MARCO ANTÔNIO PALUDO

A APLICAÇÃO DE *PATTERNS* PARA PROVER REUSO DE ANÁLISE EM PROCESSOS DE NEGÓCIO

Dissertação apresentada ao Programa de Pós-graduação em Informática Aplicada da Pontifícia Universidade Católica do Paraná como parte dos requisitos para obtenção do título de Mestre em Ciências.

Área de concentração: Sistemas Distribuídos

Orientador: Prof. Robert Burnett

CURITIBA

1999

Paludo, Marco Antônio

A Aplicação de *Patterns* para Prover Reuso de Análise em Processos de Negócio. Curitiba, 1999.
117 p.

Dissertação (Mestrado) – Pontificia Universidade Católica do Paraná. Departamento de Informática.

1. Reuso de *Software* 2.Engenharia de *Software* 3.Orientação a Objetos - I. Pontificia Universidade Católica do Paraná II. Centro de Ciências Exatas e de Tecnologia III. Curso de mestrado em Informática Aplicada.

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
Pró-Reitoria de Pós-Graduação, Pesquisa e Extensão

ATA DA SESSÃO PÚBLICA DE EXAME DE DISSERTAÇÃO DO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA APLICADA DA PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ.

Exame de dissertação nº 005

Aos 22 dias do mês de abril de 1999, realizou-se a sessão pública de defesa de dissertação **"A APLICAÇÃO DE PATTERNS PARA PROVER REUSO DE ANÁLISE EM PROCESSOS DE NEGÓCIO"**, apresentada por Marco Antonio Paludo, ano de ingresso 1996, para obtenção do título de Mestre em Ciências. A Banca Examinadora foi composta pelos seguintes professores:

MEMBROS DA BANCA	ASSINATURA
Presidente:. **Prof. Dr. Robert C. Burnett (PUC-PR)**	
Prof. Dr. Alcides Calsavara (PUC-PR)	
Profa. Dra. Itana Maria de Souza Gimenes (UEM)	
Prof. Dr. Roberto Tom Price (UFRGS)	

De acordo com as normas regimentais a Banca Examinadora deliberou sobre os conceitos a serem atribuídos e que foram os seguintes:

MEMBROS DA BANCA	CONCEITOS
Presidente:. **Prof. Dr. Robert C. Burnett (PUC-PR)**	A
Prof. Dr. Alcides Calsavara (PUC-PR)	A
Profa. Dra. Itana Maria de Souza Gimenes (UEM)	A
Prof. Dr. Roberto Tom Price (UFRGS)	A
Conceito Final	A

Observações da Banca Examinadora

Profº Júlio Cesar Nievola
Coordenador do Programa de Pós-Graduação em Informática Aplicada–PUC-PR

Rua Imaculada Conceição, 1155 – Prado Velho – CEP 80215-901 – Telefone: +55 41 330-1669 / + 55 41 330-1654
Fax: + 55 41 330- 1669 – Curitiba – Paraná – Brasil - E-mail:ppgia@rla01.pucpr.br

Mestre é aquele que nem sempre ensina,
é aquele que de repente aprende.

Guimarães Rosa.

Ao meu filho Bruno e à minha esposa Ana, fontes de inspiração.

Aos meus pais Américo e Fidélia, pelo eterno incentivo.

AGRADECIMENTOS

Ao meu orientador, Professor Robert Burnett, pela dedicação, suporte e confiança.

Ao meu co-orientador, Professor Edgard Jamhour e ao Professor Alcides Calsavara, pela imensa contribuição.

À Professora Elisa Huzita, pelas sugestões que complementaram este trabalho.

Aos amigos Maísa e Flávio Pannuti, pelo indispensável apoio.

Aos professores e à turma de 1996, pela integração e companheirismo.

Ao Banco do Estado do Paraná S/A, pelo investimento.

À Productique e ao Banco do Estado do Paraná S/A, pelos subsídios para validação deste trabalho.

À Pontifícia Universidade Católica do Paraná, pela estrutura e suporte.

À Siemens Telecomunicações, por ter financiado parcialmente esta dissertação através da Lei número 8248.

À todos os colegas e amigos que contribuíram para a realização deste trabalho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AP - Anteprojeto

BP - *Business Process*

BPR - *Business Process Re-engineering*

CSCW - *Computer Supported Cooperative Work*

DM - *Data-Management*

GoF - *Gang of Four*

HI - *Human-Interaction*

MDS - Metodologia para Desenvolvimento de Sistemas

MVC - *Model View Controller*

OMT - *Object Modeling Technique*

OOA - *Object-Oriented Analysis*

OOD - *Object-Oriented Design*

PD - *Problem-Domain*

POSA - *Pattern-Oriented Software Architecture*

SI - *System-Interaction*

UML - *Unified Modeling Language*

WfMC - *Workflow Management Coalition*

WfMS - *Workflow Management System*

RESUMO

Esta dissertação propõe o uso de *patterns* para auxiliar o desenvolvedor de *software* a modelar processos de negócio. Abrange as fases iniciais do ciclo de vida do desenvolvimento de *software* e tem como objetivo prover meios para obter o reuso dos componentes destas fases.

Processos de negócio são aqui entendidos como atividades que coletivamente realizam um objetivo de negócio em uma estrutura organizacional. Referidos processos representam para a engenharia de *software* um desafio, pois a sua fase de análise é crítica e demanda parcela muito significativa de esforços no desenvolvimento de aplicações que os suportem.

Devido a essa ênfase na fase de análise, a solução proposta é a utilização de *patterns*, que atuam como estruturas a serem seguidas durante o desenvolvimento de *software*, abstraídas de práticas desenvolvidas, evoluídas e bem sucedidas. Os *Patterns* são empregados para atender a dois objetivos: fazer a modelagem dos processos de negócio e prover reuso de elementos da análise. Para tanto, dentre as principais metodologias orientadas a *patterns*, seleciona-se uma delas para validação. A partir disso, a metodologia selecionada é complementada propondo-se novos *patterns*, diagramas e etapas em seu processo. Em complemento, a estrutura de documentação dos *patterns* e alguns diagramas são alterados.

Este trabalho pretende contribuir, portanto, apresentando novas diretrizes para utilização e obtenção de *patterns*.

Como forma de validação das proposições, dois estudos de caso são apresentados e analisados, buscando-se, também com isso, a demonstração da adequada aplicabilidade dos *patterns* em processos de negócio.

RESUMO

Esta dissertação propõe o uso de *patterns* para auxiliar o desenvolvedor de *software* a modelar processos de negócio. Abrange as fases iniciais do ciclo de vida do desenvolvimento de *software* e tem como objetivo prover meios para obter o reuso dos componentes destas fases.

Processos de negócio são aqui entendidos como atividades que coletivamente realizam um objetivo de negócio em uma estrutura organizacional. Referidos processos representam para a engenharia de *software* um desafio, pois a sua fase de análise é crítica e demanda parcela muito significativa de esforços no desenvolvimento de aplicações que os suportem.

Devido a essa ênfase na fase de análise, a solução proposta é a utilização de *patterns*, que atuam como estruturas a serem seguidas durante o desenvolvimento de *software*, abstraídas de práticas desenvolvidas, evoluídas e bem sucedidas. Os *Patterns* são empregados para atender a dois objetivos: fazer a modelagem dos processos de negócio e prover reuso de elementos da análise. Para tanto, dentre as principais metodologias orientadas a *patterns*, seleciona-se uma delas para validação. A partir disso, a metodologia selecionada é complementada propondo-se novos *patterns*, diagramas e etapas em seu processo. Em complemento, a estrutura de documentação dos *patterns* e alguns diagramas são alterados.

Este trabalho pretende contribuir, portanto, apresentando novas diretrizes para utilização e obtenção de *patterns*.

Como forma de validação das proposições, dois estudos de caso são apresentados e analisados, buscando-se, também com isso, a demonstração da adequada aplicabilidade dos *patterns* em processos de negócio.

ABSTRACT

This dissertation proposes the use of patterns to help the software designer to model business processes. It focuses on the initial phases of the software development life-cycle and has the objective of promoting reuse of the components of theses phases.

Business processes are understood as activities, which collectively realize a business objective, within the context of an organizational structure. These processes represent a challenge to the software engineering because their analysis phase is critical and demands a significant portion of the efforts to develop applications that support them.

By this emphasis on the analysis phase, the proposed solution is to use patterns, which are structures to be followed during the development of software, abstracted and evolved from practical developments, which resulted in success. The patterns are used with two objectives: to model the business processes and to provide reuse of elements of analysis. For that, among the main pattern-oriented methodologies, one is chosen to be assessed. Then, the selected methodology is complemented with new patterns, diagrams and stages in its process. Complementary, the documentation structure of patterns and some diagrams are improved.

This work intends to contribute presenting new directions to use and to obtain patterns.

As a way of assessing the propositions, two case studies are presented and analyzed, trying to demonstrate the right applicability of patterns in business processes.

ABSTRACT

This dissertation proposes the use of patterns to help the software designer to model business processes. It focuses on the initial phases of the software development life-cycle and has the objective of promoting reuse of the components of theses phases.

Business processes are understood as activities, which collectively realize a business objective, within the context of an organizational structure. These processes represent a challenge to the software engineering because their analysis phase is critical and demands a significant portion of the efforts to develop applications that support them.

By this emphasis on the analysis phase, the proposed solution is to use patterns, which are structures to be followed during the development of software, abstracted and evolved from practical developments, which resulted in success. The patterns are used with two objectives: to model the business processes and to provide reuse of elements of analysis. For that, among the main pattern-oriented methodologies, one is chosen to be assessed. Then, the selected methodology is complemented with new patterns, diagrams and stages in its process. Complementary, the documentation structure of patterns and some diagrams are improved.

This work intends to contribute presenting new directions to use and to obtain patterns.

As a way of assessing the propositions, two case studies are presented and analyzed, trying to demonstrate the right applicability of patterns in business processes.

1 INTRODUÇÃO

Como os sistemas de informação tornaram-se cada vez mais complexos e disseminados, o desenvolvimento, a manutenção e a operação estão atingindo seus limites da viabilidade material e econômica. O desenvolvimento de *software* cada vez mais é visto como uma disciplina de engenharia e, desta forma, princípios e métodos a ela inerentes vêm sendo progressivamente aplicados, como exemplo planejamento, formalismo, previsão e capacidade de reproduzir projetos bem sucedidos.

O reuso de elementos de *software* vem ganhando atenção especial pela comunidade de *software*, agindo como um facilitador no desenvolvimento de sistemas com maior qualidade e com menores custos.

Não obstante os estudos sobre o tema não sejam novos na comunidade, no passado, o reuso foi predominantemente executado na modalidade *ad hoc*, por pequenos grupos ou individualmente.

Porém, para que seja efetiva, uma abordagem sistemática para reuso dos recursos em todo o ciclo de vida (requisitos, arquiteturas, projetos, código, testes etc.) deve ser enfatizada. É importante alavancar e aplicar o reuso em todos os produtos do ciclo de vida para economizar recursos financeiros. Segundo Jay Reddy em [RED 98] [1], não se deve despender esforços para obter reuso somente na codificação e implementação, como comumente ocorre, mas todo o ciclo deve ser atendido, uma vez que o desenvolvimento do código é uma fração do custo de construção e manutenção de um sistema.

Outro fator importante reside no fato de que as atividades de reuso devem estar muito bem integradas ao processo de desenvolvimento dos sistemas. Isto assegura que o reuso seja planejado, avaliado e praticado naturalmente, pois está inserido em todas as fases do processo.

---

[1] Esta dissertação seguiu as normas para apresentação de trabalhos propostas por [UFP 95], com exceção das referências bibliográficas. Em publicações da área de engenharia de *software* comumente utiliza-se a seguinte notação para referências: três letras iniciais do sobrenome do autor seguidas pelo ano de publicação, entre colchetes.

Considerando-se esses fatores, esta dissertação enfatizará o reuso de elementos da fase de análise, porém, sem desconsiderar as etapas anteriores, como levantamento de requisitos, ou posteriores, como *design* e implementação.

Reforçando a abordagem adotada nesta dissertação para o reuso das fases iniciais de desenvolvimento de *software*, Tim Korson [KOR 96a], em uma análise de diversos artigos [2] sobre o assunto, conclui que "um axioma comum entre os artigos é que o reuso de bibliotecas de classes e componentes *stand-alone* não alteram fundamentalmente o processo de desenvolvimento de *software*". Em complemento, destaca o fato de os autores concordarem que conceitos e ferramentas de mais alto nível são necessários para agregar mais valor ao processo de desenvolvimento.

A forma proposta para contemplar esses requisitos de reuso de alto nível e de um processo integrado é a inserção de *patterns* no processo. Os *patterns* atuam como planos ou estruturas a serem seguidas, oriundos de abstrações da realidade. São geralmente divididos em seções. Uma representa a integração entre os objetos ou classes, com responsabilidades e relacionamentos estereotipados. Outra seção contém a descrição detalhada, a forma de utilização, os problemas endereçados e as demais informações textuais.

A aplicação sistemática dos *patterns* provê uma melhor comunicação interna e externa ao projeto de desenvolvimento do *software*, assim como auxilia a disseminar experiências com modelagem de objetos [COA 97].

> O termo adotado na presente dissertação é 'pattern(s)', sendo que há referências como: 'patterns' [COA 97,BUS 96], 'analysis patterns' [FOW 97b,FOW 97a] e 'design patterns' [GAM 94,PRE 95]. Traduzindo literalmente, patterns são 'padrões'. Manter-se-á, contudo, o vocábulo original devido à larga utilização na bibliografia, seja para diferenciar da técnica de padrões (protótipos, telas, campos etc.) aplicadas na engenharia de *software*, seja para diferenciar do termo 'standard', que também tem como tradução 'padrão'.

Com as considerações sobre reuso e sobre *patterns* delineadas, o contexto de aplicação selecionado para esta dissertação é o de processos de negócio, que representam um conjunto de um ou mais procedimentos ou atividades que coletivamente realizam um objetivo político ou de negócio, normalmente no contexto de uma estrutura organizacional, definindo papéis funcionais e relacionamentos [WFM 96].

O conceito de *workflow*, por sua vez, representa a forma de automatização de processos de negócio. Dentro de um *workflow*, documentos, informações ou tarefas transitam entre participantes, conforme regras procedimentais.

Pode-se enumerar algumas diferenças entre as aplicações operacionais e as aplicações orientadas para processos de negócio. Para estas, a determinação das rotas[3] que as informações devem seguir, as autenticações, as aprovações e regras são essenciais. Ao passo que para aquelas, as próprias informações e seus subprodutos, como totais e grupamentos, geralmente são mais importantes que as rotas que percorreram. Ou seja, a parte dinâmica das aplicações de processos de negócio superam, em importância, a parte estática.

O objetivo da utilização dos *patterns*, nesta dissertação, é apoiar a modelagem dos processos de negócio, permitindo que estes tornem-se *workflows*. Para isso, são identificados *patterns* que abstraem uma parcela das tarefas e atividades de um processo de negócio.

São propostas complementações e inserções de novos elementos, para atuarem em colaboração com os *patterns*. Isto devido à forma como são apresentados nas metodologias não ser satisfatória para prover o reuso de componentes de análise.

Dentre os novos elementos, estão os cenários de *patterns* com uma abordagem de serviços e estruturas de controle estereotipados, que passam a ser instanciados da mesma maneira que os *patterns*. Desta forma, é possível fazer reuso de cenários a partir de um catálogo e, eventualmente, até a partir de outros projetos. Esta abordagem difere da metodologia base, detalhada posteriormente, na qual os cenários são elaborados a partir do diagrama de classes e objetos, sem possibilitar seu reuso.

A forma adotada para o detalhamento dos assuntos citados e dos complementares, inicia pela apresentação da justificativa e motivação para o desenvolvimento desta dissertação. Neste mesmo momento, caracteriza-se o problema a ser endereçado, completando o conteúdo da subseção 1.1.

Em seguida, a hipótese e a delimitação do trabalho são detalhadas na subseção 1.2. A hipótese é apresentada como a busca do reuso de componentes de análise na modelagem de

---

[2] Os autores dos artigos são: Jim Doble, Martin Griss, Robert Kessler, John McGregor,Vijiay Vaishnavi, Asha Keddy e Rajendra Bandi. A publicação é: Object Magazine, v. 6, n. 1, April 1996.

[3] Rotas: representa o fluxo percorrido por objetos dentro de um *workflow*, conforme a determinação de condições e regras. Regras: determinam para quem e quais informações serão roteadas em um *workflow*. [COL 95].

aplicativos[4] que suportam processos de negócio. Após discorrer sobre o estado da arte das tecnologias, as proposições sobre estas e os estudos de caso, a hipótese será validada. Todas estas fases são norteadas pela delimitação ora apresentada.

Como introdução ao assunto sobre os *patterns*, o capítulo 2 apresenta algumas definições e a origem dos estudos e da utilização sobre *patterns*, *design patterns* e *frameworks*. Faz referência às principais metodologias de *patterns*, separadas pelo critério de abordagem, quais sejam, dirigidas a exemplos e dirigidas a processo. Neste capítulo, ainda, é apresentada detalhadamente a metodologia Estratégias e *Patterns* [COA 97], assim como a argumentação sobre sua seleção como metodologia base para desenvolvimento de *patterns* de análise desta dissertação.

O capítulo 3 é composto pelas proposições que complementam e estendem a metodologia base, no sentido de atender aos requisitos de processos de negócios, apresentados anteriormente. Tais alterações, juntamente com a identificação de alguns *Patterns* de Processos de Negócio, formam a metodologia a ser aplicada e validada nos estudos de caso.

No capítulo 4 são apresentados dois estudos de caso com características distintas, porém pertencentes ao mesmo domínio de processos de negócio. O primeiro estudo de caso modela uma aplicação de gerência de compras, com as aprovações, as responsabilidades e as possíveis rotas que as informações podem seguir. Após esta modelagem, os *patterns* específicos para processos de negócio, propostos por esta dissertação, são validados. Visto que os *patterns* são resultantes de aplicações práticas, é natural que sejam validados e complementados a cada novo ciclo de aplicação. Desta forma, os *patterns* propostos são evoluídos e disponibilizados para novas validações.

O segundo estudo de caso modela uma aplicação de Sistemática de Trabalho e, assim como no primeiro, valida e complementa os *Patterns* de Processos de Negócio. Juntamente com os dois estudos de caso, são evidenciados os resultados obtidos, que são confrontados com a hipótese apresentada no início. Após, são apresentadas as contribuições obtidas com as proposições e com o desenvolvimento dos estudos de caso.

Finalmente, no capítulo 5 são apresentadas as conclusões gerais deste trabalho, assim como perspectivas e proposições para trabalhos futuros.

---

[4] Na presente dissertação, os termos aplicação, aplicativo e sistema de informação possuem a mesma conotação.

1.1 Justificativa e Caracterização do Problema

Nesta subseção, são apresentadas as justificativas e motivações para o desenvolvimento da presente dissertação, através da exposição dos problemas encontrados na engenharia de *software*, particularmente no tocante ao processo de desenvolvimento de *software* que suporte processos de negócio. Inicialmente são feitas considerações sobre reuso, seguidas pelas características de *patterns* e *frameworks*, finalizando com os aspectos de processos de negócio e *workflow*.

O problema que se pretende endereçar, em síntese, é a obtenção do reuso de elementos da fase de análise e a justificativa para o desenvolvimento deste trabalho é atender ao processo de modelagem de sistemas que suportem processos de negócio, valendo-se do reuso para sua confecção. Nas próximas subseções, os componentes do problema e da justificativa serão mais detalhadamente apresentados.

1.1.1 Reuso

A engenharia de *software* vem evoluindo e refinando técnicas e métodos para o desenvolvimento de aplicações e sistemas de informação que sejam confiáveis, flexíveis, de fácil manutenção e com custo e prazos de desenvolvimento viáveis. Um dos fatores que proporciona meios para atingir essas metas é o emprego de reuso no processo de desenvolvimento.

O reuso tem como objetivo principal reaproveitar módulos de projetos que já tenham sido desenvolvidos, testados e implantados anteriormente com sucesso, visando uma diminuição considerável no tempo de desenvolvimento e uma facilidade maior de depuração, que, por conseqüência, desencadeiam os demais objetivos a serem alcançados. Segundo Charles Krueger, em [KRU 92], o reuso de *software* "é o processo de criação de sistemas de *software* a partir de *software* existente, ao invés de construir sistemas de *software* a partir do nada."

Uma técnica que possibilitou a aplicação mais extensiva do reuso no desenvolvimento de *software* foi o paradigma da orientação a objetos, não obstante seja viável o reuso sem esta. Os primeiros ensaios para aplicação de reuso ocorreram com o código fonte dos sistemas, através

da utilização de esqueletos de programas, edição de programas já existentes e utilização de bibliotecas de sub-rotinas e funções.

A programação orientada a objetos veio corroborar o reuso no nível de código fonte, pois a sua concepção vem ao encontro dos pré-requisitos para implantação de reuso. Cita-se a existência de classe, objetos, herança e polimorfismo como grandes alavancas para o reuso na programação orientada a objetos. Segundo [JOH 88], muitas das técnicas para reutilização de código em linguagens tradicionais são semelhantes às técnicas orientadas a objetos, como exemplos, tanto o fato de os esqueletos de programas serem inteiramente substituíveis por classes abstratas; quanto o de que cópia e edição de programas são substituíveis por herança de classes com substituição de métodos[5].

No final da década de 80, alguns autores começaram a estudar as formas de aplicação das técnicas já utilizadas no reuso de código, agora para reutilizar elementos da análise e de *design* de sistemas, que compõem as fases iniciais do processo de desenvolvimento de *software*, tais como [JOH 88, CAM 91, COA 91a] . Pelas mesmas características que beneficiam o reuso na programação orientada a objetos, a análise orientada a objetos (*object-oriented analysis - OOA*) e o *design* orientado a objetos (*object-oriented design - OOD*) tiveram mais enfoque de estudos.

Lajoie [LAJ 94] baseia-se em uma escala gráfica que representava características de programação para desenvolver uma classificação de níveis de reutilização e abstração em *frameworks*. Originalmente a classificação era *programming-in-the-small*, *programming-in-the-medium* e *programming-in-the-large*, representando as hierarquias de programação, entretanto foi adotada a notação *reuse-in-the-small*, *in-the-medium* e *in-the-large*, que pode ser traduzido para reuso em baixa, média e alta escala.

A Figura 1 apresenta os três níveis, nos quais observa-se que no reuso-em-baixa-escala são tratados elementos de baixo nível, como classes, métodos e/ou fragmentos de código. No reuso-em-média-escala são envolvidas microarquiteturas[6] isoladamente, assim como as interações entre microarquiteturas, aqui denominadas *frameworks*. E, no mais alto nível, o reuso-em-alta-escala, é considerado o reuso de aplicações, que são subsistemas independentes e podem ser reutilizados com poucas modificações ou extensões.

---

[5] Os conceitos de programação orientada a objetos não são expostos e discutidos, visto que a ênfase do presente trabalho recai sobre os níveis mais abstratos.

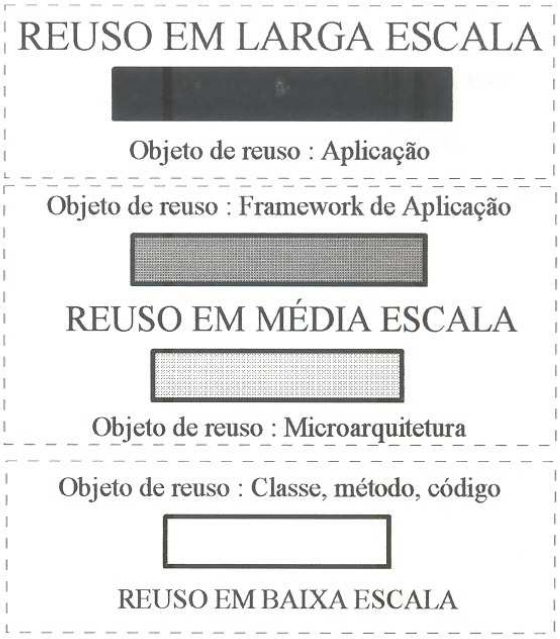


Figura 1- Três níveis de reuso [LAJ 94].

Vindo ao encontro do reuso-em-larga-escala, Tim Korson, em [KOR 96b], apresenta o axioma: "Os esforços de reuso limitados ao desenvolvimento e utilização de bibliotecas de classes não afetam fundamentalmente a produtividade do processo de desenvolvimento de *software*". Isto implica que a carteira (*portfolio*) de reuso não seja composta somente por classes, mas contenha *patterns*, linguagens de *patterns*, *frameworks* e arquiteturas. Observa, ainda, que alguns desses recursos (ex. *patterns* e arquiteturas) são conceituais em natureza, ao passo que outros (classes e *frameworks*) são a realização dos conceitos em código.

Complementa afirmando que classes como *string,* cliente, conta etc. elevam o nível de produtividade dentro de um paradigma, mas não alteram o paradigma de desenvolvimento de *software*. Para tal, o reuso em um nível mais elevado é requerido.

1.1.2 Orientação a Objetos e *Patterns*

O desenvolvimento de *software* orientado a objetos é plenamente atendido por diversas metodologias de autores que as têm evoluído há mais de uma década, deixando, pois, de representar um problema. Cada metodologia possui uma notação particular, uma ênfase em especial e um processo bem definido, contudo não apresentam soluções ou ferramentas para obter o reuso em larga escala do domínio do problema e das decisões tomadas nos projetos [PAL 98b].

---

[6] Microarquiteturas - termo apresentado em [GAM 94], que representa estruturas de classes e suas interações.

O relacionamento entre *patterns* e orientação a objetos é próximo, e a preocupação atual da comunidade de orientação a objetos não mais está voltada exclusivamente para ferramentas, técnicas, notações ou código. Segundo Ward Cunningham, no prefácio de [FOW 97a], as falhas ocorridas podem ser creditadas à falta de experiência e *patterns* representam esta experiência.

Os primeiros estudos divulgados sobre *patterns* receberam a denominação de *design patterns*, por focarem principalmente as fases de *design* e implementação através de classes abstratas. Na busca para alcançar o reuso nos níveis mais altos, onde os *patterns* são aplicados desde o início da análise, passaram por modificações estruturais ganhando, em virtude disso, denominação de *analysis patterns* (*patterns* de análise) ou somente *patterns*.

Embora no capítulo 2 sejam apresentadas as definições e aprofundados os conceitos sobre *framework*, *design patterns* e *patterns*, é importante salientar desde já, ainda que brevemente, a diferenciação existente:

- *Design patterns* possuem classes abstratas em suas estruturas, que objetivam expressar um projeto abstrato. Enfatizam fortemente a fase de *design*.
- *Frameworks* são arquiteturas de desenvolvimento, normalmente automatizadas, compostas por *design patterns*. Os *frameworks* podem ser considerados como ambientes de desenvolvimento onde o aplicativo é obtido através da montagem de partes existentes. Procuram atingir tanto o *design* e implementação, quanto as etapas da análise.
- *Patterns* ou *Analysis Patterns* possuem objetivos similares aos *design patterns*, porém com ênfase declarada para as fases iniciais do ciclo de desenvolvimento de *software*.

Os conceitos de orientação a objetos estão presentes nas três categorias anteriores, pois todos valem-se de hierarquias, classes, objetos, métodos e atributos em suas estruturas.

1.1.3 Processos de Negócio e *Workflow*

O *Workflow Management Coalition* (WfMC) [7] apresenta as seguintes definições:

- Processo de Negócio (*Business Process*): conjunto de um ou mais procedimentos ou atividades que coletivamente realizam um objetivo político ou do negócio, normalmente

---

[7] WfMC (Coalizão para gerenciamento de *workflow*): organização sem fins lucrativos, que objetiva proporcionar avanços nas tecnologias de gerenciamento de *workflow* e o desenvolvimento de terminologias comuns e padrões para *workflow* [LAW 97].

no contexto de uma estrutura organizacional, definindo papéis funcionais e relacionamentos [WFM 96, LAW 97].

- *Workflow*: a automação de um processo de negócio, em parte ou totalmente, durante o qual documentos, informações ou atividades são passadas de um participante para outro, de acordo com regras procedimentais [WFM 96, LAW 97].
- Definição do Processo (*Process Definition*): a representação de um processo de negócio de forma a suportar manipulação automatizada (modelagem). É composta pelas atividades e seus relacionamentos, os critérios de início e término do *workflow*, os participantes e dados e aplicações de informática associados [WFM 96, LAW 97].

Um processo de negócio é normalmente associado aos objetivos operacionais e relacionamentos de negócio entre empresas, como exemplo, processo de solicitação de seguros ou processo de fornecimento de manufaturas. Pode estar contido somente dentro de uma organização, ou contemplar diversas, como no caso dos relacionamentos entre clientes-fornecedores. Na Figura 2 são evidenciados alguns exemplos de processos de negócio.

Para automatizar os processos de negócio e obter um *workflow*, é necessário que seja elaborada a definição do processo. Nesta, devem constar as atividades, regras de procedimentos e controle de dados utilizados para gerenciar o *workflow*.

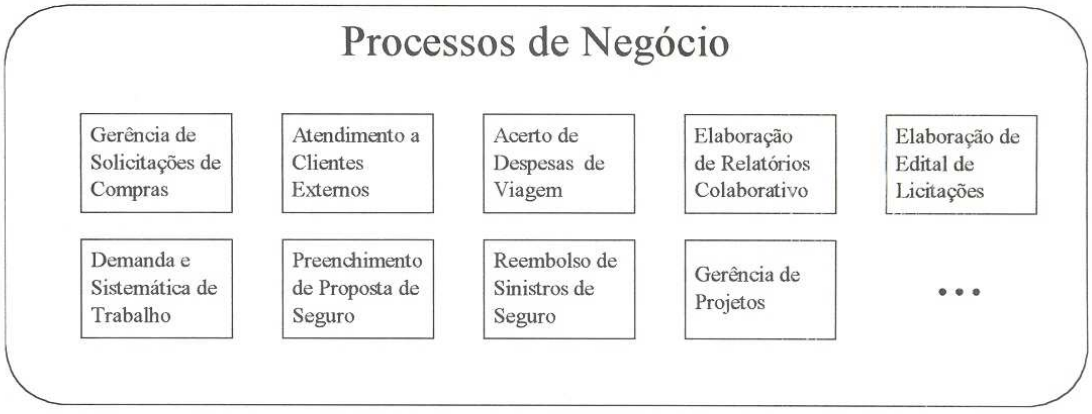


Figura 2 – Exemplos de Processos de Negócio.

É importante conceituar BPR (*Business Process Re-engineering*), ou Reengenharia de Processos de Negócio, sendo as atividades de verificação, análise, modelagem, definição e subseqüente implementação operacional dos processos de negócio essenciais de uma organização [LAW 97].

Segundo Michael Amberg, em [LAW 97], a modelagem de *workflow* e BPR surgiram independentemente e, visto que os objetivos e os conceitos metodológicos são similares, são comumente usados como sinônimos. Porém, BPR procura analisar os processos de negócio e apresentar variantes na forma de alternativas que são reveladas, investigadas e avaliadas. Seu foco primário é planejar melhores processos de negócio.

Já a modelagem de *workflow* enfatiza a execução do negócio e dos processos de negócio. A análise e o *design* são diretamente mapeadas para a implementação. As variantes de processos descartadas não são importantes para a implementação do *workflow*.

Segundo o modelo de referência de *workflow* apresentado por WfMC em [WFM 95], nem todas as atividades do BPR resultam em *workflow*, mesmo esta sendo uma tecnologia adequada para implementação, assim como nem toda a modelagem de *workflow* é proveniente dos resultados de BPR.

O relacionamento entre *workflow,* processos de negócio e *groupware* ocorre através das técnicas de coordenação. Entende-se por *groupware* uma tecnologia mais genérica de suporte à colaboração, focada em coordenação e segundo [LOT 95], *groupware* é a interseção de três tecnologias de desenvolvimento de *software*, representadas na Figura 3 e relacionadas na seqüência:

- Comunicação: é a transmissão do conhecimento através das trocas de mensagens pelos meios eletrônicos, principalmente *e-mail*.[LOT 95]
- Colaboração: procura aprimorar o trabalho em equipe, armazenando as informações em espaços compartilhados e fazendo as combinações de espaço (localidade) e tempo necessárias [LOT 95]. Já em [POM 98], esses dois aspectos (comunicação e colaboração) são considerados como Cooperação: "...recursos de comunicação para que os usuários possam trocar mensagens" e "... o compartilhamento de informações através de base de dados ou através de documentos".
- Coordenação: representada principalmente pelo *workflow*, sustenta as políticas das empresas em fazer as pessoas que trabalham em colaboração atingirem os objetivos determinados. Para tal, seguem um conjunto estruturado de tarefas, em uma seqüência particular, respeitando as condições e restrições de tempo. Com isto, os processos do negócio são modelados e integrados à comunicação e à colaboração [LOT 95]. Em [POM 98], a abordagem é sobre o uso controlado de diversas pessoas sobre os mesmos

objetos e a integração das contribuições individuais de cada participante em um único produto.

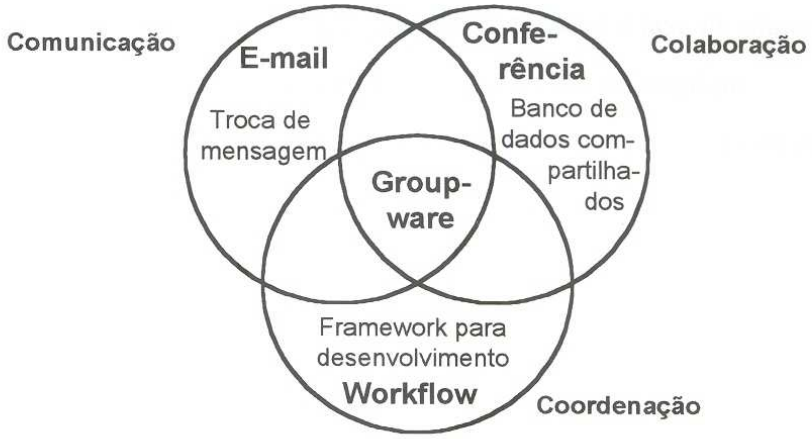


Figura 3 – Componentes de *groupware* [LOT 95].

Stefan Schereyjak, em [SCH 98], propõe que o ciclo de vida de um *workflow* pode ser dividido em três fases: planejamento, execução e análise.

1) Fase de planejamento: é feita a modelagem da estrutura organizacional e a ordem de todos os procedimentos de negócio. Todo o trabalho é dividido em passos individuais, que são conectados através de um relacionamento com uma ordem cronológica, construindo, desta forma, o processo. Um passo do processo, também chamado atividade, é uma parte de um trabalho contínuo executado por uma pessoa. É determinado, para cada atividade, quais objetos de trabalho (dados e documentos) e quais recursos humanos, técnicos e organizacionais são necessários para a sua execução. O processo de negócio modelado é especificado como um 'esquema de *workflow*', em uma linguagem de texto formal ou *workflow* visual.

2) Fase de execução: os esquemas são usados para controlar, supervisionar e arquivar as atividades executadas. A instância executável de um 'esquema de *workflow*' é chamado simplesmente de *workflow* . O ator de uma atividade pode usar programas aplicativos interativos para suprir os objetivos da atividade. Como uma alternativa, atividades manuais, sem assistência de computador, também podem ocorrer, assim como atividades que dispensam a interação humana.

3) Fase de análise: os dados armazenados são avaliados. Os resultados extraídos servem como base para a melhoria dos esquemas do *workflow*. Se todas as fases forem aplicadas diversas vezes em ciclos, os processos de negócio podem ser otimizados gradativamente.

Dentre essas fases, a endereçada pela presente dissertação é a fase de planejamento, visto que o foco é essencialmente a análise e modelagem dos processo de negócio.

Como último aspecto analisado para *workflows*, é utilizada a proposta de classificação de Esther Dyson, em [COL 95], onde o controle das atividades são considerados como foco primário para a classificação:

- *Workflow* centrado em usuário: os usuários interagem com o grupo de trabalho através de *e-mail* e decidem o roteamento das mensagens. Não há nenhuma gerência pelo sistema [8], nem integração com banco de dados.
- *Workflow* centrado em objetos (ou documentos): o objeto é um documento que contém as informações de um processo executado em etapas. Define a seqüência que o objeto percorre, porém apresenta problemas de integridade, no caso de perda do objeto ou documento.
- *Workflow* centrado em processos: as informações do andamento do trabalho estão contidas também no sistema (normalmente um banco de dados) e não somente no objeto. O sistema segue o estado do trabalho e faz o gerenciamento para que este seja completado.

Pela ênfase à modelagem de processos de negócio nesta dissertação, a abordagem de *workflow* endereçada é a centrada em processos, especialmente devido ao sistema exercer controle sobre os fluxos e rotas das atividades.

1.1.4 Forma de Abordagem do Problema

As diversas metodologias para desenvolvimento de sistemas (MDS) existentes contemplam todo o ciclo de desenvolvimento de aplicativos tradicionais (*on-line*, *batch*, tempo-real etc.), no entanto são pouco eficientes para modelar as rotas, regras e papéis existentes nos *workflows* e nos processos de negócio.

---

[8] Sistema de Gerenciamento de *Workflow* (WfMS) : Um sistema que define, cria e gerencia a execução de um *workflow* através do uso de *software*. [WFM 96, LAW 97].

A motivação principal para o desenvolvimento deste trabalho é atender a dois dos aspectos apresentados anteriormente: o reuso de elementos de análise (subseção 1.1.1) e a modelagem de processos de negócio (subseção 1.1.3). A forma de abordar estes aspectos é através da aplicação dos *patterns*, procurando viabilizar o reuso nas fases iniciais do processo de desenvolvimento de *software*.

Desta forma, o objetivo é a obtenção de modelos de processos de negócio eficientes e reutilizáveis, tendo o suporte dos *patterns* para fornecer a arquitetura básica, a notação e o processo para solucioná-lo. Na próxima subseção são apresentados, de forma mais detalhada, a hipótese, os objetivos a serem alcançados e a delimitação do trabalho.

1.2 Hipótese e Delimitação do Trabalho

Esta subseção delimita as fronteiras de atuação e apresenta a hipótese que se pretende comprovar. As justificativas para a aplicação de técnicas de reuso em todas as fases da concepção de um *software* estão apresentadas na subseção 1.1, ao passo que o capítulo 2 tratará da conceituação das técnicas para viabilizar tal reuso.

A primeira atividade a ser desenvolvida é o levantamento das diversas metodologias orientadas a *patterns*, seguida pela escolha daquela que possibilite o reuso das fases iniciais do desenvolvimento. A partir da metodologia escolhida, serão feitas proposições e complementações visando atender a modelagem de processos de negócio.

Para delimitar o escopo de atuação, é proposta a separação dos processos de negócio segundo um critério de agrupamento por funcionalidade. Desta forma, os processos que desempenham funções similares são agrupados em subconjuntos denominados Domínios [9] de Processos de Negócio.

Para exemplificar essa delimitação, a Figura 4 apresenta três domínios de processos de negócio distintos: Solicitação e Aprovação de Serviços, Atendimento a Clientes e Controle de Ressarcimentos. Cada um desses domínios contém alguns processos de negócio que condizem com o seu escopo. Como exemplo, o processo de negócio Gerência de Solicitações de Compras, assim como o processo Demanda e Sistemática de Trabalho possuem atividades

---

[9] Ressalta-se a diferença entre o termo Domínio, ora apresentado, e a expressão Domínio do Problema, originário de *problem-domain*, extensivamente utilizado na engenharia de *software*.

similares, tornando-os integrantes de um único domínio, no caso Solicitação e Aprovação de Serviços.

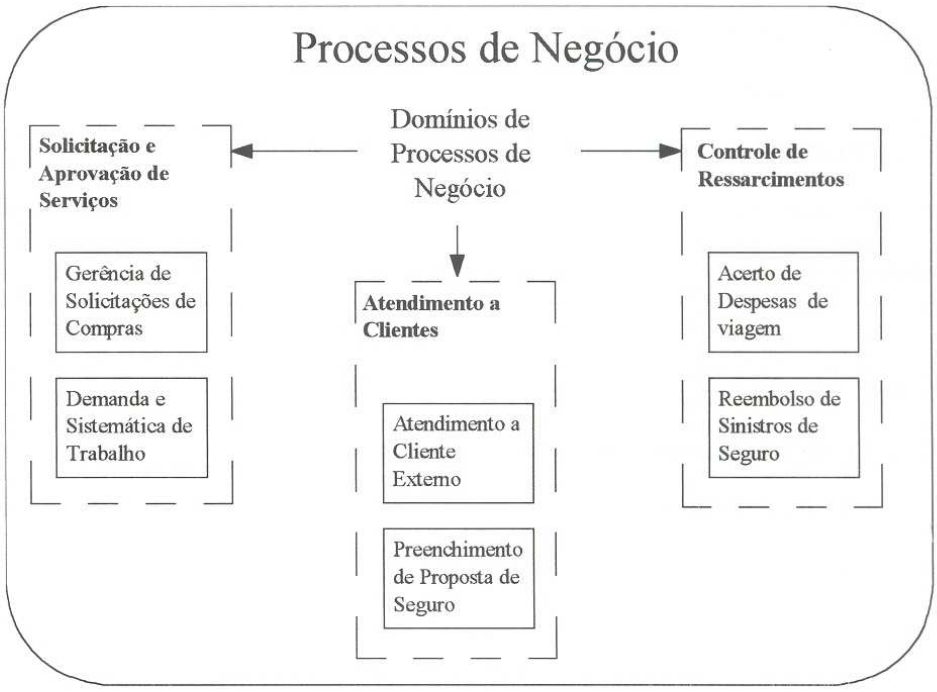


Figura 4 - Delimitação de Domínios para Processos de Negócio.

O domínio de processos de negócio Solicitação e Aprovação de Serviços foi o selecionado para ser estudado e desenvolvido por esta dissertação. Para este, são feitas as proposições de *patterns*, modelos de objetos e cenários. Ou seja, a estrutura das proposições pretende abranger qualquer domínio de processos de negócio, porém os *patterns*, modelos de objetos e cenários desenvolvidos fazem referência a somente um domínio.

A validação da metodologia, e principalmente das propostas de complementação para processos de negócio, será feita a partir do desenvolvimento de dois estudos de caso.

Para o primeiro estudo de caso, o processo de negócio a ser endereçado é um sistema de Gerência de Solicitações de Compras, sobre o qual espera-se aplicar a metodologia e sua complementação, obtendo-se, assim, os modelos e diagramas que compõem a modelagem da aplicação. Visto que o objetivo principal é atingir os componentes da fase de análise, esta terá maior ênfase, contudo o início da fase de *design* também poderá ser contemplada.

Como resultado deste primeiro estudo de caso, espera-se obter os produtos da fase de análise e a modelagem lógica do sistema. Adicionalmente, espera-se que este esforço

retroalimente a própria metodologia, validando e complementando os *patterns* propostos e, eventualmente, até adicionando novos *patterns* ao conjunto existente.

Após a conclusão e validação, será desenvolvido um segundo estudo de caso com o objetivo de evidenciar o reuso de elementos da análise. Esta comprovação de reuso deve ser evidenciada através da utilização dos *patterns* e dos Cenários do Domínio.

O segundo estudo de caso endereça o processo de automação de Demanda e Sistemática de Trabalho. O resultado e a mensuração do reuso serão obtidos em função do mapeamento das entidades do modelo resultante, provenientes do catálogo de *patterns* e cenários ou do primeiro estudo de caso.

Dentro da engenharia de *software*, foi selecionada a área de *workflow*, que automatiza processos de negócio, para embasar a aplicação das técnicas propostas nesta dissertação. Pela grande diversidade de conceitos que os sistemas de *workflow* podem suportar, apresentados em [WFM 96] e ilustrados na Figura 5, foi delimitado o escopo do trabalho para procurar atender às seguintes características: processos de negócio, definição de processos, atividades e itens de trabalho. As definições destas características são apresentadas nos capítulos 3 e 4. Desta forma, as demais características de aplicações *workflow* são relevadas, como os sistemas de gerenciamento de *workflow*, instâncias de processos e atividades, atividades manuais e aplicações chamadas.

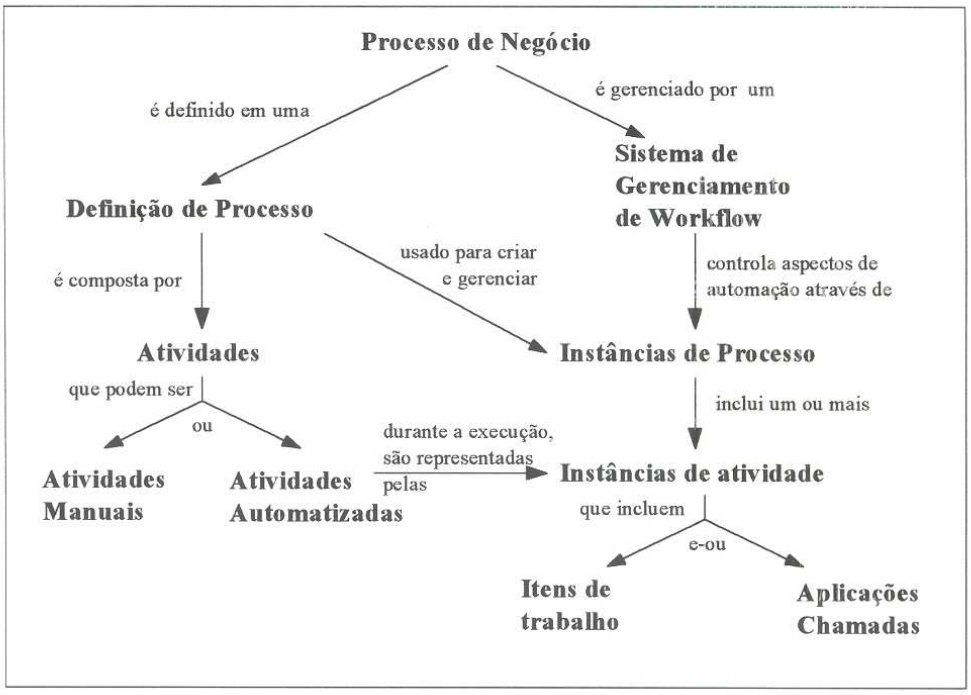


Figura 5 - Relacionamento entre os componentes de um *workflow* [WFM 96].

Nos estudos de caso, a seleção de contextos diferentes do problema, embora dentro do escopo de sistemas *workflow*, é proposital e sua justificativa está em procurar fazer reuso dos componentes da fase de análise entre escopos de aplicação distintos.

Para os dois estudos de caso, a granularidade dos modelos deve oscilar entre as fases de análise e *design* do ciclo de vida do desenvolvimento de um *software* orientado a objetos. Rumbaugh, em [RUM 94], ao fazer referência à modelagem orientada a objetos, estabelece que: "... a separação entre as fases do ciclo de vida é muito menos nítida... o trabalho é conduzido rumo ao refinamento do modelo em níveis progressivamente mais detalhados, em lugar de se converter uma representação em outra".

Considerando-se esta característica de não haver uma separação bem clara entre as fases, os modelos serão elaborados de forma tal que o nível de detalhamento das entidades contemple integralmente a fase de análise e, eventualmente, uma parte da fase de *design*. Ou seja, algumas classes, serviços e atributos poderão ser refinados até atingirem a granularidade próxima à esperada na fase de *design*.

Esta subseção determina a amplitude a ser alcançada por este trabalho, sendo considerado como guia para o desenvolvimento das proposições dos *Patterns* de Processos de Negócio e dos estudos de caso.

1.3 Considerações sobre o capítulo

Em síntese, este capítulo está voltado para a análise da abordagem e dos conceitos utilizados nesta dissertação. Inicialmente é feita uma introdução aos assuntos abordados, juntamente com a forma como são apresentados. Na seqüência, ao definir a justificativa e o problema, são feitas considerações sobre reuso, orientação a objetos, *patterns* e processos de negócio, assim como a interação entre eles. Finalmente, são delimitadas as fronteiras de atuação e apresentada a hipótese que se pretende comprovar.

2 *PATTERNS*

Este capítulo aprofunda os conceitos sobre *patterns* e atua em parceria com o Anexo 1 - Metodologias de *Patterns*, para fornecer embasamento teórico sobre *patterns* e suas metodologias[10].

Inicialmente, na subseção 2.1, é apresentada uma coletânea de definições sobre *framework*, *design pattern* e *patterns*, seguida pela origem dos estudos nessas áreas, na subseção 2.2.

Nesta dissertação são caracterizadas metodologias dirigidas a exemplos aquelas que possuem uma abordagem de aplicação de *patterns* a partir de um catálogo ou repositório. Neste catálogo, os *patterns* são dispostos de maneira organizada e estruturada, a fim de proporcionar a localização de forma fácil e direta.

Já as metodologias dirigidas a processos são aquelas que apresentam os passos a serem seguidos para a concepção e elaboração dos sistemas. Esta abordagem faz uso dos *patterns* conforme predeterminado no processo. Difere, então, da abordagem orientada a exemplos, onde o catálogo de *patterns* é o principal produto da estrutura da metodologia.

Com base nessas abordagens, foram selecionadas as mais disseminadas e referenciadas na literatura. Dentre as dirigidas a exemplos, foram consideradas: *Design pattern* [GAM 94] de Gamma et alii, POSA [BUS 96] de Buschmann et alii e Analisys *Patterns* [FOW 97a] de Fowler. Dentre as metodologias dirigidas a processo: Taligent [TAL 94a, TAL 94b] da empresa Taligent, *Hot spots* [PRE 95] de Pree e Estratégias e *Patterns* [COA 97] de Coad et alii.

Como resultado dessa análise de metodologias, a selecionada foi Estratégias e *Patterns* [COA 97] e os argumentos para tal são apresentados na subseção 2.3.

A metodologia Estratégias e *Patterns* é apresentada na subseção 2.4 com um grau de detalhamento muito superior ao das demais. O seu processo e os seus elementos são

---

[10] O termo metodologia é também utilizado em casos onde método seria o mais adequado, isto para diferenciar do uso que é feito na orientação a objetos, onde método representa uma operação ou serviço.

abordados profundamente pois formam a base das proposições para atender aos processos de negócio.

Ainda resultante dessa avaliação, são apresentadas no Anexo 1 - Metodologias de *Patterns* as demais metodologias avaliadas, através de uma visão geral e as principais características de cada uma.

2.1 Definições

Erich Gamma [11] et alii [GAM 94] inicialmente propõe que *design patterns* capturam soluções que foram desenvolvidas e evoluídas no decorrer do tempo, de uma forma sucinta e facilmente aplicável. Cada *design pattern* sistematicamente nomeia, explica e avalia um subsistema importante e que ocorre diversas vezes, nos projetos orientados a objetos.

Posteriormente, complementam sua definição afirmando que os *design patterns* de seu livro são "descrições de classes e objetos comunicativos, que são customizados para resolver um problema geral de projeto em um contexto particular", ou seja, registram o conhecimento do domínio da aplicação de forma a possibilitar a sua reutilização em outros domínios.

A metodologia Taligent [TAL 94a] define *design patterns* como o meio para identificar, nomear e abstrair os temas comuns dos projetos orientados a objetos. Eles capturam as intenções escondidas do projeto, identificando os objetos, suas interações e como as responsabilidades são distribuídas entre eles gerando, assim, a base de experiência para construção de *software* reutilizável, e agindo como blocos de construção, os quais podem ser utilizados para confecção de projetos mais complexos.

Lajoie [LAJ 94] define *framework* como um conjunto de classes que promovem uma base de soluções para um conjunto de problemas, normalmente construído por especialistas em um domínio particular de negócio e utilizado por leigos. Complementa que os *frameworks* e as suas aplicações derivadas são composições de microarquiteturas[12] enquanto *design patterns* são formas de identificá-las e nomeá-las: "Eles (*design patterns*) representam um mecanismo para expressar como componentes inter-relacionam-se, assim como uma técnica de alto nível para capturar e expressar as experiências do projeto de uma maneira apropriada, objetivando facilitar o reuso".

---

[11] Gamma et alii é comumente referenciado como [GoF], Gang of Four, representando seus quatro autores.
[12] Para [LAJ 94] as microarquiteturas: "codificam o conhecimento do projeto em termos de comportamento e colaboração entre objetos... são um modo de abstrair e reutilizar as experiências do projeto".

Para Campbell et alii [CAM 91]: "Um *framework* é um projeto de arquitetura para sistemas orientados a objetos. Ele descreve os componentes do sistema e a forma que eles interagem. Em *frameworks*, as classes definem os componentes do sistema. As interações no sistema são definidas por restrições, herança, polimorfismo e regras informais de composição".

Ralph Johnson [JOH 88], um dos precursores desses estudos, define: "... Uma coleção de classes abstratas podem ser utilizadas para expressar um projeto abstrato...Um projeto abstrato orientado a objetos, também conhecido como *framework*, consiste em classes abstratas para cada componente principal. As interfaces entre os componentes do projeto são definidas em termos de conjunto de mensagens". Complementando posteriormente sua definição, [JOH 92], afirma que: "Uma classe abstrata é o projeto para um único objeto. Um *framework* é o projeto para um conjunto de objetos que cooperam para alcançar um conjunto de responsabilidades. Então, *frameworks* são projetos de maior escala do que classes abstratas. *Frameworks* promovem uma forma de reuso do conteúdo intelectual de um *software*..."

Em [GAM 94], Gamma e Johnson et alii especificam ainda mais, afirmando que *frameworks* implementam a arquitetura geral do desenvolvimento: o particionamento em classes e objetos, suas responsabilidades, a colaboração recíproca e a fila de controle, tudo para que o projetista possa concentrar seus esforços somente nos aspectos específicos de sua aplicação.

As definições de *framework* e *design pattern* variam conforme as correntes de estudo, mas pode-se evidenciar a essência entre elas, que é a intenção de tornar persistente e disponível não somente as decisões e resultados obtidos em um projeto, como também as razões que os motivaram. Desta forma, aliada aos conceitos básicos de orientação a objetos, pode-se obter reuso do *design* e da análise de sistemas, e não apenas de código e funções.

Sendo *frameworks* e *design patterns* os precursores dessas metodologias que visam reuso, ocorreu uma evolução gradativa para as fases iniciais do processo de desenvolvimento, ou seja, o foco tornou-se a análise e eventualmente até a obtenção de requisitos. Um dos representantes desta linha é Martin Fowler [FOW 97a] através de seus "*analysis patterns*" ou *patterns* de análise. *Pattern*, em seu entendimento é "...uma idéia que já foi útil em um contexto prático e que provavelmente será útil em outros". Enfatiza o termo "idéia" para concordar tanto com [GAM 94] ao fazer referência a "grupo de objetos colaborativos", quanto com outros autores ao referenciar "princípios para organização de projetos"

Peter Coad [COA 97] também enfatiza a importância das fases iniciais do processo de desenvolvimento e, para fazer suas proposições, utiliza uma composição entre estratégias e *patterns*. Define estratégia como "um plano de ação que pretende alcançar um objetivo específico" e *pattern* como "uma estrutura (*template*), digna de emulação". Volta à definição de *pattern* para entendê-lo mais como um plano do que uma implementação, uma estrutura para ser seguida durante uma construção e não uma solução, uma abstração oriunda de observações da realidade e, por fim, um exemplo digno de ser emulado.

Complementa que um *pattern* de um modelo de objetos (*object-model*) é uma estrutura de objetos com responsabilidades e interações estereotipadas. Estas estruturas podem ser consecutivamente aplicadas para construção de modelos de objetos e, juntamente com as estratégias, expressam exemplos de boas práticas e auxiliam na comunicação da experiência com modelagem de objetos.

O objetivo desta subseção é apresentar uma relação ampla das definições de *patterns* e *framework* existentes na literatura. Nesta dissertação, ao fazer referência a *pattern*, será considerada a última definição apresentada, proposta por Peter Coad.

2.2 Origem

A utilização de *design patterns* no processo de desenvolvimento de *software* orientado a objetos tem como origem, na indicação quase unânime dos autores [LAJ 94, GAM 94, JOH 92, PRE 95, BEC 94, TAL 94a], a publicação do arquiteto Christopher Alexander, "A *Pattern Language*", onde discorre sobre sua teoria, aplicando padrões para a construção de prédios, casas, cidades etc. A essência de seu trabalho é apresentada em [GAM 94]: "Cada *pattern* descreve um problema que ocorre constantemente em nosso ambiente, e então descreve o núcleo da solução para aquele problema, de tal forma que você pode utilizar esta solução milhões de vezes novamente, sem fazê-lo da mesma forma duas vezes. Christopher Alexander".

A sua abordagem parte de uma larga escala, explicando como o mundo é subdividido em nações, as nações em regiões, prosseguindo nesta linha até a organização de ruas, casas, cômodos, paredes, materiais etc. Referido trabalho foi concebido para ser utilizado por não-arquitetos na construção de casas e comunidades, pois destina-se a utilização daqueles que não possuem experiência na área, procurando prover soluções aos problemas comuns, como construção de um quarto e sua disposição de janelas.

Gamma et alii [GAM 94] traça o paralelo entre o seu trabalho e o de Alexander, explicando que este utiliza padrões para construção de casas e cidades e o engenheiro de *software* utiliza-os para construção de objetos e interfaces.

Em contrapartida, Fowler, em [FOW 97a], afirma que o próprio trabalho de Gamma et alii teve mais influência em *patterns* de *software* do que o trabalho de Alexander. Acrescenta que três dos quatro autores de [GAM 94] não haviam lido o trabalho de Alexander antes de escrever sobre *design pattern*.

Fowler [FOW 97a], ao citar que "Peter Coad aponta que a notação de *patterns* é usada por muitos escritores em diversas áreas, muitos dos quais considera melhores exemplos do que Alexander", concorda com aqueles que negam o papel principal concedido para Alexander.

A comunidade de *software* está propensa a considerar Alexander como o precursor dos *patterns* em *software*. Entretanto, como alguns autores não concordam com esta posição, não há consenso sobre a origem dos *patterns*.

A origem de *frameworks* é relativamente antiga. Uma das primeiras aplicações a ser largamente utilizada foi a MVC (*Model/View/Controller* - Modelo/Visão/Controle) em 1980, valendo-se de *Smalltalk*-80 para construção de interfaces de usuários, desenvolvida por Krasner e Pope [GAM 94].

Pree, em [PRE 96], aponta dois projetos como os precursores da utilização de *framework*: o projeto "*Simulation*" (simulação), desenvolvido pela equipe norueguesa da linguagem Simula67 e o projeto "*SKETCHPAD*" (prancheta de desenho), um sistema editor de interface gráfica de usuário, desenvolvido por Doug Engelbart, no início dos anos 60.

2.3 Justificativa para Seleção da Metodologia

Na metodologia Estratégias e *Patterns* [COA 97] o reuso de elementos de análise possui grande ênfase, vindo ao encontro do requisito proposto por esta dissertação, qual seja, obter reuso de análise em modelagem de processos de negócio. Este foi um dos fatores determinantes para sua seleção.

Sob o ponto de vista de *patterns*, esta metodologia é uma das mais completas. O fato de contemplar as duas características adotadas como diferenciais é um de seus fatores mais positivos [PAL 98b]. Esta metodologia é dirigida a exemplos, visto que possui um catálogo com trinta e um *patterns* prontos para serem utilizados ou instanciados. Também é dirigida a

processo, pois, através da ordem em que as estratégias são aplicadas, possui uma seqüência proposta de passos a serem seguidos que compõem o processo de desenvolvimento.

Esta estrutura pode ser dirigida a exemplos e a processo simultaneamente, propiciando que sejam extraídos os benefícios de ambas as abordagens, sendo este um fator muito importante para diferenciá-la de outras metodologias que também levam a análise em consideração, tais como apresentadas em [BUS 96] e [FOW 97a].

Outro diferencial desta metodologia é a separação de elementos, dentro do modelo de objetos e dos cenários, por assunto ou componente (*separation of concerns*). Estes componentes são: domínio do problema, interação humana, gerenciamento de dados e interação com sistemas. Como conseqüência, os elementos podem ser visualizados separadamente, facilitando o entendimento e a clareza dos modelos.

Porém, o principal benefício dessa separação é a maior probabilidade de reuso dos elementos. Tomando-se como exemplo um modelo de objetos pronto e implantado, é possível fazer o reuso somente dos elementos do domínio do problema, mesmo que os elementos de interação humana ou gerenciamento de dados não permitam. Desta forma, a gama de opções de reuso é maior do que a dos modelos tradicionais.

A seleção da metodologia Estratégias e *Patterns* foi feita com base nos critérios e nas argumentações anteriormente apresentadas. Este capítulo não tem por objetivo classificar ou pontuar as metodologias de *patterns*, mas sim fornecer a base necessária para a execução deste trabalho, através da utilização e complementação de uma delas, no caso, a que mais se aproximou dos requisitos impostos por esta dissertação.

A partir deste ponto, as referências feitas à metodologia Estratégias e *Patterns* [COA 97] levam a denominação de "metodologia original ", "*patterns* originais" ou "cenários originais". Isto para diferenciar da estrutura de *patterns* proposta no capítulo 3, que terá a denominação de "*patterns* propostos".

2.4 Metodologia Estratégias e *Patterns*

Esta subseção traz, de forma aprofundada, o detalhamento da metodologia Estratégias e *Patterns* [COA 97]. Na subseção anterior é apresentada a argumentação sobre a sua seleção como metodologia base para desenvolvimento de *patterns* que atendam processos de negócio.

Ao abordar a metodologia de desenvolvimento de *software* orientado a objetos, [COA 97] o faz de forma inovadora. De fato, em [COA 91a] e [COA 91b] é feita uma exposição

tradicional dos tópicos a serem abordados durante a análise e o *design* de um sistema orientado a objetos, onde são apresentados os passos a serem seguidos para a confecção de um sistema.

Já em [COA 97], a ênfase recai sobre os exemplos, e com base neles são apresentadas as teorias, as estratégias, os cenários e os *patterns*. Esta composição de elementos gera a metodologia. Cada exemplo apresentado endereça um tipo diferente de domínio do problema e é diferenciado dos demais tanto pela ordem em que os componentes da metodologia são aplicados, quanto pela ênfase dada à alguns componentes especificamente.

São expostos cinco sistemas modelados orientados a objetos, iniciando pela fase de identificação dos requisitos e passando por todas as demais, até o instante anterior à implementação. Os exemplos apresentados representam, na ordem de exposição, uma aplicação para um ponto-de-venda, um almoxarifado, um centro de ordens, e duas aplicações de tempo real, a primeira de distribuição de peças e a segunda essencialmente de controle.

Incrementando o grau de complexidade e de reuso gradativamente, após o estudo dos cinco sistemas, é destinado um capítulo, na publicação original, para desenhar novamente todos os sistemas utilizando-se *patterns* desde o início, onde são feitas comparações entre as formas de desenvolvimento tradicional e em alta velocidade (capítulo 6 "All five applications, at high speed").

Em [COA 97] são apresentadas 148 estratégias e 31 *patterns* no decorrer da construção dos sistemas exemplo, que são provenientes de introspecções feitas em centenas de modelos durante um período de cinco anos. Neste capítulo são relacionadas as principais etapas, diagramas e algumas estratégias e *patterns* constantes desta metodologia. Para facilitar a referência à fonte original, são sempre fornecidos os números das estratégias e dos *patterns* utilizados.

Os componentes apresentados na metodologia são aplicáveis aos diversos tipos de sistemas, como ponto-de-venda, estoque, tempo-real e de controle. A variação proposta está limitada à ordem de aplicação das estratégias, conforme a característica do sistema a ser desenvolvido. Nesta dissertação é feita a exposição das fases propostas para o primeiro e mais abrangente dos sistemas, o ponto-de-venda, entretanto, visando atingir uma abrangência total da metodologia, são acrescentadas características também apresentadas nos demais sistemas.

A estruturação é composta de duas fases: a primeira representando as linhas gerais, o escopo e os principais requisitos do sistema e a segunda a modelagem dos objetos, responsabilidades e interações.

A primeira etapa define a abordagem da metodologia, conforme o tipo do sistema a ser modelado. Esta abordagem é resultante de dois fatores: o primeiro é a ordem em que as atividades são executadas e o segundo é a ordem que os componentes são tratados. Atividades e componentes serão definidos quando citados para utilização.

Para guiar estas decisões, são utilizadas como base as estratégias n.° 1 até n.°1 f, e como forma de ilustração, é apresentada a estratégia n.° 1 [13] na Figura 6.

n.° 1. Estratégia "Quatro Atividades Principais, Quatro Componentes Principais"

• Organize o seu trabalho sobre quatro atividades principais, com quatro componentes principais.

• Quatro atividades principais:

Padrão: Identificar propósitos e requisitos, selecionar objetos, estabelecer responsabilidades, aprimorar a parte dinâmica com cenários.

Variação 1: Pode ser produtivo trabalhar com a parte dinâmica com cenários inicialmente, estabelecendo as responsabilidades no decorrer da atividade. Isto é especialmente indicado para sistemas de tempo real.

Variação 2: Pode ser produtivo selecionar objetos de transações, agregações e plano, após utilizar os *patterns* correspondentes para guiar a seleção de objetos adicionais, estabelecendo responsabilidades na seqüência e por último aprimorar a parte dinâmica com cenários.

• Quatro componentes principais:

Padrão: Domínio do problema, interação humana, gerenciamento de dados e interação com sistemas.

Variação 1: Pode ser produtivo iniciar com a interação humana, seguida do domínio do problema, gerenciamento de dados e interação com sistemas. Isto é especialmente indicado quando os peritos do domínio querem abordar a interação humana desde o início.

Variação 2: Pode ser produtivo iniciar com o domínio do problema e interação com sistemas, seguido de interação humana e gerenciamento de dados. Isto é especialmente indicado em aplicações de tempo real, onde os peritos estão interessados nas aquisições de dados e nos aspectos de controle.

Figura 6 - Estratégia n.° 1.

Seguindo nesta fase, o próximo passo é identificar o propósito do sistema, em uma frase de até 25 palavras (estratégia n.° 2) e os seus requisitos (estratégia n.° 3 a n.° 12). Dentre os

---

[13] As estratégias são apresentadas dentro de molduras com bordas simples, com a numeração, o nome, o tipo e o seu conteúdo.

principais documentos gerados, o glossário, a lista de requisitos e os tipos de requisitos são os mais importantes. Estes tipos são assim caracterizados: guardando *log* das informações importantes, conduzindo o negócio, analisando os resultados do negócio e interagindo com outro sistemas.

Na segunda etapa é feita a modelagem dos objetos. Seguindo-se a ordem padrão, inicia-se a seleção dos objetos, que são organizados seguindo a estratégia n.° 25 em componentes do modelo de objetos, quais sejam:

PD (*Problem-Domain*) Domínio do problema: este componente contém os objetos que correspondem diretamente ao domínio do problema sendo modelado. Estes objetos são neutros com relação à tecnologia e conhecem muito pouco sobre os outros componentes.

HI[14] (*Human-Interaction*) Interação Humana: componente que contém os objetos que provêem a interface entre o domínio do problema e as pessoas e normalmente correspondem a objetos-tela e relatório.

DM (*Data-Management*) Gerenciamento de Dados: objetos que provêem interface entre os objetos do domínio do problema e as bases de dados e gerenciadores de arquivos. Normalmente são objetos do domínio do problema que necessitam persistência de busca.

SI (*System-Interaction*) Interação com Sistemas: objetos que provêem interface entre objetos do PD [15] e outros sistemas ou equipamentos. Estes objetos encapsulam os protocolos de comunicação, liberando os objetos PD dos detalhes de baixo nível e de implementação.

O último componente é o NT (*Not this Time*) "Não neste momento", que representa os objetos considerados que por razões técnicas, políticas, econômicas ou outras foram descartados do modelo atual, podendo, porém, ser lembrados de tempos em tempos.

A Figura 7 representa graficamente todos os componentes do modelo de objetos e suas interações. Os objetivos desta organização são: facilitar e simplificar o modelo do sistema em questão e proporcionar um maior reuso nos futuros modelos. Este reuso pode ser exemplificado pelo desenvolvimento de um novo sistema, onde alguns objetos do domínio do problema são reutilizados mesmo, que os objetos de interação humana sejam totalmente diferentes.

---

[14] O termo HI é mantido em função da sua utilização na metodologia, porém, na engenharia de *software* é mais comumente encontrado HCI (*Human Computer Interaction*).

[15] As referências posteriores serão feitas com as abreviaturas PD, HI, DM, e SI.

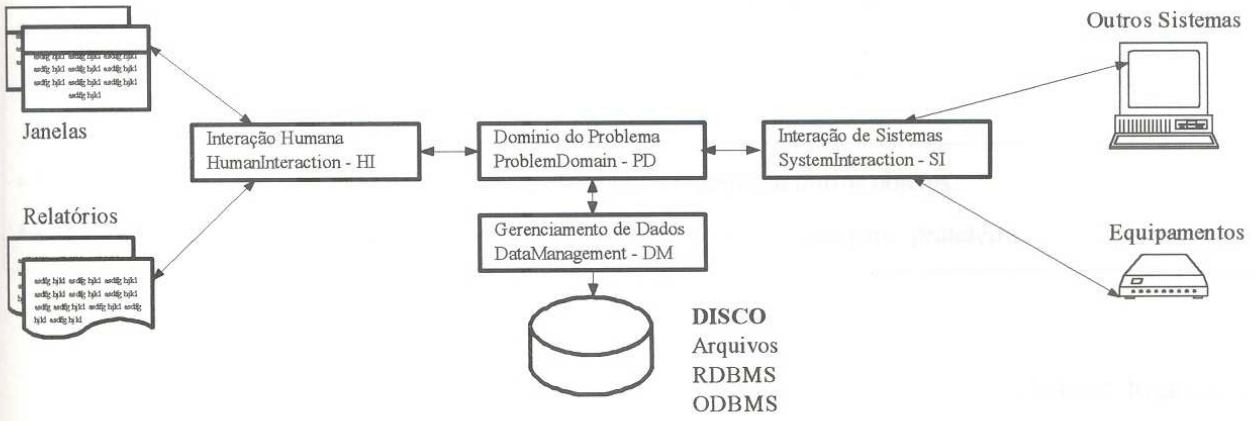


Figura 7 - Componentes do modelo de objetos e interações [COA 97].

Os passos seguintes para cada um dos componentes do modelo de objetos são: selecionar os objetos, definir responsabilidades e aprimorar a parte dinâmica com cenários.

2.4.1 Objetos PD

Selecionar objetos PD:

Para os objetos PD são considerados os seguintes aspectos: atores e participantes, lugares e coisas. Para os demais componentes do modelo de objetos, além destes, serão apresentados novos itens.

Selecionar atores e participantes:

Ator é uma pessoa, uma organização ou outro agente que participa de uma ou mais formas no decorrer do tempo. O participante, por sua vez, atua de uma forma específica, desempenha um papel ou cumpre uma missão específica. São exemplos de atores: pessoas, agências ou corporações. De participantes: clientes, empregados, caixas, estudante ou oficial. Estratégias n.° 13 e n.° 14.

Selecionar lugares:

Um lugar representa um dos objetos PD, e para a sua identificação é aplicada a estratégia n.° 15 que é resumida na Figura 8.

n.º 15 Estratégia "Selecionar Lugares"	selecionar objetos (participantes de *pattern*)
• Procure lugares onde as coisas descansam, lugares que comportam outros objetos. • Exemplos: aeroporto, linha de montagem, banco, clínica, depósito, garagem, prateleira	

Figura 8 - Estratégia n.º 15.

Esta seleção é complementada pela estratégia n.º 22, que procura abstrair lugares dos contentores de outros objetos, como por exemplo, almoxarifado que contém alas, que contém corredores, que contém prateleiras e assim por diante.

Selecionar coisas [16]:

A primeira seleção, nesta fase, refere-se às coisas tangíveis do domínio do problema, que irão compor os objetos deste componente (estratégia n.º 16). Por exemplo: produtos em uma loja, caixa registradora, conta ou cronograma. Outros objetos pertinentes são os abstraídos dos tipos de coisas (estratégia n.º 19), sempre que ocorre de um tipo de objeto implicar em diferentes serviços, especializando a classe e incluindo os novos serviços.

Ainda na seleção dos objetos, transações são consideradas. Entende-se por transação a gravação ou *log* de algum evento significativo, como quando um objeto transação conhece um evento significativo e seus participantes, fazendo cálculos pertinentes ao evento. Exemplos de objetos transação são venda, pagamento, depósito ou acordo.

Assim como para outros objetos, tais como os lugares, pode ser aplicada a estratégia n.º 34 que conduz à seleção de tipos de objetos utilizando a generalização-especialização (*gen-spec*). Como exemplo destes objetos pode-se citar o pagamento, em suas diversas modalidades: dinheiro, cheque ou cartão de crédito.

Por último é aplicada a estratégia n.º 22, já apresentada para lugares e que possui a mesma conotação de abstrair novos objetos oriundos de contentores de coisas.

Organizar os objetos do domínio do problema com *patterns*:

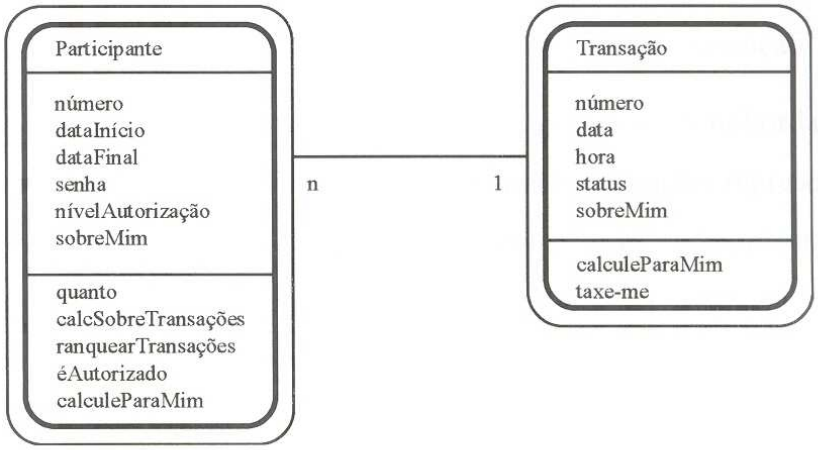
Uma vez selecionados os objetos iniciais do modelo, os *patterns* são utilizados servindo como uma estrutura ou um modelo de objetos com responsabilidades e interações estereotipadas. Estes modelos podem ser repetidamente aplicados por analogia.

---

[16] Things, com a conotação de coisas ou utensílios.

Nesse instante, os *patterns* têm a responsabilidade de auxiliar a conexão entre os objetos dispersos. Para ilustrar seu formato, a Figura 9 apresenta o *pattern* n.° 3 [17], que faz a conexão entre participantes e transações.

n.° 3. *Pattern* "Participante-Transação" — *Pattern* de transação



==========

• Interações típicas entre objetos

quanto->calculeParaMim

calcSobreTransações->calculeParaMim

classificarTransações->taxe-me

• Exemplos

Participantes: agente, aplicador, comprador, caixa, encarregado, cliente, procurador, freguês, negociador, delegado, distribuidor, doador, empregado, investidor, membro, oficial, entre outros citados.

Transações: acordo, designação, autorização, contrato, entrega, depósito, incidente, inquérito, ordem, pagamento, problema, compra, registro, reserva, entre outras citadas.

•Combinações

ator-participante, participante-transação, lugar-transação, item específico-transação, transação-linha de item da transação, transação-transação subseqüente.

Figura 9- *Pattern* 'Participante-Transação'.

A aplicação de um *pattern* em objetos já selecionados forma o início do modelo de objetos. Tomando-se como exemplo um usuário, na condição de participante e um projeto na

[17] Os *patterns* são envolvidos por uma moldura dupla para diferenciar das estratégias, com o número, o título e o tipo do *pattern*. Os tipos podem ser *pattern* de transação, de agregação, de plano e interação.

condição de transação, obtém-se a seguinte conexão entre os objetos, representada na Figura 10.

Figura 10 - Conexão após utilização do *pattern* "Participante-Transação" [PAL 98a].

A notação adotada para os modelos é a seguinte: quadrados com bordas arredondadas e moldura dupla representam classe com objetos; com moldura simples representam classes que não possuem correspondência a objetos diretamente, mais comumente utilizadas nas generalizações-especializações. Na primeira divisão da classe está contido o nome, na segunda estão os atributos e na terceira os métodos ou serviços. O diferencial em relação a outras metodologias é a localização da multiplicidade, que é representada juntamente ao objeto envolvido. Na Figura 10, a multiplicidade n representa que o usuário 'conhece' n projetos, e o número 1 (ou a ausência de um número) no lado do projeto representa que um projeto somente 'conhece' um usuário.

> A principal motivação para essa quebra de paradigma, instituído nas convenções de entidade-relacionamento, foi o questionamento de clientes que argumentavam que as restrições para os objetos estavam 'do outro lado da linha' dificultando o seu entendimento. A posição do autor é que a restrição 'quem eu conheço' pertence à proximidade do objeto que está recebendo esta restrição, tornando as responsabilidades 'quem eu conheço' mais claras e melhor suportando a filosofia de objetos [COA 97]. Esta característica da metodologia não é mantida na presente dissertação. Nos modelos propostos e nos estudos de caso a notação utilizada é a tradicional, oriunda das convenções de entidade-relacionamento.

Estabelecer responsabilidades para os objetos PD:

Consideram-se responsabilidades de um objeto o seu conhecimento sobre si, o seu conhecimento sobre os outros objetos e sua capacidade de fazer, tanto por si próprio quanto em colaboração com outros. Traçando um paralelo com outras metodologias [RUM 94], o conhecimento é traduzido para os atributos das classes e as suas capacidades são traduzidos para os métodos.

Para compor estas informações, impõem-se três perguntas aos objetos: 'O que conheço?', 'Quem conheço?' e 'O que faço?'. Essas para os atributos e esta para os métodos[18].

A responsabilidade 'O que conheço' é contemplada pelas estratégias n.º 50 a n.º 71, 'Quem conheço' n.º 72 a n.º 85 e 'O que faço' n.º 86 a n.º 126.

Desta forma são selecionadas as responsabilidades para cada um dos tipos de objetos selecionados no modelo de objetos:

Responsabilidades para atores e participantes. Estratégias n.º 52, n.º 74,n.º 94

Responsabilidades para lugares. Estratégia n.º 53, n.º 75,n.º 95

Responsabilidades para coisas.

Responsabilidades para contentores.

Responsabilidades para transações. Estratégia n.º 54a, n.º 76, n.º 96

Como exemplo de uma estratégia genérica para obtenção de responsabilidades, é apresentada na Figura 11 a estratégia 'Estabelecer os objetos que conheço'.

n.º 72. Estratégia "Estabelecer os Objetos que Conheço"	estabelecendo responsabilidades/ 'Quem conheço' (fundamental)
• Este é um aspecto de um objeto de *software* sendo criado: "Eu conheço outros objetos, aqueles que são relacionados com o objeto real do qual eu sou uma abstração"	
• Selecione as conexões entre objetos para satisfazer estas duas questões:	
Para diretamente saber "para quem enviar uma mensagem" (dentro de um ou mais cenários).	
Para responder a um questionamento sobre objetos que estão diretamente relacionados a este.	

Figura 11 - Estratégia para definição de responsabilidades.

Como uma forma de complementação das responsabilidades dos objetos, já trabalhadas com as estratégias, aplicam-se novamente os *patterns,* não mais com o intuito de definir as conexões e hierarquias dos objetos, mas para obter novas responsabilidades, pois nas classes dos *patterns* são definidas responsabilidades estereotipadas.

[18] O termo "método" não é utilizado em [COA 97] porém, tendo em vista a extensiva utilização nas literaturas, é eventualmente empregado nesta dissertação.

Valendo-se da Figura 9, algumas responsabilidades podem ser abstraídas para participantes e para transações. Como no exemplo citado anteriormente entre usuário (participante) e projeto (transação), pode-se selecionar alguns atributos para usuário, como número (*number*), senha (*password*), nível de autorização (*authorizationLevel*) e alguns métodos, como 'fornecer projetos vencidos' (*calcOverTransactions*), 'é autorizado' (*isAuthorized*) ou 'informar responsável' (*calcForMe*).

Com a aplicação dos *patterns* encerra-se o primeiro ciclo para seleção e definição de responsabilidades para os objetos PD. O objetivo principal do próximo tópico é definir a parte dinâmica do modelo. Todavia, com a aplicação das estratégias e dos *patterns* novas responsabilidades podem ser acrescentadas.

Aprimorando a parte dinâmica do PD com cenários:

Um cenário é uma seqüência temporalmente ordenada de interações entre objetos. Nele, são representadas as mensagens que os objetos emitem e recebem, e os métodos invocados como em decorrência a essas. Uma visão de um cenário é a representação gráfica das classes envolvidas, das mensagens e dos métodos invocados. As principais motivações para utilização de cenários são:

- Para achar objetos adicionais.
- Para melhor distribuir e refinar as responsabilidades.
- Para melhorar o entendimento da dinâmica do sistema.
- Para verificar se o modelo é completo.
- Para testar um modelo de objetos.

Os cenários a serem desenvolvidos devem ser inicialmente aqueles cujas seqüências de interação de objetos sejam críticas. Um proposto critério de parada para esta fase é o momento em que não se observam objetos ou responsabilidades novas ao se criar novos cenários, momento este em que os cenários já desempenharam o seu papel.

Notadamente, os cenários são mais interessantes quando iniciam com uma interface humana (HI) ou uma interface de sistema (SI), contudo podem ser desenvolvidos para objetos do domínio do problema (PD). O suporte para a parte dinâmica é de responsabilidade das estratégias n.° 127 a n.° 141. A Figura 12 apresenta a estratégia n.° 127 com o intuito de exemplificar a sua utilização e a Figura 13 e Figura 14 apresentam duas formas de visões de cenários. A primeira, simplificada, e a segunda, para o mesmo exemplo, detalhada.

n.º 127. Estratégia "Selecione os Cenários Chaves" Aprimorando a parte dinâmica

- Aprimore e demonstre a satisfação de um requisito do sistema.

Inclua cenários que o fazem trabalhar com as interações necessárias para atender aos requisitos do sistema.

Considere a utilização de subcenários, para facilitar o trabalho com o cenário e o seu entendimento.

- Percorra todo o modelo, examinando se está completo.

Inclua cenários que realmente percorram minuciosamente o seu modelo de objetos; use-os para verificar se o modelo está completo.

- Examine as interações chave dentro do modelo.

Inclua cenários que o deixem investigar a parte dinâmica para serviços importantes no seu modelo.

Figura 12 - Estratégia n.º 127.

No topo da Figura 13 está representado o nome do cenário e suas restrições (dependências, tempos). Neste exemplo, um objeto é solicitado para calcular o seu total, no caso o objeto venda. Para tanto, o serviço "calculaTotal" invoca o serviço "calculaSubtotal" que retorna o subtotal. Após, o "calculaTotal" invoca o "calculaTaxa" que retorna "taxaTotal". Por último o serviço "calculaTotal" retorna o total.

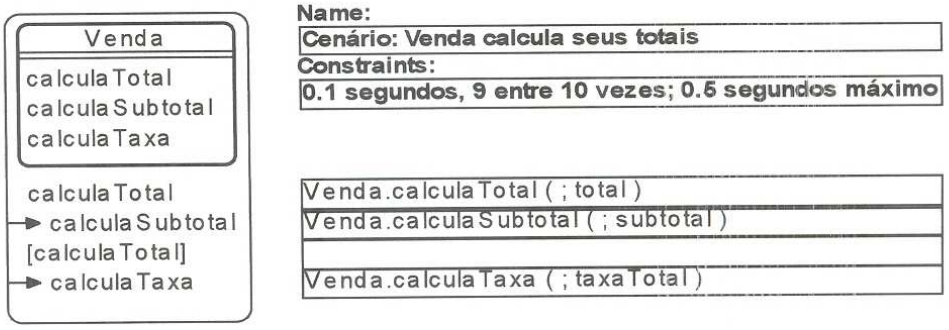


Figura 13 - Cenário do cálculo de total de uma venda – Simples [COA 97].

O primeiro serviço do cenário é colocado imediatamente abaixo da classe na primeira coluna. O símbolo das setas pontilhadas representa o envio das mensagens, podendo o destinatário ser o próprio objeto emitente. Como no fluxo "calculaSubtotal" da Figura 14, pode haver o símbolo 'n' na seta que indica chamadas sucessivas da mesma mensagem. No lado direito do cenário é apresentado um resumo deste, com a seguinte sintaxe: objeto.serviço([argumentos de entrada, ..] ; [argumentos de retorno, ...]).

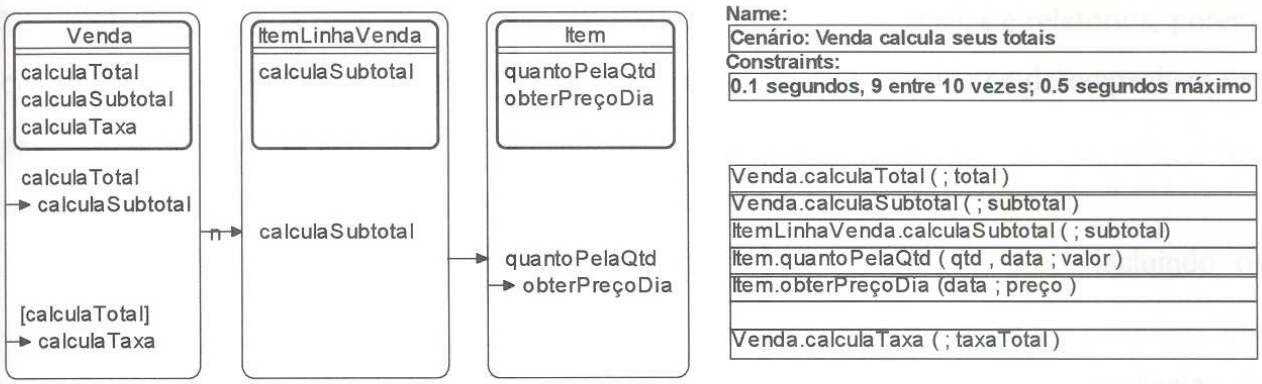


Figura 14- Cenário do cálculo de total de uma venda – Detalhado [COA 97]

Objetivando complementar a ilustração sobre cenários, na Figura 15 é apresentado um cenário que, além das invocações entre objetos e passagem de parâmetros, contém uma estrutura de controle de fluxo, no caso a condição IF-ENDIF.

Figura 15 - Cenário de manutenção de projetos [PAL 98a]

A ênfase maior da aplicação dos cenários recai sobre a fase de análise, embora eventualmente adentre a fase de *design*. Pela sua abrangência, por vezes, é responsável pelo surgimento de novas responsabilidades, ou seja, novas interações, serviços e até mesmo novos objetos para compor o modelo.

2.4.2 Objetos HI

Selecionar objetos HI:

Pela característica de proposição de atividades e não de etapas rígidas, as atividades de objetos do domínio do problema (PD) e interação humana (HI) podem ser alternadas.

Utilizados em sistemas tradicionais, os objetos HI são basicamente janelas e relatórios, porém, quando empregados em sistemas de controle e de tempo real estes objetos desempenham um papel importante e requerem grande atenção.

A correta separação entre objetos PD e HI é essencial por dois aspectos:

- Entendimento: para todos os membros do time de desenvolvimento, incluindo os especialistas do domínio;
- Incremento da probabilidade de reuso: para os objetos do domínio e instâncias de *patterns* que são independentes dos compromissos das interações humanas e aplicações específicas.

Uma forma de separação entre objetos PD e HI é fazer modificações na interface humana e verificar quais objetos são alterados. Como exemplo, em um simulador de vôo, após relacionar os objetos, pode-se inferir quais são alterados ao imaginar que os controles são ativados pela voz e não manualmente. Neste caso, os que sofrerem alteração irão compor os componentes HI e os demais irão compor os componentes PD.

Selecionar janelas:

As estratégias n.º 27 a n.º 30 são referentes às janelas e tratam basicamente dos seguintes tipos: janelas de *logon*; janelas de configurações; janelas 'conduzindo o negócio', que são as transações (e seus participantes); e janelas 'analisando os resultados do negócio', que moldam e sintetizam as informações para este objetivo.

Selecionar relatórios:

Os relatórios são modelados como objetos e a estratégia n.º 31 faz alguma referência a isso, conforme Figura 16.

n.º 31. Estratégia "Selecionar Relatórios"	Selecionar objetos (componentes do modelo)
• Agrupe os sumários chave e saídas especializadas, atendendo às especificações legais e do negócio. • Procure cuidadosamente quem (público) necessita o que (conteúdo) e por quê (propósito) • Não inclua toda procura '*ad hoc*' que alguém possa eventualmente solicitar; não inclua relatórios *batch* desatualizados.	

Figura 16 - Estratégia n.º 31.

Estabelecer responsabilidades para os objetos HI:

Semelhante às responsabilidades dos objetos PD, as dos objetos HI são resultantes das respostas às três perguntas: 'O que conheço?', 'Quem conheço?' e 'O que faço?'. Novamente há uma diferenciação entre sistemas aplicativos normais e sistemas de controle e tempo real. Naqueles, normalmente são analisadas as responsabilidades de janelas e relatórios e nestes todos os objetos de interação humana. As estratégias n.° 57, n.° 79, n.° 79a e n.° 99 servem como base.

Aprimorando a parte dinâmica do HI com cenários:

As proposições e os objetivos para utilização de cenários são os mesmos para objetos PD ou HI, contudo são mais naturais em interações humanas, pois normalmente a maior parte das transações partem de um estímulo humano. Tendência esta não seguida em sistemas de controle e tempo real, onde os estímulos são de origem variada.

As estratégias propostas são as mesmas dos objetos PD, porém são propostas outras estratégias para que sejam feitas considerações e mudanças, numeradas de n.° 132 a n.° 141.

2.4.3 Objetos SI

Selecionar objetos SI:

Os objetos de interação com sistemas (SI) são aqueles que interagem com o sistema proposto, mas são externos, isto é, não o integram. Esta interação pode ocorrer com outro sistema ou com outros equipamentos, como equipamentos de aquisição de dados e de controle.

A sua modelagem possui a particular característica de, para cada interação, ser modelado um objeto como PD, e um objeto SI com o sufixo "SI". Como exemplo, pode haver um objeto PD chamado "SistemaDeAutorização" e outro SI chamado "SistemaDeAutorizaçãoSI", tendo o objeto SI por objetivo encapsular os detalhes da comunicação de baixo nível e o objeto PD participar do modelo independentemente. A Estratégia n.° 33 é responsável por tais proposições.

Estabelecer responsabilidades para os objetos SI:

As responsabilidades dos objetos SI são obtidas da mesma forma que para os objetos PD e HI. A particularidade está na existência de pares PD e SI, como no sistema de autorização citado anteriormente ao selecionar objetos SI. Algumas estratégias tratam dessas

responsabilidades, como as n.° 23, n.° 56 e n.° 98, todavia é importante ressaltar que os objetos SI encapsulam detalhes das interações, como número da porta, estrutura de comunicação etc.

Esta dependência entre os objetos pode dificultar a decisão sobre o local onde inserir algumas funcionalidades, como, por exemplo, a de conversão de dados. Sugere-se a colocação nos objetos SI, deixando os objetos PD somente tratarem do domínio do problema. Outros exemplos de responsabilidades dos objetos SI são as funções de baixo nível e dependentes do equipamento: conectar, fazer requisições e desconectar.

Aprimorando a parte dinâmica do SI com cenários:

As proposições e os objetivos para utilização de cenários são os mesmos já apresentados para os objetos PD e HI. A particularidade é que os pares PD-SI sempre fazem parte dos cenários desta etapa, cada qual com suas responsabilidades e escopo.

2.4.4 Objetos DM

Selecionar objetos DM e estabelecer responsabilidades:

Cada objeto de gerenciamento de dados (DM) possui um correspondente PD, pois seu objetivo é garantir a persistência em meios não voláteis, assim como suprir suas necessidades de armazenamento e busca através de uma coleção de objetos. A estratégia n.° 32, conforme apresentada na Figura 17, faz referência a estes objetos.

n.° 32. Estratégia "Selecionar Objetos de Gerenciamento de Dados(DM)"	Selecionar Objetos (componentes do modelo)
• Adicione um objeto DM para cada classe de objetos PD que você queira que seja persistente - armazenado entre invocações de programas. • Porquê: use objetos de gerenciamento de dados (DM) para encapsular buscas e mecanismos de armazenagem através de todos os objetos dentro de uma classe do PD. • Exemplos: CaixaDM, VendaDM, ItemDeVendaDM, ItemDM	

Figura 17 - Estratégia n.° 32.

Os objetivos da utilização de objetos de gerenciamento de dados (DM) são:

- Os objetos DM sabem como armazenar e recuperar objetos independentemente dos mecanismos de armazenamento: arquivos isolados, relacionais, orientados a objetos.
- Isolam a complexidade do gerenciamento de dados do resto da aplicação.

- Isolam a transição entre um ambiente orientado a objetos e um ambiente não orientado a objetos.
- Nos casos necessários, os objetos DM podem desempenhar a função de 'cache' quando os sistemas interativos estão desconectados.

As responsabilidades dos objetos DM são particulares, resumidas às conexões com o seu par do domínio do problema e com os serviços básicos: procurar, salvar e carregar, conforme as estratégias n.° 80 e n.° 100.

Em alguns tipos particulares de aplicações, onde os objetos são configurados no início da execução, como em alguns sistemas de controle, podem não existir objetos DM.

Aprimorando a parte dinâmica do DM com cenários:

Inicia-se a construção dos cenários com um componente da interação humana, como gravar ou carregar, ou com as necessidades de busca de informações dos componentes do domínio do problema (PD). Um exemplo de um cenário para um objeto DM pode ser "obter item através do UPC (*Universal Product Code*)", representado na Figura 18.

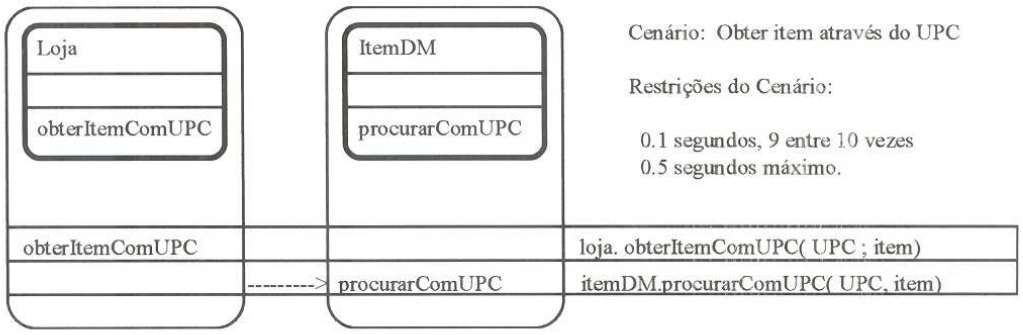


Figura 18 - Cenário de objetos DM.

Após cumpridas todas as etapas acima, o resultado obtido é o modelo de objetos completo, com todos os seus objetos, os seus relacionamentos e as multiplicidades, assim como suas responsabilidades, representadas pelos atributos e serviços.

Além do modelo de objetos, outros produtos resultantes são os cenários, que desempenham um papel importante para a representação da parte dinâmica do sistema. Em conjunto com o modelo de objetos, formam a base conceitual dos elementos da análise do sistema.

Conforme a granularidade dos diagramas, dos atributos e dos serviços, a fase de *design* pode ter sido parcialmente contemplada, isto é, a metodologia afirma que aplicando as suas estratégias e *patterns*, a fase de análise será cumprida. Entretanto, quanto mais detalhados forem os objetos, serviços e atributos, mais se adentra ao *design* e mais próximo fica o projeto da implementação.

De forma resumida, o foco principal da metodologia é a modelagem de aplicativos a partir de *patterns* e estratégias. Além da integridade e completitude destes modelos, o principal benefício obtido é a capacidade de promover reuso, parcial ou integral, dos elementos da fase análise.

2.5 Considerações sobre o capítulo

Este capítulo conceitua *patterns* com o objetivo de fornecer embasamento teórico sobre estes e sobre suas metodologias.

Inicialmente são apresentadas as definições e as origens dos estudos sobre esse assunto, seguidas pela apresentação da justificativa pela seleção da metodologia Estratégias e *Patterns* [COA 97], juntamente com o detalhamento do seu processo e dos seus elementos.

Uma visão geral das metodologias de *patterns* consideradas para avaliação são apresentadas no Anexo 1 - Metodologias de *Patterns*.

3 *PATTERNS* APLICADOS A PROCESSOS DE NEGÓCIO

Neste capítulo, são propostas alterações e complementações à metodologia selecionada, apresentada em [COA 97]. Desta forma, pretende-se fazer com que a metodologia resultante suporte a modelagem de processos de negócio de maneira adequada. Além disso, deve permitir capturar experiências de análise, evoluir os *patterns* e os cenários e promover reuso destes em futuros desenvolvimentos.

Para tal, inicialmente é proposta uma nova estrutura de documentação na subseção 3.1, sobre a qual serão desenvolvidos seis *patterns* específicos para atender a processos de negócio, compondo a subseção 3.2.

Juntamente com esses *Patterns* de Processos de Negócio, propõe-se que atuem os modelos particulares do domínio. Estes conceitos são introduzidos na subseção 3.3. Já a integração desses elementos é feita na subseção 3.4, onde é apresentada a adequação do roteiro e das fases para aplicação das estruturas de *patterns* e dos diagramas. A última subseção faz referência aos diagramas que utilizam a notação UML (*Unified Modeling Language*), amplamente referenciados nas literaturas de *patterns*.

3.1 Proposição de Estrutura para Documentação de *Patterns*

Para que se tenha uma reutilização efetiva das soluções propostas pelos *patterns*, eles devem ser documentados e registrados de uma maneira uniforme. Existem basicamente três tipos de coleções de *patterns* [FER 98]: (i) os catálogos de *patterns* [GAM 94], que apresentam um conjunto de *patterns* em um formato uniforme, independente de domínio, e que não possuem um relacionamento forte; (ii) os sistemas de *patterns* [BUS 96], que apresentam uma coleção de *patterns* relacionados, que ajudam a desenvolver sistemas com propriedades particulares; (iii) as linguagens de *patterns* [MAR 98], que formam um conjunto fechado de *patterns* fortemente relacionados, ligados por regras bem definidas, que resolvem todos os aspectos relacionados a um domínio de problema.

Em todos os tipos de representação de *patterns*, há uma grande preocupação em descrever muito bem cada um dos *patterns* e o contexto onde são aplicados. Inclusive apontando eventuais similaridades ou combinações com outros *patterns*.

Conforme visto na subseção 2.4, os *patterns* originais da metodologia ora empregada são documentados de forma simples, com pouca informação textual sobre onde e como aplicá-los. Os elementos originais da documentação são: o nome do *pattern*, as duas classes envolvidas com serviços e atributos estereotipados, as interações típicas entre os objetos, exemplos, combinações e notas.

Para propor a complementação dessa documentação, foram consideradas as seguintes estruturas para *patterns*, com as respectivas referências:

1) *Design pattern* de Gamma et alii [GAM 94]: intenção, 'também conhecido como', motivação, aplicabilidade, estrutura, participantes, colaborações, conseqüências, implementação, código exemplo, usos conhecidos e *patterns* relacionados;

2) POSA [BUS 96]: exemplo, contexto, problema, solução, estrutura, parte dinâmica, implementação, exemplo resolvido;

3) Taligent [TAL 94a]: precondições, problema, restrições e soluções;

4) John McGregor [MCG 96]: problema, contexto, forças, solução, contexto resultante e *patterns* relacionados.

Com base nessas estruturas de documentação de *patterns*, são propostos novos itens para a metodologia original, principalmente complementando a parte descritiva e textual, porém, sem acrescentar muita complexidade, visto que a facilidade de criação e manipulação, aliados à simplicidade, são fatores de sucesso para os *patterns*.

Devido à necessidade de complementação da metodologia original, na Figura 19 é apresentada a proposta de uma nova estruturação. Para cada item é ressaltada a indicação sobre tratar-se de item original, alterado ou proposto, assim como uma breve descrição do seu conteúdo e a forma de utilizá-lo.

Tal estrutura é empregada para compor a documentação do catálogo inicial de *Patterns* de Processos de Negócio, apresentado na próxima subseção.

- Nome do *pattern* (original): o nome do *pattern* é derivado dos nomes das classes que o compõem. Por exemplo, o *pattern* denominado Participante-Transação é composto pela classe Participante e pela classe Transação.
- Problema (proposto): descreve o problema que ocorre repetidamente em um contexto. Neste item, podem ser citadas algumas forças envolvidas na aplicação do *pattern*. As forças representam requisitos, restrições ou propriedades desejadas.
- Contexto (alterado): descreve o ambiente no qual o problema existe. Pode ser considerado como precondição para a aplicação do *pattern*.
- Solução (proposto): representa a ação tomada para resolver o problema no contexto. É composta por três subdivisões:
- Modelo de objetos (original): diagrama com as classes envolvidas (normalmente duas) e com o relacionamento proposto. Internamente às classes, devem constar os serviços e os atributos estereotipados, que irão nortear a instanciação destas classes.
- Interação típica de objetos (original): enumera as interações entre as classes, através dos serviços que possuem e que são relacionados.
- Exemplos (original): apresenta exemplos de ocorrência de cada uma das classes. Para o *pattern* Participante-Transação, são apresentados exemplos de possíveis participantes (agente, cliente, comprador) e possíveis transações (contrato, pagamento, compra).
- Combinações (alterado): originalmente, as combinações representam as demais classes que, em outros *patterns*, relacionam-se com aquele que se está operando. A justificativa para a existência deste item, na metodologia original, é para prover independência do *pattern* de qualquer outro diagrama ou documento. Como a subseção 3.3.1 propõe a criação e uso de um modelo de objetos dos *patterns* para cada domínio, a necessidade deste item é discutível, pois através do Modelo de Objetos do Domínio tais informações podem facilmente ser extraídas. Optou-se por enumerar este item e deixá-lo a critério do desenvolvedor do *pattern* o seu preenchimento.
- Notas (original): espaço reservado para observações gerais.
- Referência aos Modelos de Objetos do Domínio (proposto): descreve a forma de localizar os Modelos de Objetos do Domínio relacionados com o *pattern*, caso existam, preferencialmente através do nome ou do código do modelo.
- Referência aos Cenários do Domínio (proposto): forma de localizar os Cenários do Domínio relacionados com o *pattern*, caso existam. Preferencialmente através do nome ou do código do cenário.

Figura 19 – Proposta de documentação de *patterns*.

Através dessa nova estrutura de documentação, propõe-se que existam, pelo menos, dois níveis de abstração para os *patterns*. O primeiro nível é representado pelo catálogo original da

metodologia, mantendo-se inalterada sua estrutura simplificada de documentação. Este nível é mais conceitual e independente do domínio de aplicação.

O segundo nível é representado pelos *patterns* específicos para cada domínio, possuindo uma estrutura de documentação mais completa, seguindo a forma proposta na Figura 19. Neste nível, a abstração dos *patterns* está próxima aos processos que endereçam, tornando-os, desta forma, mais dependentes do domínio onde são aplicados.

Essa proposição de hierarquia de *patterns* de acordo com a granularidade e abstração, possibilita a existência de *patterns* mais especializados em determinados problemas, conforme a necessidade e disponibilidade. Quanto mais níveis existirem e quanto mais detalhados forem os *patterns*, mais próximos estarão das fases de *design* e implementação.

Como o objetivo desta dissertação é enfatizar a fase de análise, foi criado somente um nível de *patterns* além do proposto pela metodologia original, que é apresentado na próxima subseção. Tal profundidade foi decorrente da delimitação proposta para este trabalho, contudo ressalta-se que não há limitação para estes níveis de abstração, senão os impostos pelos processos de negócio sendo modelados.

3.2 Catálogo de *Patterns* de Processos de Negócio

Conforme a estrutura de documentação proposta na subseção 3.1, são apresentados seis *Patterns* de Processos de Negócio. A nomenclatura e a abrangência desses *patterns* seguem a filosofia da metodologia original [COA 97], que determina o escopo do *pattern*, conforme as classes que o compõe, normalmente aos pares.

Para definir o escopo do catálogo de *patterns* propostos, faz-se uma analogia com a citação de Fowler, em [FOW 97a]: "Um *framework* efetivo não deve ser muito complexo ou muito grande. Este livro não tenta definir *frameworks* para várias indústrias. Este livro descreve maneiras alternativas de modelar uma situação". Da mesma forma, as proposições de *patterns* desta dissertação limitam-se a um domínio específico, qual seja, a automação de processos de negócio que tratam da criação, aprovação e efetivação de solicitações de atividades.

Em síntese, *Patterns* de Processos de Negócio são *patterns* que possuem abrangência delimitada, são documentados de forma padronizada e endereçam problemas específicos no contexto de processo de negócio.

3.2.1 *Pattern* de Processo de Negócio n.º 1 – Solicitante–Solicitação

Nome do *pattern*: Solicitante–Solicitação

Problema:

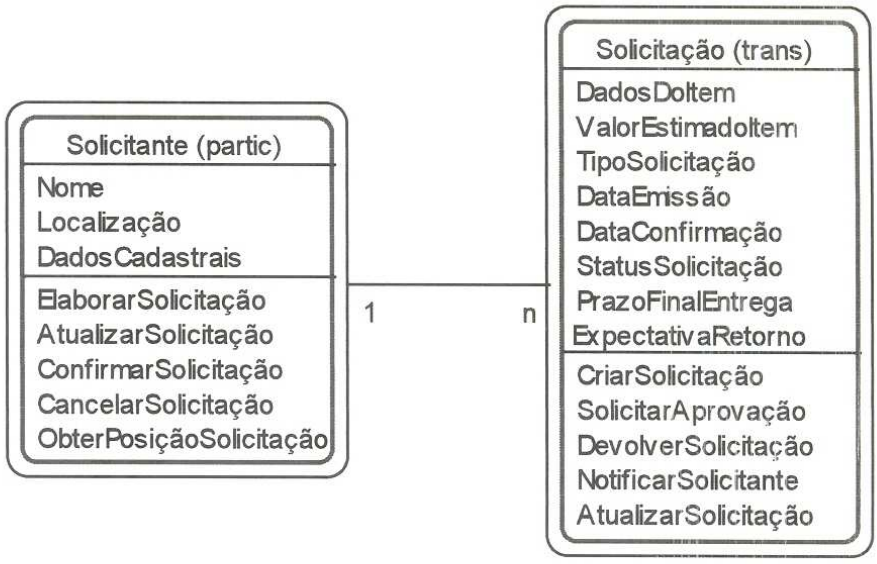
Este *pattern* deve ser aplicado nos processos de negócio envolvendo um solicitante de uma atividade e a solicitação desta atividade. Exemplos de atividades são: um orçamento, uma ordem de serviço, um anteprojeto (passível de aprovação), a criação de um produto etc.

O solicitante da atividade participa na elaboração da solicitação com as informações necessárias para o seu cadastramento inicial. Esta solicitação segue a rota determinada no processo, podendo passar por etapas como aprovação, retorno e efetivação da atividade.

Contexto: *workflow* de confecção de uma solicitação de atividade.

Solução:

Modelo de objetos:



Interação de objetos:

ConfirmarSolicitação –> CriarSolicitação

AtualizarSolicitação –> AtualizarSolicitação

CancelarSolicitação <– DevolverSolicitação

ObterPosiçãoSolicitação <– NotificarSolicitante

Nome do *pattern*: Solicitante–Solicitação (continuação)

Exemplos:

Solicitante: Funcionário, Cliente interno ou externo, Analista, Auditor ou qualquer participante que tenha necessidade de criação de uma solicitação e poder para tal.

Solicitação: Solicitação de compras, Solicitação de serviços, Ordem de produção, Anteprojeto (passível de aprovação), Orçamento, Balancete.

Combinações:

Administrador de Recursos-Solicitação; Aprovador-Solicitação; Desenvolvedor-Solicitação; Solicitação-Item de Linha Solicitação; Agente Coordenador-Solicitação

Notas:

Para aplicar a interação Solicitação-Item de Linha Solicitação, fazer referência ao *pattern* original n.° 6 Transação-Item de linha Transação.

Referência aos Modelos de Objeto do Domínio:

Modelo de Objetos do Domínio - Visão Geral e Completo

Referência aos Cenários do Domínio:

Cenário do Domínio - Aprovação de Serviço

3.2.2 *Pattern* de Processo de Negócio n.º 2 – Aprovador–Solicitação

Nome do *pattern*: Aprovador–Solicitação

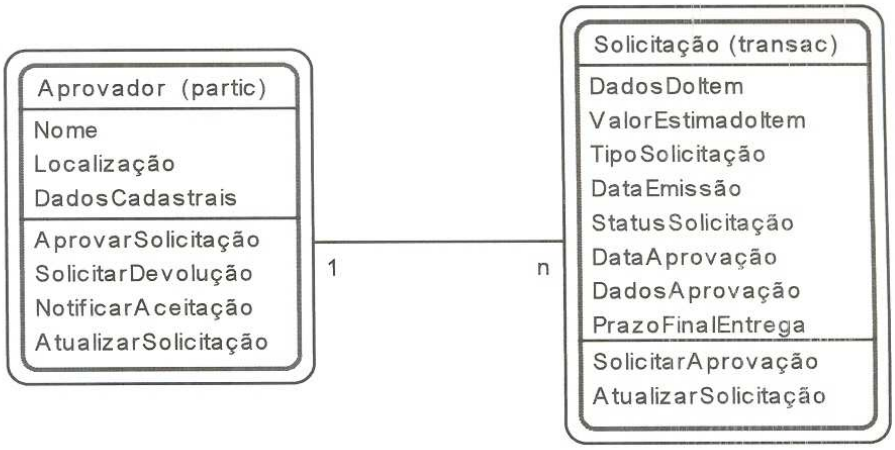
Problema:

Este *pattern* deve ser aplicado nos processos de negócio envolvendo solicitações de atividades com aprovadores. Os aprovadores são responsáveis pela análise e pelo andamento das solicitações de atividades. Conforme as regras de negócio, o aprovador pode optar por permitir o andamento, cancelar ou aprovar com restrições cada solicitação de atividade.

Contexto: *workflow* de aprovação de uma solicitação de atividade.

Solução:

Modelo de objetos:



Interação de objetos:

AprovarSolicitação <– SolicitarAprovação

AprovarSolicitação –>AtualizarSolicitação

SolicitarDevolução ou NotificarAceitação –>AtualizarSolicitação

AtualizarSolicitação –>AtualizarSolicitação

Exemplos:

Aprovador: Gerente, Encarregado, Chefe, Aprovador de compras, Usuário, Cliente final.

Solicitação: Solicitação de compras, Solicitação de serviços, Ordem de produção, Anteprojeto (passível de aprovação), Orçamento, Balancete.

Nome do *pattern*: Aprovador–Solicitação (continuação)

Combinações:

Solicitante-Solicitação; Administrador de Recursos-Solicitação; Desenvolvedor-Solicitação; Solicitação-Item de Linha Solicitação; Agente Coordenador-Solicitação

Notas:

Referência aos Modelos de Objeto do Domínio:

Modelo de Objetos do Domínio - Visão Geral e Completo

Referência aos Cenários do Domínio:

Cenário do Domínio - Aprovação de Serviço

3.2.3 *Pattern* de Processo de Negócio n.º 3 - Administrador de Recursos-Desenvolvedor

Nome do *pattern*: Administrador de Recursos-Desenvolvedor

Problema:

Este *pattern* deve ser aplicado nos processos de negócio que envolvem um administrador de recursos e um ou mais desenvolvedores de atividades. Exemplos de atividades são: um orçamento, uma ordem de serviço, um anteprojeto (passível de aprovação), a criação de um produto etc.

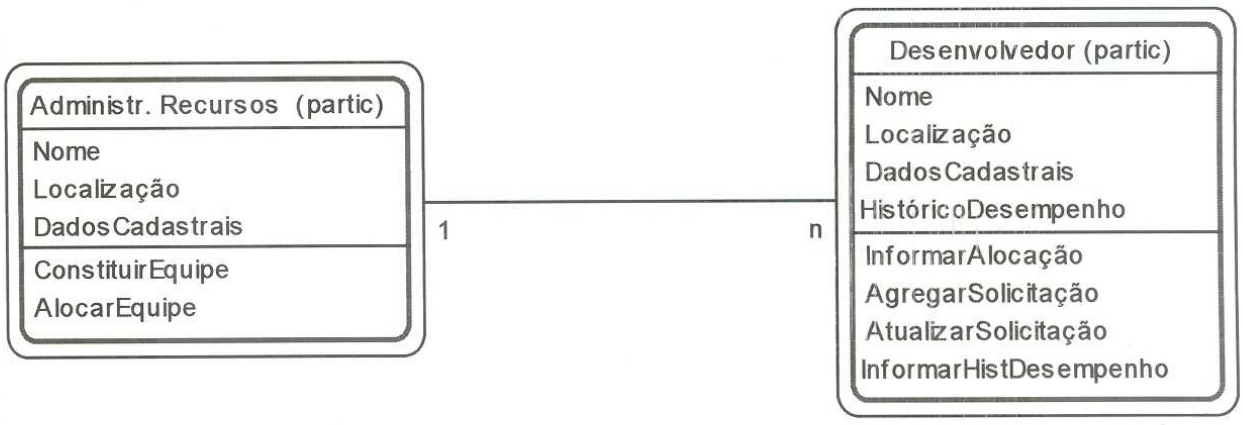
O administrador de recursos atua como um líder de equipe ou gerente de projeto. Ele obtém as informações sobre a alocação de desenvolvedores e constitui equipes de desenvolvedores, quando necessário.

O desenvolvedor é somente responsável por elaborar ou desempenhar a atividade solicitada, sem preocupação com a sua aprovação inicial e o seu encaminhamento posterior. Faz a verificação das atividades pendentes e registra acompanhamento das atividades ativas.

Contexto: *workflow* de formação de equipe em processos de negócio.

Solução:

Modelo de objetos:



Interação de objetos:

ConstituirEquipe –> InformarAlocação

ConstituirEquipe –> InformarHistDesempenho

AlocarEquipe –> AgregarSolicitação

Nome do *pattern*: Administrador de Recursos-Desenvolvedor (continuação)

Exemplos:

Administrador de Recursos: Gerente de projeto, Líder de equipe, Gerente de compras, Comitê, Coordenador.

Desenvolvedor: Prestador de Serviços, Comprador, Vendedor, Analista de Sistemas, Programador ou qualquer participante que elabore ou desempenhe a atividade solicitada.

Combinações:

Solicitação-Administrador de Recursos; Solicitação-Desenvolvedor

Notas:

Com referência à formação de equipes, times ou grupos de pessoas para atuarem em conjunto, na metodologia original há o *pattern* n.°16 denominado *Group-Member*, ou Grupo-Membro, porém a sua composição de atributos e serviços é muito semelhante aos *patterns* n.°14 *Container-Content* (Contentor-Conteúdo) e n.° 6 *Transaction-TransactionLineItem* (Transação-ItemDeLinhaTransação). Estes *patterns* como são apresentados, individualmente, não agregam informações suficientes para a modelagem de processos de negócio.

Referência aos Modelos de Objeto do Domínio:

Modelo de Objetos do Domínio - Visão Geral e Completo

Referência aos Cenários do Domínio:

Cenário do Domínio - Efetivação do Processo de Negócio

3.2.4 *Pattern* de Processo de Negócio n.º 4 - Desenvolvedor-Solicitação

Nome do *pattern*: Desenvolvedor-Solicitação

Problema:

Este *pattern* deve ser aplicado nos processos de negócio que envolvem uma solicitação de atividades e os desenvolvedores destas atividades. Exemplos de atividades são: orçamentos, uma ordem de serviço, um anteprojeto (passível de aprovação), a criação de um produto etc.

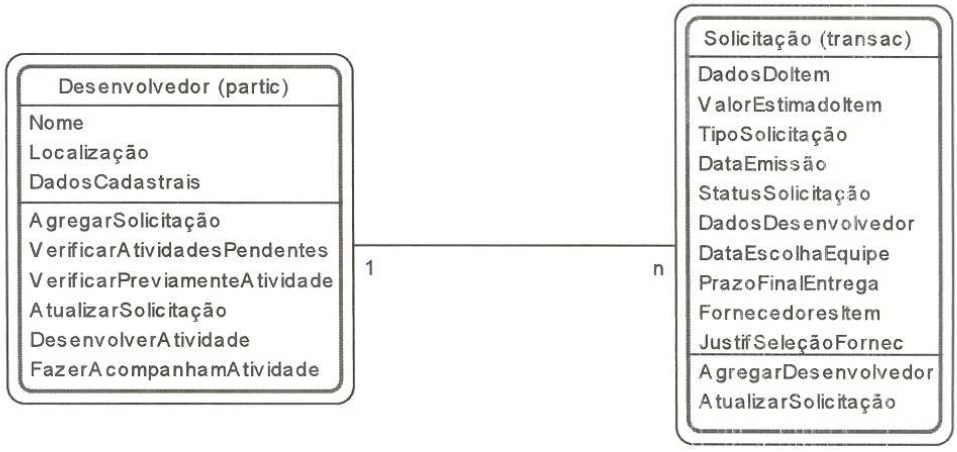
O desenvolvedor é somente responsável por elaborar ou desempenhar a atividade solicitada, sem preocupação com a sua aprovação inicial e o seu encaminhamento posterior.

O relacionamento entre o desenvolvedor e a solicitação é predeterminado.

Contexto: *workflow* de efetivação de uma atividade de processo de negócio

Solução:

Modelo de objetos:



Interação de objetos:

AtualizarSolicitação –> AtualizarSolicitação

AgregarSolicitação –> AgregarDesenvolvedor

FazerAcompanhamento –> AtualizarSolicitação

DesenvolverAtividade –> AtualizarSolicitação

Nome do *pattern*: Desenvolvedor-Solicitação (continuação)

Exemplos:

Desenvolvedor: Prestador de Serviços, Comprador, Vendedor, Analista de Sistemas, Programador ou qualquer participante que elabore ou desempenhe a atividade solicitada.

Solicitação: Solicitação de compras, Solicitação de serviços, Ordem de produção, Anteprojeto (passível de aprovação), Orçamento, Balancete.

Combinações:

Solicitante-Solicitação; Administrador de Recursos-Solicitação; Aprovador-Solicitação; Solicitação-Item de Linha Solicitação; Agente Coordenador-Solicitação

Notas:

Referência aos Modelos de Objeto do Domínio:

Modelo de Objetos do Domínio - Visão Geral e Completo

Referência aos Cenários do Domínio:

Cenário do Domínio - Efetivação do Processo de Negócio

3.2.5 *Pattern* de Processo de Negócio n.º 5 - Aprovador-Aprovador Subseqüente

Nome do *pattern*: Aprovador – Aprovador Subseqüente

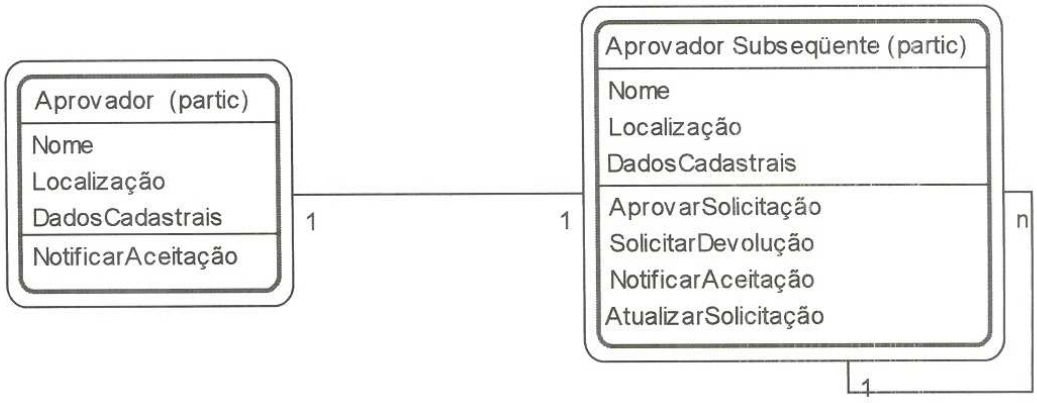
Problema:

Este *pattern* deve ser aplicado nos processos de negócio envolvendo uma seqüência de aprovações de solicitações de atividades. Em parceria com o *Pattern* de Processo de Negócio n.º 2 Aprovador-Solicitação, faz o encadeamento e controla o fluxo das aprovações. Conforme as regras de negócio, o aprovador pode optar por permitir o andamento (ou encaminhar a outros aprovadores), cancelar ou aprovar com restrições cada solicitação de atividade.

Contexto: *workflow* de aprovações encadeadas de uma solicitação de atividade.

Solução:

Modelo de objetos:



Interação de objetos:

NotificarAceitação –> AprovarSolicitação

Exemplos:

Aprovador, Aprovador Subseqüente: Gerente, Encarregado, Chefe, Aprovador de compras, Usuário, Cliente final.

Combinações:

Aprovador-Solicitação;

Nome do *pattern*: Aprovador – Aprovador Subseqüente (continuação)

Notas:

Este *pattern* é baseado no *pattern* n.° 7 da metodologia original, Transação-Transação Subseqüente, onde duas classes representam uma seqüência de transações. Neste caso, a seqüência é de aprovadores.

De forma alternativa, o modelo de objetos pode ser representado com somente uma classe que possui relacionamento com ela própria, atuando ao mesmo tempo como aprovador e aprovador-subseqüente.

Referência aos Modelos de Objeto do Domínio:

Modelo de Objetos do Domínio - Visão Geral e Completo

Referência aos Cenários do Domínio:

Não há.

O Aprovador-Subseqüente pode ser anexado ao Cenário do Domínio - Aprovação de Serviço, após a classe aprovador.

3.2.6 *Pattern* de Processo de Negócio n.º 6 - Agente Coordenador-Solicitação

Nome do *pattern*: Agente Coordenador[19]-Solicitação

Problema:

Este *pattern* possui uma característica voltada para a plataforma sobre a qual os *patterns* de negócio são implementados. Pode ser utilizado nas aplicações onde o *workflow*, que implementa as definições dos processos de negócio, é centrado no processo[20]. Na maior parte das arquiteturas centradas em processo, a comunicação e a coordenação é desempenhada por um agente coordenador.

Este agente é responsável pela troca de mensagens entre as entidades, de acordo com seu conteúdo. Outra atribuição é o roteamento do fluxo das informações e atividades.

A utilização deste *pattern* deve ser criteriosa, pois a inserção de um agente coordenador pode tornar o modelo dependente da plataforma de implementação. A sua modelagem traz, como fator positivo, a maior proximidade do modelo à realidade de implementação, auxiliando as fases de *design* e implementação. Porém, sob a ótica da análise, não é recomendada sua utilização, devido a característica dos modelos de análise, que são mais puros e independentes de implementação, ou seja, mais próximos da realidade do negócio.

O uso deste *pattern* deve ser feito com a consciência das suas implicações. Caso não haja esta convicção por parte do projetista, a modelagem da comunicação e da coordenação deve ser feita sem considerar o agente coordenador.

Contexto: plataforma de implementação de *workflow*, na qual uma solicitação de atividade interage com outros participantes.

---

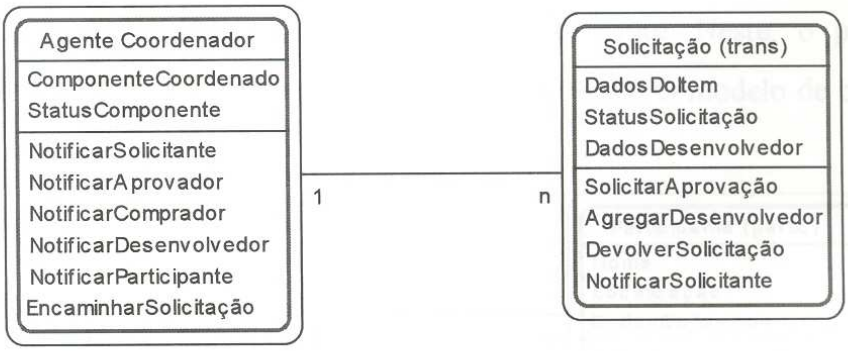
[19] Nesta dissertação, o agente coordenador foi apresentado somente com o objetivo de ilustrar a sua existência, não tendo participação efetiva nos modelos.

[20] Conforme apresentado na subseção 1.1.3

Nome do *pattern*: Agente Coordenador-Solicitação (continuação)

Solução:

Modelo de objetos:



Interação de objetos:

NotificarParticipante (Solicitante) <– NotificarSolicitante

NotificarParticipante(Desenvolvedor) <– AgregarDesenvolvedor

EncaminharSolicitação <– DevolverSolicitação

Exemplos:

AgenteCoordenador: máquina de *workflow* para automatização de processos de negócio.

Solicitação: Solicitação de compras, Solicitação de serviços, Ordem de produção, Anteprojeto (passível de aprovação), Orçamento, Balancete.

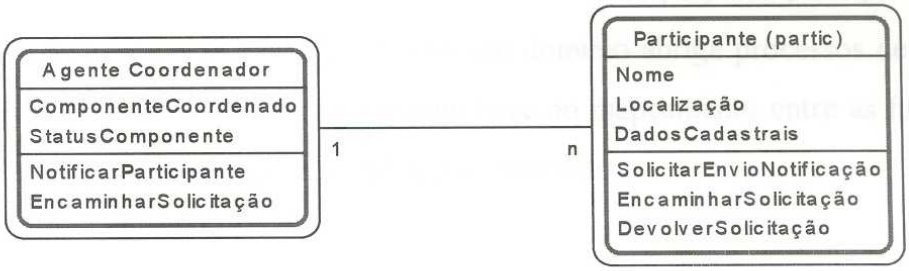
Combinações:

Notas:

O serviço NotificarParticipantes representa, genericamente, cada serviço de notificação de cada um dos participantes, como exemplo NotificarSolicitante, NotificarAprovador etc.

Nome do *pattern*: Agente Coordenador-Solicitação (continuação)

Para exemplificar a criação de novos *patterns* que tenham relacionamento com o agente coordenador e, conseqüentemente, considerem a plataforma de implementação em suas estruturas, é apresentado o Agente Coordenador-Participante. Neste, o participante atua conforme a determinação (comprador, cliente, aprovador etc.). O modelo de objetos para este exemplo é o seguinte:



Referência aos Modelos de Objeto do Domínio:

Não há.

Referência aos Cenários do Domínio:

Não há.

3.3 Estruturas Particulares do Domínio de Processos de Negócio

Como forma de particionar a grande abrangência dos processos de negócio é feita uma proposta de abordagem por domínios. As proposições feitas neste capítulo, através dos *patterns* da subseção 3.2 e dos diagramas desta subseção, referem-se a um domínio específico.

A forma de identificar o domínio adequado para um processo de negócio se dá através de uma análise e levantamento, em um alto nível de abstração, das principais atividades e funcionalidades definidas pelo processo. Como um domínio abriga processos de negócio com funcionalidades similares, a seleção ocorre com base no mapeamento entre as funcionalidades oferecidas pelos domínios com as requeridas pelo processo.

Dessa forma, o processo é designado para o domínio que suprir a maior parte dos seus requisitos. Vale ressaltar que, em não havendo um domínio que satisfaça parcialmente os requisitos, novos domínios podem ser definidos. Nesta nova definição de domínio, uma relação das principais atividades e funcionalidades deve igualmente ser criada, viabilizando futuras consultas.

O domínio endereçado por esta dissertação é a automação de um processo de negócio que trata da criação, aprovação e efetivação de solicitações de atividade. O detalhamento deste *workflow* pode ser extraído dos *patterns* apresentados anteriormente na subseção 3.2, porém, considerando-se que estes não são suficientes para guiar o projetista a desenvolver modelos completos de *workflow*, são propostos novos elementos e diagramas para esta modelagem, descritos na seqüência.

3.3.1 Proposição dos Modelos de Objetos do Domínio

O primeiro elemento, denominado Modelo de Objetos do Domínio, é originado do agrupamento dos *Patterns* de Processos de Negócio que possuem relