SERVIÇOS DE DESENVOLVIMENTO DE SOFTWARE: UMA ABORDAGEM PARA REUTILIZAÇÃO DE MODELOS CONCEITUAIS

Fabio Perez Marzullo

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.

Orientador(es): Jano Moreira de Souza.

Rio de Janeiro

Março de 2014

SERVIÇOS DE DESENVOLVIMENTO DE SOFTWARE: UMA ABORDAGEM PARA REUTILIZAÇÃO DE MODELOS CONCEITUAIS

Fabio Perez Marzullo

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. Jano Moreira de Souza, Ph.D.

_____

Prof. Geraldo Bonorino Xexéo, D. Sc.

_____

Prof. Paulo de Figueiredo Pires, D.Sc.

_____

Prof. Astério Kiyoshi Tanaka, Ph.D.

_____

Prof. Leonardo Guerreiro Azevedo, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

MARÇO DE 2014

Marzullo, Fabio Perez

Serviços de Desenvolvimento de Software: Uma Abordagem para Reutilização de Modelos Conceituais/ Fabio Perez Marzullo. – Rio de Janeiro: UFRJ/COPPE, 2014.

XV, 124 p. 29,7 cm.

Orientador: Jano Moreira de Souza.

Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2014.

Referências Bibliográficas: p. 104-114.

1. Reuso de Modelos. 2. Model-Driven Development. 3. Service-Oriented Architecture. I. Souza, Jano Moreira de. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Este trabalho é dedicado aos meus filhos:

– Miguel Dias Marzullo e Lucas Dias Marzullo

Agradecimentos

Gostaria de agradecer, acima de tudo, aos meus pais Maria Lúcia Perez Marzullo e Antônio João Marzullo Filho, por sempre estarem ao meu lado, amparando, apoiando, ajudando e amando. Sem tamanho apoio certamente não teria conseguido finalizar este trabalho. Tudo que sou hoje devo a vocês, incondicionalmente!

Agradeço a minha esposa Renata Braga Dias por aceitar e entender as inúmeras horas, dias, meses e anos de estudos. Sei que não foram fáceis e por isso acredito que seu amor foi fundamental para que eu pudesse enfrentar e vencer todos os obstáculos.

Agradeço aos meus irmãos Thatiana Perez Marzullo e Marcio Perez Marzullo, por terem paciência e sabedoria nos momentos mais difíceis e turbulentos, e da mesma forma, por me apoiarem e amarem.

Ao professor Jano Moreira de Souza pela sua orientação e por confiar em meu trabalho nos projetos desenvolvidos pelo laboratório de Banco de Dados. Aos professores Geraldo Bonorino Xexéo, Paulo Pires, Astério Tanaka e Leonardo Azevedo, por aceitarem fazer parte da banca de avaliação.

Agradecimentos especiais a todos aqueles que fizeram parte deste trabalho direta ou indiretamente e à Universidade Federal do Rio de Janeiro pelos quase 20 anos de estudos e pesquisas.

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

SERVIÇOS DE DESENVOLVIMENTO DE SOFTWARE: UMA ABORDAGEM PARA REUTILIZAÇÃO DE MODELOS CONCEITUAIS

Fabio Perez Marzullo

Março/2014

Orientador: Jano Moreira de Souza.

Programa: Engenharia de Sistemas e Computação

O objetivo desta tese é apresentar como a interseção de tecnologias como desenvolvimento orientado a modelos, arquiteturas orientadas a serviço e reuso de software podem, por meio de uma abordagem apoiada por modelos de processo, orientar a proposição de um ambiente de reutilização e recuperação de modelos conceituais, promovendo práticas de reuso, como os aplicados em desenvolvimento baseado em componentes, e em linhas de produtos de software.

O trabalho, portanto, propõe a criação de um ambiente de reuso dentro do paradigma de “Software as a Service”, utilizando como entrada modelos conceituais e produzindo como resultado serviços de TI representados por pacotes de código fonte.

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

SOFTWARE DEVELOPMENT SERVICES: AN APPROACH TO CONCEPTUAL MODEL REUSE

Fabio Perez Marzullo

March/2014

Advisor: Jano Moreira de Souza.

Department: Computer and Software Engineering

The purpose of this work is to present how the intersection of technologies such as model-driven development, service-oriented architecture and software reuse can, through a process model supported approach, guide the proposal of a conceptual model reuse environment, promoting reuse practices close to those applied to component-based development and software product lines.

This work, therefore, proposes the creation of a reuse environment in a Software as a Service paradigm, using as input conceptual models and producing IT services as presented by source code packages.

Índice

Índice de Tabelas

Índice de Figuras

1. Introdução

Há décadas, é possível observar como a Engenharia de Software (ES) oferece ferramentas que podem ser vistas como elementos fundamentais para o sucesso na condução de projetos de software [1]. O uso sistematizado de paradigmas como os de arquiteturas orientadas a serviço (SOA – Service-Oriented Architecture) [2] e o de desenvolvimento orientado a modelos (MDD – Model-Driven Architecture) [3] oferecem oportunidades de inovação e auxiliam na construção de soluções eficientes e aderentes às necessidades do mundo real.

A todo momento, é possível identificar necessidades de utilização de abordagens mais eficientes e sistematizadas de desenvolvimento de software com o intuito de se buscar melhores mecanismos de controle, tendo como meta o aumento da produtividade [4]. Ao observar a complexidade inerente aos os domínios de negócio que são atendidos pelas inúmeras soluções de software criadas mundialmente, percebe-se que há espaço para a introdução constante de novas abordagens que contribuam para a tarefa contínua de buscar novas soluções para problemas comumente enfrentados [5].

Cenários como estes demonstram como a descoberta de novas abordagens levam a novas formas de ultrapassar desafios e ajudam a influenciar a mentalidade daqueles que estão envolvidos nas tarefas de captura dos conceitos e regras tratados [6]. Dessa forma, o uso sistematizado dos paradigmas de desenvolvimento, tais como os de SOA e MDD, e da aplicação de técnicas específicas de ES, tal como a de reuso de software, vêm em auxílio à concepção de novas abordagens e oferecem condições mais simples de desenvolvimento, auxiliando na identificação de soluções aos problemas provenientes dos domínios de negócio tratados [7].

Com o objetivo de contribuir com esse constante esforço de criação de soluções, que sejam capazes de ajudar na forma como as organizações conduzem seus projetos de software, este trabalho propõe uma abordagem sistematizada de reuso de modelos conceituais. Como consequência, busca-se propor um ambiente de desenvolvimento onde o reuso dos modelos conceituais, quando aliado a teorias como desenvolvimento orientado por modelos (MDD) e arquiteturas orientadas por serviços (SOA), ofereça ganhos significativos à produtividade dos projetos.

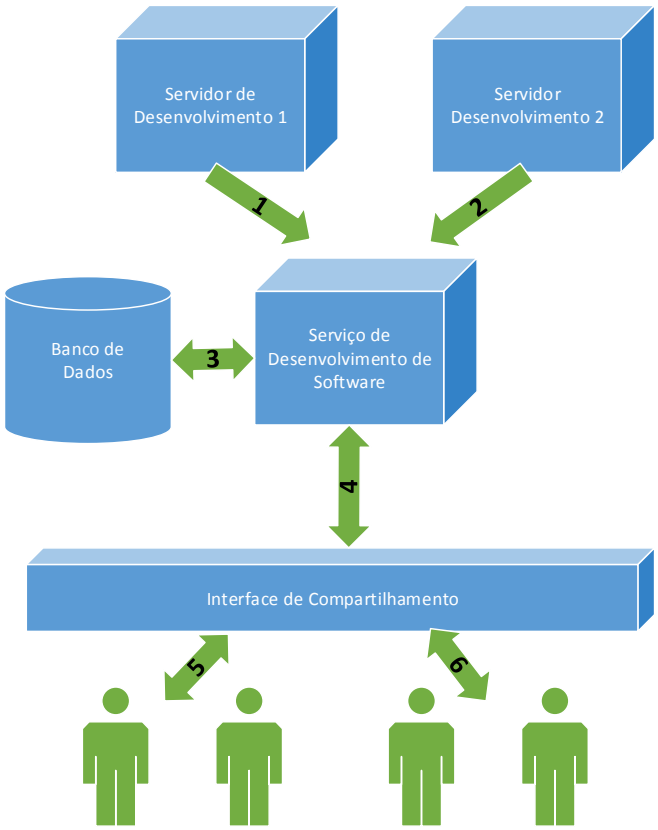

Figura 1: Arquitetura do serviço de desenvolvimento de software - SDS.

A Figura 1 apresenta uma visão global da proposta, onde o objetivo é o de criar um ambiente, um serviço de desenvolvimento de software (SDS), que seja capaz de armazenar modelos conceituais desenvolvidos em projetos de software distintos (como

destacado pelos passos 1 e 2), em uma base de dados remota (passo 3) e, então, seja compartilhado e reutilizado por usuários que possuam acesso ao SDS (passos 4, 5 e 6).

Para criar uma visão processual, a gerência de reuso do nível E do Modelo de Processo de Software Brasileiro (MPS-BR) [8] vem ao encontro dos objetivos deste trabalho. Ela traz à proposta um conjunto de atividades que devem ser atendidas para que o usuário consiga conduzir ações de reuso dentro do ambiente de desenvolvimento aqui proposto, tais como:

- Categorização e organização dos modelos conceituais.
- Reutilização dos modelos de forma sistemática.
- Estabelecimento de uma base central para armazenamento das informações e dos ativos que serão reutilizados.
- Automatização do processo por meio do uso de ferramenta que implemente a solução proposta.

Além disso, buscando maior sustentação conceitual, considerou-se os seguintes paradigmas e técnicas para formalização desta abordagem em diferentes níveis, a saber:

1. A introdução do paradigma de gerenciamento de linhas de produto de software (SPL – Software Product Line) [9] se dá pela busca de uma abordagem sistematizada de categorização e agrupamento dos modelos conceituais que são os ativos definidos como reutilizáveis. Por meio desta categorização, permite-se agrupar os modelos em conjuntos definidos por características de projetos e/ou domínios de negócio, facilitando sua catalogação e consequente compartilhamento e reuso.

2. A introdução do paradigma de desenvolvimento orientado por modelo é feita para garantir a composição arquitetural dos pacotes de código gerados a partir dos modelos conceituais inseridos no ambiente proposto. Com essa organização lógica,

objetiva-se a uniformidade dos pacotes de código produzidos, assim como os serviços que eles representam.

3. A introdução do paradigma de arquitetura orientada por serviços se dá pela necessidade de visualizar cada modelo inserido no ambiente proposto como um serviço a ser reutilizado em outros projetos e, ao mesmo tempo, embarcar no ambiente de reuso a capacidade de integração e execução destes serviços com a sua inserção em um barramento de serviços.

1.1. Diferencial

Objetivamente, o problema a ser abordado nesta tese é o de reuso de modelos conceituais, como abordagem de otimização de custo e tempo. Como produto, buscou-se a criação de um serviço de desenvolvimento de software (SDS) capaz de oferecer uma abordagem de desenvolvimento orientada ao reuso de modelos conceituais dentro de uma infraestrutura de provimento de serviço, com a meta de obter resultados relacionados à redução de custo e tempo de desenvolvimento.

Por meio de buscas efetuadas nas principais bibliotecas digitais de indexação de textos de pesquisa no mundo tais como IEEE, ACM, Portal Capes, Scopus, DBLP e ISI, além de uma busca não exaustiva no Google, é possível comprovar que o tema é relevante e abordado em diferentes publicações e trabalhos correlacionados. A comprovação de que este trabalho se traduz em um diferencial técnico é apresentado no Anexo I. Da avaliação feita, não foi identificado qualquer trabalho que apresentasse um ambiente de reuso de modelos conceituais, com alinhamento ao qual se pretende com o MPS-BR, e com o objetivo de otimizar o custo e tempo de desenvolvimento de projetos de software. A partir dos resultados obtidos nas buscas, é possível portanto, avaliar que

o tema abordado possui um diferencial que justifique a elaboração desta tese, sendo o fato significante a inexistência de uma abordagem como a que se propõe, alinhada ao processo de gerência de reuso do MPS-BR, assim como definido no nível E do seu guia geral [8].

1.2. Hipótese

Sabe-se que o reuso sistematizado de modelos conceituais, seja ele intra ou inter domínios, é uma tarefa árdua [9]. No entanto, apesar das dificuldades, busca-se validar a proposta de que é possível criar um ambiente de reuso de modelos conceituais dentro de uma abordagem sistematizada e apoiada pelas atividades definidas pela gerência de reuso do nível E do MPS-BR, com o objetivo de otimizar custo e tempo nos projetos de software.

Sendo a hipótese verdadeira, objetiva-se a sua validação a partir da varredura na literatura, assim como apresentada no capítulo 2, do estabelecimento teórico da abordagem, assim como apresentado no capítulo 3 e da prova de conceito descrita no capítulo 4. Nesta validação, são apresentadas as virtudes e limitações associadas ao reuso de modelos conceituais e como se aplica em modelos com o grau de generalização necessária para que haja sucesso. Pretende-se, portanto, demonstrar como a abordagem é eficiente utilizando modelos que sejam verdadeiramente genéricos e passíveis de utilização em diferentes domínios de negócio.

1.3. Organização do trabalho

O trabalho está organização em 5 capítulos. O Capítulo 1 corresponde à introdução. O Capítulo 2 apresenta o amparo teórico, oferecendo fundamentação das técnicas que inspiraram a construção da tese, além de apresentar trabalhos relacionados às abordagens de reuso de modelos, corroborando as principais decisões técnicas apresentadas. O Capítulo 3 apresenta a proposta conceitual para a criação do serviço de desenvolvimento de software (SDS), definindo sua arquitetura e as atividades que devem ser executadas para a condução do reuso dos modelos conceituais. O Capítulo 4 apresenta o protótipo criado para apoiar a proposta de reuso dos modelos conceituais e a prova de conceito com o objetivo de validar a hipótese da tese. Finalmente, o último capítulo apresenta as conclusões deste trabalho e os possíveis desdobramentos em trabalhos futuros.

1.4. Roadmap

Das ações realizadas para alcançar estes resultados destacam-se:

1. Revisão bibliográfica, apresentada no capítulo 2 contendo:
    a. Estudos relacionados à teoria de arquitetura orientada a serviços.
    b. Estudos relacionados à teoria de desenvolvimento orientado por modelo.
    c. Estudos relacionados ao reuso em software e a linhas de produtos de software.

2. Avaliação das principais iniciativas existentes e que tenham como objetivo o reuso de modelos conceituais em apoio ao a abordagens de redução de custo e prazo de desenvolvimento, como apresentado no capítulo 3.

3. Concepção do arcabouço técnico, apresentado no capítulo 3 contendo:

    a. Definição do serviço de desenvolvimento de software.

    b. Projeto da infraestrutura de reuso.

4. Prototipação do SDS apresentado no capítulo 4.

5. Validação da hipótese e avaliação dos resultados conforme prova de conceito efetuada em projetos de software apresentados no capítulo 5.

6. Conclusões.

2. Revisão bibliográfica

Este capítulo oferece uma revisão das teorias e técnicas utilizadas como base para consolidação da proposta do serviço de desenvolvimento de software.

2.1. O reuso em software

Leach [10] apresenta que técnicas de reuso em projetos de desenvolvimento de software têm sido propostas há anos e, em sua maioria, determinam a busca por padrões recorrentes e que possam ser reaproveitados em cenários semelhantes, sejam relacionados ao processo como aos artefatos que nos projetos são criados. Tanto conceitual como tecnicamente, o reuso determina a fatoração destes padrões de modo a tornar o que foi destacado em algo passível de ser reaproveitado em contexto de projeto semelhante.

É possível entender o reuso como a identificação de um padrão de codificação e/ou modelagem recorrente o qual é destacado e formatado em um artefato autocontido e então reutilizado em projetos distintos de software [1]. Alguns autores, como EZRAN *et al.* [11], associam o termo à composição de linhas de código em pacotes definidos tais como componentes de software. Esse procedimento, por sua vez, determina um comportamento de reuso tal que o custo associado à criação daquele bloco de código (ou componente) pode ser diluído conforme o número de vezes que este é embarcado em um novo projeto de software.

Por outro lado, KRUEGER [12] apresenta que o reuso pode ser visto como uma ação que se estende ao reaproveitamento de estruturas conceituais, tais como especificações, modelos, processos de desenvolvimento, documentação, entre tantos outros artefatos que sejam importantes para a criação de soluções de software.

Não distante às atividades normais de um desenvolvimento de software, deve-se observar que diferentes são as abordagens que as organizações de TI podem adotar para criar uma cultura organizacional de reuso. Em duas frentes, o reuso definido como em [11]: (1) vertical, que determina um reuso interno ao domínio de negócio no qual se está operando; e o (2) horizontal, que busca similaridades em diferentes domínios de negócio. Ambos, entretanto, tentam identificar similaridades técnicas e semânticas que tornem possível a fatoração e a consequente capacidade de reuso.

Seguindo o mesmo raciocínio, dentro das abordagens definidas para reuso em software, há ainda a possibilidade de se classificá-las como em [11] e [13]: (1) caixa branca, que determina ações de reuso por meio da intervenção direta à construção interna do artefato; (2) caixa preta, que determina um tipo de reuso onde a construção interna torna-se inviolável e se dá somente por meio do acesso às interfaces do artefato; (3) caixa cinza, onde o reuso ocorre mediante parametrização intensa; e por fim (4) caixa de vidro, onde há a necessidade de entender a construção interna do artefato, porém sem ter que interferir diretamente em sua lógica interna.

Levando em consideração essas abordagens, deve-se observar o tipo de estratégia que a organização deve seguir para que as ações de reuso produzam o resultado esperado. As estratégias citadas indicam a melhor relação de custo e benefício associada ao custo do artefato, aplicabilidade e sua origem, pois os artefatos candidatos à reutilização podem ser construídos internamente ou fornecido por partes terceiras [14].

Em ambos os casos, para se garantir um reuso consistente, é necessário entender como sua lógica interna, seu escopo, seu contexto de aplicação, sua aplicabilidade e os ganhos associados a sua utilização, quando vistos em conjunto, conseguem agregar positivamente ao produto final de software. Nesse momento, destacam-se as

perspectivas *substância* e *técnica* [15]. A primeira enfoca o reuso de ideias, incluindo conceitos formais e soluções generalizadas de problemas recorrentes, além de procedimentos e processos. A segunda, por sua vez, enfoca o reuso em nível de especificação, que engloba as fases de análise e projeto, e consequentemente os modelos conceituais que nelas são desenvolvidos.

Da teoria de reuso às linhas de produtos de software

A trajetória da evolução da teoria de reuso até a convergência de ideias que compõem a teoria de Linhas de Produtos de Software (SPL – Software Product Line) demonstra o amadurecimento a que se teve que alcançar para alavancá-la como uma das mais promissoras abordagens de reuso existentes atualmente. Presente desde o final dos anos 90, a abordagem de SPL tem por mérito olhar de forma mais abrangente o processo de desenvolvimento de software e determinar, já em tempo de análise conceitual, os aspectos que são relevantes para a ocorrência do reuso. Como apresentado em [29], na abordagem determinada pelo SPL, observa-se uma tendência onde o reuso é regido por um domínio de negócio e que todo artefato de software desenvolvido para este domínio tende a seguir padrões e funções específicas orientadas pelo domínio de negócio, logo a capacidade de reutilizar estes artefatos é alavancada e as chances de sucesso dentro do domínio, consequentemente, aumentam.

Desde os anos 60 até os dias de hoje o processo evolutivo da teoria de reuso determinou diversas iniciativas, cada qual com suas características e relevâncias, que, bem ou mal sucedidas, contribuíram para a construção de uma teoria viável tanto técnica como financeiramente [17]. Seja por meio da reutilização de trechos de código, passando pela capacidade de se criar e compartilhar componentes semanticamente completos, até a convergência em linhas e/ou famílias de produtos, entende-se que

independente de qual abordagem adotar, é possível incorporar tais ações de maneira a melhorar, ou minimizar custos associados ao desenvolvimento de produtos de software. Isso significa que ao se alinhar iniciativas de reuso à teoria de SPL, deve-se garantir que as ações impetradas estejam coerentes com a necessidade de categorização e organização dos ativos a serem reutilizados.

Iniciativas em SPL

Devido ao estímulo do *Information Technology for European Advancement* (ITE2) [20], que prevê o aumento da competitividade da indústria de software europeia, algumas iniciativas em SPL podem ser destacadas como das mais eficientes (Figura 1), tanto no que diz respeito a plataformas de software, como a capacidade de customização em massa.

Dentre os que fazem parte do portfólio de projetos apresentados na Figura 2, os que melhor se destacam como iniciativas de SPL, pelos resultados alcançados em termos de ações executadas visando o reuso e ativos reutilizáveis efetivamente criados ao longo dos projetos, são os de criação de software.

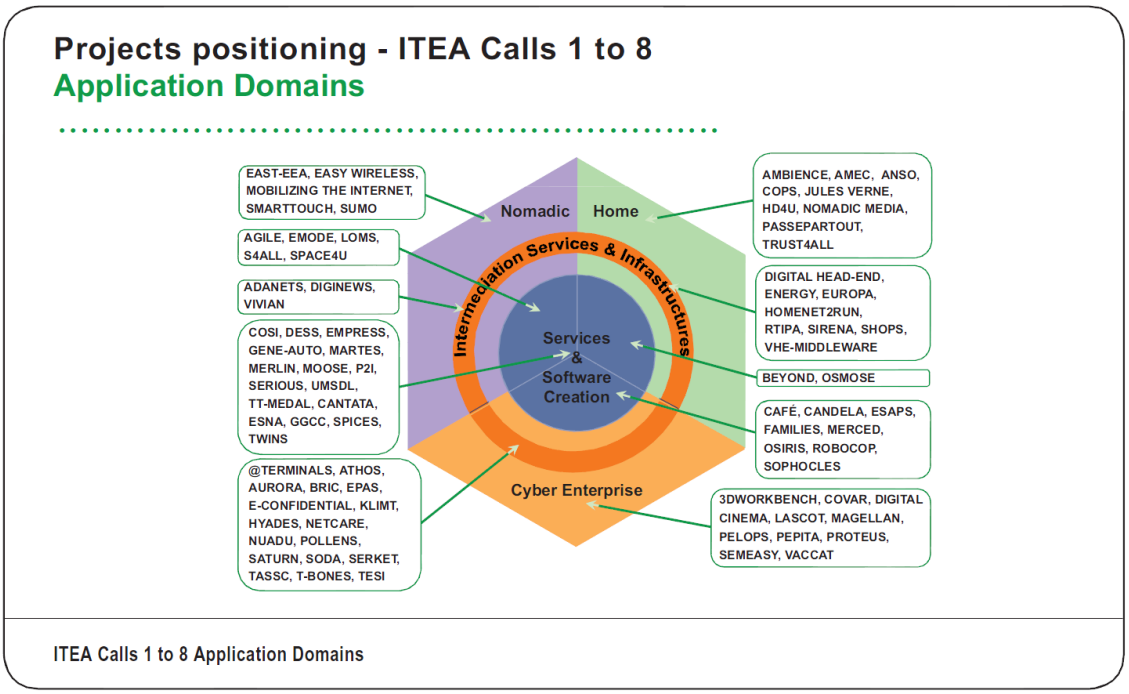

Figura 2: Posicionamento dos principais projetos do ITEA que atualmente utilizam SPL como iniciativa organizacional de reuso (extraído do site do ITEA).

REINEHR em [17] faz um resumo dos principais projetos relacionados à SPL, e demonstra que três se destacam pela qualidade de seus resultados em se tratando de ativos de software reutilizáveis:

- ESAPS – Engineering Software Architectures, Processes and Platforms, que teve como foco o desenvolvimento de famílias de sistemas embarcados para diversos tipos de negócio, desde dispositivos médicos, até eletrônicos automotivos.

- CAFÉ – Concepts to Application in System-Family Engineering, onde o principal objetivo foi o de aplicar os conceitos desenvolvidos pelo ESAPS, consolidando e analisando seus dados.

- FAMILIES - Fact-based Maturity through Institutionalization, Lessons-learned and Involved Exploration of System-family engineering, que em geral teve o objetivo de organizar o conhecimento obtido em projetos anteriores, com a

consequente criação de um catálogo global contendo os processos, métodos e técnicas utilizadas.

Resultados mais significativos destas iniciativas podem ser vistos em [21], onde há o relato da implantação completa dos conceitos derivados dos projetos, destacando que a empresa *Market Maker* conseguiu uma redução do seu ciclo de desenvolvimento em cerca de 50% e uma diminuição de custos de até 70%. Em outro relato, este da Thomson-CSF [17], percebe-se que para se alinhar às iniciativas de SPL existe a necessidade de se escolher antecipada e corretamente as ferramentas que serão usadas ao longo do projeto, além da necessidade de treinamento da equipe que for selecionada a conduzir o projeto.

Em se tratando das iniciativas Americanas (EUA), é possível apontar aquelas que ajudam a definir a teoria de linhas de produto de software, sendo a de maior interesse da comunidade a desenvolvida pelo *Software Engineering Institute* (SEI) [22]. Seu programa, o *Product Line System Program*, iniciou em 1995 e é composto por três iniciativas, a saber:

(1) *Product Line Practice* (PLP);

(2) *Software Architecture Technology* (SAT); e

(3) *Predictable Assembly from Certifiable Components* (PLCC);

Dos resultados alcançados desde então, observa-se que houve uma formalização da teoria de SPL convergindo para um modelo denominado *"A Framework for Software Product Line Practice"* (doravante denominado SPLP) [23]. Nesta proposta do SEI, busca-se a criação de um modelo capaz de influenciar as organizações em seus processos de software, tendo como meta a capacitação e a criação de soluções com

baixo risco global agregado e alto retorno do investimento [24]. O SEI determina que a organização foque nas três atividades apresentadas na Figura 3, que destacam:

(1) o desenvolvimento formal dos ativos a serem reutilizados;

(2) o desenvolvimento do produto a partir dos ativos reutilizáveis;

(3) a gerência destes ativos de modo a serem utilizados e implantados nos produtos.

A condução destas atividades deve ser fluida e se manter em movimento, sempre buscando critérios de correlação e regras gerais que sejam capazes de montar um modelo especializado e que permita a criação de famílias de software que compartilhem características semelhantes.

Figura 3: As três atividades essenciais, assim como vistas pelo SEI, para implantação de linhas de produtos de software (adaptado de SEI [25]).

Ao entender como as três atividades essenciais se relacionam, e como podem ser utilizadas dentro do contexto de SPL, observa-se a necessidade de composição teórica de outras especificações que, no final, permitem a utilização real do framework (Figura 4). Tais especificações são denominadas *"Áreas de Prática"*, e são descritas a seguir:

- Áreas de Prática de Engenharia de Software: que determinam o tipo de tecnologia que será utilizado no processo de institucionalização e uso de um SPL, sendo elas:
    - Definição da arquitetura.
    - Avaliação da arquitetura.
    - Desenvolvimento de componentes.
    - Mineração dos artefatos existentes.
    - Engenharia de requisitos.
    - Integração de sistemas de software.
    - Teste.
    - Usar software externo disponível.
- Áreas de Prática de Gerência Tecnológica: que determinam os procedimentos gerenciais que serão aplicados para criação, monitoramento e controle das tecnologias identificadas nas áreas anteriores e que neste caso são apresentadas como:
    - Gerência da configuração.
    - Análise de desenvolvimento, compra e mineração.

    - Medição e rastreamento.
    - Identificação de processos.
    - Definição de escopo.
    - Planejamento tecnológico.
    - Gestão de risco.
    - Ferramentas de apoio.

- Áreas de Prática de Gerência Organizacional: que controlam o esforço aplicado pela organização na institucionalização da SPL. Entende-se que são:
    - Criação de um caso de negócio.
    - Gestão da interface com o usuário.
    - Estratégias de desenvolvimento e aquisição.
    - Levantamento de fundos financeiros.
    - Lançamento e institucionalização.
    - Análise de mercado.
    - Operação.
    - Planejamento organizacional.
    - Gerência de risco.
    - Organização e estruturação.
    - Prospecção tecnológica.
    - Treinamento.

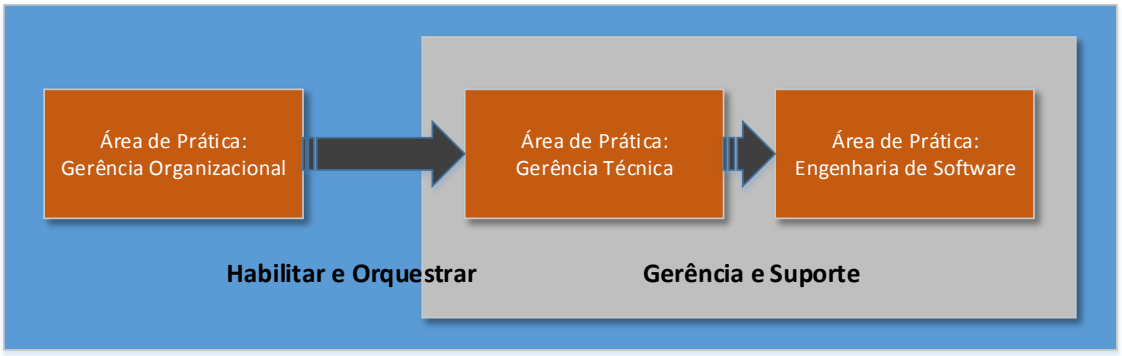

Figura 4: Relacionamento das áreas de prática (adaptado de [25])

2.2. O reuso de software como abordado em modelos de processo

Em complemento ao arcabouço teórico que ajudará a sustentar a proposta de abordagem para reuso de modelos conceituais, os três principais modelos de software existentes atualmente, o MPS-BR [8], o CMMI [25] e a ISO/IEC 12207 [26], foram avaliados pela formalização que possuem das atividades que um projeto de software deve conter para reuso.

O reuso como abordado pelo MPS-BR

A formalização das atividades de reuso em software é importante para que haja comprometimento dentro da organização e para que suas atividades sejam oficializadas dentro do processo de desenvolvimento adotado pela organização. Com este propósito, observa-se como o MPS-BR [8], em seu nível E, denominado Parcialmente Definido, apresenta o processo de Gerência de Reutilização (doravante denominado GRU) e possui cinco atividades, a saber:

1. GRU 1: define em uma documentação formal a estratégia de gerenciamento de ativos que serão utilizados no processo de reutilização. Como descrito na norma, este processo contempla a definição do que deve

ser considerado um ativo reutilizável, os critérios para aceitação, certificação, classificação, descontinuidade e avaliação de ativos reutilizáveis.

2. GRU 2: define os mecanismos de armazenamento e recuperação de ativos reutilizáveis que serão utilizados pela organização.

3. GRU 3: define como devem ser registrados os dados de utilização dos ativos reutilizáveis.

4. GRU 4: define como os ativos reutilizáveis serão mantidos, e como será o controle das modificações ao longo do seu ciclo de vida.

5. GRU 5: finalmente, define como os usuários de ativos reutilizáveis são notificados sobre problemas detectados, modificações realizadas, novas versões disponibilizadas e descontinuidade de ativos.

O reuso como abordado pelo CMMI

Formalmente, até o momento, assim como descrito pelo próprio órgão regulador, o CMMI não possui um processo específico para reuso em software. Entretanto, existe uma iniciativa denominada "System Family Maturity Framework" (FMF) que foi desenvolvida com o intuito de estabelecer uma extensão ao modelo do CMMI e que introduz processos e atividades diretamente associados ao reuso em software.

Resultado dessa iniciativa é a proposta de extensão CMMI-SFE [27] que representa as dimensões dos processos relacionados ao "Family Evaluation Framework" (FEF). Seu objetivo é o de ser um documento adicional aos processos do CMMI, incluindo os processos necessários para que a organização consiga formalizar internamente o SPL.

O reuso como abordado pela ISO/IEC 12207

Publicada originalmente em 1995, a ISO/IEC 12207 [26] em conjunto com suas emendas também define um processo formal para reuso em software. Em sua estrutura o processo de reuso é definido como um Processo Organizacional (Figura 5) composto por atividades exploratórias e sistemáticas que buscam identificar oportunidades de reuso dos artefatos de software construídos ao longo do projeto.

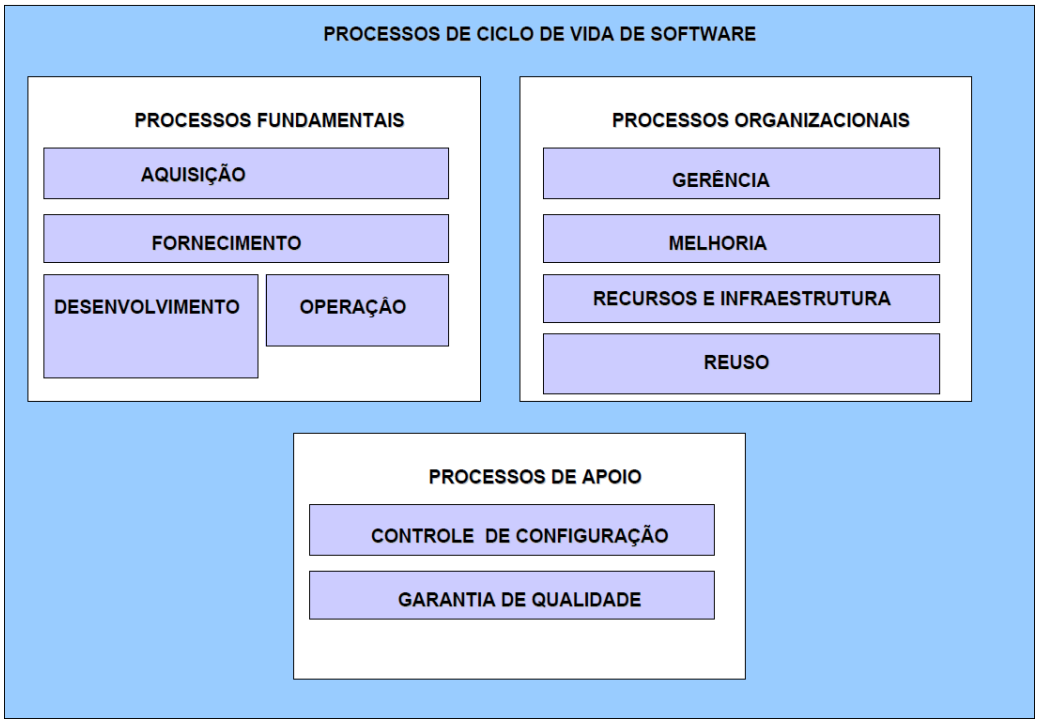

Figura 5: Processos de software como definidos pela ISO/IEC 12207 (adaptado de [26]).

As atividades para o processo de reuso são:

- Gerência dos Ativos, que tem como objetivo gerenciar o ativo reutilizável do momento da sua criação até a sua desativação. Subatividades conhecidas:
    - Implementação do processo.
    - Definição, armazenamento e recuperação dos ativos.
    - Gerenciamento e controle do ativo.
- Gerência do Programa de Reuso, que formaliza o planejamento, a gerência, o controle e o monitoramento das atividades que a organização estabelece em

seu processo de desenvolvimento como sendo específicas de reuso. Subatividades conhecidas:

  - Iniciação: conjunto de atividades para condução do reuso.
  - Identificação do domínio: conjunto de atividades para escolha do domínio de negócio candidato ao reuso.
  - Avaliação do reuso: conjunto de atividades de identificação e qualificação do conjunto de artefatos que serão candidatos para o reuso.
  - Planejamento: conjunto de atividades de planejamento para o reuso, destacando-se as ações de priorização, definição de escopo, arquitetura, disponibilidade técnica da equipe e viabilidade financeira.
  - Execução e controle: conjunto de ações que garantirão a condução das atividades de reuso dentro do projeto de desenvolvimento.
  - Revisão e avaliação: conjunto de ações que buscam avaliar os resultados das atividades de reuso conduzidas dentro do projeto de desenvolvimento.

- Engenharia de Domínio, que por fim determina como a organização deve desenvolver e manter seus modelos de domínio, arquiteturas e todos os artefatos relacionados ao contexto do projeto para que possam ser candidatos ao reuso. Subatividades conhecidas:
  - Implementação do processo.
  - Análise de domínio.
  - Projeto.
  - Fornecimento dos ativos.

- Manutenção dos ativos.

A escolha do processo de gerência

Do que foi apresentado pelos três modelos de processo, dois se destacaram: o MPS-BR e a ISO/IEC 12207. Ambas apresentaram em detalhes suas abordagens de reuso, sendo bem completas e adequadas à proposta desta tese.

De modo a se limitar o escopo de abrangência e análise dos resultados da tese, escolheu-se o MPS-BR, por ser um processo nacional. Caberia, em trabalhos futuros, definir como os elementos aqui apresentados se enquadrariam no processo de reuso da ISO/IEC 12207.

2.3. Desenvolvimento orientado a modelo

O desenvolvimento de software orientado a modelos (doravante denominado MDD) [28] surge com a visão de fornecer independência de plataforma aos elementos que constituem a solução de software a ser gerada a partir do reuso do modelo.

Há um esforço de se separar estes elementos visões distintas:

- A visão de Modelo Independente de Computação (ou Computation Independent Model, doravante denominado CIM).
- A visão de Modelo Independente de Plataforma (ou Platform Independent Model, doravante denominado PIM).
- A visão de Modelo Específico de Plataforma (ou Platform-Specific Model, doravante denominado PSM.

- A visão de codificação.

Como visto em [29], a visão do CIM apresenta apenas os requisitos de sistema e ignora detalhes de implementação e estrutura. Na visão do PIM tem-se o início do desenvolvimento sob um paradigma de meta-modelagem. Neste momento há a criação de um modelo conceitual com o qual os responsáveis pelo projeto estão apenas interessados no domínio de aplicação. Em um segundo momento, tem-se a criação de um PSM, no qual os responsáveis pelo projeto irão adicionar elementos mais próximos a arquitetura e plataforma associadas ao sistema. Módulos, componentes, pacotes, classes, entre outros, são alguns dos elementos que, melhor detalhados, oferecem mais recursos para as máquinas geradoras de código criarem as soluções parciais.

Finalizando este ciclo, há o processo de geração de código. Baseando-se em uma arquitetura pré-estabelecida e embarcada no agente de geração de código, ocorre a construção parcial do sistema, contendo os pontos de extensão necessários para a complementação das regras de negócio. É a partir deste ponto que o desenvolvedor entra em ação e passa a ter a responsabilidade de seguir com a arquitetura criada e não ultrapassar os limites que lhe são permitidos para inserção do código representativo das regras de negócio. A Figura 6 apresenta o esquema padrão para este tipo de desenvolvimento.

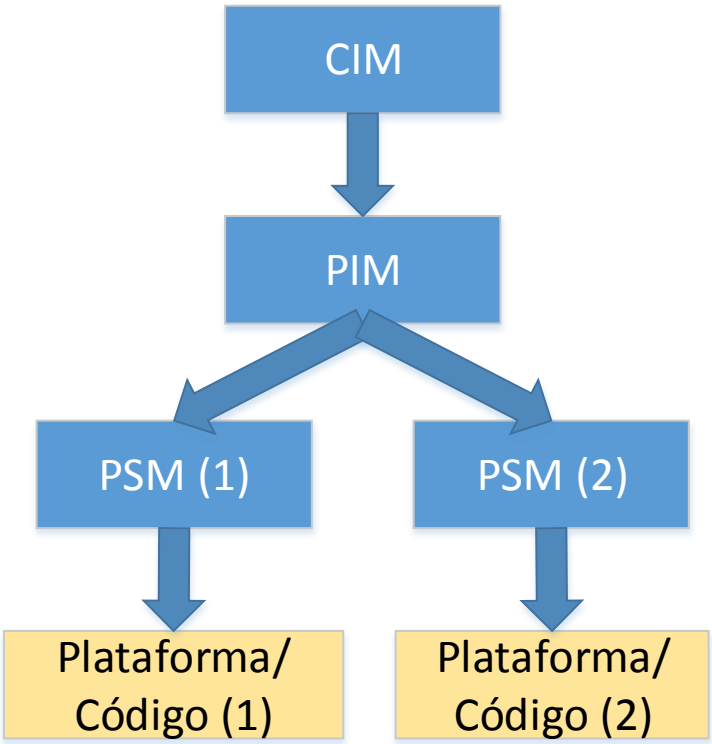

Figura 6: Modelo de transformação do MDA (adaptado de [2]).

Estas ferramentas se propõem, portanto, a diminuir a dependência do projeto em relação ao programador, minimizando a quantidade de código que é gerada por ele [29]. Os modelos conceituais passam a exercer um papel mais ativo e são utilizados como entrada direta a um processo sistêmico de geração automatizada de código.

Também como apontado por [30], inúmeras são as empresas que, ao utilizarem estes recursos, observaram ganhos consideráveis em termos de tempo e custo de desenvolvimento, já em [31], por exemplo, é possível observar ganhos de até 50 vezes menos linhas de código criadas diretamente pelo programador.

Vantagens e desvantagens de abordagens orientadas a modelos

Existem vantagens e desvantagens em se utilizar esse tipo de abordagem para o desenvolvimento de sistemas. Dentre as principais vantagens, e como apresentadas por [32], [33] e [34], destacam-se:

- Produtividade: como parte do projeto é automatizado, a tendência é que haja aumento de produtividade ao longo do ciclo de vida do projeto. Essa produtividade está diretamente associada ao percentual global de linhas de código produzidas pela máquina geradora em relação ao que é gerado pelo ser humano. Quanto maior o percentual produzido automaticamente maior a produtividade associada ao uso da tecnologia.

- Portabilidade: típico de abordagens de desenvolvimento MDD, de um modelo conceitual é possível associar máquinas geradoras para diferentes tipos de linguagens e paradigmas de programação.

- Interoperabilidade: novamente, por ser capaz de gerar código independente de plataforma, também é capaz de gerar conectores e módulos de troca de mensagens para acoplamento em barramentos e/ou infraestrutura de comunicação distintos.

- Manutenção e documentação: benefício direto do uso de modelos como elemento principal do desenvolvimento está associado à carga de informação que neles é embarcada. Nesta tecnologia a forma como os modelos são criados é fator determinante do sucesso do projeto, garantindo atualização constante das informações ali presentes, facilitando manutenções futuras.

- Comunicação: como o modelo é o principal artefato, a comunicação é facilitada, pois são utilizados meios abstratos e de alto nível, longe do código, o que ajuda no processo de comunicação.

- Reutilização: objetivo inicial desta tese, a reutilização é o foco dentro de uma abordagem que utiliza modelos conceituas, dentro de um cenário de agrupamento em termos de nichos de negócio, ou domínios de negócio.

- Qualidade: aplicando-se técnicas de verificação e avaliação, torna-se possível aumentar a qualidade final do produto. Processos de auditoria de modelos e testes de código podem ser utilizados e automatizados, permitindo maior controle da qualidade.

Em contrapartida, quando se avaliam as desvantagens, como apreciado por [28], é possível destacar:

- Rigidez: abordagens desse tipo tem a propensão de dar maior rigidez ao produto final, pois o código gerado fica "fora do alcance" do desenvolvedor.

- Complexidade: o autor indica que as ferramentas necessárias para se compor uma abordagem MDD em seu potencial completo, impõem um aumento de complexidade no processo, pois demandam conhecimento específico e ajustes no processo de desenvolvimento utilizado pela organização.

- Desempenho: o desempenho, apesar de contestável, determina que, por conta da geração de código, há a criação de código desnecessário, o que no final pode causar perda de desempenho.

- Curva de aprendizado: apresenta o conceito que abordagens desse tipo sofrem com o acúmulo de informações e ferramentas, o que causa um aumento relevante na curva de aprendizado associado a inclusão de novas pessoas ao projeto.

- Alto investimento inicial: finalmente, o autor discorre sobre a necessidade de se investir mais que em projetos convencionais de software. Sua visão é de que para se construir a infraestrutura de apoio ao projeto, há que se empenhar mais tempo e esforço.

Em geral, este tipo de tecnologia oferece mais vantagens do que desvantagens; destacando-se a sua capacidade de melhorar a qualidade do produto final, por meio do controle dos riscos e do código que é gerado sem a intervenção do desenvolvedor.

A indústria e o uso de abordagens orientadas a modelos

Em se tratando das abordagens existentes no mercado, destaca-se a desenvolvida pela OMG [35], que há anos fomenta a evolução do MDD por meio de suas especificações de meta-modelagem como o Meta Object Facility (doravante denominado MOF) [36]. Tal evolução vem a reboque de toda a teoria desenvolvida para a abordagem de MDD, como visto anteriormente.

Outro padrão é o QVT (Queries/Views/Transformations), também especificado pela OMG em [36]. O QVT é composto por uma linguagem textual e uma notação capaz de representar consultas e transformações baseadas no MOF. Por meio do uso desta linguagem, é possível criar regras de mapeamento entre diferentes tipos de modelos, independente da linguagem com a qual foram descritos.

Outra abordagem relevante é a desenvolvida pela Oracle, e que conjuga as tecnologias de Java Metadata Interface (doravante denominado JMI) e a MetaData Repository (doravante denominada MDR) [37]. Ambos os projetos no âmbito da plataforma NetBeans utilizam o JMI como mecanismo de mapeamento de interfaces MOF para a tecnologia JAVA, o que significa a manipulação de elementos MOF em linguagem JAVA; e o MDR como um subprojeto para manipulação de dados também baseado na especificação MOF da OMG, como apreciados em [35] e [38].

Outro aliado na criação de abordagens MDA é o projeto Eclipse da IBM [39]. Tem-se o Enterprise Modelling Framework (doravante denominado EMF) como uma alternativa a especificação da OMG, tendo como base a criação de uma ferramenta de manipulação de modelos conceituais. Sua abordagem segue um metamodelo denominado Ecore [40], que no início seguia a especificação da OMG, mas evoluiu para um modelo tido como mais eficiente, resultado de avaliações empíricas de projetos em que foi aplicada. Na prática, e como apresentado em [41] utiliza um subconjunto mapeamentos do JMI, otimizando a manipulação em memória principal, tornando-a mais eficiente e simples.

Também associado ao projeto eclipse, há o Generative Modeling Technologies [42] (doravante denominado GMT) que, baseando-se num modelo específico, porém orientado pelos padrões da OMG, tem como destaque as iniciativas: (1) Atlas Transformation Language (doravante denominada ATL), que desenvolve uma linguagem para definição de transformações entre modelos; (2) o MOF Script, que objetiva a transformação de modelos para texto; (3) a Textual Concrete Syntax (doravante denominada TCS) que visa desenvolver ferramentas para especificação de linguagens específicas de domínio; e (4) o Java Emitter Templates (doravante

denominado JET) que consistem na geração de código baseado em templates, como o especificado por [43].

Por fim, algumas outras abordagens podem ser apontadas como relevantes para a evolução da teoria de MDA/MDD. A Vanderbilt University, por meio de seu projeto de computação por modelos integrados (ou Model Integrated Computing) [44] já alcançou avanços em três áreas relacionadas ao MDD: a modelagem específica de domínio; ferramentas de modelagem; e a criação de um framework de análise; culminando na criação do ambiente de modelagem genérica o GME (Generic Modelling Environment). Finalmente, a Microsoft aparece com uma solução mais focada no conceito de fábrica de software, propondo uma abordagem para reutilização sistemática com MDD. Em sua abordagem a Microsoft lança mão de três elementos: (1) um esquema representativo da fábrica; (2) templates para criação e instanciação da fábrica; (3) um ambiente, apoiado pelo Visual Studio, no qual há estes templates são utilizados no desenvolvimento da solução [44].

2.4. O reuso em software com o apoio de modelos conceituais

Do ponto de vista técnico, parece natural que ferramentas como as de MDD apoiem teorias como as de reuso. A combinação é instigante e promissora, entretanto ainda é algo a ser pesquisada no sentido de se alcançar a maturidade adequada para esse tipo de solução.

Alguns autores, como em [8], [11] e [19], destacam exatamente esse pensamento e discutem como a reutilização pode ser útil para abordagens de MDD. Defendem, portanto, a ideia de que é possível obter resultados significativos com a reutilização quando todos os artefatos (ou ao menos aqueles que são passíveis de reuso) que estão

envolvidos na cadeia de desenvolvimento de um projeto de software são considerados e avaliados quanto ao ganho que irão proporcionar. Logo, nesta e em outras abordagens do gênero, tais como em [45] e [46], percebe-se que o modelo conceitual torna-se fundamental para uma abordagem de reuso que utilize MDD como meio de criação de código. A própria OMG destaca que, apesar de serem áreas distintas, possuem correlação suficiente para serem analisadas e combinadas visando alcançar novos meios de desenvolvimento de software. O objetivo, portanto deve ser a criação de abordagens sistemáticas que consigam alavancar em todo seu potencial esta combinação e extrair os melhores resultados possíveis dentro do escopo de trabalho no qual se está interessado. Percebe-se, portanto, que o caminho mais natural é o de entender como esta união poderá alcançar seu potencial máximo.

O MDD como ferramenta de reuso de modelos conceituais

Como visto em [46], existem abordagens que estendem a utilização do paradigma de MDD, bem como buscam funcionalidades como as de reuso dentro de um processo de automação, porém também demonstram que são viáveis para somente algumas etapas do processo de desenvolvimento de software. Para se maximizar a capacidade de reuso, deve-se assegurar que os produtos gerados são tais que observam padrões específicos de codificação, significando que, por exemplo, a arquitetura seja planejada de tal forma a viabilizar o processo de reuso.

Quando analisado de forma mais abrangente, vê-se que o reuso em software, quando aplicado em modelos conceituais é dificultado, principalmente, quando não se tem uma formalização do seu processo de concepção. Significando que para se garantir que um modelo possa ser reutilizado, é necessário criar uma abordagem de grupamento, uma junção semântica, na qual o reuso faça sentido.

É neste ponto que a ideia de famílias de software entra em cena. A complexidade na criação de modelos conceituais é proporcional ao domínio de negócio que se deseja entender. A partir de um domínio específico pode-se chegar a diferentes subdomínios (subconjuntos de elementos que fazem parte do domínio) logo a capacidade de se categorizar, descrever e armazenar este modelo fará a diferença nas diversas abordagens de reuso criadas para este fim.

Há a necessidade de se garantir que a inclusão de novas regras de negócio, ou a exclusão das mesmas, seja feita sem que afete o produto final gerado pela ferramenta de MDD, e como definido em [46], os elementos necessários, e que devem ser controlados, para se implantar abordagens de reuso em conjunto com técnicas de MDD são:

(1) a obsolescência dos modelos, que são criados no início das diversas iterações em um projeto para ganhar entendimento do problema no qual se está trabalhando; e

(2) a complexidade e volatilidade das regras de negócio, que contribuem para dificultar a implantação deste tipo de abordagem [47].

Se há um esforço para desenvolver um software, há um esforço muito superior de se acompanhar as mudanças que ocorrem ao longo do tempo. Esse fator é fundamental para um bom processo de reutilização de modelos, pois, na grande maioria das empresas de software, quando há uma alteração, nem sempre essa alteração é replicada no modelo.

2.5. Arquiteturas orientadas a serviços

Arquiteturas Orientadas a Serviço (SOA) representam uma abordagem para a utilização dos recursos de TI em apoio ao negócio da organização. SOA apresenta a

renovação em níveis técnico e organizacional, da forma como a TI deve ser usada para apoiar o processo de alinhamento estratégico da organização [48].

SOA significa pensar em termos de serviços e de aplicações baseadas em serviços e, em seu nível mais abstrato, é composto de três elementos que representam papéis distintos de interação:

- O Consumidor do Serviço;
- O Prestador de Serviço; e
- O Registro do Serviço.

O Modelo Operacional Triangular

É o modelo que representa a interação entre os três elementos de SOA [2] e [49]. Dentro do modelo organizacional de uma arquitetura orientada a serviço (Figura 7), eles se comportam da seguinte forma:

O provimento do serviço (Provedor): determina o comportamento daquele que está disponibilizando o serviço, ou seja, é considerado o dono do serviço, o que significa que é o responsável por fornecer a infraestrutura de acesso (via rede), e é capaz de responder a requisições internas (Intranet) e externas (Internet).

O consumo do serviço (Consumidor): determina o comportamento daquele que assume o papel do cliente da organização provedora do serviço, podendo ser representado por uma pessoa, uma organização, uma máquina ou um componente de software. A personificação do papel de consumidor ou cliente é irrelevante no sentido de que um consumidor representa aquele (ou aquilo) que localiza um serviço, entende seu protocolo de operação e se utiliza desse protocolo para executá-lo.

O registro do serviço (Registro): determina o comportamento que a organização deve ter para divulgar o seu serviço, e o do cliente que deve proceder para localizar o serviço desejado. É responsável por gerenciar os repositórios que armazenam informações sobre os serviços e organizações que os fornecem.

Figura 7: O Modelo operacional triangular (extraído de [6]).

O Modelo Operacional em Diamante

O modelo operacional triangular [6] tem sido o modelo tradicional para aplicações orientadas a serviço desde os primórdios da teoria. Dentre várias limitações, destaca-se o papel passivo do registro, principalmente quando se avalia sua capacidade de representar e organizar as informações necessárias para uso dos serviços. Tal comportamento foge um pouco ao que vemos atualmente em termos de busca e aquisição de informações. Registros são componentes importantes quanto ao conjunto de informações que devem (ou deveriam) armazenar de maneira a retratar adequadamente os serviços que ali estão cadastrados.

O modelo em forma de diamante, assim como apresentado em [50], expande o modelo triangular com a adição de um novo ator denominado Proxy de Invocação do

Serviço. O proxy tem como objetivo enriquecer o modelo operacional na medida em que os metadados associados aos serviços contenham mais e melhores informações a respeito de como um cliente deve utilizar um serviço de um provedor.

A Figura 8, demonstra como o proxy interage dentro do modelo de maneira a viabilizar o descobrimento e a seleção dos serviços. Tecnicamente, o proxy também é um Web Service o que facilita a aderência à evolução de soluções que, atualmente, são baseadas no modelo tradicional.

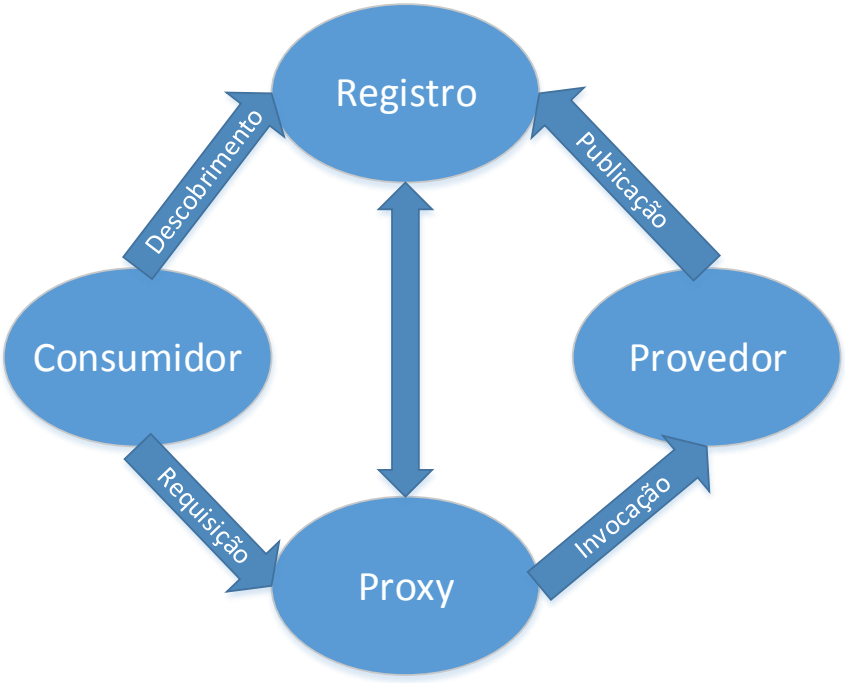

Figura 8: O Modelo operacional em diamante (adaptado de [49]).

O modelo diamante introduz uma nova perspectiva de desenvolvimento de aplicações SOA, pois enriquece a descrição dos serviços em termos de negócio e técnicos, o que melhora os termos de acordo em nível de serviço entre o cliente e o provedor.

SOA como Modelo de Alinhamento entre o Negócio e a TI

Como visto em [51], o alinhamento entre o Negócio e a TI, por meio do uso de SOA, é possível quando se desmembram os elementos de um processo de negócio e os associam a serviços de TI. Este deve ser conduzido dentro do ambiente organizacional, e deve aprofundar os processos e subprocessos de negócio até o nível de atividades. Desse mapeamento são projetados os serviços de TI e desenvolvidos os componentes/módulos que irão interagir com outros módulos, formando o ecossistema de serviços.

Conforme apresentado em [6], [52], [53] e [54], o processo de mapeamento pode ser feito a partir de duas abordagens: de cima para baixo (*top-down*) ou de baixo para cima (*bottom-up*):

A abordagem *top-down* (Figura 9) determina que, a partir de uma visão geral do seu domínio de negócio, a organização deve identificar os processos que lhe são prioritários e, a partir desta enumeração, mapeá-los em serviços de TI.

Em contra partida, dentro de uma abordagem *bottom-up*, a organização, inicialmente, identifica os ativos de TI disponíveis e, baseando-se nesta composição e disponibilidade de recursos, determina os tipos de serviços que será capaz de desenvolver e, só então, segue com a priorização dos processos de negócios que se ajustam a esta realidade.

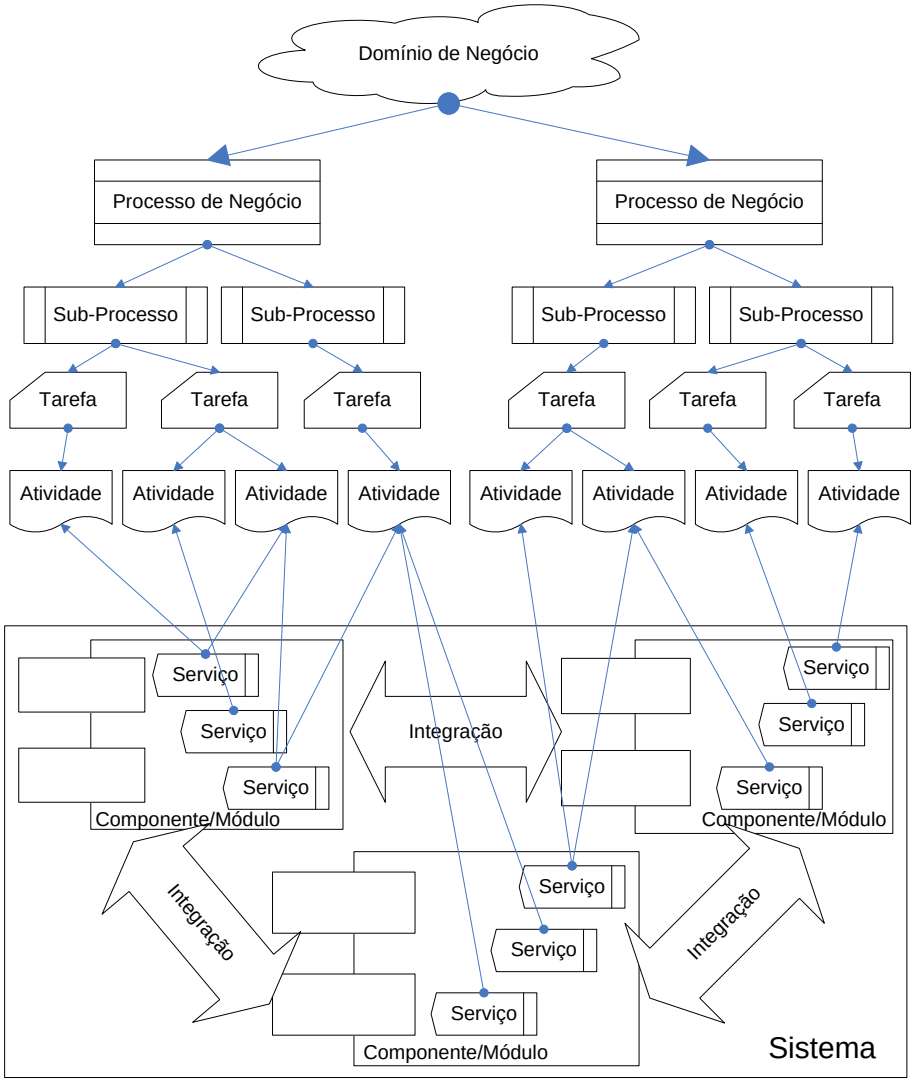

Figura 9: Modelo geral de alinhamento entre o Negócio e a TI, através do uso de arquitetura orientada a serviço. (Extraído de 6)

O Barramento de Serviço - Enterprise Service Bus (ESB)

Um barramento de Serviço (ou Enterprise Service BUS, doravante denominado ESB) é um modelo conceitual tecnológico composto por padrões e ferramentas de modelagem e desenvolvimento, que constroem, unem e conectam serviços, aplicações e recursos de TI da organização [55]. Um ESB interfere no modo como o arquiteto deve projetar sua arquitetura e, normalmente, provê um modelo abstrato de troca de mensagens para integração e comunicação dos serviços. Vale destacar que um barramento de serviços não representa uma arquitetura orientada a serviços, mas um

conceito que viabiliza o uso de SOA como infraestrutura de soluções corporativas. Deve-se pensar um ESB como aquele que regulamenta a forma como os serviços se comunicam e interagem uns com os outros.

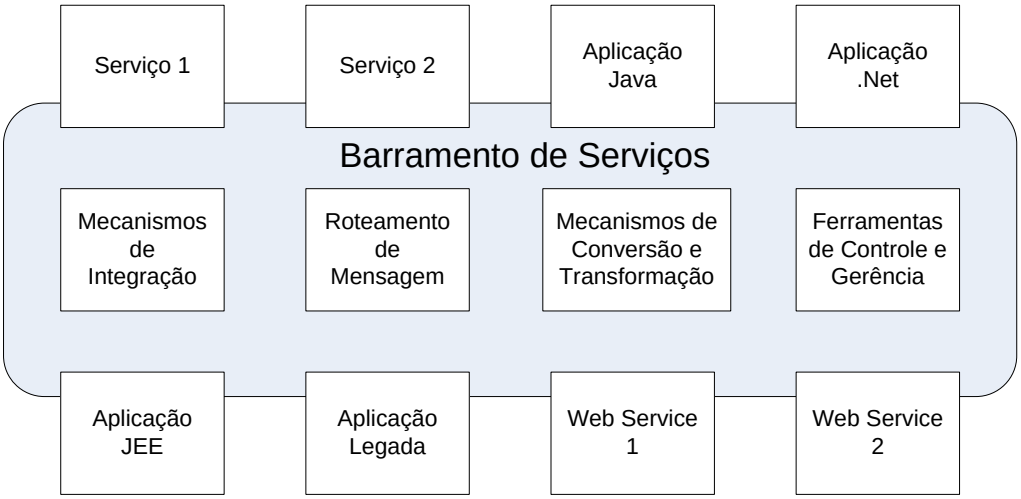

Figura 10: Barramento de serviço composto por tecnologias utilizadas na indústria.

Uma solução para barramentos de serviço (Figura 10) deve ser definida de maneira a permitir a integração de aplicações desenvolvidas em diferentes linguagens e plataformas, e deve estar alinhada com as atividades comuns em um ciclo de vida SOA. O barramento de serviços deve garantir que, por exemplo, aplicações desenvolvidas em Java consigam interagir com aplicações construídas em .Net, ao mesmo tempo que Web Services possam se comunicar com aplicações legadas.

Arquitetura de Referência SOA (Reference Architecture - SOA-RA)

Os elementos fundamentais para a concepção de arquiteturas de referência em SOA são aqueles responsáveis por fornecerem os mapeamentos de negócio, e os requisitos técnicos para o uso dos padrões e ferramentas envolvidas no seu processo de modelagem e construção. Da teoria é possível entender que são muitos os fatores que

influenciam no desenvolvimento de soluções SOA, entretanto para que se consiga obter soluções verdadeiramente alinhadas às necessidades de negócio, é necessário analisar, para cada um dos quatro elementos apresentados na Figura 11, o nível de proficiência e capacidade da organização. Esse mapeamento fornece a ideia de maturidade de uma organização em relação às soluções SOA que produz [6].

Figura 11: Elementos fundamentais para a criação de uma arquitetura de referência.

- Domínio de Negócio: determina as estratégias, objetivos e prioridades de negócio. Conceber um modelo de negócio correto é essencial para a implementação de soluções de TI de sucesso;

- Soluções Tecnológicas: Representa o conjunto de padrões e ferramentas que farão parte da infraestrutura de SOA;

- Arquitetura de Informação: uma solução de TI que não tenha passado por um processo de identificação e modelagem da arquitetura de informação provavelmente incorrerá em uma solução deficiente. O processo de definição da arquitetura de informação é importante para o processo de levantamento de requisitos, que gera insumo para o processo de modelagem e projeto, e que por

sua vez afeta a construção da solução. Neste caso, ao falhar no início do processo de desenvolvimento a organização propagará o erro até o produto final;

- Governança: identificar as responsabilidades, direitos e deveres dos atores envolvidos na construção de soluções SOA, é essencial para a concepção de soluções verdadeiramente alinhadas ao negócio.

Elementos de uma Arquitetura de Referência - AR

Como apresentado em [76], uma arquitetura de referência (AR), Figura 12, auxilia na criação de componentes de arquitetura reutilizáveis, caracterizados em três camadas: camada de visualização (ou cliente), camada de serviço e camada de negócio.

- *Camada de Visualização (ou Cliente):* determina que todos os serviços sejam acessados por aplicações que ofereçam interface de acesso e controle. Exemplo de camada de visualização é a WEB.

- *Camada de Serviço:* determina a camada em que os serviços são publicados no registro, acessados via o barramento e gerenciados pelo orquestrador. Viabiliza os mecanismos de integração e colaboração, e fornece a infraestrutura necessária para a criação das soluções de negócio mediante a formação dos grupos semânticos dos serviços de TI.

- *Camada de Negócios:* esta camada pode ser vista como aquela que possui os componentes que contém as regras de negócio. Normalmente são desenvolvidas em JEE e acessadas local ou remotamente pelos serviços.

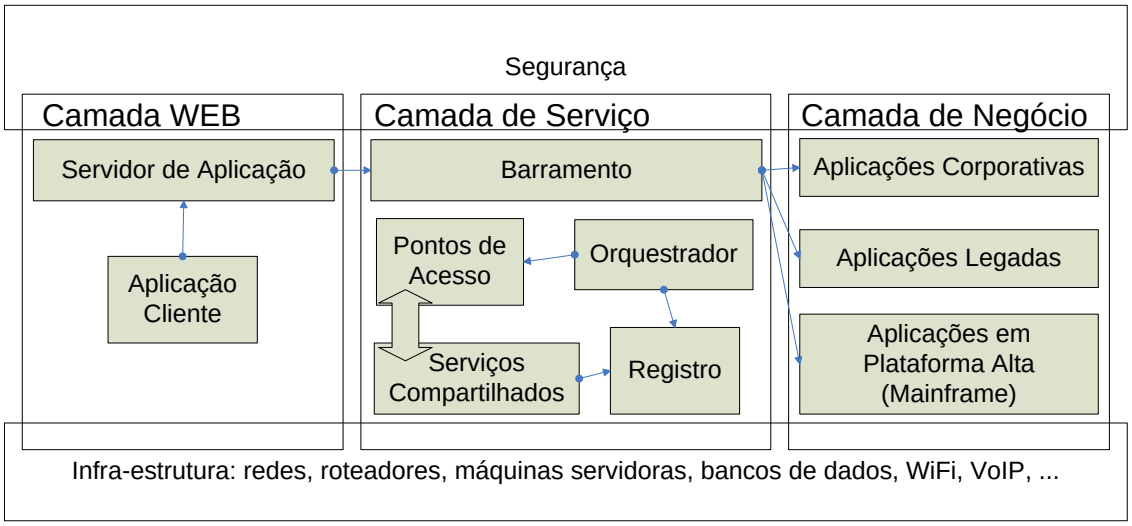

Figura 12: Arquitetura de referência para soluções SOA, observando o paradigma de desenvolvimento de soluções web.

Vantagens no uso de arquiteturas de referência SOA residem no fato de conseguir otimizar os investimentos em soluções de TI, e ajudar a pavimentar o caminho para a inovação, por exemplo, por meio da incorporação de soluções como a de computação nas nuvens, entre outras.

2.6. Trabalhos relacionados ao reuso de modelos

Como abordado em [44], a modelagem conceitual é uma etapa importante no processo de desenvolvimento de software. Desde iniciativas de aplicação de soluções de TI em processos de negócio de pequenas e médias empresas, até grandes sistemas desenvolvidos por gigantes desenvolvedoras de software, o uso de modelos conceituais é fundamental para o entendimento do que precisa ser desenvolvido.

Como explicado por [56], os procedimentos empregados na modelagem conceitual formal definem como os conceitos associados ao negócio guiarão a construção do sistema. Entende-se, portanto, que ao se embarcar o conhecimento em um

conjunto de artefatos formais, perpetuam-se as regras inerentes ao negócio, permitindo que sejam consultadas e reutilizadas em momentos futuros.

Como visto em [57], ao pensar no uso de modelos conceituais em auxílio a iniciativas de reuso em desenvolvimento de software, o primeiro elemento a ser considerado é a representação deste modelo em um nível de abstração de baixa complexidade, que ajude a lidar com as diferentes visões de negócio existentes. Busca-se, então, extrair informações em um patamar de generalização que permita avaliar a real capacidade de se reutilizar o conhecimento embarcado nos modelos conceituais.

A meta-modelagem permite extrair informações que ajudam a avaliar a diferenciação entre modelos que buscam representar informações dentro de um mesmo contexto de negócio. Essa capacidade de generalização de conceitos é fundamental para viabilizar o reuso, pois determina os limites de transformação que um modelo pode alcançar.

O mapeamento de diferentes meta-modelos a partir de um modelo conceitual por meio da identificação das suas diferenças e variações [58] também é uma ação importante e ajuda a determinar como modelos resultantes da meta-modelagem serão aderentes às soluções definidas posteriormente. Tal mapeamento é conduzido pela definição dos níveis de variabilidade dos modelos, utilizando atributos e parâmetros assim como definidos em [59] e [60]. Este mapeamento pode ser feito em dois níveis de abstração: generalização e especialização; ambos vistos como mecanismos de modificação que ajudam a definir as etapas para desenvolvimento, uso e reuso dos modelos.

No tocante a generalização, busca-se adicionar informações aos modelos que os ajudem a representar adequadamente as suas variações, de acordo com o nível de especificação necessária no momento do reuso. Em contra partida, o procedimento de

especialização impõe a necessidade de reduzir a variabilidade do modelo, buscando-se maior aderência com o processo de negócio específico em análise. Ambos são conceitos utilizados pela teoria de SPL e são fundamentos importantes na conceituação e categorização de cada modelo em espectros próprios representativos do negócio.

Neste sentido, a engenharia de domínio [61] ajuda a estabelecer uma linha de raciocínio onde os modelos conceituais são concebidos dentro de uma abordagem como a definida pela Figura 13.

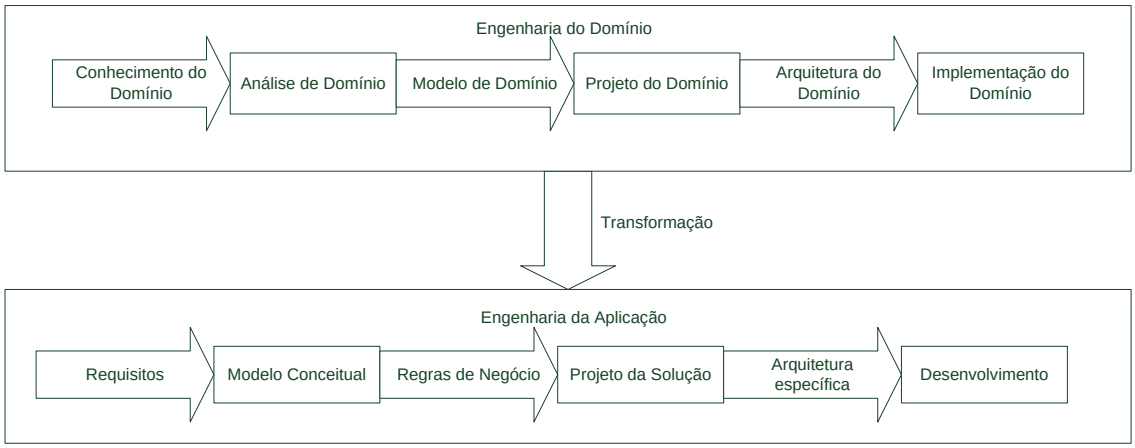

Figura 13: Abordagem de transformação entre engenharia de domínio e engenharia da aplicação.

A Figura 13 aborda as atividades necessárias nos dois níveis de engenharia para se partir de uma análise de domínio e se chegar a uma implementação em software. Das atividades de engenharia de domínio chega-se a de engenharia da aplicação, a partir da transformação intrínseca e sistêmica das atividades conhecimento do domínio à sua representação em modelos e código.

Como analisado em [62], de um modelo de domínio, no qual são especificados os requisitos atuais e futuros de uma família de aplicações, considerando o conhecimento do domínio e as experiências de desenvolvimento extraídas a partir de especialistas, chega-se ao projeto de domínio, que aborda a arquitetura e o modelo detalhado do domínio. Da transformação, segue-se o mapeamento dos conceitos de projeto para um a

determinada plataforma, na qual ocorrerá o processo de desenvolvimento abrangendo as atividades definidas pela engenharia da aplicação.

O entendimento dos conceitos que farão parte da solução se inicia a partir da identificação da origem do conhecimento do domínio, resultando na engenharia deste domínio [5] e [63], onde são executadas as atividades de análise, projeto e, posteriormente, implementação. O momento de transformação implica numa especialização de conceitos onde o conhecimento do negócio é mapeado e representado logicamente por uma série de artefatos concebidos durante a engenharia da aplicação. A engenharia de domínio torna-se peça fundamental na determinação dos níveis de variabilidade e da identificação das informações que serão necessárias para a criação dos meta-modelos viabilizando a estruturação dos conceitos em famílias de produtos, como também apresentado em [64].

A estruturação dos meta-modelos como mecanismos de auxílio ao procedimento de reuso de modelos conceituais, pode ser analisado em [65], onde se observa um processo que leva em consideração a especialização dos conceitos em um modelo alvo a partir da generalização obtida durante as atividades de engenharia de domínio. A Figura 14 demonstra como este procedimento deve ser planejando saindo de um meta-modelo e alcançando o modelo definitivo que será alvo de reuso.

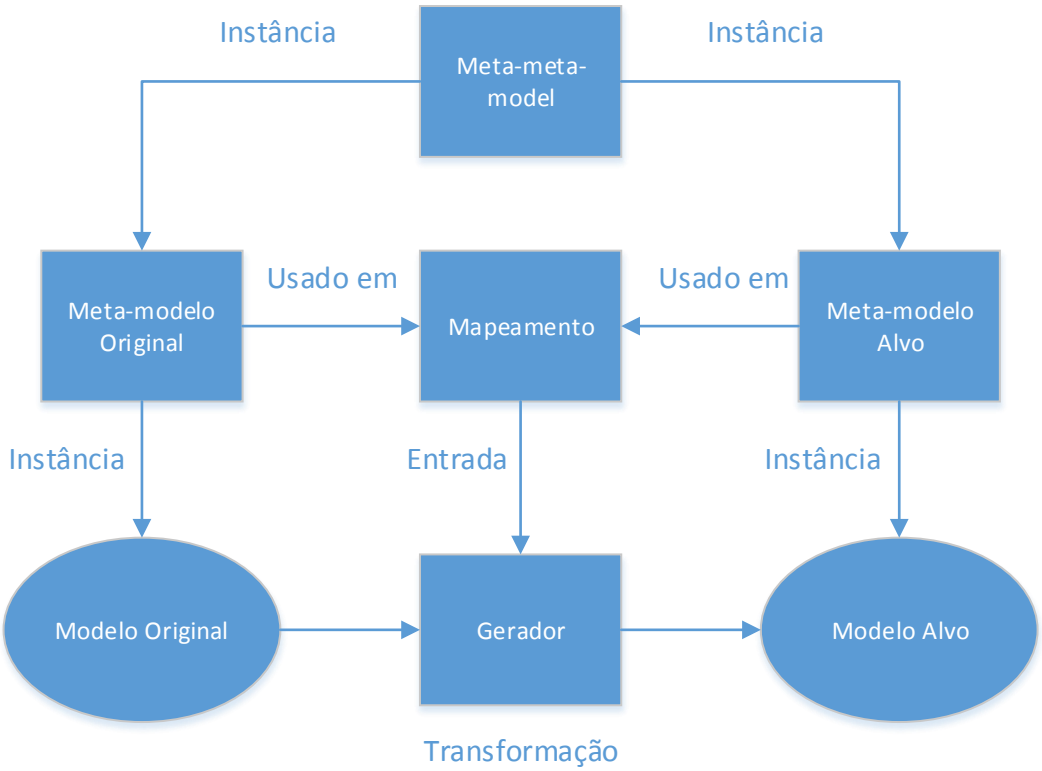

Figura 14: Da generalização a especificação dos conceitos e concepção do modelo conceitual. (adaptado de 65)

A partir do meta-modelo de domínio, ou seja, do modelo que define as principais informações relacionadas ao domínio, é possível instanciar e mapear os modelos específicos de origem e destino para aquele domínio, permitindo que o procedimento de transformação seja realizado dentro dos limites de variação estabelecidos pelos atributos de variabilidade.

Iniciativas de Reuso de Modelos

Abordagens para o reuso de modelos datam desde o projeto Draco e fornecem fundamentação teórica abrangente e completa da relação entre construção de software e utilização de componentes preexistentes [66].

Como bem apresentado em [67], o sistema transformacional Draco oferece uma visão holística de como domínios de negócio podem ser descritos formalmente em termos de objetos e operações, refletindo em si as regras de negócio necessárias ao desenvolvimento da solução. Esta pode ser vista como uma abordagem de uso de descrição de domínios de negócio para a construção automatizada de artefatos de software. Conceitos elementares para a definição e formalização de abordagens de reuso que tenham objetivos semelhantes.

Outra iniciativa que se alinha à necessidade de se garantir a captação das variações representativas do contexto e das regras do negócio é apresentada em [68]. Já em [69], os autores demonstram como a engenharia orientada a modelos (MDE) é útil para a especificação de abordagens que levem em consideração a geração de código a partir de modelos livre de implementação. Como visto na Figura 15 a análise detalhada do domínio de negócio é fundamental para a correta abstração do contexto de negócio. O processo de transformação é sempre iniciado com o isolamento funcional dos elementos mais significativos do negócio, representando-os em seu nível mais puro e livre de especificidades. Consegue-se, portanto, viabilizar o processo de transformação, adicionando os detalhes técnicos, chegando ao momento em que os envolvidos no projeto utilizam o modelo para gerar o máximo de código possível; dependendo da capacidade e do framework e arquitetura embarcados na máquina MDD.

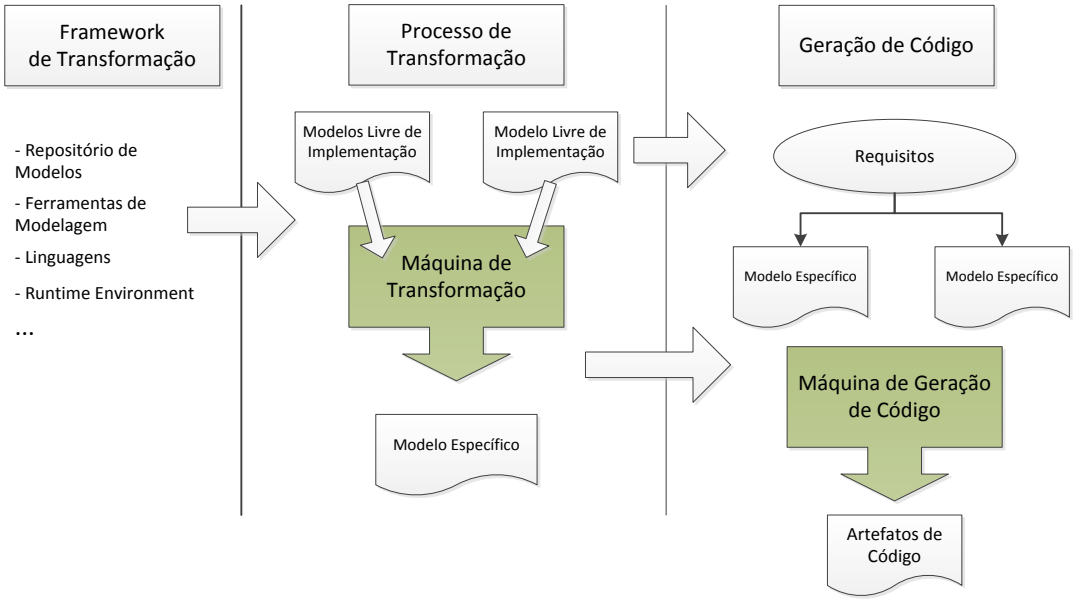

Figura 15: Abordagem de utilização de MDE para reuso de modelos [adaptado de 69].

Adicionalmente, inúmeras abordagens existem em se tratando de reuso de software e modelos. Como apontado anteriormente, em [59] a busca pela melhoria e evolução de abordagens de desenvolvimento que resultem na criação de linhas de produtos é um fator predominante para se encontrar soluções inteligentes ao reuso de modelos. Na abordagem apresentada, o foco em SPL é evidente e os resultados mesclam boa parte dos conceitos utilizados nesta tese, no qual a junção do paradigma de desenvolvimento orientado por modelo à teoria de reuso leva a um novo caminho a se percorrer quando se tem o objetivo de diminuir custos e tempo de desenvolvimento em projetos de software.

Finalmente, a definição, criação e utilização de ontologias podem ajudar na gestão dos níveis de variabilidade que cada modelo conceitual possui quando avaliados sob perspectivas distintas de uso e aplicação. Como apresentado em [65], ao criar categorias semânticas nas quais associam-se os modelos conceituais, abre-se espaço para a indexação dos conceitos mais utilizados em modelos e no desenvolvimento de

software e permite-se abstrair de maneira mais eficiente as características associadas aos níveis de variabilidade daquele modelo.

A Engenharia de Domínio

Teoria e técnicas que devem ser consideradas ao se planejar e, consequentemente, pesquisar abordagens de reuso é a de Engenharia de Domínio (doravante denominado ED). Há anos a teoria associada a engenharia de domínio vem evoluindo e, atualmente, possui inúmeras abordagens que conduzem à sistematização do processo de ED, tais como ODM [68], FODA [69], OODE [70] e FODACom [71].

A Engenharia de Domínio tem garantido ganhos consideráveis no desenho e implantação de soluções que ajudem a viabilizar o reuso [72]. Projetos que focam em ED permitem a produção de aplicações por meio da reutilização de conhecimento acumulado sobre determinada área e oferece apoio e viabiliza a catalogação e preparação dos modelos para (re)uso e consequente construção dos componentes reutilizáveis.

Finalmente, em [73] é possível avaliar uma coleção completa de abordagens e iniciativas que sustentam a proposta desta tese, e exemplifica categoricamente boa parte dos trabalhos relacionados existentes na área de reuso de modelos conceituais, destacando-se as abordagens como o projeto ModelWare que possui resultados expressivos como o MOFScript e ATL [74]; o método Kobra que é mais direcionado a metodologia de linhas de produto de software; e o Pulse-MDD e Pulse-DSSA, que alinham técnicas de MDA/MDD em apoio ao reuso de software [75].

3. Proposta conceitual do serviço de desenvolvimento de software (SDS)

A partir da hipótese apresentada na introdução da tese, objetiva-se criar um ambiente de desenvolvimento que viabilize o reuso de modelos conceituais. Adicionalmente, propõe-se tal ambiente em uma abordagem de “software as a service” pelo fato de se buscar abrangência própria a este tipo de implantação.

A primeira etapa da proposta busca identificar o conjunto de atividades que são necessárias para a composição da abordagem de reuso, sendo assim, a Figura 16 apresenta as atividades que foram definidas para a utilização do ambiente. Observa-se que da modelagem conceitual, até a atividade de geração de código, ocorrem as atividades de verificação, validação, armazenamento e ajustes do modelo para a sua inclusão no ambiente. Uma vez disponível, o reuso do modelo é operacionalizado por meio da geração de pacotes de código, aqui definidos como serviços de TI, que podem ser executados em conjunto dentro de uma visão de integração e troca de mensagens.

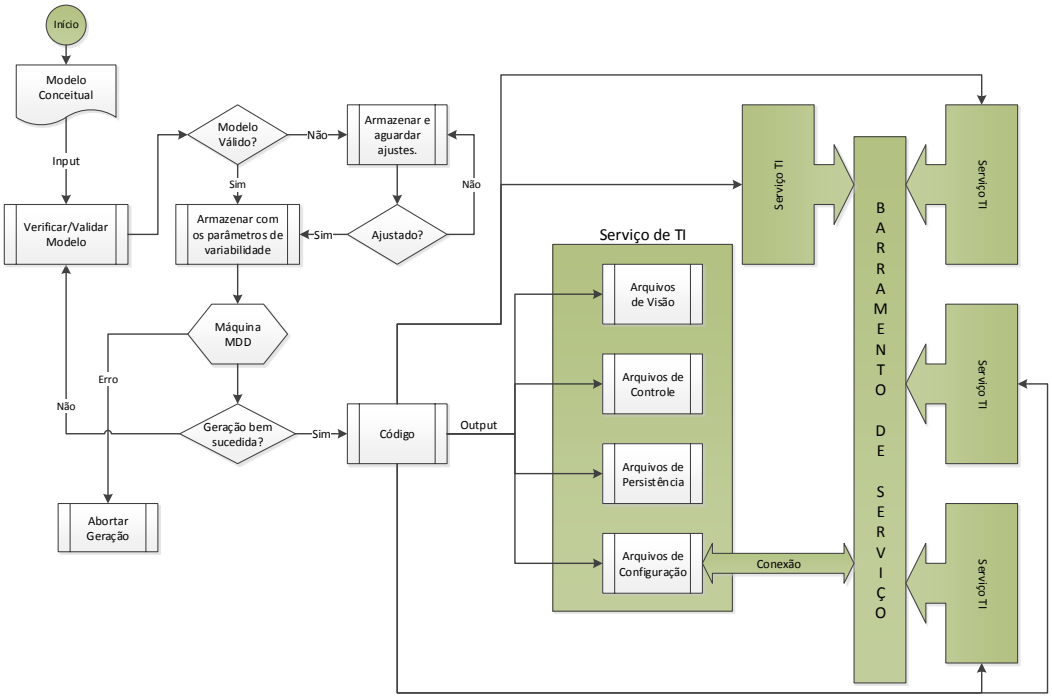

Figura 16: Fluxo de atividades propostas para o SDS.

Para tanto, tem-se que Erl em [49], define que um serviço de TI, de maneira abstrata, pode ser analisado como um encapsulamento lógico dentro de um contexto distinto. Tal abstração ajuda na definição concreta do serviço assim como implementado no SDS, sendo este construído em tecnologia de Web Service. Marzullo em [32], estabelece as regras essenciais para a criação e apresenta que um serviço de TI, criado pelo SDS, é resultado da geração do pacote executável de software contendo elementos de configuração e "deployment" embarcados e preparados para execução em um servidor de aplicação.

Da modelagem ao armazenamento do modelo conceitual, o ambiente permite a sua identificação e inclusão dos parâmetros de variação, determinando um mecanismo de categorização e agrupamento, criando assim grupos semelhantes de modelos. A Figura 17 introduz o modelo de classes utilizado pelo SDS para o armazenamento dos elementos necessários ao reuso, definindo os aspectos de armazenamento existentes no ambiente. Este modelo é apresentado em detalhes na próxima subseção.

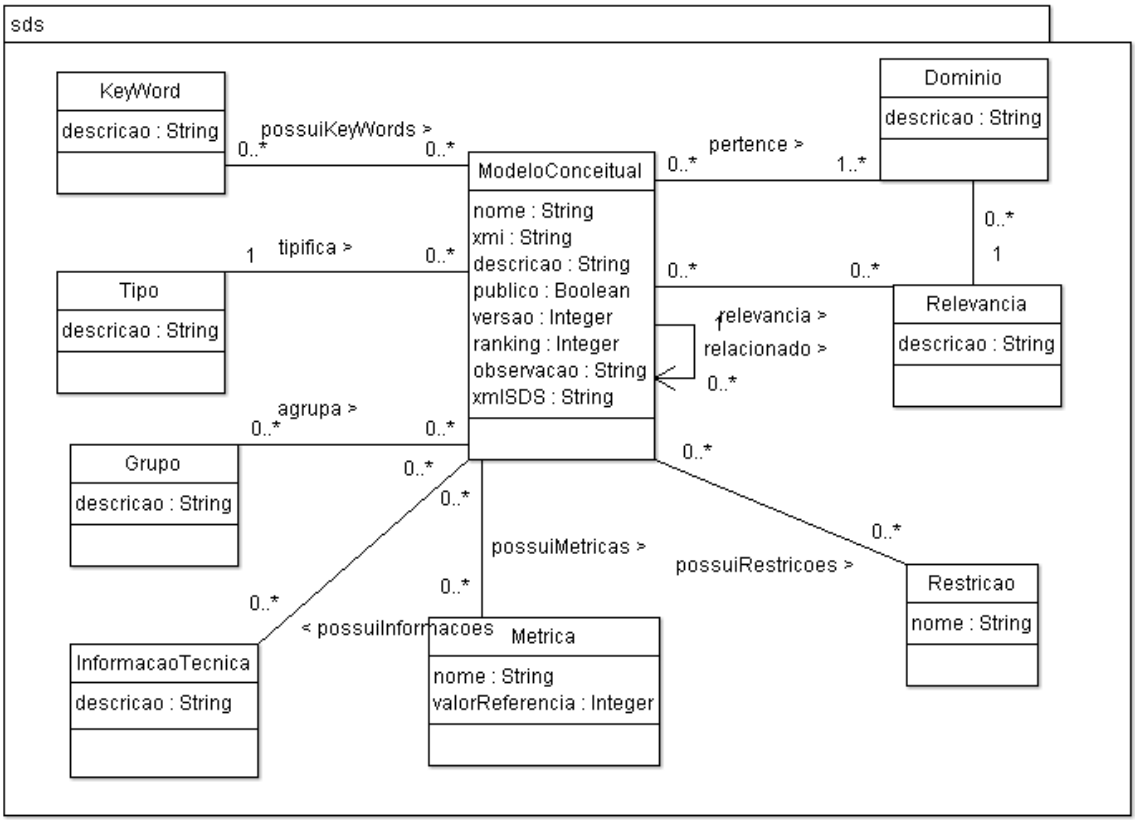

Figura 17: Modelo conceitual do ambiente SDS para armazenamento e viabilização do reuso.

3.1. O modelo conceitual do SDS

No contexto desta tese, o modelo conceitual do SDS é apresentado por meio de um diagrama UML no qual são apresentadas as entidades concebidas para o ambiente de reuso e seus relacionamentos. Adiante, cada entidade é detalhada e suas restrições de integridade identificadas ao longo do projeto.

A Classe "Dominio"

Esta classe representa o domínio no qual um modelo será inserido. É composta pelas informações descritas na Tabela 1.

Tabela 1: Definição da classe Dominio.

Atributo	Tipo	Descrição	Domínio
descricao	String	Define o domínio ao qual o modelo será associado.	Texto livre de até 250 caracteres.

A Classe "ModeloConceitual"

Define o conjunto de informações que devem ser armazenadas para inserção de um modelo conceitual no SDS. É composta pelas informações descritas na Tabela 2.

Tabela 2: Definição da classe ModeloConceitual.

Atributo	Tipo	Descrição	Domínio
nome	String	Atribui um nome para o modelo conceitual inserido no SDS.	Texto livre de até 250 caracteres.
xmi	String	Arquivo xmi em formato específico.	Texto livre de sem limite de caracteres.
descrição	String	Atribui uma descrição ao modelo.	Texto livre de até 1000 caracteres.
publico	Boolean	Determina se o modelo possui acesso público ou privado.	{true, false}
versao	Integer	Determina o número da versão do modelo.	Numérico, sem limite de valor.
ranking	Integer	Determina um valor específico para o posicionamento do modelo dentro do seu contexto de aplicação.	1 a 3, onde: 1 é não prioritário; 2 é indiferente; e 3 é prioritário.
observacao	String	Observações gerais sobre o modelo.	Texto livre de até 1000 caracteres.
xmlSDS	String	Armazena o XML criado pelo SDS para dar agilidade na manipulação das informações em tempo de execução.	Texto livre sem limite de caracteres.

A Classe "Tipo"

Esta classe permite a identificação do modelo nos termos do tipo de XMI que é gerado. Está associado a ferramenta CASE e à versão da UML para geração do XMI. É composta pelas informações descritas na Tabela 3.

Tabela 3: Definição da classe Tipo.

Atributo	Tipo	Descrição	Domínio
descricao	String	Define qual ferramenta CASE criou o XMI e qual versão da UML.	Texto livre de até 250 caracteres.

A Classe "Grupo"

Esta classe determina o agrupamento ao qual o modelo pertence. É composta pelas informações descritas na Tabela 4.

Tabela 4: Definição da classe Grupo.

Atributo	Tipo	Descrição	Domínio
descricao	String	Define qual é o agrupamento definido para o modelo ou àquele que o modelo deve ser inserido.	Texto livre de até 250 caracteres.

A Classe "InformacaoTecnica"

Esta classe é utilizada para armazenar as informações associadas aos "*engines*" de geração de código (MDD). Essas informações são importantes para se definir o cartucho de geração de código que deve ser usado para a geração dos serviços. Cada modelo criado pode seguir regras específicas de formação e embarcadas no próprio XMI gerado pela ferramenta CASE, onde cada projeto determina os valores marcados e estereótipos que são usados para gerar o serviço. Para cada iniciativa, deve-se oferecer a possibilidade de geração de código a partir de um cartucho MDD que entenda estas informações, permitindo maior flexibilidade e abrangência na capacidade de reuso dos modelos inseridos no SDS. É composta pelas informações descritas na Tabela 5.

Tabela 5: Definição da classe InformacaoTecnica.

Atributo	Tipo	Descrição	Domínio
descricao	String	Informações técnicas gerais, porém a mais utilizada é a de definição do cartucho MDD que deve ser usado pela ferramenta.	Texto livre de até 250 caracteres.

A Classe "Metrica"

Esta é classe utilizada para indicar fatores de qualidade do modelo assim como definido por [83], [84] e [85], a saber:

- Diretas: Custo, esforço (HH), Número de classes, Número de atributos e métodos por classe, Profundidade de herança (DIT), Número de Filhos (NOC), Média de métodos por classe (WMC), Acoplamento entre Objetos (CBO), Falta de coesão em métodos, Pontos de Função;
- Indiretas: Funcionalidade, Qualidade, Complexidade, Eficiência, Confiabilidade, Manutenibilidade.

É composta pelas informações descritas na Tabela 6.

Tabela 6: Definição da classe Metricas.

Atributo	Tipo	Descrição	Domínio
nome	String	Define o nome da métrica.	Texto livre de até 250 caracteres.
valorReferencia	Integer	Define o valor de referência para avaliação da qualidade do modelo.	Numérico, sem limite de valor.

A Classe "KeyWord"

Agrupa as palavras-chave para facilitação da busca do modelo. É composta pelas informações descritas na Tabela 7.

Tabela 7: Definição da classe KeyWord.

Atributo	Tipo	Descrição	Domínio
descricao	String	Cada objeto define as palavras que compõem o conjunto de palavras-chave associado ao modelo.	Texto livre de até 250 caracteres.

A Classe "Relevancia"

Define o contexto e quão relevante é o modelo para um determinado domínio de negócio. É composta pelas informações descritas na Tabela 8.

Tabela 8: Definição da classe Tipo.

Atributo	Tipo	Descrição	Domínio
descricao	String	Define a relevância do modelo no domínio de negócio.	1 – diretamente associado. 2 – indiretamente associado.

A Classe "Restricao"

Define restrições de uso deste modelo em um determinado domínio de negócio. É composta pelas informações descritas na Tabela 9.

Tabela 9: Definição da classe Tipo.

Atributo	Tipo	Descrição	Domínio
descricao	String	Define a relevância do modelo no domínio de negócio.	1 – diretamente associado. 2 – indiretamente associado.

3.2. O alinhamento ao MPS-BR

O ambiente, assim como descrito no Capítulo 2, busca alinhamento com o nível E do MPS-BR [8], pois determina as regras gerais por meio das atividades das GRU's, e foi escolhido pela sua abrangência no mercado de software Brasileiro [78]. Sendo assim, o SDS se alinha ao MPS-BR da seguinte forma:

1. Alinhamento ao GRU 1: definindo como artigo reutilizável os modelos conceituais e seus submodelos.
2. Alinhamento ao GRU 2: definindo como mecanismo de facilitação do reuso o serviço de desenvolvimento de software (SDS).
3. Alinhamento ao GRU 3: definindo como mecanismo de metadado os parâmetros que estabelecem os níveis de variação dos modelos dentro de um domínio de negócio.

4. Alinhamento ao GRU 4: definindo como mecanismo de controle de versão o próprio SDS, como descrito nos capítulos subsequentes.

5. Alinhamento ao GRU 5: definindo como os usuários são informados quando novos modelos são inseridos no serviço e sobre modificações nos modelos armazenados no SDS.

3.3. O serviço de desenvolvimento de software (SDS)

Como visto em [73], a reutilização, quando obtida por meio do reuso de modelos conceituais, depende fundamentalmente dos domínios de negócio que se deseja mapear, pois possuem variações e regras próprias que devem ser monitoradas ao longo do tempo.

Como abordado em [75], uma forma de se gerir estes quesitos recai sobre a utilização de métodos de análise de variabilidade dos elementos que compõem o modelo conceitual representativo do domínio de negócio, sendo este método regido por duas vertentes:

- Configuração de rotina: que determina um controle da variabilidade em termos de estruturas simples. Neste caso, tem-se a possibilidade de utilização de listas encadeadas, árvores ou outras que ajudem a definir uma estrutura semântica apropriada à identificação daquele domínio.

- Configuração criativa: que determina o uso de estruturas complexas para o controle da variabilidade. Neste caso, propõe-se utilizar estruturas como grafos e linguagens descritivas para este fim.

Conforme a abordagem apresentada em [89], métodos de análise de variabilidade ajudam a avaliar, dentro do domínio do negócio, as nuances, as faixas limítrofes de equivalência entre as necessidades/regras de negócio (em nível conceitual) e a solução de TI (em nível de implementação).

Deve-se, portanto, identificar os elementos de equivalência e diferenciação dos modelos conceituais de forma a maximizar a capacidade da abordagem adotada para a reutilização. Como resultado, busca-se criar grupamentos semânticos, associados ao domínio de negócio, adotando assim a noção de famílias de modelos, equivalente a famílias de produtos de software.

Com orientação do que foi apresentado e seguindo o modelo conceitual definido para o SDS, a Figura 18 apresenta o mapeamento de como a abordagem de reuso adotada opera mediante os procedimentos e funcionalidades idealizados e implementados no protótipo do SDS (capítulo 4). O conceito gira em torno da disponibilização de algumas funções essenciais no SDS, tais como:

1. Armazenamento de modelos conceituais para viabilização do processo de reutilização;

2. Reutilização de modelos conceituais, concebendo uma metodologia de agrupamento a partir de atributos definidos nesta tese, dando à organização capacidade de reuso;

3. Geração de código, que determina a capacidade de definir os serviços de TI que representarão os serviços de negócio dentro de um agrupamento lógico das estruturas de código geradas.

4. Integração destes agrupamentos lógicos, ou serviços, utilizando uma arquitetura de referência SOA.

Cada modelo conceitual necessita ser representado por um conjunto de informações, logo, inspirando-se na teoria de variabilidade, à cada grupo composto por modelos conceituais, agrega-se o conjunto de informações que caracterizam os domínios que a ele estão associados.

Definindo a categoria ou agrupamento de cada modelo, busca-se criar o conjunto de artefatos em código que representem estes modelos como serviços de TI, adotando-se a premissa de desenvolver um serviço de TI para cada modelo conceitual (1:1). Como trabalho futuro, vislumbra-se a possibilidade de desenvolver funcionalidade que permita a atendimento às seguintes regras: vários serviços de TI resultantes de um modelo conceitual (1:n); ou vários serviços de TI vários modelos conceituais (n:m); Logo, um ou mais grupos poderiam definir um sistema completo ou mesmo parte de um sistema maior, assim como destacado na Figura 18.

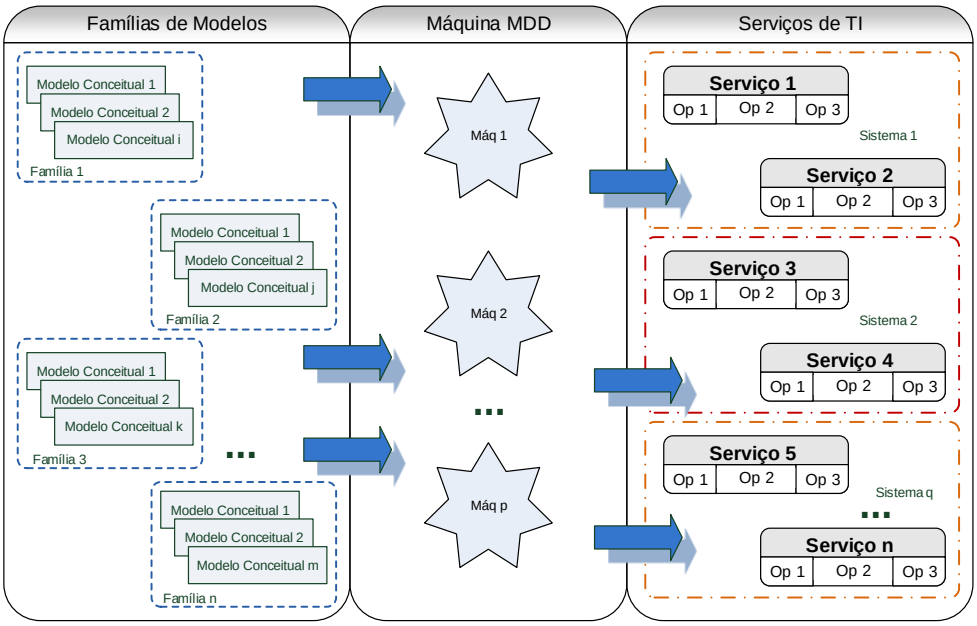

Figura 18: Arquitetura conceitual da solução de reuso de modelos orientado por famílias de software e apoiado por MDD.

Como consequência, a Figura 19 apresenta a proposta do conjunto de informações que devem ser coletadas no momento do armazenamento do modelo para as atividades de reuso no futuro. Inspirado pela abordagem de configuração de rotina para o controle de variabilidade [79], definiu-se um conjunto de parâmetros com a capacidade de organizar a informação referente aos modelos, levando-os ao seu agrupamento por semelhança.

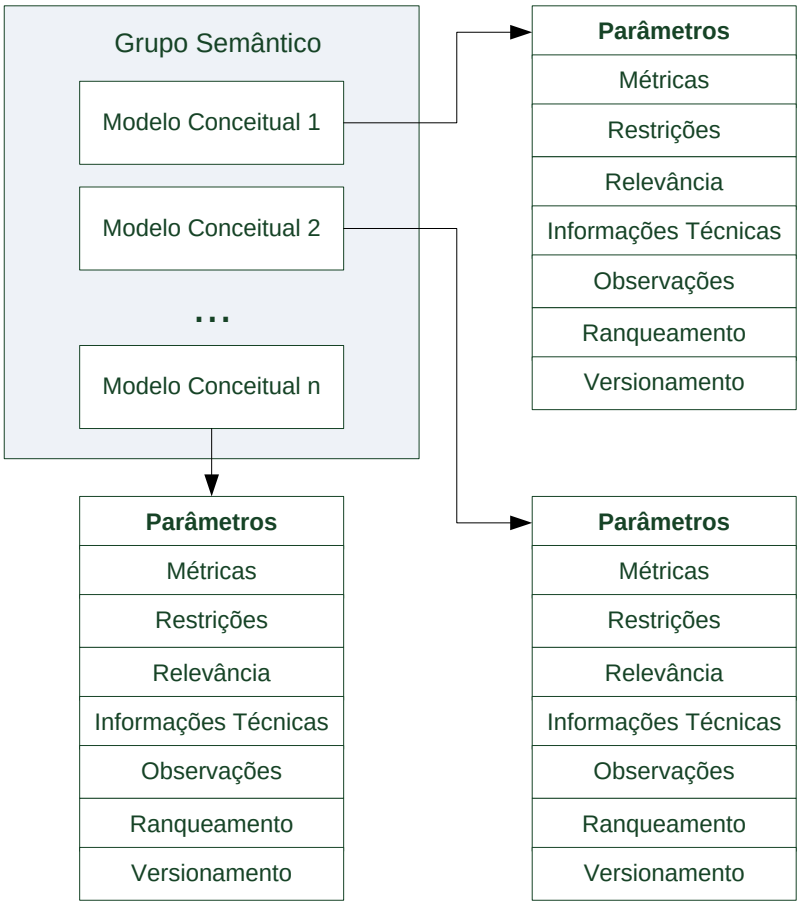

Figura 19: Parâmetros associados ao cadastramento de modelos para criação dos grupos semânticos.

Como abordado em [80] e estendido nesta tese, cada dado associado ao modelo pode ser analisado como um atributo que oferece ao usuário informações específicas no momento da escolha do modelo a ser (re)utilizado. A escolha do conjunto de parâmetros se deu a partir da identificação e combinação de uma série de atributos de controle e

qualidade, relevantes para a categorização de modelos, assim como apresentado em [11], [73], [79], [81], [82]:

- Métricas: expõem informações associadas a qualidade do modelo; como apresentadas em [83], [84] e [85].
    - Diretas: Custo, esforço (HH), Número de classes, Número de atributos e métodos por classe, Profundidade de herança (DIT), Número de Filhos (NOC), Média de métodos por classe (WMC), Acoplamento entre Objetos (CBO), Falta de coesão em métodos, Pontos de Função
    - Indiretas: Funcionalidade, Qualidade, Complexidade, Eficiência, Confiabilidade, Manutenibilidade.
- Restrições: informações a respeito das premissas e restrições consideradas quando da criação deste modelo. Tais restrições estão associadas aos requisitos funcionais utilizados para construção do modelo. Neste caso, há a necessidade de se incluir estes requisitos como forma de informar ao usuário as regras de negócio consideradas.
- Contexto/Relevância: define o contexto e quão relevante é o modelo para um determinado domínio de negócio. Especificado em dois níveis: (1) diretamente ao domínio de negócio, o que significa que este modelo foi criado para desenvolvimento de um sistema que dá suporte a um domínio de negócio diretamente; e (2) indiretamente ao domínio de negócio, que significa que o modelo foi criado para desenvolvimento de uma parte de um sistema associado a um domínio de negócio correlato.

- Informações técnicas: ferramenta com a qual foi desenvolvido o modelo, estruturas que foram utilizadas como entidades, valores marcados, diagramas que contém, etc.

- Ranqueamento: qual a prioridade de uma determinada versão do modelo em relação a outro pertencente ao mesmo grupo. Informação definida pelos usuários. Determina uma ordenação dentro de uma escala de 1 a 3, onde: 1 é não prioritário; 2 é indiferente; e 3 é prioritário.

- Versionamento: versão em que o artefato se encontra, que também definido pelo usuário, aderente ao GRU 4.

- Observações: outras informações sobre o modelo.

Buscando escalabilidade, a estrutura de agrupamento dos modelos conceituais pode ser planejada por meio da decomposição destes em submodelos, o que, por sua vez, pode representar regras e aplicações individuais e correlatas. Neste caso, o agrupamento pode ser refinado conforme apresentado na Figura 20.

A subdivisão promove um encadeamento de elementos próximos e semelhantes que viabilizam a correlação entre os modelos e seus sub-modelos. Isso implica em um levantamento determinístico das principais características do domínio que levem a um mapeamento adequado para a submissão do modelo à máquina de geração de código MDD, onde, finalmente, mediante o projeto arquitetural embarcado na máquina MDD, obtém-se o produto final de software.

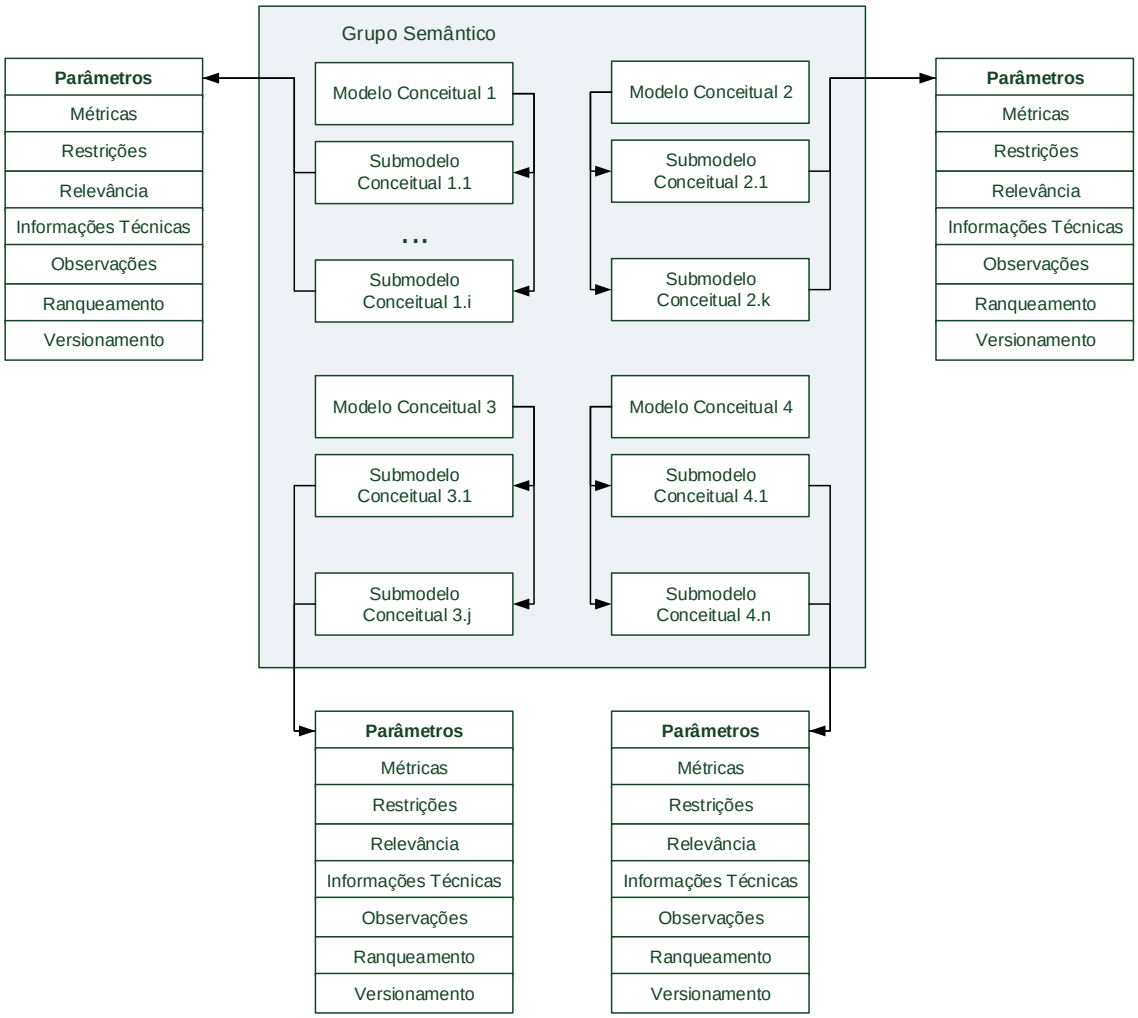

Figura 20: Refinamento do modelo de grupamento semântico para associação dos modelos conceituais (definido pelo autor).

Tão logo as regras de negócio, requisitos de software e os parâmetros de identificação estejam adequadamente inseridos, é possível anexar o(s) modelo(s) correspondente(s) ao seu grupo no SDS, deixando-o(s) apto(s) à reutilização.

3.4. Os critérios de integração dos serviços no SDS

Vistos como serviços de TI, os artefatos/produtos de software gerados a partir dos modelos podem ser agregados em uma solução maior ao embarcarem uma infraestrutura própria para troca de mensagens. Sendo assim, para garantir que os serviços gerados pelo SDS pudessem ser autocontidos e isolados, técnica e conceitualmente, evitando intervenções de customização após o procedimento de geração de código, criou-se uma estrutura de entrada e saída de requisições/mensagens, genérica, que permitisse o uso/reuso do serviço em diferentes sistemas de informação.

A cada geração de um Serviço de TI, a partir de um modelo conceitual, observa-se a estruturação lógica dos arquivos conforme a Figura 21. O conjunto de artefatos criados podem ser analisados estruturalmente em componentes interdependentes que são autocontidos e isolados. Dessa arquitetura, é possível observar que há a geração de um componente (*Servico de Integração*), responsável por integrar toda a estrutura do serviço representado pelo modelo conceitual no encaixe global dele no sistema. Este mecanismo deve ter a função de conectar o Serviço de TI ao "mundo externo", criando a interface de troca de mensagens entre outros serviços.

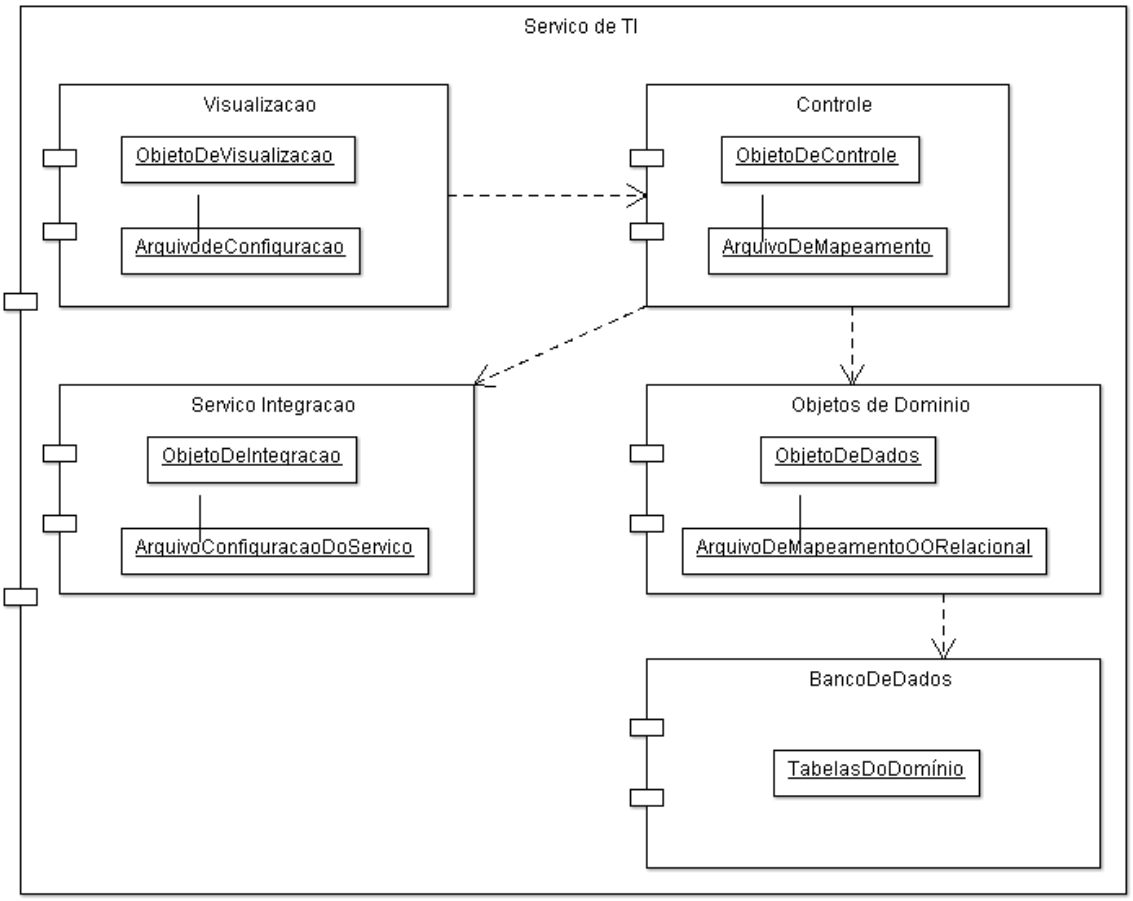

Figura 21: Arquitetura de um serviço gerado pelo SDS a partir de um modelo conceitual. Observa-se o componente denominado "Servico de Integracao" que é responsável pela integração do componente ao barramento de serviço.

Este Serviço de Integração foi planejado e criado como um Web Service, contendo os arquivos de configuração e lógica de encapsulamento, tais como o WSDL de representação das funções existentes no serviço de TI e elementos SOAP. Este serviço serve como um ponto único de troca de mensagens, espelhando o padrão de projeto Facade.

Como consequência, a Figura 22 destaca como todos os elementos se comunicam oferecendo uma visão geral dos relacionamentos existentes entre os componentes. Dos elementos apresentados, observa-se como ocorre a integração entre

os serviços, após sua instalação no servidor de aplicação. A conexão é feita pelo "Servico de Integracao" por meio de classes, objetos e arquivos de configuração que expõem o Serviço de TI ao mundo externo e ao receber uma mensagem, esta é repassada para o serviço, internamente.

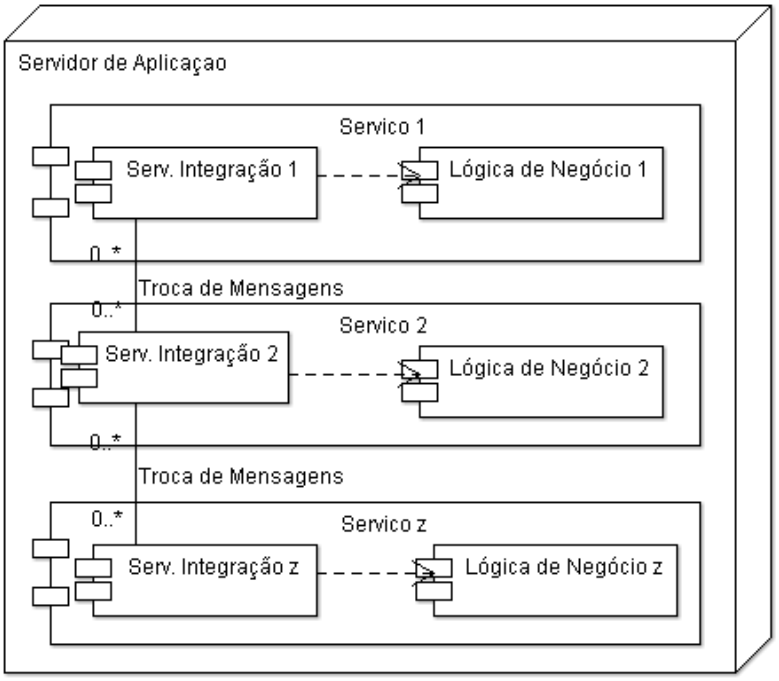

Figura 22: Integração dos Serviços de TI dentro do ambiente de execução no servidor de aplicação. A troca de mensagens é feita por intermédio do serviço de integração diretamente por acesso Web Service.

3.5. Operando como um Software as a Service

Com o objetivo de oferecer abrangência de uso do ambiente, considerou-se o paradigma de *"software as a servisse"* na implementação do protótipo. Um conceito importante para desenvolvimento desta tese foi a busca de criar uma abordagem de reuso que pudesse ser instalada como um serviço e acessada por usuários de todas as partes do mundo. É fato que o ambiente apresentado cabe, perfeitamente, em um ambiente local, podendo ser replicado e utilizado em escopo reduzido. No entanto, a

Figura 23 apresenta o conceito imaginado para a execução do SDS operando como um serviço. O SDS passa a ser visto como uma base central de armazenamento de modelos conceituais que deve ser acessado no momento da busca e da aquisição destes modelos a partir de critérios de identificação definidos pelos usuários.

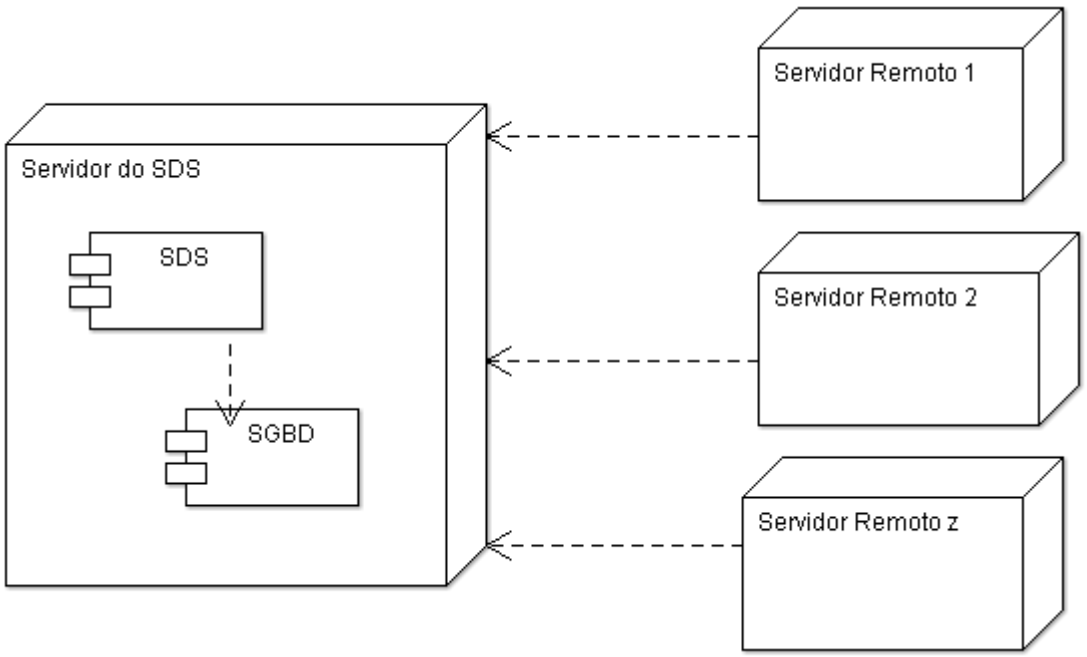

Figura 23: Conceito do SDS como um serviço. Cria-se uma base central de armazenamento e compartilhamento de modelos, oferecendo maior abrangência de uso do SDS e permitindo que um número maior de usuários possam cadastrar modelos conceituais que sejam desenvolvidos em seus projetos.

Conceituação Técnica

A conceituação técnica descrita a seguir ajuda a analisar um cenário de uso alinhado às atividades definidas pelo MPS-BR e oferece uma visão de entrega de serviços estruturada [2].

1) Definição do ativo: Modelo Conceitual.

Seja a necessidade de criar um sistema com um conjunto bem definido e finito de requisitos: `R = {r1, r2, r3, ..., rn}`. Cada `rn` definindo uma funcionalidade que deve ser implementada no software e que representa um conjunto de

regras de negócio do domínio em análise. Do conjunto de requisitos, busca ter como resultado o modelo conceitual apresentado na Figura 24.

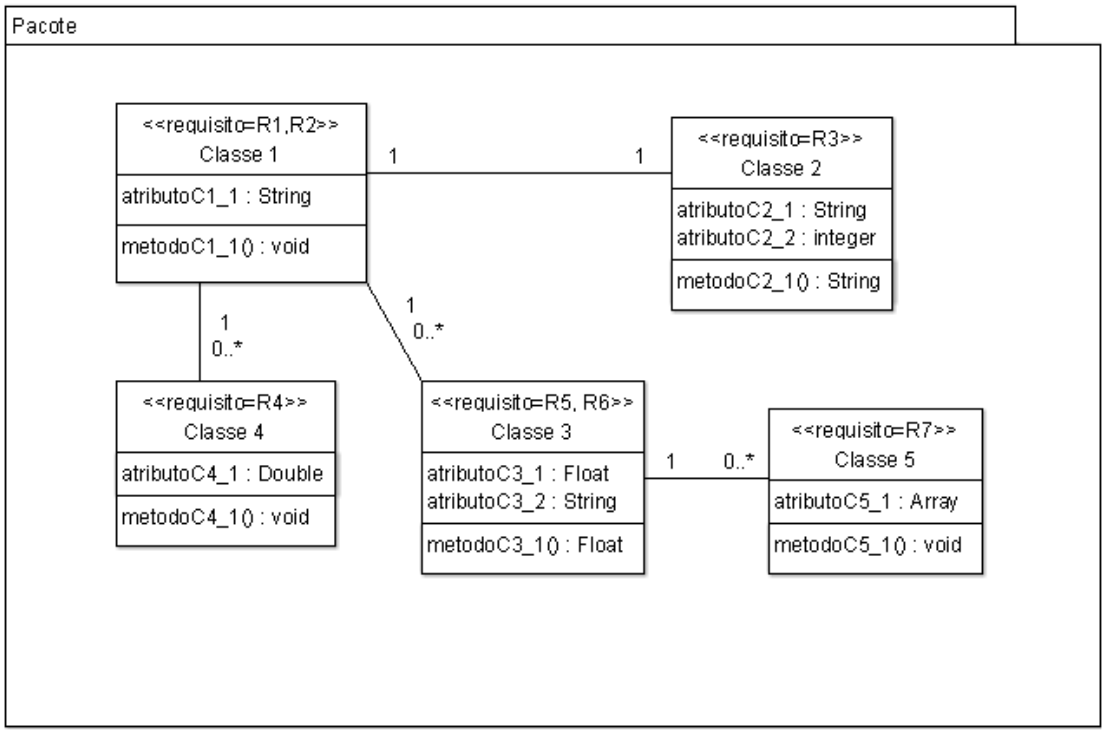

Figura 24: Modelo conceitual genérico utilizado para ilustrar a operacionalização do protótipo.

Este modelo, criado em uma ferramenta CASE [92], gera um documento XMI como o apresentado:

```
<?xml version="1.0" encoding="UTF-8" ?>
<XMI xmi.version="XX" xmlns:UML="org.omg.xmi.namespace.UML"
timestamp="Tue Mar 23 13:56:51 BRT 2010">
<XMI.header>...</XMI.header>
<XMI.content>
  <UML:Model xmi.id="00B06" name="Pacote" >
    <UML:Namespace.ownedElement>
      <UML:Class xmi.id="00CAE" name="Classe1" visibility="public">
       ...</UML:Class>
      <UML:Class xmi.id="00CAF" name="Classe2" visibility="public" >
```

```
      ...</UML:Class>
      <UML:Class xmi.id="00CB1" name="Classe3" visibility="public" >
      ...</UML:Class>
      <UML:Class xmi.id="00CB2" name="Classe4" visibility="public" >
      ...</UML:Class>
      <UML:Class xmi.id="00CB3" name="Classe5" visibility="public" >
      ...</UML:Class>
    </UML:Namespace.ownedElement>
    ...
  </UML:Model>
</XMI.content>
</XMI>
```

A construção do XMI deve seguir regras típicas de um desenvolvimento orientado a modelo onde boa parte das regras de negócio são descritas e marcadas por meio de elementos de extensão da UML como os *tagged values*, os *stereotypes* e os *constraints* [3] e [36].

2) Definição do mecanismo de reuso: SDS.

Visando a reutilização destes modelos, são inseridos os parâmetros para categorização e agrupamento dos modelos. Neste caso, são incorporadas informações referentes aos parâmetros de metadados definidos anteriormente, e então estas estruturas são armazenadas na base de dado do SDS.

3) Definição dos metadados: inclusão dos atributos para identificação do grupo do modelo.

Os modelos são tratados como elementos de entrada (input) do SDS, e são pareados por saída (output) em código. A cada modelo são agregadas todas as informações de categorização e agrupamento dentro da estrutura definida de metadados, própria do SDS, e então ele é publicado e disponibilizado no SDS conforme apresentado na Figura 17. Antes de ser inserido na base de dados, cria-se uma estrutura

em memória para organização e manipulação dessas informações em tempo de execução. O trecho de XML a seguir apresenta essa estrutura que, ao final do procedimento de manipulação do modelo em memória, é persistida a título de agilizar eventual recuperação do modelo por um usuário.

```
<sds>
    <servicos>
      <servico id=”S01” nome=”Servico_01”>
          <info>
            <projeto nome=”projeto”/>
            <data tipo=”inicio” valor=”01/01/2000”/>
            <data tipo=”fim” valor=”12/12/2000”/>
            <descricao>Descrição do modelo</descricao>
          </info>
          <modelo id=”M01” nome=”Conceitual_01” path=”Conceitual_01.xmi>
            <metadados>
                <parametro nome=”metrica_01” valor=”01”/>
                   ...
                <parametro nome=”metrica_n valor=”n”/>
            </metadados>
          </modelo>
          <links>
            <link id=”S02” nome”Servico_02”/>
            ...
          </links>
      </servico>
      ...
    </servicos>
</sds>
```

Uma vez armazenado no SDS, há a indexação do modelo pelos parâmetros de categorização.

4) Definição do mecanismo de versão: função de versionamento do SDS.

O versionamento é necessário para garantir integridade conceitual ao longo do tempo, sem que discrepâncias e inconsistências sejam inseridas em modelos armazenados no SDS. O procedimento inicia com a busca por um modelo que se alinhe

às necessidades do usuário. Após a identificação da instância do modelo que mais se adequa ao problema em particular, o usuário possui a opção de utilizá-lo em seu projeto "AS-IS", ou de refiná-lo. Em caso de alteração do modelo recuperado do SDS, sugere-se que ele seja submetido novamente à ferramenta, determinando assim uma nova instância ou nova versão do modelo, permitindo que seja (re)utilizado.

5) Definição do mecanismo de comunicação: função de notificação do SDS.

Toda e qualquer intervenção de um usuário em um determinado modelo é identificada e armazenada em formato de *log* no SDS. Procedimentos de inclusão, alteração e exclusão (lógica) são informados aos usuários que possuem interesse naquele modelo. O usuário pode definir no SDS quais os tipos de notificações que deseja receber e a Tabela 10 apresenta as que estão disponíveis no SDS.

Tabela 10: Definição dos tipos de notificação do SDS.

Notificação	Procedimento
Inserção	Quando um usuário insere um modelo o SDS envia um e-mail para a sua caixa postal cadastrada.
Alteração	Da mesma forma que na inserção, na ocorrência de uma alteração o usuário que cadastrou o modelo recebe a notificação em conjunto com aquele que efetuou a alteração. Modelos podem ser públicos ou privados. Modelos públicos possuem a característica de notificar qualquer intervenção a todos os usuários que interagem com ele. Alterações não geram novos registros no SDS.
Exclusão	Procedimento parecido com o de alteração. A exclusão é sempre lógica, permitindo a recuperação do modelo em caso de erro.
Versionamento	A cada nova versão de um modelo, todos os usuários relacionados são notificados. Versionamentos geram, necessariamente, novos registros no SDS.

4. O protótipo, aplicação e validação da abordagem

Para provar a viabilidade da hipótese definida na introdução, foi desenvolvido um protótipo contendo as principais funcionalidades destacadas ao longo da tese, sendo este disponibilizado como um serviço e permitindo o compartilhamento desejado entre os usuários. Em seguida, para validar os conceitos embarcados no protótipo efetuou-se uma prova de conceito efetuando o reuso de dois modelos genéricos em dois projetos de software.

4.1. Critérios de Validação

A partir da hipótese apresentada na introdução, fez-se necessário estabelecer os critérios de análise e validação do protótipo, de modo a apresentar evidências de que a abordagem a que se propões é viável. Para tanto, buscou-se apoio na teoria de Goal-Question-Metric [106], visando a formalização dos critérios que serão levados em consideração.

Dos objetivos

O objetivo, como destacado ao longo da tese, é o de validar a proposta de criação de um ambiente de reuso de modelos conceituais, alinhado ao processo de gerência de reuso do MPS-BR, assim como apresentado em seu nível E, e disponibilizá-lo como um Serviço de Desenvolvimento de Software.

Neste cenário, observam-se os seguintes objetivos:

i. Criação de um protótipo a partir das definições apresentadas nos capítulos anteriores.

ii. Validação do protótipo por meio da sua execução em cenários de uso.

Das questões a serem respondidas

Para cada objetivo, definiu-se as seguintes questões:

i. Uma vez criado o protótipo, como avaliar a sua capacidade de reutilizar modelos conceituais?

ii. Como calcular os percentuais de redução de esforço e investimento para cada modelo reutilizado?

Das métricas consideradas

Para cada pergunta, definiu-se as seguintes métricas:

i. Uma vez criado o protótipo, como avaliar a sua capacidade de reutilizar modelos conceituais?

   a. Métrica: Utilizar pontos de função para identificar o tamanho do código gerado pelos modelos.

ii. Como calcular os percentuais de redução de esforço e investimento para cada modelo reutilizado?

   a. Métrica: Obter cálculo da razão entre o valor total de pontos de função do sistema em construção e o total de pontos de função representados pelo código gerado pelo modelo.

4.2. O protótipo

O protótipo construído permite ao usuário armazenar os modelos conceituais, agrupando-os em categorias, e reutilizá-los, tanto como fonte de informação para a

evolução em uma nova versão de um modelo específico às suas necessidades ou na criação dos artefatos de software, representando serviços de TI.

No protótipo, destacam-se as funcionalidades:

Cadastro de usuários

Necessário para a identificação dos indivíduos que criaram o modelo, indicando informações que ajudem a mapear melhor o contexto no qual este modelo está inserido.

Cadastro de modelos

O cadastro de modelos, assim como o cadastro dos metadados, serve para armazenar e identificá-los como elementos reutilizáveis e que poderão ser utilizados pela ferramenta de MDD para a criação dos serviços de TI. Como apresentado na Figura 25, para incluir um modelo na base de dados, o usuário deve inserir todas as informações de metadados (variabilidade) para que seja possível indexá-lo na ferramenta de busca.

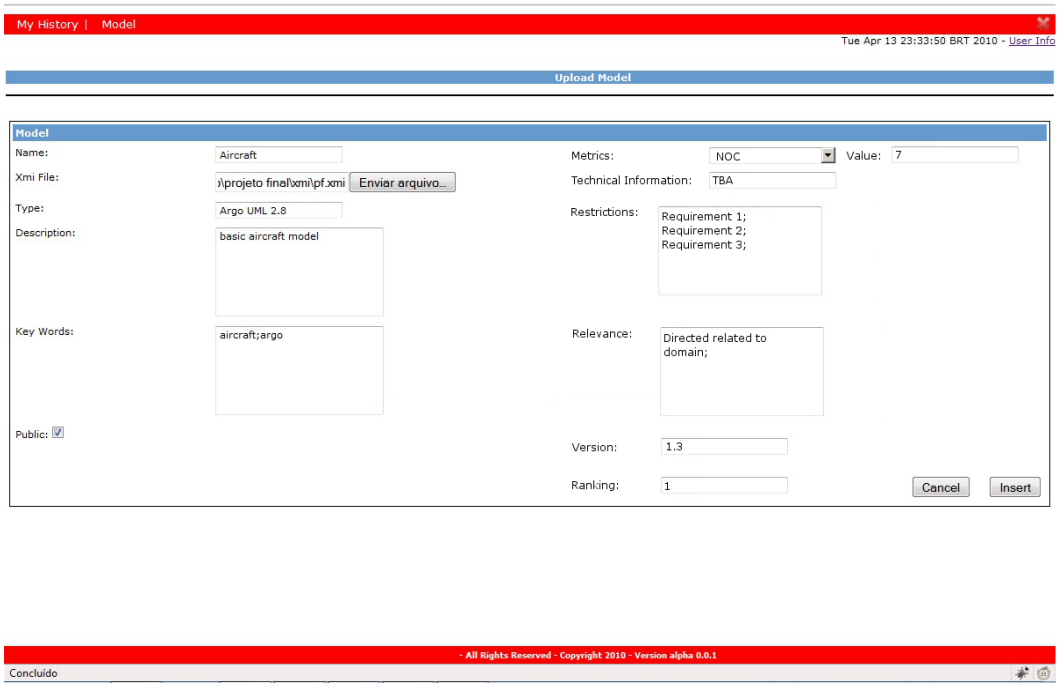

Figura 25: Upload do arquivo XMI do modelo conceitual. Neste caso as GRU's 2 e 3 são atendidas pois são definidas estratégias de inclusão, recuperação e manuseio dos modelos conceituais.

Geração de código (Serviço de TI).

De posse do arquivo XMI, o usuário se autentica no SDS e efetua o upload do arquivo ao mesmo tempo em que cadastra todas as informações de categorização e especificidades do negócio ao qual o modelo pertence.

Com estas etapas ultrapassadas, e com o modelo agora disponível no SDS, o usuário pode acessá-lo em uma página onde aparecem todas as versões e o seu histórico de modelos cadastrados. A Figura 26 destaca esta página e os modelos que foram armazenados pelo usuário.

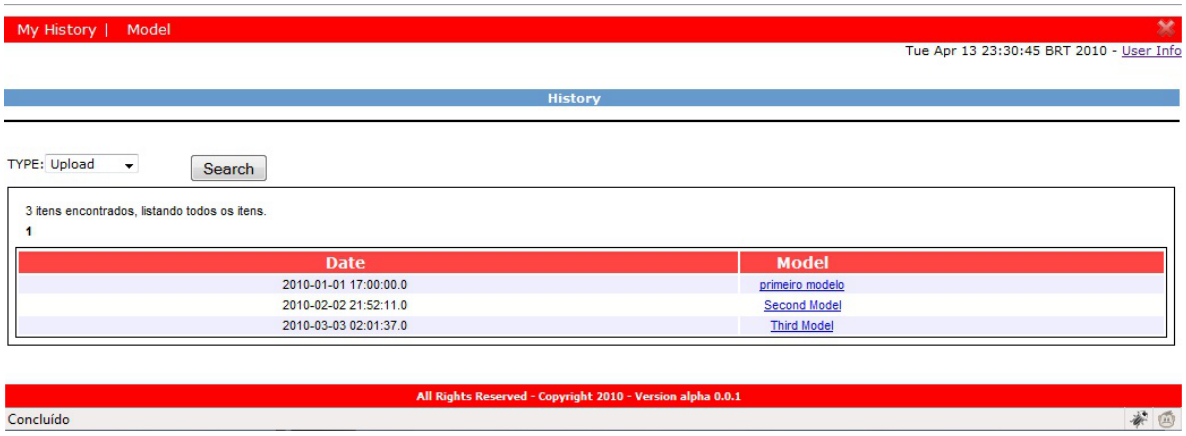

Figura 26: Versionamento dos modelos. Atendendo a GRU 4, que define o controle de versões e "feedback" da qualidade do modelo.

Em seguida, como destacado na Figura 27, após selecionar um dos modelos, ao usuário é facultada a possibilidade de efetuar o download do mesmo ou de gerar um pacote contendo todos os arquivos de código e configuração necessários para a customização e execução do Serviço de TI dentro de um servidor de aplicação.

O escopo definido para a instanciação da arquitetura de implementação do Serviço de TI, optou-se pela criação de componentes que observavam características de

uma aplicação WEB desenvolvida em Java, no caso optou-se pelo framework JEE da Oracle [93]. Sendo assim, definiu-se duas categorias elementares, a saber:

- Arquivos para customização leve: necessários para ajustes de visão e configuração do serviço de software em construção. Para cada classe encontrada no modelo conceitual os seguintes arquivos são gerados:
    - Componentes de visão: Java Server Pages (JSP) e Servlets [93];
    - Componentes de controle: Classes Actions do Struts [94];
    - Componentes de Serviço: Classes Service e arquivos de comunicação assíncrona Ajax (DWR) [96]
    - Componentes de Mapeamento Orientação a Objeto – Relacional: Classes DataObject para acesso ao banco de dados e arquivos de mapeamento xml para o Hibernate [95];
    - Componentes do Serviço de Integração: Web Service e os diversos arquivos de configuração para mapeamento com o servidor de aplicação e com o barramento de serviços (JBoss) [97].

A Figura 27 apresenta as opções de geração do pacote de código com o seu consequente download.

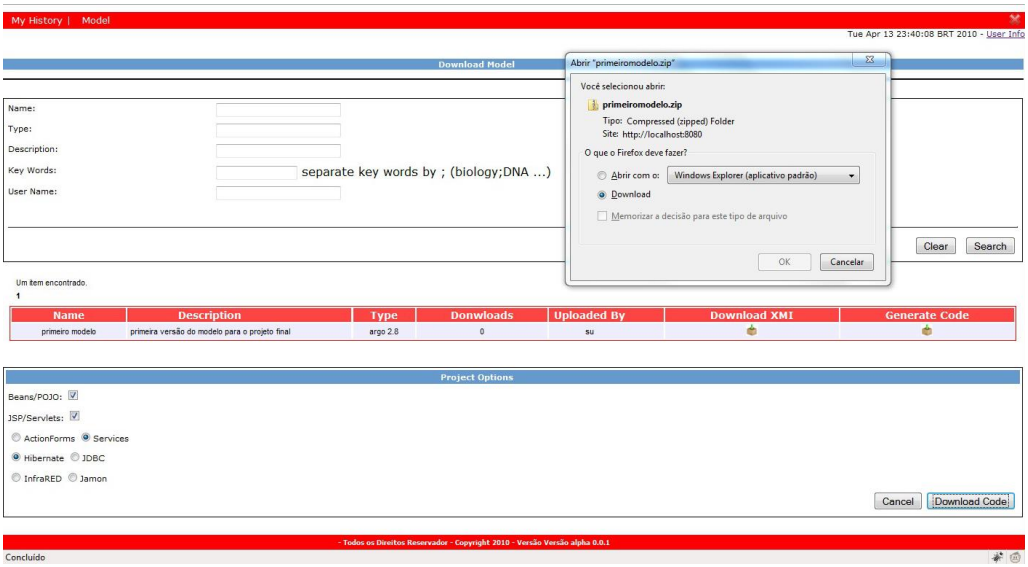

Figura 27: Download da aplicação gerada. Atende a GRU 4, recuperação e utilização dos artefatos.

Mais explicitamente, o procedimento de geração dos Serviços de TI foi desenvolvido visando a criação de aplicações Web dentro de um modelo arquitetural definido assim como apresentado na Figura 28. A escolha desta instanciação da arquitetura se deu pela abrangência no uso desses componentes pela comunidade de desenvolvedores JEE e pelas pesquisas realizadas pelo autor na área de desenvolvimento orientado a modelos [98].

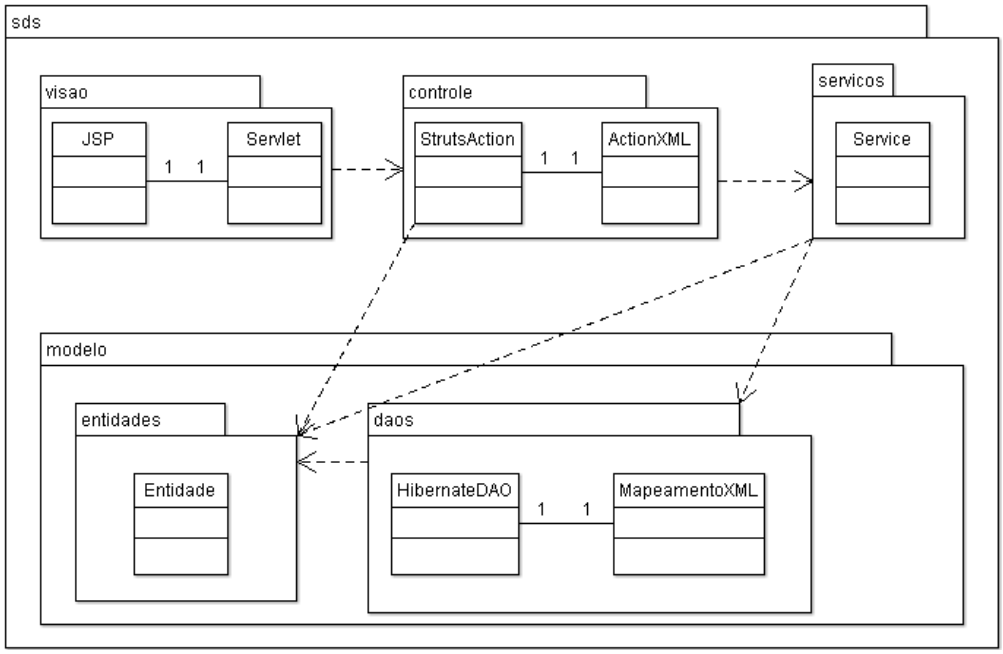

Figura 28: A arquitetura dos pacotes de código fonte gerados pela aplicação é padronizada e estabelecida a partir da interação entre os componentes de visão, controle e modelo, assim como estabelecidos no padrão M-V-C.

Versionamento e histórico de modelos.

Funcionalidade necessária para garantia do alinhamento às GRUs do MPS-BR, e como visto na Figura 29, o versionamento é feito de maneira automática e permite a identificação dos modelos mediante data de entrada e os parâmetros de variabilidade.

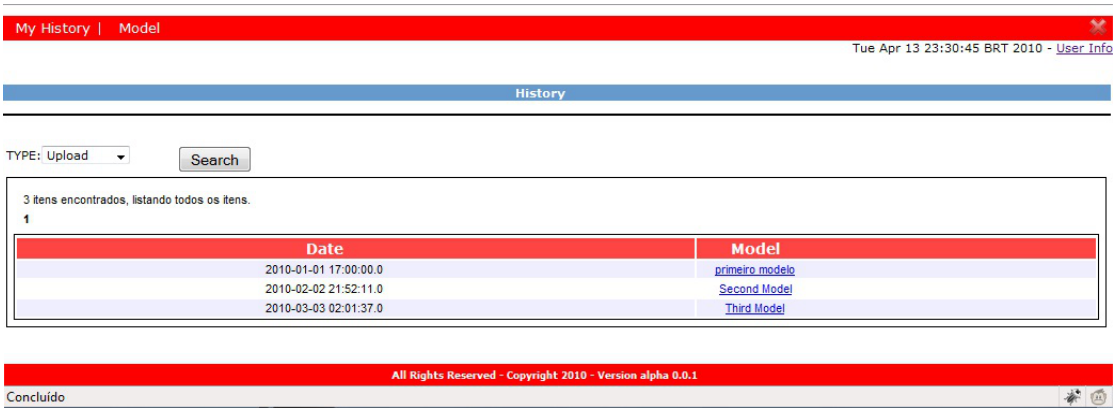

Figura 29: Assim como na Figura 26, o controle de versão é efetuado no SDS. Essa abordagem atende a GRU 4, que define o controle de versões e "feedback" da qualidade do modelo.

4.3. Prova de Conceito.

Assim como abordado em [99], esta prova de conceito foi elaborada contendo três fases: planejamento, execução e análise dos resultados. A fase 1, definida como *Planejamento*, serviu para identificar os modelos conceituais e os projetos que permitiram a divulgação e utilização dos modelos candidatos ao reuso. A fase 2, definida como execução, serviu para verificar se as funcionalidades criadas no SDS sustentam a hipótese definida no capítulo de introdução. Finalmente, a fase 3, análise dos resultados, apresenta se os artefatos de software reutilizados resultam em benefícios relacionados ao tempo e o custo dos projetos.

4.4. Fase 1: Planejamento – Definido os projetos e os modelos candidatos ao reuso.

Com o objetivo de testar a abordagem a partir de dois projetos desenvolvidos em domínios de negócios distintos, os tópicos que seguem descrevem o Projetos do Registro Integrado de Empresa (REGIN) e o Sistema para o Apoio Logístico Integrado da Marinha (SISALI).

4.4.1. Projeto 1: O Registro Integrado de Empresas

O REGIN [100] é um sistema que foi desenvolvido para facilitar e desburocratizar o processo de abertura de empresas no Brasil. Como apresentado em [101], o REGIN propõe uma centralização de responsabilidade (de serviços) em um órgão específico de governo, tal como a prefeitura ou a junta comercial. Quando instalado (Figura 30), ele tem como mérito unificar o pedido de abertura de empresas para que o empresário precise apenas ir até um único posto de atendimento e entrar com todos os dados necessários (pessoais e da empresa) para dar início ao processo.

O sistema, então, identifica os dados de cada órgão, dividindo-os em mensagens distintas, e os envia por um conjunto de serviços instalados em Web Services. Todos os órgãos têm contrapartes onde seus Web Services recebem as mensagens; e um sistema acoplado a cada um deles permite a validação dos dados.

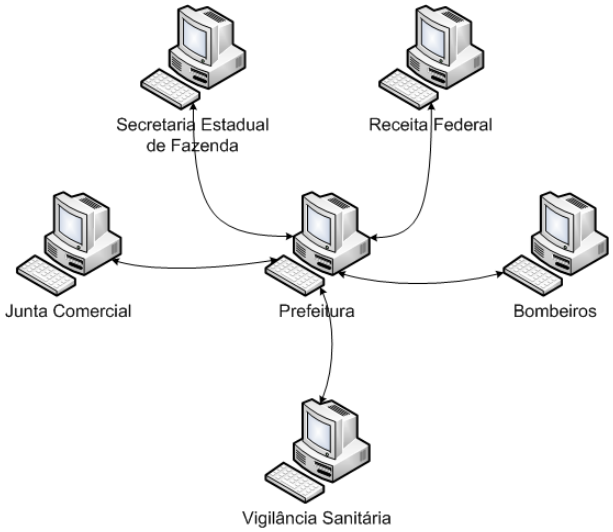

Figura 30: A partir de um único local – a Prefeitura –, o contribuinte poderá abrir sua empresa em um prazo muito inferior aos 152 dias em média.

Da Arquitetura do REGIN

O sistema REGIN foi desenvolvido fora do ambiente do SDS, possuindo dois módulos: (1) o módulo centralizador, desenvolvido em .Net; e (2) o módulo remoto desenvolvido em Java/JEE. Como prática de programação, a organização estrutural e o modelo de integração (comunicação) entre as três camadas M-V-C estão definidos como na Figura 31.

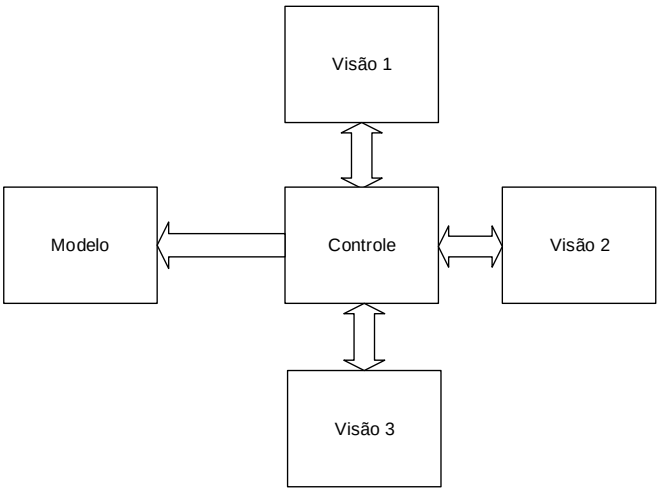

Figura 31: Abordagem MVC para organização dos componentes do sistema em termos de separação de responsabilidades.

Concomitantemente, os elementos utilizados para a construção do REGIN pode ser apreciado na Figura 32, onde é possível identificar que a organização dos componentes é semelhante a definida no SDS.

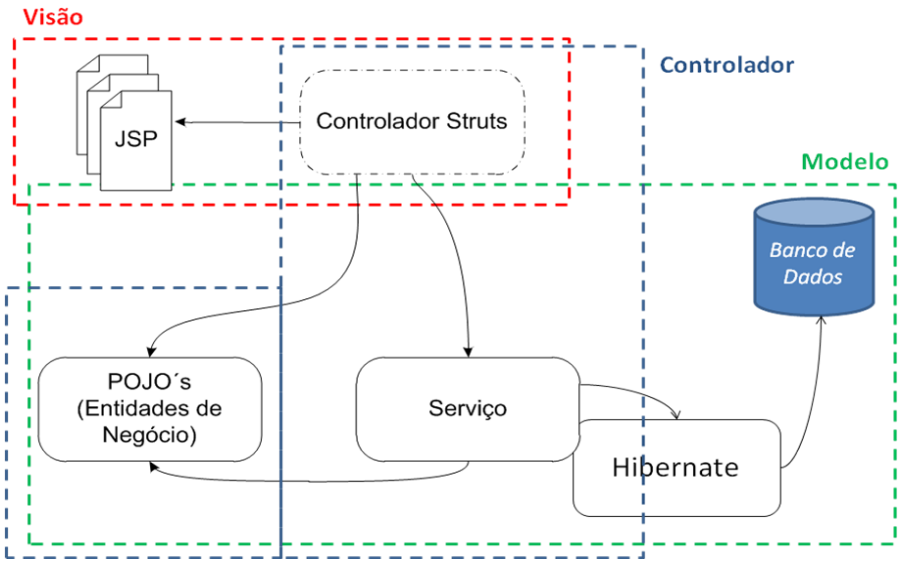

Figura 32: Definição da arquitetura do REGIN em termos dos frameworks e componentes utilizados em sua construção.

O Planejamento para o Reuso do Modelo de Autenticação e Autorização

Durante esta fase de planejamento foi possível identificar duas funcionalidades transversais no sistema que continham grau de generalização adequada para esta validação do SDS: a de autenticação e a de relatórios.

A funcionalidade de autenticação adotada é bastante comum e utilizada com frequência em aplicações WEB. Há uma página de identificação, que possui dois campos, um de usuário e outro de senha. Esta senha é validada diretamente no servidor de banco de dados, onde fica armazenada criptografada. No momento da validação, a

senha que é inserida pelo usuário é criptografada e comparada à que está no banco. Ambos os campos possuem 11 caracteres de tamanho.

Cada ação efetuada dentro do sistema está associada a um evento e estes eventos estão agrupados em papéis. Cada papel determina um perfil que é dado a um ou mais usuários de modo que consigam acessar o sistema.

Internamente, o sistema possui dois elementos de controle:

1 – Acesso às funcionalidades, onde aos menus e itens de menus são atribuídos eventos.

2 – As ações internas (mapeadas pelos Actions), aos quais também são associados eventos.

Todos os eventos são criptografados usando chave de vida curta, definida por usuário e por sessão da aplicação. Sempre que uma requisição é efetuada, um filtro avalia se aquele evento é válido para aquele usuário, usando a chave de vida curta. Caso o evento seja válido, a ação prossegue, do contrário a requisição é bloqueada e redirecionada para uma página de erro.

Para esta funcionalidade foi desenvolvido um modelo conceitual que é detalhado na Figura 33.

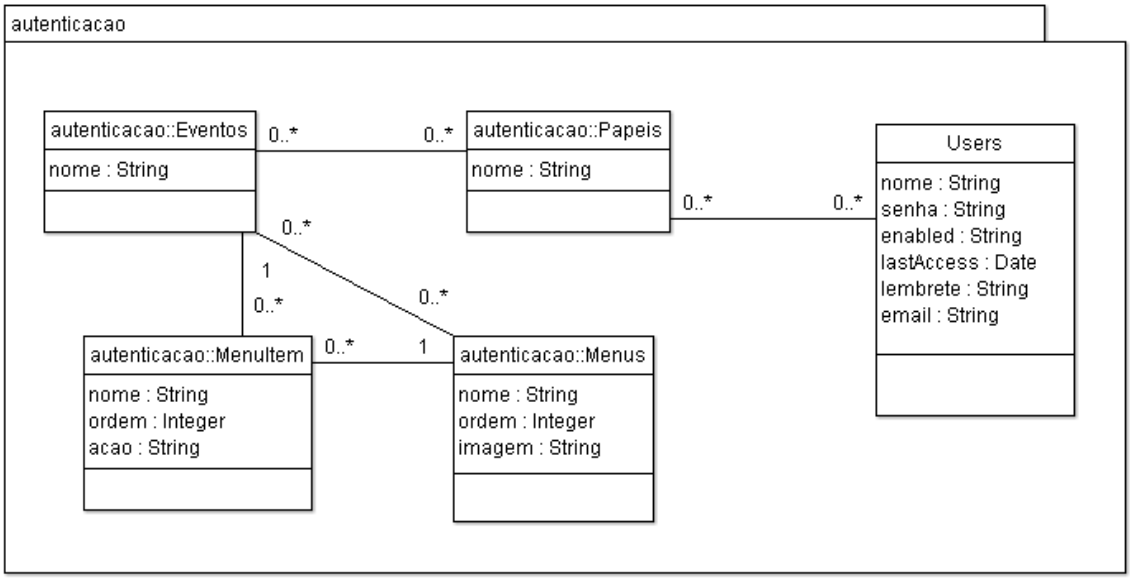

Figura 33: Modelo conceitual do controle de acesso às funcionalidades do sistema.

As Tabelas de 11 a 15 detalham as classes do modelo.

Tabela 11: Definição da classe Papeis.

Atributo	Tipo	Descrição	Domínio
nome	String	Define o um papel a ser atribuído a um usuário.	Texto livre de até 250 caracteres.

Tabela 12: Definição da classe Eventos.

Atributo	Tipo	Descrição	Domínio
nome	String	Define o um evento a ser tratando no sistema.	Texto livre de até 250 caracteres.

Tabela 13: Definição da classe Menu.

Atributo	Tipo	Descrição	Domínio
nome	String	Define o nome do menu a ser apresentado no sistema.	Texto livre de até 250 caracteres.
ordem	Integer	Define a ordem de apresentação deste menu dentro do sistema.	Valor numérico sem limite.
imagem	String	Define uma imagem a ser apresentada em conjunto com o	Texto livre de até 250 caracteres.

		nome no sistema.	

Tabela 14: Definição da classe MenuItem.

Atributo	Tipo	Descrição	Domínio
nome	String	Define o nome do item de menu que será apresentado no sistema.	Texto livre de até 250 caracteres.
ordem	Integer	Define a ordem deste item dentro do menu.	Valor numérico sem limite.
acao	String	Define a ação a ser executada quando o usuário clica este item.	Texto livre de até 1000 caracteres.

Tabela 15: Definição da classe Users.

Atributo	Tipo	Descrição	Domínio
nome	String	Define o nome utilizado pelo usuário para acessar o sistema.	Texto livre de até 250 caracteres.
senha	String	Define a senha usada pelo usuário para entrar no sistema	Texto livre de até 11 caracteres.
enable	String	Flag para identificar se o usuário está ativo no sistema	{1,0}
lastAccess	Date	Data do último acesso do usuário no sistema	Data e hora do acesso.
lembrete	String	Lembrete da senha cadastrada pelo usuário.	Texto livre de até 250 caracteres.
email	String	Endereço eletrônico cadastrado pelo usuário.	Texto livre de até 250 caracteres.

O Planejamento para o Reuso do Modelo de Relatórios

A outra funcionalidade criada para o projeto REGIN é a de relatórios. Da mesma forma como o módulo de autenticação, a de relatórios foi pensada de forma isolada e modelada conforme apresentado na Figura 34. O modelo destaca a associação de entidades que representam um componente de geração de "relatórios genéricos" e que foi desenvolvido de maneira a diminuir o tempo de desenvolvimento de relatórios simples que eram comuns no sistema.

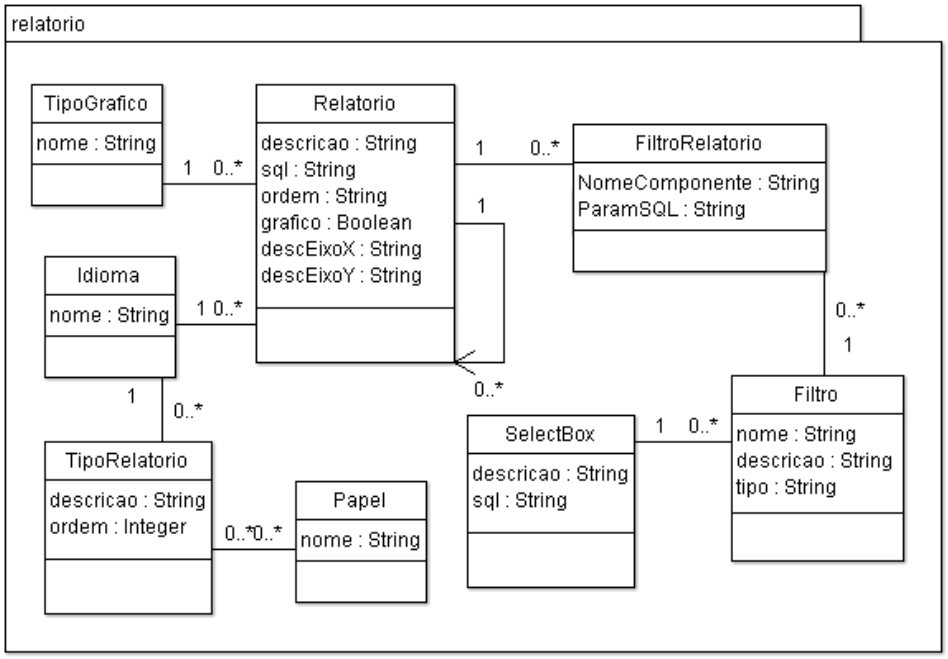

Figura 34: Modelo conceitual do módulo de relatórios do sistema REGIN.

As Tabelas de 16 a 22 detalham as classes do modelo.

Tabela 16: Definição da classe Relatório.

Atributo	Tipo	Descrição	Domínio
descricao	String	Define um texto descritivo para este relatório.	Texto livre de até 250 caracteres.
sql	String	Query criada em linguagem SQL que é executada pelo sistema para a geração do relatório.	Texto livre de até 1000 caracteres.
ordem	String	Define a ordem deste item dentro do menu.	Valor numérico sem limite.
grafico	Boolean	Define se este relatório deve renderizar um gráfico.	{true, false}
descEixoY	String	Quando o atributo grafico = true, este campo deve conter a descrição que será apresentada no eixo Y.	Texto livre de até 250 caracteres.
descEixoX	String	Quando o atributo grafico = true, este campo deve conter a descrição que será apresentada no eixo X.	Texto livre de até 250 caracteres.

Tabela 17: Definição da classe TipoGrafico.

Atributo	Tipo	Descrição	Domínio
nome	String	Define os tipos de gráficos que este relatório pode renderizar.	Texto livre de até 250 caracteres.

Tabela 18: Definição da classe Idioma.

Atributo	Tipo	Descrição	Domínio
nome	String	Define o idioma que deve ser usado na criação dos gráficos.	Texto livre de até 250 caracteres.

Tabela 19: Definição da classe TipoRelatorio.

Atributo	Tipo	Descrição	Domínio
descricao	String	Define grupos de relatórios.	Texto livre de até 250 caracteres.
ordem	Integer	Define a ordem deste item dentro do menu.	Valor numérico sem limite.

Tabela 20: Definição da classe Papel.

Atributo	Tipo	Descrição	Domínio
nome	String	Define o papel pode visualizar este relatório.	Texto livre de até 250 caracteres.

Tabela 21: Definição da classe Filtro.

Atributo	Tipo	Descrição	Domínio
nome	String	Define o nome utilizado pelo filtro e apresentado junto ao relatório. Compõe a claúsula "Where" da query SQL.	Texto livre de até 250 caracteres.
descrição	String	Detalha a funcionalidade deste filtro	Texto livre de até 250 caracteres.
tipo	String	Cria grupos de filtros	Texto livre de até 250 caracteres.

Tabela 22: Definição da classe SelectBox.

Atributo	Tipo	Descrição	Domínio
nome	String	Esta classe especializada define um elemento gráfico denominado ComboBox.	Texto livre de até 250 caracteres.
sql	String	Define a query SQL que é utilizada para compor os valores apresentados na ComboBox.	Texto livre de até 1000 caracteres.

Do Desenvolvimento do REGIN

Os números apurados ao final do projeto REGIN podem ser vistos na Tabela 23. Nela são apresentados os artefatos que foram criados ao longo do projeto e o que cada um dos modelos representa do total do projeto.

Tabela 23: Valores apurados do projeto destacando os módulos de relatório e controle de acesso.

Código Fonte REGIN	Total	Autenticação	%	Relatórios	%
Módulos	7	1	14,28	1	14,28
Arquivos Java	643	39	6,06	21	3,26
Arquivos de Visão JSP	84	6	7,14	5	5,95
Arquivos de Configuração (XML)	3	1	33,33	1	33,33
Arquivos JavaScript	5	1	20,00	1	20,00
Arquivos de Propriedades	4	1	25,00	1	25,00
Arquivos TLD	11	1	9,09	1	9,09
Arquivos CSS	2	1	50,00	1	50,00
Scripts por módulo do REGIN	3	1	33,33	1	33,00
Total de Artefatos de Software	762	52	6,84	33	4,33

Em termos de números absolutos dos artefatos, os dois módulos (autenticação e relatórios) representam um percentual de 11,17% de todos os artefatos do projeto. Em termos de tamanho, o número de pontos de função de cada módulo no sistema e o conjunto de horas gasto para o seu desenvolvimento podem ser vistos na Tabela 24.

Tabela 24: Total de Pontos de Função por Modelo.

Modelo	Pontos de Função	Horas
Autenticação	47	307,5
Relatório	197	1477,5

4.4.2. Projeto 2: O Sistema de Informação para o Apoio Logístico Integrado da Marinha – SISALI

Seguindo ordem expedida pelo Comandante da Marinha [102], o Núcleo de Apoio Logístico Integrado iniciou o desenvolvimento de um sistema de informação denominado: Sistema de Informação para o Apoio Logístico Integrado (SISALI). Este sistema deveria ser capaz de subsidiar os níveis da cadeia hierárquica naval com indicadores de desempenho e que deveriam ser utilizados em atividades de tomadas de decisão relativas a manutenção e obtenção.

Abrangência do SISALI

O sistema visava atender às seguintes demandas:

1. Integração dos bancos de dados corporativos da Marinha.
2. Módulo de Controle das Organizações Militares Prestadoras de Serviço Industriais – OMPS-I: atender ao processo desde o recebimento do Pedido de Serviço (PS) até a sua conclusão.
3. Módulo de Serviço de Manutenção Planejada - SMP: controle informatizado da execução do Sistema de Manutenção Planejada (SMP)
4. Módulo de Relatório de Análise ao Apoio Logístico - RAAL: controle dos registros de dados de análise do apoio logístico;
5. Módulo de documentação técnica, que deverá conter as normas técnicas, manuais de fabricantes de todos os meios/sistemas/equipamentos utilizados pela Marinha.

As principais atividades associadas as funções do SISALI

Em geral, as atividades gerais do processo de negócio implantado no SISALI pode ser visto na Figura 35. Nela, é possível verificar todas as etapas existentes no processo de manutenção de meios navais da Marinha e quais são os módulos que deveriam ser desenvolvidos ao longo do projeto.

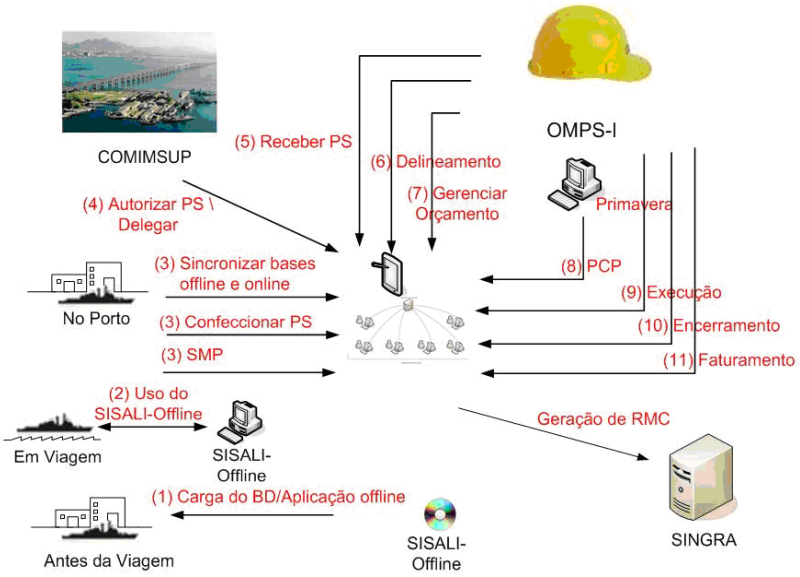

Figura 35: Atividades gerais do apoio logístico.

As atividades de desenvolvimento

A modelagem do sistema seguiu uma abordagem incremental e dividida em fases, onde foram realizadas entrevistas com órgãos dos diversos níveis hierárquicos, seguindo-se à modelagem da Arquitetura de Informações (AI) e contemplando a integração dos bancos de dados corporativos e do SISALI. As Figuras 36 e 37 apresentam as principais atividades que foram contempladas nos processos de negócio que o SISALI deveria atender.

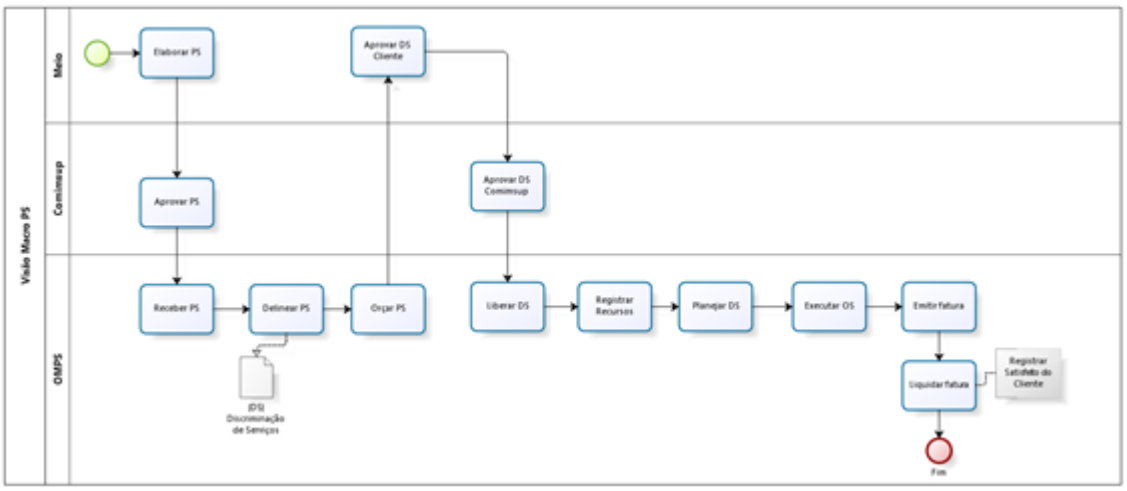

Figura 36: Processo de negócio contendo as atividades das OMPS-I.

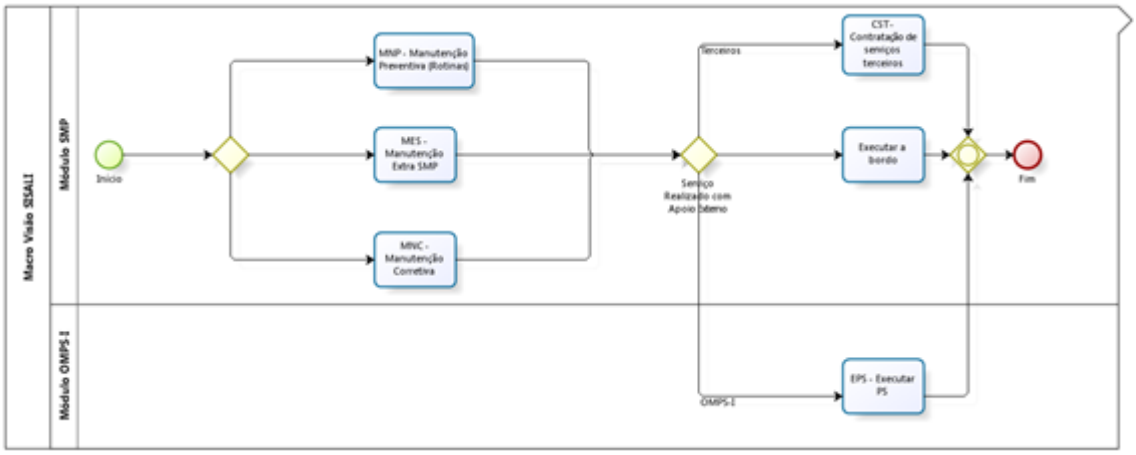

Figura 37: Processo de negócio contendo as atividades de SMP.

O Projeto Piloto

O projeto piloto designado pela Coordenadoria de Manutenção de Meios (CMM), determinou a implantação do sistema no Centro de Armas da Marinha e em um navio do tipo Fragata da classe Niterói.

As metas estabelecidas para a arquitetura do projeto SISALI foram definidas como:

1. A arquitetura foi projetada em tecnologia comum de mercado, buscando diminuir os custos associados com a mão-de-obra de programação.

2. A arquitetura foi projetada em plataforma WEB, facilitando a sua portabilidade, abrangência, manutenção e evolução, novamente, evitando a necessidade de retrabalho, e como consequência, minimizando custos desnecessários à Marinha.

3. A arquitetura foi projetada para viabilizar uma organização lógica de fácil entendimento, estruturando o sistema em pacotes e pastas de modo a facilitar a imersão do desenvolvedor no código.

O projeto conceitual da arquitetura buscou unir os componentes do JEE, Struts e do Hibernate, em uma abordagem de fácil programação, teste, implantação e manutenção.

Do Desenvolvimento do SISALI

Uma vez estabelecidas as regras para o desenvolvimento do sistema SISALI, e mediante o modelo conceitual definido, deu-se início ao desenvolvimento do sistema. Conforme apresentado na Tabela 25, o SISALI deveria conter o seguinte conjunto de funcionalidades.

Tabela 25: Conjunto de requisitos do SISALI.

ID	Funcionalidade	Pontos de Função
1	Elaborar Pedido de Serviço Preventivo	41
2	Elaborar Pedido de Serviço Corretivo	82
3	Aprovar PS pelo Chefe de Departamento	35
4	Aprovar PS pelo Comandante	35
5	Aprovar PS pelo Comimsup	35
6	Receber PS	41
7	Aprovar PS pelo Gerente	23
8	Delinear PS	59
9	Orçar PS	59
10	Aprovar Orçamento pelo Gerente	59
11	Aprovar DS pelo Cliente	59
12	Aprovar DS pelo COMIMSUP	59
13	Aprovar DS pelo Gerente	59
14	Registrar no SIAFI	17
15	Liberar DS Sem Recursos	46

16	Planejar PS	43
17	Abrir OS	35
18	Apropriar HH	40
19	Indicar Gastos	37
20	Aprovar Indicação de Gastos	35
21	Faturar Gastos	41
22	Aprovar Fatura pelo Cliente	34
23	Liquidar Fatura	34
24	Interromper OS	17
25	Reabrir OS	23
26	Relatório de Disponibilidade do Equipamento	23
27	Relatório de Disponibilidade do Meio	34
28	Relatório de Tempo Médio de Falhas	23
29	Relatórios de Pedidos de Serviço por status	35
30	Relatório de Horas Apropriadas	41
31	Relatório de Acompanhamento de Serviços	41
32	Controle de Autenticação	59
	Total de Pontos de Função	**1304**

*** Cálculos de Pontos de Função realizados a partir das regras estabelecidas pelo IFPUG. Fonte: www.ifpug.org**

Com um total de 1304 pontos de função desenvolvidos ao longo de 4 anos de desenvolvimento, a Tabela 26 apresenta o número total de artefatos criados.

Tabela 26: Números de artefatos de código implementados no sistema.

Artefatos de Código Fonte	Quantidade
Módulos	5
Arquivos Java	1319
Arquivos de Visão JSP	635
Arquivos de Configuração (XML)	496
Arquivos JavaScript	488
Arquivos de Propriedades	4
Arquivos TLD	3
Bibliotecas utilizadas	47
Arquivos CSS	29
Imagens (GIF)	398
Imagens (JPG)	73
Imagens (PNG)	142
Tabelas para os dados do SISALI	200
Tabelas com os dados do Singra	44
Total de Artefatos de Software	**3878***

*** Excluindo-se os módulos que não são artefatos.**

4.5. Fase 2: Execução – o reuso no SDS.

Nesta fase busca-se apresentar o cenário de uso e execução da abordagem por meio do reuso dos modelos de autenticação e relatórios. Nestes termos, a avaliação empírica do processo de reuso teve início no momento da inclusão de ambos os modelos do sistema REGIN no SDS. Ao incluí-los na base de dados do SDS e permitir os seus compartilhamentos dentro do ambiente, permitiu-se validar as premissas aqui apresentadas, principalmente quando observados os critérios que viabilizam a reutilização em outros projetos.

A estruturação dos modelos dentro do SDS

Tendo ambos os modelos candidatos ao reuso, a estrutura interna ao SDS pode ser analisada conforme o XML de configuração apresentado a seguir. Nele é possível verificar como os modelos estão armazenados e como foram utilizados na criação dos serviços de TI.

```
<sds>
    <servicos>
      <servico id=”Auth_Service_01” nome=”Authentication Service”>
          <info>
            <projeto nome=”REGIN”/>
            <data tipo=”inicio” valor=”23/10/2010”/>
            <data tipo=”fim” valor=”12/11/2012”/>
            <descricao>
                  Modelo de autenticação do projeto REGIN
            </descricao>
          </info>
          <modelo id=”Auth_Model_01”
                  nome=”Modelo_Autenticacao” path=”modelo_autenticacao.xmi>
            <metadados>
                <parametro nome=”versao” valor=”02”/>
                <parametro nome=”restricao” valor=”R01,R02,R03”/>
                <parametro nome”relevancia” valor=”direta ao domínio”/>
                <parametro nome=”ranking” valor=”01”/>
                <parametro nome=visibilidade” valor=”publico”/>
```

```
                <parametro nome=”feramenta_case” valor=”ArgoUML”/>
            </metadados>
        </modelo>
    </servico>
    <servico id=”Report_Service_01” nome=”Report Service”>
        <info>
            <projeto nome=”REGIN”/>
            <data tipo=”inicio” valor=”23/10/2010”/>
            <data tipo=”fim” valor=”12/11/2012”/>
            <descricao>
                  Modelo de relatórios do projeto REGIN
            </descricao>
        </info>
        <modelo id=”Req_Model_01”
                nome=”Modelo_Relatorio” path=”modelo_relatorio.xmi>
            <metadados>
                <parametro nome=”versao” valor=”01”/>
                <parametro nome=”restricao” valor=”R21,R22”/>
                <parametro nome”relevancia” valor=”direta ao domínio”/>
                <parametro nome=”ranking” valor=”02”/>
                <parametro nome=visibilidade” valor=”publico”/>
                <parametro nome=”feramenta_case” valor=”ArgoUML”/>
            </metadados>
        </modelo>
    </servico>
  </servicos>
</sds>
```

Uma vez estruturados e após serem persistidos, os modelos passam a fazer parte da base de reuso do SDS e estão aptos a serem utilizados em outros projetos.

Procedendo com o Reuso do Modelo de Autenticação e Autorização

Da mesma forma como apresentado no projeto REGIN, o mecanismo de autenticação adotado foi o padrão em aplicações do tipo WEB, onde há uma página de autenticação, possuindo dois campos, o de usuário e o de senha. Ambos são validados pelas mesmas regras, promovendo a identificação do usuário no momento de acesso ao sistema. No catálogo de modelos de SDS é possível encontrar o modelo desenvolvido

para o projeto REGIN, por meio da consulta utilizando o valor *"autenticação"*, sendo esta a versão utilizada como base para implantação no SISALI.

No projeto havia uma necessidade de se incorporar a ideia de serviços de TI, alinhando definitivamente a arquitetura do sistema com a do SDS. Neste caso, promoveu-se alteração no modelo de autenticação assim como apresentado na Figura 38.

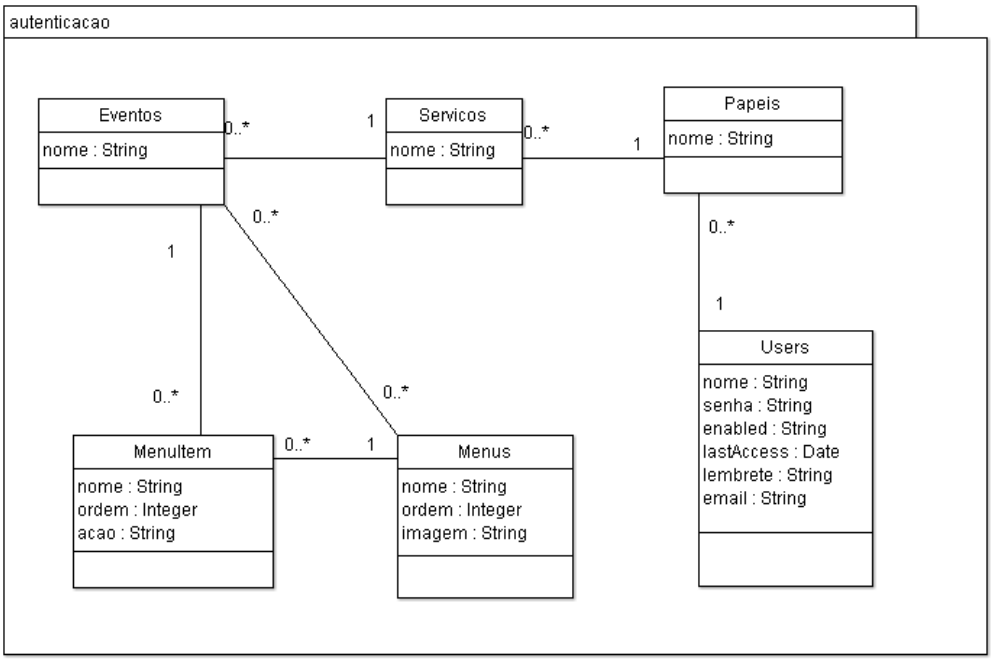

Figura 38: Segunda versão do modelo conceitual do controle de acesso do sistema.

Uma vez concluída a alteração, assim como apresentado na Tabela 27, a nova versão do modelo de autenticação foi reinserida no SDS, por meio da funcionalidade de cadastramento, e foi disponibilizada para o reuso.

Tabela 27: Definição da classe Servicos.

Atributo	Tipo	Descrição	Domínio
nome	String	Define o nome do serviço que passa a agrupar os eventos dentro de um determinado papel.	Texto livre de até 250 caracteres.

Procedendo com o Reuso do Modelo de Relatórios

O reuso do modelo de relatórios também se deu da forma planejada. Neste caso, as funcionalidades geradas para este serviço foram suficientes e não houve necessidade de criar uma nova versão do modelo. Sendo assim, a Figura 39 (réplica da Figura 34) apresenta o modelo que foi recuperado no SDS e do qual foi gerado o pacote de código que representa este serviço de TI no sistema SISALI.

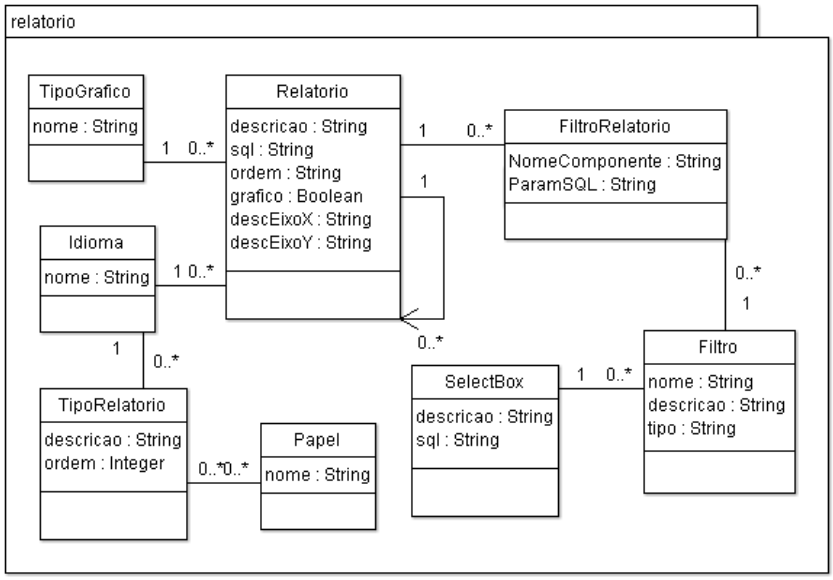

Figura 39: Modelo de dados do módulo de relatórios criado durante o projeto REGIN e reutilizado no projeto SISALI.

4.6. Fase 3: Análise dos resultados – avaliação sob a ótica de tempo e custo.

Os critérios utilizados para a análise da prova de conceito foram guiados pelo número de Pontos de Função economizados no desenvolvimento do projeto SISALI a partir do reuso dos modelos conceituais do projeto REGIN.

Dos Resultados Avaliados no Reuso de Ambos os Modelos

Como visto na Tabela 25, o número total de pontos de função contabilidades para o sistema SISALI foi de 1304 PF. Deste total, as funcionalidades as apresentadas na Tabela 28, foram atendidas pelo resultado do reuso dos modelos de relatórios e autenticação.

Tabela 28: Funcionalidades atendidas pelo reuso dos modelos de relatório e autenticação.

ID	Funcionalidades	Pontos de Função
26	Relatório de Disponibilidade do Equipamento	23
27	Relatório de Disponibilidade do Meio	34
28	Relatório de Tempo Médio de Falhas	23
29	Relatórios de Pedidos de Serviço por status	35
30	Relatório de Horas Apropriadas	41
31	Relatório de Acompanhamento de Serviços	41
32	Controle de Autenticação	59
	Total de Pontos de Função	**256**

Do total apurado de 1304 PF, 256 PF foram apropriados para as funcionalidades atingidas pelo reuso, o que determina um percentual de 19,63% do esforço total empreendido para o desenvolvimento do sistema.

Tabela 29: Comparação entre os números reais do sistema e os gerados pelos modelos conceituais de autenticação e autorização e relatórios.

Artefatos de Código Fonte do SISALI	Total	Autenticação	%	Relatórios	%
Módulos	5	1	20,00	1	20,00
Arquivos Java	1319	43	3,30	21	1,59
Arquivos de Visão JSP	635	17	2,70	6	0,94
Arquivos de Configuração (XML)	496	15	3,00	4	0,80
Arquivos JavaScript	488	7	1,40	3	0,61
Arquivos de Propriedades	4	1	25,00	1	25,00
Arquivos TLD	3	1	33,30	1	33,3
Arquivos CSS	11	1	9,00	1	9,00
Scripts por módulo do SISALI	4	1	25,00	1	25,00
Total de Artefatos de Software	2960	87	3,00	39	1,32

Paralelamente, como apresentado na Tabela 29, esse esforço foi concentrado em um total de 126 artefatos, de um total de 2960, o que representa um percentual de 4,32% do total de artefatos projeto.

O BFPUG avalia em [103], que o tempo médio para o desenvolvimento de um ponto de função de uma aplicação WEB na plataforma Java é de 7,5 h/PF. Sendo assim, a Tabela 30 apresenta o tempo mínimo estimado para cada funcionalidade do sistema SISALI.

Tabela 30: Mínimo de horas normalizadas para o desenvolvimento das funcionalidades do SISALI.

ID	Funcionalidade	Pontos de Função	Horas
1	Elaborar Pedido de Serviço Preventivo	41	307,5
2	Elaborar Pedido de Serviço Corretivo	82	615
3	Aprovar PS pelo Chefe de Departamento	35	262,5
4	Aprovar PS pelo Comandante	35	262,5
5	Aprovar PS pelo Comimsup	35	262,5
6	Receber PS	41	307,5
7	Aprovar PS pelo Gerente	23	172,5
8	Delinear PS	59	442,5
9	Orçar PS	59	442,5
10	Aprovar Orçamento pelo Gerente	59	442,5
11	Aprovar DS pelo Cliente	59	442,5
12	Aprovar DS pelo COMIMSUP	59	442,5
13	Aprovar DS pelo Gerente	59	442,5
14	Registrar no SIAFI	17	127,5
15	Liberar DS Sem Recursos	46	345
16	Planejar PS	43	322
17	Abrir OS	35	262,5
18	Apropriar HH	40	300
19	Indicar Gastos	37	277,5
20	Aprovar Indicação de Gastos	35	262,5
21	Faturar Gastos	41	307,5
22	Aprovar Fatura pelo Cliente	34	255
23	Liquidar Fatura	34	255
24	Interromper OS	17	127,5
25	Reabrir OS	23	172,5
26	Relatório de Disponibilidade do Equipamento	23	172,5
27	Relatório de Disponibilidade do Meio	34	255
28	Relatório de Tempo Médio de Falhas	23	172,5
29	Relatórios de Pedidos de Serviço por status	35	262,5
30	Relatório de Horas Apropriadas	41	307,5
31	Relatório de Acompanhamento de Serviços	41	307,5
32	Controle de Autenticação	59	442,5
	Total de Pontos de Função	**1304***	**9779,5***

*** Cálculos de Pontos de Função realizados a partir das regras estabelecidas pelo IFPUG e cálculo de horas definido pelo BFPUG. Fontes: www.ifpug.org e www.bfpug.com.br.**

Do total de horas calculado em 9779,5h, 19,63% destinar-se-iam ao desenvolvimento das funcionalidades recriadas a partir dos modelos reutilizados, ou seja, 1919,71h seriam destinadas ao desenvolvimento das funcionalidades apresentadas na Tabela 30. Dificilmente, pode-se afirmar que o desenvolvimento destas funcionalidades consumiu somente esse subtotal de horas, mas pode-se inferir que esta é uma estimativa próxima ao que foi efetivamente economizado do total do projeto.

Finalmente, do valor estimado para esta primeira etapa de desenvolvimento do projeto, também calculado a partir do total do número de pontos de função, temos: 1304 PF * R$ 400,00 = R$ 413.600,00 (quatrocentos e treze mil e seiscentos reais). Destes R$ 413.600,00, 19,63% podem ser atribuídos às funcionalidades que foram atendidas pelos procedimentos de reuso, logo, gastos em torno de R$ 81.189,68 foram economizados (observando que o valor de ponto de função foi definido no projeto).

4.7.Sumarização dos resultados

A partir dos objetivos identificados no início do capítulo, é possível sumarizar os resultados seguindo a orientação destacada.

Dos objetivos

Novamente, nos cenários de uso avaliados, provou-se que a abordagem de reuso é viável nos termos imaginados. O protótipo desenvolvido apresentou a capacidade de armazenamento dos modelos conceituais, suas categorizações em contextos de negócio, e versionamento. Há ainda a capacidade de distribuição e abrangência quando implantado como um Software as a Service.

Das questões

Definidas as questões de validação, a saber:

i. Uma vez criado o protótipo, como avaliar a sua capacidade de reutilizar modelos conceituais?

ii. Como calcular os percentuais de redução de esforço e investimento para cada modelo reutilizado?

Buscou-se a avaliação do protótipo e sua medição assim como destacados a seguir.

Das métricas

As medidas auferidas para cada uma das perguntas anteriores podem ser analisadas da seguinte forma:

i. Para o modelo conceitual de autenticação, verifica-se um cálculo de 47 PF no projeto REGIN e 59 PF no projeto SISALI.

ii. Para o modelo conceitual de relatório, verifica-se um cálculo de 197 PF em ambos os projetos.

iii. Para o modelo conceitual de autenticação, verifica-se um cálculo de 59 PF dentro de um total de 1304 PF do projeto SISALI, oferecendo uma economia de esforço e custo associado a ~4,5% do total do projeto.

iv. Para o modelo conceitual de relatório, verifica-se um cálculo de 197 PF de um total de 1304 PF do projeto SISALI, oferecendo uma economia de esforço e custo associado a ~15,13% do total do projeto.

Em termos absolutos, o percentual total de reuso chegou a aproximadamente ~19,63% do total de pontos de função do sistema. Essa economia medida apenas com o reuso de dois modelos de diferentes domínios de negócio, transversais e genéricos. Trabalhos futuros poderão analisar o reuso de modelos que possuam o mesmo domínio de negócio.

5. Conclusão

Implementar iniciativas de reuso não é uma tarefa simples e, como tal, demanda comprometimento e esforço. Por meio de um enfoque empírico e da combinação e aplicação das técnicas descritas ao longo desta tese, foi possível desenhar uma abordagem sistematizada de reuso de modelos conceituais capaz de apresentar economia real de tempo e custo.

Dos desafios enfrentados, destacam-se a concepção do arcabouço teórico e a criação do protótipo utilizado como prova de conceito, este último oferecendo um conjunto de funcionalidades que, alinhadas às GRU´s do nível E do MPS-BR, operam dentro de um fluxo de trabalho determinístico, a saber:

- Cadastramento e agrupamento dos modelos conceituais por meio da parametrização de características e variáveis de diferenciação e variação.
- Aceleração do tempo de desenvolvimento por meio do uso de mecanismos de geração de código.
- Estabelecimento de uma base central para armazenamento dos modelos conceituais e a viabilização do compartilhamento entre diferentes equipes de projetos.

À luz de uma evolução contínua, devem-se ressaltar algumas importantes características, a saber:

- O reuso como força direcionadora, conduz os usuários a compartilhar suas implementações de modelos conceituais.

- A evolução do projeto é automaticamente registrada e armazenada na conta do usuário, facilitando a inferência de indicadores de qualidade e produtividade.

- O baixo acoplamento entre os módulos torna-se uma meta importante, haja vista que a arquitetura é definida pelo serviço e não mais pelo desenvolvedor.

Da avaliação geral da abordagem de reuso

Como forma de fechar um diagnóstico de qualidade do ambiente de reuso, buscou-se identificar o seu alinhamento aos critérios de qualidade de produto de software assim como os definidos pela ISO/IEC 9126.

1. Sob a análise do fator funcionalidade: com o uso do SDS foi possível reaproveitar 100% do modelo conceitual de autenticação desenvolvido no projeto 1, o que representou 85% do modelo final do projeto 2, e recriar 3,0% do número absoluto de artefatos de software, além de contabilizar 4,5% do total de pontos de função do projeto 2.

   Ainda sob a análise do fator funcionalidade: com o uso do SDS foi possível reaproveitar 100% do modelo conceitual de relatórios do projeto 1, e recriar 1,32% do número absoluto de artefatos de software, além de contabilizar 15,13% do total de pontos de função do projeto 2.

2. Sob a análise de confiabilidade: pouco das informações embarcadas no modelo conceitual se perdeu. Praticamente, todas as classes de domínio extraídas dos modelos reutilizados foram reaproveitadas com alguns ajustes quando necessário.

3. Sob a análise de usabilidade e facilidade de entendimento: ao se utilizar o SDS, buscou-se exatamente promover a usabilidade tanto das ferramentas de desenvolvimento quanto da capacidade de entendimento do código.

4. Sob a análise da eficiência: a padronização global da arquitetura permite maior escalabilidade e facilidade de evolução do sistema dentro de um modelo padronizado e integrado.

5. Sob a análise da portabilidade: buscou-se trabalhar sempre com modelo conceitual e documentos XML. Os resultados, apesar de não serem tão precisos, demonstraram alta capacidade de reuso em outras linguagens de programação, o que vai ao encontro da teoria de MDD.

6. Sob a análise da capacidade de manutenção: com a redução de complexidade da arquitetura, os desenvolvedores conseguem aumentar a taxa de produção de artefatos na medida apontada pelos resultados apresentados no capítulo 4.

7. Sob a análise da economicidade: as análises efetuadas, a partir do cálculo global de pontos de função, apresentaram um resultado de 19,63% de economia de tempo e custo.

Adicionalmente, é possível determinar alguns pontos fortes que são importantes para o SDS. Como ele trabalha produzindo os serviços com uma arquitetura MVC, apoiada por framework padrão de mercado, utilizando o modelo solicitação-resposta do protocolo HTTP, a sua implantação torna-se muito simples e de fácil condução pelos desenvolvedores, oferecendo as seguintes características:

- Desenvolvimento rápido com baixa curva de aprendizado.

- Commodity de mercado.
- Padronização.
- Fácil de manter e testar.

Como possui uma camada de abstração exclusiva para a persistência de dados, sua migração e portabilidade são favorecidas. Em geral, são poucos os artefatos de software que devem ser criados para apoiar uma determinada funcionalidade.

A abordagem provou-se flexível, permitindo a separação de responsabilidade, por meio da divisão e composição dos modelos em serviços de TI e, como consequência, isolamento funcional que ajuda na detecção de erros.

Em termos de produtividade, os resultados mostraram que o desenvolvedor consegue minimizar o esforço global para cumprir suas metas de implementação, a cada reuso de modelo dentro do projeto, favorecendo ações que ajudem a diminuir o tempo e o custo do desenvolvimento.

Em termos de segurança, o SDS está capacitado a produzir serviços que operem em ambiente seguro HTTPS, possui tecnologia interna que permite o sigilo de informações, logo os procedimentos de autenticação e autorização foram implantados neste modelo.

Finalmente, a proposta do SDS foi a de criar um ambiente de reuso que permita que as organizações publiquem seus modelos e criem aplicações dentro um ambiente controlável. Sendo assim, dos resultados alcançados, destacam-se a capacidade de agrupamento de modelos conceituais e o encadeamento lógico dos artefatos reutilizáveis. Ao compartilhar um modelo, todo o esforço realizado para sua criação pode ser reutilizado por outras equipes em outros locais de desenvolvimento, uma vez que a infraestrutura utilizada está instalada como serviço.

Trabalhos futuros

Como apresentado nos diversos trabalhos publicados, inúmeras são as possibilidades de pesquisa e desenvolvimento dos temas aqui abordados. Um que possui potencial e deve ser estudado mais profundamente foi publicado em [105]. Este trabalho dá ênfase a uma abordagem de análise de impacto utilizando-se tecnologia MDD como mecanismo operativo. O objetivo é avaliar como mudanças sofridas pelos requisitos funcionais afetam o código fonte. Tendo essa visibilidade é possível avaliar os riscos associados à implantação da mudança e pode determinar mecanismos de controle e garantia de qualidade, bem como estimar custo e esforço necessários para se proceder com a alteração.

Outra oportunidade de evolução deste trabalho está na análise das atividades de reuso utilizando conjuntos de modelos que sejam efetivamente representativos do contexto de negócio sendo analisado. A abordagem apresentada já oferece indícios que é possível reutilizar modelos transversais em contextos de negócio diferentes, agora o objetivo é provar que modelos que são representativos do negócio também tenham suas provas de conceito.

Finalmente, tem-se como oportunidade a criação de nichos de usuários dentro de domínios específicos de negócio, formando, por vez, redes de relacionamentos que levem a melhoria constante dos modelos conceituais desenvolvidos no período de seus projetos.

Referências

[1]. PFLEEGER, S. L. Engenharia de Software: Theory and Practice. 4ª. Edição, Prentice Hall, 2009.

[2]. PAPAZOGLOU, M. Webservices and SOA: Principles and Technology. Pearson Education Canada, 2ª Edição, 2012.

[3]. MDA, Model Driven Architecture, Object Management Group (OMG), http://www.omg.org/mda. Último acesso abril de 2014.

[4]. KRUEGER, C. Software Reuse. ACM Computing Surveys, junho 1992

[5]. EVANS, E., Domain-Driven Design: Tackling Complexity in the Heart of Software, Addison Wesley, US, 2003.

[6]. MARZULLO, F. P. SOA na Prática: inovando seu negócio por meio de soluções orientadas a serviços. Editora Novatec, Rio de Janeiro, 1ª Edição, 2009.

[7]. LUCRÉDIO, D.; BRITO, K. S.; ALVARO, A.; et al. Software reuse: The Brazilian industry scenario. J. Syst. Software, Elsevier Science Inc., New York, NY, USA, 2008.

[8]. ROCHA, A. R. C., MACHADO, C. A. F., MPS-BR- Melhoria de Processo de Software Brasileiro, Guia Geral, Softex, 2009.

[9]. FRAKES, W. B.; ISODA, S. Success factors of systematic software reuse. IEEE Software, 1994.

[10]. LEACH, R. L.; Software Reuse: Methods, Models, Cost. AfterMath, 2 ª Edição, 2012.

[11]. EZRAN, M.; MORISIO, M.; THULLY, C. Practical Software Reuse. London: Springer-Verlag, 2002, 222 p.

[12]. KRUEGER, C. Software Reuse. ACM Computing Surveys, 1992.

[13]. SZYPERSKI, C. Component Software – Beyomd Object- Oriented Programming. 2. ed. Londres: Addison-Wesley Publishing Company, 2002, 589 p.

[14]. RAVICHANDRAN, T.; ROTHENBERGER, M. Software Reuse Strategies and Component Markets. Communications of the ACM, agosto 2003.

[15]. PRIETO-DÍAZ, R.: Implementing Faceted Classification for Software Reuse, Software Engineering Notes, Vol. 34, no. 5, Maio de 1991.

[16]. ROGER, S. P. Software Engineering - A practitioner's Approach, McGraw-Hill, 2001.

[17]. REINEHR, S. S.; Reuso Sistematizado de Software e Linhas de Produto de Software no Setor Financeiro: Estudos de Caso no Brasil, Tese de doutorado, Escola Politécnica da Universidade de São Paulo, 2008.

[18]. BOSCH, J. Maturity and Evolution in Software Product Lines: Approaches, Artifacts and Organization. Software Product Lines: Second Internarional Conference, San Diego, 2002.

[19]. CLEMENTS, P.C.; NORTHROP, L. Software Product Lines: Practices and Patterns. Reading, MA: Addison-Wesley Publishing Company, 2002. 563 p.

[20]. ITEA2. Information Technology for European Advancement. www.itea2.org, último acesso em março de 2011.

[21]. VERLAGE, M.; KIESGEN, T. Five years of product line engineering in a small company. Proceedings of 27th International Conference on Software Engineering, 2005 (ICSE 2005), pp. 534-543.

[22]. SEI, Software Engineering Institute, Process Maturity Profile of the Software Community 1999 Year End Update, SEMA 3.00, Carnegie Mellon University, March 2000.

[23]. SEI, Software Engineering Institute, A Framework for Software Product Line Practice, version 5.0, Carnegie Mellon University, http://www.sei.cmu.edu/productlines/, ultimo acesso, abril de 2014.

[24]. GRINOLD, R. C.; KAHN, R. N. Active Portfolio Management - A Quantitative Approach for Providing Superior Returns and Controlling Risk, Second Edition, McGraw-Hill, New York, 1999.

[25]. SEI, Software Engineering Institute, Capability Maturity Model Integration, version 1.3, Carnegie Mellon University, http://www.sei.cmu.edu/cmmi/, ultimo acesso, abril de 2014.

[26]. ISO/IEC, International Standard Organization. ISO/IEC 12207 – Software Engineering – Life Cycle Processes. 2008.

[27]. ESI, European Software Institute, Extensão a norma CMMI para famílias de produtos de software. http://www.esi.es/Families/E1.4b-Method-Catalogue/FAMILIES/Details.html#CMMI-SFE, último acesso abril de 2014.

[28]. AMBLER, S. W. Agile model driven development is good enough. IEEE Software, 2003.

[29]. MELLOR, S. J.; CLARK, A. N.; FUTAGAMI, T. Model-driven development. IEEE Software, 2003.

[30]. TOLVANEN, J.-P. Making model-based code generation work. Embedded Systems Europe, Agosto/Setembro 2004.

[31]. WILE, D. Lessons learned from real DSL experiments. Science of Computer Programming, Elsevier North-Holland, Inc., Amsterdam, Holanda, 2004.

[32]. KLEPPE, A.; WARMER, J.; BAST, W. MDA Explained - The Model Driven Architecture: Practice and Promise. Addison-Wesley, 2003. (Object Technology Series).

[33]. DEURSEN, A. van; KLINT, P. Little languages: little maintenance. Journal of Software Maintenance, John Wiley & Sons, Inc., New York, NY, USA.

[34]. MERNIK, M.; HEERING, J.; SLOANE, A. M. When and how to develop domain-specific languages. ACM Computing Surveys, dez. 2005.

[35]. OMG, Object Management Group, www.omg.org/, último acesso em janeiro de 2014.

[36]. OMG; OMG's MetaObject Facility, http://www.omg.org/mof/, Object Management Group, Inc. Último acesso em abril de 2014.

[37]. JMI; Java Metadata Interface. http://www.oracle.com/technetwork/java/index.html /, Oracle Corporation. Último acesso em abril de 2014.

[38]. MDR, MetaData Repository, First Open Source Tools Platform to Implement Model Driven Architecture (MDA), http://netbeans.org/about/press/8.html, Oracle. Último acesso em abril 2014.

[39]. ECLIPSE, Interface de Desenvolvimento Eclipse, IBM, www.eclipse.org, último acesso abril de 2014.

[40]. EMFT; Eclipse Modeling Framework Technology. Eclipse, IBM. www.eclipse.org/modeling/emft/?project=ecoretools. Último acesso: Janeiro de 2014.

[41]. MCNEIL, K.; Metamodeling with EMF: Generating concrete, reusable Java snipets. How to Extend the Eclipse Ecore Metamodel. IBM, 21 Setembro de 2010.

[42]. GMT; Generative Modeling Technologies. Eclipse, IBM. www.eclipse.org/gmt. Último acesso: Janeiro de 2014.

[43]. CLEAVELAND, J. C. Building application generators. IEEE Software, 1988.

[44]. LEVENDOVSZKY, T.; KARSAY, G.; MAROTI, M.; LEDECZIL, A.; CHARAF, H.; Model reuse with metamodel-based transformations, Institute for Software Integrated Systems, Vanderbilt University, Nashville, 2002.

[45]. LEE, T.; Hou, J.S.; Model Expansion in Model-Driven Architectures. Software Reuse in the Emerging Cloud Computing Era. IGI Global, 2012. http://www.igi-global.com/chapter/model-expansion-model-driven-architectures/65166. Último acesso em abril de 2014.

[46]. ROTHENBERGERA, M. A.; Hershauerb, J. C.; A software reuse measure: monitoring an enterprise-level model driven development process. Journal of Information and Management. Elsevier, volume 35, issue 5, 3 de maio de 1999.

[47]. MARZULLO, Fabio P.; SOUZA, Jano M. A Domain-Driven Development Approach for Enterprise Applications, using MDA, SOA and Web Services. Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, Washington, 2008.

[48]. HASSAN, Q. F.; Service-Oriented Architecture Adoption Challenges. IGI Global, 2012. http://www.igi-global.com/chapter/service-oriented-architecture/65168. Último acesso em abril de 2014.

[49]. EARL, T., Service-Oriented Architecture (SOA): Concepts, Technology, and Design, Prentice Hall, 2005.

[50]. ABUJAROUR, M.; Naumann, F.; Towards a Diamond DOA Operational Model. Service-Oriented Computing and Applications (SOCA), Perth, Australia, IEEE, 2010.

[51]. BEAL, A. Gestão Estratégica da Informação: como transformar a informação e a tecnologia da informação em fatores de crescimento e de alto desempenho nas organizações, Editora Atlas, São Paulo, 1ª. Edição, 2009.

[52]. JOSUTTIS, Nicolai M. SOA in Practice – The Art of Distributed System Design. Sebastopol: O'Reilly Media Inc, 2007.

[53]. BPM; Business Process Management Initiative, http://www.bpmn.org/. Último acesso em abril de 2014.

[54]. JBPM; JBoss JBPM, http://www.jboss.com/products/jbpm. Último acesso em abril de 2014.

[55]. HEWITT, E.; Java SOA Cookbook. O´Reilly Media, 1a. Edição, 2009.

[56]. BROTTIER, E.; FLEUREY, F.; STEEL, J.; BAUDRY, B.; TRAON, Y. L.; Metamodel-based Test Generation for Model Transformations: an Algorithm and a Tool. 17th ISSRE '06 - International Symposium on Software Reliability Engineering, 2006.

[57]. MEDEIROS, F. L.; Metamodelagem com Meta Object Facility, Relatório Final do trabalho para a conclusão do curso de Bacharelado em Ciências da Computação da Universidade Federal de Santa Catarina. Florianópolis, 2004.

[58]. POHL, K.; BÖCKLE, G.; LINDEN, F.V.D.L. Software Product Line Engineering: Foundations, Principles and Techniques. Berlin, Heidelberg: Springer-Verlag, 2005, 467 p.

[59]. ESTUBLIER, J.; VEJA, G.; Reuse and Variability in Large Software Applications. ESEC-FSE'05, Lisbon, Portugal, September 5–9, 2005.

[60]. ROSENMÜLLER, M; SIEGMUND, N; THÜM, T; SAAKE, G.; Multi-Dimensional Variability Modeling, VaMoS '11 January 27-29, 2011, Namur, Belgium.

[61]. WITHEY, J. V.; Implementing model based software engineering in your organization: An approach to domain engineering, 1994. CMU/SEI-94-TR-01,

disponível em: http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=12191. Último acesso em abril de 2014..

[62]. LEITE, A.; GIRARDI, R. Um Processo para a Engenharia de Domínio e de Aplicações Multiagente: As Fases de Projeto e Domínio e de Aplicação. III Simpósio Brasileiro de Componentes, Arquiteturas e Reutilização de Software. SBCARS 2009.

[63]. SOMMERVILLE, I., Engenharia de Software. 6ed. São Paulo, Addison Wesley 2003.

[64]. MOROS, B.; VICENTE-CHICOTE, C.; Toval, A.; Metamodeling Variability to Enable Requirements Reuse. Proceedings of EMMSAD 2008, Montpellier, France, 2008.

[65]. VASILECAS, O.; BUGAITE, D.; Applying the Meta-Model Based Approach to The Transformation of Ontology Axioms Into Rule Model, ISSN 1392 – 124X Information Technology and Control, 2007, Vol.36, No.1A.

[66]. NEIGHBORS, J. M. Software Construction Using Components. Tese (Ph.D. Thesis) University of California at Irvine, 1980.

[67]. LEITE, J. C. S. P.; Santanna, M.; Freitas, F. G.; Draco-PUC: a Technology Assembly for Domain Oriented Software Development. 3rd Conference on Software Reuse: Advances in Software Reusability, IEEE, 1994.

[68]. KANG, K. et al., "Feature-Oriented Domain Analysis - Feasibility Study", In: SEI Technical Report CMU/SEI-90-TR-21, Novembro, 1990.

[69]. SIMONS, M. "Organization Domain Modeling (ODM): Domain Engineering as a CoMetodology to Object-Oriented Technics", In: Fusion Newsletter, Volume 4.4, pp. 13-16, Hewlett-Packard Laboratories, Outubro, 1996.

[70]. LAMMERS, T. e S. Chan, “OODE”, Tutorial na 5a Conferência Internacional em Reutilização de Software (ICSR-5), ACM/IEEE, Victoria, Canadá, Junho, 1998.

[71]. VICI, A. et al., “FODACom: An Experience with Domain Analysis in the Italian Telecom Industry”, 5a Conferência Internacional em Reutilização de Software (ICSR-5), ACM/IEEE, ACM/IEEE, Victoria, Canadá, Junho, 1998.

[72]. MILER Jr., N.; A Engenharia de Aplicações no Contexto da Reutilização Baseada em Modelos de Domínio. Tese de Doutorado, COPPE/UFRJ. Rio de Janeiro, 2000.

[73]. LUCRÉDIO, D.; Uma Abordagem Orientada a Modelos para Reutilização de Software, USP, São Carlos, SP, 2009.

[74]. MODELWARE. Architecture and Platform Modelling Profiles. [S.l.], 2006.

[75]. ANASTASOPOULOS, M.; ATKINSON, C.; MUTHIG, D. A concrete method for developing and applying product line architectures. In: AKSIT, M.; MEZINI, M.; UNLAND, R. (Ed.). NetObjectDays. [S.l.]: Springer, 2002. (Lecture Notes in Computer Science, v. 2591), p. 294–312. ISBN 3-540-00737-7.

[76]. SOA-RA. Reference Architecture for Service Oriented Architecture. OASIS, 2008. http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-pr-01.html. Último acesso em abril de 2014.

[77]. SOA Reference Architecture, The Open Group Architecture Framework. https://www2.opengroup.org/ogsys/catalog/C119, Dec 2011. Último acesso em abril de 2014.

[78]. MONTONI, M., A.; ROCHA, A. R.; WEBER, K. C.; MPS-BR A Successful Program for Software Process Improvement in Brazil. Published online in Wiley Inter Science. www.interscience.wiley.com. DOI: 10.10002/spip,428. 2009.

[79]. CZARNECKI, K. et al. Model-driven software product lines. 20th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2005). San Diego, CA, USA, 2005.

[80]. BARUCHELLI, F.; A fuzzy approach to faceted classification and retrieval of reusable software components. ACM SIGAPP Applied Computing Review – Special issue on open perspective in software reuse. Volume 5, issue 1, Spring, 1997.

[81]. JACOBSON, I., GRISS, M., JONSSON, P., Software Reuse. Architecture, Process and Organization for Business Success, Addison Wesley, 1997.

[82]. MORISIO, M.; EZRAN, M.; TULLY, C. Success and failure factors in software reuse. IEEE Transactions on Software Engineering, 2002.

[83]. BOEHM, B. COCOMO II - COnstructive COst MOdel II, http://sunset.usc.edu/csse/research/COCOMOII/cocomo_main.html, último acesso março de 2014.

[84]. BORGES, B. G. Extração de métricas em software orientado a objetos. Engenharia de Software Magazine, edição 26, Brasil, 2011.

[85]. MARTHALER, V. Function Point Counting Practices Manual, International Function Point Users Group (IFPUG), EDS, Troy, Michigan, 2000.

[86]. SOON, C. K.; UML and Function-Class Decomposition for Embedded Software Design. Department of Computer Science, Texas State University, San Marcos, R.T. 2006.

[87]. OASIS. Organization for the Advancement of Structured Information Standards, 2009.

[88]. MARZULLO, F. P. ; SOUZA, J. M. . A Domain-Driven Development Approach for Enterprise Applications,using MDA, SOA and Web Services. In: IEEE

Conference on E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services (CEC and EEE 2008), 2008, Washington. 2008.

[89]. ANASTASOPOULOS, M.; GACEK, C. Implementing product line variabilities. In: Symposium on Software Reusability (SSR). Toronto, Canada , 2001.

[90]. W3C, World Wide Web Consortium, www.w3c.org, último acesso: fevereiro 2014.

[91]. IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications, Software Engineering Standards Committee of the IEEE Computer Society, IEEE-SA Standards Board, 1998.

[92]. ARGOUML, UML Modelling Tool. http://argouml.tigris.org. Último acesso: dezembro de 2013.

[93]. JEE, Java Enterprise Edition, Oracle. http://java.oracle.com, último acesso janeiro de 2014.

[94]. STRUTS, Struts Project, Apache, http://struts.apache.org/. Último acesso abril 2014.

[95]. HIBERNATE, Hibernate Project, Red Hat , http://www.jboss.org, último acesso setembro de 2013.

[96]. DWR, Direct Web Remoting, http://directwebremoting.org/dwr/index.html. Último acesso em abril de 2014.

[97]. JAS, JBoss Application Server, Red Hat. http://www.jboss.org, último acesso dezembro de 2013.

[98]. MDA4ECLIPSE, MDA4eclipse Research Project, www.mda4eclipse.com.br. Último acesso abril 2014.

[99]. TRAVASSOS, G. H., BARROS, M. O., WERNER, C. M. L., 2002, "Um Estudo Experimental sobre a Utilização de Modelagem e Simulação no Apoio à Gerência e Projetos de Software". In: XVI Simpósio Brasileiro de Engenharia de Software, Gramado, RS.

[100]. Marzullo, F.; Souza, J. M.; Integrating Brazilian e-Government Initiatives in a Service-oriented Architecture, accepted at IADIS, WWW-Internet Conference, Fort Worth, Texas, 2013.

[101]. O Globo Online Brasil é o país mais afetado pela burocracia em todo o mundo, conclui pesquisa, June, 28th, 2007, Rio de Janeiro, Brazil http://oglobo.globo.com/economia/mat/2007/06/28/296558171.asp. Último acesso em abril de 2014.

[102]. Diretoria-Geral do Material da Marinha, Portaria no. 51/DGMM, Marinha do Brasil, Rio de Janeiro, 20 de abril de 2007.

[103]. BFPUG, Brazilian Function Point User Group, Qual a produtividade do Java? http://www.bfpug.com.br/Produtividade_Java.htm, Acesso em: abril de 2014.

[104]. MD, Ministério da Defesa, Comando da Aeronáutica, Diretoria de Material Aeronáutico, Termo de Homologação do Pregão Eletrônico número 00016/2012 (SRP), 2012.

[105]. MARZULLO, F. P.; SOUZA, J. M.; et al. A Model-Driven Development (MDD) Approach to Change Impact Analysis. In: 31st ICIS 2010 - International Conference on Information Systems 2010, 2010, St. Louis.

[106]. SOUTHEKAL, P. H. A Measurement Framework for Software Projects, Amazon Digital Services, Edição 1. Novembro de 2011.

[107]. B. Kitchenham, “Procedures for performing systematic reviews,” Keele, UK, Keele University, vol. 33, p. 2004, 2004.

Trabalhos mais importantes publicados no período da tese

Artigos completos publicados em periódicos

MARZULLO, F. P.; SOUZA, J. M. . New Directions for IT Governance in the Brazilian Government. International journal of electronic government research, v. 5, p. 57-69, 2009.

MARZULLO, F. P.; SOUZA, J. M. . A Domain-Driven Development Approach, using MDA, SOA and Web Services. JOURNAL OF ELECTRONICS AND COMPUTER SCIENCE, v. 10, p. 1-6, 2008.

MARZULLO, F. P.; SOUZA, J. M. . MDA4Eclipse - An MDA Approach for Project Assessment. Journal of Electronics & Computer Science, v. 10, p. 1-6, 2008.

Livros publicados/organizados ou edições

MARZULLO, F. P.. SOA na Prática: inovando seu negócio por meio de soluções orientadas a serviços. 1. ed. São Paulo: Novatec, 2009. v. 1. 392p .

MARZULLO, F. P.. iPhone na Prática: aprenda passo a passo como desenvolver soluções para iOS. 1. ed. São Paulo: Novatec, 2012. v. 1. 272p .

Capítulos de livros publicados

MARZULLO, F. P.; SOUZA, J. M. . New Directions for IT Governance in the Brazilian Government. In: Dr. Vishanth Weerakkody. (Org.). Applied Technology Integration in Governmental Organizations: New E-Government Research. Pennsylvania: IGI Global, 2011.

MARZULLO, F. P.; SOUZA, J. M. . An MDA Approach for Database Profiling and Performance Assessment. In: Roger Lee; Haeng-Kon Kim. (Org.). Studies in Computational Intelligence: Computer and Information Science. 1ed.Berlim/Alemanha: Springer-Verlag, 2008, v. 131, p. 1-282.

Trabalhos completos publicados em anais de congressos

MARZULLO, F. P.; SOUZA, J. M. . A Case Study on Software as Service Approach to Model-Driven Development. In: The 2011 World Congress in Computer Science, Computer Engineering, and Applied Computing (WORLDCOMP'11), 2011, Las Vegas.

MARZULLO, F. P.; SOUZA, J. M. . A Case Study on a Student Relationship Management Approach Supported by a Knowledge Management Tool. In: The 2010 World Congress in Computer Science, Computer Engineering, and Applied Computing, 2010, Las Vegas, Nevada.

MARZULLO, F. P.; SOUZA, J. M. . A Model-Driven Development (MDD) Approach to Change Impact Analysis. In: 31st ICIS 2010 - International Conference on Information Systems 2010, 2010, St. Louis. 31st ICIS 2010 - International Conference on Information Systems 2010, 2010.

MARZULLO, F. P.; SOUZA, J. M. . A COST MODEL FOR DOMAIN-DRIVEN DEVELOPMENT USING BPM, MDA, SOA AND WEB SERVICES. In: IADIS INTERNATIONAL CONFERENCE WWW/INTERNET 2009, 2009, Roma.

MARZULLO, F. P.; SOUZA, J. M. . REQUIREMENT-BASED COSTING. In: The 2009 World Congress in Computer Science, Computer Engineering, and Applied Computing, 2009, Las Vegas, Nevada. The 2010 International Conference on e-Learning, e-Business, Enterprise Information Systems, and e-Government (EEE'10), 2009. v. 1.

MARZULLO, F. P.; SOUZA, J. M. . NEW DIRECTIONS FOR IT GOVERNANCE IN THE BRAZILIAN GOVERNMENT. In: IEEE - International Conference for Enterprise Information Systems, 2008, Barcelona. ICEIS 2008, 2008.

MARZULLO, F. P.; SOUZA, J. M. . An MDA Approach for Database Profiling and Performance Assessment. In: 7th IEEE/ACIS International Conference on Computer and Information Science, 2008, Portland.

MARZULLO, F. P.; SOUZA, J. M. . A Practical MDA Approach for Autonomic Profiling and Performance Assessment. In: ECMDA08 - European Conference for Model Driven Architecture, 2008, Berlin.

MARZULLO, F. P.; SOUZA, J. M. . Entendendo a Governança de Tecnologia da Informação: novas direções para o setor público. In: Congresso CONSAD de Gestão Pública, 2008, Brasília. Congresso CONSAD de Gestão Pública, 2008.

MARZULLO, F. P.; SOUZA, J. M. . A Study Case on Domain-Driven Development, using MDA, SOA and Web Services. In: SOA Summit, 2008, Hawaii. http://conferences.computer.org/services/2008/soasummit.htm, 2008.

MARZULLO, F. P.; SOUZA, J. M. . A Competence Based Approach to IT Governance in the Brazilian. In: WORLDCOMP'08 - The 2008 World Congress in Computer Science, Computer Engineering, and Applied Computing, 2008, Las Vegas, Nevada.

MARZULLO, F. P.; SOUZA, J. M. . An Autonomic Approach for Model Driven Development and Database Performance Assessment. In: 32nd Annual IEEE International Computer Software and Applications Conference, 2008, Turku. 32nd Annual IEEE International Computer Software and Applications Conference, 2008.

MARZULLO, F. P.; SOUZA, J. M. . A Domain-Driven Development Approach for Enterprise Applications,using MDA, SOA and Web Services. In: IEEE Conference on E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services (CEC and EEE 2008), 2008, Washington.

MARZULLO, F. P.; SOUZA, J. M. . A Domain-Driven Approach for Enterprise Development, using BPM, MDA, SOA and Web Services. In: The 5th International Conference on Innovations in Information Technology (Innovations 08), 2008, Dubai. The 5th International Conference on Innovations in Information Technology (Innovations 08), 2008.

MARZULLO, F. P.; SOUZA, J. M. . O ALINHAMENTO DA TI AO NEGÓCIO DA ORGANIZAÇÃO - DIFERENCIAL COMPETITIVO. 2008. (Apresentação de Trabalho/Outra).

Anexo I

Assim como os procedimentos apresentados em [107], a abordagem escolhida para justificar hipótese de reuso de modelos em um ambiente de Software as a Service e alinhamento de processo com o nível E do MPS-BR, tem como objetivo certificar que o tema é relevante para a academia e possui mérito próprio, sendo válido como tese de doutorado.

A sistemática de identificação das pesquisas na área de interesse, foi definida utilizando o GQM [106] e abrangeu as principais bibliotecas digitais e sistemas indexadores de pesquisas, a saber:

1. Plataforma Lattes, por mio da plataforma Aquarius.
2. ACM.
3. IEEE Xplore.
4. Scopus.
5. DBLP.
6. ISI.
7. Google.

Contexto

A premissa foi a de que havia uma lacuna na literatura de pesquisas que endereçassem o tema reuso de modelos conceituais em ambiente de software as a servisse em conjunto à gerência de reuso do nível E do MPS-BR.

Objetivo

Dentro desse contexto, o objetivo foi executar uma busca sistemática nos principais índices de pesquisas existentes, de forma a estabelecer o real escopo da tese, bem como apoiar a escolha das tecnologias e a abordagem de convergência em uma tese de doutorado. Utilizou-se o GQM [106] como método de análise dentro do seguinte cenário:

Meta: identificar publicações que possuam o uso combinado dos termos "reuso de modelos conceituais", "Software as a Service" e "Gerência de Reuso do Nível E do MPS-BR".

Questão: existe pesquisa associada a alguma abordagem de reuso de modelos conceituais, disponibilizada em ambiente de Software as a Service e alinhada à Gerência de Reuso do nível E do MPS-BR?

Métrica: número de publicações encontradas dentro do perfil teórico desejado.

Linguagens

As linguagens de interesse e que foram utilizadas na busca são Português (Brasil) e Inglês.

Escopo

A pesquisa foi realizada usando os principais mecanismos de indexação de publicações. Em cada biblioteca digital foram considerados termos individuais, tais como "reuso" e "modelos", os quais resultavam em um conjunto de pesquisas muito abrangente e sem foco; e termos compostos como os apresentados em cada uma das

tabelas a seguir. Sendo assim, as Tabelas de 1 a 7 apresentam o escopo e os seus resultados a partir do conjunto termos usados como critérios de busca.

Tabela 31: Busca efetuada na plataforma Lattes, fonte: aquarius.mcti.org.br. Total de registros pesquisados 49.760.820.

Termos da busca	Resultados
reuso = %reuso%	2661
reuso de modelos = %reuso%modelos%	1
modelo de processo de software = %modelo%processo%software%	174
reuso de modelos conceituais = '%reuso%modelos%conceituais%'	0
reuso de modelos conceituais no MPS-BR = %reuso%modelos%mps%br%como serviço"	0
reuso e MPS-BR = %reuso%mps-br%	0

Tabela 32: Busca efetuada na biblioteca digital da ACM, fonte: acm.org. Total de registros pesquisados 2.221.107.

Termos da busca	Resultados
reuse	26.714
model reuse	22.200
conceptual model reuse	4.174
"conceptual model" + reuse	806
"conceptual model reuse"	0
conceptual model reuse mps-br	0
"conceptual model" + reuse + mps-br	0

Tabela 33: Busca efetuada na biblioteca digital do IEEE Xplore, fonte: http://ieeexplore.ieee.org/Xplore/home.jsp. Total de registros pesquisados 3.688.013.

Termos da busca	Resultados
reuse	14.005
model reuse	4.875
conceptual model reuse	145
"conceptual model" + reuse	102
"conceptual model reuse"	0
conceptual model reuse mps-br	0
"conceptual model" + reuse + mps-br	0

Tabela 34: Busca efetuada na biblioteca digital do Scopus, fonte: http://www.scopus.com/. Total de registros pesquisados não apresentado.

Termos da busca	Resultados
reuse	56.632
model reuse	12.968

conceptual model reuse	584
"conceptual model" + reuse	202
"conceptual model reuse"	1
conceptual model reuse mps-br	0
"conceptual model" + reuse + mps-br	0

Tabela 35: Busca efetuada na biblioteca digital DBLP, fonte: http://dblp.uni-trier.de/db/. Total de registros pesquisados: 2.587.353.

Termos da busca	Resultados
reuse	3.561
model reuse	315
conceptual model reuse	11
"conceptual model" + reuse	0
"conceptual model reuse"	0
conceptual model reuse mps-br	0
"conceptual model" + reuse + mps-br	0

Tabela 36: Busca efetuada na biblioteca digital ISI, fonte: http://www.isi.edu. Total de registros indexados não apresentado.

Termos da busca	Resultados
reuse	239
model reuse	93
conceptual model reuse	13
"conceptual model" + reuse	0
"conceptual model reuse"	0
conceptual model reuse mps-br	0
"conceptual model" + reuse + mps-br	0

Por fim, como forma de dar mais completude e oportunidade àquelas publicações que não constam no acervo das bibliotecas digitais, foi realizada uma busca no Google, também utilizando os termos definidos anteriormente, porém, neste caso, optou-se por realiza-la nas duas linguagens definidas.

Tabela 37: Busca efetuada no Google ®, fonte: www.google.com. Total de páginas indexadas pelo Google até a data da finalização desta tese: ~35 milhões.

Termos da busca	Resultados
reuse	~22.800.000
“model reuse”	~28.000
“conceptual model reuse”	10
“conceptual model reuse as a service”	0

“conceptual model reuse as a service aligned with MPS-BR”	0
“conceptual model reuse as a service aligned with MPS-BR as a service”	0
“conceptual model reuse as a service aligned with MPS-BR as a service environment”	0
“reuso de modelos conceituais”	7
“reuso de modelos conceituais como serviço”	0
“reuso de modelos conceituais alinhado ao MPS-BR”	0
“reuso de modelos conceituais alinhado ao MPS-BR como serviço”	0
"ambiente de reuso de modelos conceituais alinhado ao MPS-BR como serviço"	0
“reutilização de modelos conceituais”	6
“reutilização de modelos conceituais como serviço”	0
“reutilização de modelos conceituais alinhado ao MPS-BR”	0
“reuso de modelos conceituais alinhado ao MPS-BR como serviço”	0
"ambiente de reutilização de modelos conceituais alinhado ao MPS-BR como serviço"	0

Análise

Dos resultados encontrados em cada biblioteca, um levantamento quantitativo foi feito para identificar, dentro do escopo da tese, quais eram relevantes. De todo o conjunto de pesquisas, apenas uma endereçava o reuso de modelo com uma abordagem viável e aplicável. A tese em questão foi desenvolvida por Lucrédio em [73], e apresenta arcabouço teórico sobre como abordagens orientadas a modelos podem ser utilizadas como ferramentas de reuso. Finalmente, nenhuma apresentou o nível de alinhamento esperado à gerência de reuso do MPS-BR ou foi desenvolvida em abordagem Software as a Service, visando a distribuição e amplitude de uso do ambiente.