COPPE/UFRJ

INFO CASES:
UM MODELO INTEGRADO DE REQUISITOS COM CASOS DE USO

Michel Heluey Fortuna

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): Cláudia Maria Lima Werner
Marcos Roberto da Silva Borges

Rio de Janeiro
Dezembro de 2008

INFO CASES:

UM MODELO INTEGRADO DE REQUISITOS COM CASOS DE USO

Michel Heluey Fortuna

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.

Aprovada por:

Prof.ª Cláudia Maria Lima Werner, D.Sc.

Prof. Marcos Roberto da Silva Borges, Ph.D.

Prof.ª Ana Regina Cavalcanti da Rocha, D.Sc.

Prof. Guilherme Horta Travassos, D.Sc.

Prof. Júlio Cesar Sampaio do Prado Leite, Ph.D.

Prof.ª Rénata Mendes Araujo, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

DEZEMBRO DE 2008

Fortuna, Michel Heluey

Info Cases: Um Modelo Integrado de Requisitos com Casos de Uso/ Michel Heluey Fortuna. – Rio de Janeiro: UFRJ/COPPE, 2008.

XIV, 200 p.: il.; 29,7 cm.

Orientadores: Cláudia Maria Lima Werner/ Marcos Roberto da Silva Borges

Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2008.

Referencias Bibliográficas: p. 172-180.

1. Info cases. 2. Casos de uso. 3. Nível informacional de Objetivos. 4. Interface informacional de casos de uso. I. Werner, Cláudia Maria Lima *et al*. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Titulo.

Aos meus pais Feliciano e Jorgette (*in memorian*),
à minha esposa Teresinha e
às minhas filhas Luiza e Raquel.

Agradecimentos

Aos meus orientadores Cláudia Werner e Marcos Borges, pelo apoio e orientação nos momentos certos.

Aos professores Ana Regina Rocha, Júlio Leite, Guilherme Travassos e Renata Araújo, que aceitaram participar da minha banca. Aos dois primeiros, também pela orientação valiosa durante o meu exame de qualificação.

Aos meus professores do PESC/COPPE, Cláudia Werner e Guilherme Travassos, e aos do NCE/IM, Marcos Borges, Flávia Santoro, Maria Luiza Campos, Renata Araújo, Carlo Emmanoel Oliveira, bem como a muitos outros que os precederam, pelo tanto que contribuíram para a minha formação.

Aos amigos e colegas Carlos Henrique Duarte, Leonardo Murta, Marco Antônio Araújo e Tarcísio Lima, pela valiosa ajuda durante o trabalho de tese.

Aos meus alunos William Fernandes, Vitor Padilha Gonçalves e Gilmar Pedretti, além de muitos outros, pela amizade, participação e contribuição nos estudos experimentais.

Aos colegas do NCE e da COPPE, pela convivência amiga e instrutiva.

À Taísa, às secretarias do PESC/COPPE e do Departamento de Ciência da Computação do IM, bem como a outros órgãos da UFRJ, pelo apoio e pronto atendimento às minhas inúmeras solicitações.

À UFJF por ter viabilizado o meu doutorado, ao PESC/COPPE pelo apoio financeiro para apresentação de artigos em conferências, e à CAPES pelo suporte financeiro concedido na forma de bolsa de estudos.

À minha mãe Jorgette (*in memoriam*) que, silenciosamente, modelou boa parte do que eu sou. Ao meu pai Feliciano, que me mostrou o valor do trabalho e da perseverança.

Ao meu irmão Alexandre, que me apoiou e foi um companheiro sempre presente nos momentos decisivos.

À minha esposa Teresinha, pelo amor, apoio e paciência em todos esses anos.

Às minhas filhas Luiza e Raquel, pela doação espontânea e amorosa que, principalmente, as crianças são capazes.

A DEUS, expressão infinita do Amor, revelada em cada passo da minha vida.

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

INFO CASES:
UM MODELO INTEGRADO DE REQUISITOS COM CASOS DE USO

Michel Heluey Fortuna

Dezembro/2008

Orientadores: Cláudia Maria Lima Werner
Marcos Roberto da Silva Borges

Programa: Engenharia de Sistemas e Computação

Existem evidências de um "*gap*" entre a modelagem com casos de uso e a modelagem de (classes de) domínio, no desenvolvimento de um sistema, particularmente durante a fase de definição dos requisitos do sistema. Por exemplo, o nível de automatização alcançado nas propostas de geração do modelo de domínio a partir do modelo de casos de uso, ou na verificação da consistência entre eles, é baixo ou depende da interpretação do modelador. Além disso, já foi observado que diferentes modeladores, trabalhando de forma independente e com base nos casos de uso de um mesmo sistema, produzem modelos de domínio bastante discrepantes. Este trabalho analisa esses problemas e propõe uma especialização do modelo de casos de uso, para que o mesmo passe a servir como um modelo integrado de requisitos, a partir do qual um modelo de domínio possa ser derivado. Regras semiformais são apresentadas para demonstrar essa capacidade, assim como resultados de estudos experimentais realizados para avaliar o modelo proposto.

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

INFO CASES:
A REQUIREMENTS INTEGRATED MODEL WITH USE CASES

Michel Heluey Fortuna

December/2008

Advisors: Cláudia Maria Lima Werner
Marcos Roberto da Silva Borges

Department: Computer and Systems Engineering

There is evidence of a gap between use case modeling and domain modeling in the development of a system, particularly during the system requirements definition phase. For example, the level of automation achieved in proposals to generate the domain model from use cases, or to verify the consistency between them, is low or depends on the interpretation of the modeler. Moreover, it has already been seen that different modelers, working independently, produce very different domain models based on use cases of the same system. This work analyzes these problems and proposes a specialization of the use case model to serve as an integrated requirements model from which a domain model can be derived. Semi-formal rules are presented to demonstrate this capacity, as well as results of experimental studies carried out to assess the proposed model.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas e Abreviaturas

AE Análise Essencial

AOO Análise Orientada a Objetos

ERq: Engenharia de Requisitos

H-C: Humano-Computador (em: Interação H-C e Interface H-C)

IC e ICs: Info Case e Info Cases, respectivamente

II: Interface Informacional (de um IC)

IU: Interface com o Usuário

LRqs: Lista de requisitos

MIC: Modelagem (Modelo) de Info Cases

MIR: Modelagem (Modelo) Informacional de Requisitos, ou (dependendo do contexto) Modelo Integrado de Requisitos

MD: Modelagem (Modelo) (de Classes de Objetos) do Domínio

MOObj Modelagem de Requisitos Orientada a Objetivos

MUC: Modelagem (Modelo) de Requisitos com Casos de Uso

NIO: Nível Informacional de Objetivos

SI e SI's: Sistema de Informação e Sistemas de Informação, respectivamente.

UC e UCs: *Use Case* e *Use Cases*, respectivamente.

1. Introdução

1.1 Motivação

A técnica de casos de uso (do inglês *use cases* - UCs) (JACOBSON *et al.*, 1992) é bastante utilizada, principalmente para especificação de requisitos funcionais de sistemas de software. Em geral, ela emprega duas especificações: a) um diagrama denominado Diagrama de Casos de Uso, que apresenta graficamente os UCs, seus atores[1], e os canais de comunicação entre os atores e os UCs; e b) uma descrição, em linguagem natural, de cada caso de uso. Essa descrição foca a interação que deve ocorrer entre o sistema e seus atores, quando um UC é executado.

Os atores e outros usuários do futuro sistema (coletivamente designados como *stakeholders*) tendem a perceber o modelo de casos de uso (MUC) como familiar (KULAK, GUINEY, 2003). O diagrama de casos de uso é simples e os atores se vêem lá representados; além disso, a descrição dos UCs foca um aspecto muito concreto para eles – a interação com o sistema, descrevendo-a em uma linguagem acessível. Essas características certamente contribuíram para a ampla difusão da técnica.

Entretanto, UCs também têm problemas. A começar pela diversidade de formatos para a descrição dos UCs, que tende a confundir os modeladores. COCKBURN (2005) relacionou 18 (dezoito) formatos distintos. E a variação não é só no formato, mas também no conteúdo e no nível de abstração das descrições. Outro problema é que o critério normalmente utilizado para particionar o sistema em UCs é muito subjetivo (ROBERTSON, ROBERTSON, 2006), fornecendo pouca orientação para os modeladores, na escolha dos casos de uso. Estes e outros problemas da técnica de UCs têm sido debatidos pela comunidade científica de Engenharia de Requisitos.

A motivação para o estudo que resultou nesta tese veio da percepção empírica de outro problema, pelo autor desta tese. UCs enfatizam a modelagem dos aspectos comportamentais da interação dos atores com o sistema, deixando em segundo plano o tratamento da informação trocada entre eles. No contexto de sistemas de informação, onde, antes de tudo, o que interessa é saber que informação o sistema precisa receber e que informação se espera que ele seja capaz de prover, o esforço empregado em

[1] Pessoas ou outros sistemas que interagem com os UCs.

especificações predominantemente comportamentais, sem a devida atenção à informação envolvida na interação, não resulta em uma boa relação custo-benefício.

1.2 O Problema

No escopo desta tese estaremos utilizando o termo *modelo de domínio* (MD) no sentido indicado por LARMAN (2004): modelo de classes de objetos que constituem a camada de domínio do sistema, ou seja, de objetos de software que representam coisas do espaço de domínio do problema, com métodos relacionados à lógica da aplicação ou à lógica de domínio.

O problema atacado nesta tese não é aquele que motivou, inicialmente, os estudos e a pesquisa sobre UCs, embora guarde relação com ele[2]. O problema-alvo da tese ocorre durante a utilização conjunta do modelo de casos de uso (MUC) e do modelo de domínio do sistema (MD), durante a fase de levantamento e modelagem de requisitos de um sistema de informações, segundo o paradigma da Orientação a Objetos. Ele é aqui referido como "uma transição difícil entre os modelos", e foi identificado a partir de evidências, obtidas na literatura, do baixo grau de automatização possível para a derivação de um MD a partir do MUC, ou para a verificação da consistência entre os dois modelos. A literatura mostra que ambos os processos são altamente dependentes da interpretação do modelador, o que gera outros problemas, como por exemplo, a acentuada discrepância entre MDs obtidos por diferentes modeladores, para um mesmo sistema (SVETINOVIC *et al.,* 2005).

Vários são os trabalhos disponíveis na literatura que propõem algum tipo de solução para incrementar o grau de automatização da derivação de um MD a partir do MUC (por exemplo, (SUBRAMANIAM *et al.*, 2004) (KÄRKKÄINEN *et al.*, 2008)), ou da verificação da consistência entre os modelos (por exemplo, (GLINZ, 2000)). O caminho seguido para isso, contudo, é o da análise lingüística das descrições em linguagem natural dos UCs, o que força manter um grande envolvimento do modelador para interpretar os resultados, por conta da ambigüidade inerente às linguagens naturais. A solução construída nesta tese avança em relação às disponíveis na literatura no sentido de diminuir a intervenção do modelador para a obtenção de um MD a partir do MUC, reduzindo também o problema de se manter a consistência entre esses modelos.

[2] A técnica proposta nesta tese constitui uma solução para ambos.

1.3 Objetivo e Enfoque da Solução

O objetivo deste trabalho foi resolver o problema da transição difícil entre o MUC e o MD. Para isso, construímos uma variante da técnica de UCs, mais especificamente uma especialização dela, capaz de diminuir o grau de dependência da interpretação do modelador, no momento de se obter um MD a partir do MUC, ou de se verificar a consistência entre esses modelos. A nova técnica resultou da introdução de uma regra de determinação de UCs, e da exigência de um maior rigor na descrição dos fluxos de informações trocados entre o sistema e seus atores.

Além de caracterizar e analisar o problema envolvido na utilização dos dois modelos (MUC e MD), o enfoque de solução adotado é justificado e são apresentadas evidências de que ele é capaz de propiciar uma solução aceitável para o problema, incluindo a manutenção de pontos fortes da modelagem de requisitos com UCs.

1.4 Organização

Esta tese está organizada em sete capítulos, incluindo este primeiro de introdução, onde é feita, ainda, uma breve revisão da área de Engenharia de Requisitos, focando basicamente os assuntos mais relevantes para a tese.

O Capítulo 2 apresenta um resumo da técnica de UCs, discute as diversas abordagens dessa modelagem encontradas na literatura, e identifica e analisa seus pontos fortes e fracos.

O Capítulo 3 justifica a utilização conjunta do MUC e do MD na fase de requisitos, dá um panorama geral do estado da arte e da prática dessa utilização, discute o problema-alvo da tese, e detalha os objetivos deste trabalho.

O Capítulo 4 retoma o problema-alvo para identificar e analisar suas causas. Com base nessa análise, propõe um caminho para responder as questões colocadas no capítulo anterior e para construir uma solução para o problema. Termina apresentando uma lista de requisitos para a solução e a hipótese geral da tese.

O Capítulo 5 apresenta, exemplifica e analisa a solução proposta. Em especial, apresenta um conjunto de regras e heurísticas para a derivação de MD a partir do MUC. A análise da solução envolve a qualidade das especificações produzidas, a aplicabilidade da nova técnica, a sua relação com outras técnicas existentes, o grau de automatismo das regras, e a completude dos MDs delas derivados. Também são

discutidos dois trabalhos descritos na literatura, mais diretamente relacionados com a solução proposta.

O Capítulo 6 descreve os estudos experimentais realizados ao longo do trabalho, para avaliar a hipótese enunciada no Capítulo 4. Esses estudos também comparam a técnica proposta nesta tese com a técnica tradicional de UCs. São descritos três experimentos e dois estudos de caso, além de estudos preliminares.

O Capítulo 7 fecha o corpo da tese, apresentando suas conclusões, principais contribuições, limitações e trabalhos futuros.

1.5 Requisitos na Modelagem de Sistemas

Buscar uma definição simples e consensual para "requisito de sistema" (ou simplesmente, "requisito") não é tarefa fácil. A diversidade de definições indica não existir um entendimento claro, não-ambíguo, sobre esse termo (SODHI, SODHI, 2003).

Para se ter uma idéia de quão amplo pode ser o significado de "requisito", tomemos a definição de SOMMERVILLE e SAWYER (1997), segundo a qual, **requisitos** são:

- Uma especificação do que deve ser implementado;
- Descrições de como o sistema deve se comportar;
- Propriedade ou atributo que o sistema deve possuir;
- Restrição sobre o processo de desenvolvimento ou sobre o sistema.

É usual a classificação de requisitos nos níveis **de negócio**, **de usuário** ou **de sistema**. Os requisitos de sistema são ainda comumente subdivididos em **funcionais** e **não-funcionais**.

Requisitos devem ser completos, corretos, necessários, priorizáveis, não-ambíguos, verificáveis e possíveis. Entretanto, não é fácil conseguir tudo isso. Os seguintes problemas ocorrem com freqüência na modelagem de requisitos (SOMMERVILLE, SAWYER, 1997):

- Os requisitos não refletem as reais necessidades do cliente do sistema;
- São inconsistentes ou incompletos;
- Sai caro fazer mudanças nos requisitos após eles terem sido acordados;
- São mal entendidos entre os usuários, analistas e desenvolvedores.

1.6 A Engenharia de Requisitos

A **Engenharia de Requisitos** (ERq) é uma subárea da Engenharia de Software e cobre todas as atividades envolvidas com o descobrimento (elicitação), documentação e manutenção de um conjunto de requisitos, para um sistema baseado em computador (WIEGERS, 2003). Seu domínio pode ser dividido em desenvolvimento e gerência de requisitos.

O processo de desenvolvimento de requisitos é ilustrado na Figura 1.1 (ROCCO 2002, *apud* (M. KNIGHT, 2004)):

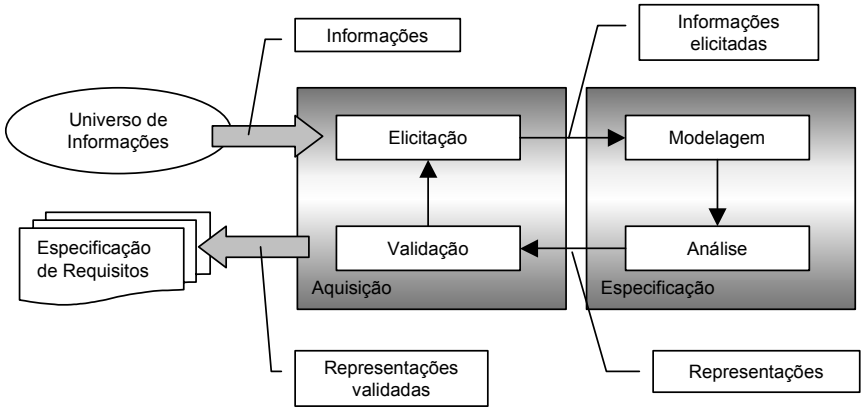


Figura 1.1 – O Desenvolvimento de Requisitos

O pessoal que participa diretamente nas atividades de desenvolvimento da ERq pode ser subdividido nas seguintes categorias, que admitem sobreposição:

- Cliente: que se beneficia direta ou indiretamente do sistema;
- Usuário: quem usa o sistema;
- Analista (de requisitos): responsável técnico;
- Desenvolvedores: responsáveis pela implementação dos requisitos no sistema.

O termo ***stakeholder*** tem sido utilizado para designar qualquer pessoa ou organismo capaz de influir direta ou indiretamente sobre os requisitos.

A atividade de **elicitação de requisitos** é responsável por juntar, negociar, e compreender os requisitos. É comumente considerada a parte mais difícil, crítica, sujeita a erros, e dependente de comunicação e colaboração, de todo o processo de software. Os meios mais freqüentemente utilizados na elicitação são: entrevista, reuniões facilitadas, protótipos e observação direta.

O resultado da elicitação deve ser representado de alguma forma. Isso é feito na atividade de **modelagem de requisitos**, onde as informações elicitadas são filtradas e utilizadas na elaboração de modelos. Inicialmente, essa modelagem se restringia a uma lista mais ou menos organizada dos requisitos do sistema. Entretanto, embora de fácil leitura e altamente flexível, essa técnica abstrai em demasia a estrutura esperada para o sistema almejado, dificultando uma verificação da sua completude e consistência internas, bem como da consistência em relação aos outros modelos subseqüentes de análise. Por isso, há vários anos, a literatura especializada tem dado destaque a outro modelo – o modelo de casos de uso (PENDER, 2003) – que captura os requisitos do sistema ao modelar o comportamento e a interação entre ele e seus atores. Além disso, vários autores têm sugerido a elaboração, em paralelo, de um modelo de (classes de) domínio, visando uma maior explicitação e compreensão dos conceitos do domínio do sistema, através de uma primeira classificação e estruturação dos mesmos.

Na atividade de **análise de requisitos**, os modelos produzidos são analisados para aperfeiçoar a classificação e a estruturação de conceitos, e para verificar a completude e a consistência interna de cada modelo, bem como a consistência entre modelos. Diferentemente da atividade de modelagem, onde se lança mão principalmente de abstração, na análise ocorre acréscimo de detalhe, muitas vezes pela reelaboração de um modelo já existente em nível de detalhamento maior, ou pela elaboração de novos modelos para capturar outras dimensões do sistema ou simplesmente para detalhar uma parte do mesmo que carece de uma melhor compreensão. São exemplos de técnicas empregadas na atividade de análise de requisitos: Cenários, Diagrama de Atividades, e Diagrama de Estados e Transições.

Por fim, na atividade de **validação de requisitos**, os modelos e especificações produzidos são discutidos pelos analistas e *stakeholders*, para verificar se eles atendem os anseios desses últimos. Um exemplo de técnica de validação de requisitos é a inspeção realizada através de reuniões colaborativas (WIEGERS, 2003).

1.7 A Pesquisa em Engenharia de Requisitos

Um levantamento dos tópicos de interesse divulgados no programa das conferências internacionais do IEEE sobre ERq (IEEE ICRE, 2007), de 2001 a 2007, evidenciou vários tópicos de pesquisa em ERq. Dentre eles, destacamos os seguintes, de interesse mais direto para esta tese.

ERq Orientada a Objetivos

A ERq Orientada a Objetivos (*Goal-oriented Requirements Engineering*) utiliza objetivos para descobrir, elaborar, estruturar, especificar, analisar, negociar, documentar e modificar requisitos (LAMSWEERDE, 2001).

Um *objetivo* (*goal*) de sistema é algo que deve ser obtido ou atingido, e pode ser formulado em diferentes níveis de abstração. Cobrem tanto características funcionais quanto não-funcionais.

Diferentemente de requisitos, um objetivo pode, em geral, requerer a colaboração de vários agentes para ser alcançado. Os objetivos devem ser refinados em sub-objetivos que possam ser atribuídos a agentes individuais, do software ou do ambiente. *Objetivos terminais* se tornam requisitos no primeiro caso, e hipóteses no segundo.

Objetivos são considerados importantes como um critério preciso de completude e pertinência de requisitos; como um mecanismo natural para estruturar requisitos; e como facilitador da detecção e resolução de conflitos entre requisitos. Têm sido apresentadas evidências de que o conceito de objetivo está entre as forças motrizes básicas para um processo de desenvolvimento sistemático de requisitos (LAMSWEERDE, 2001).

ERq com Casos de Uso

O conceito de caso de uso, originalmente elaborado por Jacobson (JACOBSON, 1987) (JACOBSON *et al.*, 1992), foi incorporado na UML - *Unified Modeling Language* (OMG, 2008) em 1996, para suprir uma lacuna até então existente neste padrão: a falta de elementos essenciais que fossem centrados no usuário. Casos de uso representam como os usuários (ou atores) interagem com o sistema (PENDER, 2003). Eles podem ser decompostos e reutilizados.

A modelagem por casos de uso inclui, além do diagrama de casos de uso, uma descrição de cada caso de uso, e a extração de cenários. Esse tipo de modelagem está focado em objetivos para evitar que os modeladores passem a considerar soluções antes de aprofundar a compreensão dos requisitos.

ERq e Cenários

Segundo LEITE *et al.* (2000), cenários descrevem situações que ocorrem no universo de discurso (domínio da aplicação). Um cenário é também comumente definido como um caminho lógico através de um caso de uso (BITTNER, SPENCE, 2002). Com cenários, é possível raciocinar e argumentar com base em exemplos ou detalhes específicos do mundo real, durante o processo cognitivo de generalização (SUTCLIFFE, 2003).

Cenários são úteis para mostrar como o sistema irá se comportar na prática. Cada cenário pode ser visto como um caso de teste, sendo assim um subsídio importante para o teste de um sistema (BITTNER, SPENCE, 2002).

Uma pesquisa (NEILL, LAPLANTE, 2003) indicou que eles são utilizados (juntamente com casos de uso) em 50% dos projetos levantados. A especificação da linguagem UML incorporou o conceito a partir da sua versão 2.0 (OMG, 2008).

ERq e Modelagem Organizacional

Casos de uso (*use cases* - UCs) focam os requisitos de um sistema, que por sua vez deve atender a requisitos mais amplos - os requisitos organizacionais. Normalmente, considera-se que a ERq deve modelar aspectos organizacionais visando entender melhor as intenções, relacionamentos e motivações entre os membros de uma organização. Algumas propostas para esse tipo de modelagem têm sido apresentadas. Um exemplo notável é o modelo i* (YU, 1997). A partir desse tipo de modelagem, pode-se compreender melhor o funcionamento do ambiente organizacional bem como as relações humanas e de trabalho entre os participantes da organização. Com essas informações, os requisitos de uma solução computacional para processos organizacionais podem ser melhor elicitados e especificados.

É interessante observar que todas essas linhas de pesquisa têm intercessão com UCs. Para a ERq Orientada a Objetivos, UCs exprimem objetivos. Cenários são caminhos dentro de um UC. E a Modelagem Organizacional pretende ser uma etapa anterior de modelagem, a partir da qual, os UCs são elicitados e construídos.

2. Casos de Uso

A técnica de *casos de uso* (*use cases* - UCs) teve sua origem com JACOBSON (1987). Desde então, ela tem sido amplamente adotada na prática da modelagem de requisitos de sistemas de software. Mesmo com essa sedimentação na prática e com reconhecidos méritos, UCs despertam muitas críticas, tanto no meio profissional quanto no acadêmico.

Este capítulo tem por objetivo fazer uma breve recapitulação de UCs e da modelagem feita a partir deles (seção 2.1 e 2.2, respectivamente), para em seguida apresentar um panorama das diversas abordagens desse tipo de modelagem, encontradas na literatura (seção 2.3). Depois, são apresentados e analisados os principais problemas e vantagens da modelagem com UCs (seção 2.4) e, por fim, algumas considerações adicionais fecham o capítulo (seção 2.5).

2.1 Definição

A especificação da OMG (*Object Management Group*) para a UML (*Unified Modeling Language*) (OMG, 2008) define "caso de uso" (*use case* - UC) como um conjunto de ações realizadas por um sistema, o qual produz um resultado observável que é, tipicamente, de valor para um ou mais atores ou outros *stakeholders* do sistema. Além disso, acrescenta que: a) cada UC especifica algum comportamento, possivelmente com variantes (incluindo comportamento de exceção e de manipulação de erro), que o sistema pode realizar em colaboração com um ou mais atores; e b) esse comportamento, envolvendo interações entre os atores e o sistema, pode resultar em mudanças no estado do sistema e comunicação com o seu ambiente. Atores são papéis desempenhados pelos usuários e outros sistemas que interagem com o sistema em questão (aquele ao qual se aplica o UC). Os atores representam entidades externas a esse sistema. Cada UC especifica uma parte da funcionalidade que o sistema fornece a seus usuários. Além disso, UCs também expressam requisitos que o sistema espera do seu ambiente, pois definem como os atores devem interagir com o sistema, de forma que esse sistema possa prover seus serviços. A execução de um UC é iniciada a partir de um estímulo proveniente de um dos seus atores. (OMG, 2008).

A lista de atores é um bom ponto de partida para a elicitação dos UCs. Para cada ator identificado, pode-se relacionar aquilo que o ator necessita obter através da

utilização do sistema. Cada UC deve representar uma tarefa cuja realização traduza um valor mensurável para o ator. Por exemplo, em um sistema bancário, o ator que desempenha o papel de cliente do banco pode desejar obter o saldo de sua conta bancária, ou fazer uma transferência de fundos entre contas. Cada um desses objetivos dá origem a um UC.

2.2 O Modelo de Casos de Uso (MUC)

Um modelo de UCs (MUC) é um modelo do sistema, definido em termos de UCs, atores, e relacionamentos entre eles. Ele contém, em geral, vários UCs, e é freqüentemente representado por um diagrama de UCs (BITTNER, SPENCE, 2002).

O diagrama de UCs (OMG, 2008) é um elemento gráfico que dá uma visão geral dos UCs e seus respectivos atores. A Figura 2.1 exemplifica esse diagrama para um sistema de biblioteca, ilustrando também três dos quatro tipos de relacionamentos normalmente considerados entre UCs.

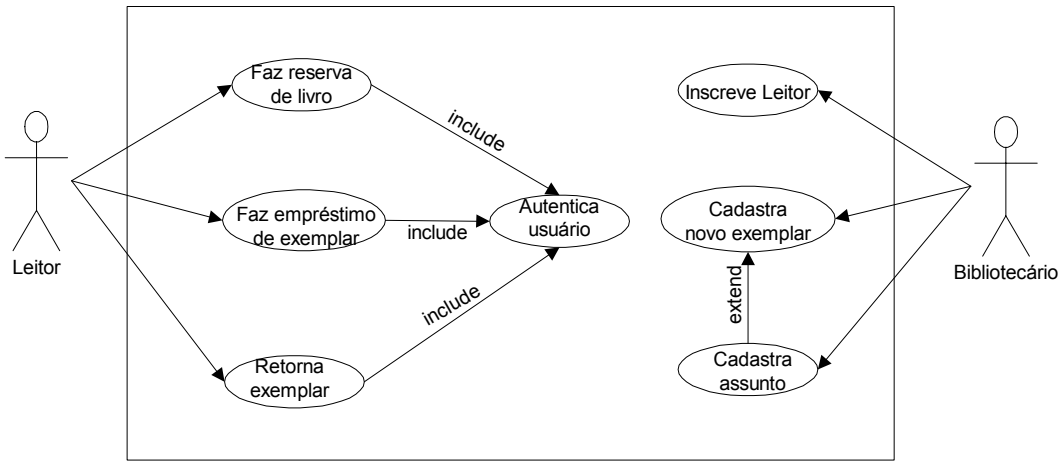


Figura 2.1 – Diagrama de UCs

O relacionamento ***use*** associa um ator e um UC. Ele representa a comunicação do ator com o UC.

O relacionamento ***include*** (inclusão) fornece uma maneira de extrair seções comuns a duas ou mais descrições de UCs, para transformá-la em um UC separado que pode ser referenciado pelos UCs de origem. Ou seja, o UC de origem deixará de ter aquela seção de descrição e, no seu lugar, conterá uma referência ao UC que passa a estar incluído. Operacionalmente, isso significa que, durante a realização do UC de origem, será inserido o comportamento especificado pelo UC incluído. Uma vez

concluída a execução do UC incluído, o controle retorna para o UC de origem, no ponto seguinte ao da chamada. O UC incluído não necessita de qualquer conhecimento sobre os UCs que o incluem.

O relacionamento ***extend*** (extensão) é utilizado em situações em que comportamento opcional ou de exceção deve ser inserido em um UC pré-existente. O propósito original da extensão foi prover um mecanismo para especificar opções que poderiam ser acrescentadas a um produto ou serviço, a partir de uma configuração básica do mesmo, como por exemplo, a contratação de um serviço de caixa postal ou de identificação de chamadas para ser acrescentado a um serviço básico de telefonia. A característica definidora de um UC de extensão é que ele não requer mudanças no UC que ele estende. Isso significa que o UC estendido deve poder ser executado sem o UC de extensão, ou seja, deve ser completo e prover valor a seus atores sem que para isso tenha de se apoiar no UC que o estende. Diferentemente do que acontece no relacionamento *include*, o UC básico não toma conhecimento da existência dos UCs que podem estendê-lo.

O relacionamento ***generalization*** (generalização) permite criar descrições genéricas de comportamento que podem ser especializadas para atender necessidades específicas. A motivação para generalizar UCs surgiu principalmente da necessidade de se descrever famílias de sistemas, nas quais diversas versões de um sistema podem ser vistas como especializações de uma única descrição geral do sistema.

Em geral se considera que os três últimos tipos de relacionamento promovem a reutilização de descrições, ao mesmo tempo em que simplificam os UCs envolvidos.

Além do diagrama de UCs, o MUC também inclui uma especificação descrevendo cada UC nele contido. Embora a descrição de um UC possa utilizar notações (semi-) formais, tais como diagramas de interação, de atividades, ou máquinas de estado, ou ainda uma combinação delas, na prática, prevalece a forma de narrativa com texto em linguagem natural, estruturado em seções. A seguir, é apresentada a descrição de um UC que adota um esquema de descrição proposto em (BITTNER, SPENCE, 2002).

CASO DE USO: Cadastra novo exemplar

Descrição

Descreve o cadastramento de um novo exemplar de livro já pertencente ao acervo da biblioteca. Esse cadastramento pode ou não decorrer da doação do exemplar por um usuário, como reposição ou compensação pela perda (extravio ou estrago) de exemplar a ele emprestado. A compensação deve ser previamente autorizada pela autoridade competente da biblioteca.

Pré-condições

Não há.

Fluxo Básico

1. O caso de uso inicia quando o ator Staff indica o objetivo de cadastrar um novo exemplar de livro já pertencente ao acervo da biblioteca.
2. O Sistema solicita e o ator Staff informa:

   - **identificador do livro**;

   O ator Staff deve indicar ainda se se trata de reposição/compensação por exemplar perdido, caso em que deve informar:

   - **identificador do usuário** que está fazendo a doação de exemplar para repor/compensar a perda. Deve corresponder a um usuário que teve exemplar extraviado ou estragado enquanto emprestado a ele, ainda não compensado ou ressarcido;
   - **identificador do exemplar** que está sendo reposto/compensado pela doação. Deve corresponder a um exemplar que tenha sido baixado por extravio ou estrago na vigência do um empréstimo ao usuário. Caso o livro cujo exemplar está sendo cadastrado tenha um exemplar baixado por extravio ou estrago na vigência de empréstimo pelo usuário, trata-se de uma reposição, e este identificador será, obrigatoriamente, o identificador a ser informado aqui. Caso contrário, trata-se de uma compensação.
3. Após confirmação do ator Staff, o Sistema armazena as informações acima, registra o novo exemplar do livro com a data do cadastramento (data corrente), gerando um identificador para o exemplar. O exemplar passa a estar disponível para empréstimo. No caso de reposição/compensação, a data do cadastramento deve ser posterior à data da baixa do exemplar que está sendo reposto/compensado, e o Sistema deve rever a **situação do usuário** (de suspenso para regular), desde que as demais condições para isso estejam satisfeitas (nenhum outro livro perdido por ele e ainda não compensado, reposto ou ressarcido; sem débito de multas; sem devolução atrasada em mais de 5 dias).

   {Verifica reserva}
4. O sistema verifica que inexiste **reserva ativa** *aguardando retorno* para o livro correspondente ao exemplar retornado.
5. O caso de uso termina.

Fluxos Alternativos

A1. ***Trata existência de reserva ativa***

Em {**Verifica reserva**} se existir **reserva ativa** aguardando retorno para o livro correspondente ao exemplar retornado

1. O sistema coloca o exemplar na situação *disponível com reserva*, e alerta o ator Staff da existência da reserva para que ele coloque o exemplar na prateleira de livros reservados para empréstimo. Em seguida, notifica, por e-mail, o retorno do exemplar ao próximo Usuário da lista de reservas do livro (usuário regular com reserva ativa mais antiga *aguardando retorno* e, portanto, ainda não notificado), passando a reserva correspondente para a situação *aguardando retirada*. A notificação contém pelo menos as seguintes informações:
   - **identificador do livro**;
   - título do livro;
   - autor principal do livro (primeiro autor, pela ordem de apresentação no livro);
   - data limite para a retirada do livro (é a data do próximo dia útil seguinte a data do retorno do livro).
2. O caso de uso retoma o passo seguinte ao da interrupção.

Sub-fluxos

Não há.

Pós-condições

Não há.

Pontos de Extensão Públicos

Não há.

Requisitos Especiais

Não há.

Cenários

Cenário "Livro sem reserva de empréstimo". Fluxos: básico.

Cenário "Livro com reserva de empréstimo". Fluxos: básico, "Trata existência de reserva ativa".

Informações Adicionais

Não há.

Figura 2.2 – Exemplo de descrição de um UC

2.3 Abordagens de Modelagem com Casos de Uso

Muitos são os formatos distintos já empregados na descrição de UCs. COCKBURN (2005) contou 18 (dezoito). Eles variam desde a versão textual livre em linguagem natural, passando por formatos estruturados (BITTNER, SPENCE, 2002), até propostas mais formais que sacrificam a facilidade de leitura pela precisão na comunicação do seu conteúdo (menos comuns) (OMG, 2008).

As abordagens formais são menos empregadas, pois os UCs devem ser utilizados na comunicação com os *stakeholders*. A linguagem natural propicia um entendimento universal e a contribuição dos *stakeholders* na concepção e validação dos UCs. Nessa linha, pode-se utilizar formatos mais ou menos estruturados, dependendo principalmente do estágio em que se encontra a definição dos requisitos. Inicialmente, um formato menos estruturado pode facilitar o emprego de UCs na representação ligeira de requisitos, elaborada em paralelo com o processo de elicitação; depois, uma forma estruturada talvez seja mais adequada como instrumento de modelagem apurada, análise e documentação dos requisitos obtidos.

Mas as propostas existentes de MUCs não variam apenas no formato. Elas se diferenciam também quanto ao conteúdo e nível de abstração da descrição dos UCs.

De imediato, uma variação de conteúdo aparece nas formas estruturadas de descrição, onde elementos que compõem a estrutura de algumas propostas de UCs estão ausentes em outras.

Outra variação de conteúdo, menos facilmente percebida, decorre dos diferentes níveis de abstração adotados na descrição do principal elemento dos UCs: os fluxos de comportamento. São, principalmente, variações ao longo de duas dimensões descritivas: a do comportamento e a da informação.

Todas as propostas de UCs colocam em primeiro plano o comportamento, embora variando em grau de detalhamento. Existem propostas de UCs que detalham todo o comportamento, interno e externo, enquanto outras se limitam a registrar o comportamento externo, ou seja, o comportamento observável do ponto de vista dos usuários que interagem com o sistema. Embora a primeira categoria de propostas seja a mais popular, a segunda categoria tem um representante de destaque – os Casos de Uso Essenciais (CONSTANTINE, 2000) (BIDDLE, NOBLE, 2002).

A maioria das propostas de UCs dá pouca ou nenhuma atenção à informação trocada entre os atores e o sistema, conforme observa LAUESEN (2002, pp. 100, 132, 432) e HAY (2003, pp. 43, 185, 240), entre outros. Nessa categoria estão propostas como as contidas em (JACOBSON *et al.*, 1992) (JACOBSON *et al.*, 1994) (ROBERTSON, ROBERTSON, 1999) (JACOBSON, NG, 2004) (COCKBURN, 2005) (OMG, 2008) e (ROBERTSON, ROBERTSON, 2006). Dentre as propostas que enfatizam a importância do detalhamento da informação, se destaca a proposta contida em (BITTNER, SPENCE, 2002)[3].

Também variam os critérios adotados para a escolha dos UCs. Um dos critérios mais comumente recomendados para isso é o UC ter um valor mensurável para algum stakeholder (adotado, por exemplo, em (JACOBSON *et al.*, 1994) (BITTNER, SPENCE, 2002) (JACOBSON, 2004) (JACOBSON, NG, 2004) e (OMG, 2008)). Com base na Análise Essencial (MCMENAMIM, PALMER, 1991), outro critério é sugerido em (ROBERTSON, ROBERTSON, 1999) (ROBERTSON, ROBERTSON, 2006): cada UC deve corresponder a uma atividade essencial. Na mesma linha segue HAY (2003), mas enquanto os ROBERTSON (1999, pp. 76) permitem a decomposição dos UCs do nível essencial em outros de nível mais baixo, HAY (2003, pp. 262) afirma que os analistas de requisitos só devem se interessar pelos UCs do nível essencial e de níveis superiores.

A partir da análise da literatura levantada, principalmente do artigo "*Use cases - Yesterday, today and tomorrow*", em que JACOBSON (2004) faz um balanço dos últimos 18 anos dessa técnica e propõe algumas mudanças, pudemos identificar algumas tendências na evolução da MUC:

- Maior detalhamento da informação trocada entre o sistema e seu ambiente, em cada UC. A literatura mais antiga aponta a falta desse detalhamento como uma falha dos UCs, enquanto a mais recente tem proposto medidas que aliviam o problema. Embora muitos ainda se limitem a recomendar a elaboração de um modelo (de objetos ou de dados) do domínio, BITTNER e SPENCE (2002) preconizam, com o aval do próprio JACOBSON (2004), o detalhamento completo da informação.
- Fixação de um nível mínimo para os UCs. Os critérios acima mencionados para a escolha dos UCs, juntamente com o critério onipresente de "valor do UC",

[3] Endossada por Jacobson (2004).

apontam para a adoção implícita do nível de atividades essenciais da Análise Essencial, como nível mínimo a considerar nessa elicitação. Mais recentemente, JACOBSON (2004) propôs que UCs utilizados em relacionamentos *extend* e *include* com outros UCs, e que não possam ser instanciados separadamente, passem a ser considerados fragmentos de UCs, e não mais UCs, o que reforça essa impressão. De certa forma, Jacobson está propondo um nível mínimo de abstração para os UCs, mesmo que o faça de forma indireta e subjetiva.

2.4 Análise da Modelagem de Requisitos com UCs

Nesta seção, analisaremos a MUC no contexto da modelagem de software segundo o paradigma da orientação a objetos, por ser esse um paradigma muito popular nos dias atuais, para o desenvolvimento de software. Com base em um levantamento feito na literatura, a seção seguinte (2.4.1) apresenta e discute as principais vantagens da MUC, enquanto a seção 2.4.2 faz o mesmo com relação às desvantagens dessa técnica de modelagem.

2.4.1 Vantagens da MUC

Uma prática comum de modelagem de requisitos, principalmente antes do advento dos UCs, consistia em listar os requisitos ou *features* (LAUESEN, 2002) a serem implementados pelo sistema (ROBERTSON, ROBERTSON, 1999) (ROBERTSON, ROBERTSON, 2006). Uma *feature* pode ser definida como uma afirmação de alto-nível sobre uma capacidade desejada para o sistema, em termos de funcionalidade ou algum atributo de qualidade (performance, segurança, etc.) (JACOBSON, NG, 2004). Por exemplo, a *Lista de Requisitos* (LRqs) para um sistema de hotelaria poderia incluir estes dois requisitos: (1) o sistema deverá ser capaz de colocar um quarto indisponível em um determinado período, para a realização de reparos; (2) o sistema deverá ser capaz de reservar quartos pelo seu tipo.

A literatura aponta pelo menos três vantagens da modelagem de requisitos com UCs (MUC) em relação à LRqs, relacionadas e discutidas a seguir:

1. Facilita a verificação de completude e consistência da especificação de requisitos;
2. Reduz o risco do sistema modelado não atender as reais necessidades dos *stakeholders* (LARMAN, 2004);

3. Facilita a rastreabilidade de requisitos entre os artefatos produzidos ao longo do desenvolvimento (KULAK, GUINEY, 2003).

A Figura 2.3 mostra as vantagens listadas anteriormente e a relação delas com algumas características também apontadas na MUC, servindo de pano de fundo para a análise que se segue.

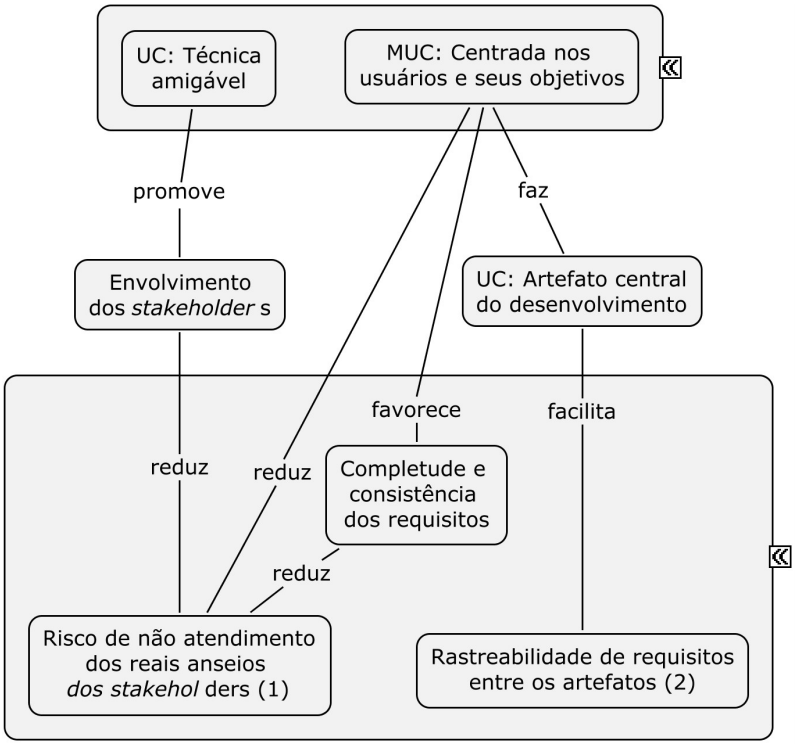


Figura 2.3 – Características da MUC e vantagens em relação à LRqs

Existem diversos fatores de qualidade que devem estar presentes em uma especificação: completude, consistência, precisão, legibilidade, manutenibilidade, concisão, compreensibilidade, testabilidade, etc. Desses, a utilização de UCs em substituição (ou acréscimo) à Lista de Requisitos (LRqs) impacta mais diretamente a consistência, completude e rastreabilidade de requisitos, conforme indicado na Figura 2.3.

Uma especificação incompleta pode resultar em um sistema incapaz de atender todas as necessidades dos seus *stakeholders*, por não implementar algumas funções imprescindíveis a eles. Se uma especificação de requisitos é inconsistente, então ela possui pelo menos dois requisitos que não podem ser satisfeitos ao mesmo tempo, qualquer que seja a implementação do sistema. Trata-se de uma falha da especificação que, se não resolvida, pode igualmente impedir o sistema de satisfazer seus *stakeholders*.

No caso de uma LRqs extensa, é difícil avaliar se ela é consistente e completa. Também, é difícil para os *stakeholders* se certificarem de que seus objetivos de negócio serão efetivamente apoiados pelo sistema especificado na LRqs (LAUESEN, 2002). A raiz dessas dificuldades está na estruturação inexistente ou inadequada que esta abordagem de especificação promove. Diferentemente, na modelagem com UCs (MUC) os requisitos são estruturados segundo os usuários (atores) do futuro sistema, e principalmente, em torno dos objetivos desses atores e demais *stakeholders*. Cada UC mostra como o sistema deverá interagir com os atores para alcançar um objetivo de algum *stakeholder*. Enquanto que para a LRqs, o fundamental é responder "o que os usuários desejam que o sistema faça", para a MUC é "quem vai usar o sistema, quais são seus objetivos ao fazer isso, e quais os cenários típicos dessa utilização" (WIEGERS, 2003) (LARMAN, 2004). A visibilidade dada aos objetivos dos *stakeholders*, e a estruturação dos requisitos segundo esses objetivos, contribuem para facilitar a verificação da consistência e completude da especificação, e a sua validação pelos *stakeholders*. Em conseqüência, essa característica especial da MUC também ajuda a reduzir o risco do sistema resultante não atender adequadamente as reais necessidades dos *stakeholders*.

Outra característica da MUC também é fundamental para propiciar a redução do risco do sistema especificado não satisfazer os *stakeholders*: o sentimento de familiaridade que ela desperta neles. Eles não se sentem intimidados ao serem apresentados à técnica de UCs (LARMAN, 2004). Isso ocorre por dois motivos principais. Primeiro, porque essa técnica é, na essência, um relato de estórias ou cenários de utilização do sistema pelos usuários (atores) que com ele interagem, e como tal, é percebida como simples e familiar por eles (KULAK, GUINEY, 2003). Além disso, ela emprega, pelo menos na forma mais comum, a linguagem natural para a descrição dos cenários (casos) de uso. Nas fases iniciais da modelagem, essa linguagem é utilizada sem maiores restrições ou elementos estruturantes e, portanto, não há necessidade de um treinamento extenso para capacitar os *stakeholders* a ler, ou mesmo escrever, versões iniciais (*outlines*) dos UCs. Isso tudo facilita e estimula o envolvimento dos mesmos na definição e revisão dos requisitos, o que ajuda a garantir a sintonia do sistema especificado com os anseios dos *stakeholders*.

A estruturação baseada nas necessidades dos *stakeholders* também tem efeitos benéficos sobre a rastreabilidade de requisitos entre os diversos artefatos produzidos ao

longo do ciclo de desenvolvimento de um sistema. Rastreabilidade de requisitos é a capacidade de se descrever e acompanhar a vida de um requisito, tanto na direção da sua implementação no sistema (*forward traceability*), quanto na direção de sua origem (*backward traceability*) (PINHEIRO, 2004). Segundo KULAK, GUINEY (2003), UCs são rastreáveis, pois as estórias contidas neles são mapeadas naturalmente em artefatos de análise, projeto e testes, enquanto que a LRqs não tem rastreabilidade nesses artefatos. Na nossa visão, o foco nos *stakeholders* e em seus objetivos, promovido pelos UCs, fez com que o modelo de UCs se tornasse o artefato central do ciclo de desenvolvimento de um sistema, referenciado e utilizado em todas as fases desse ciclo (Figura 2.4, traduzida de (JACOBSON, 2004)).

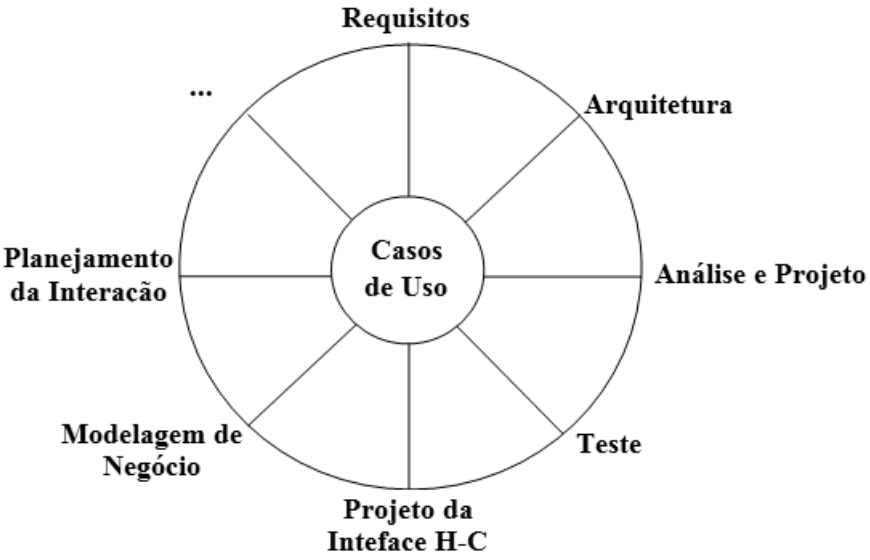


Figura 2.4 – UC como artefato central para o desenvolvimento

Durante as fases de análise e projeto do sistema, cada UC tem o comportamento modelado através da colaboração entre classes de objetos. Na fase de testes, cada UC resulta em um conjunto de casos de teste. Os UCs também são importantes para o projeto da interface com o usuário e na estruturação do manual do usuário. Por fim, eles também se inserem na modelagem de negócio, uma vez que são capazes de capturar processos de negócio (JACOBSON, 2004). Portanto, diferentemente do que acontece quando os requisitos são modelados na forma de uma LRqs, o artefato central do desenvolvimento é o caso de uso, e não um artefato de análise ou de projeto. Essa "promoção" do artefato de requisitos significa o encurtamento da distância representacional entre ele e os demais artefatos produzidos nas fases posteriores do desenvolvimento. Embora isso não dispense o tratamento específico usualmente dado

para se garantir uma adequada produção, captura e extração de rastros (PINHEIRO, 2004), o rastreamento de requisitos entre os artefatos tende a ficar facilitado.

A despeito das vantagens acima apresentadas da modelagem de requisitos utilizando UCs, é preciso reconhecer que, muitas vezes, os desenvolvedores (por exemplo, em fábricas de software) recebem uma LRqs pronta e, antes de mais nada, precisam validá-la. Embora essa validação possa ser feita elaborando-se o modelo de UCs do sistema pretendido, isso é custoso, principalmente se a qualidade da LRqs for baixa. Nesse caso, é preciso uma validação preliminar e expedita dessa lista, que pode ser feita, por exemplo, através de um processo de revisão ou inspeção (HE *et al.*, 2008). A partir daí, durante a modelagem com UCs essa validação é retomada, e de forma mais efetiva, pois agora a compreensão do problema é maior, tanto por parte dos desenvolvedores quanto dos *stakeholders*.

2.4.2 Problemas da MUC

Alguns problemas da modelagem de requisitos com UCs (MUC) têm sido apontados freqüentemente na literatura, dentre os quais destacamos:

- **Indução à decomposição funcional**. Muito facilmente, os modeladores são levados a tratar os UCs como se fossem funções e a decompô-los em outros de nível mais baixo.
- **Interferência no projeto da interface H-C**. Muitos modeladores descrevem UCs comprometidos com uma interface H-C específica.
- **Diversidade de abordagens para a MUC**. Existem muitas abordagens distintas para a modelagem com UCs, algumas delas conflitantes entre si.
- **Transição difícil para o modelo de objetos do domínio (MD) do sistema**. O MUC tem se mostrado um ponto de partida fraco para a obtenção do MD do sistema.

Os três primeiros problemas são detalhados a seguir, enquanto que o último, pela sua importância para esta tese, será discutido em profundidade no próximo capítulo.

Indução à decomposição funcional.

A decomposição funcional de um sistema é inerente à modelagem com UCs. Essa modelagem envolve pelo menos dois níveis de decomposição funcional. No primeiro nível, o sistema é decomposto em um conjunto de UCs, onde cada UC abstrai

um processo subjacente que realiza a funcionalidade do UC. No segundo nível, cada UC é decomposto na seqüência de ações que expressam o comportamento do sistema e dos atores que com ele interagem. Essas ações também são abstrações de (sub) processos, cada qual realizando uma parte da funcionalidade do UC.

Muitos autores alertam para os perigos da decomposição funcional na modelagem com UCs (BITTNER, SPENCE 2002) (ANDERSON, 1999a, 1999b) (JACOBSON, 2004). Certamente, eles não estão se referindo à decomposição mencionada no parágrafo anterior, intrínseca aos UCs. Eles se referem a uma decomposição funcional recursiva, em vários níveis adicionais, comum na modelagem com UCs, e cujo principal indutor é o relacionamento *include*[4]. Por isso, eles recomendam uma disciplina rígida no uso desse relacionamento (BITTNER, SPENCE, 2002), ou mesmo, a sua completa desconsideração (KULAK, GUINEY, 2003) (ROBERTSON, ROBERTSON, 2006).

O efeito negativo da decomposição funcional recursiva de UCs está na geração de UCs que não preenchem o atributo-chave de possuir valor real para, pelo menos, um dos *stakeholders* do sistema (BITTNER, SPENCE, 2002) (JACOBSON, 2004), dificultando a percepção dos *stakeholders* quanto ao que o sistema faz, e conseqüentemente, do seu valor para eles. Outro efeito desfavorável está na consideração prematura de aspectos de projeto do sistema.

Interferência no projeto da interface H-C

Durante a fase de requisitos, em algumas situações pode ser útil construir um protótipo (de parte) do sistema a desenvolver, ocasião em que questões de interface H-C são consideradas. No entanto, no caso geral e no contexto de sistemas de informação, deve-se evitar considerar aspectos do projeto da interface H-C durante a definição dos requisitos do sistema. Por isso, vários autores têm chamado a atenção para a intromissão em potencial que a MUC representa para o projeto da interface H-C. Por exemplo, LAUESEN (2002) afirma que a maioria dos tipos de UCs especifica parte do projeto da interface H-C, sendo, portanto, inadequados para especificar requisitos no nível do domínio da aplicação. Outros autores enfatizam a necessidade de se evitar detalhar aspectos de projeto e implementação da interface H-C nos UCs (ANDERSON, 1999a, 1999b) (DOBING, PARSONS, 2000) (BITTNER, SPENCE, 2002). CONSTANTINE

[4] O relacionamento *extend* também gera decomposição (JACOBSON, 2004, pp. 218)

(2000) considera que o “caso de uso de Jacobson” (JACOBSON *et al.*, 1992) usualmente assume uma interface de usuário em particular. Para ele, é importante expressar requisitos independentemente da sua implementação e realização em uma forma particular de IU (CONSTANTINE, 2001). Conseqüentemente, Constantine e seus seguidores propõem uma modalidade de UC livre desse tipo de comprometimento, denominada **caso de uso essencial** (BIDDLE, NOBLE, 2002).

Diversidade de abordagens para a MUC.

Conforme mostrou a seção 2.3, a literatura técnica e profissional sobre UCs apresenta diversos enfoques e propostas diferentes, e algumas vezes conflitantes, para a modelagem de requisitos de sistemas com UCs. Portanto, o leque de opções para o analista é grande, e isso se torna um problema na medida em que falta orientação clara sobre quais as vantagens e desvantagens de cada abordagem, bem como critérios para a escolha da abordagem mais adequada a uma dada situação de modelagem em particular.

2.5 Considerações Finais

Este capítulo fez um breve resumo da modelagem de requisitos de sistemas com UCs (MUC), para em seguida apresentar e analisar algumas de suas vantagens e problemas, apontados na literatura.

A técnica de UCs é muito flexível, podendo ser aplicada aos mais diversos tipos de sistemas. Além disso, ela veio ao encontro da necessidade de fazer dos *stakeholders* de um sistema a desenvolver, co-participantes mais ativos do processo de definição dos requisitos do sistema.

Pode-se dizer também que ela contém uma certa dose de subjetividade. Por exemplo, o critério de "todo UC ter valor para algum stakeholder", utilizado para a escolha dos UCs de um sistema, é considerado excessivamente subjetivo (o que é "ter valor"?).

Parte da popularidade da técnica de UCs talvez se deva, exatamente, a essa combinação de flexibilidade e subjetividade, pois cada autor se sente livre para adaptá-la a seu gosto. Prova disso são as diversas abordagens para a modelagem com UCs, ligeiramente distintas entre si, publicadas na literatura. Mas essas características dos UCs também parecem estar na raíz de alguns dos problemas apontados na técnica, como por exemplo, interferências indevidas no projeto da interface H-C e a fraca contribuição

para a a obtenção, a partir do MUC, de um MD consistente e completo (em relação ao MUC).

Os próximos dois capítulos (3 e 4) aprofundam essas questões, o que permitiu, no capítulo 5, a proposição de uma especialização do UC, equipada para resolver alguns dos seus problemas, e obtida trilhando exatamente o caminho sugerido acima, ou seja, menor flexibilidade e maior precisão.

3. Os Problemas-Alvo

3.1 Introdução

O contexto considerado nesta tese é o do desenvolvimento de sistemas de informação, segundo o paradigma da orientação a objetos. Nesse contexto, o modelo de UCs (MUC) e o modelo de (classes de objetos) de domínio (MD) são largamente utilizados.

Em nossa experiência anterior na utilização desses modelos, pudemos perceber algumas dificuldades associadas à modelagem de requisitos com UCs, bem como dificuldades para utilização conjunta de ambos os modelos, de forma complementar, na fase de (elicitação e modelagem de) requisitos. Isso motivou uma pesquisa de trabalhos publicados na literatura especializada, buscando verificar se as mesmas dificuldades eram reportadas por outros autores. O resultado dessa primeira pesquisa está resumido no capítulo anterior (seção 2.4.2).

Dentre os problemas listados na seção 2.4.2, escolhemos aquele designado por "transição difícil (do MUC) para o MD do sistema" como alvo da pesquisa nesta tese, pelos seguintes motivos:

1. Dentre os problemas levantados na MUC, esse é um dos mais evidentes a partir da literatura que trata desses modelos[5];
2. Qualquer avanço que se puder obter na utilização conjunta dos dois modelos será relevante, dada a popularidade do modelo de UCs na fase de requisitos (NEILL, LAPLANTE, 2003) (KULAK, GUINEY, 2003), e um consenso cada vez maior de que o MD é um complemento importante para o MUC, nessa fase.

O restante deste capítulo tenta caracterizar melhor o problema considerado nesta tese. Primeiro, são consideradas duas questões relacionadas à utilização dos dois modelos na fase de requisitos (seção 3.2). Em seguida, descrevemos um levantamento abrangente realizado na literatura científica e técnica, sobre os modelos. São apresentadas e comentadas algumas propostas que, a nosso ver, fornecem um panorama representativo da pesquisa e da técnica de utilização desses dois modelos (seção 3.3). Por fim, e com base no que foi levantado, selecionamos algumas questões de investigação e detalhamos o objetivo a ser buscado nesta tese (seção 3.4).

