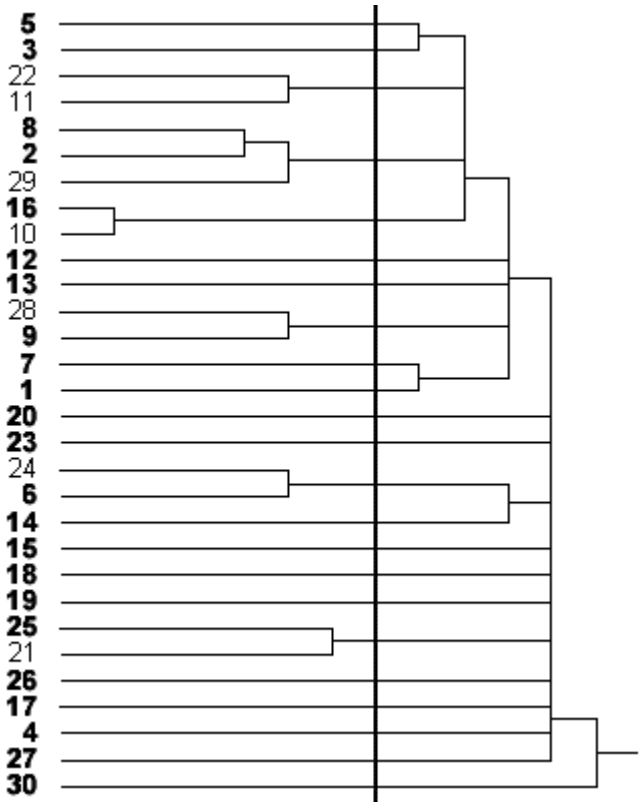

Figura 71 – Dendograma do domínio negócios com linha vertical indicando a melhor divisão dos grupos para 20% de similaridade

A Figura 72 mostra o gráfico dos resultados de *AE* e *UCG* para os níveis de similaridades de 1% até 50% do domínio Negócios. Analisando o gráfico é possível ver que, conforme aumenta o nível de similaridade, melhores ficam os resultados, ou seja, mais próximo do resultado requerido pelos participantes. Para o nível de similaridade de 50%, por exemplo, o *UCG* foi igual a 97% e o *AE* foi 0%, mostrando com isso que a opinião do usuário e os resultados obtidos pela ferramenta são convergentes.

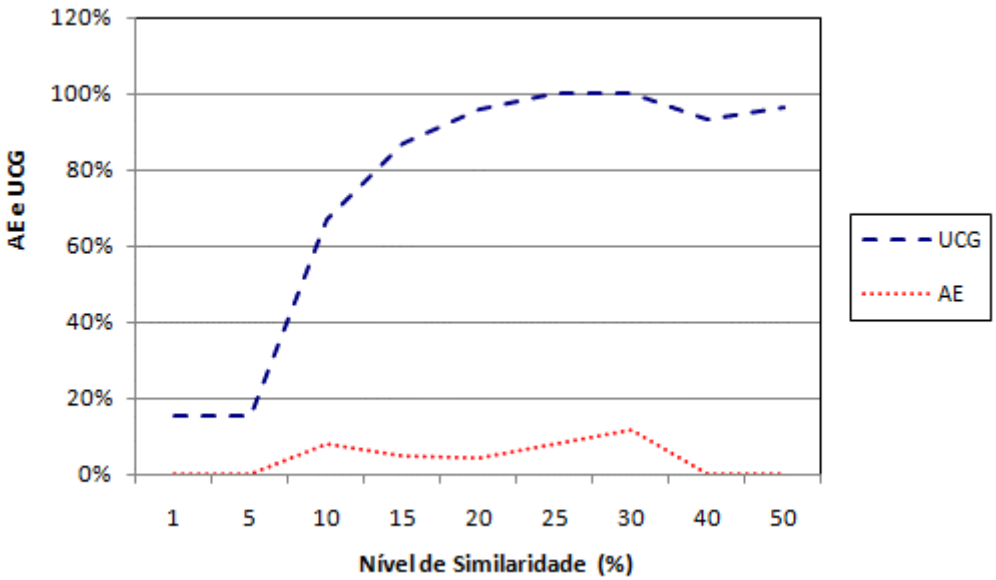

Figura 72 – Resultado de *AE* e *UCG* para o domínio negócios

A Figura 73 apresenta o dendograma do domínio Esportes. Utilizando o mesmo procedimento adotado para analisar a Figura 71, é possível verificar que, para um nível de similaridade de 15%, a linha vertical escura indica o corte ótimo dos grupos, gerando 23 grupos diferentes. Os números em negrito indicam as notícias retornadas pela ferramenta para este nível de similaridade, sendo seu total igual a 24 notícias. Para este caso, o *UCG* foi igual a 96% e o *AE* foi 8%.

A Figura 74 mostra o gráfico dos resultados de *AE* e *UCG* para os níveis de similaridades de 1% ate 50% do domínio Esportes. Assim como no gráfico do domínio Negócios, é possível ver que, conforme aumenta o nível de similaridade, melhores ficam os resultados. Para o nível de similaridade de 50%, por exemplo, o *UCG* foi igual a 97% e o *AE* foi 0%, demostrando com isso ótimos resultados.

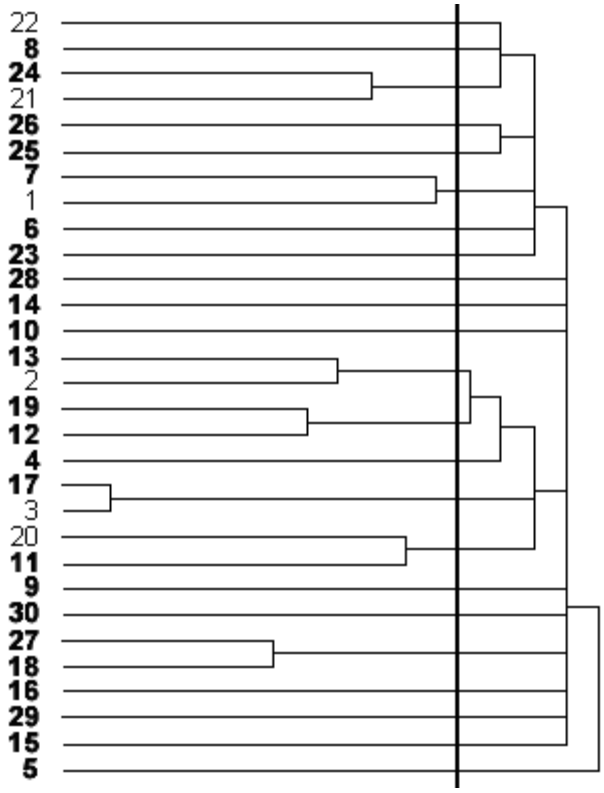

Figura 73 – Dendograma do domínio esportes com linha vertical indicando a melhor divisão dos grupos para 15% de similaridade

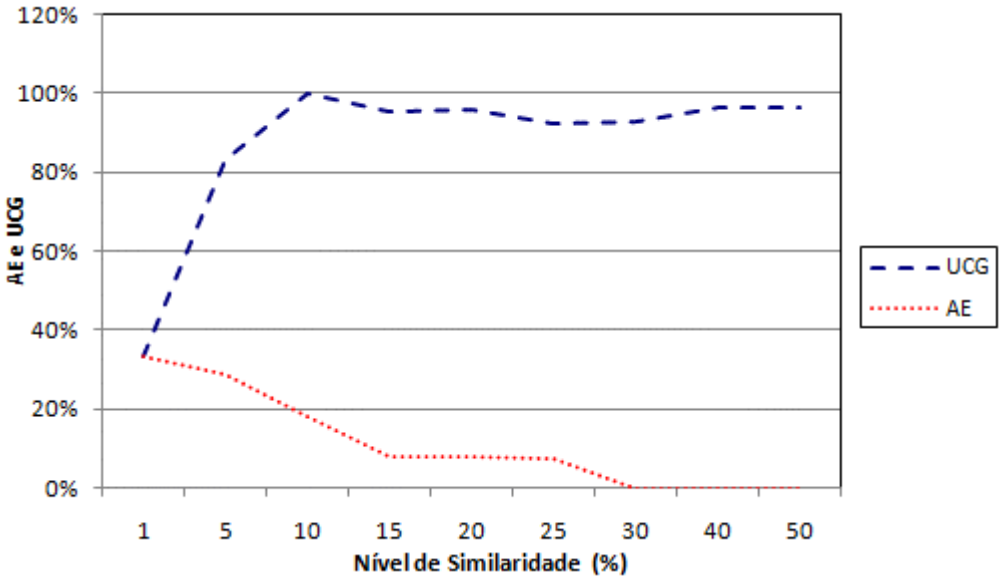

Figura 74 – Resultado de *AE* e *UCG* para o domínio esportes

Por fim, a Figura 75 apresenta o dendograma do domínio Notícias Gerais dos EUA. Utilizando o mesmo procedimento adotado as análises de dendogramas anteriores, é possível verificar que, para um nível de similaridade de 10%, a linha vertical escura indica o corte ótimo dos grupos, gerando 21 grupos. Os números em negrito indicam as notícias retornadas pela ferramenta para este nível de similaridade, sendo seu total igual a 23 notícias. Para este caso, o *UCG* foi igual a 95% e o *AE* foi 13%.

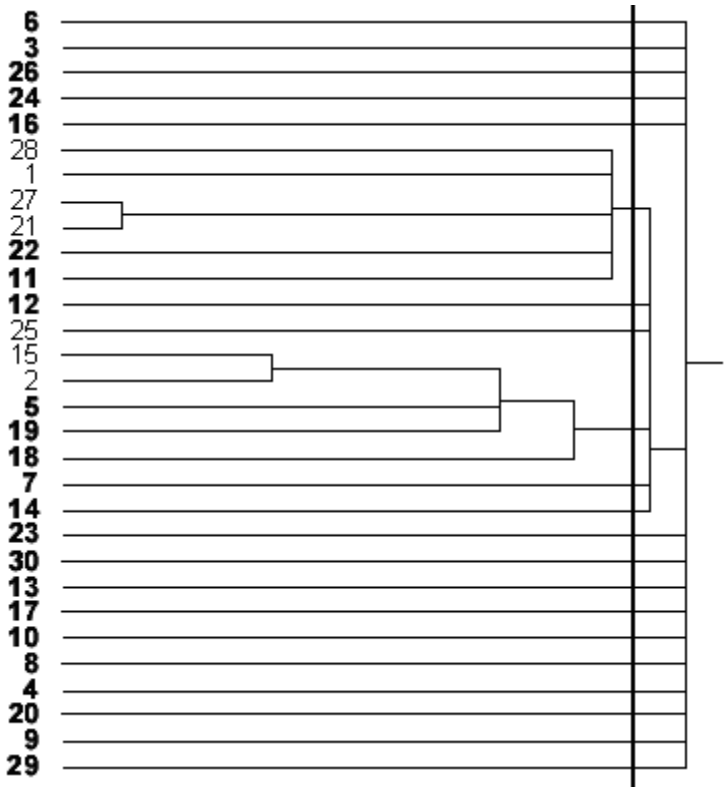

Figura 75 – Dendograma do domínio notícias gerais com linha vertical indicando a melhor divisão dos grupos para 10% de similaridade

A Figura 76 mostra o gráfico dos resultados de *AE* e *UCG* para os níveis de similaridades de 1% ate 50% do domínio Notícias. Assim como nos demais gráficos é possível observar que, de modo geral, conforme aumenta o nível de similaridade, a tendência é que *AE* fique próximo de 0% e que *UCG* fique próximo de 100%. Para o nível de similaridade de 50%, por exemplo, o *UCG* foi igual a 100% e o *AE* foi 0%, demostrando com isso excelentes resultados.

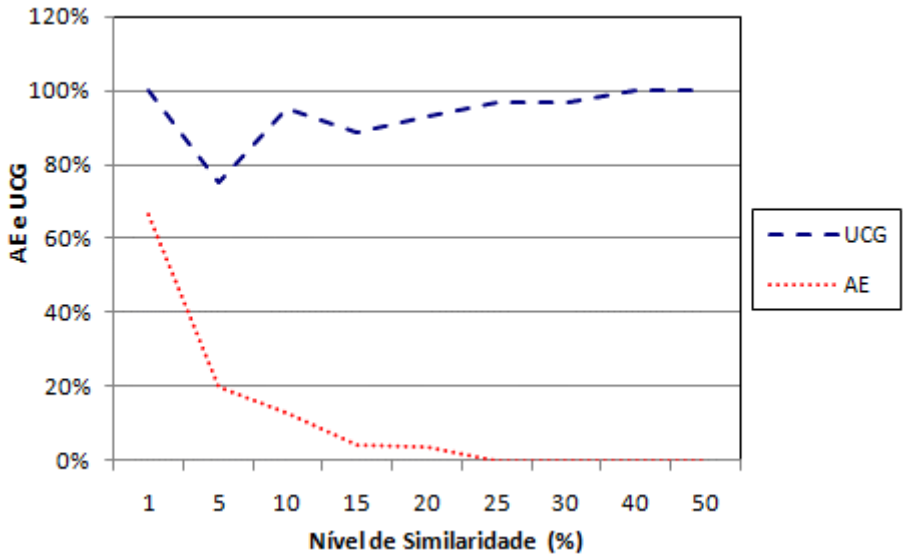

Figura 76 – Resultado de *AE* e *UCG* para o domínio notícias gerais dos EUA

Apesar de as curvas referentes aos três domínios apresentarem resultados em que o UCG tende para 100% e o AE tende para 0%, observa-se que as curvas apresentam diferentes formatos. Isto se deve ao fato de os domínios apresentarem diferentes números de notícias relacionadas em função do nível de similaridade. Considerando o mesmo número de notícias para os diferentes domínios, nos casos em que o número de grupos de notícias é maior para o mesmo nível de similaridade (cada grupo tende a ser formado por apenas uma notícia), os valores de UCG e AD tendem a ser mais próximos de 100% e 0%. Isto ocorreu no caso do domínio Notícias que apresentou muitos grupos formados por apenas uma notícia e cujos valores de UCG e AD convergiram mais rapidamente para 100% e 0%, respectivamente.

É importante ressaltar que foram utilizados dois métodos de agrupamento diferentes para comparar os resultados obtidos pela ferramenta. O agrupamento baseado no coeficiente de Jaccard, considerando o padrão Eliminador de Dados Similares utilizado pela ferramenta, com agrupamento baseado no método do vizinho mais próximo, considerando uma matriz de similaridade criada a partir da opinião dos participantes.

7.3.5.4 Análise Comparativa da Ferramenta *Feed Organizer* Versão Beta e da Ferramenta Yahoo *Pipe*

Esta análise considera o experimento realizado com a versão Beta da ferramenta. Ela objetiva analisar o uso combinado dos operadores de eliminação de dados. Para alcançar este fim, foi solicitado a cada participante do experimento que selecionasse dentro de um conjunto de 20 notícias, escolhidas randomicamente, de um total de 400 notícias, quais notícias seriam relevantes, considerando as seguintes regras:

1) São consideradas notícias relevantes as notícias que possuem as seguintes palavras-chave no título ou descrição: *acupuncture, health, méxico, Liverpool, Manchester, Nadal, G20, Phelps, Dylan, inflation, software, Microsoft*.

2) São consideradas notícias irrelevantes as notícias que possuírem as seguintes palavras-chave no título ou descrição: *exercise*, *acid*, *Japan*, *Barcelona*.

3) São consideradas notícias irrelevantes as notícias que estiverem relacionadas ao assunto política.

4) As regras 2) e 3) têm preferência sobre a regra 1).

Dentro do conjunto de 400 notícias distribuídas aos participantes, 49 não foram avaliadas pelos participantes, não sendo consideradas, portanto, na avaliação da ferramenta. Os anexos VII, VIII e IX apresentam os questionários realizados, os dados dos participantes dos experimentos, o conjunto de notícias utilizado e o resumo das respostas obtidas.

Para comparação com a ferramenta *feed Organizer* foram considerados os agregadores de *feeds* mais populares selecionados pelos participantes do experimento, quais sejam: Akregator[57], Google Reader[58], Internet Explorer[59], Firefox[60] e Yahoo *Pipe*[61]. Entretanto, com exceção do Yahoo Pipe, estes agregadores oferecem poucas opções para seleção de notícias, limitando-se a recebê-las dos sites produtores e exibí-las ordenadas pela data de publicação. Assim, atualmente, eles não permitem a criação de regras como as descritas anteriormente. Dessa forma, a comparação foi feita somente

[57] http://akregator.kde.org/
[58] http://www.google.com/intl/pt-BR/googlereader/tour.html
[59] http://www.microsoft.com/windows/IE/ie7/tour/rss/
[60] br.mozdev.org/
[61] http://pipes.yahoo.com/pipes/

com o agregador Yahoo *Pipe*[62], um dos agregadores mais completos atualmente. Como visto no Capítulo 2, esta ferramenta oferece mais opções para seleção de *feeds*, permitindo que se especifique o atributo do *feed* em que será buscada uma determinada palavra, além da combinação de regras. Em virtude desses fatores, essa ferramenta foi escolhida para fazer parte deste experimento.

A Figura 77 mostra a configuração necessária na interface da ferramenta Yahoo *Pipe* para atender as regras descritas anteriormente. É possível observar que um dos módulos mais importantes dessa ferramenta na filtragem de notícias, o módulo *filter*, oferece a comparação por palavras-chave para a seleção das notícias. Além disso, cada palavra a ser buscada exige que se crie uma nova regra, pois dentro da regra não existe a opção para buscar qualquer palavra. Essa opção está disponível somente para a combinação de regras, o que torna tedioso o processo de criação de regras se forem utilizadas muitas palavras. Adicionalmente, essa ferramenta não permite o uso de funções de similaridade ou lista negras. Dessa forma, para permitir a criação da regra referente ao domínio "política" (regra 3), foi utilizado o campo *category* dos *feeds*.

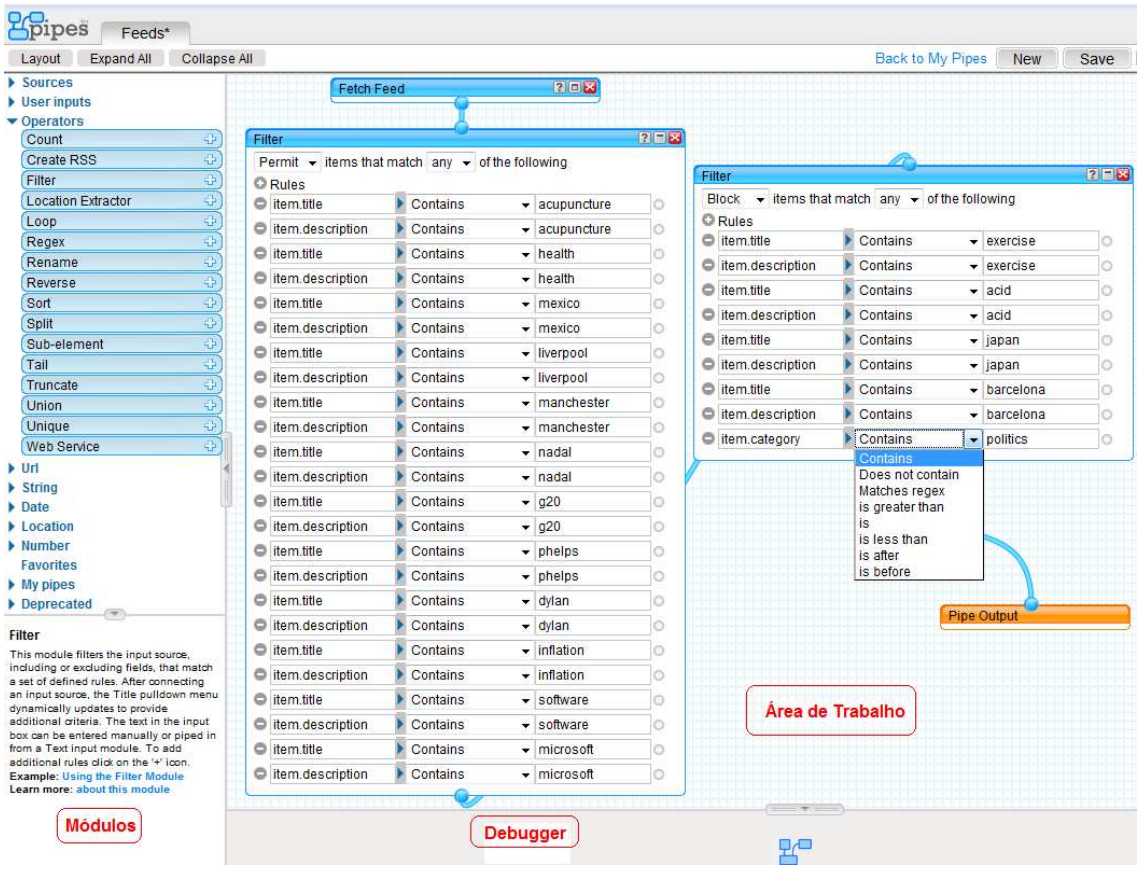

Figura 77 – Configuração da Ferramenta Yahoo *Pipe*

[62] http://pipes.yahoo.com/pipes/

Para realizar a comparação das ferramentas Yahoo *Pipe* e *Feed Organizer*, esta última utilizou **duas configurações diferentes**. A primeira configuração utilizou somente palavras-chave aplicadas ao operador Dados Permitidos, para a seleção das notícias de interesse, e ao operador Eliminador de Dados Proibidos, para o descarte das notícias proibidas e de notícias do domínio política através do uso de uma lista negra. A segunda configuração foi similar à primeira com exceção do uso da função de similaridade de Jaccard a 1%, aplicada sobre a lista negra.

A Figura 78 mostra a segunda configuração utilizada na interface da ferramenta *Feed Organizer* para atender as regras descritas anteriormente.

Figura 78 – Configuração da Ferramenta *Feed Organizer*

Com relação aos resultados obtidos, a ferramenta Yahoo *Pipe* retornou 80 respostas, o equivalente a 22,79% do total de notícias, das quais 49 estavam de acordo

com as respostas dadas pelos participantes do experimento, o equivalente a 61,25% de acertos. A cobertura máxima obtida foi de 29,70%.

A ferramenta *Feed Organizer*, na configuração que utilizou somente palavras-chave, retornou 45 respostas, o equivalente a 12,82% do total de notícias, das quais 33 estavam de acordo com as respostas dadas pelos participantes do experimento, o equivalente a 73,33% de acertos. A cobertura máxima obtida foi de 20%.

A ferramenta *Feed Organizer*, na configuração que utilizou palavras-chave combinadas as funções de Jaccard, retornou 99 respostas, o equivalente a 28,21% do total de notícias, das quais 70 estavam de acordo com as respostas dadas pelos participantes do experimento, o equivalente a 70,71% de acertos. A cobertura máxima obtida foi de 42,42%.

As repostas obtidas pela ferramenta *Feed Organizer*, em ambas as configurações usadas, foram ordenadas de duas formas: por valores decrescentes de data de publicação e por similaridade (valores decrescentes de similaridade para o operador AD e crescentes para o operador DP, conforme explicado no final da Seção 5.13.1).

A Figura 79 mostra o gráfico com os resultados obtidos para a primeira configuração da ferramenta. A linha referenciada como *Key_Data* e *Key_Sim* apresentam os dados obtidos com a ferramenta *Feed Organizer*, ordenados por data e similaridade, respectivamente. A linha *Yahoo* mostra os resultados obtidos com a ferramenta Yahoo *Pipe*. É importante considerar que os agregadores de notícias atualmente ainda são muito insipientes, sendo o Yahoo Pipe um dos agregadores mais completos, oferecendo diversos tipos de filtros que podem ser configurados facilmente pelos seus usuários.

Tomando por base os resultados obtidos, é possível dizer que a ferramenta *Feed Organizer* superou a ferramenta Yahoo *Pipe*. Na primeira configuração a precisão da *Feed Organizer* foi sempre superior a da Yahoo *Pipe* para os valores de cobertura até 20%. Entretanto, a cobertura foi menor (20% contra 29,70%). Em outras palavras, o número de notícias retornadas foi menor, com maior precisão, porém com uma cobertura menor das respostas relevantes. Basicamente, isto ocorreu porque a ferramenta *Feed Organizer* utiliza as técnicas de *stemming*, extração de *stopwords* e utilização de listas negras, não disponíveis na ferramenta Yahoo Pipe. As técnicas de *stemming* e *stopwords* podem conferir um maior número de batimentos (acertos) das

palavras selecionadas quando comparadas com as palavras extraídas das notícias. As listas negras permitem uma filtragem mais acurada dos tópicos proibidos, independente da categoria atribuída para a notícia pelos produtores de notícias. Assim, mesmo que uma notícia não seja incluída numa categoria que a representa, ou seja, incluída erroneamente, este tipo de abordagem consegue identificá-la corretamente. Isto, ao mesmo tempo, pode aumentar a precisão em função da dimuição no número de notícias irrelevantes, como pode reduzir o número de notícias relevantes retornadas em função dos falsos positivos. Isto ocorreu neste caso, gerando uma cobertura menor do que a obtida pela ferramenta Yahoo Pipe. Dessa forma, deve ser considerado o compromisso precisão versus cobertura na configuração das listas negras. A segunda configuração utilizada neste experimento diminui o peso das listas negras e tenta aumentar o valor de cobertura obtido, pois substitui busca por palavras-chave por uma relação de similaridade (Jaccard a 1%).

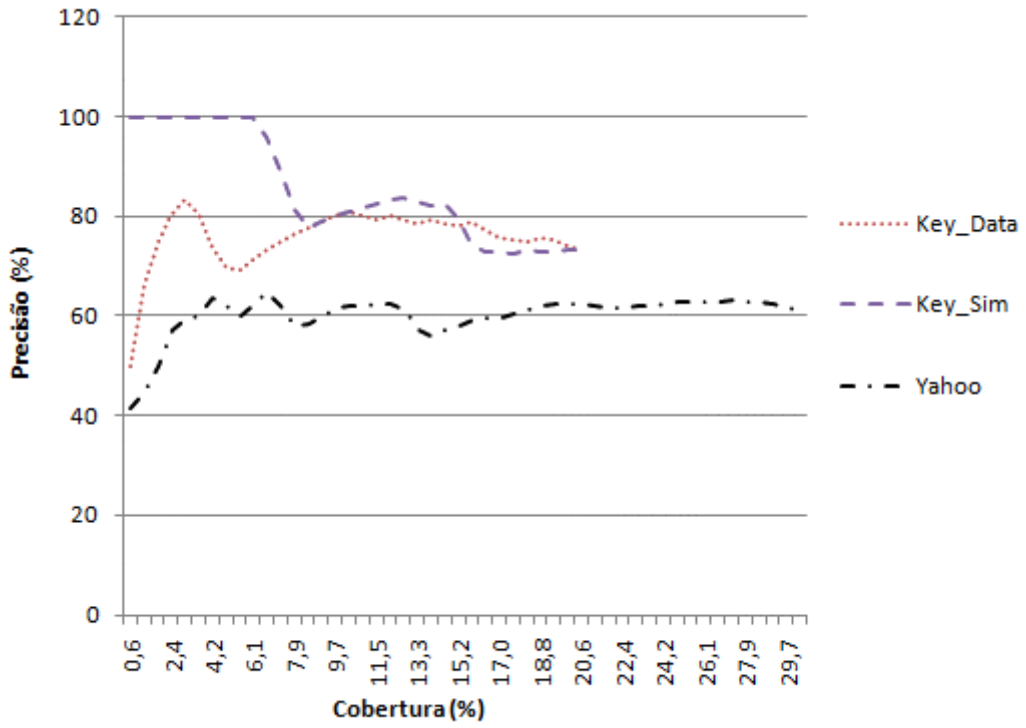

Figura 79 – Precisão versus Cobertura das Ferramentas Analisadas

Visualizando um aspecto diferente, as ordenações usadas, observa-se que as respostas ordenadas por similaridade forneceram maior precisão na maior parte de curva, principalmente para os valores mais baixos de cobertura. Isto significa que o usuário da ferramenta encontrará mais respostas relevantes sendo mostradas antes. Isto é

coerente, uma vez que as palavras-chave e assuntos selecionados pelos participantes refletem o que ele gostaria de ler em primeiro lugar (palavras permitidas) ou de não encontrar (palavras proibidas e listas negras). Dessa forma, as notícias mais similares às palavras permitidas devem ser mostradas nos primeiros lugares, enquanto as mais dissimilares às palavras e assuntos proibidos não devem ser mostradas ou, pelo menos, mostradas nas últimas posições de ordenação. Atualmente, apesar de comum nos buscadores *Web*, a ordenação por similaridade não é encontrada nos agregadores de feeds de notícias.

A Figura 80 mostra o gráfico com os resultados obtidos para a segunda configuração da ferramenta. A linha referenciada como *Jac_Data* e *Jac_Sim* apresentam dados obtidos com a ferramenta *Feed Organizer*, ordenados por data e similaridade, respectivamente. A linha *Yahoo* mostra os resultados obtidos com a ferramenta Yahoo *Pipe*.

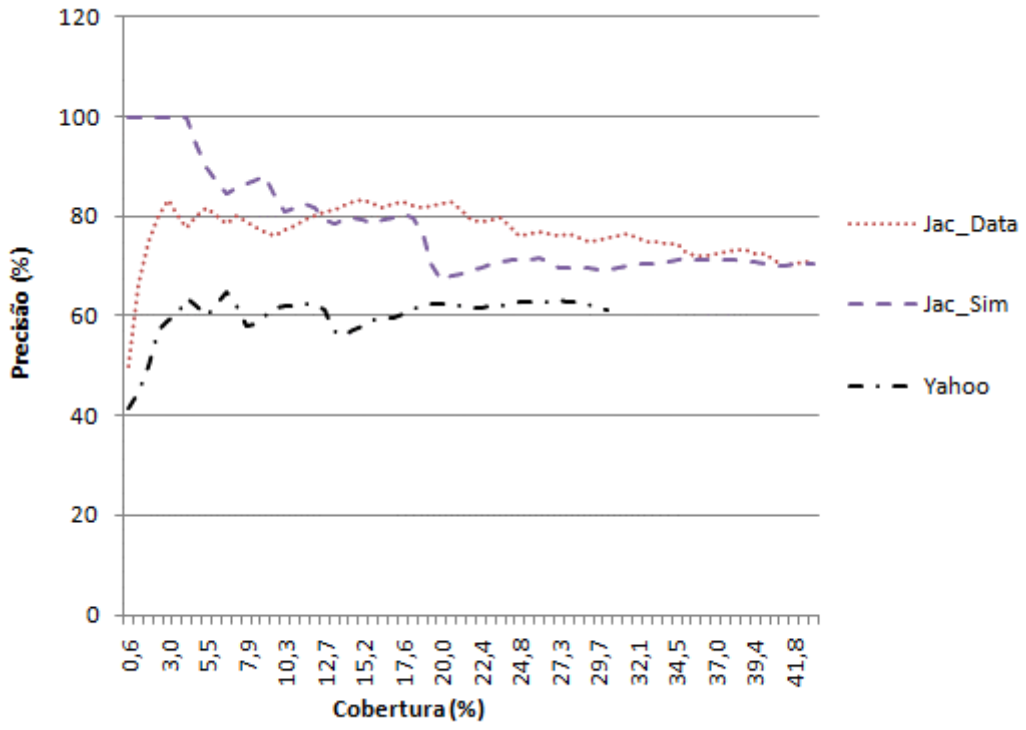

Figura 80 – Precisão versus Cobertura das Ferramentas Analisadas

Na segunda configuração, a ferramenta *Feed Organizer*, mesmo tendo um número maior de respostas retornadas, conseguiu obter resultados superiores ao da ferramenta Yahoo *Pipe* em todos os pontos da curva, maior precisão e maior cobertura (42,42% contra 29,70%). Os motivos que justificam este resultado são os mesmos

apresentados na primeira configuração da ferramenta. Entretanto, dessa vez, a cobertura foi maior, pois as listas negras permitiram que palavras proibidas existentes em seu conteúdo pudessem existir nas notícias, desde que em razão inferior a 1% do seu conteúdo total. Como o domínio escolhido para as listas negras foi Política, isto não representa um grande problema para a maioria das pessoas, diferentemente, por exemplo, do assunto Pornografia.

Com relação às ordenações usadas, observa-se que as respostas ordenadas por similaridade continuam fornecendo maior precisão para os valores mais baixos de cobertura. Entretanto, nota-se um maior equilíbrio para os outros valores, sendo que a ordenação por data é ligeiramente superior nos valores de cobertura maiores que 12%. Vale lembrar que as notícias correspondentes aos valores mais baixos de cobertura normalmente são lidas em primeiro lugar e, por isso, é desejável que apresentem um valor maior de precisão, como ocorre no caso da ordenação por similaridade.

7.3.5.5 Análise do Cálculo Inicial e Ajuste Autonômico do Valor de Similaridade para o Operador Eliminador de Dados Similares

Uma das funcionalidades autonômicas oferecidas pela ferramenta Feed Organizer versão Beta envolve o cálculo inicial e ajuste dos valores de similaridade em função da porcentagem ou do número desejado de notícias. Assim, o usuário pode configurar indiretamente os valores de similaridade dos operadores, o que facilitará o uso da ferramenta por parte dos usuários iniciantes.

A relação entre o número de notícias e os níveis de similaridade foi obtida com base no experimento com o operador de Eliminação de Dados Similares realizado com a versão Alfa da ferramenta. A Figura 81 mostra os resultados obtidos com este experimento e a curva derivada a partir dele. Percebe-se pelas curvas do experimento realizado que, em alguns domínios com os dados utilizados, não foi possível o descarte de mais de 90% de notícias, baseando-se simplesmente na similaridade. Adicionalmente, descartes menores que 10% do total de notícias, em função do pequeno número de dados descartados, não foram completamente mapeados no experimento. Dessa forma, o trecho utilizável para descarte de notícias fica limitado de 10% a 90%. É importante ressaltar que tal análise de valores baseia-se puramente nos valores obtidos no experimento e que este utilizou dados aleatórios de diferentes veículos de comunicação. Teoricamente, podem existir grupos de notícias que não apresentem

similaridade entre si ou notícias que apresentem um nível muito alto de similaridade. Nestes casos, as curvas obtidas experimentalmente não serão válidas.

O critério escolhido para a representação da curva foi utilizar por base uma função simples, de fácil representação e que permanecesse o maior trecho utilizável possível entre os limites dados pelas curvas do experimento mencionado anteriormente. Assim, foi escolhida uma função de segundo grau que perpassou três pontos (ajustáveis através do arquivo de configuração da ferramenta) escolhidos de modo a atender as condições dadas anteriormente (pontos: (4; 88), (8; 65) e (15; 30)).

É importante ressaltar que essa função simplesmente fornece o primeiro valor que relaciona o número de notícias e a similaridade. Este valor é ajustado a cada nova busca realizada (em segundo plano) pela ferramenta, de modo a atender as configurações feitas pelo usuário.

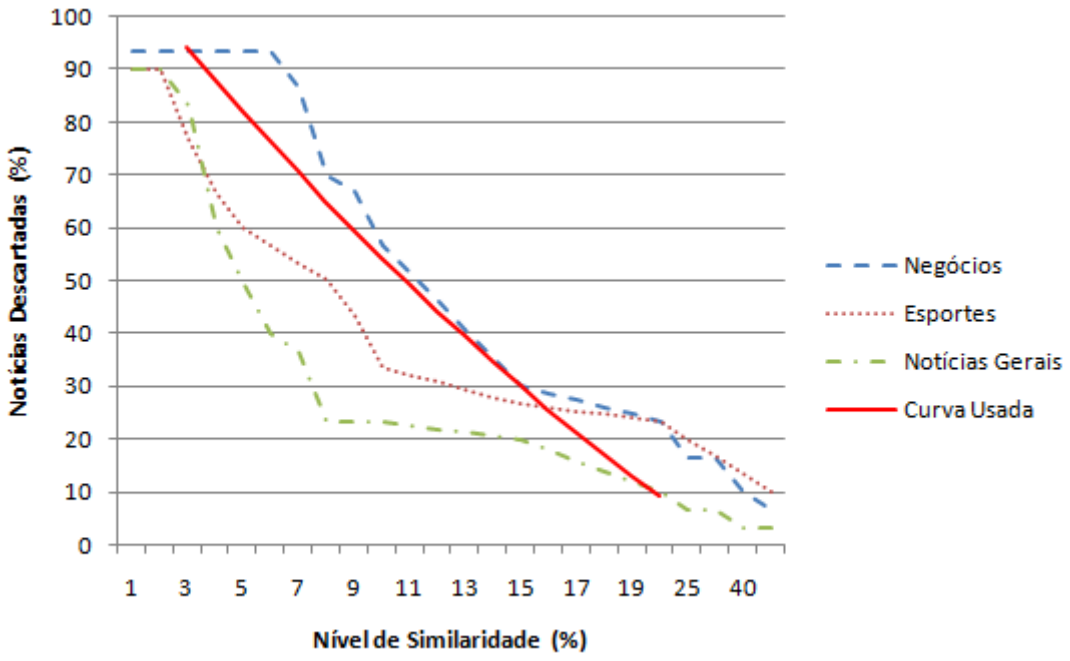

Figura 81 – Relacionamento entre Notícias e Similaridade

Este experimento utilizou 100 notícias, apresentadas no Anexo IX, que têm como identificador (ID) os valores de 1 a 1000 escolhidos randomicamente.

O experimento verificou, em intervalos de 10 notícias, variando de 10 a 90 notícias, quantas iterações eram necessárias para se obter o maior e o menor valor mais próximos do valor desejado, utilizando-se apenas o operador Eliminador de Dados Similares. Destaca-se que a obtenção do valor exato usando apenas este operador nem

sempre é possível em função das características das notícias (que podem, por exemplo, não apresentar nenhuma similaridade) e do nível de granularidade usado para a detecção de similaridade (no caso deste experimento, a similaridade pode variar a intervalos de 1%). Os resultados obtidos são apresentados na Tabela 46.

Tabela 46: Número de Iterações e Número de Notícias Obtido

Número de Notícias Desejado	Número de Notícias Obtido			
	1ª Iteração	2a Iteração	3a Iteração	4a Iteração
10	20	15	12	6
20	23	20	-	-
30	28	37	-	-
40	43	37	-	-
50	65	56	47	-
60	73	65	56	-
70	79	75	73	64
80	85	83	79	-
90	85	89	92	-

Os dados apresentados indicam que no máximo em 4 iterações a ferramenta encontra os maiores e menores valores mais próximos do valor desejado. O melhor resultado foi para a execução que envolveu 20 notícias, onde o valor exato foi encontrado. Os piores resultados, com quatro iterações, envolveram as execuções da ferramenta com 10 e 70 notícias. No caso de 10 notícias, este resultado pode ser explicado pela proximidade do limite de descarte de 90% de notícias, o que confere uma maior imprecisão no acerto do valor inicial de similaridade. No caso de 70 notícias, pode ser verificado na Figura 81 que este é o ponto de maior distância entre a curva usada e a curva de Notícias Gerais, o que também confere uma maior imprecisão para o valor inicial obtido.

7.3.5.6 Análise do Bloqueio Autonômico de URLs

Este experimento objetiva avaliar a viabilidade de detecção do evento composto cumulativo utilizando o arcabouço. Neste sentido, foi usada a funcionalidade de bloqueio autonômico de URLs, utilizando o operador DI, que visa bloquear endereços de feeds de notícias que geram grande número de notícias descartadas em determinado tempo. As notícias descartadas dependem das regras de filtragem criadas pelo usuário em um determinado perfil. Neste experimento foram utilizadas as regras definidas na seção 7.3.5.4, utilizando somente palavras-chave para todos os operadores e o conjunto de notícias apresentado no Anexo IX, identificadas pelos IDs alfanuméricos.

Na configuração de detecção utilizada neste experimento, um evento primitivo relativo a uma notícia descartada é gerado toda vez que uma notícia de um endereço é descartada e nenhuma notícia deste mesmo endereço é aceita numa mesma consulta. Um evento composto cumulativo é detectado caso o número de eventos primitivos relativos a um mesmo endereço seja igual ou superior a 20 num intervalo de 24 horas. Os endereços dessas notícias contemplaram três diferentes veículos de comunicação com diferentes endereços, apresentados na Tabela 47.

Tabela 47: Veículos de Comunicação e Endereços dos *Feeds* de Notícias

Veículo de Comunicação	Endereços dos *Feeds* de Notícias
New York Times	http://www.nytimes.com/pages/arts/music/index.html?partner=rss http://www.nytimes.com/pages/world/index.html?partner=rss http://www.nytimes.com/pages/health/index.html?partner=rss http://www.nytimes.com/pages/technology/index.html?partner=rss
The Guardian	http://www.guardian.co.uk/business http://www.guardian.co.uk/film http://www.guardian.co.uk/culture http://www.guardian.co.uk/politics http://www.guardian.co.uk/music http://www.guardian.co.uk/environment http://www.guardian.co.uk/money http://www.guardian.co.uk/lifeandstyle/health-and-wellbeing http://www.guardian.co.uk/lifeandstyle/besttreatments http://www.guardian.co.uk/sport
Washington Post	http://www.washingtonpost.com?nav=rss_nation/special/8 http://www.washingtonpost.com/wp-dyn/content/business/localbusiness/index.html?wprss=rss_business/localbusiness http://www.washingtonpost.com/wp-dyn/content/technology/index.html?wprss=rss_technology

Após a execução da consulta com a ferramenta Feed Organizer e as regras descritas anteriormente, 48 notícias foram selecionadas relativas aos endereços apresentados na Tabela 48.

Tabela 48: Veículos de Comunicação e Endereços Aceitos

Veículo de Comunicação	Endereços dos *Feeds* de Notícias
New York Times	http://www.nytimes.com/pages/health/index.html?partner=rss http://www.nytimes.com/pages/technology/index.html?partner=rss
The Guardian	http://www.guardian.co.uk/music http://www.guardian.co.uk/environment http://www.guardian.co.uk/money http://www.guardian.co.uk/lifeandstyle/health-and-wellbeing http://www.guardian.co.uk/lifeandstyle/besttreatments http://www.guardian.co.uk/sport

A Tabela 49 apresenta os endereços bloqueados, sem nenhuma notícia aceita, e o número de notícias bloqueadas, após a execução da ferramenta Feed Organizer.

Tabela 49: Veículos de Comunicação e Endereços Bloqueados

Veículo de Comunicação	Endereços dos *Feeds* de Notícias Bloqueados	Nº Notícias
New York Times	http://www.nytimes.com/pages/arts/music/index.html?partner=rss	1
	http://www.nytimes.com/pages/world/index.html?partner=rss	4
The Guardian	http://www.guardian.co.uk/business	48
	http://www.guardian.co.uk/film	9
	http://www.guardian.co.uk/culture	8
	http://www.guardian.co.uk/politics	20
Washington Post	http://www.washingtonpost.com?nav=rss_nation/special/8	6
	http://www.washingtonpost.com/wp-dyn/content/business/localbusiness/index.html?wprss=rss_business/localbusiness	3
	http://www.washingtonpost.com/wp-dyn/content/technology/index.html?wprss=rss_technology	5

Assim, a funcionalidade de bloqueio autonômico de URLs bloqueou, para as futuras consultas a serem realizadas sobre o perfil com as regras configuradas anteriormente, os endereços “http://www.guardian.co.uk/business” e “http://www.guardian.co.uk/politics”, pois possuem um número de notícias bloqueadas igual ou superior a 20.

7.3.6 Empacotamento

A última fase do experimento, o empacotamento, realiza a organização e armazenamento dos documentos construídos nas etapas anteriores, com o intuito de facilitar a repetição do estudo experimental no futuro. Todos os resultados foram arquivados e organizados em meio digital, sendo apresentados nos anexos IV, V, VI, VII, VIII e IX. O presente capítulo apresenta os gráficos e o sumário dos resultados relacionados nos questionários. Desse modo, estes documentos em conjunto contêm a identificação dos participantes do experimento e as perguntas e respostas existentes nos questionários e as e tabelas utilizados no desenvolvimento dos experimentos.

CAPÍTULO 8 CONCLUSÃO

Este trabalho realizou o estudo de áreas associadas às redes de Petri, regras ativas e computação autonômica, de forma a fornecer uma solução para os problemas relativos aos processos de eliminação de dados. Um arcabouço computacional é proposto, de forma a apoiar uma nova forma de modelagem de processos. A combinação dos paradigmas das diferentes áreas abordadas ocorre na medida em que são modelados processos de eliminação de dados, utilizando regras ativas e redes de Petri de alto nível, que monitoram o ambiente de forma autonômica através da detecção de eventos.

De forma a organizar os elementos do arcabouço computacional, é proposta uma arquitetura que comporta os diversos elementos necessários à implementação da proposta, juntamente com uma álgebra de eliminação de dados baseada nos padrões. Adicionalmente, no sentido de detectar eventos compostos que possam disparar os padrões, são propostas duas novas formas de modelar eventos utilizando RPAN: a modelagem estrutural e a modelagem baseada em marcas-modelo. Todo este conjunto de ferramentas e técnicas é combinado numa arquitetura que apoia a proposta do arcabouço autonômico de eliminação de dados.

Finalmente, como uma forma de avaliar o arcabouço, foi construída a ferramenta *Feed Organizer* que elimina notícias de acordo com os padrões Dados Permitidos, Eliminador de Dados Proibidos, Eliminador de Dados Similares, Eliminador de Dados Obsoletos e Eliminador de Dados Irrelevantes. Experimentos foram feitos de forma a comparar os resultados retornados por essa ferramenta com a opinião dos participantes dos experimentos e a ferramenta Yahoo *Pipe*, sendo que os resultados obtidos foram satisfatórios.

8.1 Contribuições

Ao final deste trabalho, as seguintes contribuições podem ser relacionados, estando de acordo com os objetivos propostos:

- Foi desenvolvido um método para descrever e formalizar padrões computacionais autonômicos utilizando RPANs e regras ativas;

- Foi proposta uma arquitetura para apoiar o arcabouço computacional autonômico desenvolvido;

- Foram elicitados os principais padrões de eliminação de dados voltados para informações consideradas proibidas, obsoletas, irrelevantes e duplicadas (ou muito similares). Estes padrões podem ser acionados por eventos gerados por sensores externos que monitoram os ambientes;

- Foi proposta uma álgebra de eliminação de dados, baseada nos padrões sugeridos;

- Foram descritos e modelados os padrões básicos de detecção de eventos compostos utilizando RPANs; e

- O arcabouço foi avaliado por meio de experimentos que analisaram alguns dos padrões formalizados, utilizando a ferramenta *Feed Organizer*.

Uma contribuição adicional, apresentado no Capítulo 4, foi o modelo que relaciona vários dos elementos utilizados neste trabalho com conceitos utilizados do laboratório TRASGO. Isto permitirá que o desenvolvimento de futuros trabalhos do laboratório possa ser mais facilmente alinhado com os conceitos apresentados nesta tese.

8.2 Trabalhos Futuros

O arcabouço autonômico não está limitado a uma aplicação ou uso específico. Os seus componentes podem ser utilizados em diferentes aplicações, como também podem ser usados na formalização de outros processos, permitindo que os mesmos atuem de forma autonômica.

Assim, um possível trabalho futuro consiste em formalizar outros processos que enriqueçam e arcabouço, permitindo o seu uso em outras áreas, além da eliminação de dados.

Outras melhorias consistem em desenvolver ferramentas que possam ser utilizadas nas diferentes camadas da arquitetura sugerida, principalmente a Camada de Aplicação, a Camada de Interface e a Camada de Comunicação. Destaca-se neste sentido, a necessidade de ferramentas e interfaces alternativas que permitam a diferentes aplicativos utilizar os processos modelados.

O tempo de processamento dos dados não foi considerado como um fator importante dos padrões e ferramentas apresentadas neste trabalho. Isto, de certa forma, ocorreu, pois o processamento das ferramentas apresentadas foi geralmente executado em segundo plano, de forma autonômica, não alterando as atividades realizadas pelos usuários. Entretanto, isto pode ser um requisito essencial em alguns sistemas e aplicações. Desse modo, uma possível melhoria é transportar o conhecimento armazenado nas RPAN na forma de padrões para outras linguagens e plataformas. Assim, os padrões podem ser convertidos para a linguagem C++ e usados no SECONDO, sem a necessidade de uso da ferramenta CPN *Tools*, o que melhorará o tempo de processamento dos dados. Outra possível solução envolve os trabalhos relacionados à melhoria do tempo de processamento das redes de Petri. Isto poderia, por exemplo, envolver a utilização de índices para os dados a serem processados e melhoria das estruturas de dados utilizadas pelas redes.

REFERÊNCIAS

ABITEBOUL, S., BENJELLOUN, O., MANOLESCO, I., et al., 2002, Active XML: Peer-to-Peer Data and Web Services Integration, In: *Proceedings of the 28th International Conference on Very Large Data Bases*, Hong Kong, China, pp 1087-1090.

ABITEBOUL, S., BONIFATI, A., COBENA, B., et al., 2003, Dynamic XML documents with distribution and replication, In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, San Diego, California,USA, pp 527-538.

ABITEBOUL, S., AMANN, B., CLUET, S., et al., 1999, Active Views for Electronic Commerce, In: *Proceedings of the 25th International Conference on Very Large Data Bases*, San Francisco, CA, USA, pp 138-149.

ACHENBACH, J., 1999, The Too-Much-Information Age. Today's Data Glut Jams Libraries and Lives. But Is Anyone Getting Any Wiser?, In: *Washington Post*, USA.

ANDERSON, G., 2003, *SAP Planning: Best Practices in Implementation*, Sams, Indianapolis, Indiana, USA, ISBN: 0789728753.

ANDREOLINI, M., BULGARELLI, A., COLAJANNI, M., et al., 2005, HoneySpam: honeypots fighting spam at the source, In: *SRUTI'05: Proceedings of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet Workshop*, Cambridge, MA, USA, pp 11-11.

ARANTES, A.S., VIEIRA, M.R., TRAINA, C., et al., 2003, Operadores de Seleção por Similaridade para Sistemas de Gerenciamento de Bases de Dados Relacionais., In: *Anais do XVIII Simpósio Brasileiro de Banco de Dados*, Manaus, Brasil, pp 341-355.

BABA-HAMED, L., 2006, Rules Termination Analysis Based on Petri Nets: Implementation Issues, In: *Information and Communication Technologies*, Computer Science Dept., University of Oran ES-SENIA, Oran, ALGERIA.

BAEZA-YATES, R., RIBEIRO-NETO, B., 1999, *Modern Information Retrieval*, 1st ed., Addison Wesley, ACM Press, New York, USA, ISBN: 020139829X.

BAILEY, M., OBERHEIDE, J., ANDERSEN, J., et al., 2007, "Automated Classification and Analysis of Internet Malware", In :*Proceedings of the 10th International Symposium on Recent Advances in Intrusion Detection* (RAID '07), pp 178-197.

BANCROFT, N.H., SEIP, H., SPRENGEL, A., 1997, *Implementing Sap R/3 : How to Introduce a Large System into a Large Organization*, 2 ed., Prentice Hall, Upper Saddle River, NJ, USA, ISBN: 013889213X.

BARATA, K., CAIN, P., ROUTLEDGE, D., 2001, *Principles and Practices in Managing Financial Records: A Reference Model and Assessment Tool*, Rights and Records Institute, May 1, London, UK, Disponível em: <http://202.54.104.236/intranet/eip/immunizationmanager/pdf/Principals_Financial%20_Records_WB_InfoDEV.pdf>.

BARROS, M.O., WERNER, C.M.L., TRAVASSOS, G.H., 2002, Um Estudo Experimental sobre a Utilização de Modelagem e Simulação no Apoio à Gerência de Projetos de Software, In: *Anais do XVI Simpósio Brasileiro de Engenharia de Software*, Gramado, RS, Brasil, pp. 191, 206.

BERKHIN, P., 2006, "A Survey of Clustering Data Mining Techniques", *Recent Advances inClustering*, Grouping Multidimensional Data, The Netherlands, pp 25-72.

BERLIND, D., 2005, "Is unsubscribing from spam enough?", *zdnet*, US Edition, 2005, Disponível em: <http://www.zdnet.com/blog/btl/is-unsubscribing-from-spam-enough/1810>.

BERMAN, F., 2006, "One hundred years of data", In: *Proceedings of the 2006 International Conference on Digital Government Research*, San Diego, California, USA, pp 3-4, Disponível em: <http://portal.acm.org/citation.cfm?doid=1146598.1146600>.

BERNARDINELLO, L., CINDIO, F.D., 1992, "A survey of basic net models and modular net classes", In: *Advances in Petri Nets 1992, The DEMON Project*, London, UK, pp 304-351.

BERNDTSSON, M., CALESTAM, B., 2003, Graphical notations for active rules in UML and UML-A, In: *Proceedings of the ACM SIGSOFT Software Engineering Notes*, New York, NY, USA, pp 2-2, Disponível em: <http://portal.acm.org/citation.cfm?id=638774&coll= GUIDE&dl=GUIDE&CFID=440658&CFTOKEN=59432777>.

BERTALANFFY, L.V., 2008, *Teoria Geral dos Sistemas: Fundamentos, Desenvolvimento e Aplicações.*, 3 ed, Petrópolis, Rio de Janeiro, p. 360, Brasil, ISBN: 8532636900.

BEYGELZIMER, A., KAKADE, S., LANGFORD, J., 2006, Cover trees for nearest neighbor, In: *Proceedings of the 23rd international conference on Machine learning*, Pittsburgh, Pennsylvania, pp 97-104.

BOYKIN, P., ROYCHOWDHURY, V., 2005, "Leveraging social networks to fight spam", IEEE Computer Society Press, Los Alamitos, CA, USA, v. 38, n. 4, pp. 68, 61.

CAREY, M.J., DEWITT, D.J., FRANKLIN, M.J., et al., 1994, "Shoring up persistent applications", *ACM SIGMOD Rec*, New York, NY, USA, v. 23, n. 2 (1994), pp. 383-394.

CARVALHO, M., GONÇALVES, M., LAENDER, A., et al., 2006, Learning to Deduplicate, In: *Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital libraries*, New York, NY, USA, pp. 41-50, Disponível em: <http://portal.acm.org/citation.cfm?doid=1141753.1141760>.

CASATI, F., CERI, S., PARABOSHI, S., et al., 1999, Specification and implementation of exceptions in workflow management systems, In: *ACM Transactions on Database Systems*, New York, NY, USA, Volume 24, Issue 3, pp. 405-450, Disponível em: <http://www.workflowpatterns.com/documentation/documents/p405-casati.pdf>.

CAVALCANTI, E., 1995, "Revoluçao da Informação: Algumas Reflexões", *Caderno de Pesquisa em Administração*, São Paulo, Brasil, vol 1, n 1, pp. 40-46.

CHAKRAVARTHY, S., KRISHNAPRASAD, V., ANWAR, E., et al., 1994, Composite Events for Active Databases: Semantics, Contexts and Detection, In: *Proceedings of the 20th International Conference on Very Large Data Bases*, San Francisco, CA, USA, pp 606-617.

CHAKRAVARTHY, S., MISHRA, D., 1994, "Snoop: An Expressive Event Specification Language for Active Databases", *Data Knowledge Engineering*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, v. 14, n. 1, pp. 1-26.

CHAUDHURI, S., DAYAL, U., 1997, "An Overview of Data Warehousing and OLAP Technology", ACM SIGMOD Record, New York, NY, USA, v. 26, n. 1, pp. 65-74.

CHEN, A., TSAI, P., KOH, J., 1996, Identifying Object Isomerism in Multidatabase Systems, In: *Distributed and Parallel Databases*, Kluwer Academic Publishers Hingham, MA, USA, v. 4, Issue 2, pp. 143-168.

CHIOLA, G., 1991, Simulation Framework for Timed and Stochastic Petri Nets, In: *International Journal in Computer Simulation*, v. 1, n. 2, pp. 153-168.

CHRISTEN, P., CHURCHES, T., 2005, "Febrl - Freely Extensible Biomedical Record Linkage", Australian National University, Dept. of Computer Science, Canberra, Australia.

CHRISTEN, P., CHURCHES, T., HEGLAND, M., 2004, "A Parallel Open Source Data Linkage System", *Advances in Knowledge Discovery and Data Mining*, Springer, Heidelberg, Berlin, pp. 638-647.

CODY, R.P., 1999, *Cody's Data Cleaning Techniques Using SAS Software*, 1st ed., SAS Publishing, NC, USA, ISBN: 1580256007.

COMAI, S., TANCA, L., 2003, Termination and Confluence by Rule Priorization, In: *IEEE Transactions on knowledge and Data Engineering*, Piscataway, NJ, USA, v. 15, Issue 2, pp. 257-270, Disponível em: <http://ieeexplore.ieee.org/xpl/freeabs_all.jsp? arnumber=1185831>.

COPLIEN, J., 1996, "Software Patterns", *Bell Laboratories, The Hillside Group*, 1996.

CORDEIRO, J., 2004, "Reflexoes sobre a Gestao da Qualidade Total: fim de mais um modismo ou incorporaçao do conceito por meio de novas ferramentas de gestão?", *Revista FAE, Curitiba, Paraná, Brasi, v. 7, n. 1*.

CUKIER, W., CODY, S., NESSELROTH, E., 2006, Genres of Spam: Expectations and Deceptions, In: *Proceedings of the 39th Annual Hawaii International Conference on System Sciences*, IEEE Computer Society, Washington, DC, USA, pp 51-51.

DAVIS, C., STEVENS, E., 2006, "The iECM Initiative: Enabling ECM Across System and Organizational Boundaries", *AIIM E-DOC Magazine*, Silver Spring, USA.

DE CARVALHO, A., BRAYNER, A., LOUREIRO, A., et al., 2006, *Seminário Grandes Desafios da Pesquisa em Computaçao no Brasil - 2006 - 2016*, SBC, São Paulo, Brasil, Disponível em: <http://www.sbc.org.br/ index.php? language=1&subject=8&content=downloads&id=272>.

DEY, D., SARKAR, S., DE, P., 1998, A Probabilistic Decision Model for Entity Matching in Heterogeneous Databases, In: Management Science, Institute for Operations Research and the Management Sciences (INFORMS), Linthicum, Maryland, USA, pp. 1379-1395.

DEY, D., SARKAR, S., DE, P., 2002, A Distance-Based Approach to Entity Reconciliation in Heterogeneous Databases, In: *IEEE Transactions on Knowledge and Data Engineering*, Piscataway, NJ, USA, pp. 567-582.

DIEKER, S., GUTING, R., 2000, Plug and play with query algebras: SECONDO, a generic DBMS development environment, In: Proceedings of the 2000 International Symposium on Database Engineering & Applications, Yokohama, Japan, pp. 380-390.

DRUCKER, P., 2000, "O futuro já chegou", Brasil, *Revista Exame*.

EHRIG, H., REISIG, W., ROZENBERG, G., et al., 2004, *Petri Net Technology for Communication-Based Systems: Advances in Petri Nets*, State of the Art Survey, LNCS 2472, Springer, Berlin, Germany.

FEHLING, R., 1993, A Concept of Hierarchical Petri Nets with Building Blocks, In: *Papers from the 12th International Conference on Applications and Theory of Petri Nets: Advances in Petri Nets*, London, UK, LNCS, v. 674, pp 148-168.

FELLEGI, I., SUNTER, A., 1969, "A Theory for Record Linkage", *Journal of the American Statistical Association*, pp. 1183–1210.

FINKEL, A., 1993, "The Minimal Coverability Graph for Petri Nets", In: *Papers from the 12th International Conference on Applications and Theory of Petri Nets:*

Advances in Petri Nets, LNCS 674, London, UK, pp 210-243.

FTC, 2003, " CAN-SPAM Act, PUBLIC LAW 108–187", The Library of Congress, USA.

FTC, 2005, "Definitions, Implementation, and Reporting Requirements Under the CAN–SPAM Act; Proposed Rule", *Federal Trade Commission*, USA, 2005.

GAMMA, E., HELM, R., JOHNSON, R., et al., 1994, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley Professional, ISBN: 0201633612.

GANTZ, J., REINSEL, D., CHUTE, C., et al., 2007, *IDC - The Expanding Digital Universe: A Forecast of Worldwide Information Growth Through 2010*, Framingham, MA, USA, Disponível em:<http://www.emc.com/collateral/analyst-reports/expanding-digital-idc-white-paper.pdf>.

GATZIU, S., DITTRICH, K., 1993, Events in an Active Object-Oriented Database System, In: *Proceedings of the 1st International Workshop on Rules in Database Systems*, Edinburgh, Scotland, Disponível em: <http://citeseer.ist.psu.edu/gatziu93events.html>.

GATZIU, S., DITTRICH, K., 1994, Detecting CompositeEvents in Active Database Systems Using Petri Nets, In: *Proceeding of the Fourth International Workshop on Research Issues in Data Engineering: Active Database Systems*, Houston, Texas, USA, pp. 2-9, Disponível em: <http://citeseer.ist.psu.edu/gatziu94detecting.html>.

GIRAULT, C., VALK, R., 2001, *Petri Nets for System Engineering: A Guide to Modeling, Verification, and Applications*, Springer-Verlag New York, Inc., ISBN: 3540412174.

GOVERNO FEDERAL, 2003, "Diretrizes da Implementação do Software Livre no Governo Federal". Disponível em: http://www.softwarelivre.gov.br/search?SearchableText=Priorizar+solu%C3%A7%C3%B5es%2C+programas+e+servi%C3%A7os+baseados+em+software+livre+.

GOVONI, D., 1999, *Java Application Frameworks*, 1st ed., John Wiley & Sons, New York, NY, USA, ISBN: 0471329304.

GÜTING, R.H., 1993, "Second-order signature: a tool for specifying data models, query processing, and optimization", *ACM SIGMOD Record, New York, NY, USA*, v. 22, n. 2, pp. 277-286.

GÜTING, R.H., BEHR, T., SPIEKERMANN, M., 2004a, "Secondo User Manual", *Praktische Informatik IV, Fernuniversität Hagen*, Germany.

GÜTING, R.H., BEHR, T., ALMEIDA, V., et al., 2004b, "Secondo: An extensible DBMS architecture and prototype". Praktische Informatik IV, Fernuniversität Hagen, Germany, Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/

summary?doi=10.1.1.10.4272>

GÜTING, R.H., ALMEIDA, V., ANSORGE, D., et al., 2005, SECONDO: an extensible DBMS platform for research prototyping and teaching, In: *Proceedings of the 21st International Conference on Data Engineering (ICDE 2005)*, IEEE Computer Society, Washington, DC, USA, pp 1115-1116.

HARPER, R., 2005, "Programming in Standard ML", *School of Computer Science, Carnegie Mellon University, Pittsburgh, USA*, Disponível em: <http://www.cs.cmu.edu/~rwh/smlbook/>

HERBST, H., KNOLMAYER, G., MYRACH, T., et al., 1994, "Methods and Associated Tools for the Information Systems Life Cycle", Cap. The Specification of Business Rules: A Comparison Of Selected Methodologies, Amsterdam, Elsevier.

HERNÁNDEZ, M., STOLFO, S., 1995, The Merge/Purge Problem for Large Databases, In: *ACM SIGMOD Record*, New York, NY, USA.

HOUAISS, A., 2001, *Dicionário Houaiss da Língua Portuguesa*, ISBN: 857302383X.

IBM, 2001, "IBM Global Services and autonomic computing", *IBM*, 2001. Disponível em:<http://www-03.ibm.com/autonomic/pdfs/wp-igs-autonomic.pdf>

IBM, 2005, "An architectural blueprint for autonomic computing", *IBM White Paper*, v. 7, Disponível em: <http://www-03.ibm.com/autonomic/pdfs/ ACBP2_2004-10-04.pdf>

IEEE, ACM, 2001, *Computing Curricula 2001 Computer Science*, IEEE e ACM, 2001, Disponível em: <http://www.sigcse.org/resources/cs-2001/ cc2001.pdf>.

INMON, W.H., 1996, *Building the Data Warehouse*, 2 ed., Wiley & Sons, Inc., New York, USA, ISBN: 0471141615.

INMON, W., 2001, "Building The Corporate Information Factory: The Order of the Components", *Powell Media LLC.*, 2001. Disponível em:<http://www.b-eye-network.com/files/inmon_050204g.pdf>

ISO 15489, 2001, *The International Standard on Records Management, ISO 15489*, The International Organization for Standardization (ISO) and Australian Standard for Records Management, Australia.

ISO 15909-1, 2004, *International Organization for Standardization - Software and system engineering -- High-level Petri nets -- Part 1: Concepts, definitions and graphical notation*, Disponível em:<http://www.iso.org/iso/iso_catalogue/ catalogue_tc/catalogue_detail.htm?csnumber=38225>.

ISO 15909-1 FD, 2002, *International Standard ISO/IEC 15909-1 Final Draft - High-*

level Petri Nets - Concepts, Definitions and Graphical Notation.

ISO 15909-2 WD, 2005, *International Standard ISO/IEC 15909-2 WD 0.9.0, Software and Systems Engineering -- High-level Petri Nets Part 2: Transfer Format*, Disponível em:<http://www.petrinets.info/docs/ISO-IEC15909-2.WD.V0.9.0.pdf>.

JAIN, A.K., MURTY, M.N., FLYNN, P.J., 1999, "Data Clustering: A Review", ACM Computing Surveys, New York, NY, US, pp. 264-323.

JENSEN, K., 1986, Coloured Petri Nets, In: *W.Brauer, W.Reisig and G.Rozenberg (eds.): Petri Nets: Central Models and Their Properties, Advances in Petri Nets, 1986 Part I, Lecture Notes in Computer Science*.

JENSEN, K., 1994, An Introduction to the Theoretical Aspects of Coloured Petri Nets, In: *A Decade of Concurrency, Reflections and Perspectives, REX School/Symposium*,LNCS 803, Berlin, Germany, pp 230-272.

JENSEN, K., 1998, An Introduction to the Practical Use of Coloured Petri Nets, In: *Lectures on Petri Nets II: Applications, Advances in Petri Nets, the volumes are based on the Advanced Course on Petri Nets*, LNCS 1492, Berlin, Germany, pp 237-292.

JTC1, "Petri Nets Standards". Joint Technical Committee 1 of the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). Disponível em: http://www.petrinets.info/standard.php.

KEPHARD, J., CHESS, D., 2003, The Vision of Autonomic Computing, In: *IEEE Computer Society*, Los Alamitos, CA, USA.

KIMBALL, R., 2002, "Two Powerful Ideas. The foundations for modern data warehousing", *IntelligentEnterprise.com*. Disponível em: <http://www.intelligententerprise.com/020917/515warehouse1_1.jhtml>.

KLEINBERG, J.M., 1999, "Authoritative sources in a hyperlinked environment", *Journal of the ACM*, New York, NY, USA, v. 46, n. 5 (1999), pp. 604-632.

KOUDAS, N., SARAWAGI, S., SRIVASTAVA, D., 2006, Record linkage: similarity measures and algorithms, In: *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, Chicago, IL, USA, pp 802-803.

KUHLINS, S., TREDWELL, R., 2003, "Objects, Components, Architectures, Services, and Applications for a Networked World", *Objects, Components, Architectures, Services, and Applications for a Networked World*, Erfurt, Germany, pp. 184-198.

KUMAR, D., HAROUS, S., 1990, An approach towards distributed simulation of timed petri nets, In: *Proceedings of the 22nd conference on Winter simulation*, New

Orleans, Louisiana, United States, pp 428-435.

KUMMER, O.W., MOLDT, D.R., 2004, An Extensible Editor and Simulation Engine for Petri Nets: Renew, In: *25th International Conference on Application and Theory of Petri Nets (ICATPN 2004), LNCS3099, Berlin, Germany*.

LAKOS, C., KEEN, C., 1991, Modelling layered protocols in LOOPN, In: *Proceedings of the Fourth International Workshop on Petri Nets and Performance Models*, Melbourne, Australia, pp. 106-115.

LAKOS, C., 1997, "Object Oriented Modelling with Object Petri Nets", *In: Advances in Petri Nets. LNCS*, pp. 1-37.

LI, X., CHAPA, S., MARIN, J., et al., 2004, An application of conditional colored Petri nets: active database system, In: *IEEE International Conference on Systems, Man and Cybernetics*, Piscataway NJ, USA, vol. 5, pp 4885-4890.

LI, X., MEDINA, J., CHAPA, S., 2007, "Applying Petri Nets in Active Database Systems", In: *Part C: Applications and Reviews, IEEE Transactions on Systems, Man, and Cybernetics*, USA, v. 37, n. 4, pp. 482-493.

LIEVEN, P., SCHEUERMANN, B., STINI, M., et al., 2007, Filtering Spam Email Based on Retry Patterns, In: *IEEE International Conference on Communications. ICC '07*. Glasgow, UK, pp. 1515-1520.

LORIATO, L.A., LORIATO, L.A., 2008, *Protótipo de Anti-vírus Estático com Verificação On-Access Utilizando a Engine do ClamAv*, Projeto Final de Curso, IME, rio de Janeiro, Brasil.

LÜDKE, M., ANDRÉ, M.E.A., 1990, *Pesquisa em Educação: Abordagem Qualitativa*, EPU, São Paulo, Brasil, ISBN: 8512303700.

LYMAN, P., VARIAN, H.R., 2003, "How much information 2003?". School of Information Management and Systems, University of California at Berkeley, California, USA. Disponível em: http://www.sims.berkeley.edu/how-much-info-2003.

LYMBEROPOULOS, L., LUPU, E., SLOMAN, M., 2003, "An Adaptive Policy-Based Framework for Network Services Management", *Journal of Network and Systems Management, New York, NY, USA*, v. 11, n. 3, pp. 277-303.

MADEIRA, F., 2007, *O Problema do SPAM e Técnicas de Combate*, Monografia, Associação do Ensino Superior de Olinda, Brasil, Disponível em: <http://www.madeira.eng.br/wiki/index.php?page= O+Problema+do+ SPAM+e+T%C3%A9cnicas+de+Combate>.

MANSOURI-SAMANI, M., SLOMAN, M., 1997, "GEM: A Generalised Event Monitoring Language for Distributed Systems", *IEE/IOP/BCS Distributed Systems Engineering Journal*, v. 4.

MCMENAMIN, S.M., PALMER, J.F., 1984, *Essential systems analysis*, Yourdon Press, Upper Saddle River, NJ, USA, ISBN: 0-917072-30-8.

MESSMER, E., 2007, New Approaches to Malware Detection Coming into View, McAfee, Symantec, Disponível em: <http://www.networkworld.com/news/ 2007/042507-malware-detection.html?page=1>.

METZ, J., MONARD, M.C., 2006, *Estudo e Análise das Diversas Representações e Estruturas de Dados Utilizadas nos Algoritmos de Clustering Hierárquico*, USP - ICMC, São Carlos, Brasil, 2006.

MICROSOFT, 2006, "Enterprise Content Management: Breaking the Barriers to Broad User Adoption", *Microsoft*. Disponível em:<http://office.microsoft.com/en-us/sharepointserver/HA102063591033.aspx>

MILLER, B., 2005, "The Autonomic computing edge: Can you CHOP up autonomic computing", *IBM*. Disponível em: <http://www-128.ibm.com/developerworks/ autonomic/library/ac-edge4>

MORESI, E., 2000, "Delineando o Valor do Sistema de Informaçao de uma Organizaçao", *Revista Ciência da Informação - Instituto Brasileiro de Informação em Ciência e Tecnologia (IBICT)*, Brasil.

MORESI, E., TARAPANOFF, K., 2001, Inteligência Organizacional e Competitiva. Cap. Gestao da Informação e do Conhecimento. Brasília, Brasil, UnB.

MORETO, D., ENDLER, M., 2001, "Evaluating composite events using shared trees", *IEE Proceedings — Software*, v. 148, n. 1, pp. 1-10.

MULYAR, N., VAN DER AALST, W., 2005, Patterns in Colored Petri Nets, In: *Sixth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, Aarhus, Denmark, Disponível em: <www.daimi.au.dk/CPnets/ workshop05/cpn/papers/ NataliyaMulyar.pdf>.

MURATA, T., 1989, "Petri nets: Properties, analysis and applications", *Proceedings of the IEEE*, v. 77, n. 4, pp. 541-580.

NAVATHE, S., TANAKA, A., MADHAVAN, R., et al., 1995, *A Methodology for Application Design Using Active Database Technology*, Georgia Inst of Tech Atlanta, Atlanta, USA, Disponível em: <http://handle.dtic.mil/100.2/ ADA294797>.

NELSON, J., 2004, "Turning Back the Wave of Junk E-Mail". BNET. Disponível em: http://findarticles.com/p/articles/mi_qa5292/is_200403/ai_n24277009/.

NETO, B.B., SCARMINO, I.S., BRUNS, R.E., 2007, *Como Fazer Experimentos*. 3 ed, Campinas, São Paulo, Brasil, Unicamp, p. 480, pp. 35-35, ISBN: 9788526807532.

NEVES, J.L., 1996, "Pesquisa qualitativa: características, usos e possibilidades", *Cadernos de Pesquisas em Administração*, São Paulo,

Brasil, v. 1, n. 3.

NWG, 2001, *RFC 2821 - Simple Mail Transfer Protocol*, Network Working Group, 2001.

OLIVEIRA, J., PINHEIRO, W.A., SOUZA, J.M., et al., 2006, Symptom Analysis of a Web Server Log, In: *The Latin American Autonomic Computing Symposium*, Campo Grande, Brasil.

OMG, 2003, *Common Warehouse Metamodel (CWM) Specification*, pp.576, Disponível em: <http://www.omg.org/spec/CWM/1.1/PDF/>.

PAGE, L., BRIN, S., MOTWANI, R., et al., 2008, "The PageRank Citation Ranking: Bringing Order to the Web". Digital Library Technologies Project. Disponível em: http://ilpubs.stanford.edu:8090/422/.

PATON, N.W., DÍAZ, O., 1999, "Active database systems", ACM Computing Survey, New York, NY, USA, v. 31, n. 1, pp. 63-103.

PERAZOLO, M., 2006, "Symptoms Deep Dive, Part 1 and Part 2: The Autonomic Computing Symptoms Format", IBM. Disponível em: <http://www-128.ibm.com/developerworks/autonomic/library/ac-symptom1/>

PETRI, C., 1962, "Kommunikation mit Automaten", *Bonn: Institut fur Instrumentelle Mathematik, Schriften des IIM Nr.3, also, English translation (1966), ``Communication with Automata''*, New York: Griffiss Air Force Base, Tech. Rep. RADC-TR-65-377, New York, USA, vol.1, Suppl. 1.

PIETZUCH, P., SHAND, B., BACON, J., 2004, "Composite event detection as a generic middleware extension", IEEE Network, v. 18, n. 1, pp. 44-55.

PINHEIRO, W.A., 2004, *Busca em Portais Semânticos: uma Abordagem Baseada em Ontologias*. Dissertação de Mestrado. Instituto Militar de Engenharia, Rio de Janeiro, RJ, Brasil, p. 170.

PINHEIRO, W.A., MOURA, A.M.C., 2004, An Ontology Based-Approach for Semantic Search in Portals. In: *15th International Workshop on Database and Expert Systems Applications*. IEEE Computer Society, Los Alamitos, CA, USA, pp.127-131.

PINHEIRO, W.A., OLIVEIRA, J., SOUZA, J.M., et al., 2006, Using Autonomic Computing and Click Stream Analysis for Problem Identification in a Continuous Production, In: *Conference on Cooperative Design, Visualization and Engineering*, Palma de Maiorca, Espanha, pp. 17-24.

PINHEIRO, W.A., BARROS, R., RODRIGUES NT., J.A., et al., 2008a, Using Active Rules and Petri Nets to Composite Event Detection in Autonomic Systems, In: *3th Latin American Autonomic Computing Symposium*, Gramado, Brasil, pp. 81-84.

PINHEIRO, W.A., XEXÉO, G.B., SOUZA, J.M., 2008b, Autonomic Patterns: Modelling Data Killing Patterns Using High-Level Petri Nets, In: *The Fourth International Conference on Autonomic and Autonomous Systems*, Gosier, Guadeloupe, pp 198-203.

PINHEIRO, W.A., XEXÉO, G.B., SOUZA, J.M., et al., 2008c, Framework para Modelagem de Processos usando Redes de Petri de Alto Nível: um Enfoque sobre a Eliminação de Dados, In: *IV Simpósio Brasileiro de Sistemas de Informação*, Rio de Janeiro, Brasil, pp 199-210.

PINHEIRO, W.A., RODRIGUES, T.S., SILVA, M.A.R., et al., 2009a, Autonomic RSS: Discarding Irrelevant News, In: *The Fifth International Conference on Autonomic and Autonomous Systems*, Valencia, Espanha, pp. 148-153.

PINHEIRO, W., SILVA, M.C.O., BARROS, R., et al., 2009b, Autonomic Collaborative RSS: an Implementation of Autonomic Data using Data Killing Patterns, In: *The 13th International Conference on Computer Supported Cooperative Work in Design*, Santiago, Chile, pp. 492-497.

PINHEIRO, W.A., SILVA, M.C.O., RODRIGUES, T.S., et al., 2010, "Discarding Similar Data with Autonomic Data Killing Framework Based on High-Level Petri Net Rules: An RSS Implementation". In: *The Sixth International Conference on Autonomic and Autonomous Systems*, Cancun, México. IEEE Computer Society, 2010. pp.110-115.

PORTER, M.F., 1997, "Readings in Information Retrieval", Cap. An algorithm for suffix stripping. Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, pp. 313-316.

POSTEL, J.B., 1982, *RFC 821 - Simple Mail Transfer Protocol*, Network Working Group.

RAHM, E., DO, H.H., 2000, "Data Cleaning: Problems and Current Approaches", *IEEE Data Engineering Bulletin*, v. 23, n. 4, pp. 3-13.

RAPID INTELLINGENCE, 2009, "State Master Encyclopedia", Cap. Data Domain, *State Master Encyclopedia*, Nation Master. Disponível em: < http://www.statemaster.com/encyclopedia/Data-domain>

RATZER, A., WELLS, L., LASSEN, H., et al., 2003, CPN Tools for Editing, Simulating, and Analysing Coloured Petri Nets, In: *Proceedings of the 24th International Conference on the Application and Theory of Petri Nets*, Eindhoven, The Netherlands, Disponível em: <http://wiki.daimi.au.dk/ cpntools/_files/ cpntools_pn03.pdf>.

ROSS, R., 1997, *The Business Rule Book: Classifying, Defining and Modeling Rules*, Business Rule Solutions Inc, ISBN: 0941049035.

ROTH, M.T., SCHWARZ, P.M., 1997, Don't Scrap It, Wrap It! A Wrapper Architecture for Legacy Data Sources, In: *Proceedings of the 23rd International*

Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp 266-275.

RUDIO, F.V., 2001, *Introdução ao Projeto de Pesquisa Científica*, 29th ed., Vozes, Petrópolis, Rio de Janeiro, Brasil, ISBN: 853260027-1.

RUSSELL, N., TER HOFSTEDE, A., EDMOND, D., et al., 2004, *Workflow Data Patterns*, FIT-TR-2004-01, Disponível em: <http://www.workflowpatterns.com/documentation/documents/data_patterns%20BETA%20TR.pdf>.

RUSSELL, N., TER HOFSTEDE, A., VAN DER AALST, W., et al., 2006, *Workflow Control-Flow Patterns: A Revised View*, BPMcenter.org, Disponível em: <http://www.workflowpatterns.com/ documentation/documents/BPM-06-22.pdf>.

SAHAMI, M., DUMAIS, S., HECKERMAN, D., et al., 1998, A Bayesian Approach to Filtering Junk E-mail, In: *AAAI Workshop on Learning for Text Categorization*, Madison, Wisconsin, USA.

SALVATORE, D., REAGLE, D. P., 2002, *Schaum's outline of theory and problems of statistics and econometrics*. 2 ed, McGraw-Hill Professional, USA, p. 338, pp. 72-72, ISBN: 9780071348522.

SARAWAGI, S., BHAMIDIPATY, A., 2002, "Interactive deduplication using active learning", In: *International Conference on Knowledge Discovery and Data Mining*, Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, Edmonton, Alberta, Canada, pp. 269-278.

SAS, 2007, "The SAS Enterprise Intelligence Platform: An Overview", *SAS White Paper*, Disponível em: <http://www.sas.com/offices/europe/switzerland/pdf/solutions/EIP_An_Overview.pdf>.

SATZINGER, J.W., 1992, Introduction To Essential Systems Analysis, In: *The Seventh Annual Conference of the International Academy for Information Management*, Dallas, Texas, USA, Disponível em: <http://dominict.myweb.uga.edu/Mist4620/Satzinger.pdf>, Acesso em: December 7, 2008.

SENGE, P., 2004, *A Quinta Disciplina*, 16 ed., Best Seller, ISBN: 8571236216.

SILVA, E.L., MENEZES, E.M., 2001, *Metodologia da Pesquisa e Elaboração de Dissertação*, 3rd ed., Florianópolis, Brasil, Universidade Federal de Santa Catarina, Florianópolis, Brasil.

SKOUDIS, E., ZELTSER, L., 2003, *Malware: Fighting Malicious Code*, Prentice Hall, Upper Saddle River, NJ, USA, ISBN: 0131014056.

SPYMAN, H., 2004, *Manual Completo do Hacker*, 5th ed., Book Express, ISBN: 8575840029.

STERRITT, R., 2005, Autonomic computing, In: *Journal of Innovations in Systems and Software Engineering*.

STOLLENWERK, M., 1999, Gestao do Conhecimento, Inteligência Competitiva e Estratégia Empresarial: em busca de uma abordagem integrada, In: *Anais do Workshop Brasileiro de Inteligência Competitiva*, Disponível em: <http://www.abraic.org.br/V2/periodicos_teses/ic_a27.pdf>.

SUN, 2007, "SAS Enterprise Intelligence Platform Powered by Sun's Datacenter of the Future", *SAS White Paper*.

TANAKA, A., NAVATHE, S., CHAKRAVARTHY, S., et al., 1991, ER-R: An Enhanced ER Model with Situation-Action Rules to Capture Application Semantics, In: *Proceedings of the 10th International Conference on the Entity RelationshipApproach*, San Mateo, California, USA, pp. 59-75.

TANEMBAUM, A.S., 2003a, *Redes de Computadores*, ed. 4, p. 955, ISBN: 8535211853, Campus.

TANEMBAUM, A.S., 2003b, *Sistemas Operacionais Modernos*, ed. 2, p. 707, ISBN: 8587918575, Prentice-Hall.

TARAPANOFF, K., 2001, *Inteligência Organizacional e Competitiva*, Brasília, Brasil, UnB, ISBN: 85-230-0637-0.

TEMPLETON, B., "Origin of the term "spam" to mean net abuse". Disponível em: http://www.templetons.com/brad/spamterm.html.

TERENCE, A.C.F., ESCRIVÃO FILHO, E., 2006, Abordagem quantitativa, qualitativa e a utilização da pesquisa-ação nos estudos organizacionais, In: *Anais do XXVI Encontro de Engenharia de Produção*, Fortaleza, CE, Brasil, pp. 1-9.

TERRY, S.H., 1997, "Fortran Tutorial". Universitetet i Oslo. Disponível em: http://folk.uio.no/steikr/doc/f77/tutorial/.

THOMPSON, R., 2005, "Why spyware poses multiple threats to security", *Communications of the ACM*, New York, NY, USA, v. 48, n. 8, pp. 41-43.

TOFFLER, A., 1984, *Future Shock*, Bantam, ISBN: 0553277375.

TRIVINOS, A.N.S., 1994, *Introdução à pesquisa em ciências sociais. A pesquisa qualitativa em educação*, Atlas, Brasil, ISBN: 852240273-6.

TROMPEDELLER, M., 1995, "A Classification of Petri Nets". Disponível em: http://www.informatik.uni-hamburg.de/TGI/PetriNets/classification/.

VAN DER AALST, W., 1998, "The Application of Petri Nets to Workflow Management", *The Journal of Circuits, Systems and*

Computers, v. 8, n. 1, pp. 21-66.

VAN DER AALST, W., TER HOFSTEDE, A., 2005, YAWL: Yet Another Workflow Language, In: *Information Systems*, v. 30, Issue 4, pp. 245-275.

VAN DER AALST, W., TER HOFSTEDE, A., KIEPUSZEWSKI, B., et al., 2003, Workflow Patterns, In: *Distributed and Parallel Databases*, Hingham, MA, USA, pp. 5-51, Disponível em: <http://www.workflowpatterns.com/ documentation/documents/ wfs-pat-2002.pdf>.

VAN DER STRAETEN, R., BRAEM, M., VANDERPERREN, W., 2007, *Aspect-Oriented Support for Workflow Languages*, Thesis and/or Apprenticeship Proposals 2006-2007, System and Software Engineering Lab, Disponível em: <https://ssel.vub.ac.be/ssel/teaching:thesis0607:proposals:aoworkflow>.

WANG, J., MADNICK, S., 1989, The Inter-Database Instance Identification Problem in Integrating Autonomous Systems, In: *Proceedings of the Fifth International Conference on Data Engineering*, Washington, DC, USA, pp. 46-55.

WEBER, M., "Petri Net Markup Language". PNML. Humboldt-Universität zu Berlin Berlin, Germany, Disponível em: http://www2.informatik.hu-berlin.de/top/ pnml/about.html.

WERNER, C.M.L., 1992, *Reutilição de Software no Desenvolvimento de Software Científico.*, Tese de Doutorado. COPPE/UFRJ, Rio de Janeiro, Brasil.

WESTERGAARD, M., 2006, *The BRITNeY Suite: A Platform for Experiments*, Department of Computer Science, University of Aarhus, Germany, Disponível em: <www.daimi.au.dk/CPnets/workshop06/cpn/ papers/Paper06.pdf>.

WIEDERHOLD, G., 1992, "Mediators in the Architecture of Future Information Systems", *IEEE Computer*, Los Alamitos, CA, USA, v. 25, pp. 38-49, Disponível em: <http://citeseerx.ist.psu.edu/ viewdoc/summary?doi=10.1.1.11.8981>

XEXÉO, G., 2007, *Modelagem de Sistemas de Informação: Da análise de requisitos ao modelo de interface*, Livro, Rio de Janeiro, Brasil, Disponível em: <http://wiki.xexeo.org/tiki-download_file.php?fileId=188>

YALAMANCHI, A., SRINIVASAN, J., GAWLICK, D., 2003, Managing Expressions as Data in Relational Database Systems, In: *Proceedings of the 2003 CIDR Conference*, Asilomar, CA, USA, Disponível em: <http://citeseer.ist.psu.edu/ yalamanchi03managing.html>.

ZHANG, Y., PAXSON, V., 2000, Detecting backdoors, In: *Proceedings of 2000 USENIX Security Symposium*, Denver, Colorado, pp. 12-12, Disponível em: <http://eprints.kfupm.edu.sa/ 34616/>.

ZIMMER, D., MECKENSTOCK, A., UNLAND, R., 1996a, Using Petri nets for rule

termination analysis, In: *Proceedings of the workshop on on Databases: active and real-time*, Rockville, Maryland, United States, pp. 29-32.

ZIMMER, D., MECKENSTOCK, A., UNLAND, R., 1996b, *Rule Termination Analysis based on Petri Nets*, Cadlab Report 3, Cadlab, Furstenallee 11, 33102 Paderborn, Germany, Disponível em: <http://citeseerx.ist.psu.edu/ viewdoc/summary? doi=10.1.1.53.1733>.

Anexo I – Biblioteca de Funções em CPN ML

De modo a permitir a utilização da ferramenta CPN Tools na execução dos padrões modelados em RPAN, tornou-se necessária a criação de um conjunto de funções para a manipulação dos dados processados. Os seguintes pacotes de funções foram criados:

- Generic Functions – funções genéricas e de conversão entre tipos;
- String Functions – funções para manipulação de strings;
- Validation Functions – funções de validação de dados e eventos a serem processados pelas RPAN;
- Data Functions – funções de manipulação de dados da forma (id : INT, metadata: STRING, valor: STRING);
- Event Functions – funções de manipulação de eventos primitivos e compostos, e marcas-modelo. São divididas nos seguintes subgrupos:
    - Combinatorial: funções que realizam a análise combinatória de eventos;
    - HandleEvent: funções para conversão e obtenção de eventos segundo alguns crittérios;
    - OrderEventList: funções para ordenar uma lista de evento de acordo com o tempo de ocorrência do evento;
    - SimilarOrRepeatedEvents: funções para manusear eventos repetidos ou similares;
    - EventTemplate: funções para manipular marcas-modelo;
    - CompositeEvents: funções para manusear e ajudar na detecção de eventos compostos;
    - AssembleCompositeEvents: funções para montar eventos compostos a partir de eventos primitivos;
    - EventCleaning: funções que eliminam eventos detectados segundo critérios diversos;
- Socket Functions – funções para comunicação das RPANs com outras aplicações usando *sockets*. São divididas nos seguintes subgrupos:

- StringToData: funções para converter strings que representam dados serializados recebidos de aplicações externas em dados que possam ser manipulados pelas RPANs;
- DataToString: funções que serializam os dados nas RPANs na forma de strings;
- StringToEvent: funções para converter strings que representam eventos serializados recebidos de aplicações externas em eventos que possam ser manipulados pelas RPANs;
- EventToString: funções que serializam os eventos nas RPANs na forma de strings;
- Communication: funções que permitem a conexão e transmissão de dados entre as RPANs e as aplicações externas;

- Pre-processing: funções que permitem, caso seja necessário, o pré-processamento dos dados;
- Similarity: funções que realizam cálculos de similaridade;
- Patterns: funções relativas dos padrões de eliminação de dados. São divididas nos seguintes subgrupos:
  - Data Lists: funções que realizam a manipulação de listas de dados. São divididas nos seguintes subgrupos:
    - Check: funções que fazem verificações sobre listas de dados;
    - Execute: funções que executam ações sobre listas de dados;
  - Pattern Dependents: funções específicas de certos padrões de eliminação de dados. São divididas nos seguintes subgrupos:
    - Check: funções que fazem verificações para os padrões de eliminação de dados;
    - Execute: funções que executam ações para os padrões de eliminação de dados.

Além das funções, foi necessário definir os tipos de dados e constantes utilizados nas funções, além das variáveis utilizadas nos padrões em redes de Petri. Todos esses elementos e as funções criadas, organizadas por pacotes, são apresentados a seguir:

```
(**********Standard declarations**********)

colset UNIT = unit;
colset INT = int;
colset BOOL = bool;
colset STRING = string;


(**********Declarations - Generic**********)

colset INT_TIME = INT timed;
colset BOOL_TIME = BOOL timed;
colset STRING_TIME = STRING timed;
colset INT_LIST = list INT;
colset STRING_LIST = list STRING;
colset LST = list STRING_LIST;
colset SMALL_INT = int with 1..100;
colset E = unit with e;
colset BI = product BOOL * INT;


(**********Declarations - Data**********)

colset DATA_TUPLE=
record idData:INT*metadata:STRING*value:STRING;
colset DATA = list DATA_TUPLE;


(**********Declarations - Event**********)

colset PRIMITIVE =
record eventTime:INT*eventUser:STRING*
eventTransaction:STRING*dataId:INT*parameters: DATA timed;
colset PRIMITIVE_LIST = list PRIMITIVE;
colset COMPOSITE1 =
record eventTime:INT*eventType:STRING*parameters: DATA*
eventList:PRIMITIVE_LIST;
colset EVENT = union
primitive:PRIMITIVE + composite:COMPOSITE1;
colset EVENT_LIST = list EVENT timed;
colset COMPOSITE =
record eventTime:INT*eventType:STRING*parameters: DATA*
eventList:EVENT_LIST;
colset LISTofEL = list EVENT_LIST;
colset PRIMITIVE_TEMPLATE =
record eventTime:INT*eventUser:STRING*
eventTransaction:STRING*dataId:INT*parameters: DATA timed;
colset PRIMITIVE_TEMPLATE_LIST = list PRIMITIVE_TEMPLATE;
colset COMPOSITE_TEMPLATE1 =
record eventInitialTime:INT*eventFinalTime:INT*priorityLevel:INT*
eventType:STRING*parameters: DATA*eventList:PRIMITIVE_TEMPLATE_LIST;
colset EVENT_TEMPLATE = union
primitiveTemplate:PRIMITIVE_TEMPLATE + compositeTemplate:COMPOSITE_TEMPLATE1;
colset EVENT_TEMPLATE_LIST = list EVENT_TEMPLATE timed;
colset COMPOSITE_TEMPLATE =
record eventInitialTime:INT*eventFinalTime:INT*priorityLevel:INT*
```

```
(**********Variables**********)

eventType:STRING*parameters: DATA*eventList:EVENT_TEMPLATE_LIST;
var b, status: BOOL;
var u: UNIT;
var k, i, j, i1, i2, j1, j2 : INT;
var s, s1, s2, s3: STRING;
var il, il1, il2: INT_LIST;
var st1, st2: STRING_LIST;
var lst: LST;
var ep, ep1, ep2 : PRIMITIVE;
var ec, ecOut : COMPOSITE;
var ev, event : EVENT;
var lp, lpOut, le, le1, le2, le3, le4, lec, lparam, dlec : EVENT_LIST;
var strList, lpe: PRIMITIVE_LIST;
var lel : LISTofEL;
var et : EVENT_TEMPLATE;
var etl, etl1, etl2 : EVENT_TEMPLATE_LIST;
var getData, getData1, getData2, getData3,
readData, readData1, readData2,
processData,
prohibitedData, notProhibitedData,
similarData, notSimilarData,
allowedData, oldData, uselessData,
relevantData, irrelevantData : DATA;

(**********Constants - Connections**********)

val eventConnectionName="cpnClient";
val eventHostAddress="127.0.0.1";
val eventPortNumber=7500;
val dataConnectionName="cpnServer";
val dataHostAddress="127.0.0.1";
val dataPortNumber=9000;
val stemConnectionName="stemming";
val stemHostAddress="127.0.0.1";
val stemPortNumber=9001;
val stemSteps=4;

(**********Constants - Data Killing**********)

val paramComplement="complement";
val paramCM="comparedMetadata";
val paramCS="comparingSignal";
val paramCV="comparedValue";
val paramSS="similarityStrategy";
val paramSC="similarityCoeficient";
val paramST="stemming";
val paramSW="stopwords";
val paramLG="language";
val paramCT="currentTimestamp";
val paramPT="periodTimestamp";
val paramMP="minimumParam";
val paramDataRelationship = "dependentDataRelationship";
val paramDataStrategy = "dependentDataStrategy";
```

```
val paramConnective = “connective”;
val paramOrder = “orderMetadata”;
val paramSort = “sort”;
val paramSL = “similarityLevel”;
val paramOrderDefault = “ascendent”;
val paramMergeDefault = “default”;

(**********Functions - Generic**********)

(*Returns a random number*)
fun random() = SMALL_INT.ran();

(*Returns the time as an integer*)
fun intTime() = IntInf.toInt (time())

(*Returns the time as a string*)
fun strTime() = ModelTime.toString(time());

(*Returns the biggest number between the parameters a and b*)
fun max(a, b) = if a>b then a else b;

(*Returns the lowest number between the parameters a and b*)
fun min(a, b) = if a<b then a else b;

(*Converts an integer to a string*)
fun intToStr (value) = Int.toString(value);

(*Converts a string to an integer*)
fun strToInt (str) = INT.input (TextIO.openString str);

(*Returns 0 if the string is not a valid integer, and the integer value otherwise*)
fun checkAndConvertStrToInt i =
let
fun f (SOME i) = i
|f NONE = 0
in f (Int.fromString i)
end;

(*Converts a value from real to int. d is the number of digits to the right of the decimal point*)
fun realToInt (r, d) =
IntInf.toInt (RealToIntInf d r)

(*Converts a value from int to real. d is the number of digits to the right of the decimal point*)
fun intToReal (i, d) =
(IntInfToReal d (IntInf.fromInt i))

(*Replaces all ocurrences of param1 in str by param2. The result is stored in strAux*)
fun replaceAll(str, param1, param2, pos, strAux)=
if pos<=(String.size(str)-String.size(param1))
then
   if substring(str, pos, String.size(param1)) = param1
   then replaceAll(str,  param1, param2, pos+String.size(param1),
                     strAux^param2)
   else replaceAll(str,  param1, param2, pos+1,
                     strAux^substring(str, pos, 1))
else strAux^substring(str, pos, String.size(str)-pos);
```

```
fun getHead(head::list) : EVENT_LIST = [head]
|getHead(nil) = [];

(*Searchs an element in an ordered list and returns the position of the element, if the element is
equal to the value*)
fun searchInOrderedIntList(list, value : INT, aux) =
let
val a = if list<>[] then length list else 0
val b = if a<>0 then ((a div 2) + 1) else 0
val c = if List.nth(list, ((length list) - 1)) >= List.nth(list, 0) then “ascendent” else
“descendent”
in
if list<>[]
then
   if c = “ascendent”
   then
      if value = List.nth(list, b-1)
      then aux + b
      else
         if value < List.nth(list, b-1)
         then searchInOrderedIntList(List.take(list,b-1),value, aux)
         else searchInOrderedIntList(List.drop(list,b),value,aux+b)
   else
      if value = List.nth(list, b-1)
      then aux + b
      else
         if value > List.nth(list, b-1)
         then searchInOrderedIntList(List.take(list,b-1),value, aux)
         else searchInOrderedIntList(List.drop(list,b),value,aux+b)
else 0
end;

(*Searchs an element in an ordered list and returns the position of the element, if the element is
equal to the value*)
fun searchInOrderedStringList(list, value : STRING, aux) =
let
val a = if list<>[] then length list else 0
val b = if a<>0 then ((a div 2) + 1) else 0
val c = if STRING.lt(List.nth(list, ((length list) - 1)), List.nth(list, 0)) then “descendent” else
“ascendent”
in
if list<>[]
then
   if c = “ascendent”
   then
      if value = List.nth(list, b-1)
      then aux + b
      else
         if STRING.lt(value,List.nth(list, b-1))
         then searchInOrderedStringList(List.take(list,b-1),value, aux)
         else searchInOrderedStringList(List.drop(list,b),value,aux+b)
   else
      if value = List.nth(list, b-1)
      then aux + b
      else
```

```
        if STRING.lt(value,List.nth(list, b-1)) = false
        then searchInOrderedStringList(List.take(list,b-1),value, aux)
        else searchInOrderedStringList(List.drop(list,b),value,aux+b)
else 0
end;

(*Inserts a integer in a ordered integer list*)
fun insertIntInOrderedList(head::list, i, auxList) =
if i>head
then insertIntInOrderedList(list, i, auxList^^[head])
else auxList^^[i]^^[head]^^list
|insertIntInOrderedList(nil, i, auxList) = auxList^^[i];

fun orderIntList(list)= sort INT.lt list;

fun orderStringList(list) = sort STRING.lt list;

(*Converts a string list to an int list, if it is possible.  In other case, it returns []*)
fun convertStringListToIntList(el::list, auxList)=
if (el<>"0" andalso checkAndConvertStrToInt(el) = 0)
then []
else convertStringListToIntList(list,
                        auxList^^[checkAndConvertStrToInt(el)])
|convertStringListToIntList(null, auxList) = auxList;

(*Converts an int list to a string list*)
fun convertIntListToStringList(el::list, auxList)=
convertIntListToStringList(list,auxList^^[intToStr(el)])
|convertIntListToStringList(null, auxList) = auxList;

(*Orders the list from the smallest to the largest, considering their elements like numbers, if it is
possible.
In other case, orders the list considering their elements like characters*)
fun orderListByNumOrChar(list) =
let
val a = convertStringListToIntList(list,[])
in
if a<>[]
then convertIntListToStringList(orderIntList(a),[])
else orderStringList(list)
end;

(*Merges two lists, according to param that is equal to lowest or highest, the default value*)
fun mergeIntList(list1, list2, param, auxList) =
if (list1<>[])andalso(list2<>[])
then
   if (param <> "lowest")
   then
      if (hd list1) > (hd list2)
      then mergeIntList(tl list1, tl list2, param, auxList^^[hd list1])
      else mergeIntList(tl list1, tl list2, param, auxList^^[hd list2])
   else
      if (hd list1) < (hd list2)
      then mergeIntList(tl list1, tl list2, param, auxList^^[hd list1])
      else mergeIntList(tl list1, tl list2, param, auxList^^[hd list2])
else auxList
```

```
(**********Functions - String**********)

(*Returns the initial position of the first param in str, starting from pos.
The minimum value of pos is 0. If the parameter is not found, it will return the size of  the string.*)
fun findFirstParam(pos, str, param) : INT=
if pos<=(String.size(str) - String.size(param))
then
   if substring(str, pos, String.size(param)) = param
   then pos
   else findFirstParam(pos+1, str, param)
else String.size(str);

(*Returns the initial position of the last param in str, starting from pos.
The minimum value of pos is 0. If the parameter is not found, it will return the size of  the string.*)
fun findLastParam(pos, str, param) =
if pos>0 andalso pos<=(String.size(str) - String.size(param))
then
   if substring(str, pos, String.size(param)) = param
   then pos
   else findLastParam(pos-1, str, param)
else
   if  (pos=0)
   then if substring(str, pos, String.size(param)) = param
         then 0
         else String.size(str)
   else findLastParam((String.size(str)-String.size(param)), str, param);

(*Returns the final position + 1 of the first param in str, starting from pos.
The minimum value of pos is 0. If the parameter is not found, it will return 0.*)
fun findAfterFirstParam(pos, str, param) : INT=
if pos<=(String.size(str) - String.size(param))
then
   if substring(str, pos, String.size(param)) = param
   then pos+String.size(param)
   else findAfterFirstParam(pos+1, str, param)
else 0;

(*Returns in str the first param between prefix and sufix*)
fun getFirstParamBetweenPrefixAndSufix(pos, str, prefix, sufix) =
if findAfterFirstParam(pos, str, prefix)=0
then ""
else
   if findAfterFirstParam(findAfterFirstParam(pos, str, prefix), str, sufix)=0
   then ""
   else substring(str,
                  findAfterFirstParam(pos, str, prefix),
                  findAfterFirstParam(findAfterFirstParam(pos, str, prefix), str, sufix) -
                  findAfterFirstParam(pos, str, prefix) - String.size(sufix));

fun findAllParam(pos, str, param, auxList) =
if pos<=(String.size(str)-String.size(param))
then
   if substring(str, pos, String.size(param)) = param
   then findAllParam(pos+String.size(param), str,  param, pos::auxList)
   else findAllParam(pos+1,str,  param, auxList)
else rev auxList;
```

```
(*Deletes all ocurrences of a substring in a string*)
fun delParam(str, param, aux) =
if findAllParam(0,str,param,[])<>nil
then
   delParam (substring(str,
                  hd (findAllParam(0,str,param,[])) + String.size(param),
                  String.size(str) -
                     (hd (findAllParam(0,str,param,[])) + String.size(param)) ),
               param,
               aux^substring(str, 0,
                              hd (findAllParam(0,str,param,[])) ))
else aux^str;

fun splitParam(str, param, auxList) =
if findAllParam(0,str,param,[])<>nil
then
   splitParam (substring(str,
                  hd (findAllParam(0,str,param,[])) + String.size(param),
                  String.size(str) -
                     (hd (findAllParam(0,str,param,[])) + String.size(param)) ),
               param,
               substring(str, 0,
                           hd (findAllParam(0,str,param,[]))::auxList )
else rev (rmall "" (str::auxList));

(**********Functions - Validation**********)

(*Tests if a string is just a data or a data parameter*)
fun isParam(str)=
let
val a = findFirstParam(0, str, "<at2>parameter</at2>")
in
if a <> 0
then
   if getFirstParamBetweenPrefixAndSufix(a, str, "<at3>",
"</at3>")="true"
   then true
   else false
else false
end;

(*Tests if a string is a dataParameter2*)
fun isData2(str)=
let
val a = findFirstParam(0, str, "<at2>dataParameter2</at2>")
in
if a <> 0
then
   if getFirstParamBetweenPrefixAndSufix(a, str, "<at3>",
"</at3>")="true"
   then true
   else false
else false
end;
```

```
(*Tests if a string is a data parameter*)
fun isData(str)=
let
val a = findFirstParam(0, str, "<at2>dataParameter</at2>")
in
if a <> 0
then
   if getFirstParamBetweenPrefixAndSufix(a, str, "<at3>",
"</at3>")="true"
   then true
   else false
else false
end;

(*Tests if a string is an event*)
fun isEvent(str)=
let
val a = findFirstParam(0, str, "<at2>registerEvent</at2>")
in
if a <> 0
then
   if getFirstParamBetweenPrefixAndSufix(a, str, "<at3>",
"</at3>")="true"
   then true
   else false
else false
end;

(*Tests if a string is an event template*)
fun isEventTemplate(str)=
let
val a = findFirstParam(0, str, "<at2>registerEventTemplate</at2>")
in
if a <> 0
then
   if getFirstParamBetweenPrefixAndSufix(a, str, "<at3>",
"</at3>")="true"
   then true
   else false
else false
end;

(*Tests if a string is to unregister an event template*)
fun isUnregisterEvent(str)=
let
val a = findFirstParam(0, str, "<at2>unregisterEventTemplate</at2>")
in
if a <> 0
then
   if getFirstParamBetweenPrefixAndSufix(a, str, "<at3>",
"</at3>")="true"
   then true
   else false
else false
end;
```

```
(**********Functions - Data**********)

(*Checks if a data contains a metadata that is equal to param*)
fun metadataExists({idData, metadata, value}::list, param) =
if metadata=param
then true
else metadataExists(list, param)
|metadataExists(nil, param)=false;

(*Gets the first data that has the metadata equals x. Moreover, it updates the metadata value to y*)
fun getAndUpdateFirstData({idData, metadata, value}::listOfRecords, x, y) =
   if metadata=x
   then [{idData=idData, metadata=metadata, value=y}]
   else getAndUpdateFirstData(listOfRecords, x, y)
|getAndUpdateFirstData(nil, x, y) = nil;

(*Returns data*)
fun putData(data) = data;

(*Returns the elements on list1 that do not exist on list2*)
fun putSub(list1, list2) =
listsub list1 (intersect list1 list2);

(*Gets the first data value that has the metadata equals x*)
fun getFirstDataValue({idData, metadata, value}::listOfRecords, x : STRING) : STRING =
   if metadata=x
   then value
   else getFirstDataValue(listOfRecords, x)
|getFirstDataValue(nil, x) = "0";

(*Gets the first data tuple that has the metadata equals x*)
fun getFirstDataTuple({idData, metadata, value}::listOfRecords, x : STRING) : DATA=
   if metadata=x
   then [{idData=idData, metadata=metadata, value= value}]
   else getFirstDataTuple(listOfRecords, x)
|getFirstDataTuple(nil, x) = nil;

(*Gets the (n-count+1) data tuple that has the metadata equals x*)
fun getNDataTuple({idData, metadata, value}::listOfRecords, x : STRING, n : INT, count : INT) : DATA=
   if metadata=x
   then
      if (n=count)
      then [{idData=idData, metadata=metadata, value= value}]
      else getNDataTuple(listOfRecords, x, n, count+1)
   else getNDataTuple(listOfRecords, x, n, count)
|getNDataTuple(nil, x, n, count) = nil;

(*Gets the id of a data tuple*)
fun getIdFromDataTuple({idData, metadata, value}::list) = idData
|getIdFromDataTuple(nil) =0;

(*Gets the data identified by idData equals param*)
fun getDataById({idData, metadata, value}::data, param, auxList) =
if idData=param
then getDataById(data, param,
      {idData=idData, metadata=metadata, value= value}::auxList)
```

```
else getDataById(data, param, auxList)
|getDataById(nil, param, auxList) = rev auxList;

(*Gets the data without repetition identified by idData equals param*)
fun getDataByIdWithoutRepetition(data, param, auxList) =
remdupl(getDataById(data, param, auxList));

(*Gets the data identified by idData equals id and metadata equals met*)
fun getAllDataByIdAndMetadata({idData, metadata, value}::listOfRecords, id, met, auxList) =
if idData=id andalso metadata=met
then getAllDataByIdAndMetadata(listOfRecords, id, met,
      {idData=idData, metadata=metadata, value= value}::auxList)
else getAllDataByIdAndMetadata(listOfRecords, id, met, auxList)
|getAllDataByIdAndMetadata(nil, id, met, auxList) = auxList;

fun getEventFromParam(param : DATA)=
{eventTime=0,
eventUser="System",
eventTransaction="delete",
dataId=0, parameters=
[""]}

(*Gets the data that are dependent of another data*)
fun getDataDependent(data : DATA, idDataList : INT_LIST, param : DATA, paramCopy : DATA, auxList :
DATA) =
let
val scope = if idDataList<>[] then getDataById(data, hd idDataList, []) else [];
val a = getFirstDataValue(param, paramDataRelationship);
val b = getFirstDataValue(scope, a);
val c = getFirstDataTuple(scope, a);
val d = getFirstDataTuple(param, paramDataRelationship);
val e = getFirstDataValue(putSub(param, d), paramDataRelationship);
in
if data<>[]
then
   if b<>"0"
   then getDataDependent(putSub(data, c), idDataList^^[strToInt(b)], param, paramCopy,
                              auxList^^getDataById(data, strToInt(b), []))
   else if e<>"0" andalso getFirstDataValue(scope, e)<>"0"
        then getDataDependent(putSub(data, getFirstDataTuple(scope, e)),
                                    idDataList^^[strToInt(getFirstDataValue(scope, e))],
                                    putSub(param, d), paramCopy,
                                    auxList^^getDataById(data, strToInt(getFirstDataValue(scope, e)), []))
        else if tl(remdupl(idDataList))<>[]
             then getDataDependent(data, tl(remdupl(idDataList)), paramCopy, paramCopy, auxList)
             else remdupl(auxList)
else auxList
end;

(*Gets data that has a data dependent*)
fun getDataThatHasDependent(data : DATA, idDataList : INT_LIST, param : DATA, paramCopy: DATA, auxList
: DATA) =
let
val scope = if idDataList<>[] then getDataById(data, hd idDataList, []) else [];
val a = getFirstDataValue(param, paramDataRelationship);
val b = getFirstDataValue(scope, a);
```

```
val c = getFirstDataTuple(param, paramDataRelationship);
val d = getFirstDataValue(putSub(param, c), paramDataRelationship);
in
if data<>[]
then
   if b<>"0"
   then getDataThatHasDependent(putSub(data, scope), tl idDataList, paramCopy, paramCopy, auxList^^scope)
   else if d<>"0"
        then getDataThatHasDependent(data, idDataList, putSub(param, c), paramCopy, auxList)
        else if tl idDataList<>[]
        then getDataThatHasDependent(data, tl idDataList, paramCopy, paramCopy, auxList)
        else auxList
else auxList
end;

(*Gets the ids from a data list*)
fun getIdDataListFromData({idData, metadata, value}::data, auxList : INT_LIST) =
getIdDataListFromData(data, idData::auxList)
|getIdDataListFromData(nil, auxList) = rev auxList;

(*Gets all first tuples with different metadata identified by idData equals param*)
fun getAllFirstMetadataFromDataById({idData, metadata, value}::listOfRecords, param, auxList) : DATA=
if idData=param andalso metadataExists(auxList, metadata)=false
then getAllFirstMetadataFromDataById(listOfRecords, param,
      {idData=idData, metadata=metadata, value= value}::auxList)
else getAllFirstMetadataFromDataById(listOfRecords, param, auxList)
|getAllFirstMetadataFromDataById(nil, param, auxList) = auxList;

(*Gets and updates the data values of a data identified by idData equals x and metadata equals y*)
fun updateMetadataValue({idData, metadata, value}::listOfRecords,
x, y, newValue, auxList) =
if idData=x andalso metadata=y
then updateMetadataValue(listOfRecords, x, y, newValue,
      {idData=idData, metadata=metadata, value=newValue}::auxList)
else  updateMetadataValue(listOfRecords, x, y, newValue,
      {idData=idData, metadata=metadata, value=value}::auxList)
|updateMetadataValue(nil, x, y, newValue, auxList) = auxList;

(**********Functions - Event - Combinatorial**********)

(*Gets the arrangement of events in a list of event*)
fun arrangement1(head::dList : LISTofEL, dListCopy, auxList1, auxList1Copy, auxList2, ord) : LISTofEL =
if (length auxList1 > 0)
then
   if (contains (hd auxList1) (head)=false)
   then arrangement1(head::dList, dListCopy, tl auxList1, auxList1Copy, ((hd auxList1)^^(head))::auxList2, ord)
   else arrangement1(head::dList, dListCopy, tl auxList1, auxList1Copy, auxList2, ord)
else arrangement1(dList, dListCopy, auxList1Copy, auxList1Copy, auxList2, ord)
|arrangement1(nil, dListCopy, auxList1, auxList1Copy, auxList2, ord)=
   if ord>2
   then arrangement1(dListCopy, dListCopy, auxList2, auxList2, [], ord-1)
   else auxList2;

(*Returns if  the event c1 is less than c2*)
fun lt_eventList(c1, c2) = EVENT.lt(c1,c2);
```

```
(*Returns a event list ordered*)
fun orderEventList(head::eventList, auxList1, auxList2) : EVENT_LIST=
if (auxList1=[]) orelse (lt_eventList(hd (rev auxList1), head))
then orderEventList(eventList, auxList1^^[head]^^auxList2, [])
else orderEventList(head::eventList, rev (tl (rev auxList1)), (hd (rev auxList1))::auxList2)
|orderEventList(nil, auxList1, auxList2) = auxList1;

(*Returns a list of event list ordered*)
fun processOrderEventList(head::listOfEventList, auxList) : LISTofEL=
processOrderEventList(listOfEventList, orderEventList(head, [], [])::auxList)
|processOrderEventList(nil, auxList)= auxList;

(*Gets the arrangement of events in a list of event list*)
fun arrangement(dList, ord) = arrangement1(dList, dList, dList, dList, [], ord) : LISTofEL;

(*Gets the combination of events in LISTofEL*)
fun combination(dList: LISTofEL, ord) : LISTofEL =
remdupl (processOrderEventList
(arrangement1(dList, dList, dList, dList, [], ord), []));

(**********Functions - Event - HandleEvent**********)

fun convertEventListToListOfEventList(head::eventList, eventOfEventList) : LISTofEL=
convertEventListToListOfEventList(eventList, [head]::eventOfEventList)
|convertEventListToListOfEventList(nil, eventOfEventList) = rev eventOfEventList;

fun convertListOfEventListToEventList(head::eventOfEventList, eventList) =
convertListOfEventListToEventList(eventOfEventList, eventList^^head)
|convertListOfEventListToEventList(nil, eventList) = eventList;

fun getPrimitiveValue (primitive(value)) = value
| getPrimitiveValue _ =
{eventTime=0, eventUser="",
eventTransaction="", dataId=0, parameters=[]};

fun getCompositeValue (composite(value)) = value
| getCompositeValue _ =
{eventTime=0, eventType="", parameters=[], eventList=[]};

fun eventListToPrimitiveList(head::list, auxList) : PRIMITIVE_LIST =
if (EVENT.of_primitive(head)=true)
then eventListToPrimitiveList(list, getPrimitiveValue(head)::auxList)
else eventListToPrimitiveList(list, auxList)
|eventListToPrimitiveList(nil, auxList) = rev(auxList);

fun primitiveListToEventList(head::list, auxList) : EVENT_LIST=
primitiveListToEventList(list, primitive(head)::auxList)
|primitiveListToEventList(nil, auxList) = rev(auxList);

fun primitiveTemplateListToEventTemplateList(head::list, auxList) :
EVENT_TEMPLATE_LIST=
primitiveTemplateListToEventTemplateList(list, primitiveTemplate(head)::auxList)
|primitiveTemplateListToEventTemplateList(nil, auxList) = rev(auxList);
```

```
fun getPrimitiveEventListFromFirstCompositeEvent(head::eventList : EVENT_LIST) :
EVENT_LIST=
if EVENT.of_composite(head)=true
then primitiveListToEventList((#eventList (getCompositeValue(head))), [])
else getPrimitiveEventListFromFirstCompositeEvent(eventList)
|getPrimitiveEventListFromFirstCompositeEvent(nil) = [];

fun getEventTypeFromFirstCompositeEvent(head::eventList : EVENT_LIST) =
if EVENT.of_composite(head)=true
then (#eventType (getCompositeValue(head)))
else getEventTypeFromFirstCompositeEvent(eventList)
|getEventTypeFromFirstCompositeEvent(nil) = "";

fun getEventParamFromFirstCompositeEvent(head::eventList : EVENT_LIST) =
if EVENT.of_composite(head)=true
then (#parameters (getCompositeValue(head)))
else getEventParamFromFirstCompositeEvent(eventList)
|getEventParamFromFirstCompositeEvent(nil) = [];

fun getLastEventTime(head::list : EVENT_LIST) : INT =
if (EVENT.of_primitive(head)=true)
then
   if  (#eventTime (getPrimitiveValue(head))) > getLastEventTime (list)
   then  (#eventTime (getPrimitiveValue(head)))
   else  getLastEventTime (list)
else
   if  (#eventTime (getCompositeValue(head))) > getLastEventTime (list)
   then (#eventTime (getCompositeValue(head)))
   else  getLastEventTime (list)
| getLastEventTime (nil) = 0;

fun updatePrimitiveTime(record : PRIMITIVE, period) =
PRIMITIVE.set_eventTime record (period+intTime());

(**********Functions - Event - OrderEventList**********)

fun getFirstEventByEventTime(head::eventList : EVENT_LIST, eventTime:INT) :EVENT_LIST=
if (EVENT.of_primitive(head)=true) andalso
(#eventTime (getPrimitiveValue(head)) = eventTime)
then [head]
else
   if (EVENT.of_composite(head)=true) andalso
      (#eventTime (getCompositeValue(head)) = eventTime)
   then [head]
   else getFirstEventByEventTime(eventList, eventTime)
|getFirstEventByEventTime(nil, eventTime)=[];

fun orderEventListByEventTimeAccordingToList(head::list, eventList, auxList) =
if getFirstEventByEventTime(eventList, head)<>[]
then orderEventListByEventTimeAccordingToList(list,
   rm (hd (getFirstEventByEventTime(eventList, head))) eventList,
   auxList^^getFirstEventByEventTime(eventList, head))
else orderEventListByEventTimeAccordingToList(list, eventList, auxList)
|orderEventListByEventTimeAccordingToList(nil, eventList, auxList)=auxList;
```

```
fun getEventTimeListFromEventList(head::eventList, auxList) : INT_LIST =
if (EVENT.of_primitive(head)=true)
then
   getEventTimeListFromEventList(eventList,
                        (#eventTime (getPrimitiveValue(head)))::auxList)
else
   if (EVENT.of_composite(head)=true)
   then
      getEventTimeListFromEventList(eventList,
                           (#eventTime (getCompositeValue(head)))::auxList)
   else
      []
|getEventTimeListFromEventList(nil, auxList)=auxList;

fun orderEventListByEventTime(eventList) =
let
val i = getEventTimeListFromEventList(eventList, [])
val j = orderIntList(i)
in
orderEventListByEventTimeAccordingToList(j, eventList, [])
end;

(**********Functions - Event - SimilarOrRepeatedEvent**********)

(*Returns true if two primitive events are equal or compatible*)
fun comparePrimitiveEvents(event1 : PRIMITIVE, event2 : PRIMITIVE) =
if (#eventTime event1)>=(#eventTime event2)
andalso
((#eventUser event1)=(#eventUser event2)
   orelse (#eventUser event2) = "any")
andalso
((#eventTransaction event1)=(#eventTransaction event2)
   orelse (#eventTransaction event2) = "any")
andalso
((#dataId event1)=(#dataId event2)
   orelse (#dataId event2) = 0)
andalso
((#parameters event1)=(#parameters event2)
   orelse (#parameters event2) = [])
then
   true
else
   false;

(*Gets a especific primitive event in a primitive event list*)
fun getSpecificPrimitiveEventOnOneList(head::list1 : PRIMITIVE_LIST, event : PRIMITIVE) =
if comparePrimitiveEvents(event, head)
then [event]
else getSpecificPrimitiveEventOnOneList(list1, event)
|getSpecificPrimitiveEventOnOneList(nil, event) =[];

(*Returns the list of events that exists on list1 and exists on list2. The value returned is the value
from list1.*)
fun getEqualPrimitiveEventOnTwoLists(head::list1 : PRIMITIVE_LIST, list2 : PRIMITIVE_LIST, auxList)
: PRIMITIVE_LIST =
if getSpecificPrimitiveEventOnOneList(list2, head)<>[]
```

```
then
   getEqualPrimitiveEventOnTwoLists
      (list1, list2,
      getSpecificPrimitiveEventOnOneList(list2, head)^^auxList)
else getEqualPrimitiveEventOnTwoLists(list1, list2, auxList)
|getEqualPrimitiveEventOnTwoLists(nil, list2, auxList) = rev(auxList);

(*Returns true if two events are equal or compatible*)
fun compareEvents(event1 : EVENT, event2 : EVENT) =
if (EVENT.of_primitive(event1)=true) andalso (EVENT.of_primitive(event2)=true)
andalso
(#eventTime (getPrimitiveValue(event1)))>=(#eventTime (getPrimitiveValue(event2)))
andalso
((#eventUser (getPrimitiveValue(event1)))=(#eventUser (getPrimitiveValue(event2)))
   orelse (#eventUser (getPrimitiveValue(event2))) = "any")
andalso
((#eventTransaction (getPrimitiveValue(event1)))=(#eventTransaction (getPrimitiveValue(event2)))
   orelse (#eventTransaction (getPrimitiveValue(event2))) = "any")
andalso
((#dataId (getPrimitiveValue(event1)))=(#dataId (getPrimitiveValue(event2)))
   orelse (#dataId (getPrimitiveValue(event2))) = 0)
andalso
((#parameters (getPrimitiveValue(event1)))=(#parameters (getPrimitiveValue(event2)))
   orelse (#parameters (getPrimitiveValue(event2))) = [])
then
   true
else
   if (EVENT.of_composite(event1)=true) andalso (EVENT.of_composite(event2)=true)
   andalso
   (#eventTime (getCompositeValue(event1)))>=(#eventTime (getCompositeValue(event2)))
   andalso
   ((#eventType (getCompositeValue(event1)))=(#eventType (getCompositeValue(event2)))
      orelse (#eventType (getCompositeValue(event2))) = "any")
   andalso
   ((#parameters (getCompositeValue(event1)))=(#parameters (getCompositeValue(event2)))
      orelse (#parameters (getCompositeValue(event2))) = [])
   andalso
   (getEqualPrimitiveEventOnTwoLists(
      (#eventList (getCompositeValue(event2))),(#eventList (getCompositeValue(event1))), []) )=
   (#eventList (getCompositeValue(event1)))
   then
      true
   else
      false;

(*Gets a especific event in a event list*)
fun getSpecificEventOnOneList(head::list1 : EVENT_LIST, event : EVENT) =
if compareEvents(event, head)
then [event]
else getSpecificEventOnOneList(list1, event)
|getSpecificEventOnOneList(nil, event) =[];

(*Returns the list of events that exists on list1 and exists on list2.
The order of the elements obeys list1. The element returned is the element from list1*)
fun getEqualEventOnTwoLists(head::list1 : EVENT_LIST, list2 : EVENT_LIST, auxList : EVENT_LIST)
: EVENT_LIST=
```

```
if  getSpecificEventOnOneList(list2, head)<>[]
then
   getEqualEventOnTwoLists
      (list1, list2, getSpecificEventOnOneList(list2, head)^^auxList)
else getEqualEventOnTwoLists(list1, list2, auxList)
|getEqualEventOnTwoLists(nil, list2, auxList) = rev(auxList);

(*Returns the primitive elements of one list that has a number of repetitions bigger than n,  n>=1.
n=1 returns all element. The eventTime is not considered  in the processing,
if the list is in ascending order by eventTime.*)
fun selectRepeated(head::list : EVENT_LIST, n, auxList : LISTofEL) =
if (n>0) andalso
(length (getEqualEventOnTwoLists(list, [head], [])) >= n-1)
then
   selectRepeated(
      listsub list (getEqualEventOnTwoLists(list, [head], [])),
      n, ([head]^^
      getEqualEventOnTwoLists(list, [head], [])::auxList
   )
else selectRepeated(list, n, auxList)
|selectRepeated(nil, n, auxList) = auxList;

(*Returns the positions of repeated elements. These positions indicate the second,
the third, etc, repeated elements. The initial position is 0.*)
fun getPositionOfRepeatedElements(head::list, pos, auxList) =
if contains list [head]
then getPositionOfRepeatedElements(list, pos+1, pos::auxList)
else getPositionOfRepeatedElements(list, pos+1, auxList)
|getPositionOfRepeatedElements(nil, pos, auxList) = rev auxList;

(*Removes a list of events composed by events that happened at the same time*)
fun removeEventListComposedByEventsWithRepeteadTime(head::listOfEventList, auxList) : LISTofEL=
if getPositionOfRepeatedElements(getEventTimeListFromEventList(head, []), 0, [])<>[]
then removeEventListComposedByEventsWithRepeteadTime(listOfEventList, auxList)
else removeEventListComposedByEventsWithRepeteadTime(listOfEventList, head::auxList)
|removeEventListComposedByEventsWithRepeteadTime(nil, auxList) = rev auxList;

(*Removes a list of events composed by equal events. In practical, equal events do not exist,
because, when all attributes are the same, at least the event time is different in two events.
Therefore,  we consider equal events, the events  that have the same attributes, except by the event time.*)
fun removeEventListComposedByEqualEvents(head::listOfEventList, auxList) : LISTofEL=
if selectRepeated(orderEventListByEventTime(head), 2, [])<>[]
then removeEventListComposedByEqualEvents(listOfEventList, auxList)
else removeEventListComposedByEqualEvents(listOfEventList, head::auxList)
|removeEventListComposedByEqualEvents(nil, auxList) = rev auxList;

(**********Functions - Event - EventTemplate**********)

fun getPrimitiveTemplateValue(primitiveTemplate(value)) = value
| getPrimitiveTemplateValue _ =
{eventTime=0, eventUser="", eventTransaction="", dataId=0, parameters=[]};

fun getCompositeTemplateValue (compositeTemplate(value)) = value
| getCompositeTemplateValue _ =
{eventInitialTime=0, eventFinalTime=0, priorityLevel=0, eventType="", parameters=[],
eventList=[]};
```

```
fun convertPrimitiveTemplateToPrimitive(primTemplate : PRIMITIVE_TEMPLATE) :
PRIMITIVE=
{eventTime = #eventTime primTemplate,
eventUser = #eventUser primTemplate,
eventTransaction = #eventTransaction primTemplate,
dataId = #dataId primTemplate,
parameters = #parameters primTemplate};

fun convertPrimitiveTemplateListToPrimitiveList
(head::primitiveTemplateList : PRIMITIVE_TEMPLATE_LIST, auxList : PRIMITIVE_LIST) =
convertPrimitiveTemplateListToPrimitiveList(primitiveTemplateList,
convertPrimitiveTemplateToPrimitive(head)::auxList)
|convertPrimitiveTemplateListToPrimitiveList(nil, auxList)=rev auxList;

fun convertCompositeTemplateToComposite(compTemplate : COMPOSITE_TEMPLATE1) :
COMPOSITE1=
{eventTime = #eventInitialTime compTemplate,
eventType = #eventType compTemplate,
parameters= #parameters compTemplate,
eventList = (convertPrimitiveTemplateListToPrimitiveList((#eventList compTemplate), []))};

fun convertEventTemplateToEvent(eventTemplate : EVENT_TEMPLATE) : EVENT =
if EVENT_TEMPLATE.of_compositeTemplate(eventTemplate)=true
then composite(convertCompositeTemplateToComposite(getCompositeTemplateValue(eventTemplate)))
else primitive(convertPrimitiveTemplateToPrimitive(getPrimitiveTemplateValue(eventTemplate)));

fun convertEventTemplateListToEventList(head::eventTemplateList :
EVENT_TEMPLATE_LIST, auxList : EVENT_LIST) =
convertEventTemplateListToEventList(eventTemplateList, (convertEventTemplateToEvent(head))::auxList)
|convertEventTemplateListToEventList(nil, auxList) = rev auxList;

fun convETLtoEL(eventTemplateList :  EVENT_TEMPLATE_LIST) : EVENT_LIST=
convertEventTemplateListToEventList(eventTemplateList, []);

fun putETInETLByPriority
(head::etl : EVENT_TEMPLATE_LIST, et : EVENT_TEMPLATE, auxList) : EVENT_TEMPLATE_LIST =
if EVENT_TEMPLATE.of_compositeTemplate(et)=true
then
   if (#priorityLevel (getCompositeTemplateValue(et)))=0
   then (head::etl)^^[et]
   else
      if (#priorityLevel (getCompositeTemplateValue(et)))<(#priorityLevel (getCompositeTemplateValue(head)))
      then auxList^^[et]^^(head::etl)
      else putETInETLByPriority(etl, et, auxList^^[head])
else (head::etl)
|putETInETLByPriority(nil, et, auxList)=auxList^^[et];

fun getEventInitialTimeListFromEventTemplateList(head::eventTemplateList, auxList) :
INT_LIST =
if (EVENT_TEMPLATE.of_primitiveTemplate(head)=true)
then
   getEventInitialTimeListFromEventTemplateList(eventTemplateList,
                        (#eventTime (getPrimitiveTemplateValue(head)))::auxList)
else
   if (EVENT_TEMPLATE.of_compositeTemplate(head)=true)
```

```
   then
      getEventInitialTimeListFromEventTemplateList(eventTemplateList,
                              (#eventInitialTime (getCompositeTemplateValue(head)))::auxList)
   else
      []
|getEventInitialTimeListFromEventTemplateList(nil, auxList)=auxList;

fun getEventFinalTimeListFromEventTemplateList(head::eventTemplateList, auxList) :
INT_LIST =
if (EVENT_TEMPLATE.of_primitiveTemplate(head)=true)
then
   getEventFinalTimeListFromEventTemplateList(eventTemplateList,
                           (#eventTime (getPrimitiveTemplateValue(head)))::auxList)
else
   if (EVENT_TEMPLATE.of_compositeTemplate(head)=true)
   then
      getEventFinalTimeListFromEventTemplateList(eventTemplateList,
                              (#eventFinalTime (getCompositeTemplateValue(head)))::auxList)
   else []
|getEventFinalTimeListFromEventTemplateList(nil, auxList)=auxList;

(**********Functions - Event - CompositeEvents**********)

(*Creates a composite event based on eventList, eventType and parameters*)
fun createCompositeEvent(eventList, eventType, parameters)=
[{eventTime = getLastEventTime(eventList),
eventType = eventType,
parameters = parameters,
eventList = eventList}];

(*Creates a composite event list*)
fun createCompositeEventList(head::listOfEventList : LISTofEL, eventType, parameters, auxList) :
EVENT_LIST=
createCompositeEventList(listOfEventList, eventType, parameters, auxList^^
[composite{eventTime = getLastEventTime(head),
eventType = eventType,
parameters = parameters,
eventList = eventListToPrimitiveList(head, [])}])
|createCompositeEventList(nil, eventType, parameters, auxList)=auxList;

(*Returns the AND of two events*)
fun AND(primitiveEventList: EVENT_LIST, compositeEventList : EVENT_LIST, param : EVENT_LIST) =
let
val a = getEventTypeFromFirstCompositeEvent(param)
val b = getPrimitiveEventListFromFirstCompositeEvent(param)
val c = getEventParamFromFirstCompositeEvent(param)
val vAND = createCompositeEventList(
                combination(
                   convertEventListToListOfEventList(
                      getEqualEventOnTwoLists(primitiveEventList, b, []),
                         []), length b), a, c, [])
in
if (vAND<>[]) andalso (contains compositeEventList vAND= false)
then compositeEventList^^(listsub vAND (intersect compositeEventList vAND))
else compositeEventList
end;
```

```
(*Returns the OR of two events*)
fun OR(primitiveEventList: EVENT_LIST, compositeEventList : EVENT_LIST, param : EVENT_LIST) =
let
val a = getEventTypeFromFirstCompositeEvent(param)
val b = getPrimitiveEventListFromFirstCompositeEvent(param)
val c = getEventParamFromFirstCompositeEvent(param)
val vOR = createCompositeEventList(
                convertEventListToListOfEventList(
                   getEqualEventOnTwoLists(primitiveEventList, b, []),
                      []), a, c, [])
in
if (vOR<>[]) andalso (contains compositeEventList vOR = false)
then compositeEventList^^(listsub vOR (intersect compositeEventList vOR))
else compositeEventList
end;

(*Returns a list of events with diferent event time*)
fun SEQ(primitiveEventList: EVENT_LIST, compositeEventList : EVENT_LIST, param : EVENT_LIST) =
let
val a = getEventTypeFromFirstCompositeEvent(param)
val b = getPrimitiveEventListFromFirstCompositeEvent(param)
val c = combination(
            convertEventListToListOfEventList(
               getEqualEventOnTwoLists(orderEventListByEventTime(primitiveEventList),
                  orderEventListByEventTime(b), []), []), length b)
val d = getEventParamFromFirstCompositeEvent(param)
val vSEQ = createCompositeEventList(removeEventListComposedByEventsWithRepeteadTime(c, []),  a, d, [])
in
if (c<>[]) andalso (contains compositeEventList vSEQ = false)
then compositeEventList^^(listsub vSEQ (intersect compositeEventList vSEQ))
else compositeEventList
end;

(*Returns a list of events with equal event time*)
fun SIM(primitiveEventList: EVENT_LIST, compositeEventList : EVENT_LIST, param : EVENT_LIST) =
let
val a = getEventTypeFromFirstCompositeEvent(param)
val b = getPrimitiveEventListFromFirstCompositeEvent(param)
val c = combination(
            convertEventListToListOfEventList(
               getEqualEventOnTwoLists(orderEventListByEventTime(primitiveEventList),
                  orderEventListByEventTime(b), []),  length b)
val d = getEventParamFromFirstCompositeEvent(param)
val vSIM = createCompositeEventList(listsub c (removeEventListComposedByEventsWithRepeteadTime(c, [])), a,
d, [])
in
if (c<>[]) andalso (contains compositeEventList vSIM = false)
then compositeEventList^^(listsub vSIM (intersect compositeEventList vSIM))
else compositeEventList
end;
```

```
(*Returns a list of different events. In practical, equal events do not exist, because,
when all attributes are the same, at least the event time is different in two events.
Therefore, we consider equal events,  the events  that have the same attributes,
except by the event time.*)
fun ANY(primitiveEventList: EVENT_LIST, compositeEventList : EVENT_LIST, param : EVENT_LIST) =
let
val a = getEventTypeFromFirstCompositeEvent(param)
val b = getPrimitiveEventListFromFirstCompositeEvent(param)
val c = getEventParamFromFirstCompositeEvent(param)
val d =  combination(
            convertEventListToListOfEventList(
               getEqualEventOnTwoLists(primitiveEventList, b, []),
                  []), length b)
val vANY = createCompositeEventList(removeEventListComposedByEqualEvents(d, []), a, c, [])
in
if (vANY<>[]) andalso (contains compositeEventList vANY = false)
then compositeEventList^^(listsub vANY (intersect compositeEventList vANY))
else compositeEventList
end;

(*Returns equal events, except by the event time. In practical, equal events do not exist, because,
when all attributes are the same, at least the event time is different in two events.
Therefore,  we consider equal events, the events  that have the same attributes, except by the event time.*)
fun HIS(primitiveEventList: EVENT_LIST, compositeEventList : EVENT_LIST, param : EVENT_LIST) =
let
val a = getEventTypeFromFirstCompositeEvent(param)
val b = getPrimitiveEventListFromFirstCompositeEvent(param)
val c = getEventParamFromFirstCompositeEvent(param)
val vHIS = createCompositeEventList(
               selectRepeated(
                  orderEventListByEventTime(
                     getEqualEventOnTwoLists(primitiveEventList, b, [])),
                        length b, []), a, c, [])
in
if (vHIS<>[]) andalso (contains compositeEventList vHIS = false)
then compositeEventList^^(listsub vHIS (intersect compositeEventList vHIS))
else compositeEventList
end;

(*Returns a composite event based on param*)
fun selectComposite(param : EVENT_LIST, primitiveEventList: EVENT_LIST, compositeEventList :
EVENT_LIST) =
let
val a = getEventTypeFromFirstCompositeEvent(param)
in
case a of
“AND” => (AND(primitiveEventList, compositeEventList, param))
| “OR” => (OR(primitiveEventList, compositeEventList, param))
| “SEQUENCE” => (SEQ(primitiveEventList, compositeEventList, param))
| “SIMULTANEOUS” => (SIM(primitiveEventList, compositeEventList, param))
| “ANY” => (ANY(primitiveEventList, compositeEventList, param))
| “HISTORY” => (HIS(primitiveEventList, compositeEventList, param))
| “And” => (AND(primitiveEventList, compositeEventList, param))
| “Or” => (OR(primitiveEventList, compositeEventList, param))
| “Sequence” => (SEQ(primitiveEventList, compositeEventList, param))
| “Simultaneous” => (SIM(primitiveEventList, compositeEventList, param))
```

```
| “Any” => (ANY(primitiveEventList, compositeEventList, param))
| “History” => (HIS(primitiveEventList, compositeEventList, param))
| _ => compositeEventList
end;

(*Removes events from an event list that have the time lower than final time. If final time is zero,
none element is removed.*)
fun removeEventFromEventListLessThanFinalTime(head::eventList : EVENT_LIST, finalTime, auxList) =
if EVENT.of_primitive(head)=true
then
   if (#eventTime (getPrimitiveValue(head)))<=finalTime orelse finalTime=0
   then removeEventFromEventListLessThanFinalTime(eventList, finalTime, head::auxList)
   else removeEventFromEventListLessThanFinalTime(eventList, finalTime, auxList)
else
   if EVENT.of_composite(head)=true
   then
      if (#eventTime (getCompositeValue(head)))<=finalTime orelse finalTime=0
      then removeEventFromEventListLessThanFinalTime(eventList, finalTime, head::auxList)
      else removeEventFromEventListLessThanFinalTime(eventList, finalTime, auxList)
   else
      removeEventFromEventListLessThanFinalTime(eventList, finalTime, head::auxList)
|removeEventFromEventListLessThanFinalTime(nil, finalTime, auxList)=rev auxList;

(*Returns the combination of events that exist in event list, according to event template list,
if the combination does not exist in a composite event list*)
fun fireComposite(head::param : EVENT_TEMPLATE_LIST, primitiveEventList: EVENT_LIST,
compositeEventList : EVENT_LIST) =
if EVENT_TEMPLATE.of_compositeTemplate(head)=true
then
   if compositeEventList<>selectComposite(convETLtoEL([head]),
      removeEventFromEventListLessThanFinalTime(primitiveEventList, (#eventFinalTime
(getCompositeTemplateValue(head))), []),
      compositeEventList)
   then fireComposite(param,
            primitiveEventList,
            selectComposite(
               convETLtoEL([head]),
               removeEventFromEventListLessThanFinalTime(
                  primitiveEventList, (#eventFinalTime (getCompositeTemplateValue(head))), []),
               compositeEventList))
   else fireComposite(param, primitiveEventList, compositeEventList)
else fireComposite(param, primitiveEventList, compositeEventList)
|fireComposite(nil, primitiveEventList, compositeEventList) = compositeEventList;

(**********Functions - Event - AssembleCompositeEvents**********)

fun fireAND(ep1, ep2, lec) = not (contains lec
[composite{eventTime = max(#eventTime ep1, #eventTime ep2),
eventType=“AND”, parameters=[], eventList = [ep1, ep2]}]);

fun fireOR(ep, lec) = not (contains lec
[composite{eventTime= #eventTime ep,
eventType=“OR”, parameters=[], eventList=[ep]}]);
```

```
fun fireSequence(ep1, ep2, lec) = not (contains lec
[composite{eventTime = max(#eventTime ep1, #eventTime ep2),
eventType="Sequence", parameters=[], eventList = [ep1, ep2]}]);

fun fireSimultaneous(ep1, ep2, lec) = not (contains lec
[composite{eventTime = max(#eventTime ep1, #eventTime ep2),
eventType="Simultaneous", parameters=[], eventList = [ep1, ep2]}]);

fun fireHistory(lp, size, eventType, param, lec) =
let
val a = createCompositeEventList(
            combination(
               convertEventListToListOfEventList(lp, []),
                  size),
                     eventType, param, [])
in
if (a<>[]) andalso (contains lec a=false)
then (listsub a (intersect lec a))
else []
end;

fun fireAny(lp, size, eventType, param, lec) =
let
val a = createCompositeEventList(
            removeEventListComposedByEqualEvents(
               combination(
                  convertEventListToListOfEventList(lp, []),
                     size), []),
                     eventType, param, [])
in
if (a<>[]) andalso (contains lec a=false)
then (listsub a (intersect lec a))
else []
end;

(**********Functions - Event - EventCleaning**********)

fun getTheBiggestEventTimeFromEventList(eventList) : INT=
hd (rev (
orderIntList(getEventTimeListFromEventList(eventList, []))))

fun getTheSmallestEventTimeFromEventList(eventList) : INT=
hd (
orderIntList(getEventTimeListFromEventList(eventList, [])))

fun getTheSmallestEventInitialTimeFromEventTemplateList(eventTemplateList) : INT=
hd (
orderIntList(getEventInitialTimeListFromEventTemplateList(eventTemplateList, [])))

fun getTheSmallestEventFinalTimeFromEventTemplateList(eventTemplateList) : INT=
hd (
orderIntList(getEventFinalTimeListFromEventTemplateList(eventTemplateList, [])))

fun removeEventTimeSmallerThanTheSmallestEventInitialTime
(head::eventList, eventTemplateList, auxList) : EVENT_LIST=
if eventTemplateList<>[]
```

```
then
   if (EVENT.of_primitive(head)=true)
   then
      if (#eventTime (getPrimitiveValue(head)))<
         getTheSmallestEventInitialTimeFromEventTemplateList(eventTemplateList)
      then
         removeEventTimeSmallerThanTheSmallestEventInitialTime(eventList, eventTemplateList, auxList)
      else
         removeEventTimeSmallerThanTheSmallestEventInitialTime(eventList, eventTemplateList, head::auxList)
   else
      if (EVENT.of_composite(head)=true)
      then
         if (#eventTime (getCompositeValue(head)))<
            getTheSmallestEventInitialTimeFromEventTemplateList(eventTemplateList)
         then
            removeEventTimeSmallerThanTheSmallestEventInitialTime(eventList, eventTemplateList, auxList)
         else
            removeEventTimeSmallerThanTheSmallestEventInitialTime(eventList, eventTemplateList,
head::auxList)
      else removeEventTimeSmallerThanTheSmallestEventInitialTime(eventList, eventTemplateList,
head::auxList)
else []
|removeEventTimeSmallerThanTheSmallestEventInitialTime(nil, eventTemplateList, auxList)=rev
auxList;

fun removeEventFinalTimeSmallerThanTheBiggestEventTime
(head::eventTemplateList, eventList, auxList) : EVENT_TEMPLATE_LIST =
if eventList<>[]
then
   if (EVENT_TEMPLATE.of_primitiveTemplate(head)=true)
   then
   if (#eventTime (getPrimitiveTemplateValue(head)))<getTheBiggestEventTimeFromEventList(eventList)
   andalso (#eventTime (getPrimitiveTemplateValue(head)))<>0
      then
         removeEventFinalTimeSmallerThanTheBiggestEventTime(eventTemplateList, eventList, auxList)
      else
         removeEventFinalTimeSmallerThanTheBiggestEventTime(eventTemplateList, eventList, head::auxList)
   else
      if (EVENT_TEMPLATE.of_compositeTemplate(head)=true)
      then
         if (#eventFinalTime
(getCompositeTemplateValue(head)))<getTheBiggestEventTimeFromEventList(eventList)
         andalso (#eventFinalTime (getCompositeTemplateValue(head)))<>0
         then
            removeEventFinalTimeSmallerThanTheBiggestEventTime(eventTemplateList, eventList, auxList)
         else
            removeEventFinalTimeSmallerThanTheBiggestEventTime(eventTemplateList, eventList, head::auxList)
      else removeEventFinalTimeSmallerThanTheBiggestEventTime(eventTemplateList, eventList, head::auxList)
else head::eventTemplateList
|removeEventFinalTimeSmallerThanTheBiggestEventTime(nil, eventList, auxList)=rev auxList;

(**********Functions - Socket - StringToData**********)

fun getTuple (stringTuple) : DATA_TUPLE=
{idData=strToInt(getFirstParamBetweenPrefixAndSufix(0, stringTuple, "<at1>",
"</at1>")),
```

```
metadata=getFirstParamBetweenPrefixAndSufix(0,stringTuple,
"<at2>","</at2>"),
value=getFirstParamBetweenPrefixAndSufix(0,stringTuple,
"<at3>","</at3>")};


fun strToList(str, pos, dataList)=
if(getFirstParamBetweenPrefixAndSufix(pos, str, "<tu>",
"</tu>")="")
then dataList
else strToList(str, findAfterFirstParam(pos,str,"<tu>"),
                getTuple(getFirstParamBetweenPrefixAndSufix(pos, str, "<tu>",
"</tu>"))::dataList)

fun strToData(str) =
rev(strToList(str, 0, []));

(**********Functions - Socket - DataToString**********)

fun dataTupleToStringTuple(dataTuple : DATA_TUPLE)  : STRING=
"<at1>"^intToStr(#idData dataTuple)^"</at1><at2>"^
(#metadata dataTuple)^"</at2><at3>"^(#value
dataTuple)^"</at3>";

fun createStringSequence(head::list, aux) : STRING=
createStringSequence(list, aux^"<tu>"^dataTupleToStringTuple(head)^"</tu>")
|createStringSequence(nil, aux)=aux;

fun dataToString(data : DATA) : STRING =
createStringSequence(data, "");

(**********Functions - Socket - StringToEvent**********)

(*Checks if a metadata and value exist in a string. It returns a pair: boolean, integer.
It will return true as the first element of the pair, if the metadata and value exist.
It will return false if some of them is not found. If the metadata exists,
it will return as the second element of the pair, the first  position after the value tuple.
If not, it will return 0.*)
fun testFirstMetadataValue(posInitial, str, metadata, value) =
let
val n = findFirstParam(posInitial, str, "<at2>"^metadata^"</at2>")
val l = getFirstParamBetweenPrefixAndSufix(n, str, "<at2>", "</at2>")
val m = findAfterFirstParam(n, str, "</at3></tu>")
val k = getFirstParamBetweenPrefixAndSufix(n, str, "<at3>", "</at3>")
in
if metadata=l andalso value = k
then (true, m)
else
   if (metadata=l) andalso (value<>k)
   then (false, m)
   else (false, 0)
end;

(*Converts a string that contains a list of primitive event to a primitive event list*)
fun strToPrimitiveList(pos, str, auxList : PRIMITIVE_LIST) =
let
```

```
val (b, m) = testFirstMetadataValue(pos, str, “parameters”, “true”)
val i = findFirstParam(m, str, “eventTime”)
val n = if (m<>0) andalso (i<>String.size(str))
                then findLastParam(i, str, “<tu>”)
                else if (m<>0) andalso (i=String.size(str))
                      then String.size(str)
                      else  m
in
if findAfterFirstParam(pos, str, “eventTime”)<>0
then
strToPrimitiveList(n,
str,
auxList^^
[{
eventTime= strToInt(getFirstParamBetweenPrefixAndSufix(
                    findFirstParam(pos, str, “eventTime”),
                    str, “<at3>”, “</at3>”)),
eventUser= getFirstParamBetweenPrefixAndSufix(
                    findFirstParam(pos, str, “eventUser”),
                    str, “<at3>”, “</at3>”),
eventTransaction= getFirstParamBetweenPrefixAndSufix(
                    findFirstParam(pos, str, “eventTransaction”),
                    str, “<at3>”, “</at3>”),
dataId= strToInt(getFirstParamBetweenPrefixAndSufix(
                    findFirstParam(pos, str, “dataId”),
                    str, “<at3>”, “</at3>”)),
parameters= (if n>m
                  then strToData(substring(str, m, n-m))
                  else [])
}])
else auxList
end;

(*Converts a string that contains an event of COMPOSITE1 type to a event list*)
fun strToEventList(str) =
let
val a = findFirstParam(0, str, “<at2>eventClass</at2>”)
val b = findFirstParam(a, str, “<at2>eventTime</at2>”)
val c = findFirstParam(b, str, “<at2>eventType</at2>”)
val (d, m) = testFirstMetadataValue(0, str, “parameters”, “true”)
val i = findFirstParam(m, str, “<at2>eventList</at2>”)
val n = findLastParam(i, str, “<tu>”)
in
if a <> 0
then
   if getFirstParamBetweenPrefixAndSufix(a, str, “<at3>”,
“</at3>”)=“composite”
   then
      [composite{
      eventTime = strToInt(getFirstParamBetweenPrefixAndSufix(b, str, “<at3>”,
“</at3>”)),
      eventType = getFirstParamBetweenPrefixAndSufix(c, str, “<at3>”,
“</at3>”),
      parameters = (if n>m
                        then strToData(substring(str, m, n-m))
                        else []),
```

```
      eventList = (if m>0
                   then strToPrimitiveList(m, str, [])
                   else [])
      }]
   else
      if getFirstParamBetweenPrefixAndSufix(a, str, "<at3>",
"</at3>")="primitive"
      then primitiveListToEventList(strToPrimitiveList(b, str, []),  [])
      else []
else []
end;

(*Converts a string to a primitive event template list*)
fun strToPrimitiveTemplateList(pos, str, auxList : PRIMITIVE_TEMPLATE_LIST) =
let
val (b, m) = testFirstMetadataValue(pos, str, "parameters", "true")
val i = findFirstParam(m, str, "eventTime")
val n = if (m<>0) andalso (i<>String.size(str))
              then findLastParam(i, str, "<tu>")
              else if (m<>0) andalso (i=String.size(str))
                   then String.size(str)
                   else  m
in
if findAfterFirstParam(pos, str, "eventTime")<>0
then
strToPrimitiveTemplateList(n,
str,
auxList^^
[{
eventTime= strToInt(getFirstParamBetweenPrefixAndSufix(
                   findFirstParam(pos, str, "eventTime"),
                   str, "<at3>", "</at3>")),
eventUser= getFirstParamBetweenPrefixAndSufix(
                   findFirstParam(pos, str, "eventUser"),
                   str, "<at3>", "</at3>"),
eventTransaction= getFirstParamBetweenPrefixAndSufix(
                   findFirstParam(pos, str, "eventTransaction"),
                   str, "<at3>", "</at3>"),
dataId= strToInt(getFirstParamBetweenPrefixAndSufix(
                   findFirstParam(pos, str, "dataId"),
                   str, "<at3>", "</at3>")),
parameters= (if n>m
                  then strToData(substring(str, m, n-m))
                  else [])
}])
else auxList
end;

(*Converts a string to a event template list*)
fun strToEventTemplateList(str) =
let
val a = findFirstParam(0, str, "<at2>eventClass</at2>")
val b = findFirstParam(a, str, "<at2>eventInitialTime</at2>")
val c = findFirstParam(a, str, "<at2>eventFinalTime</at2>")
val d = findFirstParam(a, str, "<at2>priorityLevel</at2>")
val e = findFirstParam(b, str, "<at2>eventType</at2>")
```

```
val (f, m) = testFirstMetadataValue(0, str, “parameters”, “true”)
val i = findFirstParam(m, str, “<at2>eventList</at2>”)
val n = findLastParam(i, str, “<tu>”)
in
if a <> 0
then
   if getFirstParamBetweenPrefixAndSufix(a, str, “<at3>”,
“</at3>”)=“composite”
   then
      [compositeTemplate{
      eventInitialTime = strToInt(getFirstParamBetweenPrefixAndSufix(b, str, “<at3>”,
“</at3>”)),
      eventFinalTime = strToInt(getFirstParamBetweenPrefixAndSufix(c, str, “<at3>”,
“</at3>”)),
      priorityLevel = strToInt(getFirstParamBetweenPrefixAndSufix(d, str, “<at3>”,
“</at3>”)),
      eventType = getFirstParamBetweenPrefixAndSufix(e, str, “<at3>”,
“</at3>”),
      parameters = (if n>m
                    then strToData(substring(str, m, n-m))
                    else []),
      eventList = (if m>0
                   then strToPrimitiveTemplateList(m, str, [])
                   else [])
      }]
   else
      if getFirstParamBetweenPrefixAndSufix(a, str, “<at3>”,
“</at3>”)=“primitiveTemplate”
      then primitiveTemplateListToEventTemplateList(strToPrimitiveTemplateList(b, str, []), [])
      else []
else []
end;

(**********Functions - Socket - EventToString**********)

fun primitiveListToStr(head::primitiveList : PRIMITIVE_LIST, str, id) =
primitiveListToStr(primitiveList, str^
“<tu><at1>”^intToStr(id)^”</at1><at2>eventTime</at2><at3>
“^(intToStr(#eventTime head))^”</at3></tu>”^
“<tu><at1>”^intToStr(id)^”</at1><at2>eventUser</at2><at3>
“^(#eventUser head)^”</at3></tu>”^
“<tu><at1>”^intToStr(id)^”</at1><at2>eventTransaction</at2><
at3>”^(#eventTransaction head)^”</at3></tu>”^
“<tu><at1>”^intToStr(id)^”</at1><at2>dataId</at2><at3>
“^(intToStr(#dataId head))^”</at3></tu>”^
“<tu><at1>”^intToStr(id)^”</at1><at2>parameters</at2><at3>
“^
(if (#parameters head)=[]
then (““^”</at3></tu>” )
else (“true</at3></tu>”^dataToString(#parameters head))),
id)
|primitiveListToStr(nil, str, id) = str;

fun eventListToStr(head::event : EVENT_LIST, str, id) =
if (EVENT.of_composite(head)=true)
then
```

```
eventListToStr(event, str^
“<tu><at1>”^intToStr(id)^”</at1><at2>event</at2><at3>true
</at3></tu>”^
“<tu><at1>”^intToStr(id)^”</at1><at2>registerEvent</at2><at3
>composite</at3></tu>”^
“<tu><at1>”^intToStr(id)^”</at1><at2>eventTime</at2><at3>
“^(intToStr(#eventTime (getCompositeValue(head))))^”</at3></tu>”^
“<tu><at1>”^intToStr(id)^”</at1><at2>eventType</at2><at3>
“^(#eventType (getCompositeValue(head)))^”</at3></tu>”^
“<tu><at1>”^intToStr(id)^”</at1><at2>eventList</at2><at3>
primitive</at3></tu>”^
(primitiveListToStr((#eventList (getCompositeValue(head))), ““, id)), id+1)
else
   if (EVENT.of_primitive(head)=true)
   then
   eventListToStr(event, str^

“<tu><at1>”^intToStr(id)^”</at1><at2>event</at2><at3>true
</at3></tu>”^

“<tu><at1>”^intToStr(id)^”</at1><at2>registerEvent</at2><at3
>primitive</at3></tu>”^
   (primitiveListToStr((#eventList (getCompositeValue(head))), ““, id)), id+1)
else ““
|eventListToStr(nil, str, id) = str;

(**********Functions - Socket - Communication**********)

fun openConnection(conn, hostName, portNumber) : UNIT =
(ConnManagementLayer.openConnection(conn, hostName, portNumber))
handle DupConnNameExn(conn) =>e

fun acceptConnection(conn, portNumber) : UNIT =
(ConnManagementLayer.acceptConnection(conn, portNumber))
handle DupConnNameExn(conn) =>e

fun sendToApplication(conn, s) =
ConnManagementLayer.send(conn, s, stringEncode)
handle ElementMissingExn(conn) => e;

fun receiveFromApplication(conn) : STRING =
(ConnManagementLayer.receive(conn, stringDecode))
handle ElementMissingExn(conn) => “Comms/CPN Error: Unknown Connection”

fun canReceive (conn) : BOOL=
(ConnManagementLayer.canreceive(conn))
handle ElementMissingExn(conn) =>false

fun closeConnection(conn) : UNIT =
(ConnManagementLayer.closeConnection(conn))
handle ElementMissingExn(conn) =>e
```

```
(**********Functions - Pre-processing**********)

(*Calls a stemming function using sockets*)
fun stemming (connectionName, str, u, step) =
if step=4
then stemming(connectionName, str,closeConnection(connectionName),step-1)
else
if step=3
then stemming(connectionName, str,
      openConnection(connectionName,stemHostAddress,stemPortNumber),step-1)
else
if step=2
then stemming(connectionName, str,sendToApplication(connectionName, str),step-1)
else
   if step=1 andalso canReceive(connectionName)
   then stemming(connectionName, receiveFromApplication(connectionName), u, step-1)
   else
      if step=1
      then stemming(connectionName, str, u, step)
      else str;

(*Puts parameters related to stemming, stopwords and language in a string*)
fun stemmingHead (str, param) =
"<"^paramST^">"^
getFirstDataValue(param, paramST)
^"</"^paramST^"> "^
"<"^paramSW^">"^
getFirstDataValue(param, paramSW)
^"</"^paramSW^"> "^
 "<"^paramLG^">"^
getFirstDataValue(param, paramLG)
^"</"^paramLG^"> "^
str;

(*Extracts all ocurrences of param between prefix and sufix, starting from p1 to sufix and p2 to
prefix*)
fun extractAllParam(str, prefix, sufix, p1, p2, strAux) =
let
val i=findAfterFirstParam(p1, str, prefix)
val j=findAfterFirstParam(p2, str, sufix)
in
if (j>i)
then
   extractAllParam(str, prefix, sufix, i, j,
                   strAux^
                   prefix^
                   substring(str, i, (j - String.size(sufix) - i))^
                   sufix)
else strAux
end;

(*Starting from p1 to prefix and p2 to sufix, replaces sequencially all ocurrences of param between
prefix and sufix of str1
by all the ocurrences of param between prefix and sufix of str2. Returm str1 combined with the ocurrences in str2*)
fun replaceAllParam(str1, str2, prefix, sufix, s1, f1, s2, f2, strAux) =
let
```

```
val i1=findAfterFirstParam(s1, str1, prefix)
val j1=findAfterFirstParam(f1, str1, sufix)
val i2=findAfterFirstParam(s2, str2, prefix)
val j2=findAfterFirstParam(f2, str2, sufix)
in
if (j1>i1) andalso (j2>i2)
then
   replaceAllParam(str1, str2, prefix, sufix, j1-String.size(sufix), j1, i2, j2,
                   strAux^
                   substring(str1, s1, (i1 - s1))^
                   substring(str2, i2, (j2 - String.size(sufix) - i2)) )
else strAux^substring(str1, s1, String.size(str1)-s1)
end;

(*When the first param between prefix2 and sufix2 in str2 is the same that any param between
prefix2 and sufix2 in str1,
it changes the prefix1 and sufix1 in str1 by markPrefix and markSufix. The result is stored in strAux*)
fun changePrefixSufix(str1, str2, prefix1, sufix1, prefix2, sufix2,
markPrefix, markSufix, s1, s2, f1, f2, strAux) =
let
val i=findAfterFirstParam(s1, str1, prefix1)
val j=findAfterFirstParam(s2, str1, sufix1)
val i1=findAfterFirstParam(s1, str1, prefix2)
val j1=findAfterFirstParam(s2, str1, sufix2)
val i2=findAfterFirstParam(f1, str2, prefix2)
val j2=findAfterFirstParam(f2, str2, sufix2)
in
if (j>i) andalso (j1>i1) andalso (j2>i2)
then
   if (substring(str1, i1, j1- i1)) = (substring(str2, i2, j2- i2))
   then
      changePrefixSufix(str1, str2, prefix1, sufix1, prefix2, sufix2,
                        markPrefix, markSufix,
                        j, j, 0, 0,
                        strAux^
                        substring(str1, s1, j1-s1)^
                        markPrefix^
                        substring(str1, i, (j - String.size(sufix1) - i))^
                        markSufix)
   else
      changePrefixSufix(str1, str2, prefix1, sufix1, prefix2, sufix2,
                        markPrefix, markSufix,
                        s1, s2, i2, j2,
                        strAux)
else
   if (i1<>0) andalso (i2=0)
   then
      changePrefixSufix(str1, str2, prefix1, sufix1, prefix2, sufix2,
                        markPrefix, markSufix,
                        j, j, 0, 0,
                        strAux^
                        substring(str1, s1, j-s1) )
   else strAux^substring(str1, s1, String.size(str1)-s1)
end;
```

```
(*Detects if a pre-processing is necessary, returns true, or not, returns false*)
fun detectPreProcessing(param)=
if (getFirstDataValue(param, paramST)= “true”) orelse
  (getFirstDataValue(param, paramSW)= “true”)
then true
else false;

(*Pre-processes the param between prefix and sufix using stemming and stopwords*)
fun preProcessing(s, param, prefix, sufix) =
let
val s1 = extractAllParam(s, prefix, sufix, 0, 0, ““)
in
if detectPreProcessing(param)=true
then
   replaceAllParam(s,
                        stemming(stemConnectionName,
                                 stemmingHead(s1,param),
                                 e, stemSteps),
                        prefix, sufix, 0, 0, 0, 0, ““)
else s
end

(**********Functions - Similarity**********)

fun jaccard (s1, s2) =
let
val a = (100*length(remdupl(intersect
(splitParam(s1, “ “, []))
(splitParam(s2, “ “, [])) )))
val b = length(remdupl(union
(splitParam(s1, “ “, []))
(splitParam(s2, “ “, [])) ))
in
if (b>0)
then (a div b)
else 1
end;

fun keyword (s1, s2) =
length(remdupl(intersect
(splitParam(s1, “ “, []))
(splitParam(s2, “ “, [])) ));

fun checkSimilarity(s1, s2, param) =
if (getFirstDataValue(param, paramSS)= “Jaccard”)
then
     jaccard(s1, s2)
     >=
     strToInt(getFirstDataValue(param, paramSC))
else
   if (getFirstDataValue(param, paramSS)= “Keyword”)
   then
        keyword(s1, s2)
        >=
        strToInt(getFirstDataValue(param, paramSC))
   else false;
```

```
fun getSimilarity(s1, s2, param) =
if (getFirstDataValue(param, paramSS)= “Jaccard”)
then
      jaccard(s1, s2)
else
   if (getFirstDataValue(param, paramSS)= “Keyword”)
   then
        keyword(s1, s2)
   else 0;


(**********Functions - Patterns - Data Lists - Check**********)

(*Checks if it exists a data that has metadata equals x and value equals y*)
fun detectDataByMetadataAndValue({idData, metadata, value}::listOfRecords, x, y) =
   if metadata=x andalso value=y
   then true
   else detectDataByMetadataAndValue(listOfRecords, x, y)
|detectDataByMetadataAndValue(nil, x, y) = false;

(*Checks if it exists a data that has idData equals x and metadata equals y*)
fun detectDataByIdDataAndMetadata({idData, metadata, value}::listOfRecords, x, y) =
   if idData=x andalso metadata=y
   then true
   else detectDataByIdDataAndMetadata(listOfRecords, x, y)
|detectDataByIdDataAndMetadata(nil, x, y) = false;

(*Checks if it exists a data that has metadata equals x and value is similar to y.
Param correspondes to the similatrity function*)
fun detectDataByMetadataAndSimilarValue({idData, metadata, value}::listOfRecords, x, y,
     param) =
   if (metadata=x) andalso (checkSimilarity(value, y, param))
   then true
   else detectDataByMetadataAndSimilarValue(listOfRecords, x, y,param)
|detectDataByMetadataAndSimilarValue(nil, x, y, param) = false;

(*Checks if two data have the same metadata and similar value on one list.
Using comparedMetadata, a specific metadata can be compared or all of them*)
fun detectSimilarDataOnOneList
({idData, metadata, value}::list1 : DATA, param, auxList) =
if metadata= getFirstDataValue(param, paramCM)
orelse getFirstDataValue(param, paramCM)=“any”
orelse getFirstDataValue(param, paramCM)=““
then
   if detectDataByMetadataAndSimilarValue(list1, metadata, value, param)
   then true
   else detectSimilarDataOnOneList(
      list1, param, auxList)
else detectSimilarDataOnOneList(
      list1, param, auxList)
|detectSimilarDataOnOneList(nil, param, auxList) = false;
```

```
(*Checks if a data, which belongs to list1, exists on list 2 with the same metadata and similar
value. Using comparedMetadata, a specific metadata can be compared or all of them*)
fun detectSimilarDataOnTwoLists
({idData, metadata, value}::list1 : DATA, list1Copy, list2 : DATA, param, auxList) =
if metadata= getFirstDataValue(param, paramCM)
orelse getFirstDataValue(param, paramCM)="any"
orelse getFirstDataValue(param, paramCM)=""
then
   if detectDataByMetadataAndSimilarValue(list2, metadata, value, param)
   then true
   else detectSimilarDataOnTwoLists(list1, list1Copy, list2, param, auxList)
else detectSimilarDataOnTwoLists(list1, list1Copy, list2, param, auxList)
|detectSimilarDataOnTwoLists(nil, list1Copy, list2, param, auxList) = false;

(*Gets the first data value of a data identified by p1*)
fun detectData(data, p1, v1) =
getFirstDataValue(data,p1)=v1;

(**********Functions - Patterns - Data Lists - Execute**********)

(*Merges two lists that has the same size, according to param that is equal to  highest, lowest or
combination,
that is the default. The option combination adds the values.*)
fun mergeDataList(l1::list1:DATA, l2::list2:DATA, param : STRING, auxList : DATA) =
if (length list1 = length list2)
then
   if (param = "highest")
   then
      if ((#idData l1) =(#idData l2))andalso((#metadata l1)=(#metadata l2))
      then
         if checkAndConvertStrToInt (#value l1)>checkAndConvertStrToInt (#value l2)
         then mergeDataList(list1, list2, param, auxList^^[l1])
         else mergeDataList(list1, list2, param, auxList^^[l2])
      else mergeDataList(list1, list2, param, auxList)
   else
      if (param = "lowest")
      then
         if ((#idData l1) =(#idData l2))andalso((#metadata l1)=(#metadata l2))
         then
            if checkAndConvertStrToInt(#value l1)<checkAndConvertStrToInt(#value l2)
            then mergeDataList(list1, list2, param, auxList^^[l1])
            else mergeDataList(list1, list2, param, auxList^^[l2])
         else mergeDataList(list1, list2, param, auxList)
      else
         if ((#idData l1) =(#idData l2))andalso((#metadata l1)=(#metadata l2))
         then
            mergeDataList(list1, list2, param, auxList^^
               [{idData=(#idData l1), metadata=(#metadata l1),
               value=intToStr(checkAndConvertStrToInt(#value l1)+checkAndConvertStrToInt(#value l2))}])
         else mergeDataList(list1, list2, param, auxList)
else auxList
|mergeDataList(null, list2, param, auxList) = auxList
```

```
(*Creates a similarity list from data that has metadata equals to x.
The value is given by the similarity between  the value of the metadata and y.
Param provides to the similarity function*)
fun createSimilarityDataListBasedOnMetadataAndValue({idData, metadata, value}::listOfRecords, x, y,
     param, auxList) =
   if (metadata=x)
   then  createSimilarityDataListBasedOnMetadataAndValue(listOfRecords, x, y,param,
              auxList^^
             [{idData=idData, metadata=paramSL, value=intToStr(getSimilarity(value, y, param))}])
   else createSimilarityDataListBasedOnMetadataAndValue(listOfRecords, x, y,param, auxList)
|createSimilarityDataListBasedOnMetadataAndValue(nil, x, y, param, auxList) = auxList;

(*Gives the highest value of similarity of each element in data,
compared with all elements of param with the metadata equals value of paramCM.
In this case, data does not have paramOrder that is related to similarity level.*)
fun createSimilarityDataListBasedOnParam(data: DATA, param: DATA, paramCopy, auxList : DATA)=
let
val a = (data<>[])andalso(param<>[])
val b = if (a andalso getFirstDataValue(paramCopy, paramCM) = (#metadata (hd param)))
          then createSimilarityDataListBasedOnMetadataAndValue(
             data, #metadata (hd param), #value (hd param), paramCopy, [])
          else []
val c = if auxList<>[] andalso b<>[]
         then mergeDataList(auxList, b, paramMergeDefault, [])
         else
            if b<>[] then b else auxList
in
if a
then createSimilarityDataListBasedOnParam(data, tl param, paramCopy, c)
else
   auxList
end;

(*Creates a list of metadata values, where the metadata is extracted from param. The
metadata is the order metadata.*)
fun createListOfMetadataValueBasedOnParam({idData, metadata, value}::data, param,
auxList:STRING_LIST)=
if metadata = getFirstDataValue(param, paramOrder)
then createListOfMetadataValueBasedOnParam(data, param, auxList^^[value])
else createListOfMetadataValueBasedOnParam(data, param, auxList)
|createListOfMetadataValueBasedOnParam(null, param, auxList) = auxList;

(*Orders data considering the values of a specific metadata, extracted from param, based on
other list of values.
This list of values can not have repeated values*)
fun orderDataByList(list: STRING_LIST, data : DATA, dataCopy: DATA, param, auxList)=
let
val a = if list<>[] andalso data <>[] then getFirstDataValue(param, paramOrder) else "0"
val b = if list<>[] andalso data <>[] then getFirstDataValue(data,a) else "0"
val c = if list<>[] andalso data <>[] then getFirstDataTuple(data,a) else []
val d = if c<>[] then getIdFromDataTuple(c) else 0
val e = if (data <> [] andalso list<> [] andalso d <>0) then getDataById(data,d,[]) else []
in
if data<>[] andalso list<>[] andalso e <> []
then
   if (hd list = b)
```

```
   then orderDataByList(list, putSub(data,e), dataCopy, param, auxList^^e)
   else orderDataByList(list, putSub(data,e),dataCopy,param,auxList)
else
   if (list<>[] andalso length list >1)
   then orderDataByList(tl list,dataCopy,dataCopy,param,auxList)
   else auxList
end;

(*Orders data considering a specified metadata and order extract from param.
The metadata of a same data must be in adjacents positions.*)
fun orderDataBasedOnParam(data, param)=
let
val a =getFirstDataValue(param, paramOrder)
val b =if (a<>"0" andalso getFirstDataValue(param, paramSort)<>paramOrderDefault)
        then true
        else false
val c =if a<>"0"
         then remdupl (createListOfMetadataValueBasedOnParam(data, param,[]))
         else []
val d = if (c<>[])
          then
             if (b)
             then rev (orderListByNumOrChar(c))
             else orderListByNumOrChar(c)
          else []
val e = if convertStringListToIntList(d,[]) = [] then "int" else "string"
val f = if (d<>[]) then orderDataByList(d, data, data, param, []) else []
in
f
end;

(*Orders a data list by data id*)
fun orderDataByIdDataList(e::list, data, auxList)=
orderDataByIdDataList(list, data, auxList^^getDataById(data, e,[]))
|orderDataByIdDataList(null, data, auxList)=auxList;

(*Orders data by similarity. To reach this goal, it creates internally an ordered list containing the
data ID and the similarity. The metadata of a same data must be in adjacents positions.*)
fun orderDataBySimilarityValueBasedOnParam(data,param)=
let
val a =getFirstDataValue(param, paramOrder)=paramSL
val b =if (a andalso getFirstDataValue(param, paramSort)<>paramOrderDefault)
        then true else false
val c= if a
        then createSimilarityDataListBasedOnParam(data,param,param,[])
        else[]
val d=if c<>[]
        then orderDataBasedOnParam(c, param)
        else []
val e =if d<>[] then remdupl (getIdDataListFromData(d,[])) else []
val f = if e<>[]
        then orderDataByIdDataList(e,data,[])
        else []
in
f
end;
```

```
(*Gets the line at the end of a data element in a data list. The data is identified by idData.
The search starts at startLine that is in the desired data*)
fun getLastDataLineNumber(dataList : DATA, idData, startLine)=
if dataList=[] orelse startLine>(length dataList) orelse startLine = 0 orelse idData=0
then 0
else
   if (startLine = length dataList)
   then startLine
   else
      if (#idData (List.nth(dataList, startLine-1))) <> (#idData (List.nth(dataList, startLine)))
      then startLine
      else getLastDataLineNumber(dataList, idData, startLine+1)

(*Gets the data tuple line number, starting from start*)
fun getDataTupleLineNumber(dataList, dataTuple, start, plus)=
if dataList<>[] andalso length dataList >= start + plus
then
   if start = 0
   then getDataTupleLineNumber(dataList, dataTuple, 1, plus)
   else
      if List.nth(dataList,start+plus-1) = dataTuple
      then start + plus
      else getDataTupleLineNumber(dataList, dataTuple, start, plus+1)
else start;

(*Searchs data in an ordered list, according to a specific attribute.
aux1 returns the id of the data that is equal to. If the searched data does not exist,
aux1 returns the right position where the data should be put.
w returns the line that contains the attribute in this data.
The function returns the nearest line number of the immediate higher data*)
fun searchDataInOrderedDataList(dataList : DATA, dataListCopy, value, param, aux1, aux2) =
let
val a = if dataList<>[] then length dataList else 0
val b = if a<>0 then (a div 2) else 0
val x =if param<>[] then getFirstDataValue(param, paramOrder) else "0"
val y =if x<>"0" then getFirstDataValue(param, paramSort) else "0"
val z = if dataList <> [] andalso x <>"0"
        then getFirstDataTuple(List.drop(dataList,b), x) else []
val w = if z<>[] then getDataTupleLineNumber(dataListCopy, hd z, aux2 + b, 0) else aux2
val p = if z=[] then true else
            if (checkAndConvertStrToInt(value)<>0) andalso (checkAndConvertStrToInt(#value (hd
z))<>0)
            then INT.lt(checkAndConvertStrToInt(value), checkAndConvertStrToInt(#value (hd z)))
            else STRING.lt(value, (#value (hd z)))
val n = if z<>[] then getIdFromDataTuple(z) else aux1
in
if dataList<>[] andalso param<>[]
then
   if y = "ascendent"
   then
      if z = []
      then searchDataInOrderedDataList(List.take(dataList,b), dataListCopy, value, param, n, aux2)
      else
         if (value = (#value (hd z)))
         then intToStr(getLastDataLineNumber(dataListCopy, n, w))
```

```
        else
            if length dataList > 1
            then
                if p
                then searchDataInOrderedDataList(List.take(dataList,b), dataListCopy, value, param, n, aux2)
                else searchDataInOrderedDataList(List.drop(dataList,b), dataListCopy, value, param, n, aux2+b)
            else
                if p
                then intToStr(0)
                else intToStr(getLastDataLineNumber(dataListCopy, n, w))
    else
        if z = []
        then searchDataInOrderedDataList(List.take(dataList,b), dataListCopy, value, param, n, aux2)
        else
            if (value = (#value (hd z)))
            then intToStr(getLastDataLineNumber(dataListCopy, n, w))
            else
                if length dataList > 1
                then
                    if p= false
                    then searchDataInOrderedDataList(List.take(dataList,b), dataListCopy, value, param, n, aux2)
                    else searchDataInOrderedDataList(List.drop(dataList,b), dataListCopy, value,param, n, aux2+b)
                else
                    if p=false
                    then intToStr(0)
                    else intToStr(getLastDataLineNumber(dataListCopy, n, w))
else intToStr(0)
end;

(*Orders data considering a specified metadata, extracted from param.
The metadata of a same data must be in adjacents positions.*)
fun orderDataBasedOnParamOptimized(data, param, aux)=
let
val a =if param<>[] then getFirstDataValue(param, paramOrder) else "0"
val b = if a<>"0" then getFirstDataValue(param, paramSort) else "0"
val c = if a<>"0" andalso data<>[] then getFirstDataValue(data,a) else "0"
val d = if a<>"0" andalso data<>[] then getFirstDataTuple(data,a) else []
val e = if d<>[] then getIdFromDataTuple(d) else 0
val f = if (data<>[] andalso param<>[] andalso e<>0) then getDataById(data,e,[]) else []
val g =if a<>"0"
        then searchDataInOrderedDataList(aux, aux, c, param, 0, 0)
        else "0"
val h = if checkAndConvertStrToInt(g)<>0
         then
            List.take(aux, checkAndConvertStrToInt(g))^^f^^
            List.drop(aux, checkAndConvertStrToInt(g))
         else f^^aux
in
if data<>[]
then orderDataBasedOnParamOptimized(putSub(data,f),param,h)
else h
end;
```

```
(*Orders data by similarity. To reach this goal, it creates internally an
ordered list containing the data ID and the similarity.
The metadata of a same data must be in adjacents positions.*)
fun orderDataBySimilarityValueBasedOnParamOptimized(data,param)=
let
val a =getFirstDataValue(param, paramOrder)=paramSL
val b =if (a andalso getFirstDataValue(param, paramSort)<>paramOrderDefault)
        then true
        else false
val c= if a
        then createSimilarityDataListBasedOnParam(data,param,param,[])
        else[]
val d=if c<>[]
        then orderDataBasedOnParamOptimized(c, param, [])
        else []
val e =if d<>[] then remdupl (getIdDataListFromData(d,[])) else []
val f = if e<>[]
        then orderDataByIdDataList(e,data,[])
        else []
in
f
end;

(*Gets data that belongs to list1 and exists on list 2 with the same id and metadata.
It demands that each set id+metadata must be unique.*)
fun getEqualDataOnTwoLists
({idData, metadata, value}::list1 : DATA, list1Copy, list2 : DATA, auxList) =
if detectDataByIdDataAndMetadata(list2, idData, metadata)
then getEqualDataOnTwoLists(
            putSub(
               list1,
               getAllDataByIdAndMetadata(
                  {idData=idData, metadata=metadata, value=value}::list1, idData, metadata, [])),
            putSub(list1Copy, getAllDataByIdAndMetadata(list1Copy, idData, metadata, [])),
            list2,
            getAllDataByIdAndMetadata(list1Copy, idData, metadata, [])^^auxList)
else getEqualDataOnTwoLists(list1, list1Copy, list2, auxList)
|getEqualDataOnTwoLists(nil, list1Copy, list2, auxList) = rev(auxList);

(*Gets all data, but the last, that has the same metadata and similar value on one list.
Using comparedMetadata, a specific metadata can be compared or all of them.
It demands that each set id+metadata+data must be unique.*)
fun getSimilarDataOnOneList
(list1 : DATA, list1Copy, param : DATA, auxList : DATA) =
let
val a = if list1<>[] then true else false
val b = if a = true
         andalso ((#metadata (hd list1))= getFirstDataValue(param, paramCM)
         orelse getFirstDataValue(param, paramCM)="any"
         orelse getFirstDataValue(param, paramCM)="")
         then true else false
val c = if b = true
         then detectDataByMetadataAndSimilarValue
        (tl list1, #metadata (hd list1), #value (hd list1), param)
        else false
val d = if c = true
```

```
        then getDataByIdWithoutRepetition(list1Copy, #idData (hd list1), [])
        else []
in
if a = true
then
   if b = true andalso c = true
   then getSimilarDataOnOneList(putSub(list1, d),
                                putSub(list1Copy, d),
                                param,
                                auxList^^d)
   else getSimilarDataOnOneList(
            tl list1, list1Copy, param, auxList)
else auxList
end;

(*Gets data that belongs to list1 and exists on list 2 with the same metadata and similar value.
Using comparedMetadata, a specific metadata can be compared or all of them.
It demands that each set id+metadata+data must be unique.*)
fun getSimilarDataOnTwoLists
({idData, metadata, value}::list1 : DATA, list1Copy, list2 : DATA, param, auxList) =
if metadata= getFirstDataValue(param, paramCM)
orelse getFirstDataValue(param, paramCM)="any"
orelse getFirstDataValue(param, paramCM)=""
then
   if detectDataByMetadataAndSimilarValue(list2, metadata, value,  param)
   then getSimilarDataOnTwoLists(
            putSub(
               list1,
               getDataByIdWithoutRepetition(
                  {idData=idData, metadata=metadata, value=value}::list1, idData, [])),
            putSub(list1Copy, getDataByIdWithoutRepetition(list1Copy, idData, [])),
            list2,
            param,
            auxList^^getDataByIdWithoutRepetition(list1Copy, idData, []))
   else getSimilarDataOnTwoLists(list1, list1Copy, list2, param, auxList)
else getSimilarDataOnTwoLists(list1, list1Copy, list2, param, auxList)
|getSimilarDataOnTwoLists(nil, list1Copy, list2, param, auxList) = auxList;

(*Discards all data, but the last, that have the same metadata and similar value on one list.
Using comparedMetadata, a specific metadata can be compared or all of them*)
fun discardSimilarDataOnOneList(list1, param) =
putSub(list1, getSimilarDataOnOneList(list1, list1,
param,
[]));

(*Discards data that belongs to list1 and exists on list 2 with the same metadata and similar value.
Using comparedMetadata, a specific metadata can be compared or all of them*)
fun discardSimilarDataOnTwoLists(list1, list2, param)=
putSub(
discardSimilarDataOnOneList(list1, param),
   getSimilarDataOnTwoLists(
      discardSimilarDataOnOneList(list1, param),
      discardSimilarDataOnOneList(list1, param),
      discardSimilarDataOnOneList(list2, param),
      param,
      []));
```

```
(**********Functions - Patterns - Patterns Dependents - Check******)

(*Checks if it exists a data that has a specific metadata with the value “true”*)
fun checkPattern(data : DATA, metadata : STRING) =
if detectDataByMetadataAndValue(data, metadata, “true”)
then true
else false;

(**********Functions - Patterns - Patterns Dependents - Execute****)

(*Returns all data, but the last, that has the same metadata and value on one list.
It uses the paramDS1 to find in param the metadata that will be compared*)
fun putSimilarData(data : DATA, param: DATA) =
rev(getSimilarDataOnOneList(data, data,param,[]));

(*Returns data that belongs to data list and exists on param list with the same  metadata and
value*)
fun putProhibitedData(data: DATA, param: DATA) =
let
val a = getFirstDataValue(param, paramDataStrategy);
in
if a = “keep dependents”
then getSimilarDataOnTwoLists(data, data, param, param,[])
else if a = “discard all”
      then getSimilarDataOnTwoLists(data, data, param, param,[])^^
            getDataDependent(data, getIdDataListFromData(data, []), param, param, [])
      else putSub(getSimilarDataOnTwoLists(data, data, param, param,[]),
                  getDataThatHasDependent(data, getIdDataListFromData(data, []), param, param, []))
end;

(*Returns data that belongs to data list and exists on param list with the same  metadata and
value*)
fun putAllowedData(data: DATA, param: DATA) =
getSimilarDataOnTwoLists(
data, data, param, param,[]);

(*Returns data that belongs to data list and exists on param list with the same metadata and
value*)
fun putNotProhibitedData(data: DATA, param: DATA) =
putSub(data, putProhibitedData(data, param));

(*Discards data that belongs to list1 and exists on list 2 with the same metadata and value.*)
fun putNotSimilarData(extData: DATA, intData: DATA, param : DATA) =
discardSimilarDataOnTwoLists(
extData, intData, param);

(*Returns the data that has metadata equal to timeField and is an obsolete data, according to
validityPeriod*)
fun putOldDataIntern(head::data : DATA, dataCopy, param : DATA, auxList) =
   if (getFirstDataValue(param, paramCM) =
     (#metadata head)) andalso
     (strToInt(#value (head))<
     (strToInt(getFirstDataValue(param, paramCT))-strToInt(getFirstDataValue(param, paramPT))) )
   then putOldDataIntern(
            putSub(data, getDataByIdWithoutRepetition(head::data, (#idData head), [])),
```

```
            putSub(dataCopy, getDataByIdWithoutRepetition(dataCopy, (#idData head), [])),
            param,
            auxList^^getDataByIdWithoutRepetition(dataCopy, (#idData head), [])
   else putOldDataIntern(data, dataCopy, param, auxList)
   | putOldDataIntern(nil, dataCopy, param, auxList) = auxList;

(*Returns the obsolete data*)
fun putOldData(data : DATA, param : DATA) = putOldDataIntern(data, data, param, []);

(*Returns the data that has metadata equals frequencyField and is an useless data, according to
minimumAccessFrequency*)
fun putUselessDataIntern(head::data : DATA, dataCopy, param : DATA, auxList)=
  if (getFirstDataValue(param, paramCM) =
     (#metadata head)) andalso
     (strToInt(#value (head)) <
     (strToInt(getFirstDataValue(param, paramMP))) )
then putUselessDataIntern(
            putSub(data, getDataByIdWithoutRepetition(head::data, (#idData head), [])),
            putSub(dataCopy, getDataByIdWithoutRepetition(dataCopy, (#idData head), [])),
            param,
            auxList^^getDataByIdWithoutRepetition(dataCopy, (#idData head), [])
else putUselessDataIntern(data, dataCopy, param, auxList)
| putUselessDataIntern (nil, dataCopy, param, auxList) = auxList;

(*Returns the useless data*)
fun putUselessData(data : DATA, param : DATA) = putUselessDataIntern(data, data, param, []);

(*Creates from param a list where each element is composed by 3 itens: metadata, signal and
value.
This list is used to select relevant data with these metadata that obey the signal and value*)
fun getMetadataValueAndSignalOnList(param : DATA, auxList) =
let
val a = getFirstDataTuple(param, paramCM)
val b = if a<>[]
        then #idData (hd a)
        else 0
val c = getDataByIdWithoutRepetition(param, b, [])
val d = getFirstDataValue(c, paramCM)
val e = getFirstDataValue(c, paramCS)
val f  = getFirstDataValue(c, paramCV)
in
if c=[] orelse d="0" orelse e="0"
then auxList
else getMetadataValueAndSignalOnList(listsub param c, auxList^^[d::e::f::[]])
end;

(*Gets relevant data based on a list of list (LST), composed by metadata, signal and value.
The relevant data is extracted from data by using the LST*)
fun getRelevantDataBasedOnLST(list: LST, data: DATA, dataCopy: DATA,
auxList : DATA, auxList2 : DATA, param,n) =
let
val a = if list<>[] then getNDataTuple(data, List.nth(hd list, 0), n, 1) else []
val b = if list<>[] then List.nth(hd list, 1) else ""
val c = if a<>[] then #value (hd a) else ""
val d = if list<>[] then List.nth(hd list, 2) else ""
val e = if a<>[] then getDataByIdWithoutRepetition(data, #idData (hd a), []) else []
```

```
in
if a<>[]
then
   case b of
   "=" => if c=d
            then getRelevantDataBasedOnLST(list, listsub data e, dataCopy, e^^auxList, auxList2, param,1)
            else
               if (getNDataTuple(data, List.nth(hd list, 0), n+1, 1)<>[])
               then getRelevantDataBasedOnLST(list, data, dataCopy, auxList, auxList2, param,n+1)
               else getRelevantDataBasedOnLST(list, listsub data e, dataCopy, auxList, auxList2, param,1)
   | "<" => if (strToInt(c)<strToInt(d))
              then getRelevantDataBasedOnLST(list, listsub data e, dataCopy, e^^auxList, auxList2, param,1)
              else
               if (getNDataTuple(data, List.nth(hd list, 0), n+1, 1)<>[])
               then getRelevantDataBasedOnLST(list, data, dataCopy, auxList, auxList2, param,n+1)
               else getRelevantDataBasedOnLST(list, listsub data e, dataCopy, auxList, auxList2, param,1)
   | ">" => if (strToInt(c)>strToInt(d))
            then getRelevantDataBasedOnLST(list, listsub data e, dataCopy, e^^auxList,auxList2, param,1)
            else
               if (getNDataTuple(data, List.nth(hd list, 0), n+1, 1)<>[])
               then getRelevantDataBasedOnLST(list, data, dataCopy, auxList, auxList2, param,n+1)
               else getRelevantDataBasedOnLST(list, listsub data e, dataCopy, auxList, auxList2, param,1)
   | "<=" => if (strToInt(c)<=strToInt(d))
            then getRelevantDataBasedOnLST(list, listsub data e, dataCopy, e^^auxList,auxList2, param,1)
            else
               if (getNDataTuple(data, List.nth(hd list, 0), n+1, 1)<>[])
               then getRelevantDataBasedOnLST(list, data, dataCopy, auxList, auxList2, param,n+1)
               else getRelevantDataBasedOnLST(list, listsub data e, dataCopy, auxList, auxList2, param,1)
   | ">=" => if (strToInt(c)>=strToInt(d))
            then getRelevantDataBasedOnLST(list, listsub data e, dataCopy, e^^auxList, auxList2, param,1)
            else
               if (getNDataTuple(data, List.nth(hd list, 0), n+1, 1)<>[])
               then getRelevantDataBasedOnLST(list, data, dataCopy, auxList, auxList2, param,n+1)
               else getRelevantDataBasedOnLST(list, listsub data e, dataCopy, auxList, auxList2, param,1)
   | _ => getRelevantDataBasedOnLST(list, listsub data e, dataCopy, auxList, auxList2, param,1)
else
   if list<>[]
   then
      if(param = "AND")
      then getRelevantDataBasedOnLST(tl list, auxList, [], [], [], param,1)
      else
         getRelevantDataBasedOnLST(tl list, dataCopy, dataCopy, [],
            auxList2^^putSub(auxList,getEqualDataOnTwoLists(auxList2, auxList2, auxList, [])), param,1)
   else
      if (param = "AND")
      then data
      else auxList2
end;

(*Gets relevant data*)
fun putRelevantData(data, param) =
let
val a = detectDataByMetadataAndValue(param, paramConnective, "OR")
val b = putSub(param, getFirstDataTuple(param, paramConnective))
in
if a
```

```
then
   getRelevantDataBasedOnLST(
   getMetadataValueAndSignalOnList(b, []),
   data, data, [], [],"OR",1)
else
   getRelevantDataBasedOnLST(
   getMetadataValueAndSignalOnList(b, []),
   data, data, [], [],"AND",1)
end;

(*Gets irrelevant data*)
fun putIrrelevantData(data, param) =
putSub(data, putRelevantData(data, param))

(*Provides the result considering param, selected data, trash and the original data*)
fun getResult(param, selectedData, trash, originalData)=
let
val a =if getFirstDataValue(param, paramComplement)="false" then "" else
"complement"
val b =getFirstDataValue(param, paramOrder)
val c = if b=paramSL then "orderBySimilarity" else ""
in
if (a<>"complement")
then
   if (b="0")
   then dataToString(getEqualDataOnTwoLists
       (originalData, originalData, selectedData, []))
   else
      if (c<>"orderBySimilarity")
      then dataToString(orderDataBasedOnParamOptimized(getEqualDataOnTwoLists
                                                   (originalData, originalData, selectedData, []),
                                                   param,[]))
      else dataToString(orderDataBySimilarityValueBasedOnParamOptimized(
                           getEqualDataOnTwoLists(originalData, originalData, selectedData, []),
                           param))
else
   if (b="0")
   then dataToString(getEqualDataOnTwoLists
         (originalData, originalData, trash, []))
   else
      if (c<>"orderBySimilarity")
      then dataToString(orderDataBasedOnParamOptimized(getEqualDataOnTwoLists
                                                   (originalData, originalData, trash, []),
                                                  param,[]))
      else dataToString(orderDataBySimilarityValueBasedOnParamOptimized(
                           getEqualDataOnTwoLists(originalData, originalData, trash, []),
                           param))
end;
```

Anexo II - Data Killing Algebra

```
This file was created by Wallace A. Pinheiro
based on the file PointRectangleAlgebra.cpp
that is part of SECONDO Database.
The DataKillingAlgebra.cpp creates the Data Killing Algebra.
```

email:awallace@cos.ufrj.br; wallaceapinheiro@gmail.com

June 2009, Wallace A. Pinheiro

Contents

1 Overview

This algebra provides the type constructor *xdata* related to data killing patterns. Besides, it provides a set of data killing operators to manipulate data. These operators are:

1. *ad* (Allowed Data),
2. *bp* (Blocking Prohibited Data),
3. *bs* (Blocking Similar Data),
4. *bi* (Blocking Irrelevant Data),
5. *dp* (Discarding Prohibited Data),
6. *ds* (Discarding Similar Data),
7. *do* (Discarding Obsolete Data),
8. *di* (Discarding Irrelevant Data).

Besides, it provides two auxiliar operators that help to deal with data types. These operators are:

1. *showdata* (show xdata type),
2. *conc* (concatenates two data, discarding repetitions).

Futhermore, it provides three operators that deal with events. These operators are:

1. rt (register template),
2. ut (unregister template),
3. re (register event).

2 Preliminaries

2.1 Includes

```
#include "FTextAlgebra.h"
#include "Algebra.h"
#include "NestedList.h"
#include "NList.h"
#include "LogMsg.h"
#include "QueryProcessor.h"
#include "ConstructorTemplates.h"
#include "StandardTypes.h"
#include "RelationAlgebra.h"
#include "Progress.h"
#include "c_cpn.h"
```

The file `Algebra.h` is included, since the new algebra must be a subclass of class Algebra. All of the data available in Secondo has a nested list representation. Therefore, conversion functions have to be written for this algebra, too, and `NestedList.h` is needed for this purpose. The result of an operation is passed directly to the query processor. An instance of `QueryProcessor` serves for this. Secondo provides some standard data types, e.g. `CcInt`, `CcReal`, `CcString`, `CcBool`, which is needed as the result type of the implemented operations. To use them `StandardTypes.h` needs to be included. The file `c_cpn.h` is included to provide the communication between SECONDO and CPN Tools.

```
extern NestedList* nl;
extern QueryProcessor *qp;
```

The variables above define some global references to unique system-wide instances of the query processor and the nested list storage.

2.2 Auxiliaries

Within this algebra module implementation, we have to handle values of four different types defined in namespace *symbols*: *INT* and *REAL*, *BOOL* and *STRING*. They are constant values of the C++-string class.

Moreover, for type mappings some auxiliary helper functions are defined in the file `TypeMapUtils.h` which defines a namespace *mappings*.

```
#include "TypeMapUtils.h"
#include "Symbols.h"
```

```
using namespace symbols;
using namespace mappings;

#include <iostream>
#include <fstream>
#include <string>
using namespace std;

//For maximum portability, it is recommended to replace the function
//call to bzero() as follows:
#define bzero(b,len) (memset((b), '\0', (len)), (void) 0)
```

The implementation of the algebra is embedded into a namespace *dkp* in order to avoid name conflicts with other modules.

```
namespace dkp {
```

3 Type Constructor *xdata*

To define the Secondo type *xdata*, we need to (i) define a data structure, that is a class, to (ii) decide about a nested list representation, and (iii) write conversion functions from and to nested list representation. The function for converting from the list representation is the most involved one, since it has to check that the given list structure is entirely correct.

After we have described the traditional programming interface in the previous section, here in some places we use more recent alternative programming interfaces for implementing a type.

3.1 Data Structure - Class *XData*

```
class XData: public StandardAttribute
{
 public:
  XData( bool Defined, int idData, FText* metadata, FText* value );
  XData( const XData& rhs );
  ~XData() {}

  int GetidData()   const;
  FText* Getmetadata()  const;
  FText* Getvalue() const;
  void SetidData( int idData );
  void Setmetadata( FText* metadata );
  void Setvalue( FText* value );
  void Set(XData& xdat);
```

The following 9 virtual functions: IsDefined(), SetDefined(), Sizeof(), HashValue(), CopyFrom(), Compare(), Adjacent(), Clone(), Print(), need to be defined if we want to use *xdata* as an attribute type in tuple definitions.

```
  bool     IsDefined() const;
  void     SetDefined(bool Defined);
```

```
  size_t   Sizeof() const;
  size_t   HashValue() const;
  void     CopyFrom(const StandardAttribute* right);
  int      Compare(const Attribute * arg) const;
  bool     Adjacent(const Attribute * arg) const;
  XData*   Clone() const;
  ostream& Print( ostream &os ) const;
```

Here we will only implement the following these support functions, since the others have default implementations which can be generated at compile time using C++ template functionality.

```
  static Word     In( const ListExpr typeInfo, const ListExpr instance,
                      const int errorPos, ListExpr& errorInfo,
                      bool& correct );

  static ListExpr Out( ListExpr typeInfo, Word value );


  static bool     Open( SmiRecord& valueRecord,
                        size_t& offset, const ListExpr typeInfo,
                        Word& value );

  static bool     Save( SmiRecord& valueRecord, size_t& offset,
                        const ListExpr typeInfo, Word& w );
```

These 6 functions must be defined if we want to use *xdata* as an attribute type in tuple definitions: Create(), Delete(), Close(), Clone(), SizeOf() and Cast().

```
  static Word     CreateXData( const ListExpr typeInfo );
  static void     DeleteXData( const ListExpr typeInfo, Word& w );
  static void     CloseXData( const ListExpr typeInfo, Word& w );
  static Word     CloneXData( const ListExpr typeInfo, const Word& w );
  static void*    CastXData( void* addr );
  static int      SizeOfXData();

  static bool     KindCheck( ListExpr type, ListExpr& errorInfo );


 private:
  XData() {}
  // Since we want to use some default implementations we need
  // to allow access to private members for the class below.

  friend class ConstructorFunctions<XData>;

  bool defined;
  int id;
  FText* met;
  FText* val;

};
```

```
XData::XData( bool Defined, int idData, FText* metadata, FText* value )
{
  defined = Defined; id = idData; met = metadata; val = value;
}

XData::XData( const XData& rhs )
{
  defined = rhs.defined; id = rhs.id; met = rhs.met; val = rhs.val;
}

int XData::GetidData()   const { return id; }
FText* XData::Getmetadata()  const { return met; }
FText* XData::Getvalue() const { return val; }
void XData::SetidData(int idData) { id = idData; }
void XData::Setmetadata(FText* metadata) { met = metadata; }
void XData::Setvalue(FText* value) {val = value; }

void XData::Set(XData& xdat)
{
defined = xdat.defined;
id = xdat.id;
met = xdat.met;
val = xdat.val;
}
```

In the following, we give the definitions of the 9 virtual functions which are needed if we want to use *xdata* as an attribute type in tuple definitions.

```
bool XData::IsDefined() const {return (defined); }

void XData::SetDefined(bool Defined) {defined = Defined; }

size_t XData::Sizeof() const
{
  return sizeof( *this );
}

size_t XData::HashValue() const
{
  if(!defined)  return (0);
  unsigned long h;
  h=5*(5*id+strlen(met->Get()))+strlen(val->Get());
  return size_t(h);
}

void XData::CopyFrom(const StandardAttribute* right)
{
  const XData * d = (const XData*)right;
  id = d->id;
  met = d->met;
  val = d->val;
}
```

```
int XData::Compare(const Attribute * arg) const
{
 int res=0;
 const XData * d = (const XData* )(arg);
 if ( !d ) return (-2);

 if (!IsDefined() && !(arg->IsDefined()))  res=0;
 else if (!IsDefined())  res=-1;
    else  if (!(arg->IsDefined())) res=1;
       else
       {
                 char* mp1 = (char*)(this->Getmetadata())->Get();
                 char* mp2 = (char*)(d->Getmetadata())->Get();
                 char* vp1 = (char*)(this->Getvalue())->Get();
                 char* vp2 = (char*)(d->Getvalue())->Get();

                 if ((this->GetidData() == d->GetidData()) &&
                    (strcmp( mp2, mp1 )==0)&&
                    (strcmp( vp2, vp1 )==0)){
                    res=0;
                 }
                    else res=1;
                 }
  return (res);
}

bool XData::Adjacent(const Attribute *arg) const
{
  const XData *d = (const XData *)arg;
  //both undefined or they are equal
  if( this->Compare( d ) == 0 ) return 1;

  if (!IsDefined() || !(arg->IsDefined()))
     //one is undefined and another defined
     return 0;
  else      //both defined and they are not equal
  {
      if ((this->GetidData()+1)==d->GetidData())return 1;
      if ((this->GetidData()-1)==d->GetidData())return 1;
      return 0;
  }
}

XData* XData::Clone() const
{
  return (new XData( *this));
}

ostream& XData::Print(ostream &os) const
{
  return (os << id << "." << (char*)met->Get() << "." << (char*)val->Get());
}
```

3.2 Auxiliary Functions for Operations

```
//Trim function to a char*
char *trim( char *szSource )
{
char *pszEOS;//End of string
char *pszIOS;//Begin of string
/*Set pointer to end of string to point to the character just
  before the 0 at the end of the string.*/
pszEOS = szSource + strlen( szSource ) - 1;
while( pszEOS >= szSource && *pszEOS == ' ' )
*pszEOS-- = '\0';
// Set pointer to begin of string .
pszIOS = szSource ;
while( *pszIOS == ' ' )
*pszIOS++;
return pszIOS;
}

char* convertStringToChar(string str){
        char *p;
        p=&str[0];
        return p;
}

string convertCharPointerToString(char* charPointer){
        string str = charPointer;
        return str;
}

char *removeSubstr (char* str, char* substr)
{
    int len;
    register int i; //keep in register.

        len = strlen (substr); //keep the length of substr.
        char *aux;
      //it searches and gets the initial address of substr.
        aux = strstr (str, substr);
    while (aux){
         //it discards the substr at the beginning of the str
          if (aux == str)
                 for (i = 0; i < len; i++, ++str);
          else
        //i keeps the initial position of substr, len is the
        //quantity of characters after substr
          {
                      int j = strlen(str);
                      for (i = (aux - str); i < j; i++)
                      {
                              if(i+len<j){
                                *(str +i) = *(str +i+len);
                              }
```

```
                                else{
                                  *(str + i)='\0';
                                }
                        }
          }
         //it searches and gets the initial address of substr.
          aux = strstr (str, substr);
    }
    return str;
}

/*It deletes from eData2 tuples that exist in eData1.
After, it concatenates them*/
char *concTuples(char *eData1, char *eData2){
        char *start;
        char *end;
        char *sdata = eData1;//Initialization of pointer, always
                            //set address, never value of pointer.
        start = strstr(sdata, "<tu><at1>");
        end = eData1;
        while(start){
          //Search each tuple in eData1
          sdata = end;
          start = strstr(sdata, "<tu><at1>");
          sdata = start;
          if (start){
                        end = strstr(sdata, "</at3></tu>");
                        int tupleLength = end - start + 11;
                        char *tupleFound = (char*)malloc(tupleLength+1);
                        int i=0;
                        for (i=0; i<tupleLength; i++){
                          tupleFound[i] = sdata[i];
                        }
                        tupleFound[i++]='\0';
                        //Search tupleFound in eData2
                        eData2 = removeSubstr(eData2,tupleFound);
                        free(tupleFound);
          }
        }
        int len= strlen(eData1)+strlen(eData2)+1;
        char *tData = (char*)malloc(len); bzero(tData, len);
        strcat(tData,eData1);
        strcat(tData,eData2);

        return tData;
}


//Send data and param to CPN and get the result
char* callCPN(char* tParam)
{
        if (strlen(tParam)!=0){
          //obtem parametros de arquivo
```

```
          string s;
          string key1="cpn.address=";
          string key2="cpn.port=";
          string word1;
          string word2;
          int pos;
          char buffer[256];
          ifstream myfile ("../Algebras/DataKilling/cpn.properties");
          if (myfile.is_open()){
                  while (! myfile.eof() ){
                        myfile.getline (buffer,100);
                        s=buffer;
                        if(s.find(key1,0) != string::npos){
                                pos = s.find(key1,0) + key1.length();
                                word1 = s.substr(pos,s.length());
                        }
                                if(s.find(key2,0) != string::npos){
                                        pos = s.find(key2,0) + key2.length();
                                word2 = s.substr(pos,s.length());
                                }
                  }
                  myfile.close();
          }
          else{
                  cmsg.inFunError("File cpn.properties not found!");
          }
          //Carrega parametros e se comunica com a rede de Petri
          const char* charWord1 = word1.c_str();
          int intWord2 = atoi(word2.c_str());

          //Connect to CPN
          CPN_connect(charWord1, intWord2);
          //CPN_connect("127.0.0.1", 9000);
          //Send string to CPN
          CPN_send(tParam);
          //Receive string from CPN
          char* receive=CPN_receive();
          //Disconect from CPN
          CPN_disconnect();
          return (receive);
        }
        return(tParam);
}

//Send data and param to CPN and get the result
char* callCPN(char* tParam, char* tData)
{
        if ((strlen(tParam)!=0)&&(strlen(tData)!=0)){
          //obtem parametros de arquivo
          string s;
          string key1="cpn.address=";
          string key2="cpn.port=";
          string word1;
```

```
          string word2;
          int pos;
          char buffer[256];
          ifstream myfile ("../Algebras/DataKilling/cpn.properties");
          if (myfile.is_open()){
                  while (! myfile.eof() ){
                        myfile.getline (buffer,100);
                        s=buffer;
                        if(s.find(key1,0) != string::npos){
                                pos = s.find(key1,0) + key1.length();
                                word1 = s.substr(pos,s.length());
                        }
                                if(s.find(key2,0) != string::npos){
                                        pos = s.find(key2,0) + key2.length();
                                word2 = s.substr(pos,s.length());
                                }
                  }
                  myfile.close();
          }
          else{
                  cmsg.inFunError("File cpn.properties not found!");
          }
          //Carrega parametros e se comunica com a rede de Petri
          const char* charWord1 = word1.c_str();
          int intWord2 = atoi(word2.c_str());
          //Connect to CPN
          CPN_connect(charWord1, intWord2);
          //CPN_connect("127.0.0.1", 9000);

          //Send string to CPN
          CPN_send(tParam);
          CPN_send(tData);
          //Receive string from CPN
          char* receive=CPN_receive();
          //Disconect from CPN
          CPN_disconnect();
          return (receive);
        }
        return(tData);
}

//Send data and param to CPN and get the result
char* callCPN(char* tParam, char* tData1, char* tData2)
{
        if ((strlen(tParam)!=0)&&(strlen(tData2)!=0)){
          //obtem parametros de arquivo
          string s;
          string key1="cpn.address=";
          string key2="cpn.port=";
          string word1;
          string word2;
          int pos;
          char buffer[256];
```

```
          ifstream myfile ("../Algebras/DataKilling/cpn.properties");
          if (myfile.is_open()){
                  while (! myfile.eof() ){
                        myfile.getline (buffer,100);
                        s=buffer;
                        if(s.find(key1,0) != string::npos){
                                pos = s.find(key1,0) + key1.length();
                                word1 = s.substr(pos,s.length());
                        }
                                if(s.find(key2,0) != string::npos){
                                        pos = s.find(key2,0) + key2.length();
                                word2 = s.substr(pos,s.length());
                                }
                  }
                  myfile.close();
          }
          else{
                  cmsg.inFunError("File cpn.properties not found!");
          }
          //Carrega parametros e se comunica com a rede de Petri
          const char* charWord1 = word1.c_str();
          int intWord2 = atoi(word2.c_str());

          //Connect to CPN
          CPN_connect(charWord1, intWord2);
          //CPN_connect("127.0.0.1", 9000);
          //Send string to CPN
          CPN_send(tParam);
          CPN_send(tData1);
          CPN_send(tData2);
          //Receive string from CPN
          char* receive=CPN_receive();
          //Disconect from CPN
          CPN_disconnect();
          return (receive);
        }
        return(tData1);
}


/*Convert xdata to string with the format:
"<tu><at1>1</at1><at2>m1</at2><at3>v1</at3></tu>
<tu><at1>2</at1><at2>m2</at2><at3>v2</at3></tu>"*/
char* dataToString(Relation* r)
{
 //Get a iterator over the data tuples
  GenericRelationIterator* rit1 = r->MakeScan();
  GenericRelationIterator* rit2 = r->MakeScan();
  Tuple *t;
  char* tupleData;

  /*Calculate size of the string that will be created from the tuples
  The final tupleValue size needs to be calculate before these
```

```
attributions.
This is important to allocate memory to tupleValue, because it will
receive each data attribute once a time and also separator
characters.*/
int len=0;
while ((t = rit1->GetNextTuple()) != 0)
{
  TupleType* tupleType = r->GetTupleType();
  int j = tupleType->GetNoAttributes();
       int tupleId;
       char* tupleMetadata;
       char* tupleValue;
  if(j==1){//the attribute is XData type
              tupleId = (int)((XData*)t->GetAttribute(0))->GetidData();
              tupleMetadata = (char*)((FText*)((XData*)t->GetAttribute(0))
              ->Getmetadata())->Get();
              tupleValue = (char*)((FText*)((XData*)t->GetAttribute(0))
              ->Getvalue())->Get();
       }
       else{//the attributes are types: int, text and text
         tupleId = ((CcInt*)t->GetAttribute(0))->GetIntval();
         tupleMetadata = (char*)((FText*)t->GetAttribute(1))->Get();
         tupleValue = (char*)((FText*)t->GetAttribute(2))->Get();
       }
       //convert to string from tupleId to buf
       char buf[10];itoa(tupleId, buf, 10);

       //The total lenght of a data tuple is the size of all attributes  +
       //42 characters of the sequence:
       //"<tu><at1></at1><at2></at2><at3></at3></tu>"
       len = len + strlen(buf)+strlen(tupleMetadata)+strlen(tupleValue)+42;
}
//+ One null character '/0' to close the string
len++;
//Memory allocation
tupleData = (char*)malloc(len); bzero(tupleData, len);

/*Convert data tuples to string as:
"<tu><at1>1</at1><at2>m1</at2><at3>v1</at3></tu>
 <tu><at1>2</at1><at2>m2</at2><at3>v2</at3></tu>"*/
while ((t = rit2->GetNextTuple()) != 0)
{
  TupleType* tupleType = r->GetTupleType();
  int j = tupleType->GetNoAttributes();
       int tupleId;
       char* tupleMetadata;
       char* tupleValue;
  if(j==1){//the attribute is XData type
              tupleId = (int)((XData*)t->GetAttribute(0))->GetidData();
              tupleMetadata = (char*)((FText*)((XData*)t
              ->GetAttribute(0))->Getmetadata())->Get();
              tupleValue = (char*)((FText*)((XData*)t
              ->GetAttribute(0))->Getvalue())->Get();
```

```
        }
        else{//the attributes are types: int, text and text
          tupleId = ((CcInt*)t->GetAttribute(0))->GetIntval();
          tupleMetadata = (char*)((FText*)t->GetAttribute(1))->Get();
          tupleValue = (char*)((FText*)t->GetAttribute(2))->Get();
        }
        //convert to string from tupleId to buf
        char buf[10];itoa(tupleId, buf, 10);

        //Put separators
        strcat(tupleData, "<tu><at1>");
        strcat(tupleData, buf);
        strcat(tupleData, "</at1><at2>");
     strcat(tupleData, tupleMetadata);
        strcat(tupleData, "</at2><at3>");
        strcat(tupleData, tupleValue);
        strcat(tupleData, "</at3></tu>");
  }
  //Free memory
  //free(tupleData);
  delete rit1;
  delete rit2;

return (tupleData);
}


/*Convert to data from string format:
"<tu><at1>1</at1><at2>m1</at2><at3>v1</at3></tu>
<tu><at1>2</at1><at2>m2</at2><at3>v2</at3></tu>"*/
GenericRelation* stringToData(char* receive, GenericRelation* rel, Relation* r)
{
  int l = 0;
  int end = 0;
  while((NULL!=receive)&&(end==0))
  {
        char* token1; char* token2; char* token3;
        //Memory allocation is set to string size
   token1 = (char*)malloc(strlen(receive)+1); bzero(token1, strlen(receive)+1);
        token2 =
        (char*)malloc(strlen(receive)+1); bzero(token2, strlen(receive)+1);
        token3 =
        (char*)malloc(strlen(receive)+1); bzero(token3, strlen(receive)+1);
        int stop = 0; int m = 0;
   while((receive[l]!='\0')&&(stop==0)&&(end==0)){
        if((receive[l]=='<')&&(receive[l+1]=='t')&&(receive[l+2]=='u')&&
        (receive[l+3]=='>')&& (receive[l+4]=='<')&&(receive[l+5]=='a')&&
        (receive[l+6]=='t')&&(receive[l+7]=='1')&&(receive[l+8]=='>')){
                                stop=1;
                        }else{
                                l++;
                        }
        }
```

```
	if(receive[l+9]!='\0'){
		l=l+9; m=0; stop=0;
	}else{
		end=1;
	}
	while((receive[l]!='\0')&&(stop==0)&&(end==0)){
		if((receive[l]=='<')&&(receive[l+1]=='/')&&(receive[l+2]=='a')&&
		(receive[l+3]=='t')&&(receive[l+4]=='1')&&(receive[l+5]=='>')&&
		(receive[l+6]=='<')&&(receive[l+7]=='a')&&(receive[l+8]=='t')
		&&(receive[l+9]=='2')&&(receive[l+10]=='>')){
				stop=1;
			}else{
				token1[m++]=receive[l++];
			}
	}
	if(receive[l+11]!='\0'){
		l=l+11; m=0; stop=0;
	}else{
		end=1;
	}
	while((receive[l]!='\0')&&(stop==0)&&(end==0)){
		if((receive[l]=='<')&&(receive[l+1]=='/')&&(receive[l+2]=='a')&&
		(receive[l+3]=='t')&&(receive[l+4]=='2')&&(receive[l+5]=='>')&&
		(receive[l+6]=='<')&&(receive[l+7]=='a')&&(receive[l+8]=='t')&&
		(receive[l+9]=='3')&&(receive[l+10]=='>')){
				stop=1;
			}else{
				token2[m++]=receive[l++];
			}
	}
	if(receive[l+11]!='\0'){
		l=l+11; m=0; stop=0;
	}else{
		end=1;
	}
	while((receive[l]!='\0')&&(stop==0)&&(end==0)){
		if((receive[l]=='<')&&(receive[l+1]=='/')&&(receive[l+2]=='a')&&
		(receive[l+3]=='t')&&(receive[l+4]=='3')&&(receive[l+5]=='>')&&
		(receive[l+6]=='<')&&(receive[l+7]=='/')&&(receive[l+8]=='t')&&
		(receive[l+9]=='u')&&(receive[l+10]=='>')){
				stop=1;
			}else{
				token3[m++]=receive[l++];
			}
	}	if(receive[l+11]!='\0'){
		l=l+11;
	}else{
		end=1;
	}
	TupleType* tupleType = r->GetTupleType();
int j = tupleType->GetNoAttributes();

	if (j==1){//the attribute is XData type
```

```
                XData* xd = new XData(true, atoi(token1),
                new FText::FText(true,token2),new FText::FText(true,token3));
                Tuple* tx= new Tuple( r->GetTupleType());
                tx->PutAttribute(0, (Attribute*)xd);
                rel->AppendTuple(tx);
        }
        else{//the attributes are types: int, text and text
                Tuple* tx= new Tuple( r->GetTupleType());
                tx->PutAttribute(0,new CcInt(true,atoi(token1)));
                tx->PutAttribute(1,new FText::FText(true,token2));
                tx->PutAttribute(2,new FText::FText(true,token3));
        rel->AppendTuple(tx);
        }
      //Free memory
        free(token1); free(token2); free(token3);
  } //end while

  return (rel);
}
```

3.3 List Representation and *In*/*Out* Functions

The list representation of an xdata is

```
(idData metadata value)
```

In contrast to the code examples above, we use here the class *NList* instead of the static functions `nl->f(...)`. Its interface is described in file `NList.h`. It is a simple wrapper for calls like `nl->f(...)` and provides a more object-oriented access to a nested list.

This class was implemented more recently; hence there is a lot of code which uses the older interface. But as you can observe, the code based on *NList* is more compact, easier to read, understand, and maintain. Thus we recommend to use this interface.

```
//Convert from external representation to internal representation
Word
XData::In( const ListExpr typeInfo, const ListExpr instance,
                const int errorPos, ListExpr& errorInfo, bool& correct )
{
  correct = false;
  Word result = SetWord(Address(0));
  const string errMsg = "Expecting a list of xdata!";

  NList list(instance);
  //When you check list structures it will be a good advice to detect
  //errors as early as possible to avoid deep nestings of if statements.
  if (list.length() != 3){
    cmsg.inFunError(errMsg);
    return result;
```

```
  }

  NList First =list.first();
  NList Second = list.second();
  NList Third =list.third();

  if ( First.isInt() && Second.isText()
      && Third.isText() ) {

        int id = First.intval();
                //the function str() retrieves from a ListExpr
            //the string value from a symbol, a string or a text
        string m = Second.str();
                FText* met = new FText(true, m.c_str());
                string v = Third.str();
                FText* val = new FText(true, v.c_str());
        correct = true;
                XData* r = new XData(true, id, met, val);
      result.addr = r;
  }
  else {
        cmsg.inFunError(errMsg);
  }
  return result;
}

//convert from internal representation to external representation
ListExpr
XData::Out( ListExpr typeInfo, Word value )
{
  XData* dataOut = static_cast<XData*>( value.addr );
  const char* m;
  if(((FText*)dataOut->Getmetadata())->IsDefined()){
      m = ((FText*)dataOut->Getmetadata())->Get();
  } else {
      m = "undef";
  }
  const char* v;
  if(((FText*)dataOut->Getmetadata())->IsDefined()){
      v = ((FText*)dataOut->Getvalue())->Get();
  } else {
      v = "undef";
  }

  NList threeElems(
           NList( dataOut->GetidData() ),
                          NList(   nl->TextAtom(m)    ),
                          NList(   nl->TextAtom(v)    )  );
  return threeElems.listExpr();
}
```

3.4 Storage Management: *Open*, *Save*, and *Create*

The *open* and *save* functions need an *SmiRecord* as argument which contains the binary representation of the type, starting at the position indicated by *offset*. The implementor has to read out or write in data there and adjust the offset. The argument *typeinfo* is needed only for complex types whose constructors can be parameterized, e.g. rel(tuple(...)).

```
bool
XData::Open( SmiRecord& valueRecord,
                  size_t& offset,
                  const ListExpr typeInfo,
                  Word& value )
{
  XData *xd = (XData*)Attribute::Open( valueRecord, offset, typeInfo );
  value = SetWord( xd );
  return true;
}



bool
XData::Save( SmiRecord& valueRecord,
                  size_t& offset,
                  const ListExpr typeInfo,
                  Word& value )
{
  XData *xd = (XData *)value.addr;

  { Attribute::Save( valueRecord, offset, typeInfo, xd ); }

  return true;
}
```

In the following, we give the definitions of the 6 functions which are needed if we want to use *xdata* as an attribute type in tuple definitions.

```
Word
XData::CreateXData( const ListExpr typeInfo )
{
  return (SetWord( new XData(false, 0, new FText(false), new FText(false)) ));
}

void
XData::DeleteXData( const ListExpr typeInfo, Word& w )
{
  delete static_cast<XData*>( w.addr );
  w.addr = 0;
}

void
XData::CloseXData( const ListExpr typeInfo, Word& w )
{
```

```
  delete static_cast<XData*>( w.addr );
  w.addr = 0;
}

Word
XData::CloneXData( const ListExpr typeInfo, const Word& w )
{
  XData* d = static_cast<XData*>( w.addr );
  return SetWord( new XData(*d) );
}

void*
XData::CastXData( void* addr )
{
  return (new (addr) XData);
}

int
XData::SizeOfXData()
{
  return sizeof(XData);
}
```

3.5 Type Description

The property function is deprecated. Instead this is done by implementing a subclass of *ConstructorInfo*.

```
struct xdataInfo : ConstructorInfo {

  xdataInfo() {

    name         = XDATA;
    signature    = "-> DATA";
    typeExample  = XDATA;
    listRep      = "(<idData> <metadata> <value>)";
    valueExample = "(1 <text>text</text---> <text>text</text--->)";
    remarks      = "Parameters: integer, text and text.";
  }
};
```

3.6 Kind Checking Function

This function checks whether the type constructor is applied correctly. Since type constructor *xpoint* does not have arguments, this is trivial.

```
bool
XData::KindCheck( ListExpr type, ListExpr& errorInfo )
{
  return (nl->IsEqual( type, XDATA ));
}
```

3.7 Creation of the Type Constructor Instance

Here we also use a new programming interface. As you may have observed, most implementations of the support functions needed for registering a Secondo type are trivial to implement. Hence, we offer a template class *ConstructorFunctions* which will create many default implementations of functions used by a Secondo type. For details refer to `ConstructorFunctions.h`. However, some functions need to be implemented since the default may not be sufficient. The default kind check function assumes that the type constructor does not have any arguments.

```
struct xdataFunctions : ConstructorFunctions<XData> {

  xdataFunctions()
  {
    // re-assign some function pointers
    in = XData::In;
    out = XData::Out;
    create = XData::CreateXData;
         deletion = XData::DeleteXData;
         close = XData::CloseXData;
         clone = XData::CloneXData;
         cast = XData::CastXData;
         sizeOf = XData::SizeOfXData;
         kindCheck = XData::KindCheck;

    //The default implementations for open and save are only
    //suitable for a class which is derived from class ~Attribute~,
    //hence open and save functions must be overwritten here.

    open = XData::Open;
    save = XData::Save;
  }
};

xdataInfo xdi;
xdataFunctions xdf;
TypeConstructor xdataTC( xdi, xdf );
```

4 Creating Operators

4.1 Type Mapping Functions

A type mapping function checks whether the correct argument types are supplied for an operator; if so, it returns a list expression for the result type, otherwise the symbol *typeerror*. Again we use interface *NList.h* for manipulating list expressions.

Type mapping for *showxdata* is

(xdata) -¿ (xdata)

```
ListExpr
showXDataTypeMap( ListExpr args )
```

```
{
  NList type(args);
  if ( !type.hasLength(1) ) {
    return NList::typeError("Expecting one argument.");
  }

  ListExpr attrtype = nl->Empty();
  attrtype = type.first().listExpr();
  //Return the type of the first argument
  if ( nl->SymbolValue(attrtype) == XDATA ){
                return type.first().listExpr();
  }
  return NList::typeError("Expecting one argument.");
}
```

Type mapping for *dk* is

(rel (xdata)) (rel (xdata)) -¿ (rel (xdata)) OR (rel (tuple (int text text))) (rel (tuple (int text text))) -¿ (rel (tuple (int text text)))

```
ListExpr
dkTypeMap( ListExpr args )
{
  NList type(args);
  if ( !type.hasLength(2) )
  {
    return NList::typeError("Expecting two arguments.");
  }

  NList first = type.first();
  NList second = type.second();
  ////////////////////test for xdata//////////////////////////
  //Test fist argument symbols for xdata
  if (!first.hasLength(2) ||
     !first.first().isSymbol(REL) ||
     !first.second().hasLength(2) ||
          !first.second().first().isSymbol(TUPLE) ) {
          return
          NList::typeError("First argument is not 2 relations of tuples!");
  }
  //Test second argument symbols for xdata
  if (!second.hasLength(2) ||
     !second.first().isSymbol(REL) ||
     !second.second().hasLength(2) ||
     !second.second().first().isSymbol(TUPLE) ) {
          return
          NList::typeError("Second argument is not 2 relations of tuples!");
  }

  //test xdata types
  ListExpr attrtype1 = nl->Empty();
  ListExpr attrtype2 = nl->Empty();
  //browse  over the first argument: '(rel (xdataValue xdata)'
```

```
//get xdata
attrtype1 = first.second().second().first().second().listExpr();
//browse  over the second argument: '(rel (xdata)'
//get xdata
attrtype2 = second.second().second().first().second().listExpr();

//Return the type of the first argument
if ( nl->SymbolValue(attrtype1) == XDATA && nl->SymbolValue(attrtype2) ==
XDATA  ){
              return type.first().listExpr();
}

//////////////////test for tuple//////////////////////////
//Test fist argument symbols for tuple
if (
  !first.hasLength(2) ||
  !first.first().isSymbol(REL) ||
  !first.second().hasLength(2) ||
  !first.second().first().isSymbol(TUPLE) ||
       !first.second().second().hasLength(3) ||
  !IsTupleDescription( first.second().second().listExpr() ) )
{
  return NList::typeError("Error in first argument!");
}
//Test second argument symbols for tuple
if (
  !second.hasLength(2)  ||
  !second.first().isSymbol(REL) ||
  !second.second().hasLength(2) ||
  !second.second().first().isSymbol(TUPLE) ||
       !second.second().second().hasLength(3) ||
  !IsTupleDescription( second.second().second().listExpr() ) )
{
  return NList::typeError("Error in second argument!");
}

    //test int, text, text
    ListExpr attrtype11 = nl->Empty();
      ListExpr attrtype12 = nl->Empty();
      ListExpr attrtype13 = nl->Empty();
      ListExpr attrtype21 = nl->Empty();
      ListExpr attrtype22 = nl->Empty();
      ListExpr attrtype23 = nl->Empty();

      //browse  over first argument:
      //'(rel (tuple (idData int metadata text value text)))'
      attrtype11 =
      first.second().second().first().second().listExpr();//get 'int'
      attrtype12 =
      first.second().second().second().second().listExpr();//get 'text'
      attrtype13 =
      first.second().second().third().second().listExpr();//get 'text'
      //browse  over second argument:
```

```
        //'(rel (tuple (idData int metadata text value text)'
        attrtype21 =
        second.second().second().first().second().listExpr();//get 'int'
        attrtype22 =
        second.second().second().second().second().listExpr();//get'text'
        attrtype23 =
        second.second().second().third().second().listExpr();//get 'text'

  //Check types
  if ( nl->SymbolValue(attrtype11) != INT || nl->SymbolValue(attrtype21) !=
  INT  )
  {
    return NList::typeError("Attribute type is not of type int.");
  }
  if ( nl->SymbolValue(attrtype12) != TEXT || nl->SymbolValue(attrtype22) !=
  TEXT  )
  {
    return NList::typeError("Attribute type is not of type text.");
  }
  if ( nl->SymbolValue(attrtype13) != TEXT || nl->SymbolValue(attrtype23) !=
  TEXT  )
  {
    return NList::typeError("Attribute type is not of type text.");
  }

  // return the type of the first argument
  return type.first().listExpr();
}
```

Type mapping for *dk2* is

(rel (tuple (int text text))) (rel (tuple (int text text))) (rel (tuple (int text text))) -¿ (rel (tuple (int text text))) OR (rel (xdata)) (rel (xdata)) (rel (xdata)) -¿ (rel (xdata))

```
ListExpr
dk2TypeMap( ListExpr args )
{
  NList type(args);
  if ( !type.hasLength(3) )
  {
    return NList::typeError("Expecting three arguments.");
  }

  NList first = type.first();
  NList second = type.second();
  NList third = type.third();
  ////////////////////test for xdata/////////////////////////
  //Test fist argument symbols for xdata
  if (!first.hasLength(2) ||
     !first.first().isSymbol(REL) ||
     !first.second().hasLength(2) ||
          !first.second().first().isSymbol(TUPLE) ) {
          return
```

```
        NList::typeError("First argument is not 2 relations of tuples!");
}
//Test second argument symbols for xdata
if (!second.hasLength(2) ||
   !second.first().isSymbol(REL) ||
   !second.second().hasLength(2) ||
   !second.second().first().isSymbol(TUPLE) ) {
        return
        NList::typeError("Second argument is not 2 relations of tuples!");
}
//Test third argument symbols for xdata
if (!third.hasLength(2) ||
   !third.first().isSymbol(REL) ||
   !third.second().hasLength(2) ||
   !third.second().first().isSymbol(TUPLE) ) {
        return
        NList::typeError("third argument is not 2 relations of tuples!");
}

//test xdata types
ListExpr attrtype1 = nl->Empty();
ListExpr attrtype2 = nl->Empty();
ListExpr attrtype3 = nl->Empty();
//browse  over the first argument: '(rel (xdataValue xdata)'
//get xdata
attrtype1 = first.second().second().first().second().listExpr();
//browse  over the second argument: '(rel (xdata)'
//get xdata
attrtype2 = second.second().second().first().second().listExpr();
//browse  over the third argument: '(rel (xdata)'
//get xdata
attrtype3 = third.second().second().first().second().listExpr();

//Return the type of the first argument
if ( nl->SymbolValue(attrtype1) == XDATA && nl->SymbolValue(attrtype2) ==
XDATA && nl->SymbolValue(attrtype3) == XDATA  ){
             return type.first().listExpr();
}

//////////////////test for tuple//////////////////////////
//Test fist argument symbols for tuple
if (
  !first.hasLength(2) ||
  !first.first().isSymbol(REL) ||
  !first.second().hasLength(2) ||
  !first.second().first().isSymbol(TUPLE) ||
       !first.second().second().hasLength(3) ||
  !IsTupleDescription( first.second().second().listExpr() ) )
{
  return NList::typeError("Error in first argument!");
}
//Test second argument symbols for tuple
if (
```

```
  !second.hasLength(2)  ||
  !second.first().isSymbol(REL) ||
  !second.second().hasLength(2) ||
  !second.second().first().isSymbol(TUPLE) ||
       !second.second().second().hasLength(3) ||
  !IsTupleDescription( second.second().second().listExpr() ) )
{
  return NList::typeError("Error in second argument!");
}
//Test third argument symbols for tuple
if (
  !third.hasLength(2)  ||
  !third.first().isSymbol(REL) ||
  !third.second().hasLength(2) ||
  !third.second().first().isSymbol(TUPLE) ||
       !third.second().second().hasLength(3) ||
  !IsTupleDescription( third.second().second().listExpr() ) )
{
  return NList::typeError("Error in third argument!");
}

    //test int, text, text
    ListExpr attrtype11 = nl->Empty();
      ListExpr attrtype12 = nl->Empty();
      ListExpr attrtype13 = nl->Empty();
      ListExpr attrtype21 = nl->Empty();
      ListExpr attrtype22 = nl->Empty();
      ListExpr attrtype23 = nl->Empty();
      ListExpr attrtype31 = nl->Empty();
      ListExpr attrtype32 = nl->Empty();
      ListExpr attrtype33 = nl->Empty();

      //browse  over first argument:
      //'(rel (tuple (idData int metadata text value text)))'
      attrtype11 =
      first.second().second().first().second().listExpr();//get 'int'
      attrtype12 =
      first.second().second().second().second().listExpr();//get 'text'
      attrtype13 =
      first.second().second().third().second().listExpr();//get 'text'
      //browse  over second argument:
      //'(rel (tuple (idData int metadata text value text)'
      attrtype21 =
      second.second().second().first().second().listExpr();//get 'int'
      attrtype22 =
      second.second().second().second().second().listExpr();//get'text'
      attrtype23 =
      second.second().second().third().second().listExpr();//get 'text'
      //browse  over third argument:
      //'(rel (tuple (idData int metadata text value text)'
      attrtype31 =
      third.second().second().first().second().listExpr();//get 'int'
      attrtype32 =
```

```
        third.second().second().second().second().listExpr();//get 'text'
        attrtype33 =
        third.second().second().third().second().listExpr();//get 'text'

    //Check types
    if ( nl->SymbolValue(attrtype11) != INT || nl->SymbolValue(attrtype21) !=
    INT || nl->SymbolValue(attrtype31) != INT )
    {
      return NList::typeError("Attribute type is not of type int.");
    }
    if ( nl->SymbolValue(attrtype12) != TEXT || nl->SymbolValue(attrtype22) !=
    TEXT || nl->SymbolValue(attrtype32) != TEXT )
    {
      return NList::typeError("Attribute type is not of type text.");
    }
    if ( nl->SymbolValue(attrtype13) != TEXT || nl->SymbolValue(attrtype23) !=
    TEXT || nl->SymbolValue(attrtype33) != TEXT )
    {
      return NList::typeError("Attribute type is not of type text.");
    }

    // return the type of the first argument
    return type.first().listExpr();
}
```

Type mapping for *de* is

(rel (xdata)) (rel (xdata)) -¿ (rel (xdata)) OR (rel (tuple (int text text))) -¿ (rel (tuple (int text text)))

```
ListExpr
deTypeMap( ListExpr args )
{
  NList type(args);
  if ( !type.hasLength(1) )
  {
    return NList::typeError("Expecting one argument.");
  }

  NList first = type.first();
  ////////////////////test for xdata/////////////////////////
  //Test argument symbols for xdata
  if (!first.hasLength(2) ||
     !first.first().isSymbol(REL) ||
     !first.second().hasLength(2) ||
          !first.second().first().isSymbol(TUPLE) ) {
          return NList::typeError("Argument is not 2 relations of tuples!");
  }

  //test xdata types
  ListExpr attrtype1 = nl->Empty();
  //browse  over the argument: '(rel (xdataValue xdata)'
  //get xdata
  attrtype1 = first.second().second().first().second().listExpr();
```

```
  //Return the type of the argument
  if ( nl->SymbolValue(attrtype1) == XDATA  ){
                return type.first().listExpr();
  }

  ////////////////////test for tuple//////////////////////////
  //Test argument symbols for tuple
  if (
    !first.hasLength(2) ||
    !first.first().isSymbol(REL) ||
    !first.second().hasLength(2) ||
    !first.second().first().isSymbol(TUPLE) ||
         !first.second().second().hasLength(3) ||
    !IsTupleDescription( first.second().second().listExpr() ) )
  {
    return NList::typeError("Error in the argument!");
  }

   //test int, text, text
   ListExpr attrtype11 = nl->Empty();
        ListExpr attrtype12 = nl->Empty();
        ListExpr attrtype13 = nl->Empty();

   //browse  over the argument:
   //'(rel (tuple (idData int metadata text value text)))'
   attrtype11 =
   first.second().second().first().second().listExpr();//get 'int'
   attrtype12 =
   first.second().second().second().second().listExpr();//get 'text'
   attrtype13 =
   first.second().second().third().second().listExpr();//get 'text'

  //Check types
  if ( nl->SymbolValue(attrtype11) != INT  )
  {
    return NList::typeError("Attribute type is not of type int.");
  }
  if ( nl->SymbolValue(attrtype12) != TEXT  )
  {
    return NList::typeError("Attribute type is not of type text.");
  }
  if ( nl->SymbolValue(attrtype13) != TEXT  )
  {
    return NList::typeError("Attribute type is not of type text.");
  }

  // return the type of the argument
  return type.first().listExpr();
}
```

4.2 Selection Function

A selection function is quite similar to a type mapping function. The only difference is that it doesn't return a type but the index of a value mapping function being able to deal with the respective combination of input parameter types.

Note that a selection function does not need to check the correctness of argument types; this has already been checked by the type mapping function. A selection function is only called if the type mapping was successful. This makes programming easier as one can rely on a correct structure of the list *args*.

4.2.1 The *dk* predicate

```
int
dkSelect( ListExpr args )
{
  NList type(args);
  if ( type.first().second().second().hasLength(3) )
    return 1;//type int, text, text
  else
    return 0;//type xdata
}
```

4.2.2 The *de* predicate

```
int
deSelect( ListExpr args )
{
  NList type(args);
  if ( type.first().second().second().hasLength(3) )
    return 1;//type int, text, text
  else
    return 0;//type xdata
}
```

4.3 Value Mapping Functions

4.3.1 The *showxdata* predicate

```
int
showXDataFun (Word* args, Word& result, int message,
              Word& local, Supplier s)
{
        XData* d = static_cast<XData*>( args[0].addr );
        result = qp->ResultStorage(s);
        XData* r = static_cast<XData*>( result.addr );
        r->Set(*d);
        return 0;
}
```

4.3.2 The *ds* predicate for two data relation

```
/*Convert from internal relation to external relation
I could do two different functions to each type
(xdata and "int, text, text").
In these case, these functions should be declared
when the operators are registered.*/
int
dsFun (Word* args, Word& result, int message,
             Word& local, Supplier s)

{
  //'rel' points to the address where the resut will be stored
  GenericRelation* rel = (Relation*)((qp->ResultStorage(s)).addr);
  //It cleans 'rel' to store a new result
  if(rel->GetNoTuples() > 0) { rel->Clear();}
  //Get the data and param
  Relation* r1 = (Relation*)args[0].addr;
  Relation* r2 = (Relation*)args[1].addr;
  //Convert param to string
  char* i1Param=
  "<tu><at1>0</at1><at2>parameter</at2><at3>true</at3></tu>";
  char* i2Param="<tu><at1>0</at1><at2>DS</at2><at3>true</at3></tu>";
  int ilen= strlen(i1Param)+strlen(i2Param)+1;
  char* iParam = (char*)malloc(ilen); bzero(iParam, ilen);
  strcat(iParam,i1Param);
  strcat(iParam,i2Param);
  char* eParam = dataToString(r1);
  int len= strlen(iParam)+strlen(eParam)+1;
  char* tParam = (char*)malloc(len); bzero(tParam, len);
  strcat(tParam,iParam);
  strcat(tParam,eParam);
  //Convert data to string
  char* iData =
  "<tu><at1>0</at1><at2>dataParameter</at2><at3>true</at3></tu>";
  char* eData = dataToString(r2);
  len= strlen(iData)+strlen(eData)+1;
  char* tData = (char*)malloc(len); bzero(tData, len);
  strcat(tData,iData);
  strcat(tData,eData);
  //Call CPN Processing
  char* receive = callCPN(tParam, tData);
  //Convert string to data
  rel = stringToData(receive, rel, r1);
  //store the result
  result = SetWord(rel);
  //Free memory
  free(tParam);

  return 0;
}
```

4.3.3 The *dp* predicate for two data relation

```
/*Convert from internal relation to external relation
I could do two different functions to each type
(xdata and "int, text, text").
In these case, these functions should be declared
when the operators are registered.*/
int
dpFun (Word* args, Word& result, int message,
             Word& local, Supplier s)

{
  //'rel' points to the address where the resut will be stored
  GenericRelation* rel = (Relation*)((qp->ResultStorage(s)).addr);
  //It cleans 'rel' to store a new result
  if(rel->GetNoTuples() > 0) { rel->Clear();}
  //Get the data and param
  Relation* r1 = (Relation*)args[0].addr;
  Relation* r2 = (Relation*)args[1].addr;
  //Convert param to string
  char* i1Param=
  "<tu><at1>0</at1><at2>parameter</at2><at3>true</at3></tu>";
  char* i2Param="<tu><at1>0</at1><at2>DP</at2><at3>true</at3></tu>";
  int ilen= strlen(i1Param)+strlen(i2Param)+1;
  char* iParam = (char*)malloc(ilen); bzero(iParam, ilen);
  strcat(iParam,i1Param);
  strcat(iParam,i2Param);
  char* eParam = dataToString(r1);
  int len= strlen(iParam)+strlen(eParam)+1;
  char* tParam = (char*)malloc(len); bzero(tParam, len);
  strcat(tParam,iParam);
  strcat(tParam,eParam);
  //Convert data to string
  char* iData =
  "<tu><at1>0</at1><at2>dataParameter</at2><at3>true</at3></tu>";
  char* eData = dataToString(r2);
  len= strlen(iData)+strlen(eData)+1;
  char* tData = (char*)malloc(len); bzero(tData, len);
  strcat(tData,iData);
  strcat(tData,eData);
  //Call CPN Processing
  char* receive = callCPN(tParam, tData);
  //Convert string to data
  rel = stringToData(receive, rel, r1);
  //store the result
  result = SetWord(rel);
  //Free memory
  free(tParam);

  return 0;
}
```

4.3.4 The *ad* predicate for two data relation

```
/*Convert from internal relation to external relation
I could do two different functions to each type
(xdata and "int, text, text").
In these case, these functions should be declared
when the operators are registered.*/
int
adFun (Word* args, Word& result, int message,
             Word& local, Supplier s)

{
  //'rel' points to the address where the resut will be stored
  GenericRelation* rel = (Relation*)((qp->ResultStorage(s)).addr);
  //It cleans 'rel' to store a new result
  if(rel->GetNoTuples() > 0) { rel->Clear();}
  //Get the data and param
  Relation* r1 = (Relation*)args[0].addr;
  Relation* r2 = (Relation*)args[1].addr;
  //Convert param to string
  char* i1Param=
  "<tu><at1>0</at1><at2>parameter</at2><at3>true</at3></tu>";
  char* i2Param="<tu><at1>0</at1><at2>AD</at2><at3>true</at3></tu>";
  int ilen= strlen(i1Param)+strlen(i2Param)+1;
  char* iParam = (char*)malloc(ilen); bzero(iParam, ilen);
  strcat(iParam,i1Param);
  strcat(iParam,i2Param);
  char* eParam = dataToString(r1);
  int len= strlen(iParam)+strlen(eParam)+1;
  char* tParam = (char*)malloc(len); bzero(tParam, len);
  strcat(tParam,iParam);
  strcat(tParam,eParam);
  //Convert data to string
  char* iData =
  "<tu><at1>0</at1><at2>dataParameter</at2><at3>true</at3></tu>";
  char* eData = dataToString(r2);
  len= strlen(iData)+strlen(eData)+1;
  char* tData = (char*)malloc(len); bzero(tData, len);
  strcat(tData,iData);
  strcat(tData,eData);
  //Call CPN Processing
  char* receive = callCPN(tParam, tData);
  //Convert string to data
  rel = stringToData(receive, rel, r1);
  //store the result
  result = SetWord(rel);
  //Free memory
  free(tParam);

  return 0;
}
```

4.3.5 The *bp* predicate for two data relation

```
/*Convert from internal relation to external relation
I could do two different functions to each type
(xdata and "int, text, text").
In these case, these functions should be declared
when the operators are registered.*/
int
bpFun (Word* args, Word& result, int message,
             Word& local, Supplier s)

{
  //'rel' points to the address where the resut will be stored
  GenericRelation* rel = (Relation*)((qp->ResultStorage(s)).addr);
  //It cleans 'rel' to store a new result
  if(rel->GetNoTuples() > 0) { rel->Clear();}
  //Get the data and param
  Relation* r1 = (Relation*)args[0].addr;
  Relation* r2 = (Relation*)args[1].addr;
  //Convert param to string
  char* i1Param=
  "<tu><at1>0</at1><at2>parameter</at2><at3>true</at3></tu>";
  char* i2Param="<tu><at1>0</at1><at2>BP</at2><at3>true</at3></tu>";
  int ilen= strlen(i1Param)+strlen(i2Param)+1;
  char* iParam = (char*)malloc(ilen); bzero(iParam, ilen);
  strcat(iParam,i1Param);
  strcat(iParam,i2Param);
  char* eParam = dataToString(r1);
  int len= strlen(iParam)+strlen(eParam)+1;
  char* tParam = (char*)malloc(len); bzero(tParam, len);
  strcat(tParam,iParam);
  strcat(tParam,eParam);
  //Convert data to string
  char* iData =
  "<tu><at1>0</at1><at2>dataParameter</at2><at3>true</at3></tu>";
  char* eData = dataToString(r2);
  len= strlen(iData)+strlen(eData)+1;
  char* tData = (char*)malloc(len); bzero(tData, len);
  strcat(tData,iData);
  strcat(tData,eData);
  //Call CPN Processing
  char* receive = callCPN(tParam, tData);
  //Convert string to data
  rel = stringToData(receive, rel, r1);
  //store the result
  result = SetWord(rel);
  //Free memory
  free(tParam);

  return 0;
}
```

4.3.6 The *bs* predicate for three data relation

```
/*Convert from internal relation to external relation
I could do two different functions to each type
(xdata and "int, text, text").
In these case, these functions should be declared
when the operators are registered.*/
int
bsFun (Word* args, Word& result, int message,
             Word& local, Supplier s)

{
  //'rel' points to the address where the resut will be stored
  GenericRelation* rel = (Relation*)((qp->ResultStorage(s)).addr);
  //It cleans 'rel' to store a new result
  if(rel->GetNoTuples() > 0) {rel->Clear();}
  //Get the data
  Relation* r1 = (Relation*)args[0].addr;
  Relation* r2 = (Relation*)args[1].addr;
  Relation* r3 = (Relation*)args[2].addr;
  //Convert param to string
  char* i1Param=
  "<tu><at1>0</at1><at2>parameter</at2><at3>true</at3></tu>";
  char* i2Param="<tu><at1>0</at1><at2>BS</at2><at3>true</at3></tu>";
  int ilen= strlen(i1Param)+strlen(i2Param)+1;
  char* iParam = (char*)malloc(ilen); bzero(iParam, ilen);
  strcat(iParam,i1Param);
  strcat(iParam,i2Param);
  char* eParam = dataToString(r1);
  int len= strlen(iParam)+strlen(eParam)+1;
  char* tParam = (char*)malloc(len); bzero(tParam, len);
  strcat(tParam,iParam);
  strcat(tParam,eParam);
  //Convert data(secondo parameter) to string
  char* iData1 = "<tu><at1>0</at1><at2>dataParameter2</at2><at3>true</at3></tu>";
  char* eData1 = dataToString(r2);
  len= strlen(iData1)+strlen(eData1)+1;
  char* tData1 = (char*)malloc(len); bzero(tData1, len);
  strcat(tData1,iData1);
  strcat(tData1,eData1);
  //Convert data(third parameter) to string
  char* iData2 =
  "<tu><at1>0</at1><at2>dataParameter</at2><at3>true</at3></tu>";
  char* eData2 = dataToString(r3);
  len= strlen(iData2)+strlen(eData2)+1;
  char* tData2 = (char*)malloc(len); bzero(tData2, len);
  strcat(tData2,iData2);
  strcat(tData2,eData2);
  //Call CPN Processing
  char* receive = callCPN(tParam, tData1, tData2);
  //Convert string to data
  rel = stringToData(receive, rel, r1);
  //store the result
```

```
  result = SetWord(rel);
  //Free memory
  free(tParam);

  return 0;
}
```

4.3.7 The *do* predicate for two data relation

```
/*Convert from internal relation to external relation
I could do two different functions to each type
(xdata and "int, text, text").
In these case, these functions should be declared
when the operators are registered.*/
int
doFun (Word* args, Word& result, int message,
             Word& local, Supplier s)

{
  //'rel' points to the address where the resut will be stored
  GenericRelation* rel = (Relation*)((qp->ResultStorage(s)).addr);
  //It cleans 'rel' to store a new result
  if(rel->GetNoTuples() > 0) { rel->Clear();}
  //Get the data and param
  Relation* r1 = (Relation*)args[0].addr;
  Relation* r2 = (Relation*)args[1].addr;
  //Convert param to string
  char* i1Param=
  "<tu><at1>0</at1><at2>parameter</at2><at3>true</at3></tu>";
  char* i2Param="<tu><at1>0</at1><at2>DO</at2><at3>true</at3></tu>";
  int ilen= strlen(i1Param)+strlen(i2Param)+1;
  char* iParam = (char*)malloc(ilen); bzero(iParam, ilen);
  strcat(iParam,i1Param);
  strcat(iParam,i2Param);
  char* eParam = dataToString(r1);
  int len= strlen(iParam)+strlen(eParam)+1;
  char* tParam = (char*)malloc(len); bzero(tParam, len);
  strcat(tParam,iParam);
  strcat(tParam,eParam);
  //Convert data to string
  char* iData =
  "<tu><at1>0</at1><at2>dataParameter</at2><at3>true</at3></tu>";
  char* eData = dataToString(r2);
  len= strlen(iData)+strlen(eData)+1;
  char* tData = (char*)malloc(len); bzero(tData, len);
  strcat(tData,iData);
  strcat(tData,eData);
  //Call CPN Processing
  char* receive = callCPN(tParam, tData);
  //Convert string to data
  rel = stringToData(receive, rel, r1);
  //store the result
```

```
  result = SetWord(rel);
  //Free memory
  free(tParam);

  return 0;
}
```

4.3.8 The *du* predicate for two data relation

```
/*Convert from internal relation to external relation
I could do two different functions to each type
(xdata and "int, text, text").
In these case, these functions should be declared
when the operators are registered.*/
int
duFun (Word* args, Word& result, int message,
             Word& local, Supplier s)

{
  //'rel' points to the address where the resut will be stored
  GenericRelation* rel = (Relation*)((qp->ResultStorage(s)).addr);
  //It cleans 'rel' to store a new result
  if(rel->GetNoTuples() > 0) { rel->Clear();}
  //Get the data and param
  Relation* r1 = (Relation*)args[0].addr;
  Relation* r2 = (Relation*)args[1].addr;
  //Convert param to string
  char* i1Param=
  "<tu><at1>0</at1><at2>parameter</at2><at3>true</at3></tu>";
  char* i2Param="<tu><at1>0</at1><at2>DU</at2><at3>true</at3></tu>";
  int ilen= strlen(i1Param)+strlen(i2Param)+1;
  char* iParam = (char*)malloc(ilen); bzero(iParam, ilen);
  strcat(iParam,i1Param);
  strcat(iParam,i2Param);
  char* eParam = dataToString(r1);
  int len= strlen(iParam)+strlen(eParam)+1;
  char* tParam = (char*)malloc(len); bzero(tParam, len);
  strcat(tParam,iParam);
  strcat(tParam,eParam);
  //Convert data to string
  char* iData =
  "<tu><at1>0</at1><at2>dataParameter</at2><at3>true</at3></tu>";
  char* eData = dataToString(r2);
  len= strlen(iData)+strlen(eData)+1;
  char* tData = (char*)malloc(len); bzero(tData, len);
  strcat(tData,iData);
  strcat(tData,eData);
  //Call CPN Processing
  char* receive = callCPN(tParam, tData);
  //Convert string to data
  rel = stringToData(receive, rel, r1);
  //store the result
```

```
  result = SetWord(rel);
  //Free memory
  free(tParam);

  return 0;
}
```

4.3.9 The *bi* predicate for two data relation

```
/*Convert from internal relation to external relation
I could do two different functions to each type
(xdata and "int, text, text").
In these case, these functions should be declared
when the operators are registered.*/
int
biFun (Word* args, Word& result, int message,
             Word& local, Supplier s)

{
  //'rel' points to the address where the resut will be stored
  GenericRelation* rel = (Relation*)((qp->ResultStorage(s)).addr);
  //It cleans 'rel' to store a new result
  if(rel->GetNoTuples() > 0) { rel->Clear();}
  //Get the data and param
  Relation* r1 = (Relation*)args[0].addr;
  Relation* r2 = (Relation*)args[1].addr;
  //Convert param to string
  char* i1Param=
  "<tu><at1>0</at1><at2>parameter</at2><at3>true</at3></tu>";
  char* i2Param="<tu><at1>0</at1><at2>BI</at2><at3>true</at3></tu>";
  int ilen= strlen(i1Param)+strlen(i2Param)+1;
  char* iParam = (char*)malloc(ilen); bzero(iParam, ilen);
  strcat(iParam,i1Param);
  strcat(iParam,i2Param);
  char* eParam = dataToString(r1);
  int len= strlen(iParam)+strlen(eParam)+1;
  char* tParam = (char*)malloc(len); bzero(tParam, len);
  strcat(tParam,iParam);
  strcat(tParam,eParam);
  //Convert data to string
  char* iData =
  "<tu><at1>0</at1><at2>dataParameter</at2><at3>true</at3></tu>";
  char* eData = dataToString(r2);
  len= strlen(iData)+strlen(eData)+1;
  char* tData = (char*)malloc(len); bzero(tData, len);
  strcat(tData,iData);
  strcat(tData,eData);
  //Call CPN Processing
  char* receive = callCPN(tParam, tData);
  //Convert string to data
  rel = stringToData(receive, rel, r1);
  //store the result
```

```
  result = SetWord(rel);
  //Free memory
  free(tParam);

  return 0;
}
```

4.3.10 The *di* predicate for two data relation

```
/*Convert from internal relation to external relation
I could do two different functions to each type
(xdata and "int, text, text").
In these case, these functions should be declared
when the operators are registered.*/
int
diFun (Word* args, Word& result, int message,
             Word& local, Supplier s)

{
  //'rel' points to the address where the resut will be stored
  GenericRelation* rel = (Relation*)((qp->ResultStorage(s)).addr);
  //It cleans 'rel' to store a new result
  if(rel->GetNoTuples() > 0) { rel->Clear();}
  //Get the data and param
  Relation* r1 = (Relation*)args[0].addr;
  Relation* r2 = (Relation*)args[1].addr;
  //Convert param to string
  char* i1Param=
  "<tu><at1>0</at1><at2>parameter</at2><at3>true</at3></tu>";
  char* i2Param="<tu><at1>0</at1><at2>DI</at2><at3>true</at3></tu>";
  int ilen= strlen(i1Param)+strlen(i2Param)+1;
  char* iParam = (char*)malloc(ilen); bzero(iParam, ilen);
  strcat(iParam,i1Param);
  strcat(iParam,i2Param);
  char* eParam = dataToString(r1);
  int len= strlen(iParam)+strlen(eParam)+1;
  char* tParam = (char*)malloc(len); bzero(tParam, len);
  strcat(tParam,iParam);
  strcat(tParam,eParam);
  //Convert data to string
  char* iData =
  "<tu><at1>0</at1><at2>dataParameter</at2><at3>true</at3></tu>";
  char* eData = dataToString(r2);
  len= strlen(iData)+strlen(eData)+1;
  char* tData = (char*)malloc(len); bzero(tData, len);
  strcat(tData,iData);
  strcat(tData,eData);
  //Call CPN Processing
  char* receive = callCPN(tParam, tData);
  //Convert string to data
  rel = stringToData(receive, rel, r1);
  //store the result
```

```
  result = SetWord(rel);
  //Free memory
  free(tParam);

  return 0;
}
```

4.3.11 The *rt* predicate for a data relation

```
/*Convert from internal relation to external relation
I could do two different functions to each type
(xdata and "int, text, text").
In these case, these functions should be declared
when the operators are registered.*/
int
rtFun (Word* args, Word& result, int message,
              Word& local, Supplier s)

{
  //'rel' points to the address where the resut will be stored
  GenericRelation* rel = (Relation*)((qp->ResultStorage(s)).addr);
  //It cleans 'rel' to store a new result
  if(rel->GetNoTuples() > 0) {rel->Clear();}
  //Get the param
  Relation* r1 = (Relation*)args[0].addr;
  //Convert param to string
  char* iParam =
  "<tu><at1>0</at1><at2>registerEventTemplate</at2><at3>true</at3></tu>";
  char* eParam = dataToString(r1);
  int len= strlen(iParam)+strlen(eParam)+1;
  char* tParam = (char*)malloc(len); bzero(tParam, len);
  strcat(tParam,iParam);
  strcat(tParam,eParam);
  //Call CPN Processing
  char* receive = callCPN(tParam);
  //Convert string to data
  rel = stringToData(receive, rel, r1);
  //store the result
  result = SetWord(rel);
  //Free memory
  free(tParam);

  return 0;
}
```

4.3.12 The *ut* predicate for a data relation

```
/*Convert from internal relation to external relation
I could do two different functions to each type
(xdata and "int, text, text").
In these case, these functions should be declared
when the operators are registered.*/
```

```
int
utFun (Word* args, Word& result, int message,
              Word& local, Supplier s)

{
  //'rel' points to the address where the resut will be stored
  GenericRelation* rel = (Relation*)((qp->ResultStorage(s)).addr);
  //It cleans 'rel' to store a new result
  if(rel->GetNoTuples() > 0) {rel->Clear();}
  //Get the param
  Relation* r1 = (Relation*)args[0].addr;
  //Convert param to string
  char* iParam =
  "<tu><at1>0</at1><at2>unregisterEventTemplate</at2><at3>true</at3></tu>";
  char* eParam = dataToString(r1);
  int len= strlen(iParam)+strlen(eParam)+1;
  char* tParam = (char*)malloc(len); bzero(tParam, len);
  strcat(tParam,iParam);
  strcat(tParam,eParam);
  //Call CPN Processing
  char* receive = callCPN(tParam);
  //Convert string to data
  rel = stringToData(receive, rel, r1);
  //store the result
  result = SetWord(rel);
  //Free memory
  free(tParam);

  return 0;
}
```

4.3.13 The *re* predicate for a data relation

```
/*Convert from internal relation to external relation
I could do two different functions to each type
(xdata and "int, text, text").
In these case, these functions should be declared
when the operators are registered.*/
int
reFun (Word* args, Word& result, int message,
              Word& local, Supplier s)

{
  //'rel' points to the address where the resut will be stored
  GenericRelation* rel = (Relation*)((qp->ResultStorage(s)).addr);
  //It cleans 'rel' to store a new result
  if(rel->GetNoTuples() > 0) {rel->Clear();}
  //Get the param
  Relation* r1 = (Relation*)args[0].addr;
  //Convert param to string
  char* iParam =
  "<tu><at1>0</at1><at2>registerEvent</at2><at3>true</at3></tu>";
```

```
  char* eParam = dataToString(r1);
  int len= strlen(iParam)+strlen(eParam)+1;
  char* tParam = (char*)malloc(len); bzero(tParam, len);
  strcat(tParam,iParam);
  strcat(tParam,eParam);
  //Call CPN Processing
  char* receive = callCPN(tParam);
  //Convert string to data
  rel = stringToData(receive, rel, r1);
  //store the result
  result = SetWord(rel);
  //Free memory
  free(tParam);

  return 0;
}
```

4.3.14 The *conc* predicate for two data relation

```
/*Convert from internal relation to external relation
I could do two different functions to each type
(xdata and "int, text, text").
In these case, these functions should be declared
when the operators are registered.*/
int
concFun (Word* args, Word& result, int message,
              Word& local, Supplier s)

{
  //'rel' points to the address where the resut will be stored
  GenericRelation* rel = (Relation*)((qp->ResultStorage(s)).addr);
  //It cleans 'rel' to store a new result
  if(rel->GetNoTuples() > 0) { rel->Clear();}
  //Get the data
  Relation* r1 = (Relation*)args[0].addr;
  Relation* r2 = (Relation*)args[1].addr;
  //Convert data to string
  char* eData1 = dataToString(r1);
  char* eData2 = dataToString(r2);
  //It concatenates tuples, deleting duplicates
  char* tData = concTuples(eData1, eData2);
  //Convert string to data
  rel = stringToData(tData, rel, r1);
  //store the result
  result = SetWord(rel);

  return 0;
}
```

4.4 Operator Descriptions

Similar to the *property* function of a type constructor, an operator needs to be described, e.g. for the *list operators* command. This is now done by creating a subclass of class *OperatorInfo*.

```
struct showXDataInfo : OperatorInfo {

  showXDataInfo()
  {
  name      = SHOWXDATA;
  signature = XDATA + " -> " + XDATA;
  syntax    = SHOWXDATA + "(_)";
  meaning   ="Show a specific Data.";
  }

}; // Don't forget the semicolon here. Otherwise the compiler
   // returns strange error messages

struct dsInfo : OperatorInfo {

  dsInfo()
  {
    name      = DS;
    signature =
    "(rel ( tuple (xdata))) x (rel ( tuple (xdata))) -> (rel ( tuple (xdata)))";
        // since this is an overloaded operator we append
        // an alternative signature here
//appendSignature("(rel ( tuple (int text text))) x
//(rel ( tuple (int text text))) -> (rel ( tuple (int text text)))");
    char* append1=
    "(rel ( tuple (int text text))) x (rel ( tuple (int text text)))";
    char* append2=
    " -> (rel ( tuple (int text text)))";
    int appendlen= strlen(append1)+strlen(append2)+1;
    char* append = (char*)malloc(appendlen); bzero(append, appendlen);
    strcat(append,append1);
    strcat(append,append2);
    appendSignature(append);
    syntax    = "_" + DS + "_";
    meaning   = "Discard Similar Data.";
  }

}; // Don't forget the semicolon here. Otherwise the compiler
   // returns strange error messages

struct dpInfo : OperatorInfo {

  dpInfo()
  {
    name      = DP;
    signature =
    "(rel ( tuple (xdata))) x (rel ( tuple (xdata))) -> (rel ( tuple (xdata)))";
        // since this is an overloaded operator we append
```

```
        // an alternative signature here
//appendSignature("(rel ( tuple (int text text))) x
//(rel ( tuple (int text text))) -> (rel ( tuple (int text text)))");
    char* append1=
    "(rel ( tuple (int text text))) x (rel ( tuple (int text text)))";
    char* append2=
    " -> (rel ( tuple (int text text)))";
    int appendlen= strlen(append1)+strlen(append2)+1;
    char* append = (char*)malloc(appendlen); bzero(append, appendlen);
    strcat(append,append1);
    strcat(append,append2);
    appendSignature(append);
    syntax    = "_" + DP + "_";
    meaning   ="Discard Prohibited Data.";
  }
}; // Don't forget the semicolon here. Otherwise the compiler
   // returns strange error messages

struct adInfo : OperatorInfo {

  adInfo()
  {
    name      = AD;
    signature =
    "(rel ( tuple (xdata))) x (rel ( tuple (xdata))) -> (rel ( tuple (xdata)))";
        // since this is an overloaded operator we append
        // an alternative signature here
//appendSignature("(rel ( tuple (int text text))) x
//(rel ( tuple (int text text))) -> (rel ( tuple (int text text)))");
    char* append1=
    "(rel ( tuple (int text text))) x (rel ( tuple (int text text)))";
    char* append2=
    " -> (rel ( tuple (int text text)))";
    int appendlen= strlen(append1)+strlen(append2)+1;
    char* append = (char*)malloc(appendlen); bzero(append, appendlen);
    strcat(append,append1);
    strcat(append,append2);
    appendSignature(append);
    syntax    = "_" + AD + "_";
    meaning   ="Discard Allowed Data.";
  }
}; // Don't forget the semicolon here. Otherwise the compiler
   // returns strange error messages

struct bpInfo : OperatorInfo {

  bpInfo()
  {
    name      = BP;
    signature =
    "(rel ( tuple (xdata))) x (rel ( tuple (xdata))) -> (rel ( tuple (xdata)))";
        // since this is an overloaded operator we append
        // an alternative signature here
```

```
//appendSignature("(rel ( tuple (int text text))) x
//(rel ( tuple (int text text))) -> (rel ( tuple (int text text)))");
    char* append1=
    "(rel ( tuple (int text text))) x (rel ( tuple (int text text)))";
    char* append2=
    " -> (rel ( tuple (int text text)))";
    int appendlen= strlen(append1)+strlen(append2)+1;
    char* append = (char*)malloc(appendlen); bzero(append, appendlen);
    strcat(append,append1);
    strcat(append,append2);
    appendSignature(append);
    syntax    = "_" + BP + "_";
    meaning   ="Blocking Prohibited Data.";
  }
}; // Don't forget the semicolon here. Otherwise the compiler
   // returns strange error messages

struct bsInfo : OperatorInfo {

  bsInfo()
  {
    name      = BS;
//signature =
//"(rel ( tuple (xdata))) x (rel ( tuple (xdata))) x
//(rel ( tuple (xdata))) -> (rel ( tuple (xdata)))";
    char* signature1=
    "(rel ( tuple (xdata))) x (rel ( tuple (xdata)))";
    char* signature2=
    " x (rel ( tuple (xdata))) -> (rel ( tuple (xdata)))";
    int signaturelen= strlen(signature1)+strlen(signature2)+1;
    char* sig = (char*)malloc(signaturelen); bzero(sig, signaturelen);
    strcat(sig,signature1);
    strcat(sig,signature2);
    signature = sig;
        // since this is an overloaded operator we append
        // an alternative signature here
//appendSignature("(rel ( tuple (int text text))) x
//(rel ( tuple (int text text))) x (rel ( tuple (int text text))) ->
//(rel ( tuple (int text text)))");
    char* append1=
    "(rel ( tuple (int text text))) x (rel ( tuple (int text text))) x";
    char* append2=
    " (rel ( tuple (int text text))) -> (rel ( tuple (int text text)))";
    int appendlen= strlen(append1)+strlen(append2)+1;
    char* append = (char*)malloc(appendlen); bzero(append, appendlen);
    strcat(append,append1);
    strcat(append,append2);
    appendSignature(append);
    syntax    = BS + "(_, _, _)";
    meaning   ="Blocking Similar Data.";
  }
}; // Don't forget the semicolon here. Otherwise the compiler
   // returns strange error messages
```

```
struct doInfo : OperatorInfo {

  doInfo()
  {
    name      = DO;
    signature =
    "(rel ( tuple (xdata))) x (rel ( tuple (xdata))) -> (rel ( tuple (xdata)))";
        // since this is an overloaded operator we append
        // an alternative signature here
//appendSignature("(rel ( tuple (int text text))) x
//(rel ( tuple (int text text))) -> (rel ( tuple (int text text)))");
    char* append1=
    "(rel ( tuple (int text text))) x (rel ( tuple (int text text)))";
    char* append2=
    " -> (rel ( tuple (int text text)))";
    int appendlen= strlen(append1)+strlen(append2)+1;
    char* append = (char*)malloc(appendlen); bzero(append, appendlen);
    strcat(append,append1);
    strcat(append,append2);
    appendSignature(append);
    syntax    = "_" + DO + "_";
    meaning   ="Discard Obsolete Data.";
  }
}; // Don't forget the semicolon here. Otherwise the compiler
   // returns strange error messages

struct duInfo : OperatorInfo {

  duInfo()
  {
    name      = DU;
    signature =
    "(rel ( tuple (xdata))) x (rel ( tuple (xdata))) -> (rel ( tuple (xdata)))";
        // since this is an overloaded operator we append
        // an alternative signature here
//appendSignature("(rel ( tuple (int text text))) x
//(rel ( tuple (int text text))) -> (rel ( tuple (int text text)))");
    char* append1=
    "(rel ( tuple (int text text))) x (rel ( tuple (int text text)))";
    char* append2=
    " -> (rel ( tuple (int text text)))";
    int appendlen= strlen(append1)+strlen(append2)+1;
    char* append = (char*)malloc(appendlen); bzero(append, appendlen);
    strcat(append,append1);
    strcat(append,append2);
    appendSignature(append);
    syntax    = "_" + DU + "_";
    meaning   ="Discard Useless Data.";
  }
}; // Don't forget the semicolon here. Otherwise the compiler
   // returns strange error messages
```

```
struct biInfo : OperatorInfo {

  biInfo()
  {
    name      = BI;
    signature =
    "(rel ( tuple (xdata))) x (rel ( tuple (xdata))) -> (rel ( tuple (xdata)))";
        // since this is an overloaded operator we append
        // an alternative signature here
//appendSignature("(rel ( tuple (int text text))) x
//(rel ( tuple (int text text))) -> (rel ( tuple (int text text)))");
    char* append1=
    "(rel ( tuple (int text text))) x (rel ( tuple (int text text)))";
    char* append2=
    " -> (rel ( tuple (int text text)))";
    int appendlen= strlen(append1)+strlen(append2)+1;
    char* append = (char*)malloc(appendlen); bzero(append, appendlen);
    strcat(append,append1);
    strcat(append,append2);
    appendSignature(append);
    syntax    = "_" + BI + "_";
    meaning   ="Blocking Irrelevant Data.";
  }
}; // Don't forget the semicolon here. Otherwise the compiler
   // returns strange error messages

struct diInfo : OperatorInfo {

  diInfo()
  {
    name      = DI;
    signature =
    "(rel ( tuple (xdata))) x (rel ( tuple (xdata))) -> (rel ( tuple (xdata)))";
        // since this is an overloaded operator we append
        // an alternative signature here
//appendSignature("(rel ( tuple (int text text))) x
//(rel ( tuple (int text text))) -> (rel ( tuple (int text text)))");
    char* append1=
    "(rel ( tuple (int text text))) x (rel ( tuple (int text text)))";
    char* append2=
    " -> (rel ( tuple (int text text)))";
    int appendlen= strlen(append1)+strlen(append2)+1;
    char* append = (char*)malloc(appendlen); bzero(append, appendlen);
    strcat(append,append1);
    strcat(append,append2);
    appendSignature(append);
    syntax    = "_" + DI + "_";
    meaning   ="Discard Irrelevant Data.";
  }
}; // Don't forget the semicolon here. Otherwise the compiler
   // returns strange error messages

struct rtInfo : OperatorInfo {
```

```
  rtInfo()
  {
    name      = RT;
    signature = "(rel ( tuple (xdata))) -> (rel ( tuple (xdata)))";
        // since this is an overloaded operator we append
        // an alternative signature here
//appendSignature("(rel ( tuple (int text text)))
// -> (rel ( tuple (int text text)))");
    char* append1=
    "(rel ( tuple (int text text)))";
    char* append2=
    " -> (rel ( tuple (int text text)))";
    int appendlen= strlen(append1)+strlen(append2)+1;
    char* append = (char*)malloc(appendlen); bzero(append, appendlen);
    strcat(append,append1);
    strcat(append,append2);
    appendSignature(append);
    syntax    = RT + "(_)";
    meaning   = "Register Event Template.";
  }

}; // Don't forget the semicolon here. Otherwise the compiler
   // returns strange error messages

struct utInfo : OperatorInfo {

  utInfo()
  {
    name      = UT;
    signature = "(rel ( tuple (xdata))) -> (rel ( tuple (xdata)))";
        // since this is an overloaded operator we append
        // an alternative signature here
//appendSignature("(rel ( tuple (int text text)))
// -> (rel ( tuple (int text text)))");
    char* append1=
    "(rel ( tuple (int text text)))";
    char* append2=
    " -> (rel ( tuple (int text text)))";
    int appendlen= strlen(append1)+strlen(append2)+1;
    char* append = (char*)malloc(appendlen); bzero(append, appendlen);
    strcat(append,append1);
    strcat(append,append2);
    appendSignature(append);
    syntax    = UT + "(_)";
    meaning   = "Unregister Event Template.";
  }

}; // Don't forget the semicolon here. Otherwise the compiler
   // returns strange error messages

struct reInfo : OperatorInfo {
```

```
  reInfo()
  {
    name      = RE;
    signature = "(rel ( tuple (xdata))) -> (rel ( tuple (xdata)))";
// since this is an overloaded operator we append
// an alternative signature here
//appendSignature("(rel ( tuple (int text text)))
// -> (rel ( tuple (int text text)))");
    char* append1=
    "(rel ( tuple (int text text)))";
    char* append2=
    " -> (rel ( tuple (int text text)))";
    int appendlen= strlen(append1)+strlen(append2)+1;
    char* append = (char*)malloc(appendlen); bzero(append, appendlen);
    strcat(append,append1);
    strcat(append,append2);
    appendSignature(append);
    syntax    = RE + "(_)";
    meaning   = "Register Event.";
  }

}; // Don't forget the semicolon here. Otherwise the compiler
   // returns strange error messages

struct concInfo : OperatorInfo {

  concInfo()
  {
    name      = CONC;
    signature =
    "(rel ( tuple (xdata))) x (rel ( tuple (xdata))) -> (rel ( tuple (xdata)))";
    // since this is an overloaded operator we append
    // an alternative signature here
    //appendSignature("(rel ( tuple (int text text))) x
    //(rel ( tuple (int text text))) -> (rel ( tuple (int text text)))");
    char* append1=
    "(rel ( tuple (int text text))) x (rel ( tuple (int text text)))";
    char* append2=
    " -> (rel ( tuple (int text text)))";
    int appendlen= strlen(append1)+strlen(append2)+1;
    char* append = (char*)malloc(appendlen); bzero(append, appendlen);
    strcat(append,append1);
    strcat(append,append2);
    appendSignature(append);
    syntax    = "_" + CONC + "_";
    meaning   = "Concatenation of xdata and relation int, text, text.";
  }

}; // Don't forget the semicolon here. Otherwise the compiler
   // returns strange error messages
```

5 Implementation of the Algebra Class

5.1 Class Definition

```
class DataKillingAlgebra : public Algebra
{
 public:
  DataKillingAlgebra() : Algebra()
  {
```

5.2 Registration of Types

```
    AddTypeConstructor( &xdataTC );
    //the lines below define that xdata
    //can be used in places where types of kind DATA are expected
    xdataTC.AssociateKind( "DATA" );
```

5.3 Registration of Operators

```
    AddOperator( showXDataInfo(), showXDataFun, showXDataTypeMap );
// the overloaded inside operator needs an array of function pointers
// which must be null terminated!
       ValueMapping dsFuns[] = { dsFun, dsFun, 0 };
         ValueMapping dpFuns[] = { dpFun, dpFun, 0 };
         ValueMapping adFuns[] = { adFun, adFun, 0 };
         ValueMapping bpFuns[] = { bpFun, bpFun, 0 };
         ValueMapping bsFuns[] = { bsFun, bsFun, 0 };
       ValueMapping doFuns[] = { doFun, doFun, 0 };
       ValueMapping duFuns[] = { duFun, duFun, 0 };
       ValueMapping biFuns[] = { biFun, biFun, 0 };
       ValueMapping diFuns[] = { diFun, diFun, 0 };
         AddOperator( dsInfo(), dsFuns, dkSelect, dkTypeMap );
         AddOperator( dpInfo(), dpFuns, dkSelect, dkTypeMap );
         AddOperator( adInfo(), adFuns, dkSelect, dkTypeMap );
         AddOperator( bpInfo(), bpFuns, dkSelect, dkTypeMap );
         AddOperator( bsInfo(), bsFuns, dkSelect, dk2TypeMap );
         AddOperator( doInfo(), doFuns, dkSelect, dkTypeMap );
         AddOperator( duInfo(), duFuns, dkSelect, dkTypeMap );
         AddOperator( biInfo(), biFuns, dkSelect, dkTypeMap );
         AddOperator( diInfo(), diFuns, dkSelect, dkTypeMap );

//the overloaded inside operator needs an array of function pointers
// which must be null terminated!
       ValueMapping rtFuns[] = { rtFun, rtFun, 0 };
         ValueMapping utFuns[] = { utFun, utFun, 0 };
         ValueMapping reFuns[] = { reFun, reFun, 0 };
         AddOperator( rtInfo(), rtFuns, deSelect, deTypeMap );
         AddOperator( utInfo(), utFuns, deSelect, deTypeMap );
         AddOperator( reInfo(), reFuns, deSelect, deTypeMap );

// the overloaded inside operator needs an array of function pointers
```

```
// which must be null terminated!
        ValueMapping concFuns[] = { concFun, concFun, 0 };
        AddOperator( concInfo(), concFuns, dkSelect, dkTypeMap );
  }
  ~DataKillingAlgebra() {};
};
```

6 Initialization

Each algebra module needs an initialization function. The algebra manager has a reference to this function if this algebra is included in the list of required algebras, thus forcing the linker to include this module.

The algebra manager invokes this function to get a reference to the instance of the algebra class and to provide references to the global nested list container (used to store constructor, type, operator and object information) and to the query processor.

The function has a C interface to make it possible to load the algebra dynamically at runtime (if it is built as a dynamic link library). The name of the initialization function defines the name of the algebra module. By convention it must start with `Initialize<AlgebraName>`.

To link the algebra together with the system you must create an entry in the file `makefile.algebra` and to define an algebra ID in the file `Algebras/Management/AlgebraList.i.cfg`.

```
} // end of namespace ~dkp~

extern "C"
Algebra*
InitializeDataKillingAlgebra( NestedList* nlRef,
                              QueryProcessor* qpRef )
{
  // The C++ scope-operator :: must be used to qualify the full name
  return new dkp::DataKillingAlgebra;
}
```

7 Examples and Tests

The file `DataKilling.examples` contains for every operator one example. This allows one to verify that the examples are running and to provide a coarse regression test for all algebra modules. The command `Selftest <file>` will execute the examples. Without any arguments, the examples for all active algebras are executed. This helps to detect side effects, if you have touched central parts of Secondo or existing types and operators.

In order to setup more comprehensive automated test procedures one can write a test specification for the *TestRunner* application. You will find the file `example.test` in directory `bin` and others in the directory `Tests/Testspecs`. There is also one for this algebra.

Accurate testing is often treated as an unpopular daunting task. But it is absolutely inevitable if you want to provide a reliable algebra module.

Try to write tests covering every signature of your operators and consider special cases, as undefined arguments, illegal argument values and critical argument value combinations, etc.

Anexo III – Instalação dos Componentes do Arcabouço Autonômico de Eliminação de Dados

O objetivo deste documento é descrever os passos necessários para:

- Inclusão da álgebra de eliminação de dados no SECONDO;
- Inclusão do *applet* contendo os padrões de eliminação de dados na ferramenta Briney Suite;
- Teste dos Padrões;
- Instalação do *plugin* e *servlet* Feed Organizer
- Execução do Arcabouço Autonômico de Eliminação de Dados (RPAN+SECONDO+Feed Organizer).

A pasta base para os arquivo fonte, descritos neste documento, é: ../Experimentos_SoftwaresDataKilling/. Neste documento, esta pasta é chamada simplesmente de ../$BASE/. Lá estão todos os arquivos descritos neste documento já devidamente atualizados.

A pasta base de instalação do SECONDO é: ../home/user/. Neste documento, esta pasta é chamada simplesmente de ../$USER/. A pasta ../home/user/ corresponde na realidade à pasta do usuário responsável pela instalação do SECONDO, devendo o nome *user* ser substituído pelo nome desse usuário.

1. Inclusão da álgebra de eliminação de dados no SECONDO

Os seguintes passos devem ser seguidos para inclusão da álgebra de eliminação de dados no SECONDO (todos os arquivos alterados no SECONDO ou pertencentes a álgebra estão localizados nas pastas ../$BASE/secondo/arquivosAlterados/ e ../$BASE /secondo/DataKilling/):

1. Instalar e configurar o banco de dados SECONDO[1], seguindo a documentação fornecida (baixar o guia de instalação do seu sistema operacional, o SDK correspondente e seguir as instruções de instalação). Neste trabalho, foi utilizado o arquivo Windows-Installation-Guide.pdf. Todas as instruções contidas neste arquivo devem ser seguidas, inclusive a execução do comando *make*. Uma cópia da dos documento de instalação pode ser encontrada na pasta ../$BASE/secondo/documentation e uma cópia do SDK de instalação para Windows pode ser encontrada na pasta ../$BASE/fonteSECONDO/;

2. Alterar os seguintes pontos no arquivo SecondoConfig.ini (localizado em ../$USER/secondo/bin/):

- Variável *SecondoHome=C:\msys\1.0\home\user\secondo-dbtest* (para windows, sendo que *user* deve ser trocado pelo usuário da instalação) ou *SecondoHome=/home/databases1* (para linux). Retirar o comentário referente somente a uma das distribuições;
- Variável *RTFlags += SMI:NoTransactions* (retirar o comentário, isto é, habilitar).

3. Acrescentar as seguintes linhas, antes da linha *endif*, no arquivo makefile.algebras (localizado em ../$USER/secondo/):

```
ALGEBRA_DIRS += DataKilling
ALGEBRAS     += DataKillingAlgebra
```

4. Acrescentar as seguintes linhas referentes as definições dos nomes dos tipos de dados e operadores da álgebra de eliminação, antes da linha *//stream examples types*, no arquivo Symbols.h (localizado em ../$USER/secondo/include/):

```
// data types
#undef XDATA
#undef SHOWXDATA
#undef DS
#undef DP
#undef AD
#undef BP
```

[1] http://dna.fernuni-hagen.de/Secondo.html/

#undef BS

#undef SD

#undef DO

#undef DU

#undef BI

#undef DI

#undef RT

#undef UT

#undef RE

#undef CONC

Sym XDATA("xdata");

Sym SHOWXDATA("showxdata");

Sym DS("ds");

Sym DP("dp");

Sym AD("ad");

Sym BP("bp");

Sym BS("bs");

Sym DO("do");

Sym DU("du");

Sym BI("bi");

Sym DI("di");

Sym RT("rt");

Sym UT("ut");

Sym RE("re");

Sym CONC("conc");

5. Acrescentar como última linha do arquivo AlgebraList.i.cfg (localizado em ../$USER/secondo/Algebras/Management/):

ALGEBRA_INCLUDE(52,DataKillingAlgebra)

6. Acrescentar os arquivos referentes à nova álgebra na pasta ../$USER/secondo/Algebras/DataKilling/ (Criar a pasta e adicionar os arquivos). Estes arquivos são:

- DataKillingAlgebra.cpp (contém o código referente a definição da estrutura e/ou funcionamento dos dados e operadores da álgebra);
- DataKillingAlgebra.dep (contém o endereço e nome dos arquivos necessários ao funcionamento da álgebra);
- DataKilling.examples (contém exemplos de uso dos operadores);
- DataKillingAlgebra.spec (definição do formato alternativo de entrada de dados e parâmetros para cada operador. O formato em lista aninhada *(query (OPERADOR DADO1 DADO2 ... DADO N))* está disponível como padrão para qualquer operador);
- makefile (arquivo que contém instruções para processamento e compilação dos arquivos da álgebra e dependências, descritos no arquivo DataKillingAlgebra.dep.);
- c_cpn.h (contém as declarações das classes de comunicação com a ferramenta CPN Tools);
- c_cpn.cpp (contém o código que define o comportamento das classes de comunicação com a ferramenta CPN Tools);
- c_cpn.dep (contém o endereço e nome dos arquivos necessários ao funcionamento das classes de comunicação).
- cpn.properties: arquivo que contém informações de configuração sobre endereços e portas de comunicação com a ferramenta CPN Tools.

Adicionalmente, deve se colocar o arquivo *dkptest* dentro da pasta .../$USER/secondo/bin/. Este arquivo contém um conjunto de dados de teste a serem utilizados nos operadores de eliminação de dados e está localizado na pasta ../$BASE/Secondo/baseTest/.

Finalmente, como último passo para inclusão da álgebra no SECONDO, execute o comando *make* na para .../$USER/secondo. Uma série de mensagens aparecerá e, ao final, se não houver erros, a álgebra estará instalada.

2. Inclusão do Applet contendo os Padrões de Eliminação de Dados na Ferramenta Briney Suite

A ferramenta Britney Suite permite executar através de um *applet* um simulador da rede de Petri criado a partir da ferramenta CPN Tools. Para isso, ela utiliza um arquivo correspondente a rede de Petri a ser simulada (arquivo dataKillingAndEventPatterns.x86-win32), um arquivo contendo instruções básicas de carga do simulador (script em java-like contido no arquivo dataKillingAndEventPatterns.txt) e um arquivo de inicialização do applet (arquivo webstart loader). Dessa forma, várias instâncias da rede de Petri podem ser executadas via *Web*. Além disso, o arquivo de script permite que uma rede de Petri seja executada com número infinito de disparos, característica não disponível na ferramenta CPN Tools, cujo número máximo de disparos é 999.999.999.

Os seguintes passos devem ser seguidos para inclusão do Applet contendo os Padrões de eliminação de dados na ferramenta Britney Suite[2] (todos os arquivos alterados no SECONDO ou pertencentes a álgebra estão localizados nas pastas ../$BASE/secondo/arquivosAlterados e ../$BASE/secondo/DataKilling):

1. Fazer o download da ferramenta Britney Suite[3]. Uma cópia dessa ferramenta pode ser encontrada na pasta ../$BASE/Britney/;

2. Descompactar o arquivo do Britney Suite em uma pasta (ex.: pasta c:\Britney\);

3. Salvar um simulador para a rede com extensão .cpn, criada na ferramenta CPN Tools, como a seguir:

   - Abrir o Britney Suite (Clicar no arquivo BRITNeY.jar);
   - Carregar a rede com extensão cpn na ferramenta CPN Tools;
   - Certificar-se que a rede está no estado inicial;
   - Localizar no Britney o console do simulador que corresponde à rede carregada na ferramenta CPN Tools;
   - Clicar com o botão direito do mouse sobre o console simulador;

[2] http://wiki.daimi.au.dk/britney/britney.wiki
[3] http://www.daimi.au.dk/~mw/local/tincpn/offline.zip

- Selecionar "*Save simulator as"* no menu que aparece;
- Selecionar um nome adequado e coloque o simulador dentro da pasta Britney (ex.: arquivo dataKillingAndEventPatterns.x86-win32).

4. Criar o arquivo para carregar e iniciar o simulador (ex.:dataKillingAndEventPatterns.txt) e salvar na pasta Britney. O código deste arquivo segue uma sintaxe java-like, descrita na documentação do Britney Suite. O código a seguir inicia o simulador, permitindo que a rede execute infinitos disparos:

```
s = SimulatorService.getNewSimulator(new
java.net.URL("http://localhost/Britney/dataKillingAndEventPatterns.x86-
win32"));
hs = s.getHighLevelSimulator();
hs.initialState();
oldsteps = Play.steps;
olddelay = Play.delay;
Play.steps=1;
Play.delay=1;
int i=1;
while (i>0){
    Play.steps=i;
    i++;
    event = new java.awt.event.ActionEvent(hs, 0, "");
    Play.execute(event);}
Play.steps = oldsteps;
Play.delay = olddelay;
```

5. No arquivo existente na pasta de instalação do Britney, caminho: Britney/webstart/Britney.jnlp, alterar a linha referente ao endereço do *applet* (http:://www.daimi.au.dk/~mw/local/tincpn/) para o local onde será feita a implantação do simulador (ex.:http://localhost/Britney/);

6. Criar uma página html que aponte para o arquivo britney.jnlp no Servidor *Web*. Um exemplo de código é mostrado a seguir:

```
<html>
```

```
<head>
  <title>Test</title>
<meta name="collection" content="exclude">
  </head>
 <body>
    <h1>Britney</h1>
    <hr>
    <a
href="http://localhost/Britney/BRITNeY.jnlp">http://localhost/Britney/BRITNeY.
jnlp</a></p>
    <hr>
 </body>
</html>
```

7. Copiar a pasta Britney para o Servidor *Web* desejado (ex.: C:\Program Files\Apache Software Foundation\Apache2.2\htdocs\);

8. Execute o arquivo CacheCleanup.jar, dentro da pasta Britney, para limpar as configurações de simulação no Britney Suite.

9. Abrir um navegador *Web* no endereço escolhido para o *applet* e executar a simulação. Neste caso, deve-se copiar e colar o código criado no arquivo de carga e inicialização do simulador (ex.:dataKillingAndEventPatterns.txt) na aba *script console* da ferramenta Britney. Caso não se deseje executar o *applet* via *Web*, a ferramenta Britney permite a execução do simulador localmente, bastando abrir a ferramenta localmente e executar o arquivo de simulação (ex.:dataKillingAndEventPatterns.txt) como na ferramenta via *Web*.

3. Teste dos Padrões

Com o objetivo de testar a correta instalação das ferramentas descritas até agora, os seguintes testes devem ser executados.

Para execução dos testes usando a interface Msys (instalada com o SECONDO), os seguintes passos devem ser seguidos:

1. Abrir a ferramenta Britney ou CPN Tools e executar a rede desejada;

2. Executar o programa em java que realiza o *stemming* e *stopwords* (programa TCPJavaServer_fat.jar, localizado em ../$BASE/);

3. Na interface Msys, navegar até a pasta "/USER/secondo/bin" e executar o comando *Selftest tmp/DataKilling.examples*. Aparecerão várias mensagens mostrando o que o arquivo DataKilling.examples manda os operadores fazerem.

Também é possível abrir a interface gráfica do SECONDO e fazer testes com os operadores. Para isso, os seguintes passos adicionais devem ser executados:

1. Realizar a chamada do Secondo Monitor:

   - Abrir o Msys;
   - Navegar até a pasta "$USER/secondo/bin"
   - Executar o comando "SecondoMonitor -s" no prompt do Msys.

2. Abrir a interface gráfica do SECONDO, realizando a chamada através do *prompt* do DOS do arquivo "$USER/secondo/Javagui/sgui.bat". Abrirá uma interface que permite usar os padrões de eliminação de dados como operadores.

4. Instalação do Plugin e Servlet Feed Organizer

Uma das aplicações que pode ser incorporada ao arcabouço autonômico de eliminação de dados é a aplicação que manuseia dados de notícias *Web*, eliminando as notícias consideradas menos relevantes. Essa aplicação é chamada Feed Organizer. Ela é composta de um *plugin* para o navegador Firefox, responsável por fazer a interface com os usuários, e por um *servlet*, responsável por obter as requisições dos usuários e fazer a interface com o banco de dados SECONDO.

As seguintes pastas devem ser incluídas no SECONDO (.../$USER/secondo/Javagui/gui) de modo a permitir o funcionamento do Feed Organizer:

1. ../$BASE/Secondo/feedOrganizer/alteracaoSecondo: esta pasta contém cinco arquivos que devem ser incluídos no SECONDO e um arquivo que deve substituir um arquivo já existente. Estes arquivos são:

- comunicacaoFactory.java: cria as classes de comunicação do SECONDO com o servlet usando sockets;
- comunicacao.java: contém as classes de comunicação;
- EncodeDecode.java: converte formato dos dados de *string* para *bytes* e vice-versa;
- PollServidor.java: responsável por controlar o fluxo de dados que utilizam as classes de comunicação através do uso de threads;
- MainWindow.java: esta classe já existe no SECONDO. Este arquivo foi alterado de modo a permitir que o SECONDO receba comandos emitidos por classes do arquivo PollServidor.java em vez de receber comando da interface gráfica do SECONDO;
- Informacoes.java: esta classe permite ler os dados do arquivo de configuração dataKilling.properties;
- secondo.properties: arquivo que contém informações de configuração sobre endereços e portas de comunicação com o SECONDO.

2. ../$BASE/feedOrganizer/pluginServletVersao2/pluginFirefox: esta pasta contém as páginas que são apresentadas quando o plugin Feed Organizer é acionado no Firefox, além dos arquivos necessários a criação de um *plugin* para esse navegador[4].

3. ../$BASE/feedOrganizer/pluginServletVersao2/Servlet; esta pasta contém o *servlet* (.war) da aplicação que roda num servidor *Web* (por exemplo, o JBOSS[5]).

Para instalação do aplicativo, os seguintes passos devem ser seguidos:

1. Copiar os arquivos do diretório ../$BASE/secondo/feedOrganizer/alteracaoSecondo/ para o diretório ../$USER/secondo/Javagui/gui. Antes, deve ser feita uma cópia de segurança do arquivo MainWindow.java que já existe nesta pasta. Ao realizar a cópia destes arquivos, a interface gráfica do SECONDO não poderá mais ser aberta. Case seja necessário abrir novamente

[4] https://developer.mozilla.org/en/Extensions
[5] http://www.jboss.org/

esta interface, basta substituir o arquivo MainWindow.java pela cópia de segurança do arquivo original (e executar o comando *make*);

2. Abrir o Msys, navegar até o diretório $BASE/secondo/Javagui/gui e executar o comando *make* (com isso as alterações feitas no SECONDO passarão a ser válidas);

3. Instalar o JBOSS (cópia na pasta ../$BASE/JBOSS/ da versão usada neste trabalho: JBOSS4.2.1 GA). A pasta base do JBOSS será chamada de ../$JBOSS (ex.: C:\jboss-4.2.1.GA\);

4. Copiar os arquivos FeedOrganizer.war e feedorganizer.properties (localizados na ../$BASE/feedOrganizer/pluginServletVersao2/servlet/) para as pastas ../$JBOSS/deploy/ e ../$JBOSS/conf/, respectivamente.

5. Criar um atalho para o JBOSS na área do Desktop para o run.bat do JBOSS (ex.: start C:\jboss-4.2.1.GA\bin\ e target C:\jboss-4.2.1.GA\bin\run.bat -b 0.0.0.0). A diretiva –b 0.0.0.0 permite o acesso de outros computadores ao JBOSS;

6. Iniciar o JBOSS com o atalho criado;

7. Iniciar a instalação do plugin. Abra o arquivo ../$BASE/feedOrganizer/pluginServletVersao2/pluginFirefox/FeedOrganizer/chrome/content/FeedOrganizer.html e altere o endereço "http://localhost:8090/FeedOrganizer/RecebeDados" para o endereço onde pode está sendo executado o servlet FeedOrnganizer.war no servidor JBOSS. Crie um arquivo zip com a extensão xpi (ex.: FeedOrganizer.xpi) do conteúdo da pasta ../$BASE/feedOrganizer/pluginServletVersao2/pluginFirefox/FeedOrganizer/ (a pasta não deve ser zipada, somente o seu conteúdo).

8. Clicar duas vezes no arquivo criado (o arquivo .xpi é um arquivo .zip que instala o *plugin* no Firefox);

9. A pasta C:\FeedOrganizer é criada. É nela que são colocadas as configurações da ferramenta Feed Organizer, as configurações das regras e arquivos *rss* escolhidos pelos usuários.

5. Execução do Arcabouço Autonômico de Eliminação de Dados

Para execução do arcabouço, os seguintes passos devem ser seguidos:

1. Abrir a ferramenta Britney ou CPN Tools e executar a rede desejada;

2. Executar o programa em java que realiza o *stemming* e *stopwords* (programa TCPJavaServer_fat.jar);

3. Realizar a chamada do Secondo Monitor:

   - Abrir o Msys;
   - Navegar até a pasta "$USER/secondo/bin"
   - Executar o comando "SecondoMonitor -s" no prompt do Msys.

4. Iniciar o JBOSS através do arquivo run.bat dentro da pasta ...\$JBOSS\bin\.

5. Plugin do firefox (abra o Firefox e inicie o *plugin* que foi instalado).

É importante considerar que os componentes das diferentes camadas do arcabouço de eliminação de dados se comunicam via *sockets*, sendo possível executá-los em diferentes máquinas e endereços *Web*. As seguintes configurações padrão são adotadas: a ferramenta CPN Tools provê comunicação com os padrões de eliminação de dados pela porta 9000, atuando como servidor. Ao mesmo tempo, ela comunica a detecção de eventos compostos na porta 9050, atuando como um cliente (a localização do servidor deve ser fornecida no arquivo cpn.properties, descrito anteriormente, sendo que o valor padrão está configurado para *default localhost : 127.0.0.1*). O SECONDO provê comunicação com os operadores pela porta 7000, atuando como servidor. Ao mesmo tempo comunica-se com a ferramenta CPN Tools na porta 9000, atuando como cliente (a localização do servidor deve ser fornecida no arquivo secondo.properties, descrito anteriormente, sendo que o valor padrão está configurado para *default localhost : 127.0.0.1*). A ferramenta Feed Organizer comunica-se com o SECONDO pela porta 7000, atuando como cliente (a localização do servidor deve ser fornecida no arquivo feedorganizer.properties, apresentado anteriormente, sendo que o valor padrão está configurado para *default localhost : 127.0.0.1*).

Anexo IV – Questionários Utilizados nos Experimentos com a Versão Alfa da Ferramenta

Questionário 1

Autorização

Eu, _____, autorizo a utilização dos dados deste documento contendo 3 (três) folhas (Questionário 1) e 6 (seis) folhas (Questionário 2) para fins de pesquisa, análise e posterior publicação dos resultados. Ressalto que todas as páginas estão devidamente rubricadas.

Rio, ___ de outubro de 2008.

Assinatura:_____

1ª etapa

1.1 – Qual a sua experiência com RSS?

( ) A – Nunca usei
( ) B – Conheço mas não uso freqüentemente
( ) C – Utilizo com freqüência, mas assino poucos feeds
( ) D – Utilizo com freqüência, e assino vários feeds

Atenção: As perguntas de 1.2 a 1.4 só devem ser respondidas se você NÃO marcou **A** na pergunta acima.

1.2 – Como você escolhe o uso dos feeds?

( ) A – Vejo feeds de notícias gerais. Ex.: ultimosegundo.com.br
( ) B – Assino feeds específicos por assunto.
( ) C – Assino feeds de vários sites indiscriminadamente

1.3 – Dê a sua nota para o nível de similaridade (o quão parecido) às noticias que recebe. Considere os últimos quinze dias e que as notas variam de nota 0 (para nenhuma notícia repetida ou parecida) a nota 10 (para mais de dez notícias parecidas).

(0) (1) (2) (3) (4) (5) (6) (7) (8) (9) (10)

1.4- Quais os agregadores (clientes) de feeds que você utiliza? Ex: Mozilla Firefox, Internet Explorer 7, FeedGhost, Juice, Akregator, Vienna, etc. Assinale também como vc classifica esses agregadores (clientes) de feeds: as notas variam de nota 0 (para pobres, que não fornecem opções) a nota 10 (para os completos, que fornecem muitas opções).

_____

_____

2ª etapa

2.1 – Operador Dados Permitidos

2.1.1 – Quais são os tópicos (palavras-chave) de seu interesse? Ou seja, aqueles que você gostaria de ver nos feeds apresentados, dentro dos seguintes assuntos:

-Business:

_____

_____

-Esportes:

_____

_____

-United States:

_____

_____

2.1.2 – Qual o nível de similaridade que você atribuiria entre as palavras que você escolheu no item anterior e as notícias que você gostaria de ler? (1 uma ocorrência das palavras-chave dentro das notícias e 10 todas as palavras-chaves precisam aparecer na notícia)

(1) (2) (3) (4) (5) (6) (7) (8) (9) (10)

2.1.3 – Alternativamente, você prefere fazer as consultas usando OR ou AND para combinar as palavras-chave?

( ) OR
( ) AND

2.2 – Operador Eliminador de Dados Proibidos

2.2.1 – Relacione os tópicos (palavras-chave) que você não gostaria de encontrar (ou seja, aquelas notícias com determinados tipos de palavras que você gostaria de descartar dos feeds apresentados), dentro dos seguintes assuntos:

-Business

_____

_____

-Esportes

_____

_____

-United States

_____

_____

2.2.2 – Qual o nível de similaridade que você escolheria entre as palavras que você escolheu no item anterior e as notícias que você NÃO gostaria de ler? (1 para uma ocorrência das palavras-chave dentro das notícias e 10 todas as palavras-chaves precisam aparecer na notícia)

(1) (2) (3) (4) (5) (6) (7) (8) (9) (10)

2.2.3 – Alternativamente, neste caso você prefere fazer as consultas usando OR ou AND para combinar as palavras-chave?

( ) OR
( ) AND

3ª etapa (Utilize o Questionário 2 como apoio)
3.1 – Atribua, dentro de cada assunto, um nível de similaridade entre a notícia padrão e as outras notícias. As notas variam de 0 (para nenhuma notícia repetida ou parecida) a 10 (para mais de dez notícias parecidas).

3.2 – Indique quais notícias você considera com um alto grau de similaridade (parecidas) entre si. Ex. [1-10-20] significa que as notícias com ID = 1, 10 e 20 são muito similares entre si, independentemente da notícia padrão.

3.3 – Ordene as notícias fornecidas de acordo com as palavras-chave que você escolheu na 2ª etapa (primeiro as notícias com palavras-chave que você gostaria de encontrar e, por último, notícias com palavras-chave que você NÃO gostaria de encontrar).

QUESTIONÁRIO 2
ASSUNTO: BUSINESS

Notícia padrão: Bush Presses House to Pass Rescue Package

The morning after the Senate strongly endorsed the bailout plan, President Bush tried to muster support for the $700 billion package ahead of a crucial vote in the House.

ID	Notícia	nível de similaridade com a notícia padrão	Notícias Similares	Ordem de preferência de acordo com as palavras-chave
1	**Credit Crisis Spreads a Pall Over Silicon Valley** High-tech entrepreneurs, investors and executives now believe the question is when, not if, the financial chaos will hurt the country's cradle of innovation.			
2	**European Officials Debate Need for a Bailout Package** Surprised by the exposure of several banks to the global financial crisis, European governments are discussing new rules for lenders that could change the regulatory landscape for European banking.			
3	**Your Money: A Bill Encouraging to Distressed Homeowners, but Its Reach Is Unclear** The bailout 700 billion package should ease tight credit and hold down interest rates. But it does not go as far as some Democrats would have liked in helping distressed homeowners.			
4	**Falling Oil Price Is a Positive Note Amid Turmoil** If oil prices continue to drop, it will put billions of dollars back into consumers' wallets and provide badly needed support for a battered economy.			
5	**A Curious Coalition Opposed Bailout Bill** The 25 senators who voted against the $700 billion financial rescue plan are one of the most curious coalitions of lawmakers ever to share common ground.			
6	**The Reckoning: As Credit Crisis Spiraled, Alarm Led to Action** During a 36-hour period two weeks ago, the fissures opening in the worldwide financial system convinced policy makers that they needed to act quickly.			
7	**U.S. Stocks Fall Sharply on Credit Worries** The Dow Jones industrial average closed down 348.22 points, with the worst declines coming among companies in the manufacturing, chemical production and mining industries.			
8	**European Central Bank Chief Hints at Rate Cut** The benchmark interest rate was left at 4.25 percent, and markets are now awaiting President Jean-Claude Trichet's assessment of the rate outlook.			
9	**Thain to Take Role at Bank of America** Merrill Lynch's chief, John A. Thain, will essentially retain control of the businesses run by Merrill in the combined company.			
10	**Stocks Dip as Investors Await Bailout Vote** NEW YORK, Oct. 1 -- U.S. stocks flirted with gains Wednesday but finished lower as investors waited for Congress to decide the fate of the $700 billion bailout proposal and considered the economic threats that might remain even if the legislation passed.			
11	**Senate Approves Bailout** The Senate last night easily approved a massive plan to shore up the U.S. financial system, but the measure faces a tougher test tomorrow in the House, where leaders will try to reverse the stunning defeat the legislation suffered earlier this week.			
12	**Bush Calls on House to 'Get This Bill Passed'** A day after the Senate easily approved a massive plan to shore up the U.S. financial system, President Bush today called for the House to "get this bill passed so we can get about the business of restoring confidence" in the American economy.			
13	**Wind and Solar Tax Credits Could Ride Into Law in Bailout Bill** Maybe the tenth time will be a charm. Last night was the tenth time since June 2007 that an extension of wind and solar tax credits have gone to the floor of the Senate. Seven times they have been stuck in bills that have gone to the floor of the House of Representatives.			
14	**Markets Decline on Poor Economic Reports** Stocks plummeted today as investors turned their attention from the financial rescue plan begin debated in Congress to grim economic data and the continuing credit crisis.			
15	**A Government Workforce in Need of Its Own Rescue** A throng of good-government types packed a National Press Club meeting room yesterday, determined to help smooth the transition from President Bush's administration to whatever comes next.			
16	**Stocks Dip as Investors Await Bailout Vote**			

	NEW YORK, Oct. 1 -- U.S. stocks flirted with gains Wednesday but finished lower as investors waited for Congress to decide the fate of the $700 billion bailout proposal and considered the economic threats that might remain even if the legislation passed.			
17	**Obama, McCain Stand United In Pressing Hard for Rescue** After months on the campaign trail and countless missed votes, Barack Obama and John McCain returned to the Capitol last night as just two of 99 senators voting on a massive Wall Street rescue plan, but their forceful advocacy for the controversial measure may help push it into law.			
18	**Auto Sales Fall 27% As Credit Tightens** Americans steered clear of auto dealerships in September, sending sales of new cars and trucks tumbling as credit conditions tightened.			
19	**A Two-Pronged Push To Aid Ailing Banks** Two federal agencies moved yesterday to ease the financial pressure on banks even as Congress continued to debate the wisdom of a broader intervention.			
20	**Japan's Carmakers Hit Hard As Exports to U.S. Plummet** TOKYO, Oct. 1 -- Sales of Japanese cars in the United States have increased like clockwork, rising every month for more than two years.			
21	**A shift in tone in calls to House members on bailout** Stock market drop after the rejection by the House brings a response from constituents.			
22	**U.S. Senate passes bailout plan; House to vote Friday** The Senate strongly endorsed the $700 billion bailout plan, leaving backers hopeful that the easy approval, coupled with an array of popular additions, would lead to quick passage in the House.			
23	**In bailout, still room for doubt** Added pork and an injection of private capital into the market could affect passage in the House.			
24	**36 hours of alarm and action as crisis spiraled** It was a 36-hour period two weeks ago - from the morning of Wednesday, Sept. 17, to the afternoon of Thursday, Sept. 18 - that spooked policy makers as it opened fissures in the worldwide financial system.			
25	**A shift in tone in calls to House members on bailout** Stock market drop after the rejection by the House brings a response from constituents.			
26	**Worried consumers return to Northern Rock** Northern Rock is now seen as the safest home for savings because its deposits are fully guaranteed by the British government.			
27	**E.ON Ruhrgas gets stake in Russian field** Under the accord, the German company will reduce its minority stake in Gazprom.			
28	**Merrill Lynch chief to remain at Bank of America** The announcement settles a big question circulating around Wall Street: What was next for John Thain, a veteran of Goldman Sachs who led the New York Stock Exchange before joining Merrill Lynch last fall?			
29	**Europeans weigh a shift in banking landscape** European governments have begun discussing a far-reaching overhaul of rules for lenders that could change the regulatory landscape for Europe's banking sector.			
30	**Buffett's bet on GE: Almost as good as a bailout** The billionaire investor Warren Buffet has announced he would invest $3 billion in General Electric, eight days after he said he would invest $5 billion in Goldman Sachs.			

ASSUNTO: SPORTS

Notícia padrão: Sports of The Times: In Baseball's Scheme of Things, 26 Years Isn't a Long Wait at All

Will either the Cubs or the Bewers join the Red Sox, White Sox and Angels in breaking a long spell without success in the postseason?

ID	Notícia	nível de similaridade com a notícia padrão	Notícias Similares	Ordem de preferência de acordo com as palavras-chave
1	**Baseball Postseason Analysis: In Chicago, Mood Swings All Over the Toddlin' Town** It's been 102 years since both of Chicago's baseball teams have been in the postseason. And the notion of a Cubs-White Sox World series — however unlikely — has generated rare, bilateral optimism in the fall air.			
2	**Mets Give Minaya a New Contract** The Mets extended the contract of general manager Omar Minaya through 2012 and included team options for 2013 and 2014. The team will now turn its attention to the manager, Jerry Manuel.			
3	**Ruling Backs Chinese Gymnasts** All six members of the Chinese women's gymnastics team met the age requirements to compete at the Beijing Olympics, officials from the International Gymnastics Federations said Wednesday.			
4	**Santana Pitched With Injury to Knee** Mets' Johan Santana, it turns out, pitched that three-hit shutout against Florida last Saturday with a torn meniscus in his left knee.			
5	**Cashman Cites Legacy in Coming Back** Brian Cashman told reporters uncharacteristically tersely and directly that he gave serious consideration to stepping away from a job he has held for a decade.			
6	**Dodgers 7, Cubs 2: Cubs Are Quickly Against the Wall** Taking advantage of seven walks by Cubs starter Ryan Dempster, the Dodgers erupted, hitting three home runs and seizing the momentum in baseball's lightning round.			
7	**Baseball Postseason Analysis: Dodgers and Angels, California's Yin and Yang** The free-spirited Manny Ramirez and the clean-cut Mark Teixeira each embody their new organizations.			
8	**Angels 7, Rangers 0** The Los Angeles Angels posted their team-record 100th win, with Joe Saunders pitching six sharp innings and Mike Napoli homering Sunday in a 7-0 win over the Texas Rangers.			
9	**Caution Signs for Giants Coming Off a Bye Week** The Giants, the defending Super Bowl champions,will enter this week having played only three games in 37 days after their preseason ended on Aug. 28.			
10	**Milt Davis, a Cornerback on 2 Title-Winning Teams, Dies at 79** Mr. Davis was an All-Pro defensive back who helped the Baltimore Colts win two National Football League championships in the 1950s.			
11	**Rays Leave Closer Troy Percival off ALDS Roster** ST. PETERSBURG, Fla. -- Tampa Bay closer Troy Percival was left off the Rays' roster for their AL division series against the Chicago White Sox due to back tightness.			
12	**Leg Injury to Sideline Red Wings' Chelios** DETROIT -- Red Wings defenseman Chris Chelios will be sidelined three to six weeks with a fractured shin bone.			
13	**GM Minaya Given Deal Through 2012 by Mets** NEW YORK -- The New York Mets have given general manager Omar Minaya a new contract that runs through the 2012 season.			
14	**A Humbled but Hopeful Grove Heads Into Maryland Million** After being humbled at last year's Maryland Million, trainer Chris Grove is noticeably more low-key despite having seven horses entered.			
15	**Smith, Shock Top Silver Stars** SAN ANTONIO, Oct. 1 -- Someone finally figured out how to stop Deanna Nolan. Katie Smith and Taj McWilliams-Franklin were more than enough to help the Detroit Shock make up for it.			
16	**Patriots Deny Charge of Tampering** Bill Belichick denies an accusation by Raiders owner Al Davis that the Patriots had improper contact with Randy Moss before trading for him in 2007.			
17	**Chinese Olympic Gymnasts Cleared**			

	After reviewing documents supplied by Chinese officials, the international governing body of gymnastics announced yesterday it was satisfied that China's female gymnasts met minimum-age requirements during the 2008 Beijing Olympics.			
18	**Cyclist Ricco Banned 2 Years for Doping** Cyclist Riccardo Ricco was banned two years by the Italian Olympic Committee on Thursday after admitting to doping during the Tour de France.			
19	**Leg Injury to Sideline Red Wings' Chelios** Red Wings defenseman Chris Chelios will be sidelined three to six weeks with a fractured shin bone.			
20	**Rays Leave Closer Troy Percival off ALDS Roster** Tampa Bay closer Troy Percival was left off the Rays' roster for their AL division series against the Chicago White Sox due to back tightness.			
21	**Red Sox open with victory on the road** Jason Bay hit a two-run homer, Jon Lester pitched seven scoreless innings as the Boston Red Sox opened the defense of their title with a 4-1 victory in Anaheim.			
22	**Dodgers push Cubs against the wall** The Los Angeles Dodgers gained only their second postseason victory in two decades as they hit three home runs and beat the Cubs, 7-2, in Game 1 of their best-of-five National League first-round series.			
23	**Hamels lifts Phillies to their first playoff victory since '93** C.C. Sabathia's bionic arm may get most of the attention in the National League division series, but do not overlook another California-born left-hander. Cole Hamels is slim and lanky, built nothing like the imposing Sabathia, but his effort in the Phillies' 3-1 victory over the Milwaukee Brewers in Wednesday's opener served as a reminder that he is just as capable of a dominant performance.			
24	**Red Sox ace Beckett on track for Game 3 start** ANAHEIM, Calif. (AP) -- Boston pitcher Josh Beckett threw a side session Thursday during an off-day in the Red Sox-Angels playoff series, and remains scheduled to start Game 3 Sunday in Boston.			
25	**Jankovic, Venus Williams, Zvonareva, Petrova and Safina reach quarterfinals** Jelena Jankovic won her first match since learning that she will regain the No. 1 ranking, beating Alona Bondarenko 6-2, 6-0 Thursday to reach the quarterfinals of the Porsche Grand Prix. Sixth-seeded Venus Williams served 12 aces to beat Kateryna Bondarenko 6-4, 6-2. Third-seeded Dinara Safina rallied in the second set to beat qualifier Tsvetana Pironkova 6-1, 7-6 (8-6). Two other Russians also advanced to the last eight, Vera Zvonareva and Nadia Petrova.			
26	**Ferrer, Roddick advance to quarterfinals** David Ferrer and Andy Roddick lived up to their No. 1 and No. 2 seedings Thursday by progressing to the quarterfinals of the Japan Open.			
27	**Riccardo Ricco banned for 2 years** Cyclist Riccardo Ricco was banned for two years by the Italian Olympic Committee on Thursday after admitting to doping during the Tour de France.			
28	**Late birdie blitz boosts Padraig Harrington** Five birdies in the last six holes at Kingsbarns on Thursday boosted Padraig Harrington's European order of merit hopes, taking the Irishman to within three strokes of early Dunhill Links Championship leaders Soren Hansen and Ross Fisher.			
29	**Even without Woods, golf has been gripping** The surprise over the golfing summer is that the sport remained compelling, even without Tiger Woods, who called it a season in June after winning the U.S. Open despite a broken leg and a knee in need of surgery.			
30	**Feyenoord and Tottenham advance in UEFA Cup** Feyenoord overturned a 1-0 home defeat to win, 2-1, against Kalmar and go though on away goals while a 1-1 draw for Spurs helped the Premier League strugglers eliminate Wisla Krakow, 3-2 on aggregate, in the first round.			

ASSUNTO: US

Noticia Padrão: Poll Finds Obama Gaining Support and McCain Weakened in Bailout Crisis

A CBS News poll finds that Senator Barack Obama has opened a nine percentage-point lead over Senator John McCain.

ID	Notícia	nível de similaridade com a notícia padrão	Notícias Similares	Ordem de preferência de acordo com as palavras-chave
1	**Springsteen to perform at Obama rally in Michigan** Rock star Bruce Springsteen was headlining a rally Monday afternoon for Democrat Barack's Obama's presidential campaign at Eastern Michigan University.			
2	**Wreckage of Fossett's Plane Is Found** The wreckage was discovered about 120 miles south of the Nevada ranch where Steve Fossett, the millionaire adventurer, departed over a year ago.			
3	**Bush Presses House to Pass Rescue Package** The morning after the Senate strongly endorsed the bailout plan, President Bush tried to muster support for the $700 billion package ahead of a crucial vote in the House.			
4	**Bank Limits Fund Access by Colleges, Inciting Fears** Wachovia Bank has limited the access of nearly 1,000 colleges to $9.3 billion the bank has held for them, raising worries about meeting payrolls and other obligations.			
5	**Investigators Say Train Engineer Sent Text Messages Seconds Before Crash** The engineer of a commuter train in Los Angeles was sending text messages on his cellphone seconds before a crash with a freight train that killed 25 people.			
6	**Hundreds Are Arrested in Antigang Crackdown** A nationwide crackdown has brought the arrest of 1,759 people — gang members and their associates, other criminals and immigration violators — from more than 20 countries.			
7	**Pittsburgh Episcopalians Weigh Division** The Episcopal Diocese of Pittsburgh will vote Saturday on whether to become the second diocese to secede from the American branch of the Anglican Communion.			
8	**Word Reaches Congress: As the Market Goes, So Goes the Electorate** Lawmakers had been inundated with calls urging them to reject the bailout — and when they did, those same constituents called to complain about what it had done to the stock market.			
9	**An Everyman on the Trail, With Perks at Home** Senator Joseph R. Biden Jr. has benefited from resources and relationships not available to average Americans.			
10	**Moderator's Planned Book Becomes a Topic of Debate** Gwen Ifill, moderator of Thursday's debate, is the author of a coming book, "The Breakthrough: Politics and Race in the Age of Obama."			
11	**Stocks Dip as Investors Await Bailout Vote** NEW YORK, Oct. 1 -- U.S. stocks flirted with gains Wednesday but finished lower as investors waited for Congress to decide the fate of the $700 billion bailout proposal and considered the economic threats that might remain even if the legislation passed.			
12	**Senate Backs Far-Reaching Nuclear Trade Deal With India** The Senate last night approved a historic agreement that opens up nuclear trade with India for the first time since New Delhi conducted a nuclear test three decades ago, giving the Bush administration a significant foreign policy achievement in its final months.			
13	**Court Won't Reconsider Ban on Execution for Child Rape** The Supreme Court yesterday declined to revisit its June decision that imposing the death penalty on child-rapists is unconstitutional, although two justices said they would have reopened the case and two others sharply criticized the majority.			
14	**New York City Mayor Bloomberg Says He'll Seek Third Term** NEW YORK, Oct. 2 -- Mayor Michael R. Bloomberg (I) announced today that he would seek to extend the city's term limit law to three terms, instead of the current two, to allow him to run for reelection next year.			
15	**Search Crews Find Adventurer Steve Fossett's Airplane** LOS ANGELES, Oct. 2 -- Search crews high in the Sierra Nevada mountains have found wreckage of the small plane that belonged to adventure pilot Steve Fossett, who vanished 13 months ago after taking off alone from a Nevada airstrip, a local official in California said Thursday morning.			
16	**Study: Number of New Illegal Immigrants Has Fallen Sharply** The number of illegal immigrants entering the United States each year has dropped substantially since the first half of this decade, according to a study released today by the Pew Hispanic Center. A sluggish economy and stepped up enforcement of immigration laws could			

	be behind the decline.			
17	**Judge Could Dismiss Stevens's Indictment or Declare Mistrial** A federal judge this morning scolded lawyers prosecuting Alaska Sen. Ted Stevens (R) on ethics charges for waiting until the last minute to disclose potentially exculpatory material and said he would hear arguments today from defense attorneys about whether to dismiss the indictment or declare a ...			
18	**Palin's Strengths Rooted in Alaska** WASILLA, Alaska -- The valley where Sarah Palin was raised is vast and tree-packed, and so quiet that once, when Palin's parents left a door open in their antler-festooned house, a moose poked its nose into their den.			
19	**Search for Aviator Resumes After ID Is Found Near Yosemite** LOS ANGELES, Oct. 1 -- The search for adventure pilot Steve Fossett was resumed Wednesday after a hiker came across what appeared to be Fossett's pilot license and other identification while traversing rugged terrain near Yosemite National Park more than a year after he disappeared.			
20	**Democrats See the Pros and Cons of Letting Biden Be Biden** Introducing Sen. Barack Obama at a rally in Detroit on Sunday, his running mate did not hold back.			
21	**Obama says McCain out of touch on jobs** Democratic presidential candidate Barack Obama said Thursday that his rival John McCain is out of touch with the economic struggles of Americans and doesn't understand that there's nothing more fundamental than a job.			
22	**Alabama's Wilson efficient in guiding offense** John Parker Wilson's passing numbers are mostly down, and he couldn't be happier.			
23	**Military recruiting bonuses grow by 25 percent** The Army and Marine Corps doled out nearly $640 million in the past year in bonuses to entice recruits to join the military, as the two services continue to bear the brunt of the U.S.-led wars in Iraq and Afghanistan.			
24	**Daly considering European play in 2009** John Daly is thinking about playing regularly on the European Tour next year, as the two-time major champion considers all of his options for 2009.			
25	**Obama adviser suggests Gates as possible holdover** A senior adviser to Sen. Barack Obama said Thursday that the Democrat might see Defense Secretary Robert Gates as a candidate to remain at the Pentagon if Obama wins the White House.			
26	**Eagles' Westbrook improves, day to day for Sunday** All-Pro running back Brian Westbrook practiced for the second straight day Thursday, and the injured Eagles appear to be getting a little healthier.			
27	**Obama says McCain out of touch on jobs** Democratic presidential candidate Barack Obama said Thursday that his rival John McCain is out of touch with the economic struggles of Americans and doesn't understand that there's nothing more fundamental than a job.			
28	**McCain and Obama on the issues** A look at where Democrat Barack Obama and Republican John McCain stand on a selection of issues			
29	**Pelosi paying thousands to husband's firm** House Speaker Nancy Pelosi said Thursday that it's "just foolish" to suggest that her husband is benefiting from tens of thousands of dollars one of her political committees is paying a firm he owns.			
30	**NYPD officer kills himself over Taser episode** The man was naked, teetering on a building ledge and jabbing at police with an 8-foot-long fluorescent light bulb as a crowd gathered below.			

Anexo V – Respostas da 1ª e 2ª Etapas do Questionário 1 do Anexo IV

Tabela 1 – Respostas por aluno da 1ª e 2 ª etapas do Questionário 1

Perguntas	Alunos	Cássia Francine Novello	David Sodré da Silva Ferreira	Marcelo Rezende de Fázio	Diego Diniz Pinto
1.1		B	A	B	A
1.2		A		A	
1.3		2		3	
	clientes	Firefox		Firefox	
1.4	nota	8		7	
	business	job, technology, demand, payment, weather	dollar, stock exchange	business, market, stock	economy, company, stock exchange, payment, market, GDP, countries, money
	Sports	Olympics medals, brazilian soccer team, gymnast	Flamengo, brazilian soccer team, Rio de Janeiro	soccer	Botafogo, brazilian soccer team, brazilian soccer league,beach volleyball, soccer ,olympics, world cup
2.1.1	United States	elections, economy crash, stock exchange	crash, elections	economy, shares, stock, company, music	music, movies, billionaries, hot 100, dollar, elections, Obama, Mccain, singers, tecnology, software
2.1.2		1	4	5	6
2.1.3		OR	OR	AND	AND
	Business				Senate, stock exchange
	Sports	doping, players transfers	basketball, swimming, rowing		Flamengo, baseball, basketball, Argentina
2.2.1	United States	war, terrorism	Bush, Iraq		Bush, war, law
2.2.2		1	6	5	6
2.2.3		OR	AND	OR	AND

Perguntas	Alunos	André Ribeiro Vieira	Ilana LejBrowicz	Tatiane Lima da Silva	Diego Louvisse Almeida
1.1		B	A	A	B
1.2		B			B
1.3		3			3
	clientes	inforss(firefox)			Firefox
1.4	nota	4			5
	business	crash, stock exchange, Brazil	market campaign, salary, technology, information	stock investiment, interest, economy, market exchange	money, bailout
	Sports	Flamengo, soccer, olympics	olympic gymnasts	gymnast, soccer, olimpyc games	soccer, hawaii
2.1.1	United States	elections, Obama, Mccain, politics	music, dolar, movie	dolar, bank, economy, election, technology	Pittsburgh, technology, dollar exchange, election
2.1.2		4	8	8	4
2.1.3		OR	AND	AND	AND
	Business	agribusiness	economy		rural agriculture
	Sports	Vasco	Flamengo, baseball	Flamengo, tennis, baseball, handball	Figueirense, football, baseball, golf
2.2.1	United States	Artist, Celebrity, Holywood	war	Football, war, Terrorist incidents, baseball	petroleum, real property, telephony
2.2.2		1	6	1	1
2.2.3		OR	OR	OR	OR

Perguntas	Alunos	Alexandre F.S. Mattos	Cláudio Sérgio Forain Jr	Ricardo Rocha Soares	Ana Carolina Gama e Silva Assaife
1.1		B	A	B	A
1.2		A		A	
1.3		7		3	
	clientes	Firefox, Orkut		Internet explorer e firefox	
1.4	nota	2,5			
	business	jobs, technology, google	bailout, stock	computer	market exchange
	Sports	Fluminense, premier league, germany	Martial arts, NBA, Wii, sports	soccer, futsal, Flamengo, Formula 1	tennis, olympic games
2.1.1	United States	election, interchange	bailout, Obama, Mccain	election	movie, music, economy
2.1.2		3	7	8	10
2.1.3		AND	OR	AND	AND
	Business	management market		fashion	bailout
	Sports	voleyball, automobilism	soccer, uefa, tennis, golf	tennis, baseball, gymnast	baseball, basketball, boxe
2.2.1	United States	earthquake	Bush, Holywood	interchange	politic
2.2.2		8	1	2	6
2.2.3		OR	AND	OR	OR

Perguntas	Alunos	Rafael Souza Nadir	Renato Fischer	Eduardo Maciel Cunha	Daniel Cardoso Queiroga
1.1		B	A	C	A
1.2		B		B	
1.3		0		9	
	clientes	igoogle		Firefox	
1.4	nota	1		8	
	business	Market, stock, investiment, bailout	Market, exchange	Money, oil price, credit, economic, consumers	market stock
	Sports	Soccer, brazilian, flamengo, european, english, italian league, uefa cup	soccer, tennis	gymnasts, injury, doping, victory	soccer, tennis, olimpic games, swimming
2.1.1	United States	Obama, McCain, Hollywood	catastroph, disaster, death	Obama, stocks, trade deal, military	music, movies, politic, travel
2.1.2		1	6	7	5
2.1.3		AND	AND	AND	AND
	Business	Stock, unemployment	politic	Distressed Homeowners, Take Role Auto Sales, Japan's Carmakers	
	Sports	baseball, basketball, handball, golf	motociclism, ciclism, baseball	baseball, cornerback, Red Sox, Uefa Cup	Flamengo, baseball, football
2.2.1	United States	Iraq, war, militar		Weigh Division, European, husband's firm	Jenorismo, Bush
2.2.2		4	5	5	6
2.2.3		OR	AND	AND	AND

Perguntas	Alunos	Marcelo Vieira Monteiro	Bruno José da Costa Medeiros
1.1		A	B
1.2			B
1.3			1
	clientes		igoogle
1.4	nota		8
	business	Money, economy, market, finance, investiment, bank, dolar, stock	Datawarehouse, business, intelligence
	Sports	soccer, Brazil, NBA, Fórmula 1	soccer, cup
2.1.1	United States	war, Bill Gates, Iraq, New york, Florida, Walt Disney, Hollywood, Oscar, Microsoft, NASA, Dolar, Terrorism	Statue Liberty, turism
2.1.2		7	8
2.1.3		AND	AND
	Business		marketing
	Sports		baseball, football
2.2.1	United States		Bush, politics
2.2.2			4
2.2.3			AND

Anexo VI – Experimentos Relacionados ao Padrão Eliminador de Dados Similares usado pela Versão Alfa da Ferramenta

Tabela 1 – Notícias retornadas por nível de similaridade para o padrão DS no domínio negócios

Similaridade (%)	Identificação das notícias retornadas
1	19, 20
2	19, 20
3	19, 20
4	19, 20
5	19, 20
6	19, 20
7	13, 17, 19, 20
8	4, 12, 13, 15, 16, 17, 18, 19, 20
9	4, 12, 13, 15, 16, 17, 18, 19, 20, 27
10	4, 8, 9, 12, 13, 15, 16, 17, 18, 19, 20, 23, 27
15	1, 2, 3, 4, 5, 6, 8, 9, 12, 13, 14, 15, 16, 17, 18, 19, 20, 23, 26, 27, 30
20	1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 13, 14, 15, 16, 17, 18, 19, 20, 23, 25, 26, 27, 30
25	1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 23, 25, 26, 27,28, 30
30	1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 23, 25, 26, 27, 28, 30
40	1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 22, 23, 24, 25, 26, 27, 28, 30
50	1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30

Tabela 2 – Notícias retornadas por nível de similaridade para o padrão DS no domínio esportes

Sports	
Similaridade (%)	**Notícias retornadas**
1	11, 15, 16, 17, 18, 19
2	11, 15, 16, 17, 18, 19
3	8, 11, 13, 14, 15, 16, 17, 18, 19, 28
4	8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 28
5	8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 23, 29, 28
6	8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 23, 28, 29, 30
7	8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 23, 28, 29, 30
8	8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 23, 24, 28, 29, 30
9	7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 23, 24, 27, 28, 29, 30
10	5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 23, 24, 25, 27, 28, 29, 30
15	4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 23, 24, 25, 26, 27, 28, 29, 30
20	1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 23, 24, 25, 26, 27, 28, 29, 30
25	1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 21, 23, 24, 25, 26, 27, 28, 29, 30
30	1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 21, 23, 24, 25, 26, 27, 28, 29, 30
40	1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30
50	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30

Tabela 3 – Notícias retornadas por nível de similaridade para o padrão DS no domínio notícias

Similaridade (%)	Identificação das notícias retornadas
1	10, 19, 20
2	10, 19, 20
3	4, 10, 17, 19, 20
4	4, 7, 10, 13, 16, 17, 19, 20, 24, 26, 29, 30
5	4, 6, 7, 10, 12, 13, 16, 17, 18, 19, 20, 24, 26, 29, 30
6	4, 6, 7, 8, 10, 12, 13, 14, 16, 17, 18, 19, 20, 22, 24, 26, 29, 30
7	3, 4, 6, 7, 8, 10, 12, 13, 14, 16, 17, 18, 19, 20, 22, 23, 24, 26, 30
8	3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 22, 23, 24, 26, 29, 30
9	3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 22, 23, 24, 26, 29, 30
10	3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 22, 23, 24, 26, 29, 30
15	1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 22, 23, 24, 26, 29, 30
20	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 22, 23, 24, 25, 26, 29, 30
25	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 22, 23, 24, 25, 26, 28, 29, 30
30	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 22, 23, 24, 25, 26, 28, 29, 30
40	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30
50	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30

Tabela 4 – Matriz de similaridade para o domínio negócios

XX	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	0	1	1	0	0	0	2	0	0	1	0	1	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0
2	1	0	1	1	0	0	0	5	0	1	0	1	2	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	4	0
3	1	0	0	0	2	2	1	0	0	1	1	1	0	0	1	2	1	0	0	1	0	2	0	1	1	0	0	0	0	0
4	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1
5	3	1	3	0	0	2	2	1	0	2	2	2	0	0	0	3	0	1	0	0	0	3	0	0	0	0	0	0	0	0
6	1	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0
7	3	0	2	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	4	0
9	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	4	0	0
10	1	3	2	0	1	1	2	0	0	0	0	1	0	0	0	7	0	0	1	0	0	1	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	0	0	0	0	1	0	3	0	1	1	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	2	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	2	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0
14	2	0	0	0	1	2	1	0	0	2	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0	0
15	0	0	0	0	0	1	1	0	0	0	0	0	0	2	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
16	0	0	1	1	1	2	0	0	0	5	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0
17	2	1	1	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
18	1	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	1	0	1	5	0	0	0	0	0
22	0	0	1	0	1	1	0	0	0	0	5	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	1	0	0	0	0	5	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	2	0	1	0	0	1	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0
27	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
28	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0
29	0	3	0	0	0	0	0	4	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	1	0	0
30	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Tabela 5 – Matriz de similaridade para o domínio esportes

XX	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	0	0	0	0	0	1	3	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
2	1	0	0	1	0	0	0	0	0	0	0	2	4	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	1
3	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	8	0	0	0	1	0	0	0	0	0	0	0	0	0
4	1	2	1	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	1	0	0	0	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0
7	2	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0
8	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
9	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0
12	0	2	0	2	0	0	0	0	0	0	0	0	2	0	0	0	1	0	2	1	0	0	0	0	1	0	0	0	0	0
13	0	4	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
14	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	1	7	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0
18	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0
19	0	0	0	1	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0
20	0	0	0	0	0	0	0	0	0	0	3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
21	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	3	1	0	1	0	0	0
22	0	0	0	0	0	1	0	2	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
23	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
27	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	5	0	0	0	0	0	0	0	1	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0

Tabela 6 – Matriz de similaridade para o domínio notícias

XX	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	2	1	0	0	0	0	1	2	1	1
2	0	0	0	0	5	0	2	0	0	0	0	1	0	0	7	0	0	2	3	0	0	0	0	0	1	0	0	0	0	0
3	0	0	0	0	0	1	0	0	1	0	1	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
10	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1	1	1	1	0	1	0	2	1	0	0
12	1	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0
13	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	5	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0
17	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
18	0	2	0	0	0	0	0	0	0	0	0	0	0	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	3	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
20	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
21	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	6	0	0	0
22	0	0	0	0	1	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
24	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0
26	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	10	1	0	0	0	0	0	1	0	0
28	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	3	0	0	0	0	0	2	0	0	0
29	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Anexo VII – Questionário Utilizado no Experimento com a Versão Beta da Ferramenta

Pesquisa

1ª etapa

1 – Qual a sua experiência com RSS?

( ) A – Nunca usei
( ) B – Conheço mas não uso freqüentemente
( ) C – Utilizo com freqüência, mas assino poucos feeds
( ) D – Utilizo com freqüência, e assino vários feeds

Atenção: As perguntas de 2 a 4 só devem ser respondidas se você NÃO marcou **A** na pergunta acima.

2 – Como você escolhe o uso dos feeds?

( ) A – Vejo feeds de notícias gerais. Ex.: ultimosegundo.com.br
( ) B – Assino feeds específicos por assunto.
( ) C – Assino feeds de vários sites indiscriminadamente

3 – Dê a sua nota para o nível de similaridade (o quão parecido) às noticias que recebe. Considere os últimos quinze dias e que as notas variam de nota 0 (para nenhuma notícia repetida ou parecida) a nota 10 (para mais de dez notícias parecidas).

(0) (1) (2) (3) (4) (5) (6) (7) (8) (9) (10)

4- Quais os agregadores (clientes) de feeds que você utiliza? Ex: Mozilla Firefox, Internet Explorer 7, FeedGhost, Juice, Akregator, Vienna, etc. Assinale também como vc classifica esses agregadores (clientes) de feeds: as notas variam de nota 0 (para pobres, que não fornecem opções) a nota 10 (para os completos, que fornecem muitas opções).

_____

_____

2ª etapa

É necessário ler as notícias que lhe foram designadas (total de 20 notícias por participante) e dizer se são relevantes ou não, considerando as seguintes regras:

- Palavras Proibidas: **exercise, acid, japan, barcelona**;
- Palavras Permitidas: **acupuncture, health, mexico, liverpool, manchester, nadal, g20, phelps, dylan, inflation, software, microsoft**;
- Assunto proibido: **política**

As notícias podem ser acessadas na pasta *.../tese/anexos/forms/*, contida no DVD fornecido com este questionário ou através dos links da Tabela 1.

Tabela 1 – Links dos Formulários

Formulário ou planilha	Link
Form1	http://spreadsheets.google.com/ccc?key=0Al-r8Wms_DYidFM0MVJIMktRVGhoSG1weVU0Q0hpRkE&hl=en
Form2	http://spreadsheets.google.com/ccc?key=0Al-r8Wms_DYidFI4SmFGZ3lJVTZCd08tekZUcDRHWXc&hl=en
Form3	http://spreadsheets.google.com/ccc?key=0Al-r8Wms_DYidFRBenZ0MklDcjFWN0xGSi1CSzQ5dFE&hl=en
Form4	http://spreadsheets.google.com/ccc?key=0Al-r8Wms_DYidFFwR0ZMUWhrSWRTcURHX0FzRzNaREE&hl=en
Form5	http://spreadsheets.google.com/ccc?key=0Al-r8Wms_DYidExKenFDV3A2aUYxcWk2TE54V2JWeGc&hl=en
Form6	http://spreadsheets.google.com/ccc?key=0Al-r8Wms_DYidHVleHJ6MUcxMTNCbWdNZk43RjRqSGc&hl=en
Form7	http://spreadsheets.google.com/ccc?key=0Al-r8Wms_DYidHNxbnl4SkEyUy1maDBPckd5SlJJclE&hl=en
Form8	http://spreadsheets.google.com/ccc?key=0Al-r8Wms_DYidGFaTENMeHU3c1ViZEtZb3ZqaU5raHc&hl=en
Form9	http://spreadsheets.google.com/ccc?key=0Al-r8Wms_DYidHlHYURYMFBrbGUyQlVVU3JsdGNKTFE&hl=en
Form10	http://spreadsheets.google.com/ccc?key=0Al-r8Wms_DYidHZEVXVjTVA1eklsbVViUlg3bWJub3c&hl=en
Planilha1	http://spreadsheets.google.com/ccc?key=0Al-r8Wms_DYidFZKZTNtWUZnclFGNkJPN1dTdjM1VkE&hl=en
Planilha2	http://spreadsheets.google.com/ccc?key=0Al-r8Wms_DYidGVRRGdZUFNadzdzNmFYTG1sNE5wOEE&hl=en

As listas negras que podem ser acessadas na pasta *.../tese/anexos/BlackLists/*, contida no DVD fornecido com este questionário.

Anexo VIII – Dados dos Participantes que Realizaram o Experimento com a Versão Beta da Ferramenta

Tabela 1 – Dados Gerais dos Participantes e seu Relacionamento com os Formulários e Planilhas Respondidas

Nr	Nome	email	Escolaridade	Instituição / Empresa	Formulários (total de 100 Noticias)	Planilha 1 e Planilha 2 (total de 300 noticias)
01	Rarylson Freitas	rarylson@gmail.com	Aluno 3º ano de graduação em Engenharia de Computação	IME	Form 4	Aba aluno 1
02	Leonardo Borges	lborgav@gmail.com	Aluno 3º ano de graduação em Engenharia de Computação	IME	Form 6	Aba aluno 2
03	Fábio Luiz Junior	junior_IME2011@hotmial.com	Aluno 3º ano de graduação em Engenharia de Computação	IME	Form 3	Aba aluno 3
04	Vinícius Hessel Benedito de Souza	vinicius.hessel.bs@gmail.com	Aluno 3º ano de graduação em Engenharia de Computação	IME	Form 1	Aba aluno 4
05	Samir Elias Hachem Kerbage	samirehr@gmail.com	Aluno 3º ano de graduação em Engenharia de Computação	IME	Form 9	Aba aluno 5
06	Igor Alvarenga S. Nascimento	igor.ime@gmail.com	Aluno 4º ano de graduação em Engenharia de Computação	IME	Form 5	Aba aluno 6
07	Debora Helena Job	deborajob@ime.eb.br	Aluno de mestrado em	IME	Form 2	Aba aluno 7

			Sistemas e Computação			
08	David Fernandes Cruz Moura	david@ime.eb.br	Aluno de doutorado em Defesa	IME	Form 7	Aba aluno 8
09	Ana Bárbara Sapienza Pinheiro	asapienza@uol.com.br	Graduada em Enfermagem e Pós-graduada em Neonatologia	PMRJ	Form 8	Aba aluno 9
10	Wallace Anacleto Pinheiro	awallace@uol.com.br	Aluno de doutorado em Engenharia de Sistemas e Computação	UFRJ/IME	Form 10	Aba aluno 10
11	Flavio da Silva Costa	costaflavio@gmail.com	Aluno de mestrado em Engenharia de Sistemas e Computação	UFRJ	Form 10	Aba aluno 11
12	Ricardo Barros	rbarros@cos.ufrj.br	Doutor em Engenharia de Sistemas e Computação	UFRJ	Form 9	Aba aluno 12
13	Bernardo Pacheco	bernardo.wpacheco@gmail.com	Aluno do 5° ano de Ciência da Computação	UFRJ	Form 8	Aba aluno 13
14	Luciano Terres	lucianoterres@gmail.com	Aluno de mestrado em Engenharia de Sistemas e Computação	UFRJ/Petrobrás	Form 6	Aba aluno 14
15	Bruno Osiek	baosiek@gmail.com	Aluno de doutorado em Engenharia de Sistemas e Computação	UFRJ	Form 6	Aba aluno 15
16	Leonardo Henrique Moreira	leohmoreira@gmail.com	Aluno do 5° ano de Ciência da Computação	IME	Form 2	Aba aluno 17
17	Flávia Castellan Braga	flaviacastellan@yahoo.com.br	Bacharel em Relações Internacionais	UERJ/BNDES	Form 1	Aba aluno 18
18	Ricardo Cerceau	ricardocerceau@gmail.com	Bacharel em Ciência da Computação	TOTVS/Centro Univ. de Volta Redonda	Form 10	Aba aluno 22
19	Marcelo Arêas Rodrigues da Silva	mareas@gmail.com	Bacharel em Ciência da Computação	UFRJ	Form 9	Aba aluno 16
20	Frederico Andrade de Homobono Balieiro	frederico.balieiro@chemtech.com.br	Engenharia Eletronica e Computação	UFRJ/Chemtech	Form 8	Aba aluno 19

21	Diego Vargas Jannibelli	diego.jannibelli@chemtech.com.br	Aluno do 4° ano de Ciência da Computação	UFRJ/Chemtech	Form 7	Aba aluno 20
22	Marcelo Caniato Renhe	marcelo.caniato@gmail.com	Aluno de mestrado em Engenharia de Sistemas e Computação	UFRJ	Form 5	Aba aluno 21
23	Vitor Padilha Gonçalves	vitorpadilha@gmail.com	Bacharel em Ciência da Computação	UFJF/Politec	Form 4	Aba aluno 30
24	Marcelo Pereira Lopes	marcelovrb@gmail.com	Aluno de mestrado em Engenharia de Sistemas e Computação	UFRJ	Form 3	Aba aluno 24
25	Guilherme Faria de Miranda	guilherme.miranda@chemtech.com.br	Engenheiro da Computação	UFRJ/Chemtech	Form 2	Aba aluno 23
26	Flávio Pimentel Duarte	flavio.duarte@chemtech.com.br	Mestre em Eng de Sistemas e Computação	UFRJ/Chemtech	Form 6	
27	Carlos Augusto de Souza	carlos.souza@chemtech.com.br	Bacharel em Ciência da Computação	Chemtech/Centro Univ. de Volta Redonda	Form 2	
28	João Pedro Navarini	joao.navarini@chemtech.com.br	Engenheiro de Eletronica e Computação	UFRJ/Chemtech	Form 4	Aba aluno 25
29	Rodrigo Pereira dos Santos	rps@cos.ufrj.br	Aluno de mestrado em Engenharia de Sistemas e Computação	UFRJ	Form 3	Aba aluno 26
30	Fernando Fernandes Morgado	fernandofmorgado@gmail.com	Aluno de mestrado em Engenharia de Sistemas e Computação	UFRJ	Form 6	
31	Kersom Moura Oliveira	kersom.oliveira@chemtech.com.br	Engenheiro Eletricista	UFV	Form 1	Aba aluno 29
32	Bruno César Barbosa Alves	brunobarbosa@gmail.com	Aluno de doutorado em Engenharia de Sistemas e Computação	UFRJ/DataPrev	Form 8	Aba aluno 28
33	Erick Menezes Moreira	erick.mnzs@gmail.com	Aluno 4º ano de graduação em Engenharia de Computação	IME	Form 7	Aba aluno 27

Obs.: Os formulários e abas de planilhas destacados em cinza não foram preenchidos pelos participantes.

Tabela 2 – Respostas à Primeira Parte da Questionário do Anexo VII

Nr	Nome	Qual a sua experiência com RSS?	Como você escolhe o uso dos feeds?	Nota para o nível de similaridade	Agregadores Utilizados	Nota dos Agregadores
01	Rarylson Freitas	A				
02	Leonardo Borges	A				
03	Fábio Luiz Junior	A				
04	Vinícius Hessel Benedito de Souza	A				
05	Samir Elias Hachem Kerbage	D	C	4	Google Reader	
06	Igor Alvarenga S. Nascimento	A				
07	Debora Helena Job					
08	David Fernandes Cruz Moura	A				
09	Ana Bárbara Sapienza Pinheiro	B	A	3	Firefox	5
10	Wallace Anacleto Pinheiro	D	A	10	Firefox, Yahoo Pipe	4 e 7
11	Flavio da Silva Costa	B	A	1		
12	Ricardo Barros	B	B	7	Firefox	8
13	Bernardo Pacheco					
14	Luciano Terres					
15	Bruno Osiek					
16	Leonardo Henrique Moreira	A				
17	Flávia Castellan Braga	A				
18	Ricardo Cerceau	A				
19	Marcelo Arêas Rodrigues da Silva	B	A	5	Firefox	6
20	Frederico Andrade de Homobono Balieiro	B	A	8	Firefox	9
21	Diego Vargas Jannibelli	B	A	4	Firefox	6

22	Marcelo Caniato Renhe	B	B	6	Akregator e Google Reader	4
23	Vitor Padilha Gonçalves	A				
24	Marcelo Pereira Lopes	B	A	7	Internet Explorer e Firefox	7
25	Guilherme Faria de Miranda	B				
26	Flávio Pimentel Duarte	A				
27	Carlos Augusto de Souza	C	C	5	Google Reader	10
28	João Pedro Navarini	D	B	10	Safari e Mail da Apple	7
29	Rodrigo Pereira dos Santos	B	A	1	Firefox	3
30	Fernando Fernandes Morgado	B	A	8	Firefox	8
31	Kersom Moura Oliveira	C	B	8	Firefox	7
32	Bruno César Barbosa Alves	B	B	7	Google Reader e Thunderbird	9 e 8
33	Erick Menezes Moreira	A				

Anexo IX – Notícias e Repostas Obtidas

Tabela 1 – Identificação e Título das Notícias associadas às Respostas fornecidas pelos usuários e as Repostas e Ordem das notícias fornecidas pelas Ferramentas Feed Organizer versão Beta e Yahoo Pipe
(KEY e JAC indicam as configurações usando somente palavras-chave ou a combinação da função de Jaccard com palavras-chave, respectivamente. Data e Sim indicam a ordenação por data e similaridade, respectivamente. A relevância, no caso das ferramentas Yahoo Pipe e Feed Organizer, indica se houve retorno ou não da notícia. No caso das notícias não retornadas pelas ferramentas, a ordem adotada é a do ID da notícia e não tem efeito nos cálculos realicados nos experimentos).

ID	Título da Notícia	Relevância Usuario	Ordem Yahoo	Relevância Yahoo	Ordem Feed Org. Key e Jac por Data	Relevância Feed Org. Key e Jac por Data	Ordem Feed Org. Key e Jac por Sim	Relevância Feed Org. Key e Jac por Sim	Ordem Feed Org. Key por Data	Relevância Feed Org. Key por Data	Ordem Feed Org. Key por Sim	Relevância Feed Org. Key por Sim
1	Doubts raised over Lloyds' commitment to lending	sim	135	nao	281	nao	281	nao	252	nao	252	nao
2	Rebuilding the global economy at G20: it's a lot to do in just one day	sim	136	nao	40	sim	10	sim	253	nao	253	nao
9	Banks and building societies lift savings rates	sim	137	nao	283	nao	283	nao	257	nao	257	nao
12	Halabi investors act to cut losses	sim	138	nao	286	nao	286	nao	260	nao	260	nao
18	A $191 Million Question	nao	139	nao	289	nao	289	nao	264	nao	264	nao
30	Freddie Mac Reports Profit; Won't Tap Treasury for More Aid	nao	140	nao	290	nao	290	nao	265	nao	265	nao
81	Business events scheduled for the coming month	nao	141	nao	291	nao	291	nao	266	nao	266	nao
101	District's Financial Woes Reopen Debate on Tax Burden	nao	142	nao	287	nao	287	nao	261	nao	261	nao
110	Bankruptcies	sim	143	nao	284	nao	284	nao	258	nao	258	nao
135	'Blue-eyed bankers' prompt G20 divide	nao	129	sim	111	sim	11	sim	254	nao	254	nao
139	Japan on the brink of deflation	sim	144	nao	282	nao	282	nao	255	nao	255	nao
141	Fred Goodwin's pension package under fire from top shareholders	nao	145	nao	288	nao	288	nao	262	nao	262	nao
158	Inflation or deflation?	sim	146	nao	113	sim	12	sim	263	nao	263	nao
178	FRC review looks to 'engage' shareholders and companies	sim	147	nao	292	nao	292	nao	267	nao	267	nao
185	ITV to offload Friends Reunited	sim	148	nao	350	nao	350	nao	330	nao	330	nao
200	Bob Dylan to release Christmas album	sim	75	sim	34	sim	13	sim	26	sim	8	sim
211	Oscars ceremony should be first, says Meryl Streep	nao	149	nao	300	nao	300	nao	275	nao	275	nao
212	Hollywood searches for escapism after the apocalypse	nao	150	nao	297	nao	297	nao	272	nao	272	nao
217	TV Preview of Reality Show Giuliana and Bill'	sim	151	nao	294	nao	294	nao	269	nao	269	nao
218	Lisa de Moraes's TV Column: Paula Abdul Leaves Her Judging Job on 'American Idol' After Eight Seasons	nao	152	nao	293	nao	293	nao	268	nao	268	nao

254	Music Review | Marcus Roberts: Warmly Balancing Mind and Heart	nao	153	nao	299	nao	299	nao	274	nao	274	nao
256	Harry Potter Is Their Peter Pan	nao	154	nao	298	nao	298	nao	273	nao	273	nao
258	Heinz Edelmann, Yellow Submarine Artist, Dies at 75	sim	155	nao	295	nao	295	nao	270	nao	270	nao
259	Music Review: A Place for Piano, Even When It Needs 6 Hands	nao	156	nao	296	nao	296	nao	271	nao	271	nao
265	Music Review | Summergarden: String Quartets Alfresco, Surrounded by Sculpture and Sunsets	sim	157	nao	301	nao	301	nao	276	nao	276	nao
295	Green spending in UK economic rescue package 'negligible'	sim	158	nao	285	nao	285	nao	259	nao	259	nao
315	Obama may delay signing up to Copenhagen climate deal	nao	159	nao	330	nao	330	nao	306	nao	306	nao
322	Ecuador Pursues Unusual Carbon-Credit Plan to Leave Oil Untapped	sim	160	nao	359	nao	359	nao	339	nao	339	nao
350	Democrats May Ease Bill's Emissions Rules	nao	161	nao	329	nao	329	nao	305	nao	305	nao
370	Swine Flu Should Not Close Most Schools, Federal Officials Say	sim	22	sim	318	nao	318	nao	294	nao	294	nao
375	Television Viewing Linked to Blood Pressure Increases in Children	sim	162	nao	314	nao	314	nao	289	nao	289	nao
377	With High-Profile Death, Focus on High-Risk Drug	sim	163	nao	312	nao	312	nao	287	nao	287	nao
378	Hispanics Who Move to U.S. Face Higher Cancer Rates	sim	164	nao	313	nao	313	nao	288	nao	288	nao
383	New Strain of H.I.V. Is Discovered	sim	165	nao	316	nao	316	nao	291	nao	291	nao
386	Findings May Explain Gap in Cancer Survival	sim	166	nao	319	nao	319	nao	295	nao	295	nao
390	Medical Papers by Ghostwriters Pushed Therapy	sim	167	nao	310	nao	310	nao	285	nao	285	nao
395	Warning From F.D.A. on Arthritis Drugs for Young Patients	sim	168	nao	303	nao	303	nao	278	nao	278	nao
440	Another good reason to exercise	nao	169	nao	302	nao	302	nao	277	nao	277	nao
500	How to reduce your risk of oral thrush	sim	170	nao	305	nao	305	nao	280	nao	280	nao
501	MMR vaccine and autism	sim	171	nao	304	nao	304	nao	279	nao	279	nao
502	Complications of rubella	sim	172	nao	307	nao	307	nao	282	nao	282	nao
503	Expectant mothers should buckle up to keep baby safe	sim	173	nao	306	nao	306	nao	281	nao	281	nao
504	Ginger may help ease nausea from cancer treatment	sim	174	nao	309	nao	309	nao	284	nao	284	nao
505	Folic acid in pregnancy may reduce risk of baby heart defects	nao	175	nao	308	nao	308	nao	283	nao	283	nao
506	Acupuncture needles can improve back pain - and so can toothpicks	sim	176	nao	311	nao	311	nao	286	nao	286	nao
507	What's the evidence for acupressure for normal nausea and vomiting?	sim	177	nao	72	sim	33	sim	50	sim	16	sim
519	What's the evidence for antibiotics?	sim	178	nao	315	nao	315	nao	290	nao	290	nao
521	What's the evidence for surgery to remove your fibroids?	sim	179	nao	317	nao	317	nao	292	nao	292	nao
527	Surgery to remove your womb	sim	180	nao	73	sim	101	sim	293	nao	293	nao
635	Government officials dismiss fresh calls for torture inquiry	sim	181	nao	327	nao	327	nao	303	nao	303	nao
640	Cut-price deals targeted in food strategy	sim	3	sim	44	sim	100	sim	256	nao	256	nao
643	1,381 phone and email taps a day	sim	182	nao	349	nao	349	nao	329	nao	329	nao
644	Gordon Brown to start Lake District holiday	sim	183	nao	323	nao	323	nao	299	nao	299	nao

649	MI6 chief denies complicity in torture	nao	184	nao	322	nao	322	nao	298	nao	298	nao
659	Cameron is much more than sizzle. And Obama knows it	nao	7	sim	320	nao	320	nao	296	nao	296	nao
666	Sen. Martinez of Florida to Give Up Seat Soon	nao	185	nao	331	nao	331	nao	307	nao	307	nao
668	Congress Refuels 'Clunkers' Program	nao	186	nao	328	nao	328	nao	304	nao	304	nao
671	Senate Votes 68 to 31 to Confirm Sonia Sotomayor to Supreme Court	nao	187	nao	326	nao	326	nao	302	nao	302	nao
673	Senate Committee Bipartisan Health-Care Reform Talks Move Toward Center	nao	188	nao	324	nao	324	nao	300	nao	300	nao
674	Democrats Rally for Sotomayor	nao	189	nao	325	nao	325	nao	301	nao	301	nao
680	GOP Senators Seem Unconcerned About Hispanic Backlash Over Sotomayor Opposition	nao	190	nao	321	nao	321	nao	297	nao	297	nao
706	Man Utd 1-4 Liverpool	nao	191	nao	112	sim	14	sim	309	nao	309	nao
710	Ecclestone: No Donnington, no British GP	sim	192	nao	336	nao	336	nao	315	nao	315	nao
711	Daniel Taylor: Ferguson's quest is almost complete	sim	193	nao	115	sim	34	sim	314	nao	314	nao
727	Phelps admits 'stupid mistake' over photo	sim	194	nao	114	sim	17	sim	319	nao	319	nao
729	Fairy-tale beginning for Button and Brawn in Melbourne	sim	195	nao	344	nao	344	nao	324	nao	324	nao
745	Peterborough Utd 2-0 Leicester City	sim	196	nao	345	nao	345	nao	325	nao	325	nao
780	Barcelona sign Ibrahimovic in a 66m deal	nao	197	nao	334	nao	334	nao	311	nao	311	nao
782	Giggs: Owen can emulate Van Nistelrooy	nao	67	sim	63	sim	102	sim	313	nao	313	nao
783	Liverpool owners seal debt refinance deal	sim	68	sim	62	sim	16	sim	44	sim	10	sim
784	Martins eyes Newcastle departure	sim	198	nao	335	nao	335	nao	312	nao	312	nao
788	N.F.L. Opens Door to Vick, Though Not All the Way	nao	199	nao	333	nao	333	nao	310	nao	310	nao
790	Swimmers Need Faster Times Earlier at World Championships	sim	200	nao	332	nao	332	nao	308	nao	308	nao
792	Goal: Swedish All-Star Relishes M.L.S. and Fans Enthusiasm	nao	201	nao	340	nao	340	nao	320	nao	320	nao
797	On Par: After Accident, Teaching Golf and Perseverance	sim	202	nao	338	nao	338	nao	317	nao	317	nao
799	Mexico 5, United States 0: Mexico Thumps U.S. to Win Gold Cup	sim	78	sim	341	nao	341	nao	321	nao	321	nao
809	In Mexico, a Soccer Stadium Where Visitors Tend to Gasp	sim	113	sim	339	nao	339	nao	318	nao	318	nao
811	Stewart Widens Points Lead With Win	nao	203	nao	337	nao	337	nao	316	nao	316	nao
812	Nadal Begins His Comeback in a Different Tennis World	sim	114	sim	41	sim	15	sim	30	sim	9	sim
816	Analysis: Golf and Rugby Have Best Shot at Being Added for 2016 Games	sim	204	nao	343	nao	343	nao	323	nao	323	nao
826	Roundup: Dixon Sets IndyCar Career Record With 4th Win of Season	sim	205	nao	342	nao	342	nao	322	nao	322	nao
838	First Wi - Fi Pacemaker In US Gives Patient Freedom	sim	12	sim	45	sim	103	sim	33	sim	54	sim
840	As Rivals Branch Out, SAP Is Sticking to Software	sim	13	sim	68	sim	18	sim	48	sim	11	sim
842	Toshiba to Join Rival Blu-Ray Camp on Video Discs	sim	206	nao	353	nao	353	nao	333	nao	333	nao
846	Microsoft to Sell Web Ad Agency	sim	14	sim	42	sim	19	sim	31	sim	12	sim
870	Personal Tech: Gadget News and Reviews	sim	207	nao	347	nao	347	nao	327	nao	327	nao

871	Researchers: XML Security Flaws are Pervasive	sim	208	nao	348	nao	348	nao	328	nao	328	nao
879	Va. Tech Students Design a Vehicle for Blind Drivers, Who Take a Test Spin	nao	209	nao	346	nao	346	nao	326	nao	326	nao
883	Drawing Scrutiny, Microsoft and Yahoo Strike a Partnership	sim	72	sim	351	nao	351	nao	331	nao	331	nao
884	Apple, AT and T Questioned About Google Voice Application	sim	210	nao	352	nao	352	nao	332	nao	332	nao
890	Microsoft Fixes 19 Windows Security Flaws	sim	111	sim	39	sim	20	sim	29	sim	13	sim
989	War Without Borders: Mexico's Drug Traffickers Continue Trade in Prison	nao	211	nao	43	sim	21	sim	32	sim	14	sim
991	At Least 50 Killed in Bombings in Iraq	nao	212	nao	356	nao	356	nao	336	nao	336	nao
992	Taliban Seize Building for Attack on Afghan Government Offices	nao	213	nao	357	nao	357	nao	337	nao	337	nao
993	Dozens Dead and Hundreds Feared Missing From Typhoons	sim	214	nao	354	nao	354	nao	334	nao	334	nao
994	Obama Says Immigration Changes Must Wait Till 2010	sim	104	sim	355	nao	355	nao	335	nao	335	nao
995	Rapes Are Alleged in Iranian Prison	nao	215	nao	361	nao	361	nao	341	nao	341	nao
996	Fatah Turns to Nation Building, Though It Does not Discard the Rifle	nao	216	nao	362	nao	362	nao	342	nao	342	nao
998	India Searches N. Korean Ship for Nuclear Materials	nao	217	nao	360	nao	360	nao	340	nao	340	nao
1000	Nigerian Amnesty Plan Faces Difficulties	nao	218	nao	358	nao	358	nao	338	nao	338	nao
BUSINESS_0	ID: BUSINESS_0 Title: Doubts raised over Lloyds' commitment to lending Description: • Lloyds thought to want to reduce exposure to APS• Shares fall 4% on news of £20bn fundraising pushJill TreanorThe government is understood to be close to hammering out a	sim	219	nao	116	nao	116	nao	58	nao	58	nao
BUSINESS_1	ID: BUSINESS_1 Title: Friends Provident ready to submit to Cowdery's Resolution Description: • Shareholders pushed insurer back into talks• Sir Adrian Montague set to leave after losing fightFriends Provident is poised to accept a £1.86bn takeover offer b	sim	220	nao	117	nao	117	nao	59	nao	59	nao
BUSINESS_10	ID: BUSINESS_10 Title: Halabi investors act to cut losses Description: Simon Halabi the former owner of collapsed gym chain Esporta, has been hit by plunging property values on his portfolioInvestors facing a multimillion-pound loss on a loan linked to pr	nao	221	nao	126	nao	126	nao	68	nao	68	nao
BUSINESS_11	ID: BUSINESS_11 Title: All today's business stories Description:	nao	222	nao	127	nao	127	nao	69	nao	69	nao
BUSINESS_12	ID: BUSINESS_12 Title: Pregnant staff face new wave of workplace bullying in recession Description: Maternity leave has always been problematic for female professionals. But in the teeth of recession, pregnant women - and working mothers - are under threa	sim	69	sim	67	sim	42	sim	70	nao	70	nao
BUSINESS_14	ID: BUSINESS_14 Title: Alistair Darling to give budget cash boost to poorest Description: • Ministers back call for benefit rise • Brown brokers G20 poverty dealThe chancellor is preparing to channel cash to poorer families in his budget as part of a mini	sim	128	sim	82	sim	44	sim	72	nao	72	nao
BUSINESS_15	ID: BUSINESS_15 Title: Let us merge or we'll die, say local papers Description: Survival plan envisages new regional supergroupsLocal newspaper publishers will submit a report to the Office of Fair Trading on Tuesday calling for a massive shake-up of comp	nao	223	nao	128	nao	128	nao	73	nao	73	nao
BUSINESS_16	ID: BUSINESS_16 Title: 'Blue-eyed bankers' prompt G20 divide Description: The president of Brazil's attack last week exposed the growing rift between the west and the world's emerging powerhouses over how to tackle the global crisis. This week in London's	sim	224	nao	81	sim	1	sim	74	nao	74	nao
BUSINESS_17	ID: BUSINESS_17 Title: Tax havens decide to batten down Description: Offshore financial centres from Jersey to the Cayman Islands are rallying to defend their privileges from a wave of international action against secrecy. Nick Mathiason reportsHome to 16	nao	130	sim	83	sim	45	sim	75	nao	75	nao
BUSINESS_18	ID: BUSINESS_18 Title: It should be the environment and the economy, stupid Description: In the last part of our series on fixing the financial crisis, we look at how green policy can save the planet as well as economiesHopes that Gordon Brown and other w	nao	225	nao	129	nao	129	nao	76	nao	76	nao
BUSINESS_19	ID: BUSINESS_19 Title: Fantasy boardroom Description: We canvassed opinion about the qualities needed for the perfect boardroom. Suggestions varied from obvious to outlandish	sim	226	nao	130	nao	130	nao	77	nao	77	nao
BUSINESS_2	ID: BUSINESS_2 Title: Rivals waiting to pick up the parcels if Royal Mail can't deliver Description: Latest round of strikes could give other carriers a window of opportunityWith a 24-hour strike set to upset mail deliveries in London on Wednesday, the u	sim	227	nao	118	nao	118	nao	60	nao	60	nao
BUSINESS_20	ID: BUSINESS_20 Title: Norman and Leighton vie to be M&S chairman Description: Allan Leighton and Archie Norman have thrown their hats in the ring to replace Sir Stuart Rose as chairman of Marks &	sim	228	nao	131	nao	131	nao	78	nao	78	nao

	Spencer.Norman and Leighton, the "dream team" that led the											
BUSINESS_21	ID: BUSINESS_21 Title: Lib Dems want fraud office probe into RBS Description: The Serious Fraud Office is being asked to launch a criminal investigation into the collapse of Royal Bank of Scotland, rescued by the British government at the height of the ba	nao	229	nao	132	nao	132	nao	79	nao	79	nao
BUSINESS_22	ID: BUSINESS_22 Title: Johnny Rotten helps boost Dairy Crest Description: Dairy Crest reports 10% sales rise and disposal of Yoplait stakeAn advertising campaign featuring former Sex Pistols frontman Johnny Rotten, aka John Lydon, has helped to boost sale	nao	230	nao	133	nao	133	nao	80	nao	80	nao
BUSINESS_23	ID: BUSINESS_23 Title: House prices fall by 16.5% Description: • House prices have now fallen for 18 consecutive months• Average price is same as it was in September 2004House prices in England and Wales fell by 2% in February, pushing the annual rate of	nao	231	nao	134	nao	134	nao	81	nao	81	nao
BUSINESS_24	ID: BUSINESS_24 Title: Two more quit AIG as depth of bonus anger emerges Description: Two senior European bosses have resigned from AIG citing a "hostile" environment as the insurer faces a public backlash over multimillion-pound bonus payouts.Mauro Gabri	sim	132	sim	91	sim	46	sim	82	nao	82	nao
BUSINESS_25	ID: BUSINESS_25 Title: Mervyn King refuses to bail out car industry Description: • Minister met governor to request financial aid • Unions say 1m workers could desert LabourThe Bank of England has ruled out providing billions of pounds of aid to car firms	nao	232	nao	135	nao	135	nao	83	nao	83	nao
BUSINESS_26	ID: BUSINESS_26 Title: All today's stories Description:	nao	233	nao	136	nao	136	nao	84	nao	84	nao
BUSINESS_27	ID: BUSINESS_27 Title: BAA ordered to sell Gatwick and Stansted Description: Competition Commission breaks up airport operator after two-year inquiry and switches priority to passengersThe Competition Commission further disrupted the government's airport	sim	234	nao	137	nao	137	nao	85	nao	85	nao
BUSINESS_28	ID: BUSINESS_28 Title: Geithner: 'New rules of the game' for Wall Street Description: Obama administration's six-point plan proposes federal supervision of hedge funds and derivatives trading and powers to seize troubled financial institutionsRunaway risk	sim	235	nao	138	nao	138	nao	86	nao	86	nao
BUSINESS_3	ID: BUSINESS_3 Title: Spending and house prices pick up Description: • July retail sales 3.6% higher year-on-year• Surveyors regaining optimism on house pricesStronger spending on the high street and a pick up in activity in the housing market provided fr	sim	237	nao	119	nao	119	nao	61	nao	61	nao
BUSINESS_31	ID: BUSINESS_31 Title: Madoff accountant charged with fraud Description: Bernard Madoff's long-time accountant has been released on $2.5 million bail after being arrested on fraud charges today.David Friehling, 49, is accused of helping the disgraced mone	nao	239	nao	141	nao	141	nao	89	nao	89	nao
BUSINESS_32	ID: BUSINESS_32 Title: Shell dumps wind, solar and hydro Description: Shell will no longer invest in renewable technologies such as wind, solar and hydro power because they are not economic, the Anglo-Dutch oil company said today. It plans to invest more	sim	240	nao	142	nao	142	nao	90	nao	90	nao
BUSINESS_33	ID: BUSINESS_33 Title: Slumping billionaires: financial crisis slashes ranks of world's super rich Description: • Billion dollar club cut from 1,125 members to 793 • Bill Gates claims back No 1 spot from Warren BuffettOne of the world's most exclusive clu	nao	134	sim	110	sim	27	sim	91	nao	91	nao
BUSINESS_34	ID: BUSINESS_34 Title: Inflation or deflation? Description: Opinion is divided about where the economy goes from here. The Guardian Money team offers suggestions on what to do for either case• If you think inflation is on the way …SavingsBuy National Savi	sim	241	nao	88	sim	2	sim	92	nao	92	nao
BUSINESS_38	ID: BUSINESS_38 Title: Savers withdraw cash as returns decline Description: • Savings deposits fell by £100m in February, says BBA• Mortgage approvals rise but remains subduedSavers continued to withdraw money from bank accounts in February as falling int	nao	245	nao	145	nao	145	nao	95	nao	95	nao
BUSINESS_39	ID: BUSINESS_39 Title: Average weekly pay falls 2.2% Description: • Bonus payments have been reduced sharply• First pay decline since records began in 2001Sharply reduced bonus payments across Britain's finance industry have triggered the first decline in	nao	35	sim	76	sim	48	sim	96	nao	96	nao
BUSINESS_4	ID: BUSINESS_4 Title: Rise in Heathrow passengers signals soft landing for BAA Description: • 6.5m passengers used Heathrow in July, up 1% on last year• Group numbers down 2.4% - but decline has slowedHeathrow reported its third-busiest month ever today,	nao	246	nao	120	nao	120	nao	62	nao	62	nao
BUSINESS_40	ID: BUSINESS_40 Title: New chairman Sir Win Bischoff backs Lloyds chief Description: Bischoff spent most of his career with Schroders and was appointed chairman in 1995Veteran banker Sir Win Bischoff today threw his support behind the embattled Lloyds Ban	nao	247	nao	146	nao	146	nao	97	nao	97	nao
BUSINESS_41	ID: BUSINESS_41 Title: FTSE 100 scrapes to 11-day record Description: Choppy trading in silly summer volumes has seen the FTSE 100 nip into positive territory just in time for the close and achieve that record-matching 11-day winning streak.Only just, how	nao	248	nao	147	nao	147	nao	98	nao	98	nao
BUSINESS_42	ID: BUSINESS_42 Title: Consortium expects to retain National Express's profitable contracts Description: Government could block precondition of Cosmen/CVC offer for ailing transport group, as rival Stagecoach tries to join the frayThe transport secretary,	nao	38	sim	52	sim	49	sim	99	nao	99	nao
BUSINESS_43	ID: BUSINESS_43 Title: FRC review looks to 'engage' shareholders and companies Description: • Changes recommended to City's code of boardroom behaviour• Relationship between investors and companies addressedCompanies may be required to "apply or explain"	nao	249	nao	148	nao	148	nao	100	nao	100	nao
BUSINESS_44	ID: BUSINESS_44 Title: Darling threatens banks with investigation to encourage lending Description: • Chancellor concerned over some mortgages' high profits• Further meeting with banks in September will	nao	250	nao	149	nao	149	nao	101	nao	101	nao

	step up pressureAlistair Darling appeared to threate											
BUSINESS_45	ID: BUSINESS_45 Title: Coca-Cola trials sweet, fizzy, milky 'vibrancy' drink in three US cities Description: Soft drinks giant launches new Vio drink in New York but no word yet on whether it will reach the UKIt may not quite sound the real thing but cons	nao	251	nao	150	nao	150	nao	102	nao	102	nao
BUSINESS_46	ID: BUSINESS_46 Title: Tribal leader appeals to Vedanta shareholders to oppose Indian mine Description: Bauxite mine plan threatens holy mountainActivists buy single shares to talk to annual meetingProtesters lined up outside the annual meeting of British	nao	39	sim	151	nao	151	nao	103	nao	103	nao
BUSINESS_47	ID: BUSINESS_47 Title: Government grants Vestas £6m – but factory will still close Description: The government yesterday awarded Vestas Technology £6m but the cash will not stop the Danish turbine manufacturer from controversially shutting its Isle of Wig	nao	252	nao	152	nao	152	nao	104	nao	104	nao
BUSINESS_48	ID: BUSINESS_48 Title: Ryanair reduces profit forecast Description: Shares fall 8.8% as budget airline warns investors that slashing prices will eat into profitability this yearRyanair warned of a tough winter for airlines today and admitted that fare cut	sim	253	nao	153	nao	153	nao	105	nao	105	nao
BUSINESS_49	ID: BUSINESS_49 Title: Pearson shrugs off plunging FT profits Description: • FT profits fall 40% in advertising downturn• Penguin books earnings down 23%Pearson shrugged off a 40% fall in first-half operating profits at FT Publishing today with better-th	sim	254	nao	154	nao	154	nao	106	nao	106	nao
BUSINESS_5	ID: BUSINESS_5 Title: BT dismisses O2 call fee warning Description: • Mobile phone company charged with scaremongering over Ofcom submission • 'Sky won't fall' if termination rates are scrappedBT has attacked as scaremongering a warning from O2, the UK's	sim	255	nao	121	nao	121	nao	63	nao	63	nao
BUSINESS_50	ID: BUSINESS_50 Title: ITV to offload Friends Reunited Description: • Social networking site bought by ITV for £175m• Peter Dubens understood to be in talks over £15m offerInternet entrepreneur Peter Dubens has emerged as a potential bidder for Friends Re	sim	256	nao	155	nao	155	nao	107	nao	107	nao
BUSINESS_51	ID: BUSINESS_51 Title: All today's business stories Description:	nao	257	nao	156	nao	156	nao	108	nao	108	nao
BUSINESS_52	ID: BUSINESS_52 Title: Why August is the cruellest month of all Description: Stockmarkets are recovering, property values are on the rise, and shoppers are out in force. So is it a good time for Britain's hard-pressed bosses to go abroad and relax? Sadly,	nao	258	nao	56	sim	50	sim	109	nao	109	nao
BUSINESS_53	ID: BUSINESS_53 Title: Buffett to star in children's cartoon Description: The Secret Millionaires Club to feature an animated character of the billionaire investor giving advice on the art of financeThe so-called Sage of Omaha, Warren Buffett, is a billio	sim	259	nao	157	nao	157	nao	110	nao	110	nao
BUSINESS_6	ID: BUSINESS_6 Title: Morgan Stanley's life lessons Description: With market players questioning when the stockmarket rally will run out of steam - or if it already has - equity strategists at Morgan Stanley have produced some timely research.Teun Draaism	nao	260	nao	122	nao	122	nao	64	nao	64	nao
BUSINESS_7	ID: BUSINESS_7 Title: Banks and building societies lift savings rates Description: Interest rates on certain fixed-term savings accounts have increased by more than 50% since MarchSavers are being offered rates of almost 5% above the Bank of England base	nao	261	nao	123	nao	123	nao	65	nao	65	nao
BUSINESS_8	ID: BUSINESS_8 Title: China spying claims dent Rio Tinto shares Description: Case has raised tensions between Australia and ChinaShares in Rio Tinto fell more than 3% on Monday as the market took fright at accusations that the mining company had spied on	nao	262	nao	124	nao	124	nao	66	nao	66	nao
BUSINESS_9	ID: BUSINESS_9 Title: Cattles agrees sale of subsidiary to recoup bad debts Description: • Sub-prime lender hit by £720m bad debts earlier this year• Sale of scandal-free Cattles Invoice Financing worth £70mCattles, the troubled sub-prime lender that disc	nao	263	nao	125	nao	125	nao	67	nao	67	nao
CULTURE_0	ID: CULTURE_0 Title: The Stone Roses during the recording of Fools Gold Description: Watch the Manchester band at work (ie dicking around on a boat) during the recording of their classic Fools Gold single	sim	73	sim	9	sim	51	sim	8	sim	20	sim
CULTURE_10	ID: CULTURE_10 Title: Aerosmith's Steven Tyler airlifted to hospital Description: The singer of the hard-rock legends suffered head, neck and shoulder injuries after falling off the stage at a concert last nightAerosmith were forced to cut short their per	nao	265	nao	165	nao	165	nao	118	nao	118	nao
CULTURE_11	ID: CULTURE_11 Title: Will Nick Cave's all-singing e-book change the novel for ever? Description: There's one big question you need to ask when presented with the technological enhancement of an art form: "Is it Smell-O-Vision?" Remember Smell-O-Vision? N	nao	266	nao	166	nao	166	nao	119	nao	119	nao
CULTURE_12	ID: CULTURE_12 Title: Mommas of the poppers Description: Critical maulings? Rock'n'roll excess? Clean socks? Parents of pop stars have many things to worry about these days. Stephen Kelly talks to five mothers about watching their 'wee love cups' from the	nao	76	sim	25	sim	53	sim	17	sim	22	sim
CULTURE_13	ID: CULTURE_13 Title: Prom 30 – BBCSO/Knussen Description: Royal Albert Hall, LondonAlmost unnoticed, last week's proms included a miniature Respighi season, spread across three evenings. The works that make up his Roman trilogy were featured in successiv	sim	267	nao	167	nao	167	nao	120	nao	120	nao
CULTURE_14	ID: CULTURE_14 Title: Radiohead: Harry Patch (In Memory of) Description: Thom Yorke's tribute to Britain's last surviving first world war veteran, who died last month, finds Radiohead at their most understated and sereneThose who tuned into Radio 4 this m	sim	268	nao	168	nao	168	nao	121	nao	121	nao
CULTURE_15	ID: CULTURE_15 Title: Behind the music: Who needs major labels? Description: Unsigned acts such as the Boxer Rebellion and Ani DiFranco prove it's possible to launch an international career independently -	sim	269	nao	169	nao	169	nao	122	nao	122	nao

	but it requires savvy management and a strong wor											
CULTURE_16	ID: CULTURE_16 Title: Isabelle Huppert: 'It's called Home, but it becomes hell' Description: The French cinema icon tells Andrew Pulver about how she came to star in rising director Ursula Meier's film, and why Home should be seen as an allegory on the wa	sim	270	nao	170	nao	170	nao	123	nao	123	nao
CULTURE_17	ID: CULTURE_17 Title: The Ugly Truth about romcoms Description: The truth is that the genre can't handle the intrusion of the real worldThe truth, ugly or otherwise, is a dangerous concept for a romcom to tangle with. The point of the format is surely to	nao	271	nao	171	nao	171	nao	124	nao	124	nao
CULTURE_18	ID: CULTURE_18 Title: Missing in action Description: He made his name in the 90s with the breathtaking Reservoir Dogs and Pulp Fiction. But as Quentin Tarantino's chaotic Second World War epic opens, Sean O'Hagan finds the director's genius in danger of g	nao	272	nao	172	nao	172	nao	125	nao	125	nao
CULTURE_19	ID: CULTURE_19 Title: John Hughes, bard of my backyard Description: The early films of The Breakfast Club director didn't just speak to me, they were about my home town. What a pity that he later traded truth for clicheThe success of John Hughes's 1980s t	sim	273	nao	173	nao	173	nao	126	nao	126	nao
CULTURE_20	ID: CULTURE_20 Title: Oscars ceremony should be first, says Meryl Streep Description: It's the Big Kahuna, not the caboose, says 15-time Academy Award nomineeShe has garnered more Oscar nominations than any other actor, so when Meryl Streep declares that	nao	274	nao	174	nao	174	nao	127	nao	127	nao
CULTURE_21	ID: CULTURE_21 Title: Hollywood searches for escapism after the apocalypse Description: A spate of new US movies portrays a scary life on a post-cataclysm Earth. Paul Harris in New York reports on a taste for doom-laden films that reflects anxiety about s	nao	275	nao	175	nao	175	nao	128	nao	128	nao
CULTURE_22	ID: CULTURE_22 Title: Ride on time Description: In The Time Traveler's Wife a librarian goes on a romantic journey through the past. Why don't movie time travellers do the sensible thing and hit the bookies, asks David StubbsWorking as a shelf stacker dur	nao	276	nao	176	nao	176	nao	129	nao	129	nao
CULTURE_23	ID: CULTURE_23 Title: Vincent Cassel on French gangster Mesrine: 'He was a showman' Description: The star of La Haine, Irreversible and the biopic Mesrine on what it was like to grow up where the charismatic criminal lived and died	sim	277	nao	177	nao	177	nao	130	nao	130	nao
CULTURE_24	ID: CULTURE_24 Title: Reel review on Orphan: 'A rather good bad seed' Description: Potential foster parents should stay away, but for the rest of us this evil kiddie horror is surprisingly good fun, says Catherine Shoard	nao	278	nao	178	nao	178	nao	131	nao	131	nao
CULTURE_25	ID: CULTURE_25 Title: From fall guy to comic star Description: Not long before he starred with Simon Pegg in Shaun of the Dead, he was earning £1.92 an hour as the 'wittiest waiter in the world'. Now, as his new film The Boat That Rocked opens, he t	nao	279	nao	179	nao	179	nao	132	nao	132	nao
CULTURE_26	ID: CULTURE_26 Title: This time I've come to bury Cool Britannia Description: American journalist Stryker McGuire wrote the magazine article that initiated the 'Cool Britannia' phenomenon in 1996. Back then, the City was the engine of our prosperity, Brit	nao	280	nao	180	nao	180	nao	133	nao	133	nao
CULTURE_27	ID: CULTURE_27 Title: Why it pays to be alone with the truly great works of art Description: Who wants to jostle with blockbuster crowds for a fleeting view of a masterpiece? Great artworks deserve more of our time - which is why Laura Cumming often visit	nao	54	sim	84	sim	54	sim	134	nao	134	nao
CULTURE_28	ID: CULTURE_28 Title: The priapic president laid bare Description: From Marilyn to a Mafia moll, Kennedy's conquests are breathlessly revisited. But to what purpose, asks Sean O'HaganIn 1947, Robert Penn Warren won the Pulitzer Prize for All the King's Me	nao	281	nao	181	nao	181	nao	135	nao	135	nao
CULTURE_29	ID: CULTURE_29 Title: A few pearls and many a pig's ear Description: The V&A's assessment of the baroque period throws up some fascinating curios amid the swaggering ornamental excessWas Bach orgasmic? This important question is not answered in the V&A's	nao	282	nao	182	nao	182	nao	136	nao	136	nao
CULTURE_32	ID: CULTURE_32 Title: The guest list Description: Petrolhead Vin Diesel promotes his new Fast & Furious movie, and Richard Madeley switches sides to become Piers Morgan's interviewerTuesdayPaul O'Grady, 5pm, C4Today's guests are Alison "just a top-up, Ton	nao	286	nao	185	nao	185	nao	139	nao	139	nao
CULTURE_33	ID: CULTURE_33 Title: Potter pirated Description: Harry Potter author is among writers shocked to discover their books available as free downloadsThe publishers of bestselling authors JK Rowling, Aravind Adiga and Ken Follett have been shocked by the news	nao	287	nao	186	nao	186	nao	140	nao	140	nao
CULTURE_4	ID: CULTURE_4 Title: The myth of Daniel Johnston's genius Description: Superlative praise is just one of the many ways the great outsider artist Daniel Johnston has been done a disserviceHe may not chase fame, but cult songwriter Daniel Johnston is curren	nao	288	nao	160	nao	160	nao	113	nao	113	nao
CULTURE_5	ID: CULTURE_5 Title: When underground punk meets Stars in Their Eyes Description: Our Band Could BBQ Your Life is a festival where British acts emulate their indie idols from the 80s US undergroundMission of Burma sit outside the entrance of the Windmill	nao	289	nao	161	nao	161	nao	114	nao	114	nao
CULTURE_6	ID: CULTURE_6 Title: Spider-Man the musical hangs by a thread Description: Production has been halted over cashflow crisis, showbiz magazine saysA forthcoming Spider-Man musical, set to be one of the biggest shows in Broadway history, has reportedly run i	nao	290	nao	162	nao	162	nao	115	nao	115	nao
CULTURE_7	ID: CULTURE_7 Title: Raygun apologise for viral interview Description: The London indie-pop newbies have issued an 'apology' on MySpace that shows considerable more self-awareness than their notorious 4Music interviewRaygun have spoken for the first time	nao	291	nao	163	nao	163	nao	116	nao	116	nao

CULTURE_8	ID: CULTURE_8 Title: Michael Jackson's last recordings seized by sister LaToya Description: The late singer's sister has allegedly taken possession of hard drives containing unreleased material Jackson recorded with artists such as Ne-Yo, Akon and Will.I.	sim	292	nao	164	nao	164	nao	117	nao	117	nao
CULTURE_9	ID: CULTURE_9 Title: Bob Dylan to release Christmas album Description: Joy to the world! His Bobness is recording an album of traditional Christmas carols and original festive compositionsBob Dylan is recording his first ever Christmas album, according to	sim	293	nao	27	sim	3	sim	19	sim	1	sim
ENVIRONMENT_10	ID: ENVIRONMENT_10 Title: UN panel to study climate impact on poor nations Description: Intergovernmental Panel on Climate Change determined to increase understanding of regional effects of warmingThe Intergovernmental Panel on Climate Change (IPCC), the	nao	296	nao	195	nao	195	nao	149	nao	149	nao
ENVIRONMENT_11	ID: ENVIRONMENT_11 Title: The plight of Britain's ancient trees Description: We are home to some 100,000 of the oldest trees in Europe. But is our neglect and ill-treatment in danger of killing them off?Above crumpled grey roots like the enormous feet of	nao	297	nao	196	nao	196	nao	150	nao	150	nao
ENVIRONMENT_12	ID: ENVIRONMENT_12 Title: Infrastructure woes hamper China wind farms' push for profitability Description: Rapid build-out of capacity has caused bottlenecks in connecting turbines to the grid. From BusinessGreen.com, part of the Guardian Environment Netw	nao	298	nao	197	nao	197	nao	151	nao	151	nao
ENVIRONMENT_13	ID: ENVIRONMENT_13 Title: 'Are we here just for your amusement?' Description: Our increasing demand for adventure is pushing back the frontiers of tourism, but is it also posing a threat to tribal people? John Vidal investigatesWhen the Jarawa tribe of hu	nao	299	nao	198	nao	198	nao	152	nao	152	nao
ENVIRONMENT_14	ID: ENVIRONMENT_14 Title: Activists call on Vedanta investors to oppose mine on holy site in India Description: Local councils and the Church of England will come under fire for holding shares in the mining group which is opening a new mine in forests on	nao	300	nao	199	nao	199	nao	153	nao	153	nao
ENVIRONMENT_15	ID: ENVIRONMENT_15 Title: Green spending in UK economic rescue package 'negligible' Description: UK government's fiscal stimulus package is a missed opportunity, says New Economics Foundation, as less than 1% of anti-recession spending goes on environment	nao	301	nao	200	nao	200	nao	154	nao	154	nao
ENVIRONMENT_16	ID: ENVIRONMENT_16 Title: Living walls and green roofs pave way for biodiversity in new building Description: Otters could return to urban rivers, bats could roost under bridges, swifts could flock to office blocks and peregrine falcons soar above cathedr	nao	55	sim	201	nao	201	nao	155	nao	155	nao
ENVIRONMENT_2	ID: ENVIRONMENT_2 Title: Greenpeace threatens E.ON with legal action over nuclear reactors Description: • Move triggered by reports of preparatory bore holes• EO.N claims work is to 'make sure the ground is suitable'Greenpeace is threatening to take legal	nao	305	nao	189	nao	189	nao	143	nao	143	nao
ENVIRONMENT_20	ID: ENVIRONMENT_20 Title: Water meters in all homes by 2030 to ease shortages Description: Report calls for strict controls as climate change threatens to dry up British riversMany parts of the country face crippling water shortages in the near future unl	nao	306	nao	205	nao	205	nao	159	nao	159	nao
ENVIRONMENT_21	ID: ENVIRONMENT_21 Title: Nuclear reactor plans spread fear and fission along the Energy Coast Description: Beauty spot may soon become part of a 'Lake District Nuclear Park'The unmarked white van at an isolated farm on the fringes of the Lake District Na	nao	56	sim	206	nao	206	nao	160	nao	160	nao
ENVIRONMENT_22	ID: ENVIRONMENT_22 Title: Global lights out for Earth Hour Description: For a symbolic hour global landmarks from Sydney to London were in darkness to highlight concern over climate changeLights went out across the world tonight to mark Earth Hour, the bi	nao	307	nao	207	nao	207	nao	161	nao	161	nao
ENVIRONMENT_23	ID: ENVIRONMENT_23 Title: 'Kettled' anarchists increase worry for G20 demonstrators Description: Singling out bandana-wearing contingent for special treatment not the right way to police protestNews that police have stopped a plan to set off explosives du	sim	57	sim	77	sim	4	sim	52	sim	2	sim
ENVIRONMENT_24	ID: ENVIRONMENT_24 Title: Syrian hunters threaten sociable lapwing Description: Research from the RSPB has revealed that one of the world's rarest birds has become a hunting targetNews that one of the world's rarest birds, the sociable lapwing is now unde	nao	308	nao	208	nao	208	nao	162	nao	162	nao
ENVIRONMENT_25	ID: ENVIRONMENT_25 Title: Carbon Trust launches new guide to overcoming board resistance Description: Step-by-step guide aims to help green executives pitch projects to the board. From BusinessGreen.com, part of Guardian Environment NetworkAnyone who has	nao	309	nao	209	nao	209	nao	163	nao	163	nao
ENVIRONMENT_26	ID: ENVIRONMENT_26 Title: Charleaders must cool enthusiasm for settting fire to the planet Description: Reactions to my 'biochar' stance got a lot of people fired up, but I was too soft on one champion of so-called developmentWell that got 'em going. So f	nao	310	nao	210	nao	210	nao	164	nao	164	nao
ENVIRONMENT_27	ID: ENVIRONMENT_27 Title: Heathrow third runway plans dealt massive blow Description: Planning application cannot be lodged before general election, meaning Tories could scrap schemeThe chances of a third runway being built at Heathrow airport have been d	nao	311	nao	211	nao	211	nao	165	nao	165	nao
ENVIRONMENT_28	ID: ENVIRONMENT_28 Title: US wilderness conservation law hailed as 'new dawn for American heritage' Description: California's Sierra Nevada and Jefferson National Forest in Virginia among 2m acres of land to get highest level of protectionThe US Congress	nao	58	sim	92	sim	57	sim	166	nao	166	nao
ENVIRONMENT_29	ID: ENVIRONMENT_29 Title: UK prison mattresses to be recycled Description: Some 50,000 prison mattresses are replaced every year but could be recycled under initiative to save money and waste lessToday's discarded prison mattress could be tomorrow's garde	nao	59	sim	93	sim	58	sim	167	nao	167	nao
ENVIRONMENT_3	ID: ENVIRONMENT_3 Title: Big Green Gathering festival cancelled after 'political pressure' from police Description: Last-minute injunction prompts organisers to accuse police and council of 'premeditated	nao	108	sim	53	sim	55	sim	39	sim	23	sim

	decision to prevent gathering of radicals'Police to											
ENVIRONMENT_30	ID: ENVIRONMENT_30 Title: Opposing wind farms should be taboo - minister Description: Opposition to wind farms should become as socially unacceptable as failing to wear a seatbelt, Ed Miliband, the climate change secretary, has said.Speaking at a screenin	sim	312	nao	212	nao	212	nao	168	nao	168	nao
ENVIRONMENT_31	ID: ENVIRONMENT_31 Title: Clear labels needed to end 'greenwash' Description: Tough standards on labelling should be enforced by the government to clamp down on "greenwash", in which companies exaggerate the environmental credentials of their products, a	nao	313	nao	213	nao	213	nao	169	nao	169	nao
ENVIRONMENT_32	ID: ENVIRONMENT_32 Title: Alaskan volcano Mount Redoubt erupts Description: Observatory issues 'red alert' after four explosions send smoke 15,000 metres into the airAn Alaskan volcano 100 miles (160km) south-west of the state's largest city, Anchorage, e	sim	314	nao	214	nao	214	nao	170	nao	170	nao
ENVIRONMENT_33	ID: ENVIRONMENT_33 Title: Mild winters aid long-tailed tit numbers Description: • Record number of participants in January's garden bird survey• Read the full results on our DatablogThe long-tailed tit has emerged as the surprise success story of the RSPB	nao	315	nao	215	nao	215	nao	171	nao	171	nao
ENVIRONMENT_34	ID: ENVIRONMENT_34 Title: Boost for huge Scots tide-power plan Description: Scotland plan's to host the world's largest tidal energy project have moved a step closer after Norwegian renewables giant Statkraft joined the consortium backing the £250m	sim	316	nao	216	nao	216	nao	172	nao	172	nao
ENVIRONMENT_35	ID: ENVIRONMENT_35 Title: Obama may delay signing up to Copenhagen climate deal Description: Barack Obama may be forced to delay signing up to a new international agreement on climate change in Copenhagen at the end of the year because of the scale of opp	nao	317	nao	217	nao	217	nao	173	nao	173	nao
ENVIRONMENT_36	ID: ENVIRONMENT_36 Title: Maldives' carbon neutral plan is not greenwash, just imperfect progress Description: Proposals to cut emissions in the Maldives don't include aviation, but European emissions trading will help offset tourist CO2Mark Lynas's book	sim	318	nao	218	nao	218	nao	174	nao	174	nao
ENVIRONMENT_37	ID: ENVIRONMENT_37 Title: State intervention vital if UK to meet targets, says former BP boss Description: • Browne says markets need new strategic direction • Consumers will have to pay more for renewablesBritain must revert to greater state control of e	nao	319	nao	219	nao	219	nao	175	nao	175	nao
ENVIRONMENT_4	ID: ENVIRONMENT_4 Title: US satellites reveal true extent of melting polar summer ice Description: Photos from US spy satellites declassified by the Obama administration provide the first graphic images of how the polar ice sheets are retreating	nao	320	nao	190	nao	190	nao	144	nao	144	nao
ENVIRONMENT_5	ID: ENVIRONMENT_5 Title: Cycling manners more rewarding Description: Cycling in city traffic doesn't have to be a nightmare – unexpected considerate drivers and pleasant exchanges can even make it funThe white van drove in slow and close, its front window	nao	321	nao	191	nao	191	nao	145	nao	145	nao
ENVIRONMENT_9	ID: ENVIRONMENT_9 Title: Brazil complaint over UK waste export Description: • Wiltshire police arrest three men in connection with alleged 99 containers• Environment Agency says it will return waste to the UK and prosecute those responsibleThe Brazilian g	nao	324	nao	194	nao	194	nao	148	nao	148	nao
HEALTH_0	ID: HEALTH_0 Title: The Doctor's World: Seeking Lessons in Swine Flu Fight Description: A couple with flu symptoms waited last month for medical attention in an isolated section of the Roosevelt Hospital in Guatemala City.	sim	325	nao	220	nao	220	nao	176	nao	176	nao
HEALTH_10	ID: HEALTH_10 Title: Doctor and Patient: Are Patients in Part to Blame When Doctors Miss the Diagnosis? Description:	sim	327	nao	29	sim	62	sim	21	sim	28	sim
HEALTH_11	ID: HEALTH_11 Title: Governors Fear Added Costs in Health Care Overhaul Description: Quincy Proctor has had a series of strokes. His financing for occupational therapy at a Seattle adult health center was cut off.	sim	23	sim	225	nao	225	nao	181	nao	181	nao
HEALTH_12	ID: HEALTH_12 Title: Senators Hear Concerns Over Costs of Health Proposal Description: From left, Senators Charles E. Schumer, Harry Reid, Richard J. Durbin and Patty Murray, all Democrats, on Thursday discussed what they called organized efforts by right	nao	24	sim	226	nao	226	nao	182	nao	182	nao
HEALTH_13	ID: HEALTH_13 Title: Democrats Say No to Cost Cap for Drug Makers Description: Squeezing more money out of the health care system, and from the drug industry, is still a goal for top Democrats.	nao	25	sim	227	nao	227	nao	183	nao	183	nao
HEALTH_14	ID: HEALTH_14 Title: Television Viewing Linked to Blood Pressure Increases in Children Description: Children who watched a lot of television had higher blood pressure readings regardless of whether they were more sedentary over all, researchers reported.	sim	328	nao	28	sim	63	sim	20	sim	29	sim
HEALTH_15	ID: HEALTH_15 Title: With High-Profile Death, Focus on High-Risk Drug Description: Propofol is the focus of inquiries into Michael Jackson’s death.	sim	329	nao	26	sim	64	sim	18	sim	30	sim
HEALTH_16	ID: HEALTH_16 Title: Hispanics Who Move to U.S. Face Higher Cancer Rates Description: Cuban-Americans experienced the most dramatic cancer increases after moving to Florida, while Mexican-Americans experienced the least.	sim	330	nao	30	sim	65	sim	22	sim	31	sim
HEALTH_17	ID: HEALTH_17 Title: Quick Tests for the Flu Found Often Inaccurate Description: Widely used rapid checks for the H1N1 virus — or any flu, for that matter — fail more than 50 percent of the time.	sim	331	nao	32	sim	66	sim	24	sim	32	sim
HEALTH_18	ID: HEALTH_18 Title: Studies Question Using Cement for Spine Injuries Description: Research into effectiveness of a procedure in use since the 1990s comes as the Obama administration calls for assessing how health care money is spent.	sim	26	sim	228	nao	228	nao	184	nao	184	nao
HEALTH_19	ID: HEALTH_19 Title: Vital Signs: Viral Infection May Explain Racial Differences in Oral Cancer Death	nao	332	nao	229	nao	229	nao	185	nao	185	nao

	Rates Description: African-American patients with head and neck cancers die earlier than whites, researchers say.											
HEALTH_20	ID: HEALTH_20 Title: Findings May Explain Gap in Cancer Survival Description: Scientists may have found some clues as to why white patients often outlive blacks even when they have what appear to be the same cancers.	sim	333	nao	230	nao	230	nao	186	nao	186	nao
HEALTH_21	ID: HEALTH_21 Title: Psychologists Reject Gay 'Therapy' Description: In a resolution, the American Psychological Association rejected reparative therapy, which says gay men and lesbians can change their sexual orientation.	sim	334	nao	33	sim	67	sim	25	sim	33	sim
HEALTH_22	ID: HEALTH_22 Title: Medical Papers by Ghostwriters Pushed Therapy Description: Court documents suggest a broad level of hidden industry influence on medical literature.	sim	335	nao	36	sim	68	sim	27	sim	34	sim
HEALTH_23	ID: HEALTH_23 Title: White House Affirms Deal on Drug Cost Description: The White House said it stood by a deal to shield drug makers from further costs in the health care overhaul.	sim	27	sim	231	nao	231	nao	187	nao	187	nao
HEALTH_24	ID: HEALTH_24 Title: Centrist Democrats Upbeat on Health Care Bill Description: Other Democratic senators voiced concern that they would be forced to defend tax increases in the House version of the bill.	sim	28	sim	232	nao	232	nao	188	nao	188	nao
HEALTH_27	ID: HEALTH_27 Title: Cases: A Pungent Life: The Smells in My Head Description:	nao	337	nao	1	sim	69	sim	1	sim	35	sim
HEALTH_28	ID: HEALTH_28 Title: Personal Health: Taking Steps to Cope With Chemo Brain Description:	sim	31	sim	4	sim	7	sim	4	sim	5	sim
HEALTH_29	ID: HEALTH_29 Title: Really?: The Claim: Some Dogs Look Like Their Owners Description:	sim	338	nao	5	sim	70	sim	5	sim	36	sim
HEALTH_30	ID: HEALTH_30 Title: Doctor and Patient: Treating Patients as Partners, by Way of Informed Consent Description:	sim	339	nao	46	sim	71	sim	34	sim	37	sim
HEALTH_34	ID: HEALTH_34 Title: Essay: A Doctor by Choice, a Businessman by Necessity Description:	sim	342	nao	61	sim	75	sim	43	sim	41	sim
HEALTH_35	ID: HEALTH_35 Title: Cases: Losing a Comforting Ritual: Treatment Description:	nao	343	nao	64	sim	76	sim	45	sim	42	sim
HEALTH_36	ID: HEALTH_36 Title: Mind: Where Can the Doctor Who's Guided All the Others Go for Help? Description:	nao	344	nao	65	sim	77	sim	46	sim	43	sim
HEALTH_37	ID: HEALTH_37 Title: I will die the most horrible death Description: As a reporter for NBC, Charles Sabine had experienced the horrors of war. But that was nothing compared with discovering he had inherited Huntington's diseaseI was travelling with a came	sim	41	sim	69	sim	78	sim	191	nao	191	nao
HEALTH_38	ID: HEALTH_38 Title: The shoes that mimic running barefoot Description: A new range of shoes try to reproduce the feel of running without any footwear. We try them outThe barefoot running movement has built up serious momentum over the last few years, as	nao	42	sim	70	sim	79	sim	49	sim	44	sim
HEALTH_39	ID: HEALTH_39 Title: Don't give swine flu drugs like Tamiflu to under-12s, says study Description: • Side-effects said to outweigh benefits• Government queries relevance of researchChildren under the age of 12 should not be given Tamiflu or Relenza, the t	sim	43	sim	6	sim	80	sim	192	nao	192	nao
HEALTH_4	ID: HEALTH_4 Title: Mentally Ill Offenders Strain Juvenile System Description: An inmate at the Ohio River Valley Juvenile Correctional Facility. Two-thirds of the nation's juvenile inmates have at least one mental illness, according to surveys.	nao	345	nao	221	nao	221	nao	177	nao	177	nao
HEALTH_40	ID: HEALTH_40 Title: Full list of swine flu cases, country by country Description: Get all the cases, suspected and confirmed, in our up-to-date spreadsheet• Scroll down for sortable table• Swine flu cases where you liveThe World Health Organisation has d	sim	44	sim	10	sim	30	sim	193	nao	193	nao
HEALTH_41	ID: HEALTH_41 Title: Late motherhood: time for a vital wake-up call Description: With more and more couples deciding to put off parenthood until their 30s and 40s, doctors are calling for a cultural rethink on the whole issue. Girls should be told about	sim	45	sim	16	sim	28	sim	194	nao	194	nao
HEALTH_42	ID: HEALTH_42 Title: Fertility warning to women Description: Expert calls for 'fertility MoTs' and educationBritain is facing an infertility timebomb, as couples delay parenthood and damage their chances of having children in later life.Couples are "stick	sim	46	sim	17	sim	81	sim	12	sim	45	sim
HEALTH_43	ID: HEALTH_43 Title: Healthy people are lying to get swine flu drug, warn GPs Description: Fears of Tamiflu shortages as helpline advisers are connedPeople are conning the swine flu helpline into giving them Tamiflu in case they fall ill, Britain's leadin	nao	47	sim	18	sim	82	sim	13	sim	46	sim
HEALTH_44	ID: HEALTH_44 Title: Elderly 'put at risk' by axing of 24-hour wardens | Rajeev Syal and Graham Mole Description: Retired nurse leads fight against councils and housing trusts that have withdrawn full-time careDozens of councils and housing trusts are fac	sim	48	sim	19	sim	83	sim	195	nao	195	nao
HEALTH_45	ID: HEALTH_45 Title: Survivor of the Marchioness pleasure boat talks to Tim Ecott Description: On 20 August 1989, the Marchioness pleasure boat was crushed by the dredger Bowbelle in the River Thames. In less than a minute, 51 of the 132 people on board d	sim	49	sim	20	sim	84	sim	196	nao	196	nao
HEALTH_46	ID: HEALTH_46 Title: My body & soul: Nick Robinson, journalist, 45 Description: Nick Robinson, journalist, 45Are you healthy?My doctor says I'm more healthy than I deserve to be. I'm the man who ought to exercise but never gets round to it. I work too har	nao	346	nao	235	nao	235	nao	197	nao	197	nao
HEALTH_47	ID: HEALTH_47 Title: Girl, two, dies after swine flu misdiagnosis Description: Child with possible meningitis was 'failed by system' say parentsThe parents of a two-year-old girl thought to have died from	nao	50	sim	22	sim	85	sim	15	sim	47	sim

	meningitis after they were told she was suffering											
HEALTH_48	ID: HEALTH_48 Title: The truth about lying: who does it, and why Description: Are human beings by nature duplicitous? Psychologist Robert Feldman reports on what his research reveals about fibbing. Plus Pete Docherty, Katie Price and more come clean about	sim	51	sim	71	sim	86	sim	198	nao	198	nao
HEALTH_49	ID: HEALTH_49 Title: Doctor, doctor: Losing weight and IVF Description: Dr Tom Smith on why staying in shape gets harder as we get older, and what to do when fertility treatment has failedAt 57, my weight has gradually crept up so that my BMI is at the to	nao	347	nao	236	nao	236	nao	199	nao	199	nao
HEALTH_5	ID: HEALTH_5 Title: And You Thought a Prescription Was Private Description: Randee Lonergan says a pharmacy sold her prescription history to a local Target without her knowledge. With Ms. Lonergan are her husband, Kevin, and their children, Brittany, left	sim	348	nao	7	sim	60	sim	6	sim	26	sim
HEALTH_50	ID: HEALTH_50 Title: This column will change your life: Fresh starts Description: Whose life is so perfect they don't think they'd make a better job of it the second time around? asks Oliver BurkemanHow To Disappear Completely And Never Be Found, publishe	sim	52	sim	23	sim	87	sim	200	nao	200	nao
HEALTH_51	ID: HEALTH_51 Title: Celebrities reveal the last lies they told Description: Ricky Gervais, Katie Price, Pete Doherty and more confess the last time they told a lie	sim	349	nao	24	sim	88	sim	16	sim	48	sim
HEALTH_52	ID: HEALTH_52 Title: Men with angina run double the risk of death, compared with women Description: Men with angina are more than twice as likely to suffer a heart attack or die of heart problems compared with women with the same condition, researchers ha	sim	350	nao	31	sim	89	sim	23	sim	49	sim
HEALTH_53	ID: HEALTH_53 Title: Social class still affects infant mortality Description: Nearly 1 in 100 babies die in their first year of life in some parts of England, official figures show. A new study has found that babies are most at risk if they're born in dep	sim	351	nao	37	sim	90	sim	201	nao	201	nao
HEALTH_54	ID: HEALTH_54 Title: Masks in the home may slow spread of swine flu Description: Wearing masks in the home, when someone in the household is infected with flu, may prevent others in the household from catching it, according to a new study. But the researc	nao	352	nao	38	sim	91	sim	28	sim	50	sim
HEALTH_57	ID: HEALTH_57 Title: Small risk of death from obesity surgery Description: Weight-loss surgery is the most effective way for very obese people to lose weight, but it's a major operation with serious risks. A new study shows that 3 in 1,000 people die shor	nao	355	nao	238	nao	238	nao	203	nao	203	nao
HEALTH_58	ID: HEALTH_58 Title: Preventive swine flu treatment better concentrated on the young Description: Researchers have confirmed that treating everyone who gets swine flu with antiviral drugs should reduce people's symptoms and slow the spread of the virus. B	sim	356	nao	49	sim	92	sim	204	nao	204	nao
HEALTH_59	ID: HEALTH_59 Title: Smear tests prevent cervical cancer, but under 25s unlikely to benefit Description: New research into cervical screening has found that it doesn't seem to cut the risk of cancer for women under 25 years of age. However, it does have a	sim	357	nao	51	sim	93	sim	38	sim	51	sim
HEALTH_6	ID: HEALTH_6 Title: Drug Industry to Run Ads Favoring White House Plan Description: Lobbyists have been authorized to spend as much as $150 million on television commercials supporting President Obama's health care overhaul.	nao	19	sim	222	nao	222	nao	178	nao	178	nao
HEALTH_60	ID: HEALTH_60 Title: Men can cope with a wait-and-see approach to prostate cancer Description: Men who postpone treatment for prostate cancer in favour of regular check-ups don't suffer from depression or anxiety, a new study has found. Treatment for pros	sim	358	nao	55	sim	94	sim	205	nao	205	nao
HEALTH_64	ID: HEALTH_64 Title: Healthy living makes a real difference Description: Everyone knows the basics of a healthy lifestyle. Even so, achieving it is often difficult. Day to day, it can be a challenge persuading yourself to go without that cigarette, glass	sim	362	nao	241	nao	241	nao	209	nao	209	nao
HEALTH_65	ID: HEALTH_65 Title: Just how dangerous is swine flu? Description: There seems to have been a rhythm to the news reports about swine flu. "Biggest pandemic ever" and "Swine flu could kill 65,000 in UK" say headlines one day. "We're coping just fine" and "	sim	363	nao	58	sim	96	sim	41	sim	52	sim
HEALTH_66	ID: HEALTH_66 Title: Would you want to know your risk of Alzheimer's? Description: No test can predict with total accuracy who will get Alzheimer's disease, but genetic testing can tell you if your risk is higher than average. Unfortunately, there's not m	sim	364	nao	59	sim	97	sim	210	nao	210	nao
HEALTH_67	ID: HEALTH_67 Title: Another good reason to exercise Description: People who exercise regularly make a better recovery and are less disabled if they suffer a stroke, new research suggests. Although the findings are only preliminary, it seems like another	nao	365	nao	242	nao	242	nao	211	nao	211	nao
HEALTH_7	ID: HEALTH_7 Title: Patient Money: A Guide Through a Medical Wilderness Description: Susan Redstone with Jesse, a black thoroughbred mare, at a ranch in Old Snowmass, near Aspen, Colo.	sim	366	nao	21	sim	61	sim	14	sim	27	sim
HEALTH_8	ID: HEALTH_8 Title: Beyond Beltway, Health Debate Turns Hostile Description: Val Butsicaris of Taylor, Mich., center, confronts Representative John D. Dingell over health care.	sim	21	sim	223	nao	223	nao	179	nao	179	nao
HEALTH_9	ID: HEALTH_9 Title: Swine Flu Should Not Close Most Schools, Federal Officials Say Description: Most schools should be able to stay open even if swine flu outbreaks occur again this fall, government health officials said.	sim	367	nao	224	nao	224	nao	180	nao	180	nao
INTERNET_0	ID: INTERNET_0 Title: Slipstream: The Music Streams That Soothe an Industry Description: Scenes from the Black Eyed Peas' "I Gotta Feeling," available on YouTube and accessible from many smartphones.	sim	368	nao	243	nao	243	nao	212	nao	212	nao
INTERNET_1	ID: INTERNET_1 Title: Breakfast Can Wait. The Day's First Stop Is Online. Description: Liz Steyer after	nao	369	nao	244	nao	244	nao	213	nao	213	nao

	breakfast with three of her four children, ages 5 to 16. Laptops and cellphones are banned during meals.											
INTERNET_10	ID: INTERNET_10 Title: Microsoft to Sell Web Ad Agency Description: The French company Publicis Groupe struck a deal Sunday to buy Razorfish, the interactive agency.	sim	370	nao	14	sim	9	sim	11	sim	7	sim
INTERNET_11	ID: INTERNET_11 Title: What Tweeters Have to Say About the Movies Description: A site aggregates tweets about new mainstream movies to provide a picture of what the online masses think.	nao	371	nao	251	nao	251	nao	220	nao	220	nao
INTERNET_12	ID: INTERNET_12 Title: The Media Equation: For Murdoch, It's Try, Try Again Description: The owner of News Corporation says he wants to go against the grain and "charge for all our news Web sites."	nao	372	nao	252	nao	252	nao	221	nao	221	nao
INTERNET_2	ID: INTERNET_2 Title: Big Investor to Sell Part Of Stake in Lenovo Parent Description: The biggest stakeholder in the parent company of Lenovo said it was seeking to sell 29 percent of the parent company's shares for $404 million.	nao	373	nao	245	nao	245	nao	214	nao	214	nao
INTERNET_6	ID: INTERNET_6 Title: As Rivals Branch Out, SAP Is Sticking to Software Description: Léo Apotheker, chief of SAP.	sim	377	nao	66	sim	8	sim	47	sim	6	sim
INTERNET_7	ID: INTERNET_7 Title: There's an App for That. But a Revenue Stream? Description: An app from KCRW, a public radio station.	nao	378	nao	248	nao	248	nao	217	nao	217	nao
INTERNET_8	ID: INTERNET_8 Title: With Cable, Laying a Basis for Growth in Africa Description: Contractors laying fiber optic cable in Kenya. About 10 new undersea connections are expected to serve Africa within a year.	nao	379	nao	249	nao	249	nao	218	nao	218	nao
INTERNET_9	ID: INTERNET_9 Title: Bank Will Allow Customers to Deposit Checks by iPhone Description: Customers of USAA can photograph both sides of the check, send the images through an app and then void the check.	nao	380	nao	250	nao	250	nao	219	nao	219	nao
POLITIC_0	ID: POLITIC_0 Title: Government officials dismiss fresh calls for torture inquiry Description: The government today came under fresh pressure over the abuse of detainees as officials dismissed detailed allegations and calls for a judicial inquiry.In the w	nao	381	nao	253	nao	253	nao	222	nao	222	nao
POLITIC_1	ID: POLITIC_1 Title: War criminals 'loopholes' warning Description: Proposals from the justice secretary, Jack Straw, to change the law to enable the prosecution of overseas war criminals and torturers living in Britain for crimes dating back to 1991 fail	nao	382	nao	254	nao	254	nao	223	nao	223	nao
POLITIC_10	ID: POLITIC_10 Title: Tories vow to save money by scrapping national NHS database Description: Tories pledge to decentralise IT provision in the NHS, allowing trusts to buy their own computer systems provided they are compatible with others in the health	nao	383	nao	262	nao	262	nao	232	nao	232	nao
POLITIC_11	ID: POLITIC_11 Title: MI6 chief denies complicity in torture Description: Head Sir John Scarlett says officers 'committed to human rights values' as row mounts over abuse of detainees abroadSir John Scarlett, the head of MI6, today denied his officers wer	sim	384	nao	263	nao	263	nao	233	nao	233	nao
POLITIC_12	ID: POLITIC_12 Title: Government seeks to recast relations with UK Muslims Description: Communities secretary John Denham urges policy shift to downplay emphasis on tackling on violent extremismThe communities secretary, John Denham, is to attempt a fresh	nao	385	nao	264	nao	264	nao	234	nao	234	nao
POLITIC_13	ID: POLITIC_13 Title: Encourage whistleblowing, say MPs Description: Whistleblowing within the civil service needs to be encouraged if the government wants to stem leaks, a cross-party group of MPs recommends.The report by the Commons public administratio	nao	386	nao	265	nao	265	nao	235	nao	235	nao
POLITIC_14	ID: POLITIC_14 Title: MoD criticised over idle vehicles Description: • Three Paras who died in ambush last week named• Death toll since July for British troops reaches 27 A British soldier was killed while on patrol in southern Afghanistan on 8 August, b	nao	387	nao	266	nao	266	nao	236	nao	236	nao
POLITIC_15	ID: POLITIC_15 Title: 500 children a year abducted from UK Description: New data reveals a stark rise in child kidnapping by estranged parentsAlmost 500 children were abducted from the UK and taken abroad illegally last year, according to figures released	sim	6	sim	15	sim	32	sim	237	nao	237	nao
POLITIC_18	ID: POLITIC_18 Title: Life and death? Afraid so. It will be Labour's most seismic year since 1981 Description: Election defeat will bring the party's most pivotal moment for a generation. Many may jump ship and join forces with new alliesDavid Miliband ei	nao	390	nao	269	nao	269	nao	240	nao	240	nao
POLITIC_19	ID: POLITIC_19 Title: All talk and no trousers Description: Indignant government rhetoric on torture rings hollow. The evidence tells a very different storyIt was grandstand stuff. "This is not just about legal obligations," wrote David Miliband and Alan	nao	391	nao	270	nao	270	nao	241	nao	241	nao
POLITIC_2	ID: POLITIC_2 Title: Hazel Blears' car attacked Description: Former communities secretary says she does not think the attack was a protest related to her controversial expenses claimsHazel Blears, the former communities secretary at the centre of an expen	nao	2	sim	12	sim	99	sim	224	nao	224	nao
POLITIC_3	ID: POLITIC_3 Title: Mandelson says he's not running country Description: • Promise of no headlines and few announcements• Duty minister began stint while on holiday in CorfuPeter Mandelson insisted today that he was "not in charge", as he arrived back in	nao	392	nao	255	nao	255	nao	225	nao	225	nao
POLITIC_4	ID: POLITIC_4 Title: News of World settles Cherie libel case Description: News of the World article falsely accused Cherie Blair of having an 'inhuman' attitude in a discussion about victims of crimeCherie Blair has settled a libel action against the News	nao	393	nao	256	nao	256	nao	226	nao	226	nao
POLITIC_5	ID: POLITIC_5 Title: 1,381 phone and email taps a day Description: Police, intelligence service and council surveillance requests rise by 60% in two years, report showsPolice and other officials tapped phone calls and emails an average of 1,381 times a da	nao	394	nao	257	nao	257	nao	227	nao	227	nao
POLITIC_6	ID: POLITIC_6 Title: Gordon Brown to start Lake District holiday Description: Since becoming prime	nao	395	nao	258	nao	258	nao	228	nao	228	nao

	minister, Brown has eschewed Cape Cod for English tourist destinationsGordon Brown and his family are this week starting a holiday in the Lake District.A Do											
POLITIC_7	ID: POLITIC_7 Title: 'I'm a kindly pussycat' Description: Peter Mandelson tells Decca Aitkenhead why he used to be the hard man of New Labour … but now he's just a pussycatThe most accurate article Lord Mandelson ever read about himself was written in the	nao	396	nao	259	nao	259	nao	229	nao	229	nao
POLITIC_8	ID: POLITIC_8 Title: Senior Tories rush to deny plan for post-election tax rises Description: Senior Conservatives today sought to quash suggestions that the party's high command was actively considering raising VAT to bring in revenue "within weeks" of a	nao	4	sim	260	nao	260	nao	230	nao	230	nao
POLITIC_9	ID: POLITIC_9 Title: New evidence casts light on Tory EU ally's attitude to Jews Description: Observer finds newspaper interview that Polish leader of David Cameron's European parliament group denied makingThe row about the Conservatives' new best friend	nao	397	nao	261	nao	261	nao	231	nao	231	nao
SPORT_0	ID: SPORT_0 Title: Man Utd 1-4 Liverpool Description: What a week for Rafa Benítez. A four-goal victory over Real Madrid followed by four goals to beat Manchester United at Old Trafford. Life can scarcely get any sweeter, and it could easily have been fiv	nao	95	sim	94	sim	22	sim	343	nao	343	nao
SPORT_1	ID: SPORT_1 Title: Mourinho silent on assault allegation Description: • 'I am not talking about it. Mourinho sells a lot, you know that'• Inter boss says his team were unlucky to lose at Old TraffordJose Mourinho has refused to comment on allegations he h	sim	96	sim	98	sim	104	sim	344	nao	344	nao
SPORT_12	ID: SPORT_12 Title: Ryder leads Kiwis to consolation win Description: New Zealand 151-2 (23.2 overs) beat India 149 (36.3 overs)The all-round talents of Jesse Ryder helped New Zealand to a consolation win by eight wickets in the final match of their one-	nao	400	nao	369	nao	369	nao	355	nao	355	nao
SPORT_13	ID: SPORT_13 Title: Khan return could signal end for Barrera Description: • Amir aims to put Prescott defeat behind him• Fight may be swansong for 35-year-old BarreraAs in chess, the endgame in boxing starts from an indeterminate point and when it is play	nao	101	sim	108	sim	37	sim	56	sim	17	sim
SPORT_14	ID: SPORT_14 Title: Doubles victory signals Murray's return Description: • Scot partners Hutchins to first-round win at Indian Wells• British No1 will now test recovery from virus in singles matchAndy Murray asked his friend Ross Hutchins to ease him back	nao	401	nao	370	nao	370	nao	356	nao	356	nao
SPORT_15	ID: SPORT_15 Title: Mickelson's magic as Woods flounders Description: • American world No3 makes good his boasts with 66 • Tiger Woods languishes in a tie for 35th place at DoralAs the cynics smirked and the golf historians reached for the record books, P	nao	402	nao	371	nao	371	nao	357	nao	357	nao
SPORT_16	ID: SPORT_16 Title: Chambers back to the Jungle to plug book Description: • Sprinter back at the Jungle to plug book• Evans the pick of Tigers' new recruitsDwain Chambers will receive a warm welcome at the Jungle when he returns to Castleford for Super L	nao	403	nao	372	nao	372	nao	358	nao	358	nao
SPORT_17	ID: SPORT_17 Title: Jones quits as Saracens' director of rugby Description: • Australian had been expected to leave in the summer• Richard Graham will take over until end of the seasonEddie Jones has stepped down as Saracens director of rugby with immedia	nao	404	nao	373	nao	373	nao	359	nao	359	nao
SPORT_18	ID: SPORT_18 Title: Phelps admits 'stupid mistake' over photo Description: • American admits to considering quitting the sport• 'I'm ready to go another four years, and I still have goals'The Olympic champion Michael Phelps has allayed fears that he might	sim	102	sim	106	sim	23	sim	360	nao	360	nao
SPORT_19	ID: SPORT_19 Title: Carlsen shows strength and weakness Description: Since Magnus Carlsen became, for a day only, the youngest ever world No1 on the live ratings, the Norwegian 18-year-old's career has, judged by the Olympian standards of Bobby Fischer an	nao	405	nao	374	nao	374	nao	361	nao	361	nao
SPORT_2	ID: SPORT_2 Title: Ecclestone: No Donnington, no British GP Description: • Donington Park work must be complete by summer 2010• Additional F1 races in South Korea and India scheduledThe failure of the owners of Donington Park to complete extensive revisio	sim	406	nao	363	nao	363	nao	345	nao	345	nao
SPORT_20	ID: SPORT_20 Title: Man Utd 1-4 Liverpool Description: What a week for Rafa Benítez. A four-goal victory over Real Madrid followed by four goals to beat Manchester United at Old Trafford. Life can scarcely get any sweeter, and it could easily have been fi	sim	87	sim	95	sim	24	sim	362	nao	362	nao
SPORT_21	ID: SPORT_21 Title: Mourinho silent on assault allegation Description: • 'I am not talking about it. Mourinho sells a lot, you know that'• Inter boss says his team were unlucky to lose at Old TraffordJose Mourinho has refused to comment on allegations he	sim	88	sim	99	sim	107	sim	363	nao	363	nao
SPORT_22	ID: SPORT_22 Title: Ecclestone: No Donnington, no British GP Description: • Donington Park work must be complete by summer 2010• Additional F1 races in South Korea and India scheduledThe failure of the owners of Donington Park to complete extensive revisi	sim	407	nao	375	nao	375	nao	364	nao	364	nao
SPORT_23	ID: SPORT_23 Title: Daniel Taylor: Ferguson's quest is almost complete Description: The United manager has never forgotten his early years of suffering at the hands of AnfieldThe peculiar thing is that whenever Sir Alex Ferguson is asked about Manchester	sim	89	sim	104	sim	38	sim	365	nao	365	nao
SPORT_24	ID: SPORT_24 Title: David Lacey: Europe jumps on English treadmill Description: The continent is stifling a yawn as the gap between the top of the Premier League and the rest of Europe growsSame again then. The Champions League is becoming a TV repeat bes	sim	90	sim	101	sim	39	sim	366	nao	366	nao
SPORT_25	ID: SPORT_25 Title: Shaun Edwards: Mind will matter in 'Le Crunch' Description: England's winning mentality should see them beat France at Twickenham on SundayThe big men are on the move. On both sides of the Channel this week the coaches/managers of Fran	nao	408	nao	376	nao	376	nao	367	nao	367	nao

SPORT_26	ID: SPORT_26 Title: Joy of Six: Seve Ballesteros's greatest shots Description: From driving the green at The Belfry to an amazing piece of wizardry at the 1983 Ryder Cup, here are half a dozen moments of Ballesteros brillianceFor many European golf fans,	sim	409	nao	377	nao	377	nao	368	nao	368	nao
SPORT_27	ID: SPORT_27 Title: Hiddink questions City's financial muscle Description: • Money can't buy success says Chelsea manager• Dutchman hopes Terry will stay at club 'for ever'The Chelsea manager, Guus Hiddink, has admitted he is uncomfortable with the spendi	nao	91	sim	105	sim	108	sim	369	nao	369	nao
SPORT_28	ID: SPORT_28 Title: Agbonlahor needs to get back to basics Description: • 'Real spark will return to his game as and when he scores' • England striker has scored just once in past 14 appearancesIt seems strange to be asking where it has gone wrong for Gab	sim	92	sim	97	sim	109	sim	370	nao	370	nao
SPORT_29	ID: SPORT_29 Title: West Ham enjoy more good news Description: • Hammers settle with Sheffield United over Tevez affair• Court grants owners three months to sell the clubTwo of the major questions clouding West Ham's future were resolved yesterday when it	nao	410	nao	378	nao	378	nao	371	nao	371	nao
SPORT_30	ID: SPORT_30 Title: Giovani swaps Lane for Ipswich Description: • Mexican forward failed to make an impression at Spurs• Former Barca star set to play in the ChampionshipOne minute you're playing for Barcelona, the next you find yourself turning out for I	nao	411	nao	379	nao	379	nao	372	nao	372	nao
SPORT_31	ID: SPORT_31 Title: Kauto takes place in chasing hall of fame Description: • Nicholls breaks trainers' record with five winners • Walsh smashes jockeys' record with seven winnersMaking history is supposed to be difficult. Yesterday, it looked effortless,	sim	412	nao	380	nao	380	nao	373	nao	373	nao
SPORT_32	ID: SPORT_32 Title: Ryder leads Kiwis to consolation win Description: New Zealand 151-2 (23.2 overs) beat India 149 (36.3 overs)The all-round talents of Jesse Ryder helped New Zealand to a consolation win by eight wickets in the final match of their one-	nao	413	nao	381	nao	381	nao	374	nao	374	nao
SPORT_35	ID: SPORT_35 Title: Mickelson's magic as Woods flounders Description: • American world No3 makes good his boasts with 66 • Tiger Woods languishes in a tie for 35th place at DoralAs the cynics smirked and the golf historians reached for the record books, P	nao	415	nao	383	nao	383	nao	376	nao	376	nao
SPORT_36	ID: SPORT_36 Title: Chambers back to the Jungle to plug book Description: • Sprinter back at the Jungle to plug book• Evans the pick of Tigers' new recruitsDwain Chambers will receive a warm welcome at the Jungle when he returns to Castleford for Super L	sim	416	nao	384	nao	384	nao	377	nao	377	nao
SPORT_37	ID: SPORT_37 Title: Jones quits as Saracens' director of rugby Description: • Australian had been expected to leave in the summer• Richard Graham will take over until end of the seasonEddie Jones has stepped down as Saracens director of rugby with immedia	nao	417	nao	385	nao	385	nao	378	nao	378	nao
SPORT_38	ID: SPORT_38 Title: Phelps admits 'stupid mistake' over photo Description: • American admits to considering quitting the sport• 'I'm ready to go another four years, and I still have goals'The Olympic champion Michael Phelps has allayed fears that he might	sim	94	sim	107	sim	25	sim	379	nao	379	nao
SPORT_42	ID: SPORT_42 Title: Pendleton to review workload Description: • 'I'm not sure I'd do it again,' says 28-year-old• Lizzie Armitstead wins bronze in points raceVictoria Pendleton will review her programme at future events after describing this week's Track	nao	116	sim	74	sim	110	sim	383	nao	383	nao
SPORT_43	ID: SPORT_43 Title: West Indies v England - live! Description: Press refresh for the latest updates. And in the meantime why not email Wee Rab at rob.smyth@guardian.co.uk with your mudane/sublime insights42nd over: West Indies 185-7 (Bravo 33, Miller 5) F	sim	117	sim	389	nao	389	nao	384	nao	384	nao
SPORT_44	ID: SPORT_44 Title: Oxford battle back to clinch victory Description: • Dark Blues clinch 75th victory despite slow start• Crews clash oars at Chiswick Reach as fightback beginsOxford have beaten Cambridge to claim the 2009 university Boat Race.As the hea	sim	421	nao	390	nao	390	nao	385	nao	385	nao
SPORT_45	ID: SPORT_45 Title: Said & Done: The award-winning football column Description: Man of the weekNewcastle's biggest problem: dignity. "Mike Ashley has gone in there and lowered the standards," says Dave Whelan. "He has no class whatsoever. All of the digni	sim	422	nao	79	sim	111	sim	386	nao	386	nao
SPORT_46	ID: SPORT_46 Title: Big interview: Bill Elliott meets Rory McIlroy Description: The next big thing according to many, but the 19-year-old from Belfast still takes his laundry back to his mum'sThere is always a defining moment when a gifted child alerts st	nao	423	nao	391	nao	391	nao	387	nao	387	nao
SPORT_47	ID: SPORT_47 Title: Paul Hayward: Liverpool can overcome history Description: Rafael Benítez's men will have to deal with references to their last title win as they try to overhaul Manchester UnitedCome with us now to a world called April 1990, where Nels	sim	119	sim	78	sim	26	sim	388	nao	388	nao
SPORT_48	ID: SPORT_48 Title: Kevin Mitchell: British amateurs show real promise Description: Dudley O'Shaughnessy leads a crop of youngsters who merit having their amateur career secured under a proper administrationPaul Gallico, a fine writer and a human being of	nao	120	sim	87	sim	112	sim	54	sim	55	sim
SPORT_49	ID: SPORT_49 Title: Eddie Butler: New breed of Lions emerge Description: The legacy of a largely unsatisfactory Six Nations tournament will weigh heavily on Ian McGeechan's Lions selectionSix Nations has come and gone, gloriously Irish, universally unsati	nao	424	nao	392	nao	392	nao	389	nao	389	nao
SPORT_5	ID: SPORT_5 Title: Shaun Edwards: Mind will matter in 'Le Crunch' Description: England's winning mentality should see them beat France at Twickenham on SundayThe big men are on the move. On both sides of the Channel this week the coaches/managers of Franc	nao	425	nao	364	nao	364	nao	348	nao	348	nao
SPORT_50	ID: SPORT_50 Title: Mike Selvey: Flower frustrated and deflated Description: England's inept displays against the West Indies cannot be blamed on the stand-in coachFriday was a bad day for Andrew Strauss	nao	426	nao	393	nao	393	nao	390	nao	390	nao

	and his team, but it was a worse one for Andy Flowe											
SPORT_51	ID: SPORT_51 Title: Evans confident ahead of Slovenia match Description: • United centre-back hoping for more success against Slovenia• 'We are happy and think we deserved it,' says FeeneyJonny Evans is confident Northern Ireland are on track to qualify f	nao	427	nao	394	nao	394	nao	391	nao	391	nao
SPORT_52	ID: SPORT_52 Title: Uefa slam 'ridiculous' squad sizes Description: • Liverpool amass 62-man squad under Benítez• Limit of 25 players for the Champions LeagueUefa havebranded as "ridiculous" the number of players on the payroll of the Premier League's Big	sim	121	sim	80	sim	41	sim	392	nao	392	nao
SPORT_53	ID: SPORT_53 Title: 'Sticky' pitch cost Scotland, says Miller Description: • Striker curses 'sticky surface' against Holland• 'Home games against Iceland and Macedonia are vital'Kenny Miller claimed the Amsterdam ArenA pitch cost him the chance to strike	sim	428	nao	395	nao	395	nao	393	nao	393	nao
SPORT_54	ID: SPORT_54 Title: Mourinho queries Beckham's US motives Description: • Jose Mourinho implies money was motivation for move• Former Chelsea manager lines up match against old clubInternazionale coach Jose Mourinho has criticised the set-up of football in	sim	122	sim	396	nao	396	nao	394	nao	394	nao
SPORT_55	ID: SPORT_55 Title: Peterborough Utd 2-0 Leicester City Description: How Ferguson senior would like to have replicated his son's run of results this month. Darren Ferguson's Peterborough United played six games in March and this hugely convincing victory	nao	429	nao	397	nao	397	nao	395	nao	395	nao
SPORT_56	ID: SPORT_56 Title: Benítez rules out move for Silva Description: • Benítez says it is too early to talk of transfer targets• Lee and Pellegrino extend their Anfield contractsThe Liverpool manager, Rafael Benítez, today ruled out a move for the Valencia m	sim	430	nao	89	sim	113	sim	55	sim	56	sim
SPORT_57	ID: SPORT_57 Title: Ferguson's son hints at retirement date Description: Manchester United will be looking for a new manager at the end of next season, according to Sir Alex Ferguson's son Darren, who believes that the longest-serving and oldest manager	nao	124	sim	90	sim	29	sim	396	nao	396	nao
SPORT_58	ID: SPORT_58 Title: The Dwain event Description: A clean Dwain Chambers finished fourth at the 2000 Olympics, but that wasn't enough. Now reviled as a drugs cheat, he has lost his house, his income and his credibility. OSM spends four manic days in Turin	sim	125	sim	85	sim	114	sim	397	nao	397	nao
SPORT_59	ID: SPORT_59 Title: Michael Sheen: My sporting life Description: Pitch-perfect as Brian Clough, Michael Sheen may not be the best British actor in the business - but he's in the top oneMichael Sheen was five years old and living in South Wales when Brian	nao	126	sim	398	nao	398	nao	398	nao	398	nao
SPORT_6	ID: SPORT_6 Title: Joy of Six: Seve Ballesteros's greatest shots Description: From driving the green at The Belfry to an amazing piece of wizardry at the 1983 Ryder Cup, here are half a dozen moments of Ballesteros brillianceFor many European golf fans, S	nao	431	nao	365	nao	365	nao	349	nao	349	nao
SPORT_60	ID: SPORT_60 Title: Bristol 37-18 Worcester Description: Bristol 37-18 WorcesterBristol avoided dropping out of the Guinness Premiership lifeline for at least another week as they recorded only their second success of the campaign with a bonus-point victo	sim	432	nao	399	nao	399	nao	399	nao	399	nao
SPORT_61	ID: SPORT_61 Title: Murray battles through in Miami Description: • 21-year-old still in the running to break into world's top three• Scot will not have to face tricky tie with David NalbandianIf this was a hangover from that Rafael Nadal beating in Indian	sim	127	sim	86	sim	115	sim	53	sim	57	sim
SPORT_62	ID: SPORT_62 Title: Africans deliver brutal lesson to Twell Description: • Kenya's Florence Kiplagat wins in Amman • Twell finishes poor 38th with British team ninthStephanie Twell is so precise in her preparation that she knows exactly which distances sh	nao	433	nao	400	nao	400	nao	400	nao	400	nao
SPORT_65	ID: SPORT_65 Title: Rambling good for Aintree gambling Description: Form book points to 11-year-old's chances in Saturday's Grand NationalWhen your Grand National fancy is pulled out lame just as you are putting pen to paper – metaphorically speaking – yo	nao	436	nao	403	nao	403	nao	403	nao	403	nao
SPORT_66	ID: SPORT_66 Title: South Africa attempt to derail England's 2015 World Cup bid Description: • RFU remain confident they will be hosts• Springboks believe process was biasedSouth Africa will today attempt to torpedo England's chances of hosting the 2015 r	sim	437	nao	404	nao	404	nao	404	nao	404	nao
SPORT_67	ID: SPORT_67 Title: Eye injury could end Massa's F1 career Description: • Brazilian taken out of coma after accident in Hungary• Ferrari hold back from decision on driver line-upFelipe Massa was awake and talking to his family last night but his career as	nao	438	nao	405	nao	405	nao	405	nao	405	nao
SPORT_7	ID: SPORT_7 Title: Hiddink questions City's financial muscle Description: • Money can't buy success says Chelsea manager• Dutchman hopes Terry will stay at club 'for ever'The Chelsea manager, Guus Hiddink, has admitted he is uncomfortable with the spendin	nao	99	sim	103	sim	105	sim	350	nao	350	nao
SPORT_8	ID: SPORT_8 Title: Agbonlahor needs to get back to basics Description: • 'Real spark will return to his game as and when he scores' • England striker has scored just once in past 14 appearancesIt seems strange to be asking where it has gone wrong for Gabr	sim	100	sim	96	sim	106	sim	351	nao	351	nao
TECH_0	ID: TECH_0 Title: Fourth Person Charged in D.C. Tech Office Scandal Description: A former employee of a District-based contracting firm has become the fourth person charged in a bribery and kickback scandal that exposed lax oversight of the D.C. governm	nao	440	nao	406	nao	406	nao	406	nao	406	nao
TECH_1	ID: TECH_1 Title: Science and Tech Firms Find Like Minds in Ballston Description: California has Silicon Valley. Dulles has the technology corridor. And Ballston has its own "science corridor," a quaint little neighborhood overflowing with science and t	nao	441	nao	407	nao	407	nao	407	nao	407	nao

TECH_2	ID: TECH_2 Title: Prince William County Announces Measures After Information Technology Scandal Description: New internal control measures and a background-check policy have commenced in Prince William County's information technology department to preve	nao	442	nao	408	nao	408	nao	408	nao	408	nao
TECH_3	ID: TECH_3 Title: Bubble or Bounce, Tech Stocks Are Soaring Description: If U.S. stocks have been hot recently, technology shares have been blistering. And news out of the technology sector last month only reassured some investors that the worst of the	nao	443	nao	409	nao	409	nao	409	nao	409	nao
TECH_4	ID: TECH_4 Title: Montgomery, Md., Planning Director's Technology Spending Under Reviewed Description: Auditors looking at credit card spending at the Montgomery County planning agency are examining several technology purchases that appear to have been	nao	444	nao	410	nao	410	nao	410	nao	410	nao