[5] Assim como o problema da diversidade de abordagens para a MUC, que também é bastante evidente.

3.2 O MUC e o MD na Fase de Requisitos

Nesta seção vamos analisar duas questões pertinentes à utilização do MUC e do MD. A primeira delas é: "por que utilizar ambos os modelos na fase de requisitos?". Ao considerar essa questão, pretendemos justificar a prática do uso conjunto dos modelos na fase de requisitos. A segunda questão é: "qual deve ser a ordem de elaboração dos modelos?" A resposta é difícil e existem argumentos na literatura para se decidir por qualquer uma das duas possibilidades. Entretanto, como alguns desses argumentos conflitam entre si, a proposta de se fazer ambos em paralelo surge fortalecida. Passamos agora às questões e respectivas análises.

3.2.1 Por que Utilizar o MUC e o MD na Fase de Requisitos?

Nós compartilhamos com GLINZ (2000) e outros autores (por exemplo, (KÖSTERS *et al.*, 2001)) a visão de que os dois modelos se complementam mutuamente. Isso porque, cenários modelam o *comportamento dinâmico, externamente observável* de um sistema, enquanto que o modelo de classes especifica a *estrutura e comportamento internos*, isto é, estruturas de dados, relacionamentos entre elas, e operações, necessários à determinação do comportamento externo do sistema. Segundo GLINZ (2000), quase todas as abordagens práticas que fazem uso de cenários combinam os cenários com um modelo de classes.

Também a literatura técnica apresenta indícios dessa complementaridade mútua dos dois modelos. Por exemplo:

- JACOBSON *et al.* (1992) afirmam que o MD serve de suporte para o desenvolvimento do modelo de requisitos, sendo uma de suas partes (juntamente com o MUC). Segundo eles, o MD ajuda a desenvolver uma terminologia comum a ser utilizada nos UCs.
- JACOBSON *et al.* (1994) afirmam que, para se obter um bom resultado, é necessário trabalhar interativamente entre os dois modelos.
- LAUESEN (2002) dá a entender a importância de se modelar a informação estática do domínio do sistema desde os primeiros momentos do levantamento de requisitos, ao afirmar que a análise da informação do domínio produz um modelo que pode sobreviver à implementação do sistema com poucas modificações, e que o mesmo não se aplica à modelagem das funções do sistema.

- BITTNER, SPENCE (2002) afirmam que o modelo de domínio só deve ser criado se existirem relacionamentos relevantes entre os conceitos descritos no glossário. Em nossa opinião, este é o caso para qualquer sistema não trivial. Portanto, os autores confirmam, mesmo que indiretamente, a necessidade de se elaborar o modelo de domínio em paralelo com o modelo de UCs.
- Para HAY (2003), regras de negócio são sobre dados; elas restringem que dados podem ou devem ser criados no curso de uma operação de negócio. A descoberta e a definição de regras de negócio devem ser realizadas em conjunção com o desenvolvimento de um modelo de dados.
- ROBERTSON, ROBERTSON (2006) consideram um papel exploratório para o modelo de dados, desde o início da atividade de levantamento de requisitos.

Portanto, podemos concluir ser uma prática recomendável a elaboração de um modelo de domínio, em paralelo com a modelagem de requisitos com UCs.

3.2.2 Qual deve ser a ordem de elaboração dos modelos?

A tendência atual, capitaneada pelos métodos baseados na UML (por exemplo, RUP (JACOBSON *et al.*, 1999)), dá precedência ao MUC em relação ao MD (SHOVAL *et al.*, 2006). Isso é compreensível a partir das seguintes constatações:

- O MUC e, em especial, os cenários dele derivados, são normalmente considerados mais "amigáveis" do que o MD, ou seja, são de mais fácil leitura e compreensão pelos *stakeholders*.
- O MUC tem maior escopo de modelagem do que o MD, já que ele, inerentemente, modela o comportamento do ambiente do sistema (representado pelos seus atores), além do comportamento do próprio sistema, enquanto que o MD correspondente tem o escopo de modelagem normalmente restrito ao sistema, detalhando elementos internos do mesmo.

O experimento apresentado em (SHOVAL *et al.*, 2006) resultou em diagramas de classes de melhor qualidade quando eles são feitos antes dos UCs, pelo menos nas condições do experimento (modelagem de um sistema de informação, dentre outras). A hipótese levantada pelos autores para explicar isso é que a criação dos UCs, antes da criação do diagrama de classes, é mais difícil por dois motivos:

1. Identificar funções é mais complexo: elas não são "concretas", "tangíveis" e "estáveis" e, em geral, muitos são os UCs a modelar. Em contraste, modelos

conceituais de dados consistem de objetos mais concretos, tangíveis e estáveis, mais fáceis de identificar e definir; existem apenas alguns poucos construtores (basicamente classes, atributos e relacionamentos) e, em geral, há apenas um diagrama a criar.

2. Os analistas são obrigados a considerar, ao mesmo tempo, aspectos funcionais e estruturais (descrições de UCs incluem todos os tipos de interações). Já se começarem pela modelagem conceitual de dados, eles podem se limitar a aspectos relativos aos dados (classes); depois então, ao fazerem a modelagem funcional, as classes já estarão definidas e, portanto, as tarefas mais complexas se tornam mais fáceis.

Outra estratégia seria desenvolver ambos os modelos em paralelo, de forma interativa. Na literatura existem propostas de elaboração dos dois modelos em paralelo (JACOBSON *et al.*, 1994) (GLINZ, 2000) (CONSTANTINE, 2001). GLINZ (2000), por exemplo, descreve uma abordagem para elaboração dos dois modelos em paralelo, com foco na consistência entre eles. Uma curiosidade: em 1987, quando Jacobson propôs inicialmente o modelo de UCs, segundo ele próprio (JACOBSON, 2004) esse modelo incluía objetos de domínio (*entity domain objects*), para mostrar como os UCs podiam acessá-los. E esses objetos possuíam dados e operações.

3.3 O Estado da Arte e da Prática da Utilização dos Modelos

Nesta seção, procuramos dar um panorama do estado da arte e da prática da utilização do MUC e do MD, na fase de requisitos. Inicialmente (seção 3.3.1), descrevemos a metodologia utilizada no levantamento da literatura. Em seguida, apresentamos e comentamos algumas propostas contidas em artigos (seção 3.3.2), e em livros (seção 3.3.3), que respectivamente e em conjunto, consideramos representativos do estado da arte e da prática da utilização dos modelos.

3.3.1 Metodologia de Levantamento da Literatura

Os levantamentos de material de interesse na literatura foram feitos de forma criteriosa (conforme descrito a seguir), visando, entre outras coisas, possibilitar um julgamento mais claro sobre a abrangência dos mesmos. O resultado, ao longo de dois anos, foi a coleta de mais de 2000 referências (e em muitos casos, do texto completo) de artigos, livros, relatórios técnicos, teses, dissertações, apresentações e notas de aulas, em ERq.

As fontes levantadas incluíram eventos (basicamente, conferências e *workshops*), periódicos, editores de livros, *home-pages* de pesquisadores, e diretórios e serviços de busca na *Web*. A seguir, detalhamos as principais fontes utilizadas nos levantamentos.

Eventos. Foram feitas várias buscas por eventos (conferência, *workshops*, etc.) que divulgam trabalhos em Engenharia de Requisitos (ERq), através dos seguintes diretórios disponíveis na *Web*: Google, *IEEE Conferences*, ACM, IFIP, *DB and LP Conferences and Workshops*, *All Conferences Directory*, e Qualis (CAPES). Dos 60 eventos examinados, 35 foram selecionados com base na sua classificação CAPES/Qualis (QUALIS, 2008), para o levantamento periódico de artigos de interesse. Vários desses eventos são conferências que promovem diversos *workshops*, onde são publicados trabalhos em ERq. Esses *workshops* também são considerados nos levantamentos.

Periódicos. A busca por periódicos que publicam trabalhos em ERq foi empreendida a partir dos *sites* de grandes editores de periódicos científicos em computação: *IEEE-Computer Society*, *ACM*, *Springer*, *Elsevier*, *John Wiley*, *IEE*, e *World Scientific*. Além disso, utilizamos também o diretório *Google* e a lista de publicações de pesquisadores em ERq, disponíveis nas respectivas *home-pages*. Para o acompanhamento através de levantamentos realizados freqüentemente, foram selecionados 27 periódicos com base na especificidade do periódico em Engenharia de Requisitos, classificação CAPES/Qualis (QUALIS, 2008) e fator de impacto (CITESEER, 2003).

Pesquisadores. Outra fonte de pesquisa utilizada nos levantamentos foi a *home-page* de pesquisadores da Engenharia de Requisitos e áreas afins. Um dos objetivos foi conhecer o trabalho dos principais pesquisadores em ERq. Além disso, vários recursos adicionais estão disponíveis a partir dessas *home-pages*, tais como versões preliminares (livres) de artigos, notas de aulas, orientações e comentários. A seleção dos pesquisadores foi feita a partir das listas de membros dos comitês de programa dos seguintes eventos: RE 2005/2006 (*Intl. Requirements Engineering Conference*), WER 2005/2006 (*Workshop* em Engenharia de Requisitos), CERE 2005/2006 (*Comparative Evaluation in Requirements Engineering*), REFSQ 2005/2006 (*Intl. Working Conference on Requirements Engineering: Foundation for Software Quality*), e REET 2005 (*Intl. Workshop on Requirements Engineering Education and Training*). Também foram considerados os pesquisadores pertencentes ao quadro editorial do *Requirements*

Engineering Journal (REJ). Ao longo de 2 anos foram selecionados 31 pesquisadores de um total de 161 levantados, com base, principalmente, na sua produção científica em Engenharia de Requisitos. A visita às *home-pages* desses pesquisadores é feita periodicamente, para atualização do levantamento dos recursos lá disponíveis.

Livros. Procuramos identificar um conjunto de livros sobre ERq, com relevância para a pesquisa que estamos desenvolvendo. Para isso, procuramos indicações bibliográficas nas *home-pages* dos 31 pesquisadores previamente selecionados (a maioria deles também professores), fizemos buscas no *site* da *Amazon Bookstore*, e seguimos as indicações e comentários disponíveis nesse *site* (do tipo: "quem compra esse livro também compra..."). Portanto, acreditamos que os livros levantados guardam sintonia com o que está sendo utilizado em diversos cursos de ERq pelo mundo afora, ao mesmo tempo em que refletem o que há de mais novo no mercado editorial de literatura técnica nessa área. De um total de 30 livros inicialmente selecionados, 12 foram adquiridos ou obtidos em bibliotecas. Os critérios utilizados para essa escolha incluíram: a) especificidade do livro em Engenharia de Requisitos; b) indicação do livro por parte dos pesquisadores monitorados; c) número de revisões e grau de excelência (número de estrelas) atribuído pelos revisores, no site da *Amazon*; e d) *ranking* de venda do livro na *Amazon*. Mais recentemente, obtivemos o acesso *online* a vários outros livros, através do serviço *SAFARI* (O'REILLY, 2008).

A despeito das limitações inerentes a se tomar apenas uma amostra do vasto material disponível na literatura sobre ERq, acreditamos que os recursos obtidos com os levantamentos realizados dão uma visão representativa do estado da arte e da prática da ERq, e em especial, da utilização do MUC e do MD no contexto da modelagem de requisitos.

3.3.2 Propostas da Literatura Científica

Nesta seção, são revistos alguns trabalhos científicos que tratam da utilização do MUC e do MD no processo de desenvolvimento de software, e em especial, na elicitação e modelagem de requisitos. Os trabalhos foram selecionados dentre aqueles levantados segundo a metodologia descrita na seção anterior. A nosso ver, esses trabalhos dão uma visão representativa da pesquisa sobre a utilização dos dois modelos na fase de requisitos.

A seguir, fazemos um breve relato de cada trabalho, focando principalmente os seguintes tópicos:

- O papel do MD na fase de requisitos;
- A obtenção de elementos do MD a partir do MUC;
- O estabelecimento de relações ou rastros formais entre os dois modelos;
- A consistência entre os dois modelos;
- A validação do MD.

Cada relato abaixo é seguido de uma breve análise do trabalho correspondente. No final da seção, a Tabela 3.1 mostra quanto cada trabalho avança no sentido de estabelecer relações ou rastros formais entre o MUC e o MD.

A Lightweight Approach to Consistency of Scenarios and Class Models (GLINZ, 2000)

Glinz adota a visão de que o MUC e o MD devem ser utilizados de forma complementar, e estabelece uma série de regras a serem seguidas durante a sua construção em paralelo, para promover a consistência entre eles, durante o processo construtivo.

Glinz destaca o fato de se tratarem de modelos não completamente formais, para justificar a definição de consistência adotada. Então, para ele, os dois modelos são consistentes se:

1. Não existirem contradições entre a informação presente no MUC e a informação presente no MD (tanto no compartilhamento de informação quanto na interação entre os modelos); e
2. Não houver *incompletude parcial* de um modelo em relação ao outro.

Por *incompletude parcial* ele entende a situação em que a presença de uma informação em um modelo, requer a presença de informação correspondente no outro modelo, a qual, no entanto, está faltando nesse modelo.

São identificadas três situações em que elementos de um modelo requerem elementos no outro modelo (rastros semiformais entre elementos dos dois modelos) :

- Estímulos em um UC que não se destinam imediatamente a produzir uma resposta, tipicamente produzirão uma mudança de estado, a qual requer uma **operação** ou transição de estado correspondente, no modelo de classes.

- Uma resposta em um UC que necessite dados que o estímulo desse UC não provê, requer dados armazenados. Esses **dados**, juntamente com as operações adequadas de acesso, devem estar representadas no modelo de classes.
- Toda **operação** no modelo de classes, que não é referenciada por outra operação nesse modelo, tipicamente é usada pelo UC para processar um estímulo ou produzir uma resposta.

A principal idéia na proposta é estabelecer a consistência entre os dois modelos através de uma inspeção manual sistemática, conjugada com as seguintes providências para apoiar e simplificar essa tarefa:

- Minimização de sobreposições entre os modelos;
- Introdução sistemática de referências cruzadas entre os modelos; e
- Definição de um conjunto de regras de construção e verificação dos modelos (com base nas três situações de correspondência entre elementos dos dois modelos, acima descritas).

Análise

A nosso ver, uma das principais desvantagens da proposta de Glinz é requerer obrigatoriamente dos *stakeholders* uma capacidade plena de leitura e análise do MD, pois para terem uma visão completa do sistema sendo modelado, eles precisam ler e analisar o MUC de forma conjugada com o MD. Outro aspecto a ressaltar é o baixo grau de automatização permitido pela proposta.

How to Use Linguistic Instruments for Object-Oriented Analysis (JURISTO *et al.*, 2000)

Os autores afirmam que a AOO (Análise Orientada a Objetos) não provê um método sistemático para a obtenção do MD ("*object model*") e do modelo de comportamento ("*behavior model*"). Segundo eles, os modelos resultantes da AOO são fortemente dependentes da experiência do modelador e não replicáveis. Em vista disso, propõem um método em 9 passos para a construção de modelos semiformais, cuja principal entrada é o "documento de requisitos" (descrição informal do problema em linguagem natural), traduzido manualmente para uma linguagem natural controlada (*Utility Language* - UL). A UL possui construções que correspondem formalmente (através de uma equivalência matemática) a elementos do MD (classes, relacionamentos, atributos, e operações) e do modelo comportamental. Com base na

sintaxe e semântica formais da UL, os autores apresentam um conjunto de regras e diretrizes para auxiliar o modelador a extrair os elementos desses modelos. Alguns procedimentos de verificação de consistência são preconizados ao longo das etapas do método.

Análise

Os autores reconhecem a importância de se ter um MD na fase de requisitos. Não utilizam UCs como ponto de partida para obtenção do MD: ambos os modelos (MD e modelo de comportamento) resultam do documento de requisitos e são, ao longo do processo de construção, verificados entre si. Alguns rastros formais são estabelecidos entre os modelos, por conta de sua base comum, representada pela tradução do documento de requisitos na UL.

Uma vez que o método proposto adota a abordagem de análise lingüística de texto em linguagem natural, ele sofre as mesmas limitações de trabalhos semelhantes, entre eles: dependência da qualidade do texto que serve de base para a análise, necessidade dos modeladores terem conhecimentos lingüísticos, dependência da interpretação dos modeladores, e um grau limitado de automatização. No entanto, a utilização da UL tem uma influência favorável ao diminuir a necessidade de interpretação dos modeladores (pois a UL é legível pelos *stakeholders*) e aumentar o grau de automatização (graças ao mapeamento formal entre ela e o MD).

Coupling Use Cases and Class Models as a Means for Validation and Verification of Requirements Specifications (KÖSTERS *et al.*, 2001)

Os autores também consideram que o modelo de UCs e o modelo de classes de domínio devem ser utilizados de forma complementar, durante a fase de requisitos. Eles assumem que um modelo de classes preliminar é obtido de forma independente do modelo de UCs, e propõem um método para o "casamento apropriado" (*proper coupling*) dos dois modelos. Embora não esteja explicitamente definido no artigo, um "casamento apropriado" parece corresponder à capacidade do modelo de classes atender os UCs; ou mais precisamente, à capacidade das instâncias das classes executarem os UCs.

O método proposto pelos autores introduz um modelo intermediário - *activity graphs* (AG's), para especificar, de forma mais precisa, o comportamento descrito nos

UCs. Um AG é uma variação de uma máquina de estados, onde os estados representam a realização de *ações* ou *subatividades*, e as transições são disparadas pela conclusão das ações ou subatividades. Segundo os autores, a granularidade e a semântica dos AG's permitem estabelecer uma relação rastreável e um ajuste transparente (*seamless*), entre os dois modelos. São estabelecidos rastros formais entre as ações dos AG's e as operações das classes.

A validação dos UCs é feita com base nos AG's, enriquecidos com cenários e constelações de objetos de domínio[6]. Em conseqüência, o modelo de classes é verificado em relação ao modelo validado de UCs. Segundo os autores, ao se validar as constelações de objetos (durante a validação dos UCs), o MD é parcialmente validado também.

Análise

Rastros formais entre os dois modelos são estabelecidos apenas para as operações das classes. Em um primeiro momento, a determinação das classes continua sendo feita de forma *ad hoc*, a partir do conhecimento adquirido pelo analista sobre o domínio, ou por sugestão de nomes presentes na descrição dos UCs. Depois, o método ajuda a refinar esse primeiro MD, durante a inspeção manual dos AG's enriquecidos com cenários e constelações de objetos. O método envolve um esforço considerável do analista para obter os AG's, o que não pode ser automatizado.

O MD não é propriamente validado, mas apenas verificado em relação ao modelo validado de UCs.

From Essential Use Cases to Objects (BIDDLE, NOBLE, 2002)

Esta proposta utiliza o modelo de UCs essenciais (*Essential* UCs - EUCs), produzido na fase de requisitos, para derivar um MD do sistema como um modelo preliminar de projeto.

UCs essenciais (EUCs) fazem parte do método de Projeto Centrado em Utilização (*Usage-Centered Design*), desenvolvido por CONSTANTINE e LOCKWOOD (1999). Sua motivação decorreu de limitações identificadas por esses autores nos UCs convencionais. Em particular, UCs tipicamente contém suposições,

[6] Os autores não definem o que vem a ser "constelações de objetos de domínio", mas tudo indica que seja uma visão parcial do MD, correspondente a um AG.

escondidas ou implícitas, sobre a interface do usuário a ser projetada posteriormente. Tais suposições são decisões prematuras de projeto, que ficam embutidas nos requisitos, dificultando a modificação ou a adaptação das mesmas, durante a fase de projeto da interface do usuário. EUCs foram concebidos para evitar esse problema. O termo *essencial* indica a intenção de capturar apenas a essência do sistema, através de descrições idealizadas, abstratas, livres de tecnologia.

Os autores propõem uma técnica inspirada em CRC (*class-responsibility-collaborator*) (BECK, CUNNINGHAM, 1989), para a distribuição das responsabilidades do sistema entre classes de objetos, realizando assim uma análise OO preliminar do sistema, a partir dos EUCs. As responsabilidades de um objeto incluem o conhecimento que ele deve possuir (atributos) e as ações que ele deve realizar (operações) (WIRFS-BROCK *et al.*, 1990). O procedimento proposto inicia com apenas uma classe representando todo o sistema, à qual são atribuídas todas as responsabilidades extraídas da descrição dos EUCs. Em seguida, como no CRC, outras classes candidatas são identificadas, e as responsabilidades são compartilhadas ou delegadas, em várias iterações, até que o modelo atinja um estado suficientemente estável.

Análise

A proposta não estabelece qualquer rastro formal entre elementos dos dois modelos. Entretanto, rastros podem ser explicitamente criados pelo analista entre as responsabilidades descritas nos EUCs e aquelas incluídas no modelo de classes, reforçando a rastreabilidade entre esses dois modelos.

A proposta cobre a identificação das classes e suas responsabilidades (atributos e operações), embora informalmente. A freqüência de ocorrência de um termo (na descrição das responsabilidades de uma classe) sugere a criação de novas classes, enquanto que a extração das responsabilidades, a partir dos EUCs, bem como a distribuição das mesmas entre as classes, são realizadas segundo a interpretação do analista. Nada é dito sobre como estabelecer as associações entre classes.

A consistência e a completude parcial entre os modelos é promovida pelo próprio processo construtivo (informal), que deriva o MD a partir dos EUCs. Por isso, e também por considerarem o modelo resultante dessa derivação como um modelo

preliminar de projeto, os autores não se preocupam em validar o MD com o envolvimento direto dos *stakeholders*.

Todo o processo demanda a interpretação do analista e, portanto, um esforço considerável. Por esse motivo, é pouco automatizável.

From use cases to classes: a way of building object model with UML (LIANG, 2003)

O autor apresenta uma proposta de obtenção do MD (na verdade, um diagrama de classes) a partir dos objetivos representados pelos UCs do sistema (as descrições dos UCs não são utilizadas). Lança mão de um modelo intermediário, denominado diagrama "*UC-Entity*", que mostra as entidades[7] envolvidas em cada UC, e permite a análise da colaboração entre elas, para o alcance do objetivo do UC. Essas entidades se tornam as classes do MD.

Aos atores e *stakeholders* são feitas perguntas pré-estabelecidas, para guiar a maior parte das etapas do processo de construção, que cobre a identificação das classes (entidades), atributos (*features* das entidades), associações (referências entre entidades), e operações (resultantes da análise da colaboração das entidades).

Análise

A proposta oferece uma cobertura abrangente dos elementos do MD, a custa do grande envolvimento dos analistas e *stakeholders* na interpretação dos UCs – nenhum rastro formal é identificado entre os elementos do MUC e do MD – o que impossibilita sua automatização. Os autores inovam, de certa forma, ao deixarem de lado a descrição dos UCs e considerarem apenas nos objetivos dos UCs. Talvez por isso, não tratem da validação do MD obtido. Entretanto, também não apresentam evidências suficientes da efetividade dessa abordagem.

A consistência entre os dois modelos é feita pela análise da colaboração das entidades (classes) na realização dos UCs, via diagramas *UC-Entity*.

[7] Algo que desempenha um papel para o alcance do objetivo do UC.

UCDA: Use Case Driven Development Assistant Tool for Class Model Generation (SUBRAMANIAM *et al.*, 2004)

Os autores propõem derivar o diagrama de classes (DC) a partir dos UCs. Para isso, empregam análise lingüística sobre dois documentos produzidos na fase de requisitos. O primeiro deles é um documento preliminar de requisitos, escrito em linguagem natural irrestrita; o segundo é composto das descrições de UCs, escritas em uma linguagem natural controlada, produzidas a partir do documento preliminar, com o auxilio da ferramenta proposta (UCDA).

A solução proposta é capaz de sugerir vários elementos do DC, para confirmação pelo modelador: objetos (classes), associações, mensagens entre objetos e responsabilidades (a partir das mensagens). Graças à identificação das mensagens passadas entre objetos, a ferramenta é também capaz de gerar diagramas de colaboração. Na obtenção dos objetos e das mensagens, é utilizado um glossário como forma de aumentar a consistência do DC gerado.

Análise

O diagrama de classes (DC) é derivado a partir dos UCs, sem participação ativa dos *stakeholders*. Portanto, há uma prevalência do MUC sobre o DC, na fase de requisitos.

A proposta cobre vários elementos do DC, graças à sofisticada análise lingüística realizada automaticamente, a partir da descrição em linguagem natural controlada, dos UCs. Para isso, exige dos analistas e *stakeholders* a capacitação na utilização dessa linguagem. Além disso, os elementos gerados precisam, em geral, ser validados por eles.

O tratamento da consistência e completude do DC em relação ao MUC é feito em parte pelo processo construtivo e em parte através de verificação, uma vez obtido o DC. Não está claro até que ponto a simples utilização de um glossário é capaz de reduzir a ambigüidade das descrições em linguagem natural, a ponto de garantir um nível satisfatório de consistência.

Os *stakeholders* não participam diretamente na elicitação e validação das abstrações representadas no DC.

No artigo, um dos principais pontos que fica sem esclarecimento é o grau necessário de envolvimento dos analistas e *stakeholders* na verificação e na validação dos modelos.

Leitura Baseada em Perspectiva: A Visão do Projetista Orientada a Objetos (MAFRA *et al.*, 2006)

Os autores propõem uma técnica denominada OO-PBR (*Object-Oriented Perspective-Based Reading*) que tem por objetivo revisar um documento de requisitos[8] segundo o ponto de vista ou perspectiva do projetista do sistema. Essa técnica fornece um conjunto de questões que devem ser respondidas com base no documento de requisitos escrito em linguagem natural livre, à medida que o revisor percorre as etapas de construção de um MD. Dificuldades encontradas na construção do MD podem significar a presença de um defeito no documento de requisitos. Portanto, o processo visa apoiar a compreensão do documento de requisitos e a detecção de defeitos nele. A técnica propõe 3 formulários: um para anotar as informações relevantes à construção do MD; outro para documentar as funcionalidades do sistema; e mais outro para o registro dos problemas encontrados no documento de requisitos. O MD resultante deve ser encarado como um modelo preliminar, a ser eventualmente aperfeiçoado nas etapas subseqüentes de análise do sistema.

A extração de elementos para compor o MD é feita com base em uma análise lingüística simples, onde nomes são mapeados para classes ou atributos, e verbos presentes em frases que envolvem classes, em relacionamentos entre classes (inclusive de herança). Além disso, existem diretrizes para obtenção da multiplicidade das associações e das funcionalidades do sistema.

Análise

O trabalho reconhece a importância do MD para a compreensão dos requisitos de um sistema. A obtenção de elementos que compõem o MD é feita com base em análise lingüística de um documento escrito em linguagem natural livre, o que acarreta uma maior dependência do modelador (revisor), pela necessidade de interpretar o documento, e prejudica a automatização do processo. Entretanto, isso é justificável tendo em vista que o principal objetivo da técnica é a detecção de defeitos em especificações de requisitos, e não a construção do MD.

[8] Não necessariamente um MUC.

Considerações Finais sobre as Propostas da Literatura Científica

A Tabela 3.1 agrega a informação sobre a cobertura das propostas apresentadas[9], no que diz respeito à identificação de rastros formais entre os elementos do MUC e do MD. Nela são indicados os elementos do MD contemplados com esses rastros.

Tabela 3.1 – Elementos do MD com rastros formais a partir do MUC

	[1]	[2]	[3]	[4]	[5]
Classes					x
Atributos	x				x
de estado					
Associações					x
Multiplicidade					
Operações	x	x			x
Assinatura					x

Legenda:
[1]: (Glinz, 2000)
[2]: (Kösters *et al*., 2001)
[3]: (Biddle *et al.*, 2002)
[4]: (Liang, 2003)
[5]: (Subramaniam *et al*, 2004)

Pela observação da Tabela 3.1, fica evidente que as propostas têm, em geral, dificuldade para cobrir os principais elementos do MD. Mesmo a proposta de SUBRAMANIAN *et al*. (2004), que trata a maioria deles, o faz apenas sugerindo os elementos, que devem ser, em seguida, confirmados ou não pelo modelador.

3.3.3 Propostas da Literatura Técnica

A seguir, apresentamos as orientações que alguns livros fornecem quanto à utilização dos dois modelos – UCs e MD, na fase de requisitos. Nosso objetivo é fornecer uma visão representativa da técnica empregada (ou ensinada) nessa área. Assim como fizemos com relação aos artigos relatados na seção anterior, focamos principalmente os seguintes tópicos:

- O papel do MD na fase de requisitos;
- A obtenção de elementos do MD a partir do MUC;
- O estabelecimento de relações ou rastros formais entre os dois modelos;
- A consistência entre os dois modelos;
- A validação do MD;

No final da seção incluímos uma análise visando consolidar posições extraídas dos livros, sobre os tópicos acima.

[9] Exceto (JURISTO *et al*., 2000) e (MAFRA *et al.*, 2006) por não adotarem especificamente o MUC como ponto de partida para a obtenção do MD.

Object-Oriented Software Engineering - A User Case Driven Approach (JACOBSON *et al.*, 1992)

Os autores consideram o MD parte integrante do modelo de requisitos (juntamente com o modelo de UCs). Sua função é servir de suporte ao desenvolvimento desse modelo, ajudando a desenvolver uma terminologia comum a ser utilizada nos UCs. Eles propõem que a descoberta dos candidatos a objetos se faça a partir da análise da terminologia utilizada na descrição dos UCs, e que a decisão sobre sua inclusão ou não no MD deve levar em conta o contexto de sua utilização.

Os objetos são divididos em três tipos: objetos-entidade, de controle e de interface. Os objetos-entidade são aqueles que têm uma contraparte direta no domínio do problema, e modelam informações que o sistema manipula durante um período mais longo de tempo. Tipicamente, essas informações precisam ser mantidas mesmo depois do término da execução dos UCs que as manipulam.

Os autores consideram que, em geral, não se deve identificar operações no modelo de análise, uma vez que elas freqüentemente mudam no *projeto*. Entretanto, se o ambiente de implementação tiver pouco efeito sobre esse modelo, pode-se já definir as operações nele. Propõem então a elaboração de diagramas de interação para, a partir da descrição dos UCs, obter as operações.

Os autores pouco ou nada dizem sobre a obtenção dos atributos das classes, das associações entre elas, e demais elementos do MD, na fase de requisitos. Os autores também silenciam sobre como verificar a consistência entre os dois modelos e como validar o MD.

The Object Advantage - Business Process Reengineering with Object Technology (JACOBSON *et al.*, 1994)

Segundo os autores, o melhor é trabalhar interativamente entre os dois modelos.

Eles afirmam que encontrar objetos é fácil; o difícil é encontrar os objetos "certos". E sugerem identificar, para cada UC, os objetos necessários à sua execução, pois dessa forma, maiores serão as chances de se evitar objetos desnecessários. Igualmente, passando por todos os UCs onde um objeto desempenha um papel, será possível identificar as responsabilidades do objeto no sistema. Para eles, a identificação das associações estáticas entre as classes fica para uma etapa posterior.

Os autores não discutem como verificar a consistência entre o MUC e o MD, e como validar o MD.

Mastering the Requirements Process (2a. edição) (ROBERTSON, ROBERTSON, 2006)

Os autores sugerem a utilização de um modelo de dados exploratório, nas etapas iniciais da fase de requisitos.

A descoberta das classes pode ser feita, segundo eles, a partir do modelo de contexto - o diagrama que mostra os dados que fluem para dentro e para fora do *work*[10]. Em geral, os dados armazenados, utilizados pelo *work*, entram via fluxos de dados. Portanto, se existem dados dentro do *work*, deve haver algum fluxo de dados através do qual eles entraram. Para cada informação contida nos fluxos presentes no modelo de contexto, deve-se perguntar: "O que descreve essa informação?" ou "Qual é o sujeito dessa informação". O sujeitos são as classes. Portanto, após analisar, dessa forma, todos os fluxos de dados, é muito provável que todas as classes de domínio tenham sido identificadas.

Os autores não esclarecem como obter as operações das classes, as associações entre elas, e demais elementos do MD. Também não discutem como verificar a consistência entre os modelos e como validar o MD.

Escrevendo Casos de Uso Eficazes (COCKBURN, 2005)

Este autor praticamente nada diz sobre a utilização do MD na fase de requisitos e sua integração com o MUC. Apenas sugere que a identificação das classes se faça com base na ocorrência de termos que designam conceitos do domínio, na descrição dos UCs. E acrescenta que, sem a orientação dos UCs, o modelador tende a gastar tempo excessivo modelando partes do domínio que não são relevantes para o sistema desejado. Ou seja, os UCs provêem condições limites para uma modelagem de domínio adequada.

Software Requirements - Styles and Techniques (LAUESEN, 2002)

Este autor afirma que a análise da informação de domínio produz um modelo capaz de sobreviver à implementação do sistema com pequenas modificações. Entretanto, pouco ou nada acrescenta sobre esse modelo, a não ser que a especificação de operações, no

[10] A atividade de negócio para a qual o usuário deseja o apoio do sistema.

nível de domínio, resulta em várias operações que não podem ser aproveitadas posteriormente, na fase de projeto.

Quanto à validação do modelo de domínio, o autor considera que alguns usuários não são capazes de aprender o modelo E/R, e que modelos orientados a objetos não são para validação pelos usuários; eles consideram esses modelos não intuitivos, estranhos.

Requirements Analysis - From Business Views to Architecture (HAY, 2003)

O autor dá um destaque especial ao modelo de domínio, ao afirmar que, fundamentalmente, as regras de negócio dizem respeito aos dados: elas restrigem que dados podem ou devem ser criados no decorrer das operações de negócio. Segundo ele, a elicitação e definição das regras de negócio devem ser realizadas em conjunção com o desenvolvimento de um modelo de dados.

Sobre a identificação de classes (entidades), o autor sugere procurar os termos utilizados no vocabulário do domínio. Da mesma forma, sugere buscar os fatos[11] do domínio para identificar os atributos das entidades e seus inter-relacionamentos.

Para a validação do modelo de dados, ele sugere um parafraseamento utilizando a linguagem natural.

Use Case Modeling (BITTNER, SPENCE, 2002)

Os autores afirmam que o modelo de domínio só deve ser criado se existirem relacionamentos relevantes entre os conceitos descritos no glossário (o que acreditamos ocorrer para qualquer sistema não trivial).

Nada dizem sobre como identificar as classes de domínio, ou seus atributos, operações, relacionamentos, etc. Também não discutem a questão da consistência entre os modelos e sobre a validação do modelo de domínio.

[11] Sentenças relacionando termos do domínio.

Use Cases: Requirements in Context (2a. edição) (KULAK, GUINEY, 2003)

Segundo o autor, o nome das classes vem dos nomes que aparecem na descrição dos UCs. Do conjunto assim obtido de classes candidatas, devem ser removidos os nomes redundantes, irrelevantes, e que designam atributos de outras classes.

Nada diz sobre como obter as operações das classes, associações entre elas e demais elementos do MD. Também nada é dito sobre como verificar a consistência entre o MUC e o MD, ou como validar o MD.

Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Interactive Development (3a. edição) (LARMAN, 2004)

O autor adota o *Unified Process* (UP), que define o MD como um dos artefatos que podem ser criados na fase de modelagem do negócio. Segundo ele, os UCs estabelecem limites para o desenvolvimento do MD.

São sugeridas três estratégias para encontrar as classes do MD: 1) Reutilização ou modificação de modelos existentes; 2) Utilização de listas de categorias de conceitos (que são listas pré-elaboradas de conceitos relevantes em um dado domínio); e 3) Identificação de nomes ou termos na descrição dos UCs e em outros documentos. Com relação a essa última estratégia, o autor alerta para a impossibilidade de um mapeamento mecânico *nome-classe* e para os riscos decorrentes da ambigüidade da linguagem natural.

O autor não recomenda a inclusão de responsabilidades ou métodos no modelo de domínio, por considerar que essas são noções relacionadas a software, enquanto que o modelo de domínio descreve apenas conceitos do mundo real, e não objetos de software. A consideração das responsabilidades e métodos deve ficar, segundo ele, para a fase de projeto. Entretanto, ele ressalva que o termo "modelo de domínio" pode assumir outro significado: o da "camada de domínio de objetos de software", isto é, a camada de objetos de software abaixo da camada de apresentação ou da interface com o usuário. Essa camada de domínio é composta de objetos que representam coisas no espaço de domínio do problema, com métodos relacionados à "lógica de negócio" ou à

"lógica de domínio". Não há qualquer recomendação sobre como chegar a esses métodos na fase de requisitos.

Quanto às determinação das associações entre classes, a recomendação é de que sejam consideradas: a) associações que representem relacionamentos que devem ser preservados por algum tempo; e b) associações derivadas de uma "lista de associações comuns" (que incluiria transações do tipo: "A é uma transação relacionada com outra transação B"; "A é um item-linha da transação B"; "A é um produto ou serviço para a transação B"; etc.).

Nada diz sobre como determinar outros elementos do MD, tais como atributos das classes e multiplicidades das associações. Igualmente, as questões de consistência entre o MUC e o MD, e de validação do MD, são omitidas.

Aspect-Oriented Software Development with Use Cases (JACOBSON, NG, 2004)

Segundo os autores, uma passagem pelos cenários de negócio, que o sistema pretende apoiar, permite identificar as informações que o sistema deverá guardar ou manipular. Essas informações são modeladas como classes de domínio. Em geral, basta identificar, para cada classe, atributos essenciais e relacionamentos com outras classes, pois nesse estágio se está meramente capturando conceitos e seus relacionamentos. Mais adiante na fase de análise, com os UCs já especificados, as responsabilidades das classes são determinadas ao se elaborar diagramas de colaboração entre elas, para a realização dos UCs.

Os autores nada dizem sobre como verificar a consistência entre o MUC e o MD, nem como validar o MD.

Considerações Finais sobre as Propostas da Literatura Técnica

Sobre o papel do MD e do MUC. A maioria dos autores considera o MD como um modelo de apoio para a fase de requisitos. Em geral, recomendam a elaboração em paralelo, e de forma interativa, do MUC e do MD. Entretanto, os autores que adotam o *Unified Process* (UP) (como por exemplo, (LARMAN, 2004)) costumam indicar a elaboração do MD em uma fase anterior, que denominam de "modelagem do negócio", para depois servir de subsídio à elaboração dos UCs. Para muitos autores, o MUC tem um papel importante na elaboração do MD, estabelecendo o contexto do sistema e

orientando assim o modelador, na busca pelos conceitos relevantes. Ou seja, ele estabelece os limites do espaço de busca, evitando que conceitos desnecessários para o sistema em questão sejam elicitados.

Sobre a relação (rastros) entre elementos do MUC e do MD. Normalmente, os autores sugerem atenção aos termos (nomes, verbos, etc.) utilizados pelos *stakeholders*, e presentes na descrição dos UCs, como forma de descobrir candidatos a classes do modelo de domínio, bem como seus atributos e associações. LARMAN (2004) acrescenta outra sugestão: utilizar listas de conceitos e de associações comuns a um domínio. Alguns autores reconhecem a dificuldade para se encontrar as classes "certas" (JACOBSON *et al.*, 1994). Das propostas analisadas, apenas a de ROBERTSON, ROBERTSON (2006) sugere explicitamente a análise dos fluxos de informações entre o processo de negócio (*work*) e seu ambiente, como forma de obtenção das classes. É comum considerar que o modelo de domínio é composto apenas das classes, seus atributos e associações, ficando a identificação das operações e outros elementos para a fase de análise. A despeito disso, há pouca orientação sobre como obter os atributos das classes e as associações entre elas, na fase de requisitos. Para a obtenção das operações (responsabilidades), usualmente recomendam partir da descrição dos UCs e elaborar um diagrama de colaboração para cada um deles.

Sobre a verificação da consistência entre o MUC e o MD. Em geral, a verificação da consistência entre o MUC e o MD, na fase de requisitos, não é tratada pelos autores da literatura técnica. É provável que isso se deva à escassez e imprecisão das relações ou rastros identificados entre elementos dos dois modelos, nessa fase.

Sobre a validação do MD. Pouco é dito sobre a validação do MD na fase de requisitos, pelos autores estudados. LAUESEN (2002) afirma que alguns usuários não são capazes de aprender esse tipo de modelo, e HAY (2003) sugere parafrasear o MD em linguagem natural, visando torná-lo mais acessível aos *stakeholders*.

3.4 Questões de Investigação e Objetivo da Tese

Esta seção descreve (na subseção 3.4.1) um problema observado quando vários modeladores constroem, independentemente, MDs para um mesmo sistema: esses MDs ficam muito diferentes entre si. Isso inspirou o estabelecimento de algumas questões a serem investigadas (subseção 3.4.2), a partir das quais foi definido o objetivo principal desta tese (subseção 3.4.3).

3.4.1 O Problema da Inconsistência entre MDs

Em uma série de trabalhos, SVETINOVIC (2005) e SVETINOVIC *et al.* (2005, 2006) apresentaram e discutiram um problema observado ao longo de 5 (cinco) anos lecionando um curso sobre técnicas orientadas a objetos para o desenvolvimento de software. Nesse período, os autores revisaram pessoalmente 135 especificações de requisitos (em média 120 páginas cada), de um total de 195 elaboradas com a participação de 740 estudantes de Engenharia de Software, Ciência da Computação, Engenharia Elétrica e Engenharia de Computação. Na seqüência das três disciplinas do curso, os estudantes desenvolvem um sistema de telefonia que inclui um subsistema de informações para o gerenciamento de contas. Eles utilizam UCs para capturar requisitos no nível de domínio, e a Análise Orientada a Objetos (AOO) (COAD, YOURDON, 1990) como uma ponte para o projeto OO do sistema.

Fazendo a revisão das especificações produzidas pelos estudantes, e interagindo com eles, os autores observaram muitas dificuldades surgidas ao longo do processo de especificação. A dificuldade observada mais freqüentemente (em quase todas as especificações) se localiza na análise de domínio OO, mais especificamente, na:

1. Identificação de conceitos do domínio do sistema, e na
2. Atribuição da funcionalidade do sistema a esses conceitos.

Em particular, foi observada a falta de conceitos e responsabilidades, e equívocos na atribuição de responsabilidades aos conceitos.

Além de ocorrer mais freqüentemente, os autores afirmam que essa dificuldade tende a ficar sem solução. Também foram observadas dificuldades para trabalhar apenas com UCs no nível do sistema, e para encontrar níveis apropriados de abstração.

Como resultado, os modelos de domínio produzidos por diferentes grupos, para um mesmo sistema, resultam drasticamente diferentes (provavelmente por terem os grupos considerados conjuntos muito distintos de conceitos), e com um grande número de conceitos em níveis diferentes de abstração. Os autores consideram essa uma forma particular de inconsistência dos modelos de domínio.

O estudo levado a cabo pelos autores (SVETINOVIC *et al.*, 2005) traz evidências de que essa dificuldade se manifesta tanto em sistemas grandes como em sistemas pequenos. Além disso, segundo os autores, a dificuldade não está nos estudantes, nem na sua incapacidade de compreender a orientação a objetos, mas nas

limitações das técnicas correntes de análise orientada a objetos, pelo menos em sua aplicação à engenharia de requisitos.

Os autores concluem que simplesmente não há garantias de consistência entre os MDs produzidos por diferentes analistas para um mesmo sistema, e que não é possível prever que diferenças existirão, ou mesmo entender porque essas diferenças acontecem. A ampla variação entre os modelos de domínio produzidos por diferentes analistas para o mesmo sistema levanta um questionamento sobre o benefício de se aplicar AOO. Eles consideram ainda que é preciso confrontar essa questão e descobrir meios de reduzir ou controlar essa variação, e que, apesar de difícil, é essencial estabelecer relações significativas entre os artefatos envolvidos (UCs e o MD).

Motivado por esses resultados, Svetinovic desenvolveu uma tese de doutorado cujo objetivo foi encontrar uma solução para esse problema (SVETINOVIC, 2005) (SVETINOVIC, 2006). Como veremos na próxima seção, também faz parte da presente tese investigar uma solução distinta para o mesmo problema.

3.4.2 Questões de Investigação

A seção 3.2.1 defendeu que trabalhar com os dois modelos na fase de requisitos é uma necessidade, pois ambos capturam informações complementares, úteis para as fases subseqüentes do desenvolvimento do sistema. Mas, como os modelos modelam um mesmo objeto (o sistema), e não são completamente ortogonais, surge a questão da consistência entre eles.

A análise das propostas que tentam promover a consistência entre os dois modelos (vide seção 3.3), nos permite concluir que:

1. Quando se tenta derivar automaticamente o MD a partir do MUC, o resultado é um modelo muito incompleto. Todas as propostas que visam certa automatização desse processo de transformação utilizam alguma forma de análise lingüística (por exemplo, (KÄRKKÄINEN *et al.*, 2008)), sabidamente incapaz de levar a resultados conclusivos. Ou seja, a maior parte da obtenção do MD fica a depender da interpretação do modelador.
2. Quando se tenta verificar a consistência entre os dois modelos obtidos em separado, novamente, a parte passível de automatização é mínima. Em outras palavras, os rastros formalmente identificáveis entre os dois modelos são

poucos. Isso fica evidente no trabalho de GLINZ (2000) (descrito na seção 3.3.2).

Apesar dessas dificuldades, poder-se-ia argumentar que a não automatização da obtenção do MD, ou da verificação da consistência entre ele e o MUC, não é um problema real; ou seja, que por serem esses modelos muito diferentes[12], essa obtenção e verificação precisam mesmo ser feitas manualmente pelo modelador, com base no seu conhecimento do sistema pretendido e do domínio onde ele se insere. Mas se é assim, o MD (como qualquer modelo de requisitos) precisa ser validado pelos *stakeholders*. E essa validação leva a outros questionamentos.

A validação do MD por pessoas não acostumadas a essa técnica de modelagem é problemática. O *stakeholder* típico não se sente a vontade para ler e interpretar um MD (LAUESEN, 2002). Ele parece se ressentir do alto nível de abstração e da linguagem formal, utilizados nesse modelo. A saída, a nosso ver apenas parcial, é parafrasear o MD em linguagem natural, e com isso conseguir (pontualmente) que os *stakeholders* exprimam sua avaliação sobre os aspectos modelados. Entretanto, persiste outra preocupação, ainda maior, sobre a possibilidade de uma validação efetiva do MD pelos *stakeholders*.

O modelador não é a fonte mais confiável para escolher as abstrações de domínio relevantes para o sistema. Em geral, ele não detém ou domina o conhecimento necessário para elicitar com segurança essas abstrações. Daí a necessidade de validação pelos *stakeholders*. Mas, uma coisa é extrair dos *stakeholders* as abstrações que eles utilizam ou julgam mais relevantes, e outra coisa é apresentar a eles um modelo pronto, contendo as abstrações identificadas pelo modelador, e pedir que eles as validem. Mesmo contando com o esforço e boa vontade deles, ao serem apresentados a um MD pré-existente (elaborado pelo modelador), é possível que eles sejam induzidos a "aceitar" as abstrações lá representadas, e abram mão facilmente (talvez até inconscientemente) das suas próprias abstrações, utilizadas no dia-a-dia do negócio. Isso é ainda mais preocupante se considerarmos os resultados obtidos por SVETINOVIC *et al.* (2005), que mostram uma grande variação entre os MDs obtidos por diferentes modeladores, para um mesmo sistema. Qual dos MDs deveria ser validado? Conseqüentemente, há um sério risco de que tal validação não seja efetiva.

[12] Enquanto o MUC é funcional, o MD é orientado a objetos, e empregam níveis de formalização distintos.

Em vista do exposto, vemos indícios de que, em geral:

a) O modelo de UCs é pobre como fonte de informações úteis à determinação de um MD;

b) Os modeladores não têm o conhecimento necessário para tomar uma decisão segura sobre as abstrações a serem escolhidas;

c) O espaço de abstrações de um domínio é muito amplo e, portanto, sem uma orientação para limitar a busca nesse espaço, diferentes modeladores tendem a chegar a conjuntos mais ou menos disjuntos de abstrações, ao tentarem obter um MD.

Das análises e discussões anteriores, podemos identificar várias questões cuja investigação é relevante para aperfeiçoar o papel complementar do MUC e do MD na modelagem de requisitos:

Q1. Como aumentar a cobertura de elementos do MD, deriváveis automaticamente a partir do MUC?

Q2. Como reduzir, no MD obtido a partir do MUC, a dependência da interpretação do modelador?

Q3. Como minimizar os riscos apontados para uma validação efetiva do MD?

Q4. Como aumentar o grau de formalização da verificação de consistência entre os modelos?

Q5. Como eliminar (ou reduzir) o indeterminismo apontado por SVETINOVIC *et al.* (2005), no processo de obtenção do MD a partir dos UCs?

3.4.3 Objetivo da Tese

O objetivo deste trabalho de tese é obter respostas para os questionamentos listados no final da seção anterior, e propor uma solução correspondente que:

O1. Inclua mais rastros formais entre elementos dos dois modelos, contribuindo assim para um maior automatismo e cobertura de elementos, na derivação do MD a partir do MUC, ou na verificação de consistência entre os dois modelos;

O2. Elimine ou reduza a inconsistência entre MDs obtidos por diferentes modeladores, a partir dos UCs de um mesmo sistema.

Acreditamos que o alcance do segundo objetivo acima decorra, em parte, de se alcançar o primeiro. O próximo capítulo identifica e analisa as possíveis causas para os problemas aqui caracterizados, e propõe um enfoque para resolvê-los.

4. Causas dos Problemas e Enfoque de Solução

4.1 Introdução

No capítulo anterior identificamos alguns problemas associados à utilização conjunta e complementar do modelo de casos de uso (MUC) e do modelo de (classes de objetos de) domínio (MD), na fase de (elicitação e modelagem de) requisitos. Esses problemas são:

P1. A pequena cobertura automática para a obtenção de elementos do MD a partir do MUC, ou para a verificação da consistência desses modelos. Ambos os processos exigem um alto nível de intervenção do modelador.

P2. A disparidade (inconsistência) dos MDs obtidos por diferentes modeladores, a partir do MUC de um sistema (SVETINOVIC *et al.*, 2005).

P3. O alto risco de uma validação não efetiva do MD pelos *stakeholders*, decorrente da dificuldade dos mesmos para compreender e utilizar esse tipo de modelo (LAUESEN, 2002), e da indução que eles sofrem para aceitar as abstrações escolhidas pelo modelador.

Acreditamos que os problemas não são independentes entre si. Uma solução para P1 provavelmente afetará de forma favorável P2 e P3. Mais especificamente:

- Relação P1-P2: O aumento da cobertura automática na obtenção do MD a partir do MUC (P1) significará um decréscimo no número de elementos do MD que sãoescolhidos pelo modelador sem uma maior participação dos *stakeholders*. Essa diminuição da intervenção do modelador deverá resultar em MDs mais uniformes entre si, quando elaborados por diferentes modeladores (P2).
- Relação P1-P3: Com uma menor intervenção do modelador (P1), os *stakeholders* estarão menos sujeitos a serem induzidos a aceitar as abstrações criadas pelo modelador (P3).

A Figura 4.1 dá uma visão esquemática desses problemas e dos relacionamentos entre eles.

No restante deste capítulo, identificamos e analisamos as causas dos problemas apontados (seção 4.2) e, a partir delas, derivamos alguns princípios e requisitos que nortearam a busca de uma solução para os problemas (seção 4.3). Finalizando o capítulo, é enunciada a hipótese geral deste trabalho de tese (seção 4.4).

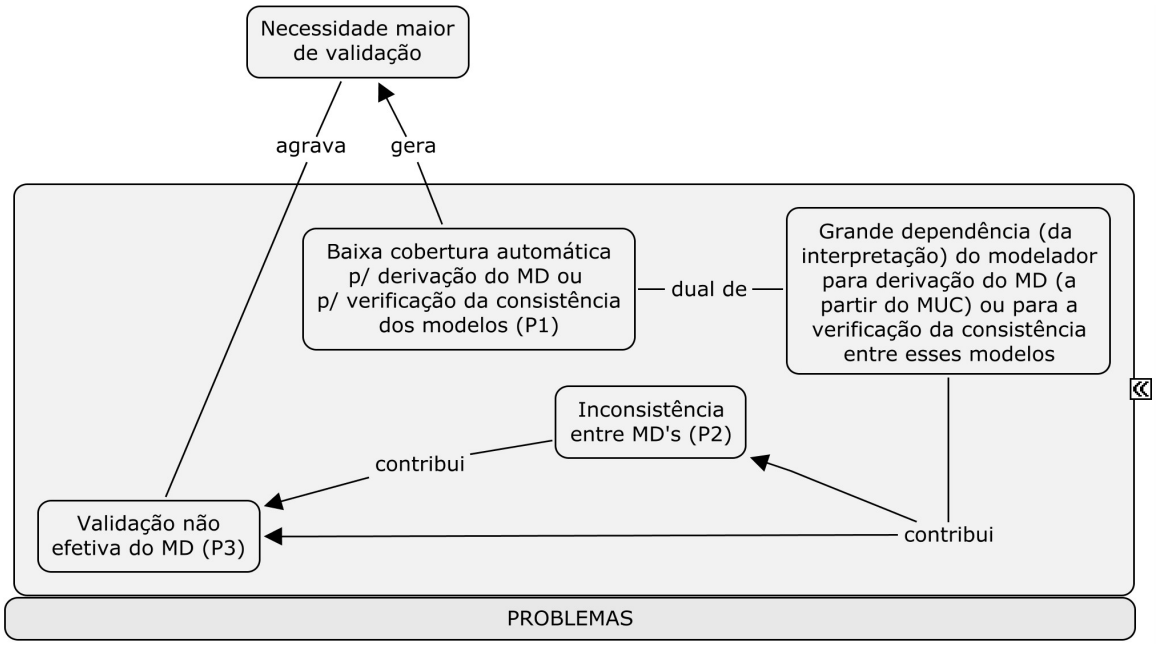


Figura 4.1 – Problemas-alvo

4.2 Causas dos Problemas

A seguir são apresentadas e discutidas as 4 causas principais dos problemas identificados na seção anterior. A Figura 4.2 mostra a relação entre os problemas e suas causas.

C1: Pobreza do MUC como fonte de informações úteis à determinação do MD

O MUC é pobre em subsídios para a elaboração do MD. Ou melhor, ele é pobre em elementos formais, a partir dos quais se possa obter, de forma segura e determinada, um MD, ainda que preliminar. As informações relevantes para o MD, quando existem, ficam "escondidas" nas descrições em linguagem natural dos UCs. Não há, durante a elaboração dos UCs, uma preocupação específica com a captura e especificação de elementos formalmente identificáveis, que possam determinar um MD ou facilitar a verificação da consistência entre os modelos.

Essa deficiência dos UCs em prover elementos facilmente mapeáveis para o MD fica evidente nas propostas que tentam garantir a consistência entre os modelos ou gerar o MD a partir do MUC (conforme vimos no capítulo anterior, seção 3.3). Nessas propostas, a automação possível se resume à análise lingüística das descrições textuais e

informais dos UCs, e por conta disso, os resultados ficam a depender, em grande parte, da interpretação que o modelador faz dessas descrições.

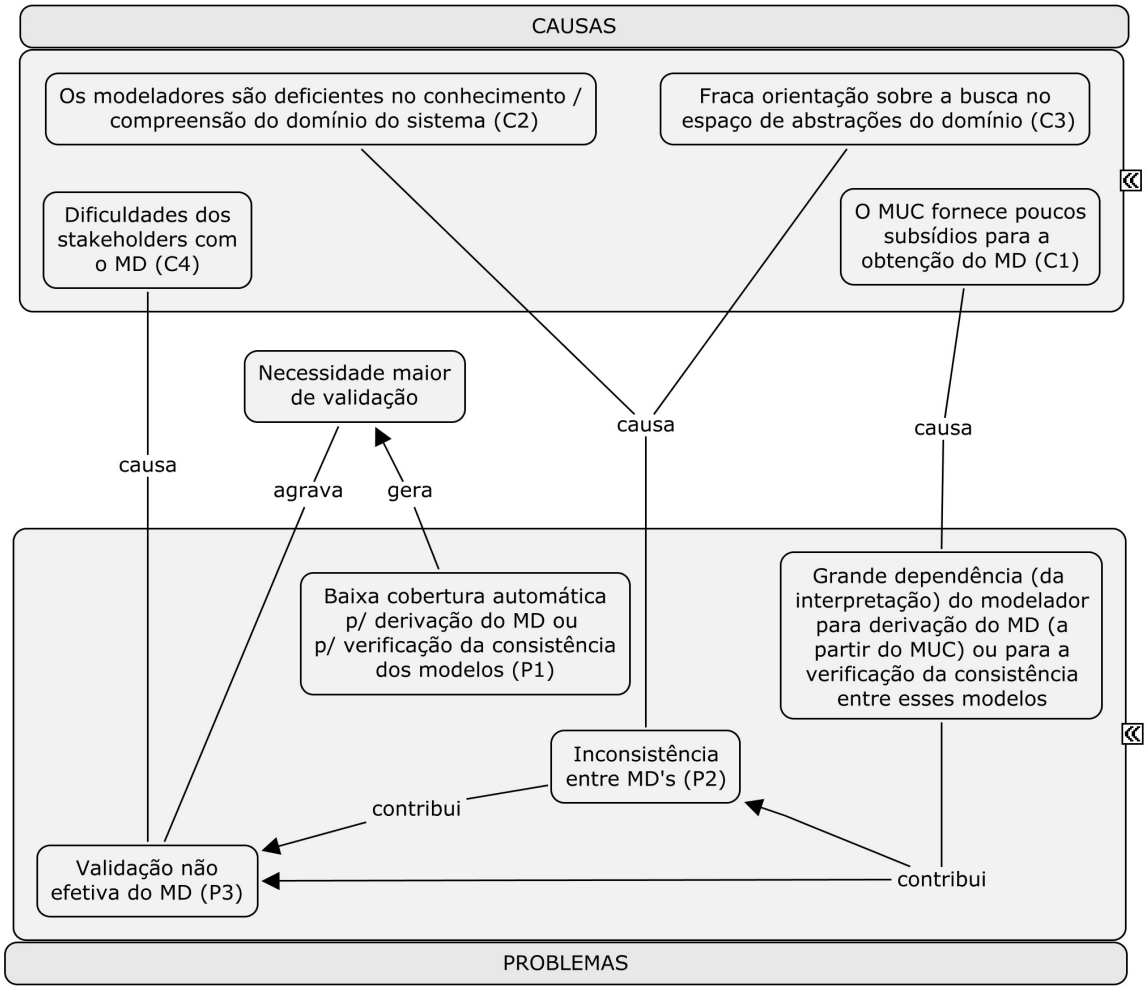


Figura 4.2 – Os problemas-alvo e suas causas

C2: Deficiência de conhecimento/compreensão do domínio pelos modeladores

Muito freqüentemente, o contato dos modeladores com o domínio do sistema se restringe ao breve período de tempo em que participam da especificação do sistema. Isso é muito pouco se comparado com a dedicação dos profissionais que desempenham suas atividades naquele domínio, ao longo de vários anos de trabalho.

A precariedade do conhecimento que os modeladores conseguem interiorizar sobre o domínio e necessidades dos *stakeholders* tem sido evidenciada em freqüentes alertas na literatura técnica e científica que trata de requisitos, sobre a falta de entendimento entre *stakeholders* e modeladores a respeito do sistema a ser construído, e

principalmente, sobre a incapacidade comumente observada da especificação de requisitos refletir as reais necessidades dos *stakeholders*.

Portanto, em principio, os modeladores não poderiam abrir mão do conhecimento e experiência dos profissionais do domínio, na hora de decidir sobre as abstrações a serem incluídas no MD, seus relacionamentos e esquemas representacionais para elas. Entretanto, não é isso que normalmente acontece. Durante a elaboração do MD, os modeladores em geral trabalham sozinhos, confiando basicamente no conhecimento precário adquirido sobre o sistema e seu domínio, para interpretar o MUC e decidir sobre qual deve ser o MD correspondente mais adequado. Não é de admirar que diferentes modeladores cheguem a MDs muito distintos entre si, para um mesmo sistema, conforme observaram SVETINOVIC *et al.* (2005).

C3: Fraca orientação sobre a busca no amplo espaço de abstrações do domínio

O espaço de abstrações de qualquer domínio não trivial é muito amplo, e pode ser observado sob diferentes níveis de abstração. Isso certamente traz dificuldades ao modelador na hora de escolher as abstrações que comporão o MD. O MUC, por não prover elementos facilmente mapeáveis para o MD (vide C1), não é um bom ponto de partida. E para agravar, o modelador não dispõe de uma orientação precisa sobre os níveis de abstração a considerar na elicitação dos UCs. O único critério geralmente recomendado para isso é todo UC ter valor para pelo menos um *stakeholder*. Infelizmente, como observa ROBERTSON e ROBERTSON (2006), esse é um critério por demais subjetivo. Nesse contexto, as observações de SVETINOVIC *et al.* (2005) sobre MDs produzidos por diferentes modeladores, para um mesmo sistema – MDs drasticamente diferentes, focando subconjuntos de conceitos muitos distintos, e com vários conceitos em diferentes níveis de abstração – parecem-nos uma constatação dos efeitos dessas deficiências da MUC, potencializados pela amplitude do espaço de abstrações do domínio.

C4: Dificuldade dos *stakeholders* para utilizar o MD

Conforme discutido na seção 3.4.2, o MD não é um modelo amigável ao *stakeholder* típico. Ele, normalmente, não se sente a vontade para ler e interpretar um MD (LAUESEN, 2002). Isso constitui um sério risco à efetividade da validação direta do MD, pelos *stakeholders*.

4.3 Enfoque de Solução

4.3.1 Deduzindo um Enfoque de Solução

Uma possível solução para os problemas indicados na seção 4.1 deve atacar as causas identificadas na seção anterior (4.2). Entretanto, não atacamos diretamente a causa C2 (Deficiência de conhecimento/compreensão do domínio pelos modeladores) e a causa C4 (Dificuldade dos *stakeholders* para utilizar o MD), pois as providências necessárias ao tratamento direto dessas causas extrapolam o tipo de solução buscada nesta tese, que é a proposição de uma forma diferente de modelagem de requisitos, que possa resolver, pelo menos em parte, os problemas apontados na seção 4.1.

No que diz respeito às causas C2 e C4, o que foi feito ao construir a solução desejada, é tentar evitar ou reduzir os efeitos delas. Por exemplo, evitamos (ou reduzimos) os efeitos de C2, transferindo para os *stakeholders* a decisão sobre a escolha das abstrações e outros elementos que deverão compor o MD. Já os efeitos de C4 podem ser reduzidos se, por exemplo, diminuímos a necessidade dos *stakeholders* validarem diretamente o MD.

Passemos agora às causas C1 (Pobreza do MUC como fonte de informações úteis à determinação do MD) e C3 (Fraca orientação sobre a busca no amplo espaço de abstrações do domínio). Essas causas são atacadas diretamente pela solução que procuramos.

Tratar a causa C1 significa elicitar e especificar, no âmbito da MUC, informações formalmente identificáveis sobre as abstrações de domínio, seus relacionamentos e representação em atributos e possivelmente operações, que constituirão o MD. O tratamento da causa C3 envolve construir uma orientação precisa sobre os níveis de abstração a considerar durante a MUC, que promova uma redução no espaço de busca de abstrações de domínio a incorporar ao MD, além de uniformizar o nível das mesmas.

Portanto, como solução para os problemas apontados, disponibilizamos uma variante da MUC que incorpora as seguintes características:

1. Capacidade de capturar e especificar, durante a modelagem com UCs, informações formalmente identificáveis, a partir das quais se possa derivar

automaticamente, boa parte de um MD adequado para o sistema a ser construído.

2. Maior orientação sobre níveis de abstração a serem utilizados na escolha dos UCs e das abstrações de domínio para o MD.

Além dessas características, a solução proposta deve ainda atender os seguintes requisitos:

3. Os elementos para a construção do MD devem ser extraídos diretamente dos *stakeholders*, de forma a evitar os efeitos das causas C2 e C4, pela minimização da intervenção do modelador e da necessidade dos *stakeholders* terem de validar diretamente o MD, respectivamente.

4. As características favoráveis da MUC (seção 2.4.1) devem ser mantidas. Ou seja, a modelagem proposta deve também ser amigável aos *stakeholders*, e estar centrada neles e em seu julgamento de valor. Aliás, essas características são imprescindíveis para possibilitar a captura, **diretamente dos *stakeholders***, dos elementos necessários à derivação do MD (item 1 acima).

5 Não deve haver um aumento excessivo no esforço de modelagem, em comparação ao esforço empregado na MUC.

A figura 4.3 resume, esquematicamente, a discussão acima.

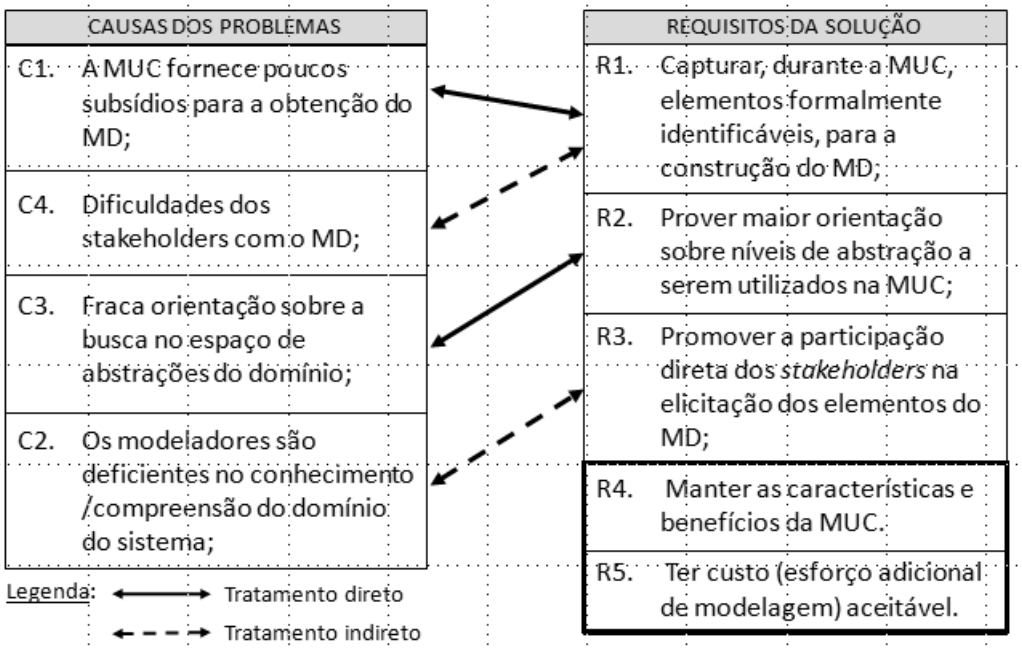


Figura 4.3 – Requisitos da solução

4.3.2 A Solução como um Modelo Integrado de Requisitos

Para GLINZ (2000), uma modelagem que emprega mais de uma técnica pode ser realizada tanto como uma *coleção de modelos* distintos, quanto como um único *modelo integrado* com diferentes visões. Em uma *coleção de modelos*, cada modelo é tratado (criado, mantido e usado) separadamente. Segundo ele, embora a separação entre os modelos pareça (inicialmente) facilitar a utilização dos mesmos, tal facilidade é obtida à custa da qualidade da modelagem como um todo, pois a consistência entre os modelos é difícil de ser alcançada. Por outro lado, um *modelo integrado* provê um único arcabouço conceitual, capaz de comportar todas as informações que necessitam ser capturadas. E, através do mecanismo de geração de visões, o modelador pode ainda trabalhar em um aspecto isolado, de um dos modelos cobertos na integração. Portanto, trabalhar com um modelo integrado é confortável para o modelador. Ainda segundo o autor, por definição, em um modelo integrado não existe qualquer sobreposição conceitual (as visões podem se sobrepor, mas elas são geradas a partir de uma única base comum). As regras de consistências são construídas uma vez por todas dentro do próprio modelo, sendo, portanto, parte integrante dele. O autor ainda menciona que praticamente todas as abordagens de modelagem OO estão na categoria de *coleção de modelos*, sendo a UML o exemplo mais proeminente. Segundo ele, a Engenharia da Informação (MARTIN, 1990) está baseada na idéia de um modelo integrado. No campo da especificação OO de requisitos, ele e seu grupo estão desenvolvendo uma linguagem e uma ferramenta denominada ADORA, baseada em um modelo integrado (GLINZ *et al.*, 2002) (GLINZ, 2008).

Levando em conta os requisitos identificados para a solução almejada neste trabalho de tese, acreditamos que um modelo integrado para requisitos, construído a partir do MUC, seja o caminho natural. Tal modelo deve capturar tanto o comportamento externo do sistema interagindo com seus atores (visão da MUC), quanto as estruturas (classes) de apoio, seus relacionamentos, informações de estado e comportamento interno do sistema (visão do MD).

4.3.3 Como Capturar Elementos do MD durante a MUC

Cabe então a pergunta: como fazer a integração das duas técnicas de modelagem (MUC e MD) em um mesmo arcabouço conceitual, de forma a obter um modelo integrado? Ou ainda, de forma mais concreta: como capturar as abstrações de domínio (e outros

elementos) relevantes para o MD, diretamente dos *stakeholders*, durante a elaboração dos UCs?

Acreditamos que a chave para conseguir isso está em aproveitar um recurso de certa forma pouco utilizado na MUC: os fluxos de informação trocados entre os atores e o sistema, durante a execução de cada UC. Conforme visto anteriormente na seção 2.3 (Abordagens de Modelagem com UCs), a maioria das propostas de modelagem de requisitos com UCs dá pouca ou nenhuma atenção a esses fluxos, e alguns autores apontam isso como uma deficiência. A análise a seguir procura evidenciar a importância desses fluxos para responder a questão colocada.

As pessoas, no âmbito de uma organização, desempenham suas atividades profissionais trocando informações entre si. Essas informações estão estruturadas em abstrações do domínio do negócio da organização. Um sistema (computacional) de informação, introduzido para apoiar a realização dessas atividades através da interação com seus atores, constitui um novo comunicador no ambiente organizacional. Na verdade, um comunicador privilegiado, pois, em grande escala, centraliza a comunicação, passando a ser um elemento intermediário na troca de informações entre os atores. Uma vez que deve se comunicar com seus atores, o sistema também precisa lançar mão das abstrações de domínio adequadas para que essa comunicação seja efetiva. Sendo um sistema computacional, essas abstrações estarão nele formalmente representadas. Portanto, acreditamos que se os fluxos de informação que modelam essa comunicação forem especificados com um mínimo de rigor formal[13], eles estarão, de alguma forma, refletindo essas abstrações, através do seu conteúdo e estruturação. Cabe-nos, então, descobrir um meio de aproveitar a composição e estruturação dos fluxos de informação assim modelados, para chegar às abstrações que eles refletem.

Se pudermos extrair abstrações (e possivelmente outros elementos relevantes para análise do domínio) a partir dos fluxos de informação, teremos descoberto uma forma de obter uma visão do MD do sistema, trabalhando dentro do arcabouço conceitual da MUC e em um nível menor de abstração[14], adequado a uma participação efetiva dos *stakeholders* – o nível das trocas de informação entre eles e o sistema. Esse é o caminho que trilhamos no presente trabalho de tese.

---

[13] Por exemplo, através de uma linguagem com sintaxe formal e semântica semiformal, sem comprometer substancialmente a sua validação pelos *stakeholders*.

[14] Em relação ao praticado no MD.

4.3.4 Restringindo o Espaço de Busca das Abstrações

Cada possível particionamento do sistema em UCs produz um conjunto diferente de fluxos de informação. Então, uma pergunta pertinente é: que influência isso pode ter na determinação dos elementos do MD, a partir dos fluxos? Haveria um particionamento particularmente favorável à determinação desses elementos?

O critério tradicionalmente utilizado para guiar a elicitação dos UCs (o UC ter valor para pelo menos um *stakeholder*) é muito subjetivo (ROBERTSON, ROBERTSON, 2006), dando muita liberdade ao modelador para a escolha do particionamento que julga adequado. Acreditamos que o excesso de liberdade acaba por levar a particionamentos e MDs muito discrepantes entre si, produzidos por diferentes modeladores, para um mesmo sistema (seção 4.2, causa C3). Por isso, caberia investigar um novo critério de particionamento, menos subjetivo e capaz de gerar particionamentos uniformes. Outros requisitos a serem satisfeitos pelo novo critério seriam:

- Ser uma especialização do critério de valor, de forma que todo UC elicitado corresponda a um objetivo a ser alcançado por algum ator. Isso é necessário para que a solução de modelagem resultante possa ser vista como uma especialização da técnica de UCs.
- Gerar UCs em um único nível de abstração. Esse requisito visa combater a dificuldade dos modeladores para trabalhar apenas com UCs no nível do sistema, e para encontrar níveis apropriados de abstração (seção 3.4.1).
- Gerar um conjunto de UCs que represente um particionamento funcional completo do sistema, ou seja, capaz de realizar toda a funcionalidade desejada para o sistema;
- Favorecer a determinação de uma visão do MD, a partir dos fluxos de informação de cada UC.

A determinação desse novo critério começou pela tentativa de atender o último dos requisitos acima listados (favorecer a determinação de uma visão do MD, a partir dos fluxos). Ao longo da investigação percebemos que, se o particionamento do sistema em UCs for feito de forma que o conteúdo dos fluxos exprima, principalmente, informações a serem persistidas entre *estados estáveis* do sistema, então é possível estabelecer rastros formais entre os fluxos e os elementos constituintes do MD. Por estado estável do sistema entendemos um estado alcançado no término da execução do

processo subjacente a um UC, com o alcance de um objetivo (de ator), alcance esse que, por sua vez, significa eliminar qualquer necessidade de *rollback* a um estado anterior[15]. Em outras palavras, é um estado em que o sistema se encontra logo após um ator ter concluído uma atividade de sua responsabilidade no fluxo de trabalho apoiado pelo sistema, podendo então "passar a bola" para outros atores darem prosseguimento a esse fluxo, no devido tempo.

Como veremos no próximo capítulo (5), o critério de particionar o sistema (em UCs) com base nos estados estáveis do mesmo, faz com que a busca por abstrações de domínio para o MD fique restrita às abstrações persistentes entre os estados estáveis do sistema. Também mostraremos que o critério de particionamento resultante é capaz de atender não apenas o requisito em questão, mas também os demais acima indicados.

4.3.5 A Solução

Concluindo, podemos dizer que a solução almejada deverá ser um modelo integrado para requisitos, resultante da especialização do modelo de UCs pela introdução de uma regra de elicitação de UCs, e pela exigência de um maior rigor na descrição dos fluxos de informação trocados entre o sistema e as entidades externas que com ele interagem (atores). Para facilitar a referência a essa variante da MUC, utilizaremos a denominação ***Modelagem Informacional de Requisitos*** (**MIR**). A Figura 4.4 resume isso através de um esquema gráfico, e evidencia a relação entre a solução e os requisitos que guiaram sua obtenção.

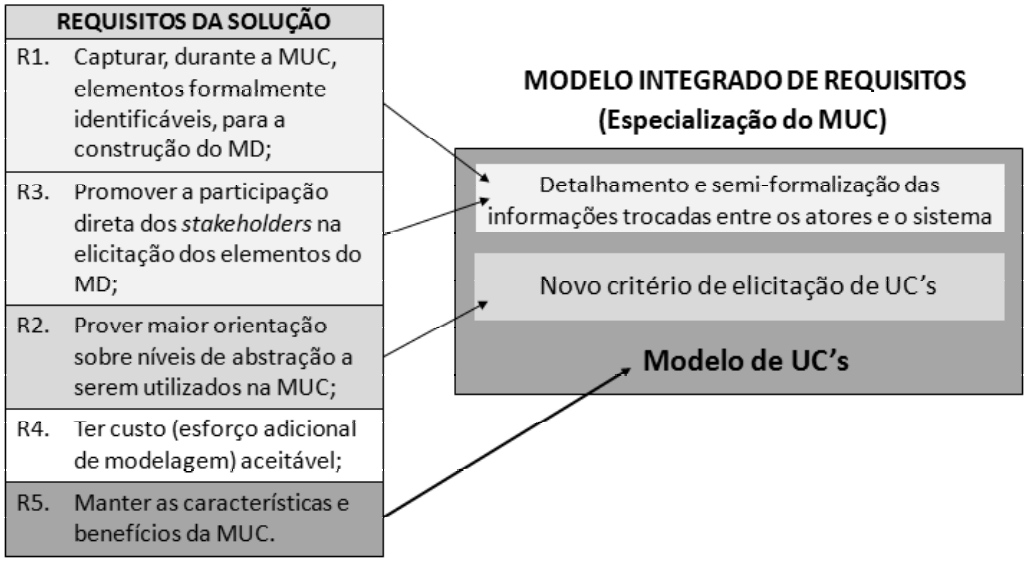


Figura 4.4 – Enfoque de solução

[15] Observe que, desta forma, estamos também atribuindo um significado mais formal (e, portanto, mais preciso) à noção de objetivo de um ator.

4.4 Hipótese Geral

Em vista do que foi exposto, a hipótese geral da presente tese é:

A MIR permite a derivação semi-automática de um MD "adequado", e promove uma maior uniformidade entre os MDs obtidos por diferentes modeladores, para um mesmo sistema. Isso, mantendo as principais vantagens alegadas para a MUC (seção 2.4.1) e sem exigir um aumento "excessivo" do esforço de modelagem.

O termo "MD adequado" será interpretado com base no conceito de adequabilidade entre dois modelos, enunciado por GLINZ (2000). Para ele, um modelo é adequando em relação a outro modelo se:

(1) Não existirem contradições entre as informações presentes em cada modelo; e

(2) Nenhum dos modelos está *parcialmente incompleto* em relação ao outro.

Ainda segundo GLINZ (2000), um modelo está *parcialmente incompleto* em relação ao outro quando uma informação nele requer uma informação correspondente no outro, a qual, entretanto, não está presente nesse outro modelo.

Uma avaliação sobre o incremento no esforço de modelagem, resultante da solução proposta neste trabalho de tese, poderá ser obtida comparando o tempo gasto na modelagem de um sistema utilizando a abordagem aqui proposta, com o tempo gasto na modelagem do mesmo sistema utilizando a abordagem tradicional da AOO.

5. A Solução: Info Cases

5.1 Introdução

Nosso objetivo no presente capítulo é mostrar como produzir, como resultado de uma modelagem com UCs ligeiramente adaptada, um artefato (modelo) de requisitos que incorpore relações formais entre o MUC e o MD, de forma a resolver os problemas apontados na seção 4.1, relativos à utilização desses dois modelos na fase de requisitos. A especialização dos casos de uso (UCs), proposta nesta tese, é denominada *info cases* (**IC**s).

Conforme antevimos na seção 4.3.3, as relações formais estarão presentes na especificação dos fluxos de informação trocados entre o sistema e seus atores, em cada UC. Para facilitar a referência a esses fluxos, passaremos a denominá-los, daqui para frente, de ***interface informacional*** (II) dos UCs.

As relações formais a introduzir na interface informacional dos UCs deverão transformar o MUC em um modelo integrado de requisitos (seção 4.3.2), permitindo a derivação, a partir dessa interface, de elementos de um MD correspondente: classes e seus relacionamentos, atributos e operações.

Com base na análise realizada no capítulo anterior (seção 4.3), é necessário tomar duas providências conjuntamente:

1) Estabelecer um novo critério para determinar os UCs, ou seja, para particionar em UCs, a funcionalidade do sistema; e
2) Especificar com um maior rigor formal a interface informacional dos UCs.

O restante deste capítulo está organizado da seguinte forma. A seção 5.2 apresenta o conceito de *nível informacional de objetivos* e mostra como implementar, com o auxílio dele, a primeira providência acima indicada. A seção 5.3, mostra como implementar a segunda providência: introduz uma linguagem semiformal para especificar a interface informacional dos UCs. A seção 5.4 apresenta as regras com as quais se pode derivar a visão MD a partir do modelo integrado, e exemplifica a utilização delas. A seção 5.5 faz um balanço da solução representada pelo modelo integrado, analisando a qualidade das especificações produzidas com a linguagem descrita na seção 5.3, o tipo adequado de sistema para aplicação do modelo integrado, o relacionamento dele com duas outras abordagens de análise de requisitos de sistemas, e

o grau de automatismo das regras e a completude dos MDs delas resultantes. A seção 5.6 discute dois trabalhos mais diretamente relacionados, extraídos da literatura. Em seguida, a seção 5.7 fecha o capítulo, apresentando algumas considerações finais.

5.2 O Nível Informacional de Objetivos

O objetivo dessa seção é definir e caracterizar um novo critério para a elicitação dos UCs de um sistema, que atenda aos requisitos identificados na seção 4.3.4:

1) Ser uma especialização do critério de valor, tradicionalmente utilizado na MUC;
2) Gerar UCs em um único nível de abstração;
3) Gerar um conjunto de UCs que represente um particionamento funcional completo do sistema; e
4) Favorecer a determinação de uma visão do MD, a partir da interface informacional dos UCs.

Conforme colocado na seção 4.3.4, se o particionamento do sistema em UCs for tal que o conteúdo da interface informacional dos UCs exprima as informações a serem persistidas entre estados estáveis do sistema, então, é possível perceber relações formais entre os fluxos e os elementos constituintes do MD. Obviamente, para isso será necessário adotar um formalismo na especificação da interface informacional dos UCs, na medida necessária à representação formal dessas relações (a ser detalhado na seção 5.3). Retomando a definição apresentada na seção 4.3.4, um **estado estável** do sistema é um estado persistente, alcançado ao término da execução do processo subjacente a um UC, representando o alcance de um objetivo (de ator), e significando a eliminação de qualquer necessidade de volta (*rollback*) a um estado anterior. Por **estado persistente** entendemos um estado cujas informações componentes persistem entre ativações subseqüentes do sistema, causadas por eventos provenientes do seu ambiente.

Para esclarecer esse conceito de estado estável, consideremos um sistema do tipo ponto-de-venda (PDV); por exemplo, aquele normalmente utilizado nos caixas de supermercados. Um sistema desse tipo, certamente, incluirá a função de registro dos itens adquiridos por um cliente, bem como a de registro do pagamento efetuado pela compra. Suponhamos, inicialmente, que o modelo de UCs do sistema inclua um UC distinto para cada uma dessas funções. O primeiro UC (registrar itens) não deixa o sistema em um estado estável, pois se a compra não for paga, o registro da mesma precisa ser desfeito (já que os itens voltam para a prateleira do supermercado); ou seja, é

necessário retornar o sistema ao estado anterior ao do registro. Tal situação não ocorreria se um único UC fosse responsável por desempenhar ambas as funções (registro e pagamento), pois o sistema estaria em um estado estável, ao término do UC.

Consideremos agora um sistema de gerenciamento de restaurante, com funções similares: a de registrar pratos e bebidas (itens de consumo) de uma refeição, e a de registrar o pagamento da refeição. No caso desse sistema, dois UCs, cada um desempenhando uma das funções, seria aceitável do ponto de vista do critério de estados estáveis. Isso porque, o registro dos itens de consumo de uma refeição deve ser mantido no sistema, mesmo que a refeição não seja paga[16].

Esse exemplo com o sistema Restaurante levanta uma questão: e se tivesse sido modelado com apenas 1 UC em vez de 2? Certamente, um UC que desempenhasse as duas funcionalidades (registrar itens e pagar a refeição)[17] também deixaria o sistema em um estado estável. Entretanto, teríamos "perdido" um estado estável: aquele que vigora ao término da execução do primeiro UC (registrar itens). Portanto, se desejarmos garantir um particionamento completo, ou seja, capaz de explicitar todos os estados estáveis do sistema, precisamos de algo mais como critério de particionamento.

Para perceber o que falta, basta analisar a diferença entre as soluções que adotam um único UC, para os sistemas considerados. Ambos UCs (o do sistema PDV e o do sistema Restaurante) necessitam de dois eventos externos[18] para serem ativados e completar sua execução, mas existe uma diferença entre os eventos que disparam a segunda parte do UC (aquela responsável pelo pagamento). No caso do sistema PDV, esse segundo evento deve obrigatoriamente ocorrer após o primeiro. O sistema deve embutir essa suposição, pois se tal não ocorrer, é necessário alguma ação para tratar essa situação excepcional. Já no caso do sistema Restaurante, o segundo evento não precisa necessariamente ocorrer em seguida ao primeiro. Por exemplo, após registrar os pratos e bebidas de uma refeição, o ator poderia registrar o mesmo para outra refeição, antes de registrar o pagamento da primeira refeição. Ou seja, nenhuma suposição pode ser embutida no sistema sobre o evento que o antecede – trata-se de um **evento autônomo**. Se um evento externo não é autônomo, ele é **subassumido** por algum outro evento

[16] Supondo que o sistema precise, por exemplo, emitir um relatório dos pratos e bebidas consumidas em um dado período de tempo; ou que o restaurante permita o pagamento postergado da refeição, para clientes habituais cadastrados no sistema.

[17] Nesta ordem para satisfazer restrições do domínio do problema.

[18] Evento disparado a partir do ambiente do sistema, ou seja, por um de seus atores.

autônomo. É o caso do segundo evento da solução de um único UC, adotada para o sistema PDV – ele é subassumido pelo evento (autônomo) que dispara a primeira parte do UC (aquela responsável pela função de registro dos itens da compra).

Resumindo, para figurar no particionamento almejado do sistema, um UC precisa satisfazer os seguintes critérios:

Critério 1: Propiciar o alcance de um objetivo de algum *stakeholder*.

Critério 2: Deixar o sistema em um estado estável.

Critério 3: Necessitar de apenas 1 evento autônomo para ser ativado e completamente executado.

Entretanto, para *UCs de consulta*, ou seja, UCs que não modificam o estado do sistema (não acrescentam, nem alteram ou eliminam nada da memória ou dos arquivos do sistema), basta verificar o atendimento de dois critérios apenas – Critério 1 e Critério 3. Isso porque, para um UC de consulta, o Critério 2 (estados estáveis) é sempre trivialmente atendido, já que após a realização do UC, o sistema permanece no mesmo estado em que se encontrava antes, presumivelmente um estado estável. No contexto do sistema Restaurante (seção 5.3), são exemplos de UCs de consulta: "(Gerente) Solicitar relatório de consumo diário" e "(Gerente) Solicitar relatório de receita". Supondo que o gerente emite (e usa) cada relatório de forma independente do outro, então cada UC é ativado pelo seu próprio evento autônomo e, portanto, eles devem ser mantidos separados.

Conforme mencionado na seção 4.3.3, pretendemos capturar elementos do MD durante a MUC, utilizando principalmente os fluxos de informações trocadas entre o sistema e seu ambiente, durante a realização dos UCs. Graças à conjugação dos três critérios acima, temos um particionamento do sistema (em UCs) particularmente favorável a esse intento. É o que procuramos mostrar a seguir.

O diagrama de UCs de um sistema representa um particionamento funcional do mesmo, onde cada UC pode ser visto como um processo – o processo subjacente ao UC. Uma vez que os UCs são ativados por eventos externos (provenientes dos atores), esses processos só se comunicam através de memória compartilhada (Figura 5.1[19]).

[19] Embora essa figura utilize a notação gráfica de DFD, o particionamento proposto nesta seção nada tem a ver com o particionamento preconizado na Análise Estruturada (YOURDON, 1990).

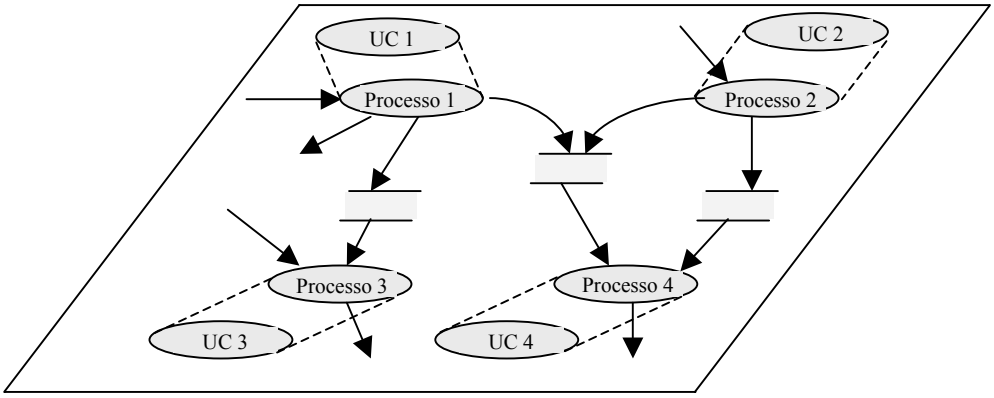


Figura 5.1 – Topologia dos processos subjacentes aos UCs

Analisando essa configuração de processos e seus fluxos de dados, podemos concluir que se uma informação entra em um processo (ou UC) e é necessária em outro processo (UC), ela deverá fazer parte da memória compartilhada pelos processos, para poder ser transmitida de um para o outro. Igualmente, se uma informação é produzida por um processo, e necessária em outro processo, se não puder ser reconstituída nesse outro processo, ela também deverá estar incluída na memória compartilhada. Ou seja, qualquer informação, de entrada ou de saída, que deva ser transmitida entre processos, deve ser incluída na memória compartilhada.

Conforme vimos acima, os três critérios propostos produzem um particionamento específico do sistema em UCs. Se o MD puder ser considerado uma representação dos estados da memória compartilhada resultante desse particionamento, então as informações trocadas pelo sistema com seu ambiente poderão ser utilizadas para a extração de elementos necessários à elaboração do MD.

O critério dos estados estáveis (Critério 2) faz com que apenas estados estáveis (persistentes e firmes) dessa memória sejam considerados (critério de suficiência), enquanto que a observância do critério de eventos autônomos (Critério 3) faz com que todos os estados estáveis sejam considerados (critério de completude). Além disso, todo estado estável modela abstrações relevantes para o sistema, pois está associado ao alcance de um objetivo de ator/*stakeholder*, mediante o uso do sistema (Critério 1). Em vista disso, podemos considerar que o MD desejado é aquele capaz de representar os estados da memória compartilhada resultante do particionamento gerado pelos três critérios, e conseqüentemente, os fluxos de entrada e de saída de informação, trocados entre o sistema e seu ambiente têm um papel importante na determinação desse MD.

Acabamos de constatar que o requisito 4 (seção 5.2) para o particionamento almejado (favorecer a determinação de uma visão do MD a partir da interface informacional dos UCs), enunciado no início desta seção, fica atendido. É possível verificar que os outros três requisitos também são atendidos.

A determinação de UCs pelos critérios propostos pode ser considerada uma especialização do critério tradicionalmente recomendado na literatura – o de todo UC ter valor para algum *stakeholder* (requisito 1). Isso porque, em obediência ao Critério 1, todo UC deve levar ao alcance de um objetivo de um ator ou *stakeholder* e, portanto, tem valor para esse ator/*stakeholder*. Considerando que o UC também deixa o sistema em um estado estável (Critério 2), sua realização dá ao ator/*stakeholder* um sentimento de "missão cumprida", ou mais especificamente, um sentimento de que a sua parte ou responsabilidade no fluxo de trabalho (*workflow*) está concluída, e que, a partir daí, outros atores (e, eventualmente, ele próprio) podem dar prosseguimento a esse fluxo. Este é um critério mais objetivo de valor, que os *stakeholders* percebem facilmente, pois exprime algo que faz parte da realidade do seu dia-a-dia, ou seja, a divisão de trabalho entre eles. Além disso, já que um evento autônomo abstrai todos os outros eventos por ele subassumidos, os UCs assim elicitados realizam toda a funcionalidade do sistema, constituindo um particionamento funcional completo do mesmo (requisito 3).

Por fim, é interessante observar que os processos subjacentes aos UCs do particionamento proposto estão em um nível especial e específico de abstração (requisito 2), pois:

- O resultado da decomposição (funcional) de um processo desse nível gera pelo menos um processo cuja ativação se dá por um evento não-autônomo (um evento anteriormente subassumido pelo evento autônomo disparador do processo decomposto, ou uma chamada síncrona proveniente de algum processo resultante da decomposição);
- O resultado da composição (funcional) de dois ou mais processos desse nível gera um processo que, para executar completamente, precisa de mais de um evento autônomo.

Para facilitar a referência a esse nível de abstração especial, passaremos a denominá-lo de ***Nível Informacional de Objetivos*** (**NIO**). O qualificador *informacional* lembra o papel especial da interface informacional dos UCs desse nível.

5.3 Interface Informacional dos Info Cases

No capítulo anterior (seção 4.3.3), antevimos a possibilidade de ter a especificação da interface informacional dos UCs refletindo formalmente as abstrações de domínio, que irão constituir a visão MD do modelo integrado de requisitos. Para isso, foram apontadas duas providências necessárias:

1. Determinar os UCs em um nível de abstração tal que suas interfaces informacionais exprimam informações a serem persistidas entre estados estáveis do sistema; e
2. Especificar as interfaces informacionais desses UCs, utilizando um formalismo capaz de permitir a captura de rastros formalmente detectáveis, para a derivação automática de elementos do MD.

A primeira das providências acima foi tomada com a instituição do NIO (Nível Informacional de Objetivos), resultante da determinação dos UCs segundo três critérios enunciados (seção 5.2). A continuação da presente seção mostra a linguagem semiformal adotada para cumprir a segunda providência, e ilustra sua utilização através de um sistema simples de gerência de restaurante, especialmente concebido para isso. Esse sistema possui dois atores – o gerente do restaurante e o cliente – e deve prover as seguintes funcionalidades:

a) Registrar o pedido de pratos e bebidas de cada cliente. Deverá também ser feita a emissão de um ticket de fechamento, quando o cliente solicitar a conta.
b) Cancelar um pedido por solicitação do cliente.
c) Registrar o pagamento ou a pendura da conta do cliente. A pendura é facultada apenas aos clientes considerados habituais do restaurante.
d) Emitir um relatório detalhado do consumo diário de pratos e bebidas, para fins de reposição de estoque.
e) Emitir um relatório detalhado da receita do restaurante, em um dado período, separando a receita realizada daquela a realizar (pelo pagamento de contas penduradas).
f) Registrar o cardápio em vigor, permitindo sua emissão, por iniciativa do gerente.

A especificação da interface informacional de um UC é constituída das duas partes listadas a seguir, e ilustradas na Figura 5.2 e na Tabela 5.1, respectivamente, para o sistema Restaurante:

- Especificação da composição dos fluxos; e

- Dicionário de itens elementares de informação.

A utilização de um glossário ou dicionário de dados já é uma prática comumente recomendada na modelagem com UCs (BITTNER, SPENCE, 2002). Portanto, basta acrescentar a especificação da composição dos fluxos da interface informacional, ao texto da especificação tradicional dos UCs (seção 2.2), para se obter o modelo integrado proposto nesta tese.

A linguagem adotada para descrever a composição dos fluxos da interface informacional é praticamente a mesma utilizada anteriormente pela Análise Estruturada de sistemas (YOURDON, 1990) (MAFFEO, 1992), para especificar fluxos e depósitos de dados. Uma referência mais recente para esse tipo de linguagem é (LAUESEN, 2002). Ela utiliza um pequeno conjunto de símbolos para a construção de expressões de dados. O Apêndice 1 desta tese descreve formalmente a sintaxe dessa linguagem em BNF (GORN, 1960), enquanto que a sua semântica é descrita informalmente a seguir.

Os sinais ➔ e ➔ indicam, respectivamente, *fluxo de entrada* e *fluxo de saída* de informações, do ponto de vista do sistema. Uma informação ou item em um fluxo pode ser elementar (indivisível) ou composto (agregado de outros itens elementares ou compostos). Os itens compostos também são chamados de *pacotes*, e indicados pelo símbolo 🗐. Os itens elementares podem ser *itens informados* ou *derivados*. Itens informados (ou de entrada) são aqueles cujo valor provém diretamente de uma entrada (itens presentes no fluxo de entrada de algum IC), enquanto que os itens derivados (ou calculados) não provêm diretamente de uma entrada, mas têm o seu valor derivado (calculado) pelo sistema a partir de outros itens informados ou derivados. Um tipo especial de item derivado são os itens *identificadores*. Eles se destacam pelo nome iniciado por *id_* (por exemplo, *id_cliente*), e representam abstrações do domínio da aplicação identificadas durante a modelagem de requisitos. A notação utilizada para descrever a composição de um fluxo ou pacote é a seguinte: + significa composição, ${}_{n}\{x\}_{m}$ significa de *n* a *m* ocorrências (ou repetições) de *x*, ( ) é utilizado para agrupar itens, | significa *ou* e [ ] delimita *itens condicionais*, ou seja, que nem sempre estarão presentes (podem não ser pertinentes, dependendo do contexto).

ATOR: Cliente	**IC 1: Abrir pedido**
➔ pedido = dt_pedido + id_mesa + itens_ped 🗐 itens_ped = $_1${id_item + quant_item} 🡐 id_pedido	
ATOR: Cliente	**IC 2: Cancelar pedido**
➔ cancela_ped = id_pedido	
ATOR: Cliente	**IC 3: Pedir a nota**
➔ solicitacao_nota = id_pedido + [id_cliente] 🡐 nota = id_pedido + nr_mesa + dt_pedido + itens_nota + vl_nota + [nome_cliente + tel_cliente] 🗐 itens_nota = $_1${id_item + cat_item + nome_item + pç_unit + quant_item + vl_item}	
ATOR: Cliente	**IC 4: Pagar a nota**
➔ pagamento = id_pedido + vl_entregue + dt_pagto 🡐 troco	
ATOR: Cliente	**IC 5: Pendurar a nota**
➔ pendura = id_pedido + id_cliente	
ATOR: Gerente	**IC 6: Cadastrar cliente habitual**
➔ cliente = nome_cliente + tel_cliente 🡐 id_cliente	
ATOR: Gerente	**IC 7: Atualizar o cardápio**
➔ item_consumo = nome_item + pc_unit + cat_item + descr_item 🡐 id_item	
ATOR: Gerente	**IC 8: Solicitar consumo diário**
➔ solic_consumo = dt_emissao + dt_consumo 🡐 consumo_dia = dt_emissao + dt_consumo + consumo_itens 🗐 consumo_itens = $_0${cat_item + {id_item + nome_item + quant_item} }$_2$	
ATOR: Gerente	**IC 9: Solicitar receita**
➔ solic_receita = dt_emissao + periodo_apur 🗐 periodo_apur = dt_inicio + dt_fim 🡐 receita = dt_emissao + periodo_apur + vl_consumo + receita_realiz + receita_pend + receita_txServ + receita_total	
ATOR: Gerente	**IC 10: Cadastrar mesa**
➔ mesa = nr_mesa 🡐 id_mesa	
ATOR: Gerente	**IC 11: Solicitar relação de notas penduradas**
➔ solic_penduras = dt_emissao + periodo_apur 🗐 periodo_apur = dt_inicio + dt_fim 🡐 penduras = dt_emissao + periodo_apur + {id_cliente + nome_cliente + tel_cliente + notas_pend + vl_pendCli} + vl_pend 🗐 notas_pend = $_1${id_pedido + dt_pedido + nr_mesa + itens_nota + vl_nota} 🗐 itens_nota = $_1${id_item + cat_item + nome_item + pç_item + quant_item + vl_item}	

Figura 5.2 – Composição dos fluxos da aplicação Restaurante

Os estudos experimentais realizados com ICs (Capítulo 6) têm indicado que os usuários, após um breve treinamento, passam a trabalhar confortavelmente com as expressões produzidas nessa linguagem. Segundo LAUESEN (2002, página 60 e seguintes), essas expressões são excelentes para os modeladores e aceitáveis para muitos usuários, que mesmo sem um formação específica em tecnologia da informação, acham essas expressões mais fáceis de compreender do que modelos E/R. Ainda

segundo ele, as expressões não têm a riqueza semântica de um bom dicionário de dados, mas as duas técnicas se complementam.

No Dicionário de Itens Elementares (DI), para cada item elementar presente na interface informacional de um UC, são informados seu *Nome*, *Descrição*, *Tipo*, e *Domínio* (Tabela 5.1). A informação de *Domínio* apresenta os valores que o item pode assumir. Uma forma estendida do dicionário incluiria ainda os campos *Unidade* e *Precisão*, aplicáveis aos itens com valores em uma determinada unidade de medida (exemplo, o item de informação *pç_unit* teria *Unidade*: Real, e *Precisão*: centavos). A experiência tem evidenciado que o DI desempenha um papel muito importante na especificação informacional dos UCs, pois é necessário definir, com precisão, a semântica de cada item de informação nela presente. Assim, o recomendável é que a *Descrição* de um item inclua não apenas a sua definição (denotação), mas também como é utilizado (conotação) (LEITE, OLIVEIRA, 1995).

Tabela 5.1 – Dicionário de itens elementares de informação, do UC 3 (parcial)

Ator: Cliente	**UC 3: Pedir a nota**		
Nome	**Descrição**	**Tipo**	**Domínio**
id_pedido	Identificador do pedido da nota a emitir.	Nr. Natural	
quant_item	Quantidade consumida de um item.	Nr. Natural	
vl_item	Valor do consumo de um item. Preço unitário × quantidade consumida do item.	Moeda	
cat_item	Categoria do item de consumo	Texto	{prato, bebida}

A visibilidade dos nomes de fluxos, pacotes e itens elementares está limitada ao próprio IC onde o nome aparece, exceto para os nomes de itens identificadores (aqueles com nome iniciado por *id_*), que são visíveis em todos os ICs. A decisão de restringir a visibilidade do nome de um item não-identificador, ao IC onde ele aparece, se justifica pela necessidade de permitir ao modelador flexibilidade e agilidade na hora de especificar a interface informacional dos ICs[20]. Já como itens identificadores ocorrem em muito menor número do que os itens não-identificadores, o benefício de se ter apenas um nome designando a mesma abstração ao longo de toda a especificação, é mais relevante.

Daqui para frente, utilizaremos o termo *info case* (IC) para designar um UC pertencente ao NIO, com interface informacional especificada na linguagem descrita nesta seção.

[20] Principalmente se o modelador não puder contar com uma ferramenta computacional de apoio, e se o sistema modelado for de grande porte.

5.4 Derivação da Visão MD do Modelo Integrado

O objetivo desta seção é apresentar um conjunto de regras para a derivação da visão do MD, a partir do modelo integrado construído com os ICs. Essas regras constituem uma evidência da integração alcançada. São também apresentados **padrões sintáticos** detectáveis na especificação da interface informacional dos ICs, que servem como heurísticas capazes de apoiar o emprego, ou complementar os resultados, das regras.

Esta seção está subdividia em duas subseções. A subseção 5.4.1 apresenta uma categorização para as regras e discute a relação entre elas e os padrões sintáticos, indicando o papel exato de cada um na derivação do MD. A seção 5.4.2 apresenta as regras e os respectivos padrões sintáticos, separados pelo tipo de elemento do MD que visam derivar (classes, associações, atributos e operações), e ilustra a aplicação das regras ao sistema de gerência de restaurante, cuja interface informacional encontra-se especificada na Figura 5.2 (seção 5.3). No final, é mostrado o MD resultante (Figura 5.3).

5.4.1 Categorias de Regras de Derivação e o Papel dos Padrões Sintáticos

A seção seguinte (5.4.2) apresenta as regras para a derivação da visão de objetos do modelo integrado de requisitos proposto. Essas regras cobrem todos os principais elementos de um modelo de classes de objetos: classes, associações, atributos e operações. Elas podem ser categorizadas segundo o grau de determinismo e automatização na obtenção dos resultados da sua aplicação.

As regras determinísticas são aquelas capazes de determinar (individualizar) o elemento ou aspecto por ela tratado, do MD. Esse é o caso, por exemplo, da regra R3a (sobre persistência), que sempre permite determinar a persistência (ou a não-persistência) dos itens de informação presentes nos fluxos de entrada dos ICs. Já a regra R3d (sobre alocação dos atributos nas classes) é um exemplo de regra não-determinística, pois não é capaz, no caso geral, de determinar uma classe única à qual alocar o atributo em questão. Além de R3a, as seguintes regras são determinísticas: R1, R2, R3b, R3c, R3e, R4a, R4b, R4c e R4d. As regras R2, R3d e R4e são não-determinísticas.

As regras automatizáveis são aquelas cuja aplicação pode ser decidida e realizada de forma automática (sem intervenção do modelador). Entre as regras da

próxima seção, apenas as regras R3e e R4a são completamente automatizáveis. Todas as demais são semi-automatizáveis, ou seja, algumas ações envolvidas na sua aplicação são automatizáveis e outras não. Por exemplo, a aplicação da regra R3a exige decidir sobre se um item é necessário (ou não) em outro IC distinto daquele em que ele entrou no sistema. Para isso, é preciso a intervenção do modelador. O *grau de automatismo de uma regra* é a relação entre o número de ações automatizáveis e o número total de ações envolvidas na aplicação da regra. A seção 5.5.5 apresenta o cálculo do grau de automatismo para cada uma das regras apresentadas na próxima seção (5.4.2).

Para auxiliar o modelador na aplicação das regras não-determinísticas e/ou não-automatizáveis, foram identificadas algumas heurísticas, aproveitando o formalismo e a estruturação da especificação da interface informacional dos ICs. Essas heurísticas se baseiam na ocorrência de *padrões sintáticos* nessa interface. Um ***padrão sintático para uma regra*** é uma configuração da interface informacional dos ICs, cuja ocorrência sugere um resultado específico para a aplicação da regra. Portanto, a cada padrão sintático corresponde uma heurística supostamente capaz de ajudar a resolver uma indeterminação da regra associada, ou de facilitar a sua aplicação (se a regra não for plenamente automatizável), ou ambos. Portanto, são desnecessários para regras determinísticas e automatizáveis. Por exemplo, na aplicação da regra R3a acima mencionada, que é semi-automatizável, a ocorrência do seguinte padrão sintático sugere a necessidade de persistir um item elementar de informação: "itens elementares homônimos, em ICs distintos, com pelo menos um deles no fluxo de entrada de um IC, e outro no fluxo de saída de outro IC". Isso é uma heurística porque o modelador não está obrigado a utilizar o mesmo nome toda vez que se referir a uma mesma informação em ICs distintos (apesar disso ser recomendável – vide seção 5.3).

A vantagem dos padrões sintáticos decorre do fato de que o seu reconhecimento (e, quase sempre, o emprego da respectiva heurística) pode ser automatizado. Cada padrão sintático está apresentado na próxima seção, logo após a apresentação da regra à qual se aplica.

5.4.2 As Regras de Derivação e seus Padrões Sintáticos

A apresentação das regras está dividida em 4 seções – uma para cada tipo de elemento do modelo de classes, que elas ajudam a determinar: classes, associações, atributos e operações. Para boa parte das regras não-determinísticas e/ou não-automatizáveis foi

possível identificar um ou mais padrões sintáticos (e respectivas heurísticas) capazes de apoiar a aplicação dessas regras. Tais padrões/heurísticas estão descritos logo após a apresentação da regra que os originou.

A especificação dos fluxos da interface informacional do sistema *Restaurante* (Figura 5.2) é utilizada para exemplificar a aplicação das regras. Ainda a título de exemplificação para o sistema Restaurante, quando uma regra possui padrão sintático/heurística de apoio, também é apresentado o percentual do número de vezes que a aplicação da heurística gerou um resultado consistente com o MIC do sistema (aqui denominado *grau de acerto* da heurística), segundo a avaliação do autor desta tese, com base em seu conhecimento e experiência na elaboração de MDs em geral, e do sistema Restaurante em particular. O modelo de domínio resultante é apresentado na Figura 5.3 (no final desta seção).

Classes de Objetos

As classes são determinadas pelos *identificadores gerados* nos ICs. Um identificador é *gerado* em um IC quando o seu valor é estabelecido durante a execução do IC. Seu propósito é ser uma referência para um objeto criado nesse IC. Por exemplo, *id_cliente* é um identificador gerado no IC 6 (*Cadastrar cliente habitual* – Figura 5.2), para servir de referência a um objeto *cliente*, criado durante o processamento do IC. A geração de um identificador só deve ser especificada em um IC, se for necessário fazer, em outro IC, referência ao objeto correspondente criado nesse IC.

Regra R1 (Determinação de classes). Cada identificador gerado dá origem a uma classe de objetos.

As classes obtidas a partir dos identificadores gerados representam abstrações utilizadas e compartilhadas pelos especialistas de domínio, que participam da definição dos requisitos do sistema. Assim, elas têm boa chance de serem classes úteis, com atributos e operações significativas no domínio (MEYER, 1997, pg. 733).

Padrão sintático R1-P1 (Identificadores gerados em um IC): IC com fluxo de saída composto apenas por identificadores.

Heurística: Os identificadores são gerados no IC.

No sistema *Restaurante* (Figura 5.2): *id_pedido*, identificador gerado no IC 1 - *Abrir pedido*, determina a classe *Pedido*. As demais classes determinadas dessa forma

são: *Cliente* (*id_cliente*, IC 6), *Item* (*id_item*, IC 7) e *Mesa* (*id_mesa*, IC 10). O *grau de acerto* da heurística[21] foi, neste caso, de (4/4) = 100%.

Associações entre Classes e suas Multiplicidades

Os relacionamentos entre os diversos objetos manipulados pelo sistema estarão representados na interface informacional (II) dos ICs, uma vez que eles devem ser necessariamente informados pelos atores ao sistema. Por exemplo, no sistema *Restaurante*, o relacionamento entre um pedido e a mesa no qual ele é feito, precisa ser informado ao sistema.

Como os objetos são representados por seus identificadores, os relacionamentos entre objetos estarão representados pela ocorrência de dois ou mais identificadores na interface informacional de cada IC. Entretanto, enquanto todos os identificadores presentes nos fluxos de entrada devem ser considerados na determinação das associações, nem todos aqueles existentes nos fluxos de saída são relevantes. Por exemplo, fluxos de saída que não incluam identificadores gerados não determinam associações inéditas, ou seja, associações que não possam ser determinadas pela presença de identificadores em fluxos de entrada e de identificadores gerados em fluxos de saída, pois não podem exprimir nada além do que já se encontra registrado no estado interno do sistema. O mesmo tipo de raciocínio permite concluir que, nos fluxos de saída, apenas os identificadores gerados contribuem para a determinação de associações entre classes.

Regra R2 (Determinação de associações). As associações entre classes estão entre aquelas indicadas pelos pares (não-ordenados) de identificadores presentes na interface informacional de cada IC (para fluxos de saída, basta considerar identificadores gerados[22]).

Em princípio, cada possível par de identificadores da interface informacional de um IC pode indicar uma associação a incluir no MD. No sistema *Restaurante*, por exemplo, os pares <*id_pedido*, *id_mesa*> e <*id_pedido*, *id_item*> obtidos do IC 1 (*Abrir pedido*), e o par <*id_pedido*, *id_cliente*>, presente nos ICs 3 (*Pedir a nota*) e 5 (*Pendurar a nota*), determinam as associações *Pedido-Mesa*, *Pedido-Item* e *Pedido-Cliente*, respectivamente.

---

[21] Segundo definição dada no início desta seção.

[22] Para se ter associações inéditas.

Padrão sintático R2-P1 (identificadores em fluxos distintos): Presença de identificador(es) no fluxo de entrada e de identificador(es) gerado(s) no fluxo de saída do IC.

Heurística (sugere associação): Para cada par <id_1, id_2>, que possa ser formado com id_1 sendo um identificador gerado (fluxo de saída), e id_2 um identificador no fluxo de entrada, criar uma associação entre as classes identificadas por id_1 e id_2. No caso de mais de um identificador gerado, as associações correspondentes a pares que compartilham o mesmo id_2 tendem a ser redundantes.

Padrão sintático R2-P2: Ocorrência de **R2-P1**, e o identificador *id₂* está no escopo de uma repetição[23].

Heurística (sugere multiplicidade): Os limites, inferior e superior, da multiplicidade no lado da classe identificada por *id₂* são, respectivamente, o número mínimo e máximo de repetições especificados no modelo informacional, ou "0" e "*", na falta dessa especificação (multiplicidade do lado da classe identificada por *id₁* não é sugerida).

Padrão sintático R2-P3: Ocorrência de **R2-P1**, e o identificador *id₂* não está no escopo de uma repetição.

Heurística (sugere multiplicidade): O limite superior da multiplicidade no lado da classe identificada por *id₂* será 1, e o limite inferior "0" se *id₂* for condicional, ou "1" se *id₂* não for condicional (multiplicidade do lado da classe identificada por *id₁* não é sugerida).

No sistema *Restaurante* (Figura 5.2), o padrão sintático R2-P1 ocorre no IC 1, com a presença dos identificadores *id_mesa* e *id_item* no fluxo de entrada, e do identificador gerado *id_pedido* no fluxo de saída. Assim, podem ser formados os pares <*id_pedido*, *id_mesa*> e <*id_pedido*, *id_item*>, o que sugere, pela heurística associada, a introdução das associações *Pedido-Mesa* e *Pedido-Item*, respectivamente (grau de acerto: 2/2 = 100%). Com relação às multiplicidades, a aplicação dos padrões sintáticos acima produz os seguintes resultados (grau de acerto: 2/2 = 100%):

a) Na associação *Pedido-Mesa*, como *id_mesa* (*id₂*) não está no escopo de repetição e não é condicional, a heurística associada ao padrão sintático R2-P3 sugere a multiplicidade "1..1" no lado da classe *Mesa* ; e

[23] Isto é, se *id₂* estiver entre { }'s.

b) Na associação *Pedido-Item*, como *id_item* (id_2) está no escopo de repetição com limite inferior "1" e limite superior não especificado, a heurística associada ao padrão sintático R2-P2 sugere a multiplicidade "1..*" no lado da classe *Item*.

Padrão sintático R2-P4 (identificadores no mesmo fluxo): Presença de identificadores (dois ou mais) somente em um dos fluxos do IC (considerando, para fluxo de saída, apenas identificadores gerados).

Heurística (sugere associação): Todo par de identificadores determina uma associação entre as classes por eles identificadas. Em fluxos com mais de dois identificadores, algumas dessas associações tendem a ser redundantes.

Padrão sintático R2-P5: Ocorrência de **R2-P4**, identificadores da associação no fluxo de entrada do IC, com um deles (digamos, id_2) no escopo de uma repetição em relação ao outro (id_1).

Heurística (sugere multiplicidade): O limite superior da multiplicidade no lado da classe identificada por id_2 é "*". O limite inferior de ambas as multiplicidades é "0", a menos que a aplicação de outro padrão sintático determine "1"[24] (o limite superior da multiplicidade do lado da classe identificada por id_1 não é sugerido).

Padrão sintático R2-P6: Ocorrência de **R2-P4**, identificadores da associação no fluxo de entrada do IC, com nenhum deles no escopo de uma repetição em relação ao outro.

Heurística (sugere multiplicidade): O limite inferior de ambas as multiplicidades é "0", a menos que a aplicação de outro padrão sintático determine "1" (os limites superiores não são sugeridos).

No sistema *Restaurante* (Figura 5.2), o padrão sintático R2-P4 ocorre nos ICs 3 (*Pedir a nota*) e 5 (*Pendurar a nota*), com a presença dos identificadores *id_pedido* e *id_cliente* no fluxo de entrada. Assim, pode ser formado o par < *id_pedido*, *id_cliente* >, o que sugere, pela heurística associada a esse padrão sintático, a introdução da associação *Pedido-Cliente* (grau de acerto: 1/1 = 100%). Com relação às multiplicidades dessa associação, a ocorrência do padrão sintático R2-P5, sugere "0" como limite inferior das multiplicidades de ambos os lados da associação (grau de acerto: 2/2 = 100%).

[24] Em outro UC, para a mesma associação.

Padrão sintático R2-P7: Ocorrência de **R2-P4**, identificadores da associação (gerados) no fluxo de saída do IC, com um deles (digamos, id_2) no escopo de uma repetição em relação ao outro (id_1).

Heurística (sugere multiplicidade): Os limites, inferior e superior, da multiplicidade no lado da classe identificada por id_2 são, respectivamente, o número mínimo e máximo de repetições especificados (ou, respectivamente, "0" e "*", na falta dessa especificação). A multiplicidade do lado de id_1 será "0..1" ou "1..1", caso id_1 seja condicional ou não, respectivamente.

Padrão sintático R2-P8: Ocorrência de **R2-P4**, identificadores da associação (gerados) no fluxo de saída do IC, com nenhum deles no escopo de uma repetição em relação ao outro.

Heurística (sugere multiplicidade): Os limites inferiores das multiplicidades são "0" ou "1", dependendo do identificador correspondente ser condicional ou não, respectivamente (os limites superiores não são sugeridos).

Os padrões sintáticos R2-P7 e R2-P8 não ocorrem no sistema Restaurante.

Atributos das Classes

A ativação de ICs depende da iniciativa de algum ator e, portanto, nenhuma suposição geral pode ser embutida no sistema sobre a co-existência, no tempo, dos processos subjacentes aos ICs. Conseqüentemente, todo item de informação a ser comunicado entre esses processos deve ser considerado informação de estado, ou seja, informação persistida nos estados do sistema alcançados pela execução dos processos. Além disso, como esses estados são sempre estáveis[25], tais itens de informação (de estado) devem aparecer como atributos das classes de domínio do sistema. Disso resultam as três regras a seguir.

Regra R3a (Persistência - item informado, na entrada). Todo item elementar não-identificador e informado, presente no fluxo de entrada de um IC, e necessário em outro IC, precisa ser persistido; caso contrário (se não for necessário em outro IC), não deve ser persistido[26].

[25] Estados consistentes, que podem persistir indefinidamente (seção 5.2).

[26] Já que estamos buscando um MD mínimo, ou seja, um MD que provê apenas o que é estritamente necessário ao suporte da funcionalidade descrita no MIC.

Padrão sintático R3a-P1. Itens elementares não-identificadores, homônimos, ocorrendo em ICs distintos, com pelo menos um deles no fluxo de entrada de um IC, e outro no fluxo de saída de outro IC.

Heurística: O item elementar de entrada em um IC é necessário em outro IC e, portanto, deve ser persistido.

No sistema *Restaurante* (Figura 5.2), por exemplo: o item elementar *dt_pedido* entra no IC 1 (fluxo de entrada *pedido*) e é reutilizado (consultado) nos ICs 3 e 11. Portanto, precisa ser persistido. Por outro lado, *vl_entregue* e *descr_item* não precisam ser persistidos; o primeiro porque é utilizado apenas no próprio IC em que entrou (IC 4, para o cálculo do *troco*), e o segundo por não ser utilizado em IC algum, nem mesmo no IC em que entrou no sistema[27] (IC 7). O *grau de acerto* do padrão sintático foi de (11/19) ≅ 58%.

Regra R3b (Persistência – item informado, na saída, valor histórico). Todo item elementar não-identificador e informado, presente no fluxo de saída de um IC, representando informação histórica[28] cujo valor não pode, em geral, ser restaurado nesse IC, precisa ser persistido; caso contrário, não deve ser persistido.

No sistema *Restaurante* (Figura 5.2): os itens elementares não-*identificadores* *nome_item* e *pç_item* são informados no IC 7 (*Atualizar o cardápio*), e estão presentes como informação histórica no fluxo de saída do IC 11 (*Solicitar relação de notas penduradas*). Portanto, esses itens precisam ser persistidos. Eles representam informação histórica no IC 11 porque entre a abertura de um pedido (IC 1) e a consulta da respectiva nota pendurada (IC 11), um item de consumo pode ter seu nome e preço unitário alterados (via IC 7).

Regra R3c (Persistência - item derivado, valor histórico). Todo item elementar não-identificador e derivado, presente (no fluxo de saída) em um IC, representando informação histórica cujo valor não pode ser restaurado nesse IC, precisa ser persistido; caso contrário, não deve ser persistido.

No sistema *Restaurante* (Figura 5.2) a condição para a aplicação desta regra não ocorre. Embora os itens elementares não-identificadores e derivados: *vl_consumo*, *receita_realiz*, *receita_pend*, *receita_txServ* e *receita_total* (IC 9), *vl_pendCli*, *vl_pend*,

[27] Uma potencial falha da especificação de requisitos, que esta regra ajuda a detectar.
[28] Informação relativa a um estado anterior do sistema, que possivelmente não mais vigora com o mesmo valor no estado atual.

vl_nota, e *vl_item* (IC 11), representem informação histórica, eles podem ser restaurados a partir de outros itens anteriormente persistidos (por exemplo, *vl_item*, no IC 11, pode ser restaurado a partir dos itens *quant_item* e *pç_item*[29] persistidos com base nas regras R3a e R3b, respectivamente).

Uma vez decidida a persistência de um item (regras R3a, R3b e R3c), o próximo passo é alocar o item como atributo de uma das classes determinadas com a regra R1. Na modelagem com ICs do NIO, a única forma de estabelecer a ligação entre um item e o objeto do qual representa uma propriedade (atributo), é incluir a identificação do objeto na mesma interface informacional que contém o item (ou seja, ambos devem participar da II do mesmo IC). É o que acontece, por exemplo, na interface informacional do IC 4, que inclui, além de *dt_pagto*, *id_pedido* para identificar o pedido que está sendo pago. A regra a seguir utiliza isso e o conceito de *classes alcançadas por um identificador*, que inclui: a) a classe que ele identifica, e b) as classes de associação[30] nas quais ele participa como um dos identificadores (se existirem). Esta regra também tem o papel de apontar as classes de associação que devem fazer parte do MD.

Regra R3d (Alocação de atributo / Criação de classe de associação). Cada item a persistir, presente na interface informacional de um IC, deve ser alocado como atributo de uma das *classes alcançadas pelos identificadores* presentes nessa mesma interface. Caso nenhuma dessas classes comporte o item, ele deve ser alocado em uma nova classe de associação criada para esse fim, correspondente a uma associação existente entre as classes alcançadas.

Padrão sintático R3d-P1. O item a persistir está, juntamente com um identificador, no escopo de uma repetição em relação a outro identificador. Heurística: Alocar o item como atributo da classe de associação determinada por esses identificadores.

Padrão sintático R3d-P2. O item a persistir está no fluxo de entrada, e o IC tem fluxo de saída com um ou mais identificadores gerados. Heurística: Alocar o item como atributo de uma das classes identificadas pelos identificadores gerados.

[29] Pela multiplicação de um pelo outro.

[30] Representam uma associação entre objetos de duas outras classes (não necessariamente distintas). Exemplo: a classe *Pedido-Item* (vide Figura 5.3).

Padrão sintático R3d-P3. O IC não tem identificador gerado, e o item a persistir está em um fluxo com apenas um identificador. Heurística: Alocar o item como um atributo de uma das classes alcançadas por esse identificador.

No sistema *Restaurante* (Figura 5.2), por exemplo: *quant_item* (IC 1) deve ser alocado na classe de associação *Pedido-item*, correspondente à associação entre as classes alcançadas *Pedido* e *Item*, criada para conter esse atributo (padrão sintático **R3d-P1**, classes alcançadas pelos identificadores *id_pedido* e *id_item*), *dt_pedido* (IC 1) deve ser alocado na classe *Pedido* (padrão **R3d-P2**, classe identificada por *id_pedido*), *pc_item* (IC 11) deve ser alocado na classe *Pedido-Item* (padrão **R3d-P1**, classe de associação determinada pelos identificadores *id_pedido* e *id_item*), e *dt_pagto* (IC 4) na classe *Pedido* (padrão **R3c-P3**, classe alcançada por *id_pedido*), etc. O grau de acerto proporcionado pelos padrões sintáticos foi de (9/9) = 100%.

A execução do processo subjacente a um IC sempre produz mudança de estado, seja do sistema ou do ambiente que com ele interage, pois corresponde ao alcance de um objetivo de algum *stakeholder*. A presente seção mostrou, até agora, que é possível detectar o tipo de mudança de estado no sistema, pela configuração da interface informacional dos ICs:

- Criação de objetos: interfaces com pelo menos um identificador gerado (no fluxo de saída).
- Criação de associação entre objetos: interfaces com pelo menos dois identificadores.
- Modificação do estado de um objeto ou de uma associação entre objetos: interfaces com algum item a persistir.

Comumente, a modificação do estado de um objeto ou de uma associação entre objetos ocorre nos ICs cuja interface informacional contém algum item de informação a persistir (última configuração relacionada acima). Entretanto, qualquer IC pode produzir mudança de estado em um objeto (ou associação entre objetos), mesmo inexistindo um item a persistir na sua interface informacional. Isso porque, a simples ocorrência do evento disparador do IC pode implicar a mudança. Obviamente, essa mudança também é implementada via um atributo do objeto (ou da associação), que pode ser um atributo criado anteriormente para persistir um item da interface, ou um atributo especialmente criado para refletir a mudança. Esse último tipo de atributo será distinguido dos demais atributos (ordinários) pela denominação ***atributo de estado***.

Portanto, diferentemente dos atributos ordinários que correspondem diretamente a itens de dados a persistir, presentes nos fluxos de entrada e de saída dos ICs, os atributos de estado não aparecem na interface informacional dos objetivos. Apesar disso, ainda é possível extrair dessa interface alguma indicação sobre a necessidade de introduzir esse tipo especial de atributo, conforme evidencia a regra enunciada a seguir.

Regra R3e (Atributo de estado). Para cada IC com interface informacional constituída apenas de fluxo de entrada, contendo exclusivamente um único identificador, deve ser introduzido um atributo de estado em uma das classes alcançadas por esse identificador.

No MD do sistema *Restaurante* (Figura 5.2), os atributos *cancelado?* e *pendurado?* foram introduzidos na classe *Pedido* (classe alcançada por *id_pedido*), pela utilização dessa regra nos ICs 2 (*Cancelar pedido*) e 5 (*Pendurar a nota*), respectivamente.

Operações das Classes

As operações a serem incluídas no modelo de classes de domínio deverão possibilitar:

a) Realizar as mudanças de estado no sistema, pela criação e alteração de objetos (regras 4a e 4d); e

b) Realizar as mudanças de estado no ambiente do sistema, pela produção das saídas previstas na interface informacional, representadas pelos itens derivados não persistidos[31] e fluxos de saída (regras 4b e 4c).

Regra R4a (construtoras). Toda classe recebe uma operação construtora.

No sistema *Restaurante*, essa regra determinou 5 operações construtoras: *Pedido( )* (IC 1), *Cliente( )* (IC 6), *Item( )* (IC 7), *Mesa( )* (IC 10) e *Pedido-Item()*, nas respectivas classes homônimas.

Regra R4b (itens derivados). Todo item derivado não persistido produz uma operação para calcular o seu valor[32].

No sistema *Restaurante*: operações *vl_consumo( )*, *vl_realiz( ), receita_pend( )*, *receita_txServ( )*, *receita_total( )*, e *vl_pend( )* (as 5 primeiras provenientes do IC 9, e a

[31] Os itens calculados persistidos não geram operações, pois estão disponíveis na interface da classe como atributos.

[32] Itens derivados persistidos não geram operações, já que eles estão disponíveis como atributos, na interface das classes.

última do IC 11); *vl_nota* (IC 3), *quant_item( )* (IC 8), e *vl_pendCli( )* (IC 11); e *troco( )* (IC 4).

Regra R4c (fluxos de saída). Todo fluxo de saída, presente em IC cujo processamento não produz mudança de estado no sistema (ou seja, não possui identificador gerado, nem item persistido ou atributo de estado, e não altera o valor de algum item persistido – tipicamente, ICs de consulta) e que não se resuma a apenas um único item não persistido[33], dá origem a uma operação para produzir o fluxo.

No sistema *Restaurante*: operações *consumo_dia( )* (IC 8), *receita( )* (IC 9) e *penduras( )* (IC 11).

Regra R4d (mudança de estado). Todo IC que causa mudança no estado do sistema produz uma operação para realizar essa mudança e para produzir o fluxo de saída (se existir), a menos que uma operação construtora, resultante da aplicação da regra R4a, realize toda a mudança de estado requerida, e não exista saída a produzir.

No sistema *Restaurante*: operações *cancelar( )* (IC 2), *pedirNota( )* (IC 3), *pagarNota( )* (IC 4), e *pendurarNota( )* (IC 5).

Regra R4e (Alocação das operações). Toda operação gerada pela aplicação de uma das regras R4b, R4c ou R4d a um IC, deve ser alocada em uma das classes alcançadas pelos identificadores presentes na interface informacional do IC, ou na classe que representa o sistema[34].

Os seguintes padrões sintáticos podem auxiliar a alocação das operações determinadas pela regra R4b, nas classes:

Padrão sintático R4e-P1. Interface informacional sem identificadores no nível mais externo (de repetição)[35] ou, caso existam, se forem condicionais. Heurística: Alocar a operação na classe que representa a aplicação.

Padrão sintático R4e-P2. Identificador presente, juntamente com o item gerador da operação, no escopo de uma repetição em relação a outro identificador. Heurística: Alocar a operação na classe de associação determinada por esses dois identificadores, se ela existir.

---

[33] Já tratado na regra R4b.
[34] Para manter a coesão das demais classes.
[35] Nível da repetição mais externa, caso existam repetições aninhadas. Se não, o fluxo só tem 1 nível, e ele é o mais externo.

Padrão sintático R4e-P3. Um ou mais identificadores no mesmo nível de repetição em que se encontra o item gerador da operação. Heurística: Alocar a operação em uma das classes alcançadas por esses identificadores.

Padrão sintático R4e-P4. Identificador(es) presente(s) apenas no fluxo de entrada. Heurística: Alocar a operação em uma das classes alcançadas pelos identificadores do nível (de repetição) mais externo do fluxo.

No sistema *Restaurante*: operações *vl_consumo( )*, *vl_realiz( ), receita_pend( )*, *receita_txServ( )*, *receita_total( )*, e *vl_pend( )* (as 5 primeiras provenientes do IC 9, e a última do IC 11), alocadas na classe *Restaurante* (pelo padrão **R4e-P1**); *vl_nota* (IC 3), *quant_item( )* (IC 8), e *vl_pendCli( )* (IC 11), alocadas nas classes *Pedido*, *Item* e *Cliente*, respectivamente (pelo padrão **R4e-P3**); e *troco( )* (IC 4), alocada na classe *Pedido* (pelo padrão **R4e-P4**). O grau de acerto proporcionado pelos padrões sintáticos foi de (11/11) = 100%.

Os seguintes padrões sintáticos podem auxiliar a alocação das operações determinadas pelas regras R4c e R4d, nas classes:

Padrão sintático R4e-P5. Interface informacional sem identificadores no nível mais externo, ou caso existam, se forem condicionais ou estiverem dentro do escopo de uma repetição não-unitária. Heurística: Alocar a operação na classe que representa a aplicação.

Padrão sintático R4e-P6. Identificador não-condicional presente no nível mais externo do fluxo de entrada. Heurística: Alocar a operação em uma das classes alcançadas por esse identificador.

Padrão sintático R4e-P7. Identificador apenas no fluxo de saída, no nível mais externo do fluxo e fora do escopo de repetição não-unitária. Heurística: Alocar a operação em uma das classes alcançadas pelo identificador.

No sistema *Restaurante*, as operações determinadas pela regra R4c são assim alocadas nas classes: operações *consumo_dia( )* (IC 8), *receita( )* (IC 9) e *penduras( )* (IC 11), alocadas na classe *Restaurante* (pelo padrão **R4e-P5**). O grau de acerto proporcionado pelos padrões sintáticos foi de (3/3) = 100%.

No sistema *Restaurante*, as operações determinadas pela regra R4d são assim alocadas nas classes: operações *cancelar( )* (IC 2), *pedirNota( )* (IC 3), *pagarNota( )* (IC

4), e *pendurarNota( )* (IC 5), alocadas na classe *Pedido* (pelo padrão **R4e-P7**). O grau de acerto dos padrões sintáticos foi de (4/4) = 100%.

A classe que representa o sistema pode ser vista como um repositório de operações a serem transferidas para classes de controle (JACOBSON *et al.*, 1992), a serem introduzidas em uma fase posterior da análise do sistema. Todas as demais classes do MD tendem a ter alta coesão semântica (HENDERSON-SELLERS, 1996), já que elas correspondem a abstrações utilizadas pelos *stakeholders* no seu dia-a-dia de trabalho. Pelo mesmo motivo, o acoplamento entre classes tende a refletir as relações que existem entre as abstrações do domínio do sistema.

A visibilidade das operações deve ser pública para os atores autorizados a desempenhar os ICs correspondentes, uma vez que elas se originam diretamente do modelo de requisitos do sistema. As operações determinadas pelos itens derivados não persistidos (regra R4b) retornam o valor calculado para o item. Os argumentos das operações podem ser obtidos a partir dos itens presentes no fluxo de entrada dos ICs. A especificação (tipo e valor inicial) dos argumentos das operações é feita com base nas informações de tipo e domínio do item correspondente, constantes no dicionário de itens elementares. Por exemplo, a partir da interface informacional (Figura 5.2) e do dicionário do IC 3 (Tabela 5.1), chega-se às especificações: *pedirNota(in cliente: Cliente)* e *vl_nota( ): Currency*.

A Figura 5.3 mostra o resultado da aplicação das regras discutidas nessa seção à interface informacional do sistema *Restaurante* (seção 5.3).

5.5 Análise da Solução

5.5.1 Qualidade das Especificações Obtidas

O MIC proposto é basicamente igual ao artefato produzido na MUC, exceto pela inserção de um conjunto de expressões de dados para especificar a interface informacional de cada UC, construídas utilizando a linguagem semiformal apresentada na seção 5.3. Na presente seção, vamos examinar os efeitos dessa diferença, sobre a qualidade da especificação de requisitos resultante.

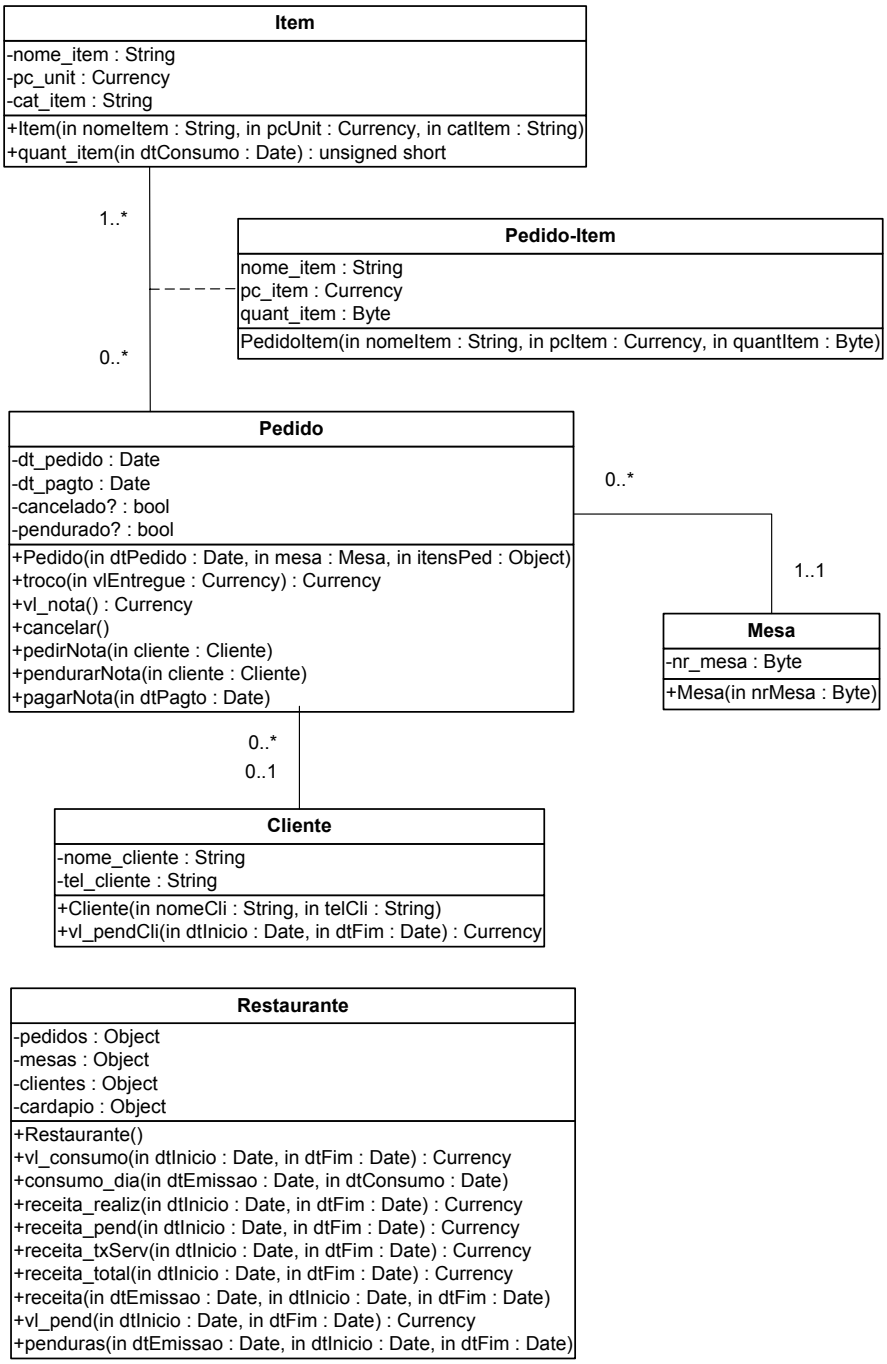


Figura 5.3 – Visão do MD do sistema Restaurante

Devido à semelhança entre os produtos das duas modelagens, é de se esperar que o modelo proposto herde muito das qualidades do modelo de UCs. Portanto, a análise a seguir se concentrará nos aspectos de qualidade que, a nosso ver, são impactados pela solução proposta.

Como o MIC basicamente acrescenta algo (a especificação da interface informacional dos UCs) ao MUC, é clara a necessidade de avaliarmos o impacto disso

sobre a *usabilidade* e a *legibilidade* do MIC. Algum impacto certamente há, pois agora os usuários e *stakeholders* devem ler e compreender uma especificação escrita em uma linguagem semiformal. Entretanto, acreditamos que esse impacto é pequeno se comparado aos benefícios da utilização do MIC. A experimentação com o MIC, relatada no capítulo 6, demonstrou que mesmo pessoas sem maior conhecimento em computação, são capazes de assimilar e usar, mediante um breve treinamento, essa linguagem e as especificações dela resultantes. LAUESEN (2002) a considera uma linguagem aceitável para muitos usuários, e afirma que mesmo aqueles sem formação específica em tecnologia da informação acham as especificações produzidas mais fáceis de compreender do que modelos E/R. Por último, existe a possibilidade de parafrasear, automaticamente, as descrições formais em linguagem natural, tornando-as acessíveis a leitores menos afeitos a formalismos.

Outro aspecto a considerar é a *completude* dos modelos produzidos com a técnica de ICs – o modelo de ICs (MIC) e o MD dele derivado. Sendo um modelo de requisitos, o MIC pode ser considerado a primeira representação dos requisitos dos *stakeholders* para o sistema e, portanto, inexiste uma especificação anterior em relação a qual se possa apurar a completude (externa) do MIC. No caso do MD, podemos avaliar a sua completude em relação ao MIC. Para isso, podemos utilizar a seguinte definição adaptada de GLINZ (2000) (seção 4.4): um modelo está (*parcialmente*) *completo* em relação ao outro quando ele contém todas as informações que são requeridas pelas informações presentes no outro modelo. Na técnica de ICs, boa parte dessa completude é alcançada por construção, ou seja, decorre da derivação do MD a partir das regras descritas na seção 5.4.2, conforme análise realizada mais adiante (seção 5.5.5). Outra avaliação desse aspecto de qualidade foi feita no capítulo 6, dessa vez experimentalmente, através do Estudo 3 (EC-3, seção 6.4), resultando em evidências de que o MD gerado com as regras é adequado (consistente e completo) em relação ao MIC. Das 25 alterações sofridas pelo MD originalmente obtido pela aplicação das regras de derivação, ao longo do projeto e implementação do sistema, apenas 3 visaram corrigir situações de incompletude (omissão) no MD em relação ao MIC (Tabela 6.18). Duas dessas situações resultaram de uma imprevisão da regra R4a (responsável pela inclusão de operações construtoras nas classes), e a outra situação decorreu de uma omissão da técnica, que deveria orientar o modelador a considerar a existência implícita,

no MIC, de consultas do tipo *retrieve*[36], na hora de aplicar a regra R3a (responsável pela persistência de itens informados, na entrada). Conseqüentemente, as devidas ações corretivas foram tomadas para que tais situações não voltem a acontecer.

A *consistência* é outro aspecto de qualidade reforçado na técnica de ICs. Aqui, podemos considerar a consistência intermodelos (entre o MIC e MD) e a consistência interna (dentro de cada modelo). A consistência entre o MIC e MD está resolvida, em parte, pela característica de modelo integrado da solução alcançada neste trabalho. O mesmo pode ser dito com relação à consistência interna do MD, já que ele resulta, em grande parte, da aplicação das regras de derivação, que não produzem elementos inconsistentes entre si. Resta, então, considerar a consistência interna do MIC. Ela também é reforçada, pois a especificação semiformal da interface informacional dos ICs permite uma verificação de completude funcional em um nível de detalhe e precisão maiores do que no MUC, possibilitando na descoberta de lacunas e omissões que, normalmente na MUC, passariam despercebidas. Por exemplo, toda informação que entra em um IC deve ser utilizada no processamento desse IC, ou de outro qualquer; caso contrário, a informação não precisa entrar no sistema, ou então está faltando especificar sua utilização. O estudo experimental EC-3 (seção 6.4) também produziu evidências da consistência do MD gerado a partir do MIC. Das 25 alterações sofridas pelo MD originalmente obtido pela aplicação das regras, ao longo das etapas de projeto e implementação do sistema objeto do estudo, nenhuma visou corrigir uma situação de inconsistência do MD em relação ao MIC (Tabela 6.18).

Outro aspecto de qualidade a considerar é a *rastreabilidade*. Prover rastreabilidade entre o MIC e o MD é mais fácil do que entre o MUC e o MD. Isso porque o MD está intimamente integrado ao MIC, a ponto de poder ser, em grande parte, derivado deste através das regras de derivação apresentadas na seção 5.4.2. Essas regras exploram rastros formais entre elementos dos dois modelos. Portanto, a necessidade de se inserir, manualmente, rastros entre o MIC e seu MD, é menor do que entre o MUC e seu MD.

Por fim, e em decorrência do favorecimento da rastreabilidade que a técnica de ICs proporciona, podemos considerar também favorecida a manutenibilidade do conjunto de modelos dela resultantes (MIC e MD). Além disso, as regras orientam a propagação de modificações feitas em um modelo, em modificações correspondentes no

[36] Uma das operações CRUD – *Create*, *Retrieve*, *Update*, Delete.

outro modelo. Do MIC para o MD, parte dessa propagação pode ser feita automaticamente (seção 5.5.5).

5.5.2 Aplicabilidade do Modelo Integrado

O MIC se aplica principalmente a sistemas interativos e, em especial, a sistemas de informação. Isso pode ser melhor compreendido considerando-se os tipos de sistemas classificados segundo o grau de interação e a quantidade de informação envolvidos na comunicação com o seu ambiente. A Figura 5.4 mostra essa classificação, discutida a seguir.

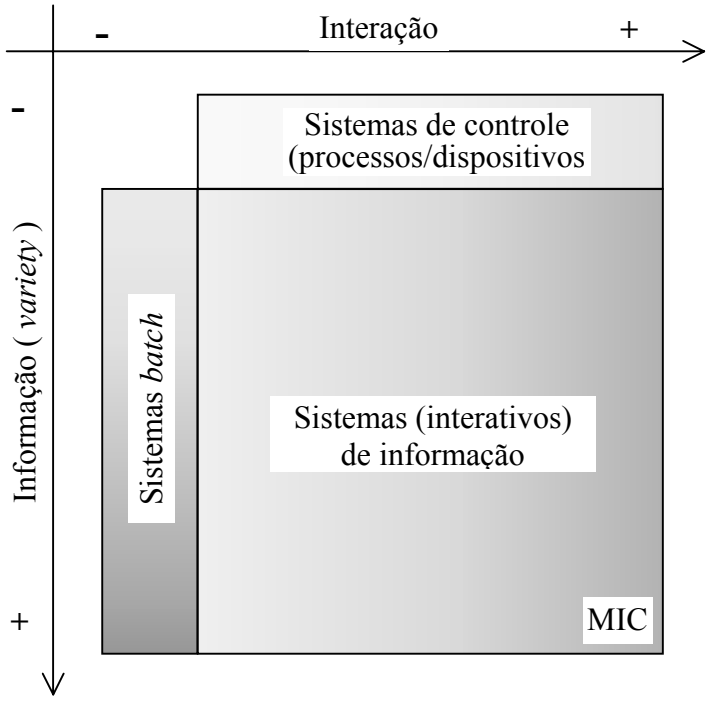


Figura 5.4 – Aplicabilidade da MIC

Sistemas de processamento em lote. São sistemas caracterizados por um baixo nível de interação; mais especificamente, a interação do sistema com o seu ambiente ocorre apenas em dois momentos - no início da execução com a entrada dos dados a processar, e no fim da execução com a saída dos resultados. A modelagem informacional nesse caso não produz particionamento funcional, pois identifica apenas um evento - aquele correspondente a ativação inicial do sistema. Conseqüentemente, o sistema é visto como constituído de um único processo, responsável pela sua execução, do início ao fim. Outra característica desse tipo de sistema é não demandar armazenamento persistente de dados, o que é facilmente percebido na modelagem informacional pelo fato de existir apenas um evento externo. Ou seja, esse tipo de

sistema prescinde da capacidade da MIC em derivar um modelo de (classes de) objetos para o sistema.

Sistemas de controle de processos e/ou dispositivos. Esses sistemas se caracterizam pela predominância da informação de controle na comunicação do sistema com o seu ambiente. Segundo a Teoria da Informação, **informação** é a quantidade de ***variety*** (variedade, diversidade) em uma comunicação, sendo *variety* o número de estados que uma situação pode ter (HAY, 2003, pp. 210). A comunicação de **informação de controle** seria, então, uma comunicação com baixa quantidade de *variety.* Com base na cibernética – a ciência da comunicação e do controle, podemos afirmar que a quantidade de *variety* comunicada pelo sistema ao seu ambiente deve igualar a quantidade de *variety* que pode ser absorvida por esse ambiente (HAY, 2003, pp. 214). Certamente, o ambiente dos sistemas de controle, formado pelos processos e dispositivos controlados (sensores e ativadores), tem capacidade de absorção de *variety* muito menor do que a capacidade dos atores humanos que interagem com um sistema de informação. Portanto, podemos considerar que a informação comunicada pelo sistema aos processos e dispositivos controlados é informação de controle, ou seja, informação filtrada para poder ser utilizada por eles. Essa característica de baixa quantidade de *variety* das informações de controle também significa que, uma vez gerada, tal informação será praticamente de uso exclusivo do ambiente, tendo pouca ou nenhuma utilidade dentro do próprio sistema[37]. Ou seja, muito provavelmente não será necessário persisti-la entre os eventos externos e, portanto, o MD obtido segundo a MIC será bastante reduzido ou mesmo inexistente. Novamente, uma subutilização da MIC.

Sistemas (interativos) de informação. Esses são os sistemas caracterizados por um grau mais elevado de interação com o seu ambiente (principalmente atores humanos), e também por comunicar ao ambiente informação com maior *variety*. Essas características exploram ao máximo a capacidade modeladora da MIC, tanto no particionamento funcional do sistema em processos, quanto na derivação de um modelo de (classes de) objetos não trivial. Os sistemas (interativos) de informação são, portanto, o principal alvo para aplicação da modelagem informacional de requisitos.

A linha divisória entre os sistemas de controle e os sistemas de informação não é necessariamente tão precisa como dá a entender a Figura 5.4. Muitas vezes, um sistema

[37] Normalmente, pode-se estabelecer um mapeamento entre uma informação de controle e a ação correspondente a ser tomada pelo receptor da informação.

de controle também possui partes que podem ser caracterizadas como sendo subsistemas de informação, por comunicarem, ao seu ambiente, informações com alta *variety*, destinadas a atores humanos que com ele interagem.

5.5.3 Análise Essencial: Uma Idéia Precursora

O modelo de info cases (MIC) proposto se baseia no particionamento da funcionalidade do sistema induzido pelos critérios apresentados na seção 5.2. A topologia do particionamento do sistema em processos, segundo esses critérios (Figura 5.1 – seção 5.2), é a mesma topologia que caracteriza o particionamento do sistema nas *atividades essenciais* da Análise Essencial (AE) de Sistemas (MCMENAMIM, PALMER, 1991). Nessa topologia, os processos só se comunicam através de uma memória compartilhada. Portanto, é interessante analisar qual seria a relação entre as duas abordagens (AE e o MIC).

Na AE, o particionamento é feito por eventos gerados pelo ambiente, fora do controle do sistema, dividindo o mesmo em *atividades essenciais*. Tais eventos podem ser de dois tipos: (1) eventos iniciados por entidades do ambiente (denominados *eventos externos* por MCMENAMIM *et al*. (1991, pp. 20), e *eventos sinalizados por fluxos reais* por MAFFEO (1992, pp. 182)); ou (2) eventos iniciados pela passagem do tempo (denominados *eventos temporais* por MCMENAMIM *et al*. (1991), e *eventos sinalizados por fluxos virtuais* por MAFFEO (1992)). Nesse particionamento, a cada evento corresponde uma atividade essencial que traduz a resposta do sistema, planejada para o evento. O conjunto das atividades essenciais são todas as tarefas que o sistema deve realizar, mesmo considerando a utilização de uma *tecnologia perfeita*[38] para sua implementação (MCMENAMIM, PALMER, 1991, pp. 23). As atividades essenciais se subdividem em *atividades fundamentais*, que são parte da finalidade do sistema, e em *atividades custodiais*, responsáveis por estabelecer e manter a *memória essencial*[39] do sistema.

O particionamento do sistema empregado na AE inspirou, inicialmente, a adoção do conceito de particionamento por eventos autônomos, e do ponto de vista da sua

[38] Com tecnologia perfeita, um sistema teria um processador e um meio de armazenamento perfeitos (MCMENAMIM, PALMER, 1991). Hoje, com a disseminação dos sistemas distribuídos, deveríamos acrescentar um canal de comunicação perfeito.

[39] Dados armazenados, produzidos pelo sistema ou capturados do mundo exterior, necessários à execução das atividades fundamentais do sistema (MCMENAMIM, PALMER, 1991, pp. 29).

topologia, podem ser considerados equivalentes. Entretanto, existem outras características específicas de cada particionamento, que os diferenciam.

Os eventos autônomos da MIC são basicamente os eventos da AE. Ambos são gerados pelo ambiente do sistema modelado, e ambos pressupõem certa independência de ordem entre os eventos, embora na MIC essa independência seja especificada de uma forma mais precisa, a saber:

- AE: eventos diferentes podem ocorrer em qualquer seqüência (MCMENAMIM, PALMER, 1991, pp. 86);
- MIC: nenhuma suposição geral pode ser embutida no sistema sobre o evento que antecede a um evento autônomo.

A AE considera que a resposta do sistema a um evento deve deixar o sistema inativo. Na MIC, esse conceito ganha uma interpretação precisa e um significado especial: a resposta do sistema a um evento autônomo deve deixar o sistema em um estado estável, ou seja, um estado no qual inexiste qualquer necessidade de retorno (*rollback*) a um estado anterior, mesmo se nenhum outro evento ocorrer.

A introdução do conceito de estado estável possibilitou a transformação da MUC em um modelo integrado, capaz de capturar, além do comportamento externo já coberto pelo UC tradicional, elementos estruturais e de comportamento interno, e assim, permitir a definição de um conjunto de regras para derivar um MD como uma visão do modelo integrado. Na AE, tal expediente não existe.

Outra vantagem introduzida com a MIC é a sinergia entre os critérios de estados estáveis e de eventos autônomos. Enquanto o primeiro possibilita uma triagem dos estados considerados na modelagem de domínio (estados persistentes e estáveis), o segundo evita que alguns desses estados passem despercebidos devido a uma decomposição insuficiente do sistema nos seus UCs.

O MIC também introduz, através do critério de todo IC levar ao alcance de um objetivo de algum *stakeholder*, uma mudança no significado dos processos oriundos do particionamento do sistema. Eles deixam de ser meras atividades do sistema e passam a representar objetivos dos *stakeholders*. De cara, isso resulta em uma simplificação importante para a elicitação dos requisitos do sistema: os processos correspondentes às atividades custodiais são desconsiderados, pois não representam objetivos dos *stakeholders*. Essas atividades, embora importantes na análise de um sistema, não são

relevantes para a especificação dos seus requisitos. Além disso, conforme vimos na seção 5.4 (Derivação da Visão MD do Modelo Integrado), é possível determinar a informação a ser persistida, e o momento em que isso deve ocorrer, unicamente a partir dos processos elicitados segundo os critérios da MIC.

Portanto, vemos que o NIO (Nível Informacional de Objetivos – seção 5.2) não é exatamente o nível das atividades essenciais da AE, mas uma simplificação (abstração) deste. Essa simplificação é significativa, pois segundo a avaliação dos próprios criadores da AE, as atividades custodiais ocorrem muito mais freqüentemente do que as atividades fundamentais (puras) (MCMENAMIM, PALMER, 1991, pp. 83). Essa e outras diferenças entre o MIC e a AE (como por exemplo, a possibilidade de decompor as atividades essenciais) decorrem do escopo distinto de cada uma: o MIC proposto visa a modelagem de requisitos e a análise do sistema no nível do domínio de aplicação, enquanto a AE foca, principalmente, a análise do sistema nos níveis lógico e físico.

MCMENAMIM e PALMER (1991, pp. 85) apontam vantagens importantes para o particionamento que propõem:

- Fornece resultados razoavelmente uniformes, independentemente de quem particiona o sistema;
- Resulta em uma modelagem concisa (número relativamente pequeno de processos);
- A interface entre os processos tende a ser mínima; e
- Resulta em processos fiéis à essência do sistema (sem interdependência temporal artificial).

MAFFEO (1992, pp. 301) acrescenta ainda as seguintes vantagens desse particionamento:

- Minimiza os riscos de paralisia por análise, comum na estratégia de particionamento estritamente *top-down*;
- Facilita a manutenção do sistema;
- Favorece o reuso de modelos (reuso da essência de sistemas).

Pela relação acima identificada entre os particionamentos das duas abordagens (AE e MIC), acreditamos que as mesmas vantagens da AE podem ser esperadas, e mesmo potencializadas, no MIC proposto.

5.5.4 O MIC e a Análise de Requisitos Orientada a Objetivos

Na Modelagem de requisitos Orientada a Objetivos – MOObj (LAMSWEERDE, 2001), os objetivos devem ser decompostos até o nível em que eles possam ser atribuídos a um agente individual, seja do sistema (requisitos) ou do ambiente do sistema (suposições). No MIC, os objetivos também correspondem a objetivos de um agente individual. Entretanto, o MIC abstrai os objetivos de agentes do sistema, capturando apenas os objetivos dos atores, ou seja, apenas os objetivos dos agentes do ambiente, que interagem com o sistema.

Outra restrição do MIC é o nível de abstração em que os objetivos dos atores são considerados: o NIO (seção 5.2). Como um objetivo do NIO também é atribuído a um ator individual, à primeira vista parece que o NIO coincide com o nível mínimo de abstração adotado na MOObj. Entretanto, se considerarmos que um objetivo informacional pode subassumir vários outros objetivos (seção 5.2), vemos que a MOObj pode, em princípio, incluir objetivos de nível inferior ao NIO.

Outra diferença é que, enquanto a MOObj possibilita a análise tanto de objetivos funcionais (*hard goals*) quanto de não-funcionais (*soft goals*), o MIC (assim como a MUC) foca basicamente objetivos funcionais.

A relação da MOObj com o MIC é de complementaridade; enquanto a MOObj está voltada principalmente para *early requirements*, o MIC (assim com a MUC) trata *late requirements*. Uma conclusão análoga é apresentada em (MYLOPOULOS *et al.*, 1999): "... a Análise Orientada a Objetivos e a Análise Orientada a Objetos devem ser vistas como complementares, a primeira focando os estágios iniciais (*early stages*) da análise de requisitos e a racionalização do processo de desenvolvimento, e a segunda focando os estágios posteriores (*late stages*) da análise de requisitos."

O MIC pode ser visto como um passo intermediário – uma ponte – entre a modelagem organizacional realizada via MOObj e a AOO (Análise Orientada a Objetos). Se, por um lado, suas características restritivas (objetivos de atores, *hard goals*, e NIO) não estão voltadas para o tratamento específico de, por exemplo, requisitos não-funcionais e conflitos, bem contemplados na MOObj, por outro lado, são exatamente essas restrições que fazem o MIC ser mais facilmente integrável à AOO, possibilitando a derivação semi-automática de um MD.

Nem sempre é necessário realizar modelagem organizacional, embora isso fique cada vez mais comum devido à complexidade e à rapidez de evolução dos sistemas atuais. Mas, se o sistema tiver complexidade menor, ou estiver inserido em um ambiente organizacional já bem estudado, tal modelagem pode ser dispensada, partindo-se logo para o MIC.

Uma sugestão de integração da MOObj com o MIC é identificar, dentre os objetivos elicitados na MOObj, aqueles pertencentes ao NIO para, então, realizar os passos subseqüentes de elaboração do MIC (seção 5.3). Dessa forma, sugerimos um processo de modelagem que faz decomposição e composição de objetivos segundo uma estratégia em *y* (espelhado): inicialmente *top-down* (na MOObj) e depois *middle-up* (na MIC). Enquanto o detalhamento na MOObj prioriza as relações entre objetivos, a MIC se dedica a detalhar principalmente os próprios objetivos, detalhando a sua interface informacional. Sendo duas dimensões complementares e (em certo grau) ortogonais de detalhamento, esperamos uma sinergia capaz de gerar aprimoramentos em cada um dos modelos.

5.5.5 Grau de Automatismo das Regras e Completude do MD Derivado

Nesta seção são analisados o grau de automatismo das regras apresentadas na seção 5.4.2 e a completude do MD com elas obtido a partir o MIC, conforme definições a seguir. Ao final da seção, a Tabela 5.2 resume as principais características de cada regra.

O *grau de automatismo de uma regra* é a relação entre o número de ações automatizáveis e o número total de ações envolvidas na aplicação da regra.

O *MD está completo em relação ao MIC* quando ele contém todas as informações que são requeridas por informações presentes no MIC. Esta definição se baseia naquela dada por GLINZ (2000) (seção 4.4). Definição análoga pode ser dada para a completude do MIC em relação ao MD.

Regra R1 (Determinação de classes). Cada identificador gerado dá origem a uma classe de objetos.

Procedimento (para aplicação da regra): Obter o conjunto dos identificadores gerados, presentes nos ICs. O MD conterá uma classe para cada um desses identificadores, nomeada com a parte do nome do identificador que segue ao *id_*.

Regra semi-automatizável

Ações automatizáveis:

(1) Determinação da condição de um item ser identificador: são aqueles itens elementares cujo nome se inicia com a seqüência de caracteres *id_*;

(2) Nomeação da classe correspondente a um identificador: parte do nome do identificador, seguinte à seqüência de caracteres *id_* .

Ações não-automatizáveis:

(1) Verificação da condição de um identificador estar sendo gerado no IC.

Grau de automatismo: 2/3 (**67%**)

Análise de completude:

Todo identificador presente no MIC representa um objeto que provê informações (de entrada, na forma de atributos, ou derivadas, na forma de operações) necessárias ao desempenho da funcionalidade do sistema que está sendo modelado. Como ICs são UCs no Nível Informacional de Objetivos (NIO - seção 5.2), tais objetos correspondem a abstrações de domínio a serem persistidas nos estados estáveis do sistema e, portanto, devem aparecer como classes no MD do sistema. A regra R1 garante isso, e conseqüentemente, o MD dela resultante é completo em relação ao MIC. Por outro lado, como todas as classes do MD são obtidas via regra R1, não haverá uma classe no MD sem que exista um identificador correspondente no MIC. Assim, o MIC também é completo em relação ao MD.

Outra constatação que pode ser feita é a de que o MD só precisa conter classes produzidas pela regra R1. Se uma classe no MD não possui um identificador correspondente no MIC, então nenhuma informação (informada ou derivada) poderia ser recuperada dos objetos dessa classe, entre estados estáveis do sistema (pois, para isso, é necessário existir um identificador para o objeto, no MIC). Conseqüentemente, ou os objetos da classe são transientes (não precisam ser persistidos) ou não são necessários ao desempenho da funcionalidade do sistema. Em ambos os casos, a classe não precisa fazer parte do MD (pelo menos, em uma visão minimalista).

Regra R2 (Determinação de associações). As associações entre classes estão entre aquelas indicadas pelos pares (não-ordenados) de identificadores presentes na interface

informacional de cada IC (para fluxos de saída, basta considerar identificadores gerados)[40].

Procedimento (para aplicação da regra): Em cada IC, cada par (não-ordenado) de identificadores deve ser analisado pelo modelador para decidir sobre a criação ou não, no MD, de uma associação entre as classes nomeadas por esses identificadores.

Regra **semi-automatizável**

Ações automatizáveis:

(1) Determinação dos pares (não-ordenados) de identificadores na interface informacional (II) dos ICs;

Ações não-automatizáveis:

(1) Decisão sobre a criação ou não de uma associação entre classes no MD, para um par de ocorrências dos identificadores correspondentes, em um IC;

(2) Verificação da condição de um identificador estar sendo gerado no IC onde ele ocorre (no fluxo de saída). Entretanto, esta ação já foi executada durante a aplicação da regra R1, sendo seus resultados passíveis de serem reutilizados aqui.

Grau de automatismo: 1/2 (**50%**) ou 1/3 (33%) caso não seja reutilizado o resultado da ação não-automatizável (2), obtido com a aplicação da regra R1.

Análise de completude:

O MIC indica, através da ocorrência de mais de um identificador na II de um IC, a possível existência de associações entre instâncias de abstrações (objetos) de domínio referenciadas pelos identificadores. Pela aplicação da regra R2, todas as associações entre objetos, elicitadas no MIC, são mapeadas para associação entre abstrações (classes) no MD. Portanto, o MD é completo em relação ao MIC, com respeito a associações entre classes. Por outro lado, como todas as associações no MD são obtidas via regra R2, não haverá uma associação no MD sem que ocorra pelo menos um par correspondente de identificadores na II de algum IC do MIC. Assim, o MIC também é completo em relação ao MD.

O fato do MD só conter associações produzidas pela regra R2 pode ser justificado como segue. Se para uma associação entre classes no MD, os identificadores

[40] Pois os demais não contribuem para a descoberta de associações inéditas (ou seja, que não possam ser geradas a partir das demais ocorrências de identificadores na II dos ICs).

das classes não aparecerem juntos na II de nenhum dos ICs, significa que a especificação da funcionalidade do sistema prescinde de qualquer associação entre os objetos dessas classes e, portanto, que pelo menos numa visão minimalista, a associação no MD é supérflua.

Regra R3a (Persistência - item informado, na entrada). Todo item elementar não-identificador e informado, presente no fluxo de entrada de um IC, e necessário em outro IC, precisa ser persistido; caso contrário (se não for necessário em outro IC), não deve ser persistido[41].

Procedimento (para aplicação da regra): Em cada IC, cada item elementar não-identificador e informado no IC deve ser analisado pelo modelador para verificar se o valor de entrada que ele representa é necessário (utilizado[42]) em algum outro IC. Se for, deve ser marcado como um item a persistir como atributo de alguma classe (a determinar - vide regra R3d); se não, o item não precisa ser persistido e, portanto, não gera um atributo de classe no MD. Além disso, deve-se realizar a seguinte verificação de consistência do MIC: se um item não-identificador e informado não for persistido, ele deve estar sendo utilizado dentro do próprio IC onde entra, seja aparecendo no fluxo de saída ou sendo utilizado no cálculo de algum item de saída deste IC (direta ou indiretamente). Caso contrário, pelo menos em uma visão minimalista, existe uma inconsistência no MIC, pois uma informação que entra no sistema não é utilizada para nada no mesmo.

Regra semi-automatizável:

Ações automatizáveis:

(1) Determinação dos itens elementares, não-identificadores e de entrada, em cada IC;

(2) Verificação de que um item aparece tanto no fluxo de entrada quanto no fluxo de saída de um IC (pois a cada IC está associado um único espaço de nomes para os itens que lá aparecem).

Ações não-automatizáveis:

(1) Verificação da condição do item ser necessário em um IC distinto daquele que está sendo considerado (onde o item aparece como entrada);

---

[41] Já que estamos buscando um MD mínimo, ou seja, um MD que provê apenas o que é estritamente necessário ao suporte da funcionalidade descrita no MIC.

[42] Como informação de saída (no fluxo de saída) ou no cálculo de outro item.

(2) Verificação de que um item elementar não-identificador e de entrada em um IC, é utilizado (direta ou indiretamente) no cálculo (obtenção) de um item de saída desse IC.

Grau de automatismo: 2/4 (**50%**)

Análise de completude: (após a regra R3c).

Regra R3b (Persistência – item informado, na saída, valor histórico). Todo item elementar não-identificador e informado, presente no fluxo de saída de um IC, representando informação histórica[43] cujo valor não pode, em geral, ser restaurado nesse IC, precisa ser persistido; caso contrário, não deve ser persistido.

Procedimento (para aplicação da regra): Em cada IC, cada item elementar não identificador e informado, presente no fluxo de saída do IC, deve ser analisado pelo modelador para verificar se o valor que ele representa é uma informação histórica não passível de restauração (em geral) naquele IC. Se este for o caso, o item constitui nova informação que precisa ser persistida como atributo de alguma classe (a ser determinada – vide regra R3d) e, portanto, deve ser marcado como tal; caso contrário (se o item não é informação histórica, ou sendo, seu valor pode ser restaurado no IC), ele não deve ser marcado como um item a persistir.

Regra semi-automatizável:

Ações automatizáveis:

(1) Determinação dos itens elementares não-identificadores e de saída, em cada IC;

Ações não-automatizáveis:

(1) Verificação da condição de um item elementar não-identificador e de saída ser informado (em contraposição a ser derivado);

(2) Verificação da condição de um item informado, presente no fluxo de saída de um IC, representar uma informação histórica não restaurável nesse IC.

Grau de automatismo: 1/3 (**33%**)

Análise de completude: (após a regra R3c).

[43] Informação relativa a um estado anterior do sistema, que possivelmente não mais vigora com o mesmo valor no estado atual.

Regra R3c (Persistência - item derivado, valor histórico). Todo item elementar não-identificador e derivado, presente (no fluxo de saída) em um IC, representando informação histórica cujo valor não pode ser restaurado nesse IC, precisa ser persistido; caso contrário, não deve ser persistido.

Procedimento: (para aplicação da regra): Para cada item elementar não-identificador e derivado, presente no fluxo de saída de um IC, o modelador deve verificar se o valor que ele representa é uma informação histórica não passível de restauração (em geral) naquele IC. Se esse for o caso, o item constitui informação que precisa ser persistida como atributo de alguma classe (a ser determinada – vide regra R3d) e, portanto, deve ser marcado como tal; caso contrário (se o item não é informação histórica, ou sendo, seu valor pode ser restaurado no IC) ele não dever ser marcado como um item a persistir.

Regra semi-automatizável

Ações automatizáveis:

(1) Determinação dos itens elementares não-identificadores e de saída, em cada IC;

Ações não-automatizáveis:

(1) Verificação da condição de um item elementar não-identificador e de saída ser derivado (em contraposição a ser informado);

(2) Verificação da condição de um item derivado em um IC representar uma informação histórica não restaurável nesse IC.

Grau de automatismo: 1/3 (**33%**)

Análise de completude (regras R3a, R3b e R3c):

Todo item elementar não-identificador presente no MIC, informado ou derivado, cuja persistência seja necessária entre estados firmes do sistema, será mapeado pelas regras R3a, R3b ou R3c em um atributo de classe no MD e, portanto, o MD estará completo em relação ao MIC, no que diz respeito aos atributos provenientes da persistência desses itens elementares.

Por outro lado, como todo atributo ordinário[44] resulta do mapeamento de itens elementares não-identificadores, sejam itens informados (regras R3a e R3b) ou itens

[44] Atributo que não é de estado (regra R3e).

derivados (regra R3c), o MIC também estará completo em relação ao MD, no que diz respeito a esses atributos.

O fato do MD só conter atributos ordinários provenientes da aplicação das regras R3a, R3b e R3c pode ser justificado como segue. Se um atributo ordinário, presente no MD, não resultar da aplicação das regras R3a, R3b ou R3c, é porque as três condições seguintes ocorrem:

a) O item de informação correspondente ao atributo não entra em nenhum IC, ou se entra, não é necessário em nenhum IC distinto daquele em que ele entrou (condição que impede a aplicação das regras R3a e R3b); e

b) O item de informação correspondente ao atributo entra em um IC, aparece no fluxo de saída de outro(s) IC(s) como um valor histórico[45], mas esse valor sempre pode ser restaurado (lembrado) nesse(s) outro(s) IC(s) (condição complementar que impede a aplicação da regra R3b); e

c) O item não é calculado pelo sistema, e se for, não é necessário em outro IC distinto daquele em que ele é calculado, ou então pode ser (re)calculado em todo IC onde é necessário (condição que impede a aplicação da regra R3c).

Se o item não entra e não é calculado pelo sistema, então o MD não precisa incluí-lo, pois ele não é utilizado no sistema e, portanto, não precisa compor nenhum estado do mesmo.

Se o item entra ou é calculado em um IC, mas não é utilizado em nenhum outro IC (mas apenas no IC em que ele entra ou é calculado), o item não é necessário à representação de um estado persistente do sistema e, portanto, não precisa entrar no MD, já que o MD representa apenas estados persistentes (e firmes) do sistema.

Se um item necessário em mais de um IC pode ser sempre[46] derivado nesses ICs (a partir de itens informados ou de outros itens derivados), então ele configura um atributo derivado no MD e, portanto, a rigor, não precisa fazer parte do mesmo.

Regra R3d (Alocação de atributo / Criação de classe de associação). Cada item a persistir, presente na interface informacional de um IC, deve ser alocado como atributo de uma das *classes alcançadas pelos identificadores* presentes nessa mesma interface. Caso nenhuma dessas classes comporte o item, ele deve ser alocado em uma nova classe

[45] Informação relativa a um estado anterior do sistema, que possivelmente não mais vigora.
[46] Em qualquer estado estável do sistema.

de associação criada para esse fim, correspondente a uma associação existente entre as classes alcançadas.

Procedimento (para alocação): Para cada item elementar a persistir, presente na interface informacional de um IC, o modelador deve escolher uma das classes alcançadas pelos identificadores presentes nessa interface, para alocar o atributo correspondente ao item. Caso nenhuma dessas classes seja adequada para conter o item como seu atributo, o modelador deve criar uma classe de associação, correspondente a uma das associações existentes entre as classes alcançadas, para alocar o item como atributo.

Regra semi-automatizável

Ações automatizáveis:

(1) Determinação das classes alcançadas pelos identificadores presentes na interface informacional do IC que contém o item a persistir.

(2) Criação e nomeação de uma classe de associação, a partir do nome das classes que participam na associação correspondente.

Ações não-automatizáveis:

(1) Escolha da classe (dentre aquelas alcançadas) para alocar o item;

(2) Escolha de uma associação entre as classes alcançadas, para criação da classe de associação correspondente,

Grau de automatismo: 2/4 (**50%**)

Análise de completude:

Os identificadores presentes na interface informacional de um IC restringem a alocação dos itens a persistir, presentes nessa interface, às classes por eles alcançadas. A regra R3d traduz essa restrição, garantindo assim, que o MD seja completo em relação ao MIC, no que diz respeito à alocação dos itens a persistir como atributos das classes do MD. Por outro lado, como toda alocação de itens de informação a persistir no MD é feita via regra R3d, o MIC também estará completo em relação ao MD.

Se um item de informação puder ser melhor alocado (pelo modelador, de forma *ad hoc*) em uma classe do MD que não seja uma das classes alcançadas pelos identificadores presentes na interface informacional onde o item aparece, o que isso significaria? Se ele está no MD é porque ele precisou ser comunicado entre estados

firmes do sistema, e isso só é possível no MIC utilizando o identificador do objeto que o contém como atributo. Portanto, se o modelador fizer isso, ele estará introduzindo uma contradição entre os dois modelos. Ou seja, se o modelador resolve colocar o atributo em uma classe, a mesma motivação deveria fazer com que ele indicasse, no MIC, o identificador de um objeto dessa classe, toda vez que o item de informação for comunicado entre estados firmes do sistema.

Regra R3e (Atributo de estado). Para cada IC com interface informacional constituída apenas de fluxo de entrada, contendo exclusivamente um único identificador, deve ser introduzido um atributo de estado em uma das classes alcançadas por esse identificador.

Procedimento (para aplicação da regra): automático.

Regra automatizável.

Ações automatizáveis:

(1) Verificação da condição do IC possuir apenas fluxo de entrada constituído exclusivamente de um único identificador.

Grau de automatismo: 1 (**100%**)

Análise de completude:

Não necessariamente todo IC que introduz informação de estado a persistir será mapeado pela regra R3d, por não se encaixar na condição de aplicação dessa regra. Portanto, em geral, o MD não estará completo em relação ao MIC no que diz respeito a atributos de estado.

Regra R4a (construtoras). Toda classe recebe uma operação construtora.

Procedimento: Designar uma operação construtora para cada classe existente no MD (determinada pela regra R1).

Regra automatizável

Ações automatizáveis:

(1) Determinação das classes existentes no MD.

(2) Designação de uma operação construtora para cada classe.

(3) Nomeação da operação construtora (mesmo nome da classe).

Grau de automatização: 1 (**100%**)

Análise de completude:

Cada ocorrência de identificador gerado no MIC representa um objeto criado a partir da classe por ele identificada. Isso exige, no MD, a existência de operações nas respectivas classes, para construir esses objetos. A regra R4a providencia essas operações. Entretanto, é comum a necessidade de se sobrecarregar a operação construtora, em decorrência das diversas formas de se inicializar um objeto de uma classe. No MIC, isso normalmente é indicado pela presença de itens de informação condicionais no fluxo de entrada de ICs que tem identificador gerado no fluxo de saída. Portanto, em geral, o MD não estará completo em relação ao MIC, no que diz respeito a operações construtoras.

Acrescentar uma operação construtora em uma das classes do MD, sem que haja alguma informação condicional especificada no MIC, significaria poder inicializar um objeto dessa classe de uma maneira não prevista no MIC e, portanto, essa operação seria desnecessária ao desempenho da funcionalidade do sistema.

Regra R4b (itens derivados). Todo item derivado não persistido produz uma operação para calcular o seu valor[47].

Procedimento: Designar uma operação para cada item não-identificador, derivado e não-persistido, presente na interface informacional dos ICs, atribuindo-lhe um nome derivado do nome do item.

Regra (semi-)automatizável

Ações automatizáveis:

(1) Determinação dos nomes de itens elementares não-identificadores e de saída, em cada IC (também necessária em R3b, R3c);

(2) Atribuição de nome à operação produzida: mesmo nome do item, ou o nome obtido pela eliminação do caracter "_" e adoção do formato "*camel case*" (para manter a legibilidade do nome). Exemplo: *multa_diaria()* ou *multaDiaria()*.

Ações não-automatizáveis:

(1) Verificação da condição de um item não-identificador e de saída ser derivado no IC (resultado reutilizável de R3c);

[47] Itens derivados persistidos não geram operações, já que eles estão disponíveis como atributos, na interface das classes.

(2) Verificação de que um item derivado não precisa ser persistido (resultado reutilizável de R3c);

Grau de automatismo: 2/2 (**100%**) ou 2/4 (**50%**) no caso não sejam reutilizados os resultados das ações não-automatizáveis (1 e 2), obtidos com a aplicação da regra R3c.

Análise de completude:

Cada item derivado não-persistido representa uma propriedade de algum objeto, cujo valor é passível de ser derivado (calculado) a partir do valor de outras propriedades, nos momentos em que ele é necessário. Do ponto de vista do MD, isso requer que alguma operação da classe do objeto implemente o comportamento correspondente ao cálculo do valor da propriedade. A regra R4b propõe que se introduza uma operação específica na classe do objeto, exclusivamente para o cálculo do seu valor, e assim, garante a completude do MD em relação ao MIC, no que diz respeito a esse tipo de operação. A introdução de operação específica e exclusivamente dedicada ao cálculo do valor do item derivado se justifica pela proximidade semântica com o MIC e pela alta coesão que tal operação apresenta.

Toda operação presente no MD de um sistema deve ser visível e significativa do ponto de vista dos *stakeholders*. É o caso das operações produzidas pela regra R4b. Então, caso uma operação de cálculo de uma informação elementar, presente no MD, não corresponda a um item derivado não-persistido presente no MIC, essa operação seria desnecessária para a especificação da funcionalidade do sistema[48]. Portanto, ela também não precisaria aparecer no MD.

Regra R4c (fluxos de saída). Todo fluxo de saída presente em IC cujo processamento não produz mudança de estado no sistema (ou seja, não possui identificador gerado, nem item persistido ou atributo de estado, e não altera o valor de algum item persistido – tipicamente, ICs de consulta), e que não se resuma a apenas um único item não persistido[49], dá origem a uma operação para produzir o fluxo.

Procedimento: Designar uma operação para cada fluxo de saída de IC de consulta e que não é constituído de apenas um item não-persistido, atribuindo-lhe um nome derivado do nome do fluxo.

---

[48] Supondo que o MIC reflita corretamente os requisitos do sistema.
[49] Já tratado na regra R4b.

Regra semi-automatizável

Ações automatizáveis:

(1) Determinação dos fluxos de saída e de seus nomes;

(2) Verificação de que um fluxo de saída é constituído de apenas um item de informação

(3) Atribuição de nome à operação produzida: mesmo nome do fluxo, ou o nome obtido pela eliminação do caracter "_" e adoção do formato "*camel case*" (para manter a legibilidade). Exemplo: *consumo_diario()* ou *consumoDiario()*.

Ações não-automatizáveis:

(1) Verificação de que o processamento de um IC não produz mudança de estado no sistema (IC de consulta);

(2) Verificação de que um item elementar de informação é não-persistido (resultado reutilizável de R3a, R3b ou R3c).

Grau de automatismo: 3/4 (**75%**) ou 3/5 (**60%**) no caso de não se aproveitar resultados obtidos com a aplicação de outras regras.

Análise de completude: Toda saída representa comportamento que, do ponto de vista do MD, deverá ser desempenhado por uma ou mais de suas operações. Em princípio, o MD já possui operações para a produção ou recuperação do valor dos itens elementares presentes nos fluxos de saída do MIC: itens informados ou derivados (e persistidos) podem ter seu valor recuperado pelas operações (implícitas) de acesso (tipo *get*), enquanto itens derivados não-persistidos podem ter seu valor calculado pelas operações geradas pela regra R4b. Portanto, a rigor, o MD já se encontra completo em relação ao MIC, com respeito ao comportamento de saída em ICs de consulta. O que a regra R4c faz é propor a introdução de uma nova operação capaz de orquestrar a produção e providenciar a exibição do fluxo de saída. Isso se justifica para aproximar o MD ao MIC, o qual tem nos fluxos de saída dos ICs de consulta, um elemento altamente visível aos *stakeholders*.

Por outro lado, uma operação representada por um procedimento cujo único resultado é o efeito colateral de produzir uma saída que não se restringe a um único item derivado não-persistido, requer no MIC a presença de um fluxo de saída composto por tais itens. Se essas operações forem exatamente aquela geradas pela aplicação da regra

R4c, isso estará garantido e, portanto, o MIC será completo em relação ao MD, no que diz respeito às operações desse tipo.

Regra R4d (mudança de estado). Todo IC que causa mudança no estado do sistema produz uma operação para realizar essa mudança e para produzir o fluxo de saída (se existir), a menos que uma operação construtora, resultante da aplicação da regra R4a, realize toda a mudança de estado requerida, e não exista saída a produzir.

Procedimento: Para cada IC que atenda as condições enunciadas na regra, verificar se existe operação construtora resultante da aplicação da regra R4a ao IC; caso exista, verificar se essa operação faz toda a mudança de estado causada pelo IC, e se não existe fluxo de saída nele. Se este não for o caso (ou seja, se não existir operação construtora no IC, ou se ela não produzir toda a mudança de estado causada pelo IC, ou ainda se existir saída a produzir), designar uma operação para realizar (ou completar) a mudança e produzir a saída (se existir), atribuindo-lhe um nome derivado do nome do IC (sugestão: justaposição da primeira palavra – que normalmente é um verbo no infinitivo – com uma ou mais das palavras seguintes, utilizando formatação "*camel case*" para manter a legibilidade. Exemplo: *pagarNota()*.)

Regra semi-automatizável:

Ações automatizáveis:

(1) Verificação de que um IC possui fluxo de saída;

Ações não-automatizáveis:

(1) Verificação de que um IC causa mudança de estado no sistema;
(2) Verificação da existência de operação construtora gerada pela aplicação da regra R4a ao IC (resultado reutilizável da regra R4a);
(3) Verificação de que uma operação construtora, derivada pela regra R4a aplicada ao IC, produz uma dada mudança de estado no sistema;
(4) Atribuição de nome à operação produzida.

Grau de automatismo: 1/4 (**25%**) ou 1/5 (20%) no caso de não se aproveitar resultados obtidos com a aplicação de outras regras.

Análise de completude:

A necessidade de efetuar uma mudança de estado no sistema, ainda não coberta por qualquer operação presente no MD, exige a introdução, nesse modelo, de uma nova

operação, ou então, a extensão da funcionalidade de alguma operação já existente nele. Para manter a boa coesão das operações do MD, é desaconselhável estender as operações construtoras, únicas até agora a realizar mudança de estado no sistema, para incluir a mudança ora necessária. Consistentemente com isso, a regra R4d propõe a introdução de uma nova operação capaz de realizar a mudança requerida no estado do sistema, bem como, se necessário, produzir qualquer saída especificada no IC, garantindo assim a completude do MD em relação ao MIC, no que diz respeito às operações geradoras de mudança de estado no sistema.

Caso o MD possua uma operação para realizar mudança de estado no sistema, que não seja uma operação construtora, deverá existir no MIC um IC que não seja de consulta, responsável pela mudança. Se toda operação desse tipo for resultante da aplicação da regra R4d, isso estará garantido e, portanto, também a completude do MIC em relação ao MD.

Por outro lado, não existe a necessidade do MD possuir uma operação para realizar mudança de estado no sistema que não seja uma operação construtora ou uma operação gerada pela regra R4d. Isso porque, a regra R4d se encarrega de gerar operação para qualquer necessidade de mudança de estado do sistema que não esteja coberta por uma operação construtora.

Regra R4e (Alocação das operações). Toda operação gerada pela aplicação de uma das regras R4b, R4c ou R4d a um IC, deve ser alocada em uma das classes alcançadas pelos identificadores presentes na interface informacional do IC, ou na classe que representa o sistema[50].

Procedimento: Para alocar uma operação resultante da aplicação da regra R4b, R4c ou R4d a um IC, deve-se considerar a lista das classes alcançáveis pelos identificadores presentes na interface informacional do IC. Se não existirem identificadores, a operação deve ser alocada na classe que representa o sistema; caso contrário, o modelador deverá decidir em que classe será feita alocação da operação.

Regra semi-automatizável (no caso geral)

Ações automatizáveis:

(1) Determinação dos identificadores presentes na interface informacional de um IC;

(2) Obtenção da lista de classes alcançáveis por um identificador;

[50] Para manter a coesão das demais classes.

Ações não-automatizáveis:

(1) Determinação da classe a alocar a operação, se a lista de classes candidatas não for vazia.

Grau de automatismo: 2/3 (**66%**) no caso geral; mas na situação especial em que não existe identificador na interface informacional do IC, a aplicação da regra pode ser feita de forma completamente automática (alocar na classe que representa o sistema).

Análise de completude:

Toda operação identificada a partir do MIC precisa estar alocada a uma das classes do MD. A regra R4e atende esse requisito, garantindo a completude do MD em relação ao MIC, no que diz respeito à alocação das operações às classes. Por outro lado, como toda alocação de operações no MD é feita via regra R4e, o MIC também estará completo em relação ao MD.

A Tabela 5.2 dá uma visão geral das principais características das regras de derivação.

5.6 Trabalhos Mais Diretamente Relacionados

As propostas extraídas da literatura, apresentadas na seção 3.3.2, têm em comum com o MIC a motivação: tornar mais efetiva a obtenção de um MD a partir do MUC, ou promover a consistência entre os dois modelos. Entretanto, nenhuma delas adota a estratégia de utilizar um modelo integrado de requisitos, mantendo o MUC como um dos modelos analisados em separado. O resultado é, invariavelmente, a obtenção de MDs bastantes incompletos e/ou uma grande dependência do modelador para interpretar os elementos de cada modelo (baixo automatismo). Nesta seção são discutidos outros dois trabalhos relatados na literatura, bem mais próximos da solução construída nesta tese – o primeiro porque também adota a abordagem de modelo integrado de requisitos; e o segundo porque ataca diretamente o problema da inconsistência entre MDs obtidos por diferentes modeladores, para um mesmo sistema.

Tabela 5.2 – Caracterização das regras de derivação

Regra	Det?	Auto?	Grau (%)	# PSint	Compl
R1(classes)	sim	semi	67	1	MIC↔MD
R2 (associações)	não	semi	50 (33)[51]	8	MIC↔MD
R3a (persistência – itens informados, de entrada)	sim	semi	50	1	MIC↔MD
R3b (persistência – itens informados, de saída)	sim	semi	33	0	MIC↔MD
R3c (persistência – itens derivados)	sim	semi	33	0	MIC↔MD
R3d (alocação de atributos/criação de classe de associação)	não	semi	50	3	MIC↔MD
R3e (atributos de estado)	sim	auto	100	-	MIC→MD
R4a (operações construtoras)	sim	auto	100	-	MIC→MD
R4b (operações – itens derivados)	sim	(semi) auto	100 (50)	0	MIC↔MD
R4c (operações de fluxo)	sim	semi	75 (60)	0	MIC↔MD
R4d (operações de mudança de estado)	sim	semi	25 (20)	0	MIC↔MD
R4e (alocação de operações)	não	semi	66	7	MIC↔MD

Legenda:
Det?: A regra é determinística?
Aut?: A regra é automatizável?
Grau (%): Grau de automatismo (em valores percentuais)
#PSint: Número de padrões sintáticos associados à regra
Compl: Completude relativa entre os MIC e o MD resultante da aplicação da regra:
MIC→MD: O MIC é completo em relação ao MD (mas não vice-versa)
MIC↔MD: Ambos os modelos são completos em relação ao outro

5.6.1 O Projeto ADORA

ADORA (GLINZ *et al.*, 2002) (GLINZ, 2008) é um acrônimo para *Analysis and Description of Requirements and Architecture*. Trata-se de um método de modelagem OO para software, voltado principalmente para a especificação de requisitos, e também para o projeto lógico da arquitetura de um sistema. Tanto pode ser aplicado para sistemas de controle industriais, quanto para sistemas de informação, inclusive sistemas distribuídos.

Segundo GLINZ *et al.* (2002), três das principais idéias em que se baseia ADORA são:

1. Usar um modelo integrado (em vez de uma coleção de modelos). Diferentemente da abordagem de coleção de modelos, adotada pela UML e

[51] Se forem desconsiderados resultados previamente obtidos para as ações não-automatizáveis, envolvidas na aplicação da regra.

outros, um modelo ADORA integra vários aspectos de modelagem (estrutura, dados, comportamento, interações com os usuários, etc.) em um único modelo coerente. Isso permitiu criar uma forte noção de consistência, e forneceu a base necessária para o desenvolvimento de mecanismos de verificação de consistência poderosos, implementados em ferramentas. Além disso, um modelo integrado torna mais sistemático a construção de modelos, reduz redundância e simplifica a verificação da completude.

2. Visualizar modelos em contexto, pela apresentação dos detalhes de um modelo juntamente com uma abstração do contexto que o envolve. A utilização de um modelo integrado não significa que tudo deve ser mostrado em um único diagrama, o que afogaria o leitor em um mar de informações. Os diagramas efetivamente exibidos são visões do modelo integrado, apresentando apenas as informações associadas a um aspecto de modelagem escolhido (estrutural, comportamental, funcional, do usuário, e de contexto). Entretanto, como todo diagrama exibido é apenas uma visão do modelo integrado, foi possível embutir na linguagem do método regras fortes de consistência e de completude, e construir ferramentas poderosas para a verificação e manutenção dessas regras.

3. Permitir aos usuários expressar diferentes partes de uma especificação com um grau variável de formalidade (adaptado a importância e ao risco de cada parte). As especificações em ADORA podem misturar texto informal e com outras notações formais. Por exemplo, transições de estado podem ser especificadas através da notação formal de *statecharts*, ou informalmente com texto, ou ainda uma combinação de ambas.

Portanto, a MIC tem vários pontos em comum com ADORA. A começar, e principalmente, pela utilização de um modelo integrado. Conseqüentemente, os mesmos benefícios reivindicados para ADORA, decorrentes dessa opção, também valem para a MIC. Outro ponto em comum é a mistura de especificações formais e informais, no modelo integrado. Por fim, ADORA, assim como a MIC, está baseada no reconhecimento de que comportamento e informação devem ser considerados conjuntamente em uma especificação de requisitos: "... cenários e objetos são complementares em uma especificação. Os cenários especificam estímulos que os atores enviam ao sistema e as reações do sistema a esses estímulos. Entretanto, quando essas reações dependem da história de estímulos e reações anteriores, isto é, da informação

armazenada [grifo nosso], uma especificação precisa das reações, apenas com cenários, se torna impossível. Os objetos especificam a estrutura, funções e comportamento que são necessários para especificar adequadamente, nos cenários, as reações." (GLINZ *et al.*, 2002).

ADORA difere da MIC em diversos aspectos. Primeiro, por buscar uma integração mais abrangente, envolvendo elementos estruturais, comportamentais, funcionais, de interação com os usuários, e de contexto de um sistema, objetivando a geração de diversas visões, em diferentes níveis de abstração. Em vez disso, a MIC foca, principalmente, estrutura e comportamento. Enquanto a MIC é aplicável a sistemas de informação, ADORA se presta a outros tipos de sistemas também. Mas a principal diferença reside no fato de ADORA não utilizar o MUC como base do modelo integrado, como faz a MIC. Neste sentido, ADORA se distancia da UML, enquanto o MIC pode ser considerado uma especialização de um de seus modelos: o MUC. Além disso, em ADORA elementos dos diversos modelos são explicitamente (graficamente) representados no modelo integrado, enquanto que na MIC, a única manifestação mais visível do MD, no modelo integrado, é a presença dos itens identificadores (aqueles com nome iniciado por *id_*); os demais elementos do MD permanecem implícitos e embutidos em elementos do MUC, podendo ser recuperados a partir desses, graças a uma nova disciplina adotada na elaboração do MUC – a adoção do NIO, juntamente com a especificação semiformal da interface informacional dos ICs.

5.6.2 Análise Propósito-Atividade

SVETINOVIC (2006) também atacou o problema da inconsistência entre MDs obtidos por diferentes modeladores, para um mesmo sistema. Ele propôs uma alteração no processo tradicional da AOO (que observa apenas conceitos), para adotar uma abordagem focada em atividades, onde se busca o conceito que é responsável por uma atividade em particular, obtendo-se assim, a validação do conceito através de uma análise "atividade-propósito". O resultado é um conjunto de artefatos intermediários mais restringidos do que o MD, capazes de influenciar o resultado da decomposição conceitual, e conseguir que MDs produzidos por diferentes modeladores, para um mesmo sistema, sejam mais consistentes entre si. O autor considera que uma combinação de "análise baseada em objetivo" e de "modelagem baseada em estado" seja suficiente, como técnicas de base, para a análise "propósito-atividade".

Portanto, a sua estratégia é avançar na análise antes de tentar uma decomposição conceitual, para com isso restringir o grau de liberdade na escolha das classes (conceitos) de domínio e na atribuição das responsabilidades às classes. A nosso ver, uma das características dessa estratégia é manter a responsabilidade principal de identificar os conceitos com o modelador, ou seja, deixar que ele decida, com base em diversos artefatos (e não apenas nos UCs), que conceitos escolher para compor o MD.

Diferentemente da abordagem de Svetinovic, a MIC assume que o modelador deve, antes de tudo, considerar os conceitos que os *stakeholders* utilizam no dia-a-dia do negócio. Idealmente, esses conceitos devem ser capturados durante o levantamento de requisitos. Assim, o MUC serve como um canal para elicitar os conceitos de domínio com a participação ativa dos *stakeholders*. Trata-se de aproveitar a oportunidade e a familiaridade dos *stakeholders* com a MUC, para extrair deles os conceitos que constituirão uma espécie de *baseline* para o modelo de domínio final. Não se descarta a possibilidade do modelador sugerir novas abstrações ou aperfeiçoamentos nas abstrações utilizadas pelos *stakeholders*, mas para isso é preciso tomar por base as abstrações consolidadas pelos *stakeholders*.

5.7 Considerações Finais

Este capítulo apresentou a solução construída nesta tese para o problema-alvo escolhido (Capítulo 3), e mostrou que ela atende uma lista de requisitos identificados a partir da análise das principais causas do problema (Capítulo 4).

O passo fundamental na direção da solução foi dado com duas providências:

1. Escolher os UCs no Nível Informacional de Objetivos (NIO, seção 5.2); e
2. Especificar com rigor formal a troca de informações entre o sistema e seu ambiente (seção 5.3). Os UCs assim obtidos foram distinguidos pelo nome de info cases (ICs).

O NIO especializa o critério normalmente utilizado na MUC para a escolha dos UCs do sistema (“o UC ter valor para um *stakeholder*") e, principalmente, constitui uma interpretação mais precisa para esse critério, considerado excessivamente subjetivo por vários autores.

As duas providências acima relacionadas fazem com que o modelo resultante de info cases (MIC) seja capaz de capturar, ao mesmo tempo, tanto informações de

comportamento, quanto informações estruturais e de estado, em um mesmo arcabouço conceitual, constituindo assim, um modelo integrado para a especificação de requisitos. Com isso, é possível obter um MD a partir do MIC, como uma visão extraída deste. Conforme mostrou a seção 5.4, é possível identificar um conjunto de regras e heurísticas para orientar e automatizar boa parte desse processo de extração.

Para caracterizar a solução construída, as seções deste capítulo contêm análises sobre:

a) A qualidade das especificações obtidas, focando sua usabilidade, legibilidade, completude, consistência, rastreabilidade e manutenibilidade (seção 5.5.1);
b) A aplicabilidade da técnica, concluindo que a mesma se destina a sistemas de informação (seção 5.5.2);
c) A relação com a Análise Essencial, pela influência que ela teve na concepção inicial do NIO (seção 5.5.3);
d) A relação com a Modelagem de Requisitos Orientada a Objetivos, uma vez que os ICs também representam objetivos dos *stakeholders* (seção 5.5.4);
e) O grau de automatismo possível na aplicação das regras de derivação da visão MD do MIC, bem com a completude do MD produzido com elas (seção 5.5.5); e
f) Dois trabalhos extraídos da literatura, mais diretamente relacionados com a solução proposta nesta tese – um deles porque também adota um modelo integrado, e o outro porque também ataca o problema da inconsistência entre MDs (seção 5.6).

Essas e outras análises teóricas contidas no presente capítulo lançaram luz sobre as propriedades da solução proposta, e forneceram indícios de que vale a pena avaliar a mesma através de estudos. O próximo capítulo (6) descreve os estudos realizados para avaliar, experimentalmente, a solução proposta.

6. Estudos Experimentais com o Modelo Integrado

Este capítulo descreve os estudos experimentais realizados com a modelagem de ICs (MIC), visando aprofundar a sua compreensão, aperfeiçoá-la, e testar a hipótese geral da tese (seção 4.4).

Inicialmente, são relatados os estudos precursores (seção 6.1), realizados antes da aprovação da proposta desta tese. Eles incluem experimentação didática e individual (seção 6.1.1), e dois estudos de observação, um real (EC-1, seção 6.1.2) e outro não (EC-2, seção 6.1.3). Após a aprovação da proposta da tese, foram realizados mais três estudos. O primeiro deles, um *quasi*-experimento (EXP-1, seção 6.2), investigou a granularidade e a cobertura funcional do MIC. O segundo deles, outro *quasi*-experimento (EXP-2, seção 6.3), tratou da granularidade e uniformidade dos MDs obtidos na modelagem com ICs, e o esforço demandado nessa modelagem. Por fim, o último estudo, um estudo de caso real (EC-3, seção 6.4), testou as regras de derivação e a adequação dos MDs por elas produzidos. O capítulo termina com uma análise sobre a cobertura dos estudos na validação da hipótese da tese (seção 6.5).

6.1 Estudos Precursores

Mesmo antes do fechamento de uma proposta de tese, alguns estudos investigativos já haviam sido realizados. Tais estudos se prestaram mais à exploração e desenvolvimento da técnica, do que à validação formal de hipóteses. Eles tiveram um papel importante na "prova de conceitos" e na compreensão, triagem e amadurecimento das idéias. Os primeiros estudos foram individuais ou executados por alunos (seção 6.1.1). Posteriormente, empreendeu-se um estudo que, diferentemente dos demais, foi um estudo de caso real (seção 6.1.2). Por último, um estudo com profissionais serviu como investigação preliminar sobre a uniformidade de MDs (seção 6.1.3). Na época desses estudos, a MIC se encontrava em uma versão preliminar, na qual os critérios para o particionamento do sistema em ICs não haviam sido ainda totalmente explicitados (embora, eles fossem aplicados intuitivamente). Igualmente, os termos *info case* (IC) e *modelagem com ICs* (MIC) ainda não existiam; no seu lugar eram utilizados os termos *objetivo informacional* e *Modelagem Informacional de Requisitos* (MIR). Isso explica a utilização desses últimos nas três próximas seções.

6.1.1 Experimentação Didática e Individual

Em suas primeiras versões, a MIR foi utilizada por alunos de graduação em Administração de Empresas da Universidade Federal de Juiz de Fora (UFJF). Os alunos eram introduzidos à técnica e desenvolviam o modelo informacional de uma aplicação. Em seguida, derivavam e implementavam (em *MS Access*) o modelo E/R associado, e construíam consultas SQL que eram executadas para concretizar algumas das saídas especificadas no modelo informacional.

Também desde as primeiras versões, a MIR foi aplicada em vários estudos individuais[52] para a modelagem de sistemas de pequeno porte em vários domínios, tais como gerência de bar, restaurante, vídeo-locadora, biblioteca, hotel, agendamento de ações e acompanhamento de projetos, agendamento de reuniões (LAMSWEERDE, 1992), medicina e segurança do trabalho, dentre outros. Alguns dos modelos produzidos foram implementados em sistemas que, durante certo tempo, eram utilizados rotineiramente.

6.1.2 Estudo Precursor 1 (EC-1): SIMEL

Para explorar o comportamento da MIR em um sistema real e de maior porte, foi empreendido o estudo de observação relatado nesta seção.

Esse estudo de caso envolveu uma equipe com 1 engenheiro de requisitos, 4 analistas-programadores e 4 especialistas de domínio, no desenvolvimento de um sistema *web* multiusuário[53], para apoiar as atividades de uma empresa de medicina e engenharia de segurança do trabalho. O sistema SIMEL (SIstema de MEdicina e Engenharia EmpresariaL) foi desenvolvido pelos analistas em PHP e MySQL, diretamente a partir do modelo informacional elaborado pelo engenheiro com o apoio e validação dos especialistas, constituído de 30 *objetivos informacionais*, totalizando 40 páginas de especificações. O desenvolvimento demandou 4 meses de trabalho. Consultas informais aos desenvolvedores, especialistas e *stakeholders*, realizadas durante o estudo e após o seu término, evidenciaram uma impressão positiva dos mesmos sobre a MIR e o sistema desenvolvido.

Alguns aspectos que valorizam os resultados obtidos são:

a) Nenhum dos analistas tinha conhecimento e experiência anteriores na MIR e no

[52] Realizados pelo autor desta tese.
[53] Aproximadamente 45 tabelas, 100.000 linhas de código escrito/gerado.

domínio da aplicação;

b) Os analistas trabalharam os 3 primeiros meses praticamente sem contato presencial com o engenheiro de requisitos, sendo a comunicação realizada por escrito via um software do tipo *instant messenger*;

c) As especificações foram elaboradas em paralelo com o desenvolvimento do sistema, cada objetivo informacional sendo liberado assim que a sua definição ficava pronta;

d) Toda a divisão de trabalho entre os analistas foi feita por objetivo informacional;

e) O esquema conceitual e o projeto do BD do sistema foram elaborados de forma incremental (por objetivo informacional) pelos analistas, sem a participação do engenheiro.

O estudo de caso SIMEL antecede a definição das regras de derivação do MD (seção 5.4.2) e, portanto, o MD do SIMEL foi obtido de forma *ad hoc* a partir da sua especificação informacional. Mesmo assim, o esquema conceitual do BD, elaborado pelos analistas, atendeu as expectativas do engenheiro. O Apêndice 2 apresenta o MD do SIMEL, com o intuito de dar uma idéia do porte do sistema construído.

6.1.3 Estudo Precursor 2 (EC-2): 1º Estudo da Uniformidade dos MDs

Outro estudo foi realizado com a finalidade de avaliar, preliminarmente, a compreensibilidade e usabilidade das regras de derivação e respectivos padrões sintáticos (seção 5.4.2), bem como ter uma primeira avaliação do grau de uniformidade dos MDs resultantes da sua aplicação. Participaram do estudo, em momentos diferentes, dois profissionais graduados em Ciência da Computação, analistas de sistemas de uma empresa de medicina e segurança do trabalho, ambos com 2 anos de experiência no desenvolvimento de sistemas e 1 ano na leitura dos objetivos informacionais de um sistema em fase de manutenção. Primeiramente, um dos analistas aplicou as regras utilizando a especificação informacional de um sistema de biblioteca com 19 objetivos informacionais, fornecida pelo experimentador[54]. O modelo de classes obtido foi comparado com o modelo de classes construído previamente pelo experimentador, pela aplicação das mesmas regras. As diferenças detectadas revelaram dificuldades do analista na compreensão de algumas regras, o que motivou a revisão do manual de regras. Posteriormente, e com o manual revisado, o outro analista executou o mesmo

[54] O autor desta tese.

estudo. Para avaliar a uniformidade dos resultados, foram contadas as diferenças entre o modelo obtido por cada analista e o modelo construído pelo experimentador. Consideramos diferença, para fins dessa avaliação, os elementos (classes, atributos, associações e operações) a mais ou faltando em relação ao modelo produzido pelo experimentador. **A** Tabela 6.1 apresenta os resultados dessa avaliação.

Tabela 6.1 – Resultados do Estudo Precursor 2

		Analista 1		Analista 2	
Elementos	**#E**	**#D**	**%S**	**#D**	**%S**
Classes	5	0	100	0	100
Associações	11	1	91	0	100
Atributos	20	4	80	0	100
Operações	41	4	90	1	97

Legenda
#E: Nr. de elementos no modelo do experimentador.
#D: Nr. de diferenças.
%S: Perc. de sucesso em reproduzir o modelo do experimentador.

6.1.4 Considerações sobre os Estudos Precursores

A experiência didática com a MIC mostrou que a técnica pode ser ensinada sem maiores dificuldades, e que mesmo profissionais fora da área da computação são capazes de ler e elaborar modelos simples utilizando a técnica. O Estudo 1 (SIMEL) funcionou como uma "prova de conceito", mostrando que a técnica pode ser utilizada para sistemas de porte médio a grande. O Estudo 2 (uniformidade dos MDs) revelou o potencial da MIC para resolver o problema das diferenças entre MDs produzidos por diferentes modeladores para um mesmo sistema.

Esses resultados favoráveis incentivaram o prosseguimento da pesquisa e a realização de novos estudos experimentais, agora mais formais e objetivando o teste das hipóteses construídas ao longo do trabalho de tese.

6.2 Estudo 1 (EXP-1): ICs - Granularidade e Cobertura Funcional

6.2.1 Definição, Planejamento e Execução

Este foi o primeiro *quasi-experimento*[55] realizado com a MIC, ainda em uma versão preliminar à atual. Nela, o conceito de *estado firme* (ou *estável* – seção 5.2) ainda não havia sido estabelecido. O particionamento do sistema em ICs utilizava exclusivamente o conceito de *eventos autônomos* que, no entanto, incluía tacitamente o conceito de estado firme.

[55] Estudo experimental controlado, mas com escolha não aleatória, mas por conveniência, dos participantes.

O estudo teve por objetivo testar o critério para o particionamento de um sistema em ICs, baseado nos eventos autônomos, em comparação ao critério baseado no valor do UC para algum *stakeholder*, utilizado na modelagem tradicional com UCs (BITTNER, SPENCE, 2002). Mais especificamente, o estudo visou avaliar, comparativamente, a *cobertura funcional* e a *granularidade* dos particionamentos obtidos com cada um desses critérios. Seguindo o esquema GQM (WOHLIN *et al.*, 2000), a definição do estudo é:

Analisar os modelos de requisitos funcionais obtidos através do particionamento pelo critério de eventos autônomos (utilizado na MIC) e pelo critério tradicional baseado no valor do UC para algum *stakeholder* (utilizado na MUC)

com o propósito de caracterizar

com respeito à cobertura funcional (número de funções cobertas pelo modelo) e granularidade (número de casos de uso do modelo)

do ponto de vista do engenheiro de requisitos

no contexto de estudantes do 4º período do curso de graduação em Ciência da Computação da UFJF, aplicando critérios de particionamento para modelar requisitos funcionais para um mesmo sistema de informação como exercício em sala de aula.

Métricas

Para os fins do estudo, a *cobertura funcional* de um particionamento foi medida pelo número de funções cobertas pelos UCs presentes no particionamento, dentre uma lista de funções extraídas de uma especificação preliminar do sistema (texto em linguagem natural), que serviu de ponto de partida para os particionamentos. A *granularidade* de um particionamento foi medida pelo número de UCs incluídos no particionamento.

Variáveis e Hipótese

A única variável independente (fator) do estudo é o critério utilizado para particionar o sistema-objeto. Dois tratamentos foram considerados para essa variável: o critério de eventos autônomos, para o particionamento em ICs, e o critério de valor, para o particionamento em UCs. O estudo possui duas variáveis dependentes: granularidade

e cobertura funcional dos particionamentos, ambas com medidas tomadas em escala do tipo razão.

A hipótese testada no estudo é que o critério de eventos autônomos produz particionamentos com maior cobertura funcional, e de grão mais fino (mais ICs).

Participantes

Participaram do estudo alunos da disciplina de modelagem de sistemas, do 2° ano do curso de graduação em Ciência da Computação da Universidade Federal de Juiz de Fora[56] (UFJF), totalizando 35 alunos, distribuídos em duas turmas, uma diurna e outra noturna (20 e 15, respectivamente). Essa seleção de participantes não seguiu o princípio ideal da amostragem aleatória, sendo feita por conveniência, para aproveitar a oportunidade do oferecimento da disciplina. Esses participantes constituíram uma amostra da população-alvo do estudo, que é a dos estudantes do curso de graduação em Ciência da Computação da UFJF.

Um questionário de caracterização, aplicado aos alunos, mostrou a falta de conhecimento e experiência prévios dos mesmos, na modelagem com UCs e ICs. O questionário também mostrou que o nível de conhecimento sobre a aplicação a ser modelada (um sistema de biblioteca) era o mesmo em ambas as turmas.

Design, Instrumentação e Execução

O estudo é um *quasi-experimento*, com contexto *multi-test within object* (1 objeto e mais de 1 participante), e do tipo-padrão *um fator com dois tratamentos* (WOHLIN *et al.*, 2000). O fator é o critério utilizado no particionamento do sistema objeto do estudo, e os dois tratamentos para esse fator são: o critério de eventos autônomos (para ICs) e o critério de valor (para UCs).

Foram feitas duas rodadas (*trials*), uma em cada turma. A turma diurna utilizou o critério de valor, enquanto que a noturna utilizou o critério dos eventos autônomos. Portanto, não se utilizou o princípio da aleatoriedade para a alocação dos tratamentos aos participantes, e o estudo ficou desbalanceado (turma diurna com 20 alunos e turma noturna com 15). Entretanto, essa divisão não foi feita apenas por conveniência, mas principalmente visou reforçar a validade interna dos resultados, por tornar mais improvável a troca de informações entre alunos alocados a tratamentos distintos. Não

[56] Do qual o autor desta tese é professor.

houve necessidade de blocagem dos participantes, já que na caracterização dos mesmos, não se evidenciaram diferenças significativas entre eles, do ponto de vista do conhecimento e experiência com qualquer das técnicas empregadas, ou em relação ao domínio do sistema.

Os instrumentos preparados pelo experimentador e utilizados pelos participantes do estudo foram: (1) um sumário do sistema a modelar, escrito em linguagem natural; (2) material didático sobre UCs e sobre ICs, para o treinamento dos participantes nessas técnicas; e (3) um questionário para caracterização dos participantes.

Antes da execução do estudo, ambas as turmas receberam treinamento em UCs, e a turma noturna recebeu, adicionalmente, treinamento complementar para aplicar ICs. A rodada de cada turma foi executada no mesmo dia, ocupando parte de uma aula.

Análise (prévia) da Validade

Durante o planejamento e a preparação do estudo, foi feita uma análise sobre a validade dos resultados a serem obtidos, visando prevenir ameaças a essa validade.

Uma das principais preocupações foi uma possível diferenciação entre as turmas, o que poderia comprometer a validade interna do estudo. Do ponto de vista do curso, as turmas só se diferenciam quanto ao turno, já que pertencem à mesma disciplina e são oferecidas no mesmo período letivo da sua grade, geralmente pelo mesmo professor. Mesmo assim, foi elaborado um questionário para caracterizar os participantes quanto ao conhecimento e experiência prévias na técnica de UCs[57] e no domínio do sistema a modelar. O questionário foi aplicado antes da execução do estudo e as respostas obtidas evidenciaram a uniformidade das duas turmas nesses aspectos. Também, foram tomadas providências para evitar troca de informações entre os participantes, o que poderia invalidar os resultados. Isso foi facilitado pelo fato das turmas serem de turnos distintos – diurno e noturno. Além disso, cada turma aplicou uma técnica distinta (UCs ou ICs), as rodadas foram executadas no mesmo dia, ocupando parte do tempo de aula, e sob a supervisão do experimentador.

O treinamento na técnica de UCs foi o mesmo para ambas as turmas, ministrado pelo mesmo instrutor, e utilizou o mesmo material didático. Embora a turma noturna fosse aplicar apenas a técnica de ICs, esta é, conforme apresentado no Capítulo 5, uma

[57] Quanto à técnica de ICs, havia certeza do seu desconhecimento por parte de todos os alunos, já que ela ainda não havia sido divulgada externamente.

especialização da técnica de UCs e, portanto, seu treinamento deve iniciar com os elementos da técnica de UCs. Após o treinamento de UCs, a turma noturna foi capacitada na técnica de ICs, mediante o treinamento na parte que é específica desta técnica. Assim, procurou-se garantir que eventuais diferenças nos resultados do estudo não fossem causadas por diferenças na aplicação dos elementos da técnica de UCs, mas sim pelas novidades introduzidas com ICs.

Graças ao número de participantes em cada turma (20 e 15), tornou-se interessante a aplicação de métodos estatísticos para avaliar a significância estatística dos resultados (validade de conclusão).

6.2.2 Análise Estatística dos Dados Apurados

A análise a seguir segue os preceitos apresentados em (ARAÚJO *et al*., 2006). Nela, "Técnica 1" representa o critério usual de valor de um UC para algum *stakeholder*, enquanto que "Técnica 2" representa o critério de eventos autônomos. Como cada participante utilizou apenas uma das duas técnicas, as amostras são consideradas independentes, um dos pressupostos para a aplicação de testes paramétricos e de alguns testes não-paramétricos.

Análise Estatística para a Variável Cobertura

A primeira consideração a ser feita ao conjunto de dados é relativa à normalidade e homocedasticidade (variância constante) das amostras utilizadas (WOHLIN *et al.*, 2000). Uma análise visual inicial da distribuição é eficiente para o conhecimento do comportamento das amostras. A Figura 6.1 apresenta um diagrama *box-plot* destas amostras em relação à variável Cobertura (no SPSS: opção Graphs / Boxplot). A análise desta figura não apresenta *outliers*.

Outra conclusão que se pode tirar da Figura 6.1 é a aparente grande variabilidade entre as duas amostras. Além disso, a Técnica 2 parece apresentar maiores valores para cobertura. Para analisar corretamente esta questão, deve-se executar um teste estatístico apropriado que, neste caso, será o Teste de Levene (WOHLIN *et al.,* 2000), que será feito posteriormente.

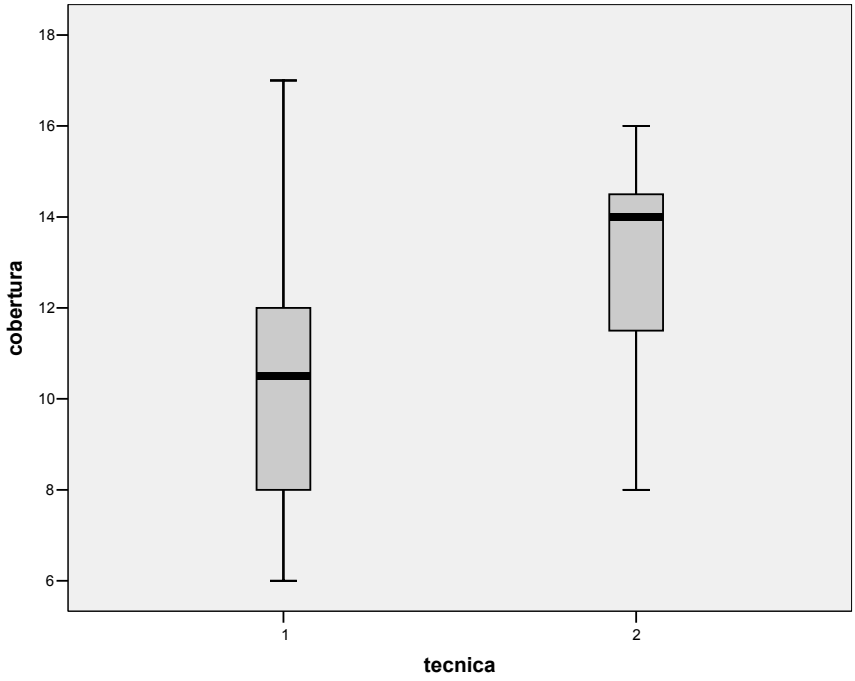


Figura 6.1 – Diagrama *Box-Plot* para a variável Cobertura

O Teste T (WOHLIN *et al.,* 2000) para duas amostras independentes exige que as amostras sigam uma distribuição normal. Desta forma, tem-se um primeiro teste de hipóteses a ser feito, considerando um nível de significância de 5%, sendo:

H0 (hipótese nula): A distribuição é normal

H1 (hipótese alternativa): A distribuição não é normal

A Tabela 6.2 abaixo apresenta a saída dos testes de normalidade do SPSS (SPSS, 2008) considerando a variável Cobertura (No SPSS: Analyze / Descriptive Statistics / Explore).

Tabela 6.2 – Testes de Normalidade para a variável Cobertura

Tests of Normality

		Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	tecnica	Statistic	df	Sig.	Statistic	df	Sig.
cobertura	1	,186	20	,067	,941	20	,254
	2	,211	15	,071	,909	15	,133

a. Lilliefors Significance Correction

O Teste de Kolmogorov-Smirnov é bastante utilizado para a verificação de normalidade em amostras com mais de 30 elementos. Entretanto, para amostras com menos de 50 elementos (pequenas amostras), costuma-se utilizar o Teste de Shapiro-Wilk (WOHLIN *et al.,* 2000).

Independente do teste, observa-se que ambas as amostras possuem o valor de significância (Sig.) superior a 0,05 e, portanto, não há indícios para rejeitar a hipótese nula a um nível de significância de 5%. Desta forma, a distribuição das amostras para a variável Cobertura é normal, logo será utilizado o teste paramétrico T para duas amostras independentes.

O Teste T pode ter duas expressões diferentes em função das variâncias poderem ou não ser assumidas como iguais, conclusão que se retira diretamente do nível de significância do Teste de Levene. Desta forma, tem-se outro teste de hipóteses a ser considerado, a um nível de significância de 5%, sendo:

H0: As variâncias são iguais ($\sigma^2_{Técnica1} = \sigma^2_{Técnica2}$)

H1: As variâncias são diferentes ($\sigma^2_{Técnica1} \neq \sigma^2_{Técnica2}$)

A Tabela 6.3 apresenta a saída do teste de Levene do SPSS, considerando a variável Cobertura (no SPSS: Analyze / Compare Means / Independent-Samples T Test).

Tabela 6.3 – Testes de amostras independentes para a variável Cobertura

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
cobertura	Equal variances assumed	1,556	,221	-2,796	33	,009	-2,583	,924	-4,463	-,703
	Equal variances not assumed			-2,897	32,919	,007	-2,583	,892	-4,398	-,769

Através da Tabela 6.3, pelas colunas do Teste de Levene para Igualdade de Variâncias, verifica-se pela primeira linha dos resultados, que as amostras possuem variâncias iguais, uma vez que a significância (Sig.) é maior que 0,05. Portanto, considerando um nível de significância de 5%, não há indícios para se rejeitar a hipótese nula.

Por fim, satisfeito o pressuposto de normalidade, e uma vez que as variâncias são iguais, pode-se proceder a análise de comparação das médias das duas amostras, gerando um novo teste de hipóteses, a um nível de significância de 5%, sendo:

H0: As médias são iguais ($\mu_{Técnica1} = \mu_{Técnica2}$)

H1: As médias são diferentes ($\mu_{Técnica1} \neq \mu_{Técnica2}$)

Retornando à Tabela 6.3, percebe-se na primeira linha, que corresponde à confirmação de que as variâncias das amostras são iguais, que a significância do Teste T (Sig. (2-tailed)) é inferior a 0,05 e, desta forma, rejeita-se a hipótese nula, concluindo-se que as médias são diferentes a um nível de significância de 5%.

A Tabela 6.4 apresenta o complemento da análise do Teste T, mostrando o valor das médias. A média da variável *cobertura* foi de 12,93 funções cobertas no critério de eventos autônomos, contra 10,35 funções cobertas no critério de valor. Conclui-se, portanto, que o critério dos eventos autônomos produz cobertura funcional maior (aproximadamente 25% maior) do que o critério de valor, a um nível de significância de 5%. Além disso, a cobertura variou menos no critério dos eventos autônomos (menor desvio padrão).

Tabela 6.4 – Complemento da análise do Teste T

Group Statistics

	tecnica	N	Mean	Std. Deviation	Std. Error Mean
cobertura	1	20	10,35	2,961	,662
	2	15	12,93	2,314	,597

Análise Estatística para a Variável Granularidade

A Figura 6.2 apresenta um diagrama *box-plot* destas amostras em relação à variável Granularidade (No SPSS: opção Graphs / Boxplot). A análise desta figura não apresenta *outliers.*

Outra conclusão que se pode tirar desta figura é a aparente igualdade de variabilidade entre as duas amostras. Apesar disso, a Técnica 2 aparenta maior granularidade. Para analisar corretamente esta questão, deve-se executar um teste estatístico apropriado que, neste caso, será o Teste de Levene, que será feito mais adiante.

O Teste T para duas amostras independentes exige que as amostras sigam uma distribuição normal. Desta forma, tem-se um primeiro teste de hipóteses a ser feito, considerando um nível de significância de 5%, sendo:

H0: A distribuição é normal

H1: A distribuição não é normal

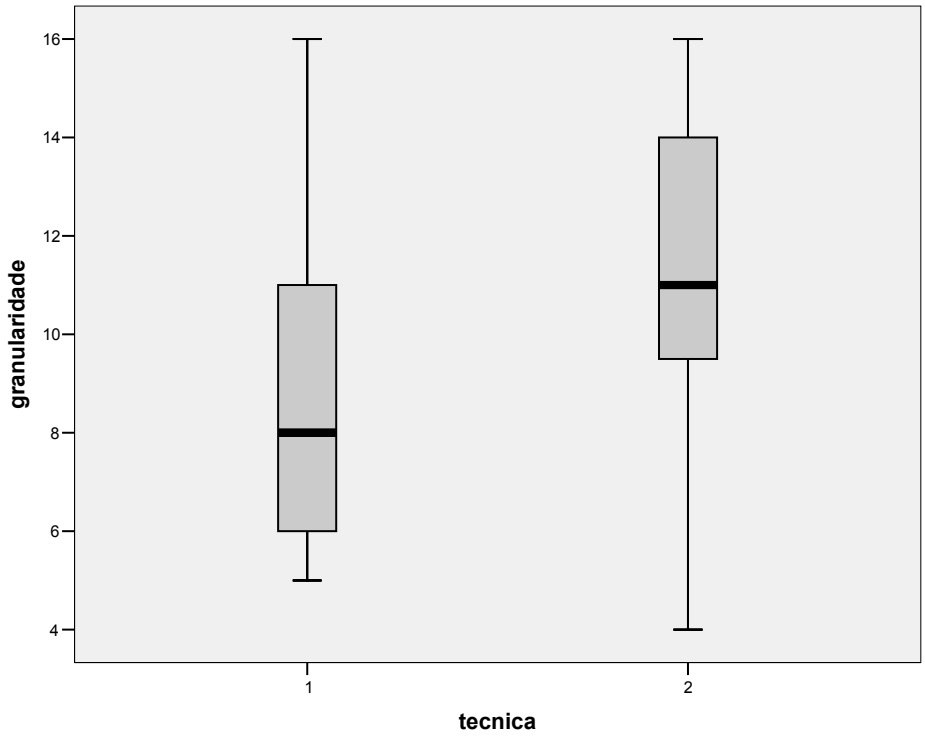


Figura 6.2 – Diagrama *Box-Plot* para a variável Granularidade

A Tabela 6.5 apresenta a saída dos testes de normalidade do SPSS, considerando a variável Granularidade (No SPSS: Analyze / Descriptive Statistics / Explore).

Tabela 6.5 – Testes de Normalidade para a variável Granularidade

Tests of Normality

		Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	tecnica	Statistic	df	Sig.	Statistic	df	Sig.
granularidade	1	,214	20	,017	,908	20	,059
	2	,151	15	,200*	,950	15	,521

*. This is a lower bound of the true significance.

a. Lilliefors Significance Correction

Pelo resultado do Teste de Shapiro-Wilk, observa-se que ambas as amostras possuem o valor de significância (Sig.) superior a 0,05 e, portanto, não há indícios para rejeitar a hipótese nula a um nível de significância de 5%. Desta forma, a distribuição das amostras para a variável Granularidade é normal, logo será utilizado o teste paramétrico T para duas amostras independentes. Antes porém, é necessário verificar se as variâncias podem ser assumidas como iguais ou não. Assim, tem-se outro teste de hipóteses, a um nível de significância de 5%:

H0: As variâncias são iguais ($\sigma^2_{\text{Técnica1}} = \sigma^2_{\text{Técnica2}}$)

H1: As variâncias são diferentes ($\sigma^2_{Técnica1} \neq \sigma^2_{Técnica2}$)

A Tabela 6.6 apresenta a saída do teste de Levene do SPSS, considerando a variável Granularidade (No SPSS: Analyze / Compare Means / Independent-Samples T Test).

Tabela 6.6 – Testes de amostras independentes para a variável Granularidade

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
granularidade	Equal variances assumed	,001	,975	-2,021	33	,051	-2,233	1,105	-4,482	,015
	Equal variances not assumed			-1,991	28,536	,056	-2,233	1,122	-4,529	,062

Através desta tabela, pelas colunas do Teste de Levene para Igualdade de Variâncias, verifica-se pela primeira linha dos resultados, que a significância (Sig.) é maior que 0,05. Portanto, considerando um nível de significância de 5%, não há indícios para se rejeitar a hipótese nula, podendo-se considerar que as amostras possuem variâncias iguais.

Por fim, satisfeito o pressuposto de normalidade e uma vez que as variâncias são iguais, pode-se proceder com a análise de comparação das médias das duas amostras, gerando um novo teste de hipóteses, a um nível de significância de 5%, sendo:

H0: As médias são iguais ($\mu_{Técnica1} = \mu_{Técnica2}$)

H1: As médias são diferentes ($\mu_{Técnica1} \neq \mu_{Técnica2}$)

Retornando à Tabela 6.6, percebe-se na primeira linha, que corresponde à confirmação de que as variâncias das amostras são iguais, que a significância do Teste T (Sig. (2-tailed)) é superior a 0,05[58] e, desta forma, não há indícios para rejeitar a hipótese nula, concluindo-se que as médias são iguais a um nível de significância de 5%. Outra maneira de verificar esta situação é a constatação de que o valor zero está entre os limites inferior e superior do intervalo de confiança, também não sendo possível rejeitar a hipótese nula.

[58] Embora muito próxima (0,051).

6.2.3 Considerações sobre o Estudo 1

O estudo permitiu concluir, a um nível de significância de 5%, que o particionamento por ICs (critério dos eventos autônomos) proporciona maior cobertura funcional do que o particionamento por UCs (critério de valor). Quanto à granularidade dos particionamentos, o estudo não validou (a um nível de significância de 5%), a hipótese de que o particionamento por ICs tem grão mais fino do que o particionamento por UCs.

Deve-se ressaltar que a hipótese adotada para a granularidade visou mais cumprir um requisito formal do estudo (ter uma hipótese formalmente enunciada) do que propriamente testar uma teoria. Como o particionamento pelo critério da técnica tradicional de UCs é muito vago e subjetivo, fica praticamente impossível tomá-lo como base de comparação. Portanto, o objetivo desta parte do estudo foi apenas conhecer um pouco mais sobre os particionamentos produzidos por cada técnica.

Em geral, considera-se que o aluno de um curso noturno possui perfil distinto do aluno de um curso diurno. Por exemplo, alunos de cursos noturnos muitas vezes trabalham durante o dia, e em decorrência, tem menos tempo para se dedicar aos estudos. Isso pode justificar a decisão de alocar a técnica de ICs para a turma noturna, por corresponder a uma situação menos favorável à hipótese do estudo. Entretanto, esse mesmo perfil também pode significar que alunos de cursos noturnos possuem maior maturidade e experiência de trabalho, até porque, é comum se observar alunos de mais idade neles. Sob essa ótica, a alocação da técnica de ICs à turma noturna pode ter favorecido a "validação" da hipótese. A caracterização dos participantes, realizada no escopo deste estudo, apenas mostrou que as turmas eram homogêneas quanto à experiência prévia nas técnicas e no domínio do sistema. Por isso, uma sugestão para novos estudos similares é avaliar também o rendimento acadêmico dos alunos, e a maturidade dos mesmos (por exemplo, medindo o tempo de inserção deles no mercado de trabalho). Outra sugestão é redesenhar o estudo para que ambos os grupos utilizem as duas técnicas.

Conforme ressaltado na introdução deste estudo (seção 6.2.1), o conceito de estado estável (ou firme) ainda não havia sido estabelecido por ocasião da realização do estudo, embora ele estivesse, de certa forma, embutido no conceito de evento autônomo. A separação, posterior, dos dois conceitos, decorreu de uma maior compreensão do que estava envolvido na prática de ICs, tendo representado a explicitação de um

conhecimento tácito. A partir daí, o ensino de como particionar um sistema em ICs ficou mais objetivo, facilitando o trabalho do instrutor. Como conseqüência, pode-se esperar uma melhor assimilação por parte dos alunos. Por isso, é bastante provável que uma replicação do estudo, contando agora com esse novo elemento, produza resultados ainda mais favoráveis à sua hipótese.

6.3 Estudo 2 (EXP-2): MD - Granularidade, Uniformidade e Esforço

Este estudo experimental foi desenvolvido no contexto de um trabalho de conclusão do curso de graduação em Ciência da Computação (GONÇALVES, 2008a) (GONÇALVES, 2008b), da Universidade Federal de Juiz de Fora (UFJF), sob a orientação do autor da presente tese. Esta seção apresenta um resumo do estudo, em nível de detalhe compatível com o espaço disponível nesta monografia; maiores detalhes podem ser obtidos nas referências acima indicadas (incluindo todo o material utilizado e produzido durante o estudo, ou seja, o seu empacotamento).

6.3.1 Objetivos, Questões e Métricas

Um dos objetivos do estudo foi comparar as duas técnicas – ICs e UCs, quanto à *granularidade* e a *uniformidade* dos MDs por elas produzidos, no contexto do desenvolvimento de um sistema de informação, por profissionais graduados em Ciência da Computação pela UFJF. Outro objetivo foi comparar o *esforço* empregado em cada técnica, para a obtenção do MD.

O *esforço* é medido pelo número de horas gasto por cada participante para obtenção do MD.

A *granularidade* de um MD é medida pelo número de classes que o modelo possui, ou seja, quanto maior o número de classes, maior a granularidade do modelo. Já a *uniformidade* entre MDs expressa o grau de semelhança entre eles.

Para uma avaliação abrangente da uniformidade entre MDs, foram considerados dois tipos de uniformidade:

- *Uniformidade de abstrações*: baseada na comparação das abstrações de domínio, existentes em cada modelo, a nível puramente semântico; e
- *Uniformidade representacional*: baseada na comparação de atributos, associações e operações utilizadas para representar cada abstração presente nos modelos.

A uniformidade de abstrações entre dois MDs é calculada como a razão entre o número de abstrações coincidentes nos dois modelos, e o número de abstrações únicas, entre os dois modelos. Essa métrica pode ser expressa pela fórmula:

$$\text{Unif-A}_{i,j} = \frac{A'_{i,j}}{(A_i + A_j) - (A'_{i,j})}$$

Onde, para $i \neq j$:

- A_i é a cardinalidade do conjunto das abstrações do Modelo i;
- A_j é a cardinalidade do conjunto das abstrações do Modelo j; e
- $A'_{i,j}$ é a cardinalidade do conjunto das abstrações comuns a ambos os modelos i e j.

A Figura 6.3 ilustra a métrica acima. Nela, a uniformidade de abstrações entre os modelos está representada pela região de intercessão. Quanto maior essa região, maior é a uniformidade de abstrações dos modelos.

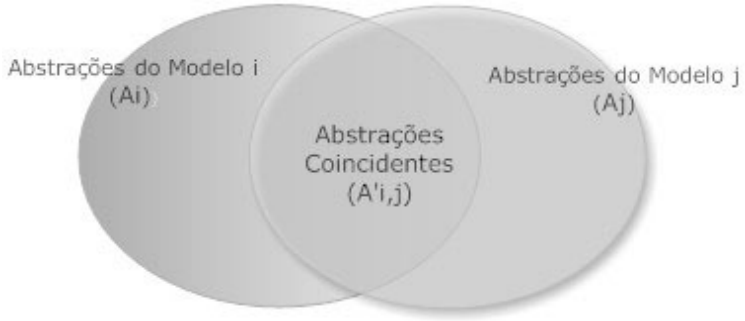


Figura 6.3 – Ilustração dos elementos da métrica de uniformidade de abstrações

A uniformidade de um tipo de elemento representacional (associações, atributos ou operações), entre dois MDs, é dada pela razão entre o número de elementos coincidentes e o número de elementos únicos, considerando apenas as classes que representam abstrações comuns aos dois MDs. Por exemplo, essa métrica, para atributos, pode ser expressa pela fórmula:

$$\text{Unif-At}_{i,j} = \frac{At'_{i,j}}{(At_{i,j} + At_{j,i}) - (At'_{i,j})}$$

Onde, para $i \neq j$:

- $At_{i,j}$ é a cardinalidade do conjunto de atributos do Modelo i, presentes nas classes que representam abstrações comuns a ambos os modelos i e j;

- $At_{j,i}$ é a cardinalidade do conjunto de atributos do modelo j, presentes nas classes que representam abstrações comuns a ambos os modelos i e j; e
- $At'_{i,j}$ é a cardinalidade do conjunto de atributos coincidentes, em todas as classes que representam abstrações comuns a ambos os modelos i e j.

A uniformidade de operações (Unif-$O_{i,j}$) e a uniformidade de associações (Unif-$L_{i,j}$), entre dois MDs (*i* e *j*), são definidas de forma análoga.

Por fim, a uniformidade representacional entre dois MDs *i* e *j*, é calculada como a média aritmética das uniformidades de atributos, operações e associações entre os modelos:

$$\text{Unif-R}_{i,j} = \frac{\text{Unif-At}_{i,j} + \text{Unif-O}_{i,j} + \text{Unif-L}_{i,j}}{3}$$

Onde:

- Unif-$At_{i,j}$: Uniformidade de atributos entre os modelos i e j; e
- Unif-$O_{i,j}$: Uniformidade de operações entre os modelos i e j; e
- Unif-$L_{i,j}$: Uniformidade de associações entre os modelos i e j.

6.3.2 Seleção de Participantes

Conforme visto na introdução deste capítulo, a população-alvo do estudo é constituída pelos profissionais graduados em Ciência da Computação pela Universidade Federal de Juiz de Fora (UFJF). A amostra da população foi selecionada por conveniência, sendo formada por seis pessoas conhecidas do autor do trabalho de conclusão de curso, e que tinham disponibilidade para participar do estudo. Portanto, não foi utilizado o princípio de aleatoriedade na seleção da amostra populacional.

Os seis participantes foram divididos em dois grupos com três participantes cada um. Cada grupo utilizou apenas uma das técnicas avaliadas neste estudo. Os participantes que formaram o grupo A, que utilizou ICs, foram designados pelos números 1, 2 e 3. Já os participantes do grupo B, que utilizou a técnica tradicional de UCs, foram designados pelos números 4, 5, e 6. Os participantes do Grupo B (UCs) não deveriam ter qualquer conhecimento em ICs, para evitar uma interferência indesejada nos resultados do estudo. Como duas das pessoas selecionadas já tinham algum conhecimento em ICs, elas foram alocadas no Grupo A (ICs). Devido ao número reduzido de participantes (6), era importante balancear os grupos, ou seja, fazê-los do

mesmo tamanho. Para isso, foi escolhido mais um participante para completar a composição do Grupo A (ICs). A escolha foi feita com base no questionário de caracterização dos participantes (Apêndice 3), cujas respostas estão resumidas na Tabela 6.7. O principal critério utilizado nessa escolha foi o de fazer o Grupo B o mais homogêneo possível.

Tabela 6.7 – Caracterização dos participantes do estudo

			Exp. Profissional (anos)			Domínio		Casos de Uso			MIC		
Partic.	Form. Grad.	Escolaridade	Estágio	Normal	Σ	Conh. Ant?	Já modelou?	Tipo Exp.	Nív. Exp.	Trein. ant?	Exp ?	Nív. Exp	Tempo
P1	3 anos	Especialização	-	3	3	Não	Não	Acadêmica	L/E	Não	Sim	L+/E-	2,5 anos
P2	0,5 anos	Graduação	1,8	0,5	1,4	Não	Não	Acadêmica	L/E	Não	Sim	L	0,5 anos
P3	1,5 anos	Graduação	2	1	2	Sim (-)	Não	Acad./Prof.	L/E	Não	Não	-	-
P4	0,5 anos	Mestrando UFJF	2,5	-	1,3	Não	Não	Acadêmica	L/E	Não			
P5	0,5 anos	Mestrando COPPE	3,5	-	1,8	Não	Não	Acadêmica	L/E	Não			
P6	0,5 anos	Especializando	0,3	-	0,2	Não	Não	Acadêmica	L/E	Não			

A Tabela 6.7 apresenta, para cada participante, o tempo decorrido desde a formatura na graduação (Form. Grad.), o nível de escolaridade, o tempo de experiência profissional, o conhecimento e experiência anteriores no domínio do sistema objeto do estudo, a experiência com casos de uso, e por fim, apenas para os participantes do Grupo A (ICs), a experiência com ICs. Para facilitar a comparação da experiência profissional dos participantes, a tabela apresenta, além dos tempos de estágio (carga horária de 20 h semanais) e normal (40 h semanais), o tempo equivalente calculado a partir desses dois (Σ). Além disso, o nível de experiência em UCs e em ICs é detalhado em experiência de leitura (L) e/ou de escrita (E)[59].

Pode-se ainda acrecentar, a título de caracterização dos participantes, que todos eles se graduaram no mesmo curso (Ciência da Computação), pela mesma universidade (UFJF). Além disso, a disciplina responsável pela ensino da técnica de UCs (Modelagem de Sistemas) foi cursada por todos eles com a mesma ementa e com mesmo professor.

Uma vez que a técnica de ICs é baseada na técnica de UCs, todos os participantes receberam o mesmo treinamento em UCs; adicionalmente, os participantes do Grupo A (ICs) receberam um treinamento nos elementos específicos da técnica de ICs (critérios adicionais para o particionamento do sistema em ICs, e regras de derivação do MD a partir dos ICs).

[59] Alguma informação sobre a intensidade dessa experiência é dada através dos sinais + (maior) e – (menor).

6.3.3 Formulação das Hipóteses

Devido ao pequeno número de participantes, não foram aplicados métodos estatísticos para a análise dos dados coletados. Mesmo assim, procurou-se utilizar na formulação das hipóteses o mesmo formalismo normalmente requerido para a aplicação de tais métodos. Com isso, se no futuro o estudo for repetido com mais participantes, a formulação adequada das hipóteses já estará pronta.

Esforço

Hipótese Nula (H_o-E): Não há diferença no esforço empregado para a obtenção do MD a partir de ICs, em relação ao esforço empregado na obtenção do MD a partir de UCs.

Hipótese Alternativa (**H_1-E**): O esforço empregado para a obtenção do MD a partir dos ICs é maior do que o esforço para se obter o MD a partir dos UCs.

Formalmente: **H_0-E**: $\mu_{E\text{-}ICs} = \mu_{G\text{-}UCs}$ e **H_1-E**: $\mu_{E\text{-}ICs} > \mu_{E\text{-}UCs}$, onde $\mu_{E\text{-}ICs}$ é a média dos esforços despendidos na obtenção do MD a partir dos ICs, e $\mu_{E\text{-}UCs}$ é a média dos esforços despendidos na obtenção do MD a partir dos UCs.

Os participantes 1, 2 e 3 aplicaram ICs, e os participantes 4, 5 e 6 aplicaram UCs. As médias acima podem ser representadas pelas fórmulas a seguir (onde E_i é o número de horas gasto para obtenção do MD pelo participante i):

- $\mu_{E\text{-}ICs} = \frac{E_1 + E_2 + E_3}{3}$; e
- $\mu_{E\text{-}UCs} = \frac{E_4 + E_5 + E_6}{3}$.

Granularidade

Hipótese Nula (H_o-G): Não há diferença na granularidade dos MDs produzidos com ICs, em relação àqueles produzidos utilizando UCs.

Hipótese Alternativa (**H_1-G**): A granularidade dos MDs obtidos utilizando ICs é menor do que a granularidade dos MDs obtidos utilizando UCs.

Formalmente: **H_0-G**: $\mu_{G\text{-}ICs} = \mu_{G\text{-}UCs}$ e **H_1-G**: $\mu_{G\text{-}ICs} < \mu_{G\text{-}UCs}$, onde $\mu_{G\text{-}ICs}$ é a média das granularidades dos MDs obtidos com ICs, e $\mu_{G\text{-}UCs}$ a média das granularidade dos MDs obtidos com UCs.

Para o estudo em questão, onde os participantes 1, 2 e 3 aplicaram ICs, e os participantes 4, 5 e 6 aplicaram UCs, as médias acima podem ser representadas pelas

fórmulas a seguir (onde C_i é o número de classes existentes no MD produzido pelo participante i):

- $\mu_{G\text{-}ICs} = \frac{C_1 + C_2 + C_3}{3}$; e
- $\mu_{G\text{-}UCs} = \frac{C_4 + C_5 + C_6}{3}$.

Uniformidade de abstrações

Hipótese Nula (H_o-U_a): Não há diferença na uniformidade de abstrações dos MDs produzidos com ICs, em relação aos MDs produzidos com UCs.

Hipótese Alternativa (H_1-U_a): A uniformidade de abstrações dos MDs produzidos com ICs é maior do que a uniformidade de abstrações dos MDs produzidos com UCs.

Formalmente: **H_0-U_a**: $\mu_{Ua\text{-}ICs} = \mu_{Ua\text{-}UCs}$ e **H_1-U_a**: $\mu_{Ua\text{-}ICs} > \mu_{Ua\text{-}UCs}$, onde $\mu_{Ua\text{-}ICs}$ e $\mu_{Ua\text{-}UCs}$ são as médias das uniformidades de abstrações entre pares distintos de MDs obtidos com ICs e com UCs, respectivamente (pares resultantes da combinação, dois a dois, dos modelos de cada grupo).

Para o estudo em questão, onde os participantes 1, 2 e 3 aplicaram ICs, e os participantes 4, 5 e 6 aplicaram UCs, as médias acima podem ser representadas pelas fórmulas a seguir:

- $\mu_{Ua\text{-}ICs} = \frac{\text{Unif-}A_{1,2} + \text{Unif-}A_{1,3} + \text{Unif-}A_{2,3}}{3}$; e
- $\mu_{Ua\text{-}UCs} = \frac{\text{Unif-}A_{4,5} + \text{Unif-}A_{4,6} + \text{Unif-}A_{5,6}}{3}$.

Uniformidade de atributos

Hipótese Nula (H_o-U_{at}): Não há diferença na uniformidade de atributos dos MDs produzidos com ICs, em relação aos MDs produzidos com UCs.

Hipótese Alternativa (H_1-U_{at}): A uniformidade de atributos dos MDs produzidos com ICs é maior do que a uniformidade de atributos dos MDs produzidos com UCs.

Formalmente: **H_0-U_{at}**: $\mu_{Uat\text{-}ICs} = \mu_{Uat\text{-}UCs}$ e **H_1-U_{at}**: $\mu_{Uat\text{-}ICs} > \mu_{Uat\text{-}UCs}$, onde $\mu_{Uat\text{-}ICs}$ e $\mu_{Uat\text{-}UCs}$ são as médias das uniformidades de atributos entre pares distintos de

MDs obtidos com ICs e com UCs, respectivamente (pares resultantes da combinação, dois a dois, dos modelos de cada grupo).

Para o estudo em questão, onde os participantes 1, 2 e 3 aplicaram ICs, e os participantes 4, 5 e 6 aplicaram UCs, as médias acima podem ser representadas pelas fórmulas a seguir:

- $\mu_{\text{Uat-ICs}} = \frac{\text{Unif-At}_{1,2} + \text{Unif-At}_{1,3} + \text{Unif-At}_{2,3}}{3}$; e
- $\mu_{\text{Ua-UCs}} = \frac{\text{Unif-At}_{4,5} + \text{Unif-At}_{4,6} + \text{Unif-At}_{5,6}}{3}$.

As hipóteses para **uniformidade de associações** ($\mathbf{H_0}$-$\mathbf{U_l}$ e $\mathbf{H_1}$-$\mathbf{U_l}$), **uniformidade de operações** ($\mathbf{H_0}$-$\mathbf{U_o}$ e $\mathbf{H_1}$-$\mathbf{U_o}$), e **uniformidade representacional** ($\mathbf{H_0}$-$\mathbf{U_r}$ e $\mathbf{H_1}$-$\mathbf{U_r}$), bem como a computação das respectivas médias, são definidas de forma análoga.

6.3.4 Design do Estudo

Trata-se de um *quasi*-experimento do tipo **um fator com dois tratamentos** (WOHLIN *et al.*, 2000), ou seja, uma única variável independente (fator) podendo receber dois valores (tratamentos). Neste *quasi*-experimento, o fator é qual técnica foi utilizada para elaborar o modelo de requisitos e o respectivo diagrama de classes, e os tratamentos são as duas técnicas utilizadas para isso: a técnica tradicional baseada em UCs, e a técnica que emprega ICs (descritas nos capítulos 2 e 5, respectivamente).

Conforme relatado anteriormente, os participantes foram divididos em dois grupos, cada um com três indivíduos, totalizando seis participantes. Cada um destes grupos ficou responsável pela aplicação de uma das técnicas. O número igual de participantes em cada grupo atende ao princípio do **balanceamento** na organização do estudo.

6.3.5 Instrumentação

Para a produção dos MDs, é muito importante que os participantes tenham conhecimento do domínio do sistema a ser modelado. Para isso, todos os participantes receberam um sumário das principais funções do sistema a modelar (Apêndice 3). Durante a execução do estudo, as dúvidas que surgiram foram esclarecidas pelos experimentadores a todos os participantes.

Cada grupo de participantes recebeu um kit contendo todo o material de apoio e um treinamento específico para a técnica a ser aplicada por eles. O material foi entregue alguns dias antes da realização do estudo, para que os participantes pudessem se familiarizar com ele, e o treinamento foi ministrado imediatamente antes do início do estudo.

Foi preparado um questionário para a caracterização dos participantes, cobrindo o tempo transcorrido desde a formatura deles na graduação, o seu perfil acadêmico, experiência profissional, conhecimento anterior no domínio do sistema, prática de modelagem nesse domínio, conhecimento e experiência anteriores em UCs e ICs, entre outras coisas. Este questionário pode ser visto no Apêndice 3.

O estudo foi realizado na empresa em que trabalhava o autor do trabalho de conclusão de curso, e pôde contar com a infra-estrutura lá existente. O treinamento foi ministrado em sala própria, com computador, *datashow* e quadro branco. Durante a execução do estudo propriamente dito, cada grupo ocupou uma sala distinta, com um computador para cada participante. Os participantes utilizaram a ferramenta *Visual Paradigm*™ (VP, 2007), previamente instalada nos computadores, para a elaboração do modelo de requisitos (UCs ou ICs) e do respectivo MD, do sistema.

6.3.6 Análise (prévia) da Validade

Durante o planejamento do estudo foi feita uma análise visando detectar e debelar possíveis ameaças à validade do mesmo. Nesta seção, é apresentado o resultado dessa análise, para cada tipo de validade: interna, de construção, de conclusão e externa.

Outra questão considerada foi que uma prioridade deveria ser atribuída a cada tipo de validade, tendo em vista que, muitas vezes, um expediente para favorecer um tipo de validade pode prejudicar outro tipo. Neste aspecto, a prioridade adotada decorreu da sugestão de COOK e CAMPBELL (1979, *apud* WOHLIN *et al.*, 2000) para experimentos com a finalidade de testar teorias. Para essa categoria de experimentos, eles sugerem a seguinte prioridade (da maior para a menor): validade interna, validade de construção, validade de conclusão e validade externa. A análise que se segue é apresentada nesta ordem.

Validade Interna

Para evitar que fatores ambientais e externos produzissem efeitos que pudessem ser confundidos com os efeitos dos tratamentos (técnicas empregadas), os tratamentos foram aplicados ao mesmo tempo (um para cada grupo de participantes) e no mesmo ambiente (na empresa em que o autor trabalha). Os participantes de cada grupo ficaram em salas distintas e foram instruídos a não trocar informações entre si. Para garantir a observância dessa regra, cada sala contou com a presença constante de um dos experimentadores.

Validade de Construção

Neste aspecto, o treinamento ministrado aos participantes e o material de apoio que acompanhou o treinamento, visaram garantir um entendimento uniforme das técnicas a serem aplicadas.

Validade de Conclusão

Não foi possível obter um número maior de participantes para a execução do estudo. Com isso, a validade de conclusão ficou de certa forma prejudicada, principalmente pela não utilização de testes estatísticos para o teste das hipóteses do estudo.

Validade Externa

Apesar de todos os participantes pertencerem à população-alvo do estudo, a saber, os profissionais formados em Ciência da Computação pela UFJF, a impossibilidade de se aplicar o princípio da aleatoriedade na seleção da amostra populacional, bem como o tamanho reduzido dela, levantam dúvidas sobre a capacidade dessa amostra em representar essa população-alvo.

6.3.7 Execução do Estudo

Nesta seção, são relatados alguns detalhes relativos à execução do estudo.

A Tabela 6.8 mostra o cronograma realizado na etapa de execução do estudo. A execução do estudo ocupou um dia e meio, tendo iniciado com o treinamento de UCs para ambos os grupos (Treinamento UC – Inicial). Em seguida, enquanto o Grupo B (UCs) iniciava a modelagem do sistema com UCs (Estudo Fase 1), o Grupo A (ICs) era treinado nos elementos específicos da modelagem com ICs (Treinamento UC –

Suplemento) para, na seqüência, modelar o sistema com ICs[60]. Cada grupo acabou gastando aproximadamente o mesmo tempo nessa modelagem. Posteriormente, seguiu-se outra etapa de treinamento, desta vez para a obtenção do MD a partir dos UCs (Treinamento MUC → MD, no caso do Grupo B), ou a partir dos ICs (Treinamento MIC → MD, no caso do Grupo A). Assim como no treinamento anterior, os participantes receberam um kit contendo o material instrucional de apoio. Concluído esse treinamento, os grupos passaram à Fase 2 do estudo: a elaboração do MD. O Grupo A demandou mais tempo nessa fase, tendo concluído os seus MDs ao longo da manhã do segundo dia. Aos participantes foi solicitado anotar o tempo gasto na elaboração dos modelos, para a posterior comparação do esforço (número de horas) despendido em cada tipo de modelagem.

Tabela 6.8 – Cronograma de execução do estudo

PRIMEIRO DIA (23/junho/07)			
	Hora	**GRUPO A (ICs)**	**GRUPO B (UCs)**
MANHÃ	8:00 – 9:00	Treinamento UC – Inicial KIT 1	Treinamento UC - Inicial KIT 1
	9:00 – 10:00	Treinamento UC – Suplemento KIT 1a	Estudo Fase 1: Elaboração do MUC
	10:00 – 10:15	Coffee Break	Coffee Break
	10:15 – 11:00	Estudo Fase 1: Elaboração do MIC	Estudo Fase 1: Elaboração do MUC
	11:00 – 12:00	Estudo Fase 1: Elaboração do MIC	Estudo Fase 1: Elaboração do MUC
	12:00 – 13:00	Estudo Fase 1: Elaboração do MIC	Treinamento MUC → MD KIT 2b
	13:00 – 14:30	Almoço	Almoço
TARDE	14:30 – 15:30	Treinamento MIC → MD KIT 2a	Estudo Fase 2: Elaboração do MD
	15:30 – 16:30	Estudo Fase 2: Elaboração do MD	Estudo Fase 2: Elaboração do MD
	16:30 – 17:00	Coffee Break	Coffee Break
	17:00 – 18:00	Estudo Fase 2: Elaboração do MD	Estudo Fase 2: Elaboração do MD
	18:00 – 18:30	Estudo Fase 2: Elaboração do MD	---

SEGUNDO DIA (24/junho/07)			
	Hora	**GRUPO A (ICs)**	**GRUPO B (UCs)**
MANHÃ	9:00 – 10:00	Estudo Fase 2: Elaboração do MD (cont.)	---
	10:00 – 11:00	Estudo Fase 2: Elaboração do MD	Entrevistas
	11:00 – 12:00	Entrevistas	Entrevistas

[60] O Apêndice 3 mostra exemplares dos ICs produzidos.

Com os MDs terminados, passou-se à fase de entrevistas com pares de participantes pertencentes a um mesmo grupo. Isto foi feito no dia 23 de junho de 2007 com os pares P1-P3, P4-P5, P4-P6 e P5-P6. Como o participante P2 não conseguiu entregar o modelo de classes no prazo da execução do estudo, mas sim na semana seguinte, as entrevistas deste participante com os demais do seu grupo foram feitas em um momento posterior. A entrevista do par P1-P2 ocorreu em 24 de outubro de 2007, e a entrevista do par P2-P3 em 26 de outubro de 2007[61]. Essas entrevistas tiveram como finalidade inicial identificar as classes representativas de abstrações comuns entre os modelos do par entrevistado. Uma vez identificadas essas classes, as entrevistas prosseguiram com a identificação dos atributos, operações e associações coincidentes, para as classes comuns.

6.3.8 Análise de Ameaças

Na fase de planejamento do estudo foram consideradas precauções a serem tomadas para uma execução adequada do mesmo. Nesta seção, são relatadas observações registradas durante a execução do estudo, com potencial de influenciar seus resultados. São basicamente ameaças à validade interna dos resultados.

Um problema observado foi o entendimento muito variado sobre o sistema a ser modelado e seu domínio, a despeito de um sumário sobre isso ter sido distribuído aos participantes (Apêndice 3). Alguns modelos produzidos pelos participantes não continham todas as funcionalidades e dados que foram citados no sumário, bem como ocorreu a adição de funcionalidades e dados que não estavam no sumário. Em vista disso, sugere-se que futuros estudos incorporem mecanismos de uniformização da assimilação do conhecimento sobre o sistema e seu domínio, como por exemplo, uma dinâmica de grupo.

O participante P2 não conseguiu entregar os modelos na data prevista. Talvez o treinamento ministrado não tenha sido suficiente para que ele tivesse a mesma desenvoltura que os dois outros participantes do seu grupo. Ele completou os modelos nos dias subseqüentes, sem ter contato com os demais colegas do seu grupo, e sob supervisão dos experimentadores. Em vista disso, o tempo gasto pelo participante P2 na elaboração dos modelos não foi considerado no cálculo da média dos esforços do grupo A.

[61] O participante P1 reside em outra cidade. Isso causou a demora na marcação das entrevistas.

Ocorreram dúvidas durante o treinamento e ao longo da construção dos modelos, tanto por participantes do grupo A como por participantes do grupo B. Foi tomado muito cuidado nas respostas, para não interferir no resultado do estudo. Estas eram dúvidas referentes ao domínio do sistema e ao uso das técnicas correspondentes utilizadas no estudo. Para tentar amenizar esse problema, em futuros estudos, é recomendável um treinamento mais prolongado e com um maior número de exemplos.

Para análise das uniformidades, algumas correções nos modelos de alguns participantes foram necessárias. No modelo do participante P2, foram encontrados dois erros referentes à interpretação das regras. O primeiro erro foi o fato do mesmo ter aplicado a regra R4d, sendo que ele já tinha aplicado a regra R4a, desta maneira gerando operações cujo nome começa com "cadastrar", que não deveriam estar no modelo. Além do erro de aplicação da regra R4d, o participante P2 não aplicou a regra de consistência, que verifica a utilidade de dados dos fluxos. Isto resultou em um número pequeno de atributos, o que foi percebido pelos experimentadores. Por se tratarem de erros primários, resultantes de falta de atenção na aplicação das regras, o participante P2 fez a revisão do modelo, corrigindo a aplicação da regra R4d e fazendo a verificação correspondente à regra de consistência.

O participante P3 introduziu a operação "Financeiro()" na classe do sistema, sendo que a mesma foi retirada pelo próprio participante durante a entrevista, pois ficou claro para ele que essa operação não é gerada por quaisquer das regras de derivação.

Também durante a entrevista, o participante P1 percebeu que tinha aplicado a regra para derivação de associações de forma equivocada, e eliminou algumas das associações (entre classes) que tinha criado com ela.

Por fim, como a seleção dos participantes não pôde ser aleatória, mas simplesmente aproveitou a disponibilidade de algumas poucas pessoas, era praticamente inevitável a ocorrência de fatores potencialmente perturbadores da validade interna dos resultados do estudo. As respostas dos participantes ao questionário de caracterização (Tabela 6.7 – seção 6.3.2) apontaram as seguintes diferenças entre os grupos do estudo:

1. O Grupo B (UCs) é mais homogêneo do que o Grupo A (ICs)

O ideal seria ter ambos os grupos igualmente homogêneos nos quesitos levantados, já que o estudo compara a uniformidade dos MDs produzidos pelos dois grupos, variando apenas a técnica empregada por cada um. Dessa forma se reduziria o

risco de outros fatores (além da técnica empregada) influenciarem os resultados. Entretanto, devido à seleção por conveniência dos participantes, apenas o Grupo B resultou (quase) homogêneo[62]. Considerando os quesitos utilizados na caracterização dos participantes (tempo transcorrido desde a formatura, nível de escolaridade, conhecimento e experiência prévios no domínio e na modelagem de sistema similares, entre outros), é razoável supor que quanto mais homogêneo um grupo, maiores são as chances dele produzir MDs mais uniformes entre si. Assim, o fato do Grupo B (UCs) ser mais homogêneo que o Grupo A (ICs) representa uma situação desfavorável à hipótese do estudo; ou seja, a se considerar apenas esse fator, o Grupo B (UCs) tenderia a produzir MDs mais uniformes entre si do que o Grupo A (ICs).

2. O Grupo B (UCs) tem maior nível de escolaridade

De uma maneira geral, é razoável supor que quanto maior o nível de escolaridade dos participantes, maior o seu conhecimento e domínio da técnica utilizada e, portanto, mais capacitados estariam para produzir MDs mais uniformes entre si. Conseqüentemente, podemos considerar essa diferença como sendo desfavorável à hipótese do estudo; ou seja, a se considerar apenas esse fator, o Grupo B (UCs) tenderia a produzir MDs mais uniformes entre si do que o Grupo A (ICs).

3. O Grupo A (ICs) tem mais tempo de exercício profissional

Em contraposição às duas diferenças anteriores, o maior tempo de exercício profissional que alguns dos participantes do Grupo A (ICs) têm em relação aos participantes do Grupo B (UCs) tende a favorecer a hipótese do estudo: o Grupo A ser capaz de produzir MDs mais uniformes entre si. Isso porque, em princípio, os participantes do Grupo A teriam maior maturidade e domínio da técnica empregada.

Dado o reduzido número de participantes em cada grupo (3), não foi possível utilizar o princípio geral de blocagem para isolar essas diferenças e ter uma avaliação experimental da sua influência nos resultados do estudo.

6.3.9 Verificação das Hipóteses

Esforço

A Tabela 6.9 mostra os dados coletados sobre o esforço (número de horas) empregado por cada participante na elaboração dos modelos.

[62] Exceto no quesito "Experiência Profissional" (vide Tabela 6.7 – seção 6.3.2).

Tabela 6.9 – Esforço de cada participante

Participante (P_i)	Técnica	Esforço (horas)
P1	ICs	7:21
P2[63]	ICs	-
P3	ICs	6:40
P4	UCs	5:35
P5	UCs	5:35
P6	UCs	5:50

A comparação do esforço foi feita a seguir, comparando-se as médias $\mu_{\text{E-ICs}}$ e $\mu_{\text{E-UCs}}$. Na seção 6.3.3, vimos que:

- $\mu_{\text{E-ICs}} = \frac{E_1 + E_2 + E_3}{3}$; e
- $\mu_{\text{E-UCs}} = \frac{E_4 + E_5 + E_6}{3}$.

onde E_i representa o número de horas gastas pelo participante i.

Assim, com base nos valores de E_i da Tabela 6.9, tem-se:

$$\mu_{\text{E-ICs}} = \frac{7:21 + 6:40}{2} = \frac{14:01}{2} \cong 7:00$$

$$\mu_{\text{E-UCs}} = \frac{5:35 + 5:35 + 5:50}{3} = \frac{17:00}{3} = 5:40$$

Portanto, a técnica de ICs demandou, em média, aproximadamente 01h46min (24%) a mais de esforço para a elaboração dos modelos, do que a técnica de UCs.

Granularidade

A Tabela 6.10 mostra os dados coletados sobre a granularidade dos modelos de classes obtidos pelos participantes, com a utilização de cada uma das técnicas investigadas.

[63] Desconsiderado (vide seção 6.3.8).

Tabela 6.10 – Número de classes por participante

Participante (P_i)	Técnica	Granularidade (N^o de Classes – C_i)
P1	ICs	12
P2	ICs	7
P3	ICs	9
P4	UCs	13
P5	UCs	9
P6	UCs	8

A comparação da granularidade dos modelos de classes foi feita a seguir, comparando-se as médias $\mu_{G\text{-}ICs}$ e $\mu_{G\text{-}UCs}$. Na seção 6.3.3, vimos que:

- $\mu_{G\text{-}ICs} = \frac{C_1 + C_2 + C_3}{3}$
- $\mu_{G\text{-}UCs} = \frac{C_4 + C_5 + C_6}{3}$.

onde C_i representa o número de classes existentes no modelo do participante i.

Assim, com base nos valores de C_i da Tabela 6.10, tem-se:

$$\mu_{G\text{-}ICs} = \frac{12 + 7 + 9}{3} = \frac{28}{3} \cong 9{,}3333$$

$$\mu_{G\text{-}UCs} = \frac{13 + 9 + 8}{3} = \frac{30}{3} = 10.$$

Portanto, a técnica de ICs produziu, em média, modelos com menos classes do que a técnica tradicional de UCs (aproximadamente 8% menos classes).

Uniformidade

A Tabela 6.11 mostra os dados coletados sobre as abstrações, considerando-se os MDs dos participantes de um mesmo grupo, tomados aos pares.

Uma abstração representada por uma única classe em um modelo pode corresponder a mais de uma classe em outro modelo. Isso foi observado na comparação do modelo de P1 com os modelos dos demais participantes do grupo. Por exemplo, na comparação P1-P2 ficou evidente que a abstração representada pela classe *Pagamento* no modelo de P2, está especializada nas classes *Despesas* e *Crédito Bancário* no modelo de P1. Com isso, a contagem de abstrações no modelo de P2, em comparação

com o modelo de P1, considerou a classe "Pagamento" "coincidindo" duas vezes – uma com a classe *Despesas*, e outra com a classe *Crédito Bancário*. Algo análogo ocorreu na comparação do modelo de P1 com o modelo de P3. Os números em negrito na Tabela 6.4 destacam a variação na contagem de abstrações dos modelos de P2 e P3, na comparação com o modelo de P1, em decorrência desse fenômeno.

Tabela 6.11 – Abstrações nos modelos de domínio (MDs)

Participantes Pi, Pj	**Abstrações no modelo de Pi (A_i)**	**Abstrações no modelo de Pj (A_j)**	**Abstrações coincidentes ($A'_{i,j}$)**	**Unif-$A_{i,j}$**
P1, P2	12	**9**	8	0,6153
P1, P3	12	**11**	10	0,7692
P2, P3	**7**	**9**	6	0,6000
P4, P5	13	9	7	0,4667
P4, P6	13	8	6	0,4000
P5, P6	9	8	5	0,4167

A comparação da uniformidade de abstração entre os modelos de classes de cada grupo (ou seja, obtidos com uma mesma técnica) foi feita a seguir, comparando-se as médias $\mu_{Ua\text{-}ICs}$ e $\mu_{Ua\text{-}UCs}$. Na seção 6.3.3, vimos que:

- $\mu_{Ua\text{-}ICs} = \dfrac{\text{Unif-A}_{1,2} + \text{Unif-A}_{1,3} + \text{Unif-A}_{2,3}}{3}$; e
- $\mu_{Ua\text{-}UCs} = \dfrac{\text{Unif-A}_{4,5} + \text{Unif-A}_{4,6} + \text{Unif-A}_{5,6}}{3}$.

onde Unif-$A_{i,j}$ representa a uniformidade de abstrações entre os modelos produzidos pelos participantes i e j, que aplicaram a mesma técnica (ou seja, pertenceram ao mesmo grupo experimental), sendo dada pela fórmula (seção 6.3.1):

$$\text{Unif-A}_{i,j} = \frac{A'_{i,j}}{(A_i + A_j) - (A'_{i,j})}.$$

Assim, com base nos valores da tabela 6.11, tem-se:

$$\mu_{Ua\text{-}ICs} = \frac{\frac{8}{12+9-8} + \frac{10}{12+11-10} + \frac{6}{7+9-6}}{3} \cong \frac{0,6153 + 0,7692 + 0,6}{3} = 0,6615$$

$$\mu_{Ua\text{-}UCs} = \frac{\frac{7}{13+9-7} + \frac{6}{13+8-6} + \frac{5}{9+8-5}}{3} \cong \frac{0,4667 + 0,4 + 0,4167}{3} = 0,4278$$

Portanto, a uniformidade de abstrações entre MDs produzidos pelo grupo que utilizou ICs foi, em média, aproximadamente 55% maior do que entre os MDs obtidos com a técnica tradicional de UCs.

Além disso, reforçam esse resultado as constatações de que: 1) a maior de todas as uniformidades de abstrações, obtida entre modelos de participantes de um mesmo grupo, foi obtida aplicando ICs; e 2) a menor uniformidade de abstrações, entre os pares de modelos construídos com ICs, é maior do que a maior uniformidade de abstrações, entre os pares de modelos construídos com a outra técnica (UCs).

A tabela 6.12 mostra os dados coletados sobre as associações, considerando-se as classes que representam abstrações coincidentes, entre os modelos dos participantes de um mesmo grupo, tomados aos pares.

Tabela 6.12 – Associações nos modelos de domínio (MDs)

Pares Pi, Pj	**Associações no modelo de Pi ($L_{i,j}$)**	**Associações no modelo de Pj ($L_{j,i}$)**	**Associações coincidentes ($L'_{i,j}$)**	**Unif-$L_{i,i}$**
P1, P2	5	6	3	0,3750
P1, P3	10	**7**	6	0,5455
P2, P3	6	4	4	0,6667
P4, P5	4	5	3	0,5000
P4, P6	2	7	2	0,2857
P5, P6	2	4	1	0,2000

Como visto na comparação de uniformidades das abstrações dos MDs, é possível que associações ocorram entre classes que representam uma abstração em um modelo, e que corresponda a mais de uma classe em outro modelo. Desta maneira, a associação "Pagamento-FormaPagamento" do modelo do participante P3 "coincidiu" com as associações "Despesa-Parcela" e "Credito-Parcela" do participante P1.

A comparação da uniformidade de associações entre os modelos de classes de cada grupo (ou seja, obtidos com uma mesma técnica) foi feita a seguir, comparando-se as médias $\mu_{\text{Ul-ICs}}$ e $\mu_{\text{Ul-UCs}}$. Na seção 6.3.3, vimos que:

- $\mu_{\text{Ul-ICs}} = \dfrac{\text{Unif-L}_{1,2} + \text{Unif-L}_{1,3} + \text{Unif-L}_{2,3}}{3}$; e
- $\mu_{\text{Ul-UCs}} = \dfrac{\text{Unif-L}_{4,5} + \text{Unif-L}_{4,6} + \text{Unif-L}_{5,6}}{3}$.

onde Unif-$L_{i,j}$ representa a uniformidade de associações entre os modelos produzidos pelos participantes i e j, que aplicaram a mesma técnica (ou seja, pertenceram ao mesmo grupo experimental), sendo dada pela fórmula (seção 6.3.1):

$$\text{Unif-}L_{i,j} = \frac{L'_{i,j}}{(L_i + L_j) - (L'_{i,j})}.$$

Assim, com base nos valores da Tabela 6.12, tem-se:

$$\mu_{\text{Ul-ICs}} = \frac{\frac{3}{5+6-3} + \frac{6}{10+7-6} + \frac{4}{6+4-4}}{3} \cong \frac{0{,}375 + 0{,}5455 + 0{,}6667}{3} \cong 0{,}5291$$

$$\mu_{\text{Ul-ICs}} = \frac{\frac{3}{4+5-3} + \frac{2}{2+7-2} + \frac{1}{2+4-1}}{3} \cong \frac{0{,}5 + 0{,}2857 + 0{,}2}{3} \cong 0{,}3286$$

Portanto, a uniformidade de associações entre MDs produzidos pelo grupo que utilizou ICs foi, em média, aproximadamente 61% maior do que entre os MDs obtidos com a técnica tradicional de UCs.

A Tabela 6.13 mostra os dados coletados sobre os atributos, considerando-se as classes que representam abstrações coincidentes, entre os modelos dos participantes de um mesmo grupo, tomados aos pares.

Tabela 6.13 – Atributos nos modelos de domínio (MDs)

Pares Pi, Pj	**Atributos no modelo de Pi ($At_{i,j}$)**	**Atributos no modelo de Pj ($At_{j,i}$)**	**Atributos coincidentes ($At'_{i,j}$)**	**Unif-$At_{i,j}$**
P1, P2	28	24	15	0,4054
P1, P3	38	24	16	0,3478
P2, P3	24	14	10	0,3571
P4, P5	34	22	14	0,3333
P4, P6	28	33	18	0,4186
P5, P6	24	14	9	0,3103

A comparação da uniformidade de atributos entre os modelos de classes de cada grupo (ou seja, obtidos com uma mesma técnica) foi feita a seguir, comparando-se as médias $\mu_{\text{Uat-ICs}}$ e $\mu_{\text{Uat-UCs}}$. Na seção 6.3.3, vimos que:

- $\mu_{\text{Uat-ICs}} = \frac{\text{Unif-At}_{1,2} + \text{Unif-At}_{1,3} + \text{Unif-At}_{2,3}}{3}$; e

- $\mu_{\text{Uat-UCs}} = \dfrac{\text{Unif-At}_{4,5} + \text{Unif-At}_{4,6} + \text{Unif-At}_{5,6}}{3}$.

onde $\text{Unif-At}_{i,j}$ representa a uniformidade de atributos entre os modelos produzidos pelos participantes i e j, que aplicaram a mesma técnica (ou seja, pertenceram ao mesmo grupo experimental), sendo dada pela fórmula (seção 6.3.1):

$$\text{Unif-At}_{i,j} = \frac{\text{At}'_{i,j}}{(\text{At}_i + \text{At}_j) - (\text{At}'_{i,j})}.$$

Assim, com base nos valores da Tabela 6.13, tem-se:

$$\mu_{\text{Uat-ICs}} = \frac{\frac{15}{28+24-15} + \frac{16}{38+24-16} + \frac{10}{24+14-10}}{3} \cong \frac{0{,}4054 + 0{,}3478 + 0{,}3571}{3} = 0{,}3701$$

$$\mu_{\text{Uat-UCs}} = \frac{\frac{14}{34+22-14} + \frac{18}{28+33-18} + \frac{9}{24+14-9}}{3} \cong \frac{0{,}3333 + 0{,}4186 + 0{,}3103}{3} = 0{,}3541$$

Portanto, a uniformidade de atributos entre MDs produzidos pelo grupo que utilizou ICs foi, em média, aproximadamente 5% maior do que entre os MDs obtidos com a técnica tradicional de UCs.

A Tabela 6.14 mostra os dados coletados sobre as operações, considerando-se as classes que representam abstrações coincidentes, entre os modelos dos participantes de um mesmo grupo, tomados aos pares.

Tabela 6.14 – Operações nos modelos de domínio (MDs)

Pares Pi, Pj	**Operações no modelo de Pi ($O_{i,j}$)**	**Operações no modelo de Pj ($O_{j,i}$)**	**Operações coincidentes ($O'_{i,j}$)**	**Unif-$O_{i,i}$**
P1, P2	13	12	9	0,5625
P1, P3	15	15	14	0,8750
P2, P3	15	13	11	0,6471
P4, P5	27	16	7	0,1944
P4, P6	22	19	5	0,1389
P5, P6	14	14	2	0,0769

A técnica de modelagem com ICs determina a criação de uma operação construtora em cada classe do modelo, com o mesmo nome da classe. Cada operação construtora gerou uma coincidência de operações. A técnica tradicional de UCs não determina a introdução de operações construtoras, embora pudesse fazê-lo da mesma

forma que na MIR. Por exemplo, o participante P5 não incluiu nenhuma operação construtora em suas classes. Em vista disso, decidiu-se calcular também a uniformidade de operações sem levar em conta as operações construtoras. As operações designadas por "cadastra ...", dos participantes P4 e P6, foram consideradas operações construtoras, já que elas têm o mesmo propósito. A Tabela 6.15 apresenta os dados coletados sobre as operações quando não são computadas as operações construtoras nos modelos de ambos os grupos.

Tabela 6.15 – Operações nos modelos, excluídas as operações construtoras

Pares Pi, Pj	**Operações no modelo de Pi ($O_{i,j}$)**	**Operações no modelo de Pj ($O_{j,i}$)**	**Operações coincidentes ($O'_{i,j}$)**	**Unif-$O_{i,j}$**
P1, P2	7	7	3	0,2727
P1, P3	7	7	6	0,7500
P2, P3	10	8	6	0,5000
P4, P5	24	16	7	0,2121
P4, P6	17	15	1	0,0323
P5, P6	14	12	2	0,0833

A comparação da uniformidade de operações entre os modelos de classes de cada grupo (ou seja, obtidos com uma mesma técnica) foi feita a seguir, comparando-se as médias $\mu_{\text{Uo-ICs}}$ e $\mu_{\text{Uo-UCs}}$. Na seção 6.3.3, vimos que:

- $\mu_{\text{Uo-ICs}} = \dfrac{\text{Unif-O}_{1,2} + \text{Unif-O}_{1,3} + \text{Unif-O}_{2,3}}{3}$; e
- $\mu_{\text{Uo-UCs}} = \dfrac{\text{Unif-O}_{4,5} + \text{Unif-O}_{4,6} + \text{Unif-O}_{5,6}}{3}$.

onde Unif-$O_{i,j}$ representa a uniformidade de operações entre os modelos produzidos pelos participantes i e j, que aplicaram a mesma técnica (ou seja, pertenceram ao mesmo grupo experimental), sendo dada pela fórmula (seção 6.3.1):

$$\text{Unif-O}_{i,j} = \frac{O'_{i,j}}{(O_i + O_j) - (O'_{i,j})}.$$

Primeiramente, com base nos valores da Tabela 6.14, tem-se:

$$\mu_{\text{Uo-ICs}} = \frac{\frac{9}{13+12-9} + \frac{14}{15+15-14} + \frac{11}{15+13-11}}{3} \cong \frac{0{,}5625 + 0{,}875 + 0{,}6471}{3} \cong 0{,}6949$$

$$\mu_{\text{Uo-UCs}} = \frac{\frac{7}{27+16-7} + \frac{5}{22+19-5} + \frac{2}{14+14-2}}{3} \cong \frac{0{,}1944 + 0{,}1389 + 0{,}0769}{3} \cong 0{,}1367$$

E agora, com base nos valores da Tabela 6.15, que desconsidera as operações construtoras, tem-se:

$$\mu_{\text{Uo-ICs}} = \frac{\frac{3}{7+7-3} + \frac{6}{7+7-6} + \frac{6}{10+8-6}}{3} \cong \frac{0{,}2727 + 0{,}75 + 0{,}5}{3} \cong 0{,}5076$$

$$\mu_{\text{Uo-UCs}} = \frac{\frac{7}{24+16-7} + \frac{1}{17+15-1} + \frac{2}{14+12-2}}{3} \cong \frac{0{,}2121 + 0{,}0323 + 0{,}0833}{3} \cong 0{,}1092$$

Portanto, a uniformidade de operações entre MDs produzidos pelo grupo que utilizou ICs foi 408% maior do que entre os MDs obtidos com a técnica tradicional de UCs (ou 365%, se forem desconsideradas as operações construtoras).

Por fim, a tabela 6.16 consolida os dados sobre a uniformidade de associações, atributos, operações e representacional, considerando-se os modelos dos participantes de um mesmo grupo, tomados aos pares.

Tabela 6.16 – Uniformidade de associações, atributos, operações e representacional

Pares Pi, Pj	**Unif-$L_{i,j}$**	**Unif-$At_{i,j}$**	**Unif-$O_{i,j}$**		**Unif-$R_{i,j}$**	
			(1)	(2)	(1)	(2)
P1, P2	0,3750	0,4054	0,5625	0,2727	0,4476	0,3510
P1, P3	0,5455	0,3478	0,8750	0,7500	0,5894	0,5478
P2, P3	0,6667	0,3571	0,6471	0,5000	0,5570	0,5079
P4, P5	0,5000	0,3333	0,1944	0,2121	0,3426	0,3485
P4, P6	0,2857	0,4186	0,1389	0,0323	0,2811	0,2455
P5, P6	0,2000	0,3103	0,0769	0,0833	0,1957	0,1979

Legenda: (1) Considerando operações construtoras
(2) Não considerando operações construtoras

A comparação da uniformidade representacional entre os MDs de cada grupo (ou seja, obtidos com uma mesma técnica) foi feita a seguir, comparando-se as médias $\mu_{\text{Ur-ICs}}$ e $\mu_{\text{Ur-UCs}}$. Na seção 6.3.3, vimos que:

- $\mu_{\text{Ur-ICs}} = \frac{\text{Unif-R}_{1,2} + \text{Unif-R}_{1,3} + \text{Unif-R}_{2,3}}{3}$; e
- $\mu_{\text{Ur-UCs}} = \frac{\text{Unif-R}_{4,5} + \text{Unif-R}_{4,6} + \text{Unif-R}_{5,6}}{3}$.

onde $\text{Unif-R}_{i,j}$ representa a uniformidade representacional entre os modelos produzidos pelos participantes i e j, que aplicaram a mesma técnica (ou seja, pertenceram ao mesmo grupo experimental), sendo dada pela fórmula (seção 6.3.1): $\text{Unif-R}_{i,j} = \frac{\text{Unif-L}_{i,j} + \text{Unif-At}_{i,j} + \text{Unif-O}_{i,j}}{3}$.

Primeiramente, com base nos valores da tabela 6.16, e considerando todas as operações, tem-se:

$$\mu_{\text{Ur-ICs}} = \frac{0{,}4476 + 0{,}5894 + 0{,}5570}{3} \cong 0{,}5313$$

$$\mu_{\text{Ur-UCs}} = \frac{0{,}3426 + 0{,}2811 + 0{,}1957}{3} \cong 0{,}2731$$

E agora, com base nos valores da tabela 6.16, e desconsiderando as operações construtoras, tem-se:

$$\mu_{\text{Ur-ICs}} = \frac{0{,}3510 + 0{,}5478 + 0{,}5079}{3} = 0{,}4689$$

$$\mu_{\text{Ur-UCs}} = \frac{0{,}3485 + 0{,}2455 + 0{,}1979}{3} = 0{,}2640$$

Portanto, a uniformidade representacional entre MDs produzidos pelo grupo que utilizou ICs foi, em média, aproximadamente 95% maior do que entre os modelos obtidos com a técnica tradicional de UCs (ou 78%, se forem desconsideradas as operações construtoras).

6.3.10 Considerações sobre o Estudo 2

O estudo produziu algumas evidências comparativas sobre o esforço empregado em cada técnica, e sobre a granularidade e uniformidade dos MDs construídos utilizando ICs e UCs.

Quanto ao esforço empregado em cada técnica, o estudo indica que ele é maior na técnica de ICs, sendo a diferença da ordem de 24%, em média.

Quanto à granularidade, o estudo indica que os MDs construídos a partir de ICs apresentam menor granularidade do que os MDs construídos a partir de UCs, sendo essa diferença, em média, de 8%.

Quanto à uniformidade dos MDs produzidos com cada técnica, o estudo indica que os MDs construídos a partir de ICs são mais uniformes entre si do que os obtidos a partir dos UCs. Isso foi verificado no nível semântico (conceitual) pela comparação das abstrações existentes nos modelos de classes, bem como no nível representacional, pela comparação dos atributos, associações e operações envolvidas nas classes correspondentes às abstrações coincidentes entre (pares de) modelos. A Tabela 6.17 dá uma visão geral desses resultados.

Dois resultados mostrados na tabela 6.17 devem ser destacados. Um deles é o acréscimo relativamente pequeno obtido na uniformidade de atributos, para os MDs construídos a partir dos ICs (+5%). Uma explicação para isso é que vários desses atributos[64] aparecem explicitamente no sumário do sistema (Apêndice 3), o que provavelmente facilitou a identificação dos mesmos pelos participantes, independentemente da técnica utilizada.

Tabela 6.17 – Comparação das médias de uniformidade

Técnica	**Uniformidade de abstrações (μ_{Ua})**		**Uniformidade Representacional**							
			Associações (μ_{Ul})		**Atributos (μ_{Uat})**		**Operações (μ_{Uo})**		**Global (μ_{Ur})**	
ICs	0,6615	**+ 55%**	0,5291	**+61%**	0,3701	**+5%**	0,6949(1) 0,5076(2)	**+408%**(1) **+365%**(2)	0,5313(1) 0,4689(2)	**+95%**(1) **+78%**(2)
UCs	0,4278	-	0,3286	-	0,3541	-	0,1367(1) 0,1092(2)	-	0,2731(1) 0,2640(2)	-

Legenda: (1) Considerando operações construtoras; (2) Não considerando operações construtoras.

O outro resultado a destacar na Tabela 6.17 é o grande acréscimo na uniformidade de operações obtido na modelagem com ICs (+365%). Algumas considerações podem ser feitas sobre esse resultado. Em primeiro lugar, o sumário distribuído aos participantes é muito menos explícito com relação a operações do que com relação a atributos. Isso significa que a orientação metodológica fornecida por cada técnica, para a obtenção das operações das classes, tende a ser decisiva do ponto de vista da uniformidade dos MDs produzidos a partir dela. Conforme discutido no capítulo 4, falta orientação metodológica precisa para a obtenção das operações a partir dos UCs. Uma conseqüência imediata disso é que a determinação das operações das classes fica a depender sobremaneira da interpretação do modelador, o que por sua vez, promove a discrepância entre os MDs obtidos por diferentes modeladores, para um mesmo sistema. Já na MIC, a situação é diferente. Há uma orientação precisa para a

[64] Por exemplo: nome, endereço, pessoa de contato, telefone e o e-mail de um credor.

obtenção das operações, dada pelas quatro regras que permitem derivar os diversos tipos de operações a partir dos ICs (seção 5.4.2). Essas regras são determinísticas, ou seja, constituem um mapeamento funcional do MIC no MD: a cada aplicação de uma regra, uma operação é identificada univocamente. Além disso, essa aplicação pode ser feita com um nível médio de automatismo de 75% (seção 5.5.5). Portanto, trata-se de uma situação bastante distinta daquela que acontece na modelagem tradicional com UCs.

Os resultados obtidos nesse estudo devem ser considerados à luz das ameaças descritas na seção 6.3.8, especialmente, as diferenças entre os grupos que aplicaram as duas técnicas – UCs e ICs. Por isso, seria interessante reproduzir o estudo com grupos maiores, visando isolar (blocar) os diversos fatores potencialmente perturbadores, e assim, obter resultados mais confiáveis. Também, um número maior de participantes motivaria a utilização dos métodos estatísticos adequados para a análise dos resultados obtidos.

6.4 Estudo 3 (EC-3): Adequação do MD

6.4.1 Objetivos e Questões de Investigação

Este estudo de caso foi uma oportunidade para o teste das regras de derivação no desenvolvimento de um sistema de porte médio, real, para uma empresa de medicina e engenharia de segurança do trabalho. Até então, as regras não tinham passado pelo crivo de sua aplicação a um sistema complexo e de maior porte.

As questões a serem investigadas no estudo de caso eram: as regras seriam capazes de derivar uma visão MD adequada a partir do ICs do sistema? Se não, que ajustes ou refinamentos deveriam ser feito nas mesmas para se conseguir isso?

A interpretação dada nesse estudo para a adequação da visão MD do MIC está baseada na definição proposta por GLINZ (2000) para o conceito de adequação de um MD em relação ao MUC correspondente. Supondo que o MIC reflete as necessidades dos *stakeholders* para o sistema a ser desenvolvido, a visão MD do MIC é adequada se nenhuma das duas situações seguintes ocorrer:

1. O MD passar a incluir, ao longo do desenvolvimento do sistema, por decisão justificada do modelador com base no MIC, um elemento que, apesar de tratado pelas regras, não pôde ser obtido através delas. Se isso acontecer, então existe uma incompletude (omissão) na visão MD do MIC.

2. O MD ter um elemento obtido com a aplicação das regras, excluído ou alterado ao longo do desenvolvimento do sistema, por decisão justificada do modelador com base no MIC. Se isso acontecer, existe uma inconsistência na visão MD do MIC.

Portanto, por essa definição, para se ter uma avaliação sobre a adequação da visão MD do MIC, será preciso registrar e analisar toda alteração que o MD venha a sofrer, desde a sua derivação utilizando as regras (logo após a elaboração do MIC), até o término do processo de implementação do sistema. Se a alteração puder ser enquadrada em uma das duas situações acima, então ela indica a existência de uma incompletude (situação 1) ou inconsistência (situação 2) da visão MD e, nesse caso, o passo seguinte deverá ser analisar o que levou à ocorrência da situação indesejada e propor alterações nas regras, de forma a evitar que a situação volte a acontecer.

6.4.2 O Sistema Utilizado no Estudo

O sistema utilizado no estudo é um sistema para o controle financeiro da empresa, envolvendo recebimentos de clientes por serviços prestados, e pagamentos a fornecedores dos diversos recursos demandados pela empresa. O sistema permite controlar caixa e bancos, e produz um fluxo financeiro para um período escolhido. Faz a gerência dos serviços prestados a cada cliente, relacionando-os com contratos firmados com eles. Gera notas fiscais e boletos para pagamento pelos clientes dos serviços prestados, e calcula a comissão devida a vendedores. Outra função do sistema é gerar a previsão de receitas e pagamentos para o próximo ano, de forma que os gerentes possam estabelecer, com a devida antecedência, metas a serem cumpridas para a obtenção dos resultados financeiros desejados.

O MIC do sistema contém 29 ICs, perfazendo 39 páginas de especificação. Até o momento em que esta tese estava sendo escrita (novembro/08), já tinham sido implementados 15 ICs, em aproximadamente 9.500 linhas de código PHP, representando algo em torno de 60% do sistema.

6.4.3 Seleção de Participantes, Preparação e Execução

Os participantes do estudo constituíram duas equipes, uma para cada etapa do desenvolvimento do sistema. A primeira equipe (equipe 1) foi a responsável pela elaboração dos ICs e aplicação das regras de derivação para obtenção da visão MD do MIC. Ela foi formada por três profissionais, dois deles analistas de sistemas

pertencentes ao quadro da empresa, e mais o autor desta tese (daqui em diante também referido como experimentador). Um dos analistas da empresa é graduado em Ciência da Computação, e o outro estava em fase final de conclusão do mesmo curso. A segunda equipe (equipe 2), responsável pela implementação do sistema, foi constituída por um dos profissionais da empresa que tinham participado da primeira equipe[65], e mais um estagiário admitido para colaborar na implementação. Nessa segunda etapa, o experimentador não participou diretamente, mas manteve contato e orientou os participantes quanto à coleta dos dados a serem analisados. Nenhum dos participantes, de ambas as equipes, tinha conhecimento significativo do domínio do sistema (financeiro).

A especificação dos ICs do sistema foi elaborada em conjunto pelos três participantes da equipe 1, sob a coordenação de um dos analistas da empresa. Esse analista elaborou uma versão preliminar dos ICs, versão essa que sofreu extensas modificações durante mais de um mês de discussões entre os membros da equipe, e com os *stakeholders*. Ambos os analistas da empresa tinham, à época, pelo menos 1 ano de experiência na modelagem de ICs.

Quando o MIC foi considerado estável o suficiente, a equipe 1 se preparou para a derivação da visão MD do MIC. Essa preparação envolveu a distribuição de documentos de apoio (uma cópia do MIC e das regras) para cada participante, e um treinamento sobre a aplicação das regras. O treinamento foi ministrado pelo experimentador, durante um dia e meio, visando nivelar o conhecimento dos analistas sobre as regras. Uma vez concluído o treinamento, procedeu-se a derivação do MD. Desta vez, os participantes trabalharam independentemente, embora juntos em uma mesma sala. Foram evitadas trocas de informação que pudessem influenciar o resultado de cada um, na obtenção do MD. O objetivo era ter três MDs obtidos independentemente, por cada um dos participantes, a partir da mesma especificação de ICs. Cada participante aplicou manualmente cada uma das regras, na ordem da sua apresentação, construindo diretamente o MD com o apoio da ferramenta JUDE (JUDE, 2008). A aplicação das regras de derivação ocupou aproximadamente 5 manhãs de trabalho. Uma idéia do porte dos MDs produzidos pode ser tirada pela contagem dos elementos presentes no MD do experimentador: 25 classes, 32 associações entre classes,

[65] Logo após o término da primeira etapa, o outro analista da empresa, que tinha participado da elaboração da MIC, deixou a empresa.

4 classes de associação, 81 atributos e 96 operações de classe. O Apêndice 4 apresenta o MD final – versão obtida pela evolução do MD inicialmente produzido pelas regras, durante o processo de implementação do sistema[66].

A participação do experimentador na primeira etapa do estudo (elaboração do MIC e derivação do MD) foi crucial para que o mesmo pudesse acompanhar minuciosamente e registrar tudo que acontecia durante os trabalhos, o que seria muito importante na hora de interpretar os dados coletados.

Uma vez concluída a derivação da visão MD do MIC por cada participante, os MDs foram comparados e as diferenças analisadas em conjunto pelos participantes, visando detectar eventuais problemas com as regras de derivação. O resultado dessa análise está descrita na seção 6.4.6. Em seguida, foi iniciada a implementação do sistema pela segunda equipe do estudo. Nessa etapa, foram feitas duas coletas de dados para análise posterior, cobrindo três meses de trabalho. A primeira coletou dados sobre a evolução dos modelos (MIC e MD), ao longo da implementação (da primeira versão) do sistema. Três pares de versões dos modelos foram produzidos durante a implementação, para consolidar modificações efetuadas nos modelos durante o período de apuração. O analista que participou das duas equipes foi o responsável pelo preenchimento de uma planilha, onde indicava, para cada par de versões, as modificações introduzidas em relação ao par anterior. As informações preenchidas nessa planilha foram: identificação do(s) modelo(s) modificado(s), indicação do ponto exato (IC no MIC, classe no MD) onde ocorreu a modificação, o elemento modificado, o tipo de modificação (inclusão, alteração ou exclusão), e uma análise dos motivos que levaram à modificação, e caso a modificação não pudesse ser obtida via regra de derivação, uma análise da razão dessa incapacidade. O Apêndice 5 apresenta a planilha resultante dessa coleta. Posteriormente, os dados coletados foram analisados pelo experimentador, que tirou as conclusões apresentadas na próxima seção (6.4.4).

6.4.4 Apuração e Análise da Evolução dos Modelos

Os dados coletados pelo analista, sobre a evolução dos modelos (MIC e MD) desde sua derivação através das regras, até o término da implementação da primeira versão do sistema, foram analisados pelo experimentador para avaliar a adequação da visão MD do MIC, produzida pela regras de derivação. Primeiramente o experimentador

[66] Essa evolução está descrita na Seção 6.4.4, mais adiante.

categorizou cada modificação registrada, analisou se uma das situações prejudiciais à adequação do MD ocorreu (Seção 6.4.1), e consolidou os resultados para extrair deles algumas conclusões, conforme apresentado a seguir.

Comparação entre o par de modelos gerados pelas regras e o par dos primeiros modelos de implementação:

A1) Inclusão de atributos no MD, na expectativa das próximas versões evolutivas do sistema. Análise: O MIC é construído na perspectiva do modelo mínimo de requisitos, ou seja, apenas aquelas informações necessárias às mudanças de estado (do sistema e seu ambiente) previstas nos ICs geram atributos ou operações no MD. A inclusão não é justificável com base no MIC, pois a funcionalidade correspondente não foi nele especificada. Portanto, não configura uma situação comprometedora da adequação da visão MD do MIC. Número de ocorrências: 7

A2) Inclusão de operação construtora em classe de associação. Análise: As regras não determinam isso; entretanto, parece que as classes de associação também deveriam ter a sua operação construtora. Portanto, temos uma incompletude (omissão) na visão MD do MIC. Solução: Alterar a regra R4a para também indicar a criação de uma operação construtora para cada classe de associação existente no MD. Número de ocorrências: 2

A3) Inclusão de atributo derivado com base na implementação. Análise: a inclusão teve motivação com base em requisitos de projeto e implementação e, portanto, não é justificável com base no MIC. Conseqüentemente, não configura uma situação de omissão na visão MD do MIC. Número de ocorrências: 1

A4) Alteração do tipo do elemento do MD: de atributo para operação. Análise: falha na aplicação das regras, decorrente principalmente de uma subespecificação do dicionário de itens elementares (DI), que deve dizer como se obtém a informação. Portanto, a modificação não pode ser justificada com base no MIC e, portanto, ela não configura uma situação de inconsistência da visão MD do MIC. Número de ocorrências: 1

Comparação entre o par dos primeiros modelos de implementação e o par dos modelos intermediários de implementação:

B1) Introdução de um identificador (*id_cliente*) no fluxo de entrada de um IC, e a correspondente associação no MD. Análise: omissão/equívoco do modelador, pois o identificador já tinha feito parte de uma versão anterior do mesmo IC, tendo sido

retirado sem justificativa válida. Portanto, trata-se de uma modificação não justificável com base no MIC, não configurando uma situação de omissão da visão MD do MIC. Número de ocorrências: 1.

B2) Inclusão de um atributo para um item de informação que entra em um IC, mas não tem utilização especificada nele e nem em nenhum outro IC. Análise: itens nessa situação podem significar duas coisas. Primeiro, que o item não é necessário ao desempenho de nenhuma função do sistema e, portanto, já que o MD produzido pelas regras é uma visão minimalista do domínio do sistema (ou seja, considera apenas a parte do domínio que interessa, ou é necessária, para o desempenho das funções do sistema), o item não precisa figurar em nenhum dos dois modelos (MIC e MD). Ou então, pode significar que o item é utilizado em um tipo de consulta que não é modelada, ou melhor, que fica implícita, no MIC. São consultas para a recuperação de informações (atributos) de um objeto do domínio (também conhecidas como operações "*Retrieve*", segundo a nomenclatura CRUD – *Create*, *Retrieve*, *Update*, *Delete*). Esse tipo de consulta não tem o seu próprio IC no MIC porque pode ser facilmente derivada a partir do IC que cria o objeto (IC correspondente à operação *Create*). Portanto, nesse segundo caso, existe uma omissão da visão MD do MIC. Solução: o modelador deve considerar essa necessidade (de operações *retrieve*) na hora de interpretar a regra da persistência de itens informados, na entrada (regra R3a). Número de ocorrências: 1.

B3) Introdução de atributo correspondente ao item de informação do MIC que deveria ter sido considerado a persistir. Análise: falha na aplicação da regra de persistência (R3a), decorrente principalmente de uma subespecificação do dicionário de itens elementares (DI), que deve dizer como se obtém uma informação. Portanto, a inclusão não pode ser justificada com base no MIC e, portanto, não configura omissão da visão MD do MIC. Número de ocorrências: 1.

B4) Introdução de operação da interface com o usuário. Análise: trata-se de uma operação que visa organizar informações disponíveis no MD, para visualização pelo usuário. Portanto, essa operação está relacionada com o projeto da interface do usuário e, como tal, estaria melhor localizada na camada de objetos de apresentação do sistema, e não na camada de objetos de domínio. Por isso ela não deve aparecer em nenhum dos dois modelos (MIC e MD). Não sendo essa inclusão justificável com base no MIC, ela não configura uma omissão da visão MD do MIC. Número de ocorrências: 1.

Comparação entre o par de modelos intermediários de implementação e o par de modelos finais de implementação:

C1) Introdução de item de informação no MIC correspondente a atributo já presente no MD (persistido a partir de outro IC). Análise: Modificação corretiva no MIC, que não gera efeito na visão MD. Número de ocorrências: 1.

C2) Introdução de um item de informação derivado no MIC, que por não precisar ser persistido, levou à introdução da operação correspondente no MD. Análise: Modificação corretiva no MIC, que gerou o devido efeito na visão MD, com a aplicação das regras. Número de ocorrências: 1.

C3) Eliminação de itens de informação de um fluxo de saída muito complexo de um IC, para possibilitar o enquadramento de todas as informações na folha de emissão do relatório (no entanto, os respectivos atributos foram mantidos no MD para futura utilização em outro layout de relatório). Análise: Essa modificação foi motivada por detalhes de implementação que não deveriam ter efeito sobre o MIC. Portanto, deve ser desfeita. Número de ocorrências: 1.

C4) Introdução de uma operação para a realização de uma consulta de interesse dos *stakeholders*, não prevista na MIC. Análise: A inclusão da operação não é justificável com base no MIC e, portanto, não deve ser considerada como uma omissão da visão MD. Sendo a consulta realmente necessária, o MIC deveria ser alterado para especificá-la, o que resultaria na inclusão da operação no MD (pela regra R4c). Número de ocorrências: 6.

C5) Introdução de uma operação para computar o valor de um item derivado não-persistido. Análise: não seria necessário incluir manualmente essa operação se a regra R4b tivesse sido aplicada, uma vez que ela faz exatamente isso. Portanto, não se pode considerar como uma omissão da visão MD obtida pelas regras. Número de ocorrências: 1.

Conclusão

A Tabela 6.18 resume o número de modificações ocorridas no MD, segundo o efeito sobre a adequação desse modelo.

Tabela 6.18 – Nr. de modificações segundo o efeito sobre a adequação do MD

Efeito da modificação / Situação	Contagem
Não afetam	22
Afetam / Omissão (A2, B2)	3
Afetam / Inconsistência	0
Total	25

Pelos resultados apresentados na Tabela 6.18, vemos que apenas 3 modificações afetam a adequação da visão MD do MIC, obtida pelas regras de derivação. Essas modificações (A2 e B2) visaram corrigir situações de omissão no modelo, que as regras não foram capazes de evitar. Conseqüentemente, com base nas conclusões tiradas da análise levada a cabo para as modificações em questão, duas providências se fizeram necessárias: (1) alterar o enunciado da regra R4a, responsável pela criação de operações construtoras nas classes, para que a mesma passe a indicar a criação de uma operação construtora para cada classe de associação existente no MD, e (2) incluir a orientação de que o modelador deve considerar a existência implícita no MIC, de consultas do tipo *retrieve*, na hora de interpretar a regra R3a (persistência de itens informados, na entrada).

Além desses ajustes nas regras, outros também foram feitos a partir de observações e análises realizadas pelos participantes do estudo, sobre os MDs produzidos por cada um deles. Esses outros ajustes estão descritos na Seção 6.4.6. A próxima seção analisa os testes de aceitação do sistema, realizados pelos *stakeholders*.

6.4.5 Análise dos Testes de Aceitação

De nada adiantaria ter um MD adequado (na ótica dos desenvolvedores), se o sistema resultante da implementação se revelasse muito fora das expectativas dos *stakeholders*. Em vista disso, foram organizados e executados testes de aceitação do sistema. Esses testes lançaram luz sobre a capacidade do MIC, e do MD gerado a partir dele, para propiciar um sistema afinado com os anseios dos *stakeholders*.

Os principais *stakeholders* envolvidos com o sistema foram solicitados a testar uma primeira versão do mesmo (aproximadamente 60% de todo o sistema). Eles puderam executar o sistema e conferir os resultados com o que tinha sido especificado nos ICs. Os testes foram realizados IC a IC, organizados em seqüências mais prováveis

de ocorrer no dia-a-dia da utilização do sistema (segundo a ótica dos *stakeholders*)[67]. Foram 3 dias de testes sob a supervisão do experimentador, com os problemas encontrados sendo anotados em uma planilha (Apêndice 6). A Tabela 6.19 resume os resultados encontrados.

Dos sete tipos de problemas encontrados no teste de aceitação realizado pelos *stakeholders* (Tabela 6.19), apenas três deles estão mais diretamente relacionados com a especificação do sistema através de ICs: 4- Erro de implementação (propiciado pela subespecificação do DI[68]), 5- Requisito não implementado (propiciado pela subespecificação do DI), e 7- Novo requisito (não apontado inicialmente pelos *stakeholders*). Isso porque a MIC não tem a pretensão de especificar em detalhe a interface com o usuário (problemas 1 e 2), não pode ser culpada por erros de implementação (problema 3), e nem pela evolução natural do sistema e seu ambiente (problema 6). O total de ocorrências dos problemas 4, 5 e 7 foi de 12 (50% do total). Entretanto, acreditamos que o número de ocorrências dos problemas 4 e 5 poderia ser bem menor se os itens elementares de informação tivessem sido melhor descritos no DI (não só sua denotação, mas também sua conotação – (LEITE, OLIVEIRA, 1995)). Igualmente, achamos que alguns novos requisitos poderiam ter sido elicitados desde o início, caso os usuários e *stakeholders* estivessem mais familiarizados com a técnica e a notação empregada na MIC[69].

Tabela 6.19 – Apuração dos problemas no teste de aceitação do sistema

Tipo de problema encontrado	Contagem
1- Problema na interface com o usuário	4
2- Novo requisito da interface com o usuário	1
3- Erro de implementação (programação)	6
4- Erro de implementação (propiciado pela subespecificação do DI)	5
5- Requisito não implementado (subespecificação do DI)	2
6- Requisito evolutivo	1
7- Novo requisito (não apontado inicialmente pelos *stakeholders*)	5
Total	24

Por fim, vale registrar que os *stakeholders* e usuários que participaram dos testes de aceitação demonstraram satisfação em ver o sistema realizando aquilo que eles tinham ativamente ajudado a especificar.

[67] Essas seqüencias são denominadas de *Objetivos Organizacionais* (mais informações na seção 7.4).
[68] Dicionário de Itens Elementares (parte integrante do MIC).
[69] Receberam um rápido treinamento e utilizaram a MIC pela primeira vez.

6.4.6 Outros Ajustes das Regras de Derivação

A Seção 6.4.4 indicou a necessidade de dois ajustes nas regras de derivação, para garantir a adequação da visão MD do MIC, por elas produzida. A presente seção apresenta e discute os outros ajustes motivados por observações, e decididos após longas reflexões realizadas entre os participantes do estudo, principalmente durante a etapa de derivação da visão MD do MIC. A versão das regras, apresentada na Seção 5.4.2, já incorpora todos esses ajustes.

Associação entre classes

Versão anterior:

R2a (Determinação de associações - identificadores em fluxos distintos): Todo par <id_1, id_2>, que possa ser formado com id_1 sendo um identificador gerado presente no fluxo de saída do UC, e id_2 um identificador presente no fluxo de entrada, determina uma associação entre as classes identificadas por id_1 e id_2. No caso de mais de um identificador gerado, as associações correspondentes a pares que compartilham o mesmo id_2 tendem a ser redundantes. As multiplicidades dessas associações podem ser parcialmente obtidas pela regra **R2a-M** abaixo.

R2b (Determinação de associações - identificadores no mesmo fluxo): Se o identificador *id₂* estiver no escopo de uma repetição[70], os limites, inferior e superior, da multiplicidade no lado da classe identificada por *id₂* serão, respectivamente, o número mínimo e máximo de repetições especificados no modelo informacional (ou "0" e "*", na falta dessa especificação). Caso contrário, o limite superior da multiplicidade será 1, e o limite inferior "0" se *id₂* for condicional, ou "1" se *id₂* não for condicional. A multiplicidade no lado da classe identificada por *id₁* não fica determinada.

Nova versão:

R2 (Determinação de associações): As associações entre classes estão entre aquelas indicadas pelos pares (não-ordenados) de identificadores presentes na interface informacional de cada IC (para fluxos de saída, basta considerar identificadores gerados).

[70] Isto é, se *id₂* estiver entre { }'s.

Justificativa:

Em sua utilização no presente estudo, a regra R2a produziu associações equivocadas a partir de alguns ICs com mais de um identificador gerado no fluxo de saída. Além disso, ambas as regras (R2a e R2b) prevêem a possibilidade de geração de associações redundantes no caso de mais de dois identificadores gerados ou no fluxo de entrada, respectivamente. Em vista disso, decidimos elaborar uma versão "menos determinística" para a regra, capaz de evitar os erros da versão anterior, e de possibilitar, com base no particionamento do sistema em ICs (seção 5.2), uma argumentação segura sobre a sua validade (seção 5.4.2). Entretanto, o conhecimento embutido na regra R2a foi mantido na forma de heurística aplicável quando ocorrer, na interface informacional de um IC, o padrão sintático nela previsto (seção 5.4.2).

Multiplicidade das associações

As três regras de multiplicidade que vigoravam anteriormente estavam baseadas na versão anterior das regras de criação de associações (R2a e R2b – vide acima). Quando essas últimas foram transformadas em heurísticas, as regras de multiplicidade também passaram a ter esse status, mantendo o mesmo conteúdo.

Determinação da persistência de itens de informação

Anteriormente ao estudo, as duas regras a seguir determinavam a persistência dos itens elementares de informação, informados ou derivados, respectivamente:

R3a (Determinação de persistência - item de entrada[71]): Todo item elementar não-identificador e de entrada, que for necessário em um UC distinto daquele em que o item entrou no sistema, deve ser persistido; caso contrário, não deve ser persistido.

R3b (Determinação de persistência - item calculado[72]): Para todo item elementar não-identificador, de saída e calculado, cujo valor originalmente calculado for necessário em outro UC, se houver chance desse valor não poder ser reproduzido nesse outro UC, ele deve ser persistido; senão, ele não precisa ser persistido.

Durante o desenvolvimento do estudo percebemos a necessidade de considerar também a persistência de itens informados, presentes no fluxo de saída de um IC, que representem informação histórica não passível de ser restaurada no IC. Isso deu origem

[71] O mesmo que informado.
[72] O mesmo que derivado.

a uma nova regra (que por questões de organização recebeu a denominação de R3b, passando a regra R3b acima a se denominar R3c):

R3b: (Persistência – item informado, na saída, valor histórico). Todo item elementar não-identificador e informado, presente no fluxo de saída de um IC, representando informação histórica[73] cujo valor não pode, em geral, ser restaurado nesse IC, precisa ser persistido; caso contrário, não deve ser persistido.

Regra para criação de classes de associação[74]

Na versão anterior do conjunto das regras, não havia uma regra explícita para a determinação das classes de associações, embora ela já fosse conhecida pelo autor desta tese. Essa situação foi corrigida com a modificação da regra anterior para alocação dos itens a persistir nas classes. Essa regra passou, também, a determinar a criação de classes de associação:

R3d (Alocação de atributo / Criação de classe de associação): Cada item a persistir, presente na interface informacional de um IC, deve ser alocado como atributo de uma das *classes alcançadas pelos identificadores* presentes nessa mesma interface. Caso nenhuma dessas classes comporte o item, ele deve ser alocado em uma nova classe de associação criada para esse fim, correspondente a uma associação existente entre as classes alcançadas.

Operações construtoras para classes de associação

A implementação do sistema modelado no estudo de caso fez com que se percebesse uma omissão da regra que trata da introdução de operações construtoras nas classes, no que diz respeito às classes de associação. A versão anterior dessa regra era:

R4a: Todo identificador gerado produz uma operação construtora na classe identificada por esse identificador.

Para que a regra passasse também a indicar a inclusão de uma operação construtora nas classes de associação, ela foi alterada para:

R4a (construtoras): Toda classe recebe uma operação construtora.

---

[73] Informação relativa a um estado anterior do sistema, que possivelmente não mais vigora com o mesmo valor no estado atual.

[74] Classes que representam associações entre classes.

Operações para mudança de estado do sistema e do ambiente

O estudo de caso propiciou a descoberta de que o conjunto vigente de regras para a determinação das operações do MD não tratava todas as situações possíveis. Especificamente, quando um IC tinha identificador gerado nele, e ao mesmo tempo produzia outras mudanças de estado no sistema, ou então saída, a aplicação das regras se restringia a gerar uma operação construtora que, como tal, não deve produzir outras mudanças de estado além daquelas correspondentes à inicialização de um objeto, nem produzir saídas previstas no MIC. Por isso, resolvemos alterar a versão vigente da regra R4d, para torná-la aplicável na situação descrita. A versão vigente desta regra era:

R4d: Todo UC que não tenha sido contemplado com operações originadas pelas regras (R4a) e (R4c), dá origem a uma operação para realizar mudança no estado do sistema, a ser incluída em uma das classes alcançadas pelos identificadores presentes na interface informacional do UC, ou na falta desses, a ser incluída na classe que representa o sistema.

Esta regra, em sua versão atual mais abrangente é:

R4d (mudança de estado): Todo IC que causa mudança no estado do sistema produz uma operação para realizar essa mudança e para produzir o fluxo de saída (se existir), a menos que uma operação construtora, resultante da aplicação da regra R4a, realize toda a mudança de estado requerida, e não exista saída a produzir.

6.4.7 Considerações sobre o Estudo 3

Este estudo de caso, realizado em ambiente real, permitiu observar que as regras de derivação da visão MD de um MIC são capazes de gerar MDs adequados[75] e propiciar a implementação de sistemas alinhados às necessidade dos *stakeholders*. O estudo utilizou uma versão das regras anterior àquela apresentada na Seção 5.4.2, que se mostrou deficiente em algumas situações encontradas durante a sua aplicação. Em decorrência, ajustes foram feitos para eliminar as deficiências detectadas.

6.5 Considerações Finais

Esta seção analisa a validação da hipótese geral da tese, e em particular, a contribuição dos estudos experimentais para essa validação.

[75] Segundo a definição dada na Seção 6.4.1

A hipótese geral da tese, enunciada no final do Capítulo 4 é:

"A MIR permite a derivação semi-automática de um MD "adequado", e promove uma maior uniformidade entre os MDs obtidos por diferentes modeladores, para um mesmo sistema. Isso, mantendo as principais vantagens alegadas para a MUC (seção 2.4.1) e sem exigir um aumento "excessivo" do esforço de modelagem."

Visando uma análise mais pormenorizada, podemos decompor a hipótese geral em 5 hipóteses, para validação conjuntamente:

H1) A MIC permite a derivação semi-automática de um MD;

H2) O MD assim obtido é "adequado";

H3) A MIC promove uma maior uniformidade entre os MDs obtidos por diferentes modeladores, para um mesmo sistema;

H4) A MIC mantém as principais vantagens alegadas para a MUC; e

H5) A MIC não exige um aumento "excessivo" do esforço de modelagem.

A Tabela 6.20 resume a contribuição de cada um desses estudos (exceto a dos individuais e didáticos) para a validação das hipóteses da tese.

Tabela 6.20 – Contribuição dos estudos para a validação das hipóteses

Estudo Hipótese	EC-1	EC-2	EXP-1	EXP-3	EC-3
H1		X		X	X
H2					X
H3	X	X		X	
H4					
H5				X	

Alguns comentários sobre a validação de cada uma das hipóteses são relevantes para ajudar a esclarecer as informações apresentadas na Tabela 6.20:

H1. A derivação da visão MD do MIC sempre foi feita manualmente nos estudos experimentais relatados, uma vez que não se dispõe ainda de uma ferramenta computacional para apoiar a aplicação das regras. A despeito disso, ficou claro para os participantes dos estudos EC-2, EXP-3 e EC-3 o potencial de automatização inerente as regras, confirmando assim, na prática, os resultados teóricos apresentados na Seção 5.5.5.

H2. A adequação da visão MD do MIC, obtida através das regras de derivação, foi validada tanto experimentalmente (EC-3), quanto teoricamente (Seção 5.5.5).

H3. O principal estudo para a validação dessa hipótese foi o EXP-3, embora o EC-2 também tenha contribuído com evidências favoráveis à mesma. Também o EC-1 pode ser considerado como tendo contribuído indiretamente nesse sentido, pois apesar dos participantes não terem utilizado as regras de derivação para a obtenção do MD, eles conseguiram produzir, trabalhando em conjunto mas sem contato com o responsável pela elaboração do MIC, um MD que atendeu plenamente as expectativas desse último.

H4. Esta hipótese estava, desde o início, validada por construção, na medida em que a MIC é uma especialização da MUC que mantém desta a característica de ser centrada nos usuários e em seu julgamento de valor. Embora o acréscimo da linguagem semiformal adotada na MIC para a especificação da interface informacional dos ICs (Seção 5.3), seja um componente em princípio desfavorável à hipótese, alguns autores a consideram adequada para os usuários (LAUESEN, 2002). A experiência mostrou que, com um breve treinamento, os usuários passam a trabalhar com a linguagem de maneira confortável. Mas, mesmo que assim não fosse, dado o formalismo da linguagem, poderia ser utilizado o expediente do parafraseamento automático da mesma em descrições em linguagem natural, para atender a usuários menos afeitos ao formalismo.

H5. O estudo EXP-3, bem como a experiência acumulada com a utilização da MIC nos demais estudos, mostraram que a sobrecarga de esforço para aplicar a técnica é aceitável. Além disso, essa sobrecarga deverá ser substancialmente reduzido quando estiver disponível uma ferramenta computacional adequada para apoiar a utilização da MIC, automatizando boa parte da aplicação das regras de derivação.

O estudo EXP-1 (Seção 6.2) não contribuiu diretamente no teste das hipóteses em questão, embora tenha sido importante para a avaliação do particionamento do sistema por eventos autônomos, através dos resultados obtidos sobre *cobertura funcional*.

Outros estudos poderiam ser feitos, como por exemplo, um para obter resultados estatisticamente significantes para a hipótese H3 (uniformidade entre MDs). A maior dificuldade é conseguir um número suficiente de participantes para isso, a não ser que

se utilize estudantes. Nesse caso, a contrapartida pode ser abrir mão de uma aplicabilidade mais geral para os resultados a serem obtidos (ameaça à validade externa). Também seria interessante um estudo que procurasse avaliar comparativamente UCs e ICs diretamente. Três questões a serem investigadas em tal estudo seriam, por exemplo: 1) qual das duas técnicas (UCs ou ICs) produz modelos mais uniformes? 2) qual das duas técnicas produz modelos que demandam menos revisões durante o processo de validação perante os *stakeholders*? 3) qual das duas técnicas produz modelos menos sujeitos a modificações ao longo do desenvolvimento do sistema, para se corrigir inconsistências, ambigüidades e omissões?

7. Conclusão

Este capítulo conclui o trabalho de tese. A seção 7.1 apresenta uma visão geral (sumário) dos principais resultados obtidos; a seção 7.2 lista as contribuições do trabalho e discute a principal contrapartida exigida; a seção 7.3 apresenta as limitações da solução proposta; por fim, a seção 7.4 apresenta áreas e questões para pesquisas ou projetos futuros, vislumbradas durante a elaboração da tese.

7.1 Visão Geral do Trabalho

Esta tese fez uma revisão da modelagem com casos de uso (MUC), levantando e analisando problemas apontados na sua utilização. Entre eles, dificuldades enfrentadas pelo modelador na utilização do modelo de (classes de objetos) de domínio, em conjunto com o modelo de UCs, durante a fase de (levantamento e modelagem de) requisitos. Um dos efeitos dessas dificuldades se manifesta na discrepância entre MDs produzidos (independentemente) por diferentes modeladores, para um mesmo sistema.

A partir da análise das possíveis causas das dificuldades, foi proposta uma técnica de modelagem denominada **info cases** (ICs), voltada para sistemas de informação, que pode ser considerada uma especialização da técnica de UCs. Ela foi construída com base, principalmente, em duas providências:

1. Definição de um novo critério para o particionamento do sistema em UCs, que estabelece um nível específico de abstração para isso. Esse nível foi denominado de Nível Informacional de Objetivos (NIO);
2. Adoção de uma linguagem semiformal para a descrição dos fluxos de informação trocados entre o sistema e seus atores, durante a realização dos UCs do NIO.

A especialização dos UCs em ICs abriu a possibilidade de extrair do modelo de ICs (MIC), um modelo de domínio (MD) adequado[76], de forma semi-automática. Assim, pode-se ver o MIC como um modelo integrado de requisitos, onde elementos comportamentais (dos UCs) e estruturais (do MD) são capturados conjuntamente, utilizando um mesmo arcabouço conceitual. Foram enunciadas regras (e heurísticas associadas) para derivar um MD a partir dos ICs, e uma série de estudos experimentais

[76] Segundo critérios definidos na tese.

foram executados para explorar as hipóteses do trabalho e produzir evidências de sua validade.

Além de análises teóricas, que corroboraram as hipóteses enunciadas na tese, estudos experimentais produziram evidências favoráveis às mesmas; entre elas a de que a MIC induz uma maior uniformidade entre os MDs produzidos por diferentes modeladores, para um mesmo sistema. Embora seja desejável a realização de novos estudos experimentais para reforçar as evidências obtidas, consideramos que os já concluídos produzem um corpo de evidências significativo, que credencia a MIC como uma técnica promissora o suficiente para motivar outros projetos, visando explorar o seu uso e propiciar sua evolução e maturidade.

7.2 Principais Contribuições e Contrapartida

A principal contribuição deste trabalho de tese foi, a nosso ver, a construção de uma técnica para modelagem de requisitos de sistemas de informação, com potencial para resolver alguns problemas existentes na técnica tradicional e muito popular de casos de uso (UCs), potencial esse, argumentado em bases teóricas e evidenciado em uma série de estudos experimentais realizados.

Sendo uma especialização dos UCs, a nova técnica mantém vários pontos fortes dos mesmos, ao mesmo tempo em que inova propondo um critério mais objetivo para a escolha dos casos de uso de um sistema. Essa providência, além de resolver um problema reconhecido na literatura, ou seja, a excessiva subjetividade e ambigüidade do critério tradicional baseado no "valor do caso de uso para algum *stakeholder*", possibilita um tratamento harmonioso entre o modelo de casos de uso e o modelo de domínio do sistema, onde este último pode ser visto como uma visão extraída do primeiro. Além disso, a nova técnica permitiu a definição de um conjunto de regras e heurísticas capaz de fazer essa extração de forma semi-automática. Assim, evita-se o problema da consistência entre modelos e promove-se a uniformização dos MDs obtidos por diferentes modeladores, para um mesmo sistema.

Além dessa contribuição geral, em termos mais específicos e pontuais, podemos também apontar outras contribuições acessórias:

- Levantamento e análise comparativa das várias abordagens da MUC (seção 2.3);
- Identificação e análise dos problemas e vantagens da MUC (seção 2.4);

- Levantamento e consolidação das orientações dadas na literatura quanto à utilização conjunta do MUC e do MD na fase de requisitos (seção 3.3);
- Identificação de questões de pesquisa relevantes para o aperfeiçoamento da utilização conjunta dos dois modelos (seção 3.4);
- Identificação e análise das prováveis causas dos problemas apontados na utilização conjunta dos dois modelos (seção 4.2);

Uma parte significativa do conteúdo da tese já foi publicada em eventos científicos específicos da área (FORTUNA, BORGES, 2005) (FORTUNA *et al.*, 2007) (FORTUNA *et al.*, 2008a)[77], obtendo comentários e sugestões dos revisores, que contribuíram para o aperfeiçoamento dos resultados.

Em princípio, os benefícios advindos da nova técnica cobram uma contrapartida em termos de um maior treinamento para a sua utilização, e um maior esforço por parte do modelador para especificar, com rigor, os fluxos de informação trocados entre o sistema e seus atores. Entretanto, duas considerações podem ser feitas com relação a essa contrapartida. A primeira é que os estudos experimentais indicaram que ela acontece em um nível aceitável. A segunda é que, mediante uma análise mais profunda e abrangente, parece ser possível justificar essa contrapartida, ou mesmo passar a vê-la como geradora de ganho em vez de custo. Essa análise levaria em conta os benefícios da consistência entre o MIC e o MD, proporcionada pela nova técnica, e que afetam favoravelmente todo o ciclo de vida de um sistema. Outro ponto a ser considerado nessa análise seria o ganho advindo da exploração plena do potencial de automatização da técnica, através de uma ferramenta computacional para apoiar a sua aplicação.

7.3 Limitações

A nosso ver, a limitação mais significativa da técnica de ICs é sua aplicabilidade plena estar restrita a sistemas de informação (seção 5.5.2). Este foi o preço pago pelas vantagens específicas que a MIC trás na modelagem desse tipo de sistema. Outra limitação (compartilhada com os UCs) é a foco apenas em requisitos funcionais, deixando a importante categoria de requisitos não-funcionais sem um tratamento específico.

[77] Com seu relatório técnico associado: (FORTUNA *et al.*, 2008b).

7.4 Trabalhos Futuros

Além dos resultados relatados nesta monografia de tese, a pesquisa realizada evidenciou diversas oportunidades para novas pesquisas, em geral para explorar a relação, ou as possibilidades de interação, da MIC com outras abordagens de Engenharia de Requisitos, ou com técnicas empregadas em outras etapas do ciclo de desenvolvimento de um sistema. A seguir, apresentamos e comentamos algumas dessas possibilidades.

1. A MIC e a modelagem de processos de negócio

Normalmente se considera que a modelagem de um sistema deve ser precedida pela modelagem dos processos de negócio de uma organização, aos quais o sistema deve estar alinhado e apoiar. A MIC procura manter uma ponte com os requisitos de negócio, adotando a visão de que cada IC representa um "objetivo de negócio" de algum *stakeholder*. Além disso, no âmbito da MIC, um novo conceito, denominado *objetivo organizacional*, está sendo investigado. Esse conceito traduz uma visão mais precisa do que seria um objetivo de negócio: seqüências admissíveis e mais prováveis de ICs desempenhados no dia-a-dia do negócio. Os objetivos organizacionais representam abstrações de mais alto nível e com forte significado no domínio de aplicação do sistema. Como tal, contribuem para a escalabilidade da MIR. Este conceito foi preliminarmente testado no Estudo 3 (EC-3, seção 6.4)[78], quando percebemos outras vantagens de sua utilização: a) é facilmente percebido e validado pelos *stakeholders*, o que contribui para a validação dos ICs; b) reforça a verificação da consistência do MIC; c) estabelece um critério centrado nos atores para organizar e priorizar a implementação e os testes do sistema; e d) facilita a definição dos testes de integração e de aceitação do sistema. O Apêndice 7 inclui, entre outras coisas, uma avaliação feita por participantes do EC-3, sobre a utilização que eles fizeram desse conceito.

Uma iniciativa concreta para integrar os resultados da pesquisa desenvolvida nesta tese, em um método para o desenvolvimento de sistemas alinhados com os processos de negócios de uma organização, já está em andamento (DE LA VARA *et al.*, 2009a) (DE LA VARA *et al.*, 2009b).

[78] Embora não tenha sido relatado na seção que apresentou o estudo (6.4), por se tratar de uma investigação preliminar ("prova de conceito"), não enquadrada nos objetivos principais do estudo. No entanto, a opinião de alguns participantes que utilizaram o conceito pode ser vista no Apêndice 7.

2. A MIC e a Modelagem Orientada a Objetivos (MOObj)

A seção 5.5.4 discutiu a relação entre a MOObj e a MIC, e indicou um caminho para uma possível integração entre elas. Essa integração é, em princípio, desejável do ponto de vista da MIC, pela capacidade da MOObj de analisar objetivos, incluindo objetivos não-funcionais (*soft goals*). Portanto, uma sugestão de trabalho futuro é investigar como essa integração poderia ser feita e determinar as propriedades do modelo combinado resultante. Essa investigação deveria também levar em conta o conceito de objetivo organizacional mencionado no item 1 acima.

3. A MIC e o projeto da interface H-C

Os estudos de caso realizados mostraram indícios do potencial da MIC para ajudar no projeto da interface H-C de um sistema. A descrição rigorosa da interface informacional dos ICs abre uma série de possibilidades para a derivação de elementos do projeto da interface H-C. Por exemplo, os ICs são naturalmente mapeados para opções do menu principal do sistema. Identificadores (seção 5.3) presentes nos fluxos de entrada dos ICs são normalmente mapeados em *combo boxes* para seleção de um objeto. A composição dos fluxos de informações, e a ordem em que os itens de informação aparecem neles, orientam o projetista da interface H-C na disposição de informações nas janelas ou telas do sistema. A agregação de itens em pacotes (seção 5.3) pode indicar a oportunidade de se agregar os elementos de uma janela em painéis. Elementos de retroalimentação (*feedback*) ao usuário podem ser derivados a partir da interface informacional dos ICs. Elementos do diálogo H-C são mais adequadamente subdivididos entre as diversas janelas do sistema com a ajuda dos objetivos organizacionais (item 1 acima), pois tais objetivos capturam as seqüências de ICs que ocorrerão mais freqüentemente quando o sistema estiver em uso. Essas e outras possibilidades deveriam ser investigadas em detalhe, à luz dos princípios e melhores práticas do projeto de interface H-C.

4. A MIC e *Model-driven Architecture* (MDA)

Segundo BERRISFORD (2004), duas questões (relacionadas a requisitos) precisam ser respondidas para se construir um modelo que possa ser transformado em um sistema de processamento de dados: (1) que unidades de trabalho os atores do sistema invocam ou requerem; e (2) que dados precisam ser persistidos em uma estrutura de dados coerente, para que aquelas unidades de trabalho possam ser

realizadas. A MIC tem respostas para ambas: os ICs e o MD deles derivado, respectivamente. Portanto, parece ser promissor investigar as relações entre a MIC e o que está sendo desenvolvido no contexto de MDA/MDD.

5. MUC e MIC: Sobreposição na especificação de comportamento

A MUC foca, principalmente, a especificação do comportamento do sistema e seus atores. São vários os tipos de comportamento incluídos nos formatos mais completos de descrição dos UCs: ativação dos UCs pelos atores, entrada de dados pelos atores, produção de saídas pelo sistema, solicitação de intervenção aos atores, fornecimento de informações sobre o estado do sistema aos atores, retroalimentação para as intervenções dos atores, manipulação interna (armazenamento e recuperação de informações) da memória persistente, início e término de procedimentos pelo sistema, e tomada ou confirmação de decisões pelos atores.

A MIC acrescentou um significado especial para os UCs (decorrente da adoção do NIO) e a especificação semiformal dos fluxos de informação trocados entre o sistema e seus atores. Essa especificação dos fluxos, para UCs do NIO, traduz não apenas especificação de dados e suas estruturas, mas também especificação dos três primeiros tipos de comportamento citados acima, que coletivamente designamos pelo termo "*comportamento informacional*". Assim, ICs que incorporem uma descrição completa de comportamento, no estilo tradicional dos UCs, podem exibir certa redundância na especificação do comportamento informacional. Além disso, a experiência com a MIC tem evidenciado que boa parte do "comportamento não-informacional" pode ser derivada do comportamento informacional e de regras de "melhores práticas" no projeto da interface H-C. Portanto, uma linha de pesquisa poderia aprofundar essas questões, visando explorar o seu potencial de simplificação das descrições dos ICs.

6. Apoio computacional à MIC

Por fim, é mais do que necessário especificar e desenvolver uma ferramenta computacional para apoiar a aplicação da técnica proposta nesta tese. Só assim os benefícios por ela prometidos serão plenamente alcançados. Entre outras coisas, essa ferramenta deverá prover mecanismos adequados para promover o trabalho cooperativo entre modeladores e *stakeholders*, durante a elicitação e a modelagem dos ICs.

Referências

ALENCAR, F. M. R., PEDROSA, F., CASTRO, J. F. B *et al.*, 2003, "New Mechanisms for the Integration of Organizational Requirements and Object Oriented Modeling", In: *VI Workshop on Requirements Engineering* (WER'03), pp. 109-123, Piracicaba, São Paulo, November.

ANDERSON, D. J. *Are Use Cases the death of good UI Design?*, 1999a. Disponível em: <http://www.uidesign.net/1999/imho/feb_imho.html>. Acesso em: Dez. de 2008.

ANDERSON, D. J. *Use Cases still considered Dangerous!*, 1999b. Disponível em: <http://www.uidesign.net/1999/imho/oct_imho.html>. Acesso em: Dez. de 2008.

ARAÚJO, M. A. P., BARROS, M. O., TRAVASSOS, G. H., *et al*., 2006, "Métodos Estatísticos Aplicados em Engenharia de Software Experimental (Tutorial)", In: *XX Simpósio Brasileiro de Engenharia de Software* (SBES'06), p. 325-325, Florianópolis, SC, Brasil, Outubro.

BECK, K., CUNNINGHAM, W., 1989, "A laboratory for teaching object-oriented thinking". In: *OOPSLA-89 ACM Conf. on Object-Oriented Programming Sysstems Languages and Applications*, ACM Press, pp. 1-6, New Orleans, Louisiana, USA, October.

BERRISFORD, G., 2004, Why IT veterans are sceptical about MDA. *2nd European Workshop on Model Driven Architecture* (MDA), Canterbury, England, September.

BIDDLE, R., NOBLE, J., 2002, "From Essential Use Cases to Objects", In: *1st. Intl. Conference on Usage-Centered Design* (*forUSE'02*), Larry Constantine (ed.).

BITTNER, K., SPENCE, I., 2002, *Use Case Modeling*. 1 ed., New York, Addison-Wesley Professional.

BREITMAN, K. K., LEITE, J. C. P., 2003, "Ontology as a Requirements Engineering Product", In: *11th IEEE Intl. Requirements Engineering Conference* (IREC), pp. 309-319, Monterey Bay, California, September.

CITESEER, 2003, *Estimated Impact of Publication Venues in Computer Science*. CiteSeer. Disponível em: <http://citeseer.ist.psu.edu/impact.html>. Acesso em: Dezembro, 2008.

COAD, P., YOURDON, E., 1990, *Object Oriented Analysis* (2nd ed.), Englewood Cliffs, NJ, Prentice Hall.

COCKBURN, A., 2005, *Escrevendo Casos de Uso Eficazes*. 1 ed., Porto Alegre, Bookman Editora.

CONSTANTINE, L. L., 2000, "What Do Users Want? Engineering Usability into Software", *Constantine & Lockwood Ltd*., June.

CONSTANTINE, L. L., LOCKWOOD, L. A. D., 2001, "Structure and Style in Use Cases for User Interface Design". In: Mark Van Harmelen (ed.), *Object Modeling and User Interface Design: Designing Interactive Systems*, pp. 245-279, Boston, MA, Addison-Wesley Longman Publishing Co.

CONSTANTINE, L. L., LOCKWOOD, L. A. D., 1999, *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*. Boston, Addison-Wesley.

HE, L., CARVER, J. C., VAUGHN, R. B., 2008, "Using Inspections to Teach Requirements Validation", *CrossTalk - The Journal of Defense Software Engineering*, January, pp. 11-15.

DAVIS, L., DAWE, M., 2001, "Collaborative Design with Use Case Scenarios", In: *1st. ACM/IEEE-CS Joint Conf. on Digital Libraries* (JCDL'01), pp. 146-147, Roanoke, VA, USA, June.

DE LA VARA, J. L, FORTUNA, M. H, WERNER, C. M. L. *et al*., 2009a, "Modelado de Requisitos de Datos para Sistemas de Información basados em Procesos de Negocio", In: *XII Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes de Software* (*IDEAS'09*), Medellin, Colombia, Abril (a ser publicado).

DE LA VARA, J. L, FORTUNA, M. H, WERNER, C. M. L. *et al*., 2009b, "A Requirements Engineering Approach for Data Modelling of Process-Aware Information Systems", In: *12th International Conference on Business Information Systems* (*BIS'09*), Poznan, Poland, April (to be published).

DE LANDTSHEER, R., LETIER, E., van LAMSWEERDE, A., 2003, "Deriving Tabular Event-Based Specifications from Goal-Oriented Requirements Models", *Requirements Engineering*, v.9, n. 2, pp. 104-120.

DIJKSTRA, E. W., 1968, "Go To Statement Considered Harmful", *Communications of the ACM*, v. 11, n. 3 (March), pp. 147-148.

DOBING, B., PARSONS, J., 2000, "Understanding the Role of Use Cases in UML: A Review and Research Agenda", *Journal of Database Management* (Idea Group Publ.), v. 11, n. 4, pp. 28-36.

FORTUNA, M. H., BORGES, M. R. S., 2005, "Modelagem Informacional de Requisitos", In: *VIII Workshop on Requirements Engineering* (WER'2005), pp. 269-280, Porto, Portugal, June.

FORTUNA, M. H., WERNER, C. M. L., BORGES, M. R. S., 2007, "Um Modelo Integrado de Requisitos com Casos de Uso", In: *X Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes de Software* (IDEAS'07), Isla de Margarita, Venezuela, pp. 313-326, Maio.

FORTUNA, M. H., WERNER, C. M. L., BORGES, M. R. S., 2008a, "Info Cases: Integrating Use Cases and Domain Models", In: *XVI IEEE Intl. Conference on Requirements Engineering* (*RE'08*), pp. 81-84, Barcelona, Spain, September.

FORTUNA, M. H., WERNER, C. M. L., BORGES, M. R. S., 2008b, *Info Cases: Integrating Use Cases and Domain Models*, RT ES-719/08, COPPE/UFRJ.

GLINZ, M., 2000, "A Lightweight Approach to Consistency of Scenarios and Class Models", In: *4th IEEE Intl. Conference on Requirements Engineering*, pp. 49-58, Schaumburg, IL, USA, June.

GLINZ, M., BERNER, S., JOOS, S., 2002, "Object-Oriented Modeling with ADORA", *Information Systems* v. 27, n. 6, pp. 425-444.

GLINZ, M., 2008, *ADORA Project*. Disponível em: <http://www.ifi.uzh.ch/rerg/research/projects/adora/>. Acesso em: Novembro.

GONÇALVES, V. P., 2008a, "*Um Estudo Experimental em Modelagem de Requisitos de Sistemas de Informação*", Monografia de Conclusão do Curso de Ciência da Computação, Universidade Federal de Juiz de Fora (UFJF), Juiz de Fora, Brasil, Fevereiro.

GONÇALVES, V. P., 2008b, "*Empacotamento do Estudo Experimental sobre Modelagem de Requisitos de Sistemas de Informação*", Suplemento da Monografia de Conclusão do Curso de Ciência da Computação, Universidade Federal de Juiz de Fora (UFJF), Juiz de Fora, Brasil, Fevereiro.

GORN, S., 1961, "Specification Languages for Mechanical Languages and Their Processors - A Baker's Dozen", *Commmunication of the ACM*, v.4, n.12, pp. 532-542.

GRUBER, T. R., 1993, "A Translation Approach to Portable Ontology Specifications". *Knowledge Acquisition*, v. 5, n. 2, pp. 199-220.

HAY, D. C., 2003, *Requirements Analysis - From Business Views to Architecture*. 1st ed. New Jersey, Prentice Hall PTR.

HENDERSON-SELLERS, B., 1996, *Object-Oriented Metrics – Measures of Complexity*, New Jersey, Prentice Hall PTR.

IEEE IREC, 2007, *Intl. Conferences on Requirements Engineering*. Disponível em: <http://www.requirements-engineering.org/>. Acesso em: Dezembro de 2007.

JACOBSON, I., 1987, "Object Oriented Development in an Industrial Environment", In: *ACM SIGPLAN Intl. Conf. on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'87)*, pp. 183-191, Orlando, Florida, USA, October.

JACOBSON, I., CHRISTERSON, M., ÖVERGAARD, G., 1992, *Object-Oriented Software Engineering - A User Case Driven Approach*. 1st ed. New York, Addison-Wesley.

JACOBSON, I., ERICSSON, M., JACOBSON, A., 1994, *The Object Advantage - Business Process Reengineering with Object Technology*. 1st ed. Wokingham, England, Addison-Wesley (ACM Press).

JACOBSON, I., BOOCH, G., RUMBAUGH, J., 1999, *The Unified Software Development. Process*. Addison-Wesley Object Techonolgy Series.

JACOBSON, I., 2004, "Use cases - Yesterday, today, and tomorrow", *Software and Systems Modeling Journal*, v.3, pp. 210-220.

JACOBSON, I., NG, P-W., 2004, *Aspect-Oriented Software Development with Use Cases*. 1st ed., Addison Wesley Professional.

JUDE, 2008, Change Vision, Inc., Tokyo, Japan. Disponível em: <http://www.change-vision.com/>. Acesso em: Novembro, 2008.

JURISTO, N., MORENO, A. M., LÓPEZ, M., 2000, "How to Use Linguistic Instruments for Object-Oriented Analysis", *IEEE Software*, May/June, pp. 80-89.

JURISTO, N., MORENO, A., 2006, *Basics of Software Engineering Experimentation*. Universidad Politécnica de Madrid, Spain.

KÄRKKÄINEN, T., NURMINEN, M., SUOMINEN, P. *et al*., 2008, UCOT: Semiautomatic Generation of Conceptual Models from Use Case Descriptions, In: *The IASTED Intl. Conf. on Software Engineering (SE 2008)*, Innsbrück, Austria, February.

KNUTH, D., 1964, "Backus Normal Form vs. Backus Naur Form", *Communications of the ACM*, v.7, n.12, pp. 735-736.

KÖSTERS, G., SIX, H-W., WINTER, M., 2001, "Coupling Use Cases and Class Models as a Means for Validation and Verification of Requirements Specifications", *Requirements Engineering Journal*, v. 6, pp. 3-17.

KULAK, D., GUINEY, E., 2003, *Use Cases: Requirements in Context*. 2nd ed., Addison Wesley Professional, 2003.

LAMSWEERDE, 1992, A. The Meeting Scheduler System - Problem Statement. Université Catholique de Louvain.

LAMSWEERDE, A., 2001, "Goal-Oriented Requirements Engineering: A Guided Tour", In: *5th IEEE Intl. Symposium on Requirements Engineering (RE'01)*, pp. 249-262, Toronto, Canada, August.

LARMAN. C., 2004, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. 3rd ed., Prentice Hall.

LAUESEN, 2002, S. *Software Requirements - Styles and Techniques*. Great Britain, Addison-Wesley.

LEITE, J. C. S. P., OLIVEIRA, A. P., 1995, "A Client Oriented Requirements Baseline", In: *2nd. Intl. Symposium on Requirements Engineering*, pp. 108-115, York, England, March.

LEITE, J. C. S. P. *et al.*, 1997, "Enhancing a Requirements Baseline with Scenarios", In: *3rd. IEEE Intl. Symposium on Requirements Engineering*, pp. 44-53, Annapolis, USA, January.

LEITE, J. C. S. P., HADAD, G. D. S., DORN, J. H, *et al.*, 2000, "A Scenario Construction Process", *Requirements Engineering Journal*, v. 5, pp. 38-61.

LIANG, Y., 2003, "From use cases to classes: a way of building object model with UML", *Information and Software Technology*, v. 45, pp. 83-93.

MAC KNIGHT, D., 2004, *Elicitação de Requisitos de Software a partir do Modelo de Negócio*. Dissertação de Mestrado, IM-NCE/UFRJ, Rio de Janeiro.

MAFRA, S. N., TRAVASSOS, G. H., 2006, "Leitura Baseada em Perspectiva: A Visão do Projetista Orientada a Objetos", In: *V Simpósio Brasileiro de Qualidade de Software (SBQS'2006)*, pp. 144-158, Vila Velha (ES), Brasil, Maio.

MAFFEO, B., 1992, *Engenharia de Software e Especificação de Sistemas*. Rio de Janeiro, Editora Campus.

MARTÍNEZ, A., CASTRO, J., PASTOR, O. *et al.*, 2003, "Closing the gap between Organizational Modeling and Information System Modeling". In: *VI Workshop on Requirements Engineering* (WER'03), pp. 93-108, Piracicaba, São Paulo, November.

MCMENAMIM, S. M., PALMER, J. F., 1991, *Análise Essencial de Sistemas*. 1st ed. São Paulo, McGraw-Hill.

MEYER, B., 1997, *Object-Oriented Software Construction*. 2nd ed., New Jersey, Prentice Hall PTR.

MYLOPOULOS, J., CHUNG, L., YU, E., 1999, "From Object-Oriented To Goal-Oriented Requirements Analysis". *Communications of the ACM*, v. 42, n. 1 (January), pp. 31-37.

NEILL, C. J., LAPLANTE, P. A., 2003, "Requirements Engineering: The State of the Practice". *IEEE Software*, v. 20, n. 6 (November/December), pp. 40-45.

O'REILLY, 2008, *SAFARI Books Online*. O'Reilly Media, Inc. Disponível em: <http://safari.oreilly.com/home>. Acesso em: Novembro, 2008.

OMG, 2008, *UML 2.0 Specification*. Object Management Group. Disponível em: <http://www.uml.org/>. Acesso em: Dezembro, 2008.

PENDER, T., 2003, *UML Bible*. 1st ed., Indianapolis, Indiana, USA, Wiley Publishing, Inc..

PINHEIRO, F. A. C., 2004, Requirements Traceability. In: *Perspectives on Software Requirements*, Dordrecht, The Netherlands, Kluwer Academic Publishers, pp. 91-113.

QUALIS, 2008, *Qualis*. Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Brasil. Disponível em: <http://www.capes.gov.br/avaliacao/qualis>. Acesso em: Novembro, 2008.

ROBERTSON, S., ROBERTSON, J., 1999, *Mastering the Requirements Process*. 1st ed., London, Addison-Wesley.

ROBERTSON, S.. ROBERTSON, J., 2006, *Mastering the Requirements Process*. 2nd ed., New Jersey, Addison Wesley Professional.

SHOVAL, P., YAMPOLSKY, A., LAST, M., 2006, "Class Diagrams and Use Cases - Experimental Examination of the Preferred Order of Modeling", In: *11th Intl. Workshop on Exploring Modeling Methods in System Analysis and Design* (EMMSAD-06), Luxembourg, 5-6 June.

SODHI, J., SODHI, P., 2003, *Managing IT Systems Requirements*. 1st ed., Management Concepts.

SOMMERVILLE, I., SAWYER, P., 1997, *Requirements Engineering - A Good Practice Guide*. 1st ed., England, Jonh Wiley & Sons Ltd.

SPSS, 2008, *Software de Análises Estatísticas*. SPSS Brasil. Disponível em: <http://www.spss.com.br/>. Acesso em: Novembro, 2008.

SUBRAMANIAM, K. LIU, D., FAR, B. H. *et al.*, 2004, "UCDA: Use Case Driven Development Assistant Tool for Class Model Generation", In: *16th Intl. Conf. on Software Engineering & Knowledge Engineering* (SEKE'2004), pp. 324-329, Banff, Alberta, Canada, June.

SUTCLIFFE, A., 2003, "Scenario-Based Requirements Engineering", In: *11th IEEE Intl. Requirements Engineering Conference* (IREC), Monterey Bay, California, September.

SVETINOVIC, D., 2005, "Improving the Consistency of the Conceptual Models Through the Activity-Purpose Analysis". In: *13th IEEE Intl. Conference on Requirements Engineering* (RE'05) *Doctoral Symposium*, Paris, France, September.

SVETINOVIC, D., BERRY, D. M., GODFREY, M., 2005, "Concept Identification in Object-Oriented Domain Analysis: Why Some Students Just Don't Get It", In: *13th IEEE Intl. Conference on Requirements Engineering* (RE'05), pp. 189-198, Paris, France, September.

SVETINOVIC, D., BERRY, D. M., GODFREY, M., 2006, "Increasing Quality of Conceptual Models: Is Object-Oriented Analysis That Simple?", In: *The Role of Abstraction in Software Engineering Workshop / 28th Intl. Conference on Software Engineering* (ROA/ICSE), Shangai, China, September.

SVETINOVIC, D., 2006, "*Increasing the Semantic Similarity of Object-Oriented Domain Models by Performing Behavioral Analysis First*". PhD. Thesis, Universtiy of Waterloo, Ontario, Canada, September.

WÁLDEN, K., NERSON, J-M., 1995, *Seamless Object-Oriented Software Architecture - Analysis and Design of Reliable Systems*. 1st ed. New York, Prentice Hall.

WIEGERS, K. E., 2003, *Software Requirements*. 2nd ed., Redmond, Washington, Microsoft Press.

WING, J. M., 1988, "A Study of 12 Specifications of the Library Problem", *IEEE Software*, v. 5, n. 4, pp. 66-76.

WIRFS-BROCK, R., WILKERSON, B., WIENER, L., 1990, *Designing Object Oriented Software*. Englewood Cliffs, NJ. Prentice Hall.

WOHLIN, C., RUNESON, P., HÖST, M., *et al*., 2000, *Experimentation in Software Engineering - An Introduction*, London, UK, Kluwer Academic Publishers .

WUsCaM, 2006, *Intl. Workshop on Open Issues in Industrial Use Case Modeling*. Disponível em: <http://www.ie.inf.uc3m.es/uml2004-ws6/>. Acesso em: Julho, 2006.

YOURDON, E., 1990, *Análise Estruturada Moderna*. Editora Campus.

YU, E., 1997, "Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering". In: *3rd IEEE Int. Symp. on Requirements Engineering* (RE'97), Washington D.C., USA, pp. 226-235, January.

Apêndice 1: Sintaxe da Ling. da Int. Informacional de ICs

A linguagem para a especificação dos fluxos de informação, que constituem a interface informacional dos ICs, tem a seguinte sintaxe formal em BNF – *Backus Naur Form* (GORN, 1960) (KNUTH, 1964):

<Fluxo de dados> ::= <Fluxo de entrada> | <Fluxo de saída>

<Fluxo de entrada> ::= ➔ <Nome de item elementar> | ➔ <Nome de fluxo> = <Expressão de dados>

<Fluxo de saída> ::= ➔ <Nome de item elementar> | ➔ <Nome de fluxo> = <Expressão de dados>

<Nome de item elementar> ::= <Nome comum> | <Nome de identificador>

<Nome de fluxo> ::= <Nome comum>

<Pacote de dados> ::= 🗐 <Nome de pacote> = <Expressão de dados>

<Expressão de dados> ::= <Nome de item elementar> | <Nome de pacote> | <Expressão de dados> + <Expressão de dados> | <Expressão de dados> '|' <Expressão de dados> | <Limite inferior de repetição> { <Expressão de dados> } <Limite superior de repetição> | (<Expressão de dados>) | [<Expressão de dados>]

<Nome de pacote> ::= <Nome comum>

<Nome comum> ::= <Caracter> | <Caracter> <Nome comum>

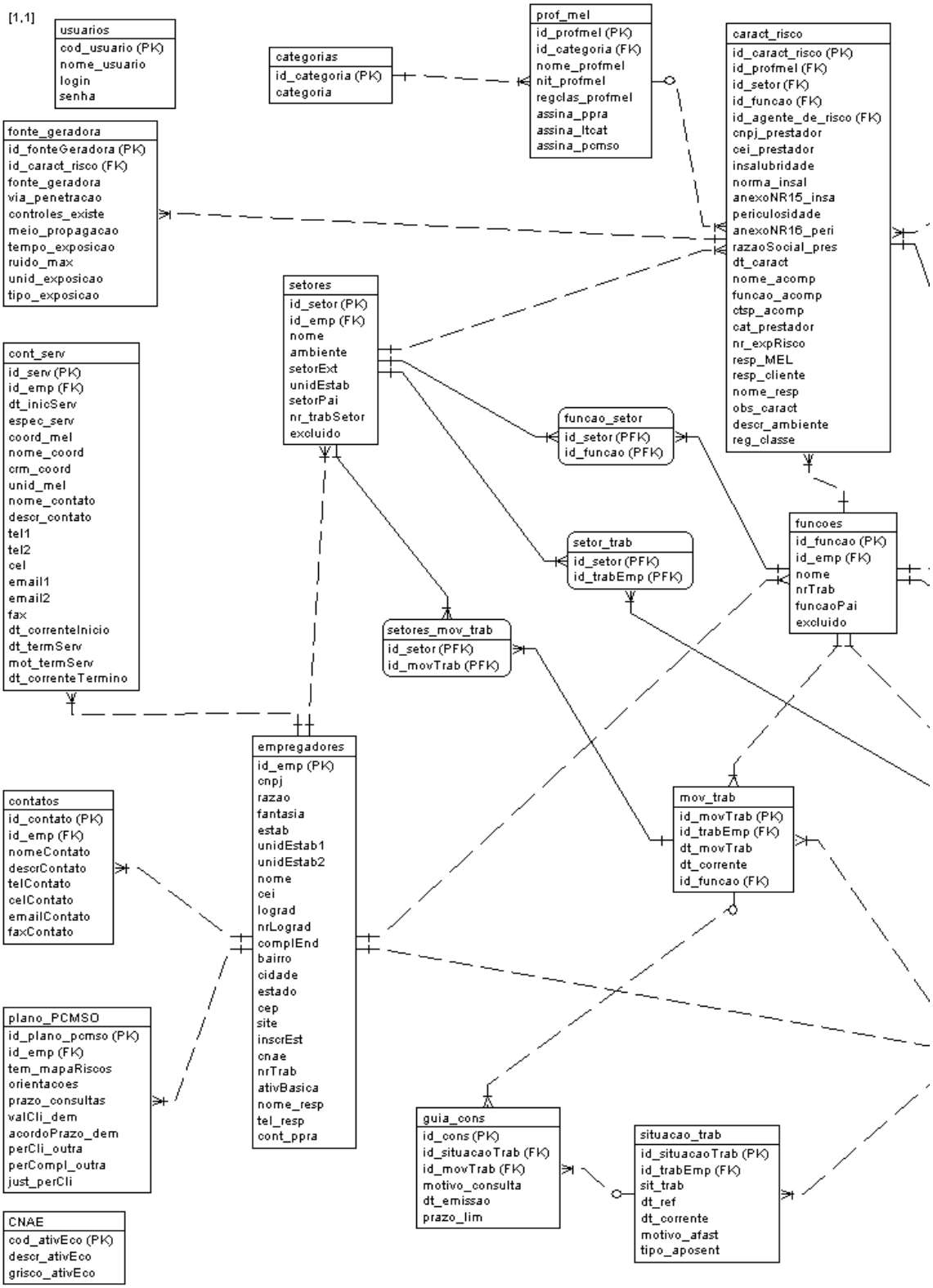
<Nome de identificador> ::= id_ <Nome comum>

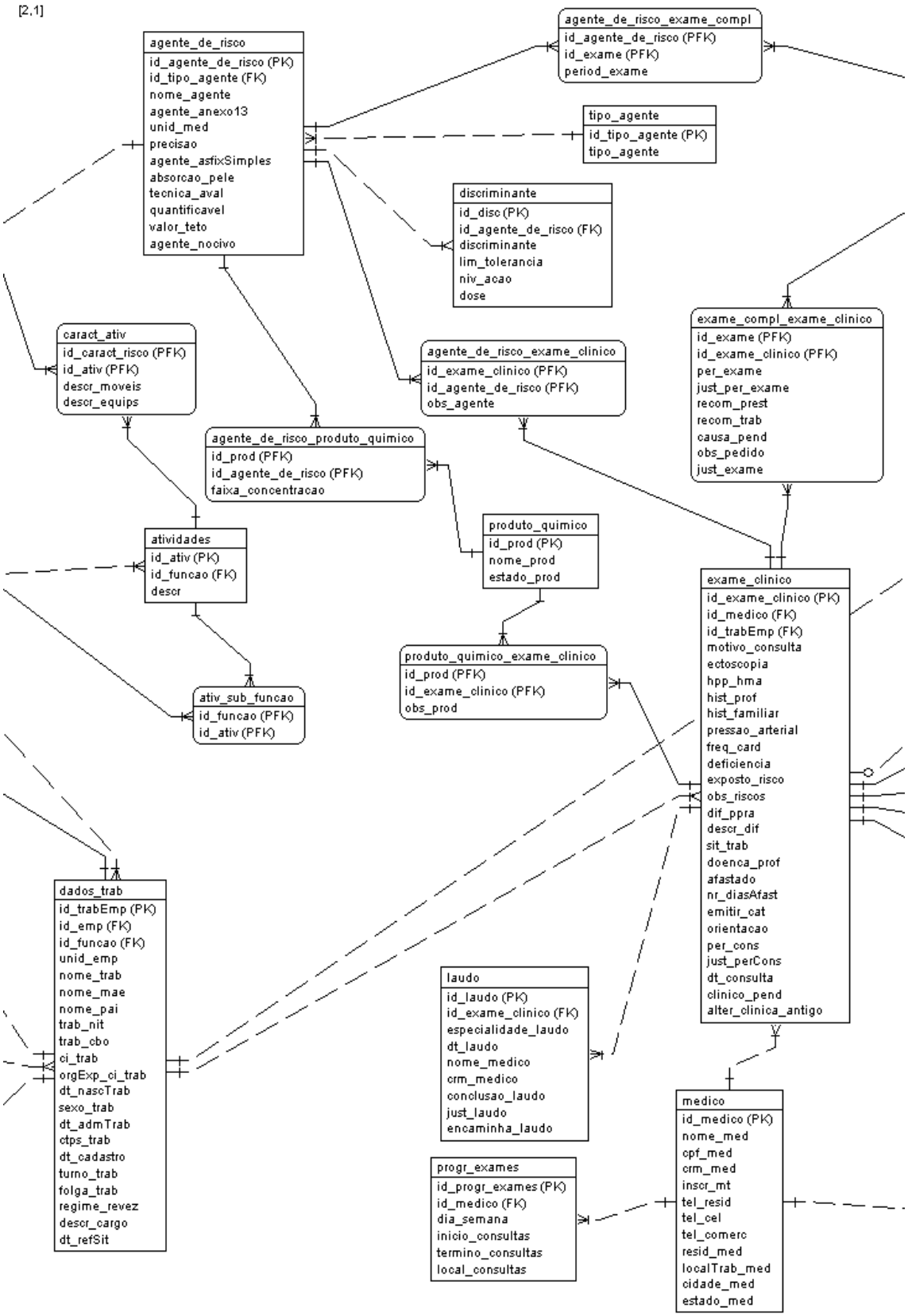
<Caracter> ::= <Letra> | <Dígito> | _ |

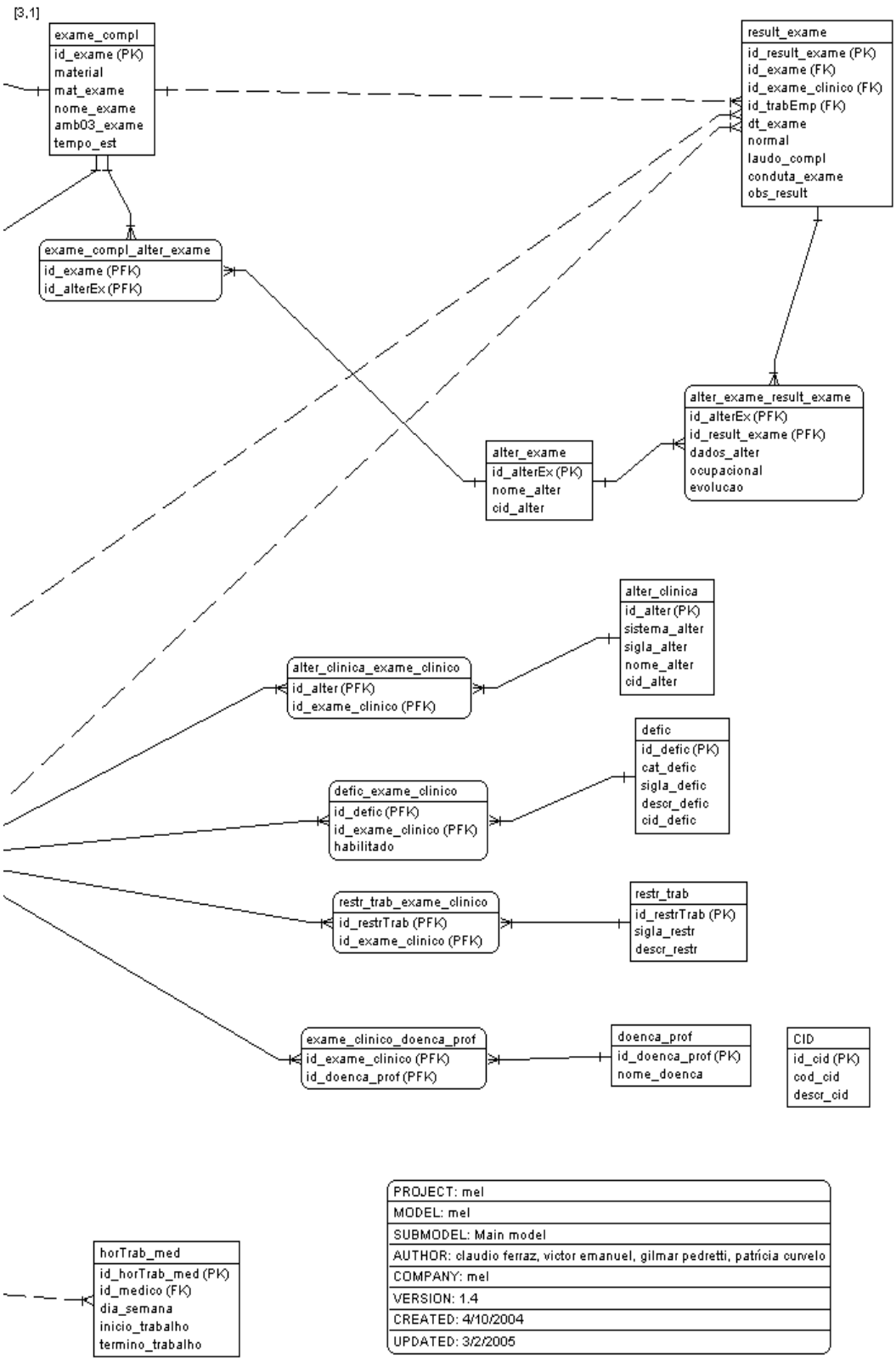
<Letra> ::= A | a | B | b | C | c | ...

<Dígito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Apêndice 2: Tabelas do Sistema SIMEL (EC-1)




[2,1]
agente_de_risco
id_agente_de_risco (PK)
id_tipo_agente (FK)
nome_agente
agente_anexo13
unid_med
precisao
agente_asfixSimples
absorcao_pele
tecnica_aval
quantificavel
valor_teto
agente_nocivo
agente_de_risco_exame_compl
id_agente_de_risco (PFK)
id_exame (PFK)
period_exame
tipo_agente
id_tipo_agente (PK)
tipo_agente
discriminante
id_disc (PK)
id_agente_de_risco (FK)
discriminante
lim_tolerancia
niv_acao
dose
caract_ativ
id_caract_risco (PFK)
id_ativ (PFK)
descr_moveis
descr_equips
agente_de_risco_exame_clinico
id_exame_clinico (PFK)
id_agente_de_risco (PFK)
obs_agente
exame_compl_exame_clinico
id_exame (PFK)
id_exame_clinico (PFK)
per_exame
just_per_exame
recom_prest
recom_trab
causa_pend
obs_pedido
just_exame
agente_de_risco_produto_quimico
id_prod (PFK)
id_agente_de_risco (PFK)
faixa_concentracao
produto_quimico
id_prod (PK)
nome_prod
estado_prod
atividades
id_ativ (PK)
id_funcao (FK)
descr
produto_quimico_exame_clinico
id_prod (PFK)
id_exame_clinico (PFK)
obs_prod
exame_clinico
id_exame_clinico (PK)
id_medico (FK)
id_trabEmp (FK)
motivo_consulta
ectoscopia
hpp_hma
hist_prof
hist_familiar
pressao_arterial
freq_card
deficiencia
exposto_risco
obs_riscos
dif_ppra
descr_dif
sit_trab
doenca_prof
afastado
nr_diasAfast
emitir_cat
orientacao
per_cons
just_perCons
dt_consulta
clinico_pend
alter_clinica_antigo
ativ_sub_funcao
id_funcao (PFK)
id_ativ (PFK)
dados_trab
id_trabEmp (PK)
id_emp (FK)
id_funcao (FK)
unid_emp
nome_trab
nome_mae
nome_pai
trab_nit
trab_cbo
ci_trab
orgExp_ci_trab
dt_nascTrab
sexo_trab
dt_admTrab
ctps_trab
dt_cadastro
turno_trab
folga_trab
regime_revez
descr_cargo
dt_refSit
laudo
id_laudo (PK)
id_exame_clinico (FK)
especialidade_laudo
dt_laudo
nome_medico
crm_medico
conclusao_laudo
just_laudo
encaminha_laudo
progr_exames
id_progr_exames (PK)
id_medico (FK)
dia_semana
inicio_consultas
termino_consultas
local_consultas
medico
id_medico (PK)
nome_med
cpf_med
crm_med
inscr_mt
tel_resid
tel_cel
tel_comerc
resid_med
localTrab_med
cidade_med
estado_med


[3,1]
exame_compl
id_exame (PK)
material
mat_exame
nome_exame
amb03_exame
tempo_est
result_exame
id_result_exame (PK)
id_exame (FK)
id_exame_clinico (FK)
id_trabEmp (FK)
dt_exame
normal
laudo_compl
conduta_exame
obs_result
exame_compl_alter_exame
id_exame (PFK)
id_alterEx (PFK)
alter_exame_result_exame
id_alterEx (PFK)
id_result_exame (PFK)
dados_alter
ocupacional
evolucao
alter_exame
id_alterEx (PK)
nome_alter
cid_alter
alter_clinica
id_alter (PK)
sistema_alter
sigla_alter
nome_alter
cid_alter
alter_clinica_exame_clinico
id_alter (PFK)
id_exame_clinico (PFK)
defic
id_defic (PK)
cat_defic
sigla_defic
descr_defic
cid_defic
defic_exame_clinico
id_defic (PFK)
id_exame_clinico (PFK)
habilitado
restr_trab_exame_clinico
id_restrTrab (PFK)
id_exame_clinico (PFK)
restr_trab
id_restrTrab (PK)
sigla_restr
descr_restr
exame_clinico_doenca_prof
id_exame_clinico (PFK)
id_doenca_prof (PFK)
doenca_prof
id_doenca_prof (PK)
nome_doenca
CID
id_cid (PK)
cod_cid
descr_cid
horTrab_med
id_horTrab_med (PK)
id_medico (FK)
dia_semana
inicio_trabalho
termino_trabalho
PROJECT: mel
MODEL: mel
SUBMODEL: Main model
AUTHOR: claudio ferraz, victor emanuel, gilmar pedretti, patrícia curvelo
COMPANY: mel
VERSION: 1.4
CREATED: 4/10/2004
UPDATED: 3/2/2005

Apêndice 3: Instrumentos do Estudo 2 (EXP-2)

Questionário de Caracterização dos Participantes

Questionário

Experiência prévia no domínio:

1) Você já teve experiência na modelagem de um sistema parecido com o sistema FGV (contas a pagar/receber)? Descreva (número de sistemas, porte, tempo dedicado).

_____
_____
_____

2) Mesmo não tendo modelado, como avalia o seu conhecimento/experiência anterior sobre este tipo de sistema?

_____
_____
_____

Experiência prévia na técnica (casos de uso):

1) Você tem experiência na modelagem de casos de uso (elaboração, leitura)? Essa experiência foi apenas acadêmica (estudo ou trabalho escolar) ou também profissional (sistema real)?

_____
_____
_____

2) Você tem experiência na modelagem informacional com casos de uso? (apenas grupo A) Descreva brevemente.

_____
_____
_____

Experiência profissional:

1) Descreva sua experiência profissional (empresa, função, tempo na função).

_____
_____
_____

Formação:

2) Qual é o seu nível de escolaridade? Descreva. Participou de treinamento sobre casos de uso? Com que carga horária?

_____
_____

Sumário do Sistema FGV

Deseja-se um sistema computadorizado para melhorar o seu controle financeiro da empresa. Entre outras coisas, a empresa deseja:

- Prever e acompanhar a entrada e saída de dinheiro decorrente do recebimento dos clientes e do pagamento de contas a credores da empresa. Ou seja, para cada dia do mês ela precisa saber o que deve pagar e o que tem a receber até aquele dia. O principal objetivo é prever eventuais dificuldades de caixa, para tomar providências (por exemplo, fazer um empréstimo) em tempo hábil.
- Controlar o saldo em caixa, bem como o saldo e a movimentação de suas contas bancárias.
- Controlar as despesas efetuadas (pagamentos) por credor e item de despesa. O objetivo é ter uma gestão permanente do quanto se gasta com cada credor e em cada item de despesa, buscando maximizar os benefícios obtidos com os recursos gastos, face as metas da empresa. A quitação pode ser feita de uma só vez ou dividida em parcelas, nas respectivas datas de vencimento acordadas.
- Controlar os recebimentos de clientes, por serviço prestado. A empresa tem um leque atual de serviços que disponibiliza a seus clientes, que pode mudar ao longo do tempo. A maioria dos clientes tem contrato com a empresa, estipulando os serviços contratados e o valor a ser pago pelo conjunto dos serviços. Mesmo clientes sem contrato podem usufruir os serviços da empresa, pagando o valor-padrão desses serviços, ou um valor especial decorrente de negociação caso a caso. A quitação pode ser feita de uma só vez ou dividido em parcelas, nas respectivas datas de vencimento acordadas. Os contratos têm renovação automática a cada 1 ano contado da data da contratação, podendo ser renegociados ou cancelados a qualquer instante, mediante solicitação prévia do cliente, com pelo menos 2 meses de antecedência. É preciso manter um histórico das renovações e renegociações de contratos.

A empresa deve poder consultar, a qualquer instante, para uma data ou período escolhido:

- Recebimentos realizados e a realizar, de todos os clientes ou de um específico, correspondente a um determinado serviço ou a todos eles. Os recebimentos atrasados (fora do prazo acordado) deverão estar destacados, para uma eventual cobrança ao cliente (daí a importância de manter informações de contato com o cliente: endereço, pessoa de contato, telefone, e-mail). Recebimentos fora do prazo estão sujeitos a multa e juros, conforme estipulado no contrato ou através de acordo.
- Pagamentos realizados e a realizar, para todos os credores ou para um específico, e associados a um determinado item de despesa ou a todos eles. Os pagamentos atrasados (fora do prazo acordado) deverão estar destacados, para que possam ser priorizados e/ou renegociados (daí a importância de manter informações de contato com o credor: endereço, pessoa de contato, telefone, e-mail). Deverão ser apresentadas informações que permitam ou facilitem o pagamento, tais como: nome do credor, forma de pagamento, valor a pagar, pessoa de contato, endereço do banco/agência ou do estabelecimento onde efetuar o pagamento, etc.
- Montante a receber e montante a pagar, bem como saldo inicial e final disponível em suas contas bancárias e no caixa, dia-a-dia, dentro do período escolhido.

Exemplos de ICs produzidos

Ator: **Funcionário da Empresa** IC1: **Cadastrar Conta**

1. O ator informa ao sistema sua necessidade de cadastrar uma conta
2. O sistema solicita as informações necessárias de uma conta
   ➔ conta = banco + conta + agencia
3. O sistema cadastra a conta e emite uma mensagem de sucesso
   ← id_conta
4. O caso de uso termina

Ator: **Funcionário da Empresa** IC2: **Cadastrar Credor**

1. O ator informa ao sistema sua necessidade de cadastrar um credor
2. O sistema solicita as informações necessárias de um credor
   ➔ credor = tipo_pessoa + (nome + CPF | razão_social + CNPJ) + endereco + contato + telefone + email
3. O sistema cadastra o credor e emite uma mensagem de sucesso
   ← id_credor
4. O caso de uso termina

Ator: **Funcionário da Empresa** IC3: **Cadastrar Cliente**

1. O ator informa ao sistema sua necessidade de cadastrar um cliente
2. O sistema solicita as informações necessárias de um cliente
   ➔ cliente = tipo_pessoa + (nome + CPF | razão_social + CNPJ) + endereco + contato + telefone + email
3. O sistema cadastra o cliente e emite uma mensagem de sucesso
   ← id_cliente
4. O caso de uso termina

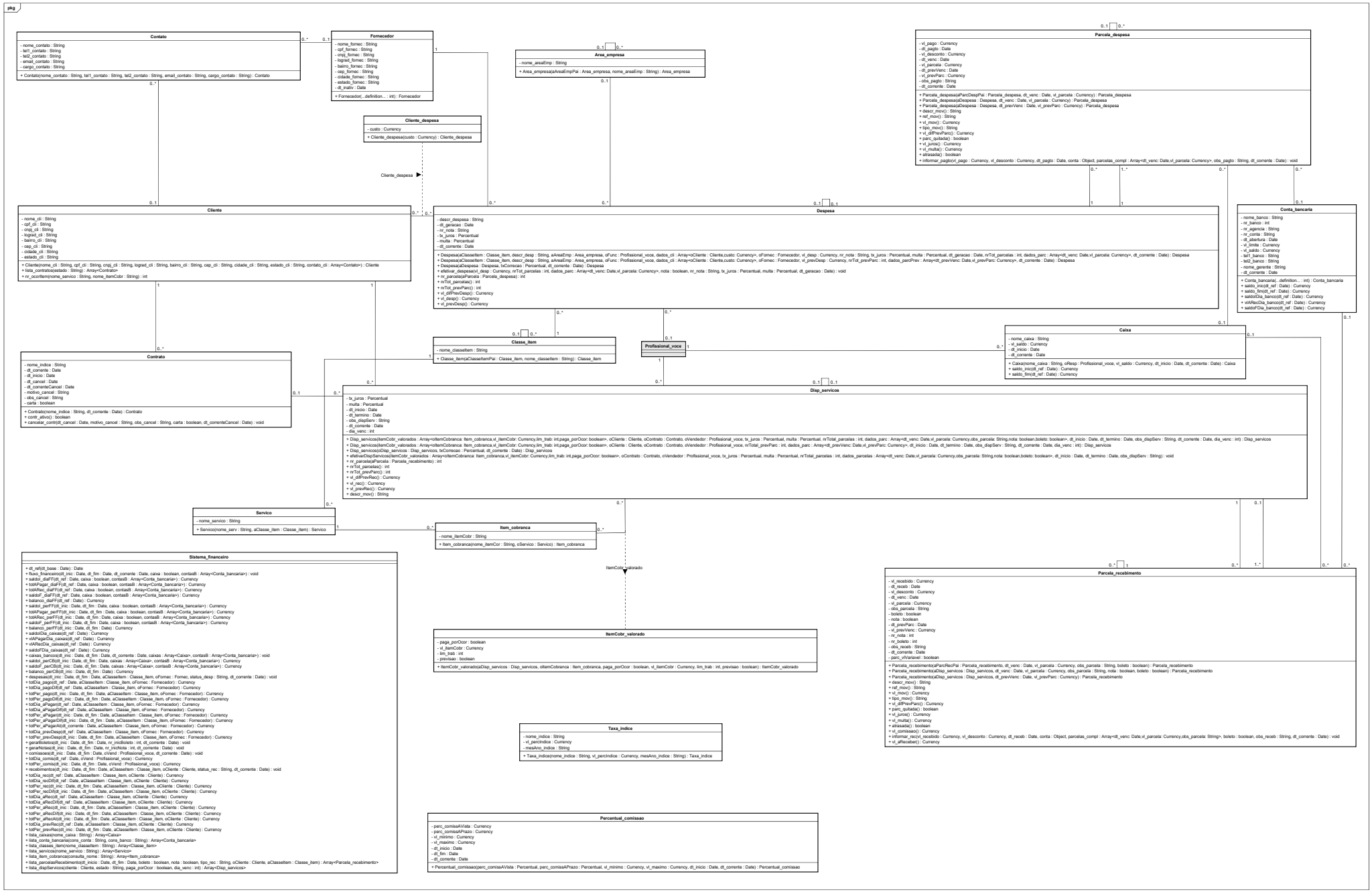
Ator: **Funcionário da Empresa** IC4: **Cadastrar serviço**

1. O ator informa ao sistema sua necessidade de cadastrar um serviço oferecido pela empresa
2. O sistema solicita as informações necessárias de um serviço
   ➔ servico = nome + descricao + preco_padrao
3. O sistema cadastra o servico e emite uma mensagem de sucesso
   ← id_servico
4. O caso de uso termina

Ator: **Funcionário da Empresa** IC5: **Cadastrar ítem de despesa**

1. O ator informa ao sistema sua necessidade de cadastrar um item de despesa
2. O sistema solicita as informações necessárias de um item de despesa
   ➔ item_despesa = nome + descrição
3. O sistema cadastra o item de despesa e emite uma mensagem de sucesso
   ← id_item_despesa
4. O caso de uso termina

Apêndice 4: MD do SisConf (EC-3)


pkg
Contato
- nome_contato : String
- tel1_contato : String
- tel2_contato : String
- email_contato : String
- cargo_contato : String
+ Contato(nome_contato : String, tal1_contato : String, tal2_contato : String, email_contato : String, cargo_contato : String) : Contato
Fornecedor
- nome_fornec : String
- cpf_fornec : String
- cnpj_fornec : String
- lograd_fornec : String
- bairro_fornec : String
- cep_fornec : String
- cidade_fornec : String
- estado_fornec : String
- dt_inativ : Date
+ Fornecedor(...definition... : int) : Fornecedor
Area_empresa
- nome_areaEmp : String
+ Area_empresa(aAreaEmpPai : Area_empresa, nome_areaEmp : String) : Area_empresa
Parcela_despesa
- vl_pago : Currency
- dt_pagto : Date
- vl_desconto : Currency
- dt_venc : Date
- vl_parcela : Currency
- dt_prevVenc : Date
- vl_prevParc : Currency
- obs_pagto : String
- dt_corrente : Date
+ Parcela_despesa(aParcDespPai : Parcela_despesa, dt_venc : Date, vl_parcela : Currency) : Parcela_despesa
+ Parcela_despesa(aDespesa : Despesa, dt_venc : Date, vl_parcela : Currency) : Parcela_despesa
+ Parcela_despesa(aDespesa : Despesa, dt_prevVenc : Date, vl_prevParc : Currency) : Parcela_despesa
+ descr_mov() : String
+ ref_mov() : String
+ vl_mov() : Currency
+ tipo_mov() : String
+ vl_difPrevParc() : Currency
+ parc_quitada() : boolean
+ vl_juros() : Currency
+ vl_multa() : Currency
+ atrasada() : boolean
+ informar_pagto(vl_pago : Currency, vl_desconto : Currency, dt_pagto : Date, conta : Object, parcelas_compl : Array<dt_venc: Date,vl_parcela: Currency>, obs_pagto : String, dt_corrente : Date) : void
Cliente_despesa
- custo : Currency
+ Cliente_despesa(custo : Currency) : Cliente_despesa
Cliente_despesa
Cliente
- nome_cli : String
- cpf_cli : String
- cnpj_cli : String
- lograd_cli : String
- bairro_cli : String
- cep_cli : String
- cidade_cli : String
- estado_cli : String
+ Cliente(nome_cli : String, cpf_cli : String, cnpj_cli : String, lograd_cli : String, bairro_cli : String, cep_cli : String, cidade_cli : String, estado_cli : String, contato_cli : Array<Contato>) : Cliente
+ lista_contratos(estado : String) : Array<Contrato>
+ nr_ocorItem(nome_servico : String, nome_itemCobr : String) : int
Despesa
- descr_despesa : String
- dt_geracao : Date
- nr_nota : String
- tx_juros : Percentual
- multa : Percentual
- dt_corrente : Date
+ Despesa(aClasseItem : Classe_item, descr_desp : String, aAreaEmp : Area_empresa, oFunc : Profissional_voce, dados_cli : Array<oCliente : Cliente,custo: Currency>, oFornec : Fornecedor, vl_desp : Currency, nr_nota : String, tx_juros : Percentual, multa : Percentual, dt_geracao : Date, nrTot_parcelas : int, dados_parc : Array<dt_venc: Date,vl_parcela: Currency>, dt_corrente : Date) : Despesa
+ Despesa(aClasseItem : Classe_item, descr_desp : String, aAreaEmp : Area_empresa, oFunc : Profissional_voce, dados_cli : Array<oCliente : Cliente,custo: Currency>, oFornec : Fornecedor, vl_prevDesp : Currency, nrTot_prevParc : int, dados_parcPrev : Array<dt_prevVenc: Date,vl_prevParc: Currency>, dt_corrente : Date) : Despesa
+ Despesa(aDespesa : Despesa, txCorrecao : Percentual, dt_corrente : Date) : Despesa
+ efetivar_despesa(vl_desp : Currency, nrTot_parcelas : int, dados_parc : Array<dt_venc: Date,vl_parcela: Currency>, nota : boolean, nr_nota : String, tx_juros : Percentual, multa : Percentual, dt_geracao : Date) : void
+ nr_parcela(aParcela : Parcela_despesa) : int
+ nrTot_parcelas() : int
+ nrTot_prevParc() : int
+ vl_difPrevDesp() : Currency
+ vl_desp() : Currency
+ vl_prevDesp() : Currency
Conta_bancaria
- nome_banco : String
- nr_banco : int
- nr_agencia : String
- nr_conta : String
- dt_abertura : Date
- vl_limite : Currency
- vl_saldo : Currency
- tel1_banco : String
- tel2_banco : String
- nome_gerente : String
- dt_corrente : Date
+ Conta_bancaria(...definition... : int) : Conta_bancaria
+ saldo_ini(dt_ref : Date) : Currency
+ saldo_fim(dt_ref : Date) : Currency
+ saldoDia_banco(dt_ref : Date) : Currency
+ vlARecDia_banco(dt_ref : Date) : Currency
+ saldoFDia_banco(dt_ref : Date) : Currency
Classe_item
- nome_classeItem : String
+ Classe_item(aClasseItemPai : Classe_item, nome_classeItem : String) : Classe_item
Profissional_voce
Caixa
- nome_caixa : String
- vl_saldo : Currency
- dt_inicio : Date
- dt_corrente : Date
+ Caixa(nome_caixa : String, oResp : Profissional_voce, vl_saldo : Currency, dt_inicio : Date, dt_corrente : Date) : Caixa
+ saldo_ini(dt_ref : Date) : Currency
+ saldo_fim(dt_ref : Date) : Currency
Contrato
- nome_indice : String
- dt_corrente : Date
- dt_inicio : Date
- dt_cancel : Date
- dt_correnteCancel : Date
- motivo_cancel : String
- obs_cancel : String
- carta : boolean
+ Contrato(nome_indice : String, dt_corrente : Date) : Contrato
+ contr_ativo() : boolean
+ cancelar_contr(dt_cancel : Date, motivo_cancel : String, obs_cancel : String, carta : boolean, dt_correnteCancel : Date) : void
Disp_servicos
- tx_juros : Percentual
- multa : Percentual
- dt_inicio : Date
- dt_termino : Date
- obs_dispServ : String
- dt_corrente : Date
- dia_venc : int
+ Disp_servicos(itemCobr_valorados : Array<oItemCobranca: Item_cobranca,vl_itemCobr: Currency,lim_trab: int,paga_porOcor: boolean>, oCliente : Cliente, oContrato : Contrato, oVendedor : Profissional_voce, tx_juros : Percentual, multa : Percentual, nrTotal_parcelas : int, dados_parc : Array<dt_venc: Date,vl_parcela: Currency,obs_parcela: String,nota: boolean,boleto: boolean>, dt_inicio : Date, dt_termino : Date, obs_dispServ : String, dt_corrente : Date, dia_venc : int) : Disp_servicos
+ Disp_servicos(itemCobr_valorados : Array<oItemCobranca: Item_cobranca,vl_itemCobr: Currency,lim_trab: int,paga_porOcor: boolean>, oCliente : Cliente, oContrato : Contrato, oVendedor : Profissional_voce, nrTotal_prevParc : int, dados_parc : Array<dt_prevVenc: Date,vl_prevParc: Currency>, dt_inicio : Date, dt_termino : Date, obs_dispServ : String, dt_corrente : Date, dia_venc : int) : Disp_servicos
+ Disp_servicos(oDisp_servicos : Disp_servicos, txCorrecao : Percentual, dt_corrente : Date) : Disp_servicos
+ efetivarDispServicos(itemCobr_valorados : Array<oItemCobranca: Item_cobranca,vl_itemCobr: Currency,lim_trab: int,paga_porOcor: boolean>, oContrato : Contrato, oVendedor : Profissional_voce, tx_juros : Percentual, multa : Percentual, nrTotal_parcelas : int, dados_parcelas : Array<dt_venc: Date,vl_parcela: Currency,obs_parcela: String,nota: boolean,boleto: boolean>, dt_inicio : Date, dt_termino : Date, obs_dispServ : String) : void
+ nr_parcela(aParcela : Parcela_recebimento) : int
+ nrTot_parcelas() : int
+ nrTot_prevParc() : int
+ vl_difPrevRec() : Currency
+ vl_rec() : Currency
+ vl_prevRec() : Currency
+ descr_mov() : String
Servico
- nome_servico : String
+ Servico(nome_serv : String, aClasse_item : Classe_item) : Servico
Item_cobranca
- nome_itemCobr : String
+ Item_cobranca(nome_itemCor : String, oServico : Servico) : Item_cobranca
ItemCobr_valorado
Sistema_financeiro
+ dt_ref(dt_base : Date) : Date
+ fluxo_financeiro(dt_inic : Date, dt_fim : Date, dt_corrente : Date, caixa : boolean, contasB : Array<Conta_bancaria>) : void
+ saldoI_diaFF(dt_ref : Date, caixa : boolean, contasB : Array<Conta_bancaria>) : Currency
+ totAPagar_diaFF(dt_ref : Date, caixa : boolean, contasB : Array<Conta_bancaria>) : Currency
+ totARec_diaFF(dt_ref : Date, caixa : boolean, contasB : Array<Conta_bancaria>) : Currency
+ saldoF_diaFF(dt_ref : Date, caixa : boolean, contasB : Array<Conta_bancaria>) : Currency
+ balanco_diaFF(dt_ref : Date) : Currency
+ saldoI_perFF(dt_inic : Date, dt_fim : Date, caixa : boolean, contasB : Array<Conta_bancaria>) : Currency
+ totAPagar_perFF(dt_inic : Date, dt_fim : Date, caixa : boolean, contasB : Array<Conta_bancaria>) : Currency
+ totARec_perFF(dt_inic : Date, dt_fim : Date, caixa : boolean, contasB : Array<Conta_bancaria>) : Currency
+ saldoF_perFF(dt_inic : Date, dt_fim : Date, caixa : boolean, contasB : Array<Conta_bancaria>) : Currency
+ balanco_perFF(dt_inic : Date, dt_fim : Date) : Currency
+ saldoIDia_caixais(dt_ref : Date) : Currency
+ vlAPagarDia_caixais(dt_ref : Date) : Currency
+ vlARecDia_caixais(dt_ref : Date) : Currency
+ saldoFDia_caixais(dt_ref : Date) : Currency
+ caixas_bancos(dt_inic : Date, dt_fim : Date, dt_corrente : Date, caixas : Array<Caixa>, contasB : Array<Conta_bancaria>) : void
+ saldoI_perCB(dt_inic : Date, dt_fim : Date, caixas : Array<Caixa>, contasB : Array<Conta_bancaria>) : Currency
+ saldoF_perCB(dt_inic : Date, dt_fim : Date, caixas : Array<Caixa>, contasB : Array<Conta_bancaria>) : Currency
+ balanco_perCB(dt_inic : Date, dt_fim : Date) : Currency
+ despesas(dt_inic : Date, dt_fim : Date, aClasseItem : Classe_item, oFornec : Fornec, status_desp : String, dt_corrente : Date) : void
+ totDia_pagos(dt_ref : Date, aClasseItem : Classe_item, oFornec : Fornecedor) : Currency
+ totDia_pagoDI(dt_ref : Date, aClasseItem : Classe_item, oFornec : Fornecedor) : Currency
+ totPer_pagos(dt_inic : Date, dt_fim : Date, aClasseItem : Classe_item, oFornec : Fornecedor) : Currency
+ totPer_pagoDI(dt_inic : Date, dt_fim : Date, aClasseItem : Classe_item, oFornec : Fornecedor) : Currency
+ totDia_aPagar(dt_ref : Date, aClasseItem : Classe_item, oFornec : Fornecedor) : Currency
+ totDia_aPagarDI(dt_ref : Date, aClasseItem : Classe_item, oFornec : Fornecedor) : Currency
+ totPer_aPagar(dt_inic : Date, dt_fim : Date, aClasseItem : Classe_item, oFornec : Fornecedor) : Currency
+ totPer_aPagarDI(dt_inic : Date, dt_fim : Date, aClasseItem : Classe_item, oFornec : Fornecedor) : Currency
+ totPer_aPagarAI(dt_corrente : Date, aClasseItem : Classe_item, oFornec : Fornecedor) : Currency
+ totDia_prevDesp(dt_ref : Date, aClasseItem : Classe_item, oFornec : Fornecedor) : Currency
+ totPer_prevDesp(dt_inic : Date, dt_fim : Date, aClasseItem : Classe_item, oFornec : Fornecedor) : Currency
+ geraBoletos(dt_inic : Date, dt_fim : Date, nr_inicBoleto : int, dt_corrente : Date) : void
+ geraNotas(dt_inic : Date, dt_fim : Date, nr_inicNota : int, dt_corrente : Date) : void
+ comissoes(dt_inic : Date, dt_fim : Date, oVend : Profissional_voce, dt_corrente : Date) : void
+ totDia_comis(dt_ref : Date, oVend : Profissional_voce) : Currency
+ totPer_comis(dt_inic : Date, dt_fim : Date, oVend : Profissional_voce) : Currency
+ recebimentos(dt_inic : Date, dt_fim : Date, aClasseItem : Classe_item, oCliente : Cliente, status_rec : String, dt_corrente : Date) : void
+ totDia_recDI(dt_ref : Date, aClasseItem : Classe_item, oCliente : Cliente) : Currency
+ totDia_recDIf(dt_ref : Date, aClasseItem : Classe_item, oCliente : Cliente) : Currency
+ totPer_recDI(dt_inic : Date, dt_fim : Date, aClasseItem : Classe_item, oCliente : Cliente) : Currency
+ totPer_recDIf(dt_inic : Date, dt_fim : Date, aClasseItem : Classe_item, oCliente : Cliente) : Currency
+ totDia_aRec(dt_ref : Date, aClasseItem : Classe_item, oCliente : Cliente) : Currency
+ totDia_aRecDI(dt_ref : Date, aClasseItem : Classe_item, oCliente : Cliente) : Currency
+ totPer_aRec(dt_inic : Date, dt_fim : Date, aClasseItem : Classe_item, oCliente : Cliente) : Currency
+ totPer_aRecDI(dt_inic : Date, dt_fim : Date, aClasseItem : Classe_item, oCliente : Cliente) : Currency
+ totPer_aRecA(dt_inic : Date, dt_fim : Date, aClasseItem : Classe_item, oCliente : Cliente) : Currency
+ totDia_prevRec(dt_ref : Date, aClasseItem : Classe_item, oCliente : Cliente) : Currency
+ totPer_prevRec(dt_inic : Date, dt_fim : Date, aClasseItem : Classe_item, oCliente : Cliente) : Currency
+ lista_caixas(nome_caixa : String) : Array<Caixa>
+ lista_conta_bancaria(cons_conta : String, cons_banco : String) : Array<Conta_bancaria>
+ lista_classes_item(nome_classeItem : String) : Array<Classe_item>
+ lista_servicos(nome_servico : String) : Array<Servico>
+ lista_item_cobranca(consulta_nome : String) : Array<Item_cobranca>
+ lista_parcelasRecebimento(dt_inicio : Date, dt_fim : Date, boleto : boolean, nota : boolean, tipo_rec : String, oCliente : Cliente, aClasseItem : Classe_item) : Array<Parcela_recebimento>
+ lista_dispServicos(cliente : Cliente, estado : String, paga_porOcor : boolean, dia_venc : int) : Array<Disp_servicos>
ItemCobr_valorado
- paga_porOcor : boolean
- vl_itemCobr : Currency
- lim_trab : int
- previsao : boolean
+ ItemCobr_valorado(aDisp_servicos : Disp_servicos, oItemCobranca : Item_cobranca, paga_porOcor : boolean, vl_itemCobr : Currency, lim_trab : int, previsao : boolean) : ItemCobr_valorado
Taxa_indice
- nome_indice : String
- vl_percIndice : Currency
- mesAno_indice : String
+ Taxa_indice(nome_indice : String, vl_percIndice : Currency, mesAno_indice : String) : Taxa_indice
Percentual_comissao
- perc_comissaAVista : Currency
- perc_comissaAPrazo : Currency
- vl_minimo : Currency
- vl_maximo : Currency
- dt_inicio : Date
- dt_fim : Date
- dt_corrente : Date
+ Percentual_comissao(perc_comissaAVista : Percentual, perc_comissaAPrazo : Percentual, vl_minimo : Currency, vl_maximo : Currency, dt_inicio : Date, dt_corrente : Date) : Percentual_comissao
Parcela_recebimento
- vl_recebido : Currency
- dt_receb : Date
- vl_desconto : Currency
- dt_venc : Date
- vl_parcela : Currency
- obs_parcela : String
- boleto : boolean
- nota : boolean
- dt_prevParc : Date
- vl_prevVenc : Currency
- nr_nota : int
- nr_boleto : int
- obs_receb : String
- dt_corrente : Date
- parc_vlVariavel : boolean
+ Parcela_recebimento(aPrevRecPai : Parcela_recebimento, dt_venc : Date, vl_parcela : Currency, obs_parcela : String, boleto : boolean) : Parcela_recebimento
+ Parcela_recebimento(aDisp_servicos : Disp_servicos, dt_venc : Date, vl_parcela : Currency, obs_parcela : String, nota : boolean, boleto : boolean) : Parcela_recebimento
+ Parcela_recebimento(aDisp_servicos : Disp_servicos, dt_prevVenc : Date, vl_prevParc : Currency) : Parcela_recebimento
+ descr_mov() : String
+ ref_mov() : String
+ vl_mov() : Currency
+ tipo_mov() : String
+ vl_difPrevParc() : Currency
+ parc_quitada() : boolean
+ vl_juros() : Currency
+ vl_multa() : Currency
+ atrasada() : boolean
+ vl_comissao() : Currency
+ informar_rec(vl_recebido : Currency, vl_desconto : Currency, dt_receb : Date, conta : Object, parcelas_compl : Array<dt_venc: Date,vl_parcela: Currency,obs_parcela: String>, boleto : boolean, obs_receb : String, dt_corrente : Date) : void
+ vl_aReceber() : Currency

Apêndice 5: Planilha de Evolução dos Modelos (EC-3)

Identificação do(s) elemento(s) envolvido(s) na manutenção		Tipo de manutenção	Análise
MUC	**MD**		
1º Modelo de Implementação 09-06-08			
	Atributo `dt_corrente` (classes Contrato, Percentual_comissao, Despesa, Disp_servicos, Parcela_despesa, Parcela_recebimento, Conta_bancaria, Caixa)	Inclusão Tipo A1	As regras não determinaram a persistência deste atributo. Contudo foi adicionado por motivos de segurança. Por que a regra não determinou a persistência? Porque essa informação será usada em outro sub-sistema não modelado nos IC's.
IC9 (Cadastrar cliente) – 09/06/08	Atributo `cargo_contato` (classe Contato)	Inclusão Tipo A1	As regras não determinaram a persistência deste atributo, mas ele foi adicionado, pois um usuário pode estar interessado nessa informação ao visualizar um determinado contato. Por que a regra não determinou a persistência? Porque essa informação será usada em outro sub-sistema não modelado nos IC's.
	Operação construtora `Cliente_despesa(custo: Currency): Cliente_despesa` (classe Cliente_despesa	Inclusão Tipo A2	As regras não determinaram a criação desta operação, mas ela foi adicionada por haver necessidade de se criar um contrutor para a classe Cliente_despesa. Por que a regra não determinou a criação da operação? Pois não há identificador gerado para esta classe na especificação.
	Atributo `dt_correnteCancel` (classe Contrato)	Inclusão Tipo A1	As regras não determinaram a persistência deste atributo. Contudo foi adicionado por motivos de segurança. Por que a regra não determinou a persistência? Porque essa informação será usada em outro sub-sistema não modelado nos IC's.
	Atributos `motivo_cancel`,	Inclusão	As regras não determinaram a persistência destes atributos.

Identificação do(s) elemento(s) envolvido(s) na manutenção		Tipo de manutenção	Análise
MUC	**MD**		
	`obs_cancel` e `carta` (classe Contrato)	Tipo A1	Contudo, eles foram adicionados, pois posteriormente pode-se desejar criar um relatório de contratos cancelados. Por que a regra não determinou a persistência? Porque essa informação não será usada na versão atual do sistema e, sim, em uma versão posterior.
IC23 (Consultar comissões de vendedores) – 09/06/08 Obs.: Os valores de comissão são calculados com base no valor histórico do percentual.	Atributo `dt_fim` (classe Percentual_comissao)	Inclusão Tipo A3	As regras não determinaram a persistência deste atributo, mas ele foi adicionado por proporcionar uma facilidade maior de implementação, visto que se dt_fim estiver preenchida será possível saber que o percentual de comissão não é mais válido. É possível obter esta informação no momento em que se informa um novo percentual de comissão. A data de fim do percentual de comissão vigente até então (dt_fim) será a data de início do novo percentual de comissão (dt_inicio) subtraída de um dia. Por que a regra não determinou a persistência? Porque esta informação não foi especificada nos IC's.
	Atributos `vl_desp` e `vl_prevDesp` (classe Despesa)	Modificação Tipo A4	Os atributos **vl_desp** e **vl_prevDesp** foram substituídos pelas operações **vl_desp(): Currency** e **vl_prevDesp(): Currency** respectivamente. Por que a regra não determinou que eles fossem operações? Porque houve um erro de interpretação do item elementar. Achava-se que no IC de Controle de Despesas (onde ele é usado) o valor dele seria o valor informado no IC de Cadastro de Despesa o que pode não ser verdade, visto que ele pode ter seu valor acrescido de

Identificação do(s) elemento(s) envolvido(s) na manutenção		Tipo de manutenção	Análise
MUC	**MD**		
			juros e multa.
	Atributo `obs_dispServ` (classe Disp_servicos)	Inclusão Tipo A1	As regras não determinaram a persistência deste atributo. Contudo, ele foi adicionado, pois posteriormente pode-se desejar criar um relatório de disponibilizações de serviços. Por que a regra não determinou a persistência? Porque essa informação não será usada na versão atual do sistema e, sim, em uma versão posterior.
	Operação construtora `DispServ_itemCobr (paga_porOcor: boolean, vl_itemCobr: Currency, lim_trab: int, previsao: boolean): DispServ_itemCobr` (classe DispServ_itemCobr que posteriormente teve seu nome trocado para itemCobr_valorado) Obs.: O nome da classe foi trocado, porque se encontrou um nome mais significativo para a classe.	Inclusão Tipo A2	As regras não determinaram a criação desta operação, mas ela foi adicionada por haver necessidade de se criar um contrutor para a classe DispServ_itemCobr (atual itemCobr_valorado). Por que a regra não determinou a criação da operação? Pois não há identificador gerado para esta classe na especificação.
	Atributos `cpf_fornec, cnpj_fornec, lograd_fornec, bairro_fornec, cep_fornec, cidade_fornec, estado_fornec` (classe Fornecedor)	Inclusão Tipo A1	As regras não determinaram a persistência destes atributos. Contudo, eles foram adicionados, pois posteriormente pode-se desejar criar um relatório de fornecedores. Por que a regra não determinou a persistência? Porque essa informação não será usada na versão atual do sistema

Identificação do(s) elemento(s) envolvido(s) na manutenção		Tipo de manutenção	Análise
MUC	**MD**		
			e, sim, em uma versão posterior.
	Atributos `tel1_banco`, `tel2_banco` e `nome_gerente` (classe Conta_bancaria)	Inclusão Tipo A1	As regras não determinaram a persistência destes atributos. Contudo, eles foram adicionados, pois posteriormente pode-se desejar criar um relatório de contas bancárias. Por que a regra não determinou a persistência? Porque essa informação não será usada na versão atual do sistema e, sim, em uma versão posterior.
2º Modelo de Implementação (17-06-08)			
`id_cliente` IC7 (Cadastrar contrato)	Associação entre as classes Cliente e Contrato O id_cliente constava no IC7, na especificação resultante da aplicação das regras. Saber do William porque ele posteriormente retirou este id.	Inclusão Tipo B1	Faltou perceber durante a elaboração do MUC que essa informação era necessária para relacionar um contrato a um cliente específico no momento do cadastramento do primeiro. Sem essa informação, os contratos estariam "soltos" depois de cadastrados.
`dt_inicio` IC7 (Cadastrar contrato	Atributo `dt_inicio` (classe Contrato)	Inclusão Tipo B2	Esta informação foi adicionada visando a facilitar a visualização dos contratos cadastrados para um dado cliente. Também está previsto a inclusão desta informação num relatório de contratos cancelados a ser criado posteriormente. Desde já existe uma consulta no sistema que permite o usuário visualizar os dados do contrato, inclusive a data de início (IC do tipo Retrieve).
`dia_venc` IC25 (Cadastrar disponibilizaç	Atributo `dia_venc` (classe Disp_servicos)	Inclusão Tipo B3	Faltou perceber durante a elaboração do MUC que essa informação era necessária para determinar a data do vencimento de cada uma das parcelas de valores variáveis a serem recebidas por uma disponibilização de serviços.

Identificação do(s) elemento(s) envolvido(s) na manutenção		Tipo de manutenção	Análise
MUC	**MD**		
ão de serviços a cliente) e IC28 (Efetivar disponibilizaç ão de serviços a cliente)			
	Operação `descr_mov(): String` (classe Disp_servicos) Perguntar ao William o porque da criação desta operação (não encontrei a justificativa nos diagramas Jude cujo nome inclui a data de 09-06-08).	Inclusão Tipo B4	Esta operação foi colocada nesta classe além da classe Parcela_recebimento, pois quando o IC1 (Informar Recebimento) estava sendo implementado, viu-se a necessidade do usuário visualizar, nesse momento, os serviços associados a uma disponibilização de serviços. Entretanto, este método ainda foi mantido na classe Parcela_recebimento, pois as parcelas de recebimento podem não estar associadas a todos os itens de cobrança valorados da disponibilização de serviços. Por exemplo, as parcelas geradas à partir do IC27 (Informar parcelas de valores variáveis) estarão associadas apenas aos itens de cobrança valorados que são pagos por ocorrência da disponibilização de serviços. Já as parcelas geradas no IC25 (Cadastrar disponibilização de serviços a cliente) e no IC28 (Efetivar disponibilização de serviços a cliente) estarão associadas somente aos itens de cobrança valorados que não são pagos por ocorrência da disponibilização de serviços. No IC1, a escolha de uma parcela de recebimento é feita pelo usuário escolhendo inicialmente a disponibilização de serviços correspondente, o que vem acompanhado da listagem dos serviços associados aos itens de cobrança, obtida com a operação descr_mov.

Identificação do(s) elemento(s) envolvido(s) na manutenção		Tipo de manutenção	Análise
MUC	**MD**		
3º Modelo de Implementação (01-09-08)			
`nome_itemCobr` IC6 (Controlar recebimentos)		Inclusão Tipo C1	Esse item já estava no MD à partir de outro IC. Faltou perceber durante a elaboração do MUC que essa informação era necessária na saída do relatório deste IC.
`vl_aReceber` IC6 (Controlar recebimentos)	Operação `vl_aReceber(): Currency` (classe Parcela_recebimento)	Inclusão Tipo C2	Esse item foi adicionado no MUC devido a necessidade de sua utilização no relatório do IC6. Com a aplicação das regras (R4b), foi obtida uma operação no MD.
`nome_contato`, `tel_cli`, `email_cli`, `obs_parcela`, `obs_receb` IC6 (Controlar recebimentos)		ExclusãoTipo C3	Essas informações foram excluídas do MUC devido a falta de espaço na folha de emissão do relatório. Entretanto, elas foram mantidas no MD, pois posteriormente pode-se desejar criar um relatório em que as informações estarão presentes.
	Operação `lista_contratos(estado: String): Array<Contrato>` (classe Cliente)	Inclusão Tipo C4	Operação adicionada devido a necessidade de se ter uma listagem de todos os contratos existentes de um cliente. Não foi determinada à partir das regras. Para refletí-la no MUC seria criado um IC para listar os contratos de um cliente, para um estado escolhido (ATIVO e INATIVO).
	Operação `nr_ocorItem(nome_servico: String, nome_itemCobr: String): int` (classe Cliente)	Inclusão Tipo C5	Operação criada devido a necessidade de se coletar do módulo PCMSO o número de ocorrências de um determinado item de cobrança associado a um serviço no IC27 (Informar parcelas de valores variáveis). Não haverá problemas com esse método, pois

Identificação do(s) elemento(s) envolvido(s) na manutenção		Tipo de manutenção	Análise
MUC	**MD**		
			pode ser colocado que ele foi obtido à partir da regra R4b.
	Operação `lista_caixas(nome_caixa: String): Array<Caixa>` (classe Sistema_financeiro)	Inclusão	Operação adicionada devido a necessidade de se ter uma listagem de todos os caixas existentes. Não foi determinada à partir das regras.
	Operação `lista_conta_bancaria(cons_conta: String, cons_banco: String): Array<Conta_bancaria>` (classe Sistema_financeiro)	Inclusão Tipo C4	Operação adicionada devido a necessidade de se ter uma listagem de todas as contas bancárias existentes. Não foi determinada à partir das regras.
	Operação `lista_classes_item(nome_classeItem: String): Array<Classe_item>` (classe Sistema_financeiro)	Inclusão Tipo C4	Operação adicionada devido a necessidade de se ter uma listagem de todas as classes de itens existentes. Não foi determinada à partir das regras.
	Operação `lista_servicos(nome_servico: String): Array<Servico>` (classe Sistema_financeiro)	Inclusão Tipo C4	Operação adicionada devido a necessidade de se ter uma listagem de todos os serviços existentes. Não foi determinada à partir das regras.
	Operação `lista_item_cobranca(consulta_nome: String): Array<Item_cobranca>` (classe Sistema_financeiro)	Inclusão Tipo C4	Operação adicionada devido a necessidade de se ter uma listagem de todos os itens de cobrança existentes. Não foi determinada à partir das regras.
	Operação `lista_parcelasRecebimento(dt`	Inclusão	Operação criada pela sua necessidade no IC6 (Controlar recebimentos), no IC20 (Gerar boletos) e no IC21 (Gerar notas

Identificação do(s) elemento(s) envolvido(s) na manutenção		Tipo de manutenção	Análise
MUC	**MD**		
	`_inicio: Date, dt_fim: Date, boleto: boolean, nota: boolean, tipo_rec: String, oCliente: Cliente, aClasseItem: Classe_item): Array<Parcela_recebimento>` (classe Sistema_financeiro)		fiscais). Não foi determinada à partir das regras. Esta operação seleciona as parcelas a serem listadas, o que no MUC é refletido pela ocorrência de id_cliente no fluxo recebimentos, além de outros filtros: per_apur, tipo_rec (no caso do IC6).
	`Operação lista_dispServicos(cliente: Cliente, estado: String, paga_porOcor: boolean, dia_venc: int): Array<Disp_servicos>` (classe Sistema_financeiro)	Inclusão Tipo C4	Operação criada devido a necessidade de sua utilização no IC1 (Informar recebimentos), IC27 (Informar parcelas de valores variáveis) e também para uma listagem das disponibilizações de serviços associadas a um cliente específico. Não foi determinada à partir das regras. Mesma análise da operação anterior.
	Atributo `parc_vlVariavel` (classe Parcela_recebimento) Não entendi. De onde veio essa necessidade? Como refletí-la no MUC?	Inclusão	Esta informação precisou ser adicionada devido a necessidade de se saber a quais itens de cobrança valorados uma parcela de recebimento está associada. Caso ela seja uma parcela de valores variáveis, sabe-se que ela está associada a itens de cobrança valorados pagos por ocorrência. Caso contrário, existe associação com itens de cobrança valorados pagos por seus valores globais (não pagos por ocorrência). No IC25 (Cadastrar disponibilização de serviços a cliente) e IC28 (Efetivar disponibilização de serviços a cliente) as parcelas geradas não são parcelas de valores variáveis, enquanto que no IC27 (Informar parcelas de valores variáveis) elas são.

Apêndice 6: Planilha dos Testes de Aceitação (EC-3)

TESTES DE ACEITAÇÃO DO SISTEMA – 14/10/2008

OO	IC	E	P	Nome Elem	Observações
1	9	F		cliente	O cadastramento de contatos deveria ser feito na mesma tela do cadastramento de cliente. 1-PROBLEMA DA INTERFACE COM O USUÁRIO
			1		Após o cadastramento de um cliente (empregador) pessoa física, a consulta não mostrou ele (a Regina acha que isso se dá antes do cliente ter um serviço cadastrado – é o que acontece hoje no sistema). 3-ERRO DE IMPLEMENTAÇÃO (ou em outro subsistema)
1	12				OK
1	10	F	1	servico	Tornar o **id_classeItem** condicional, pois o usuário nem sempre tem como classificar o serviço no momento do seu cadastramento. 5-REQUISITO (REGRA DE NEGÓCIO) NÃO ESPECIFICADO NO DI
1	25	F		disp_serv	A seleção do contrato (se for o caso) deveria mostrar o cliente (nome, ...), bem como as datas de início e de cadastramento do contrato, para facilitar a escolha do mesmo. 1-PROBLEMA DA INTERFACE COM O USUÁRIO
1	25	F	1	disp_serv	Não permite escolher previsão (em vez de real). AINDA NÃO IMPLEMENTADO (não foi considerado como problema)
1	25	P	1	servicos	Permite cadastrar mais de uma vez um mesmo item de cobrança, para um mesmo serviço. 4-ERRO DE IMPLEMENTAÇÃO (propiciado pela subespecificação do DI)
1	25	F	1	dt_termino	Aceita menor do que dt_inicio. 3-ERRO DE IMPLEMENTAÇÃO
1	25	F	1	dt_termino	Está obrigando entrar com uma data. É isso mesmo? PENDENTE DE ANÁLISE PELOS STAKEHOLDERS
	21	P		per_ger	A geração de nota deve buscar todas os recebimentos por ocorrência que ainda não foram considerados na emissão anterior, até a data fim informada (para evitar que algum recebimento passe sem ser cobrado). 5-REQUISITO (REGRA DE NEGÓCIO) NÃO ESPECIFICADO NO DI
1	25	F		id_profMel	O vendedor agora pode ser um "parceiro" (exemplo: escritório de advocacia) fora da empresa (que pode inclusive não ser um cliente). 6-MANUTENÇÃO EVOLUTIVA
1	25	F		dt_venc	Se a data de vencimento de uma parcela i for menor ou igual à data de venc. da parcela i-1, dar uma mensagem de alerta. 2-REQUISITO DA INTERFACE COM O USUÁRIO
1	25	I	1	dt_venc	Está aceitando 0 e 32, por exemplo. 3-ERRO DE IMPLEMENTAÇÃO
1	8				Permite cancelar um contrato mantendo suas disponibilizações de serviço (apesar das parcelas de recebimento deverem permanecer). 4-ERRO DE IMPLEMENTAÇÃO (propiciado pela subespecificação do DI)
					Bloquear, na 1ª. versão do sistema, o botões de exclusão (por exemplo, o botão de exclusão de contrato). PARTE

OO	IC	E	P	Nome Elem	Observações
					INTEGRANTE DA MIC QUE NÃO FOI SEGUIDA NA IMPLEMENTAÇÃO (não considerado).
	25		1		Possibilitar a alteração e cancelamento das parcelas (valor, vencimento,...) caso essas ainda não tenham sido pagas pelo cliente (motivo: resultado de negociação no cancelamento de um contrato, por exemplo). 7-NOVO REQUISITO (REGRA DE NEGÓCIO) NÃO APONTADO INICIALMENTE PELOS STAKEHOLDERS
4	21	F	1	notas	Notas zeradas (por exemplo: pcmso – cada novo que não aconteceu) estão sendo geradas. Não deve. 4-ERRO DE IMPLEMENTAÇÃO (propiciado pela subespecificação do DI)
1	25		1		A primeira parcela gerada deve ter data de vencimento igual à data base de vencimento informada. 7-NOVO REQUISITO (REGRA DE NEGÓCIO) NÃO APONTADO INICIALMENTE PELOS STAKEHOLDERS
4	21				Deve ser possível indicar algumas empresas cuja nota não deve ser emitida naquele momento. Além disso, deve ser possível gerar nota para algumas empresas escolhidas. 7-NOVO REQUISITO (REGRA DE NEGÓCIO) NÃO APONTADO INICIALMENTE PELOS STAKEHOLDERS?
					Novo IC: (Agente adm) Cancelar nota. (com o motivo do cancelamento). 7-NOVO REQUISITO (REGRA DE NEGÓCIO) NÃO APONTADO INICIALMENTE PELOS STAKEHOLDERS
4	21	F	1	notas	O sistema está gerando a mesma nota com número diferente. Não pode acontecer.4- ERRO DE IMPLEMENTAÇÃO
					A Regina sugere que o sistema mostre apenas as notas a serem geradas (as já geradas seriam mostradas em outro IC). 1-PROBLEMA DA INTERFACE COM O USUÁRIO
4	20	F	1	boletos	Boletos zerados (mesmo problema que ocorre na geração das notas). 3-ERRO DE IMPLEMENTAÇÃO
	20	F	1	boletos	O sistema está gerando a mesma nota com número diferente. Não pode acontecer. 3-ERRO DE IMPLEMENTAÇÃO
					Sugestão: que o sistema mostre apenas os boletos a serem gerados (os já gerados seriam mostrados em outro IC). 1-PROBLEMA DA INTERFACE COM O USUÁRIO
					Novo IC: (Agente adm) Cancelar boleto (ou talvez apenas uma exclusão, já que não é necessário manter registro disto). 7-NOVO REQUISITO (REGRA DE NEGÓCIO) NÃO APONTADO INICIALMENTE PELOS STAKEHOLDERS
	25				Para notas e boletos ainda não gerados ou cancelados (excluído), deve permitir alterar data de vencimento e valor das parcelas de recebimento associadas. 4-ERRO DE IMPLEMENTAÇÃO (propiciado pela subespecificação do DI).
4	21				Empresa 1177: disp. de serviço PCMSO/Coord. PCMSO cadastrado. A geração de nota não gera uma nota para essa disponibilização. 3-ERRO DE IMPLEMENTAÇÃO

Apêndice 7: Questionário dos Desenvolvedores (EC-3)

Desenvolvedores: William Fernandes e Bruno Oliveira Data: 22/08/2008

1- Como foi organizada a implementação do sistema? Qual o papel da especificação de ICs (II, DI, OO) nessa organização?
As seqüências admissíveis (SA's) dos OO's foram tomadas como base inicial para ordenar a implementação do sistema. Após ter sido decidido quais SA's seriam implementadas, estabeleceu-se a ordem de implementação delas com base na dependência dos IC's nelas contidos. Por exemplo, a seqüência admissível de Cancelamento de Contrato continha os IC's de Cadastramento de Contrato e de Cancelamento de Contrato. Por sua vez, o primeiro IC estava contido em outra SA chamada de Inclusão de Contrato que era constituída pelos IC's de Cadastramento de Cliente e Cadastramento de Contrato. Então, estabeleceu-se que a última SA devia ser implementada antes da primeira. Com a ordem de implementação das SA's estabelecida, montou-se um cronograma de desenvolvimento com os IC's a serem implementados.

2- Em que etapa se encontra a implementação? Quantificar.
Já foram implementados onze IC's num total de quatorze acordados para a primeira versão do Sistema de Controle Financeiro. Quantificando pelos OO's, seis num total de nove já foram concluídos.

3- Que diferenças existem entre o diagrama de classes e a implementação do banco de dados do sistema?
A primeira diferença notada é quanto a identificadores de classes que não existem no diagrama de classes, mas no banco de dados são necessários. Outra diferença é no que diz respeito aos relacionamentos entre as classes no diagrama de classes. No banco de dados os relacionamentos são representados pelos identificadores. A última diferença notada é a conversão de tipos que teve de ser feita de forma a se adequar aos tipos do banco de dados utilizado.

4- Como estão organizados os módulos executáveis na implementação? Qual é a relação com a especificação dos ICs? A funcionalidade especificada nos ICs é exatamente aquela implementada?
Cada IC é representado por um diretório que contém arquivos referentes a cadastro, alteração, visualização, alteração no banco de dados ou outros arquivos pertinentes ao fluxo de entrada e saída dos IC's. Deste modo, esta é a relação estabelecida entre a especificação dos IC's e os módulos executáveis. Acredita-se o que foi implementado no sistema está de acordo com o que foi especificado anteriormente.

5- Como foram organizados os testes do sistema? Que tipos de erros foram encontrados durante os testes do sistema? Qual o papel da especificação dos ICs (II, DI) nos testes?
Os testes do sistema foram organizados IC-a-IC, tendo sido realizados pelos programadores à medida que o IC era construído e ao final de sua implementação, era testado como um todo. Depois disso, os testes ficaram a cargo dos usuários. Os erros encontrados durante os testes do sistema foram apenas erros de programação.

A especificação dos IC's não foi usada diretamente nos testes, pois a implementação foi um reflexo da especificação.

6- Qual foi a influência dos Objetivos Organizacionais (OO) na implementação/testes dos módulos do sistema?
As seqüências admissíveis (SA's) que estão diretamente ligadas aos OO's foram preponderantes na implementação e conseqüentemente nos testes do sistema. Elas determinaram a ordem de implementação dos IC's e da mesma forma, a ordem dos testes.

7- Que outros modelos foram utilizados, além do modelo de ICs (II, DI, OO), durante a implementação?
O único modelo utilizado além do modelo de IC's foi o diagrama de classes obtido da aplicação das regras de derivação nos IC's.

8- Que outras observações você poderia fazer sobre a relação entre a especificação de ICs (II, DI, OO) e a implementação do sistema?
A especificação de IC's guiou a implementação do sistema, ajudou a distribuir melhor as tarefas entre os programadores e deu maior clareza aos programadores quanto ao que deveria ser implementado.

9- Descreva o ambiente de implementação: a) ambiente de desenvolvimento; b) linguagem de programação; c) banco de dados; d) Outros.
a) O ambiente de desenvolvimento utilizado é o Dreamweaver; b) As linguagens de programação utilizadas são PHP e JavaScript; c) O banco de dados utilizado é o MySQL; d) Está sendo utilizado AJAX para deixar as páginas mais dinâmicas e CSS para padronizar o estilo das páginas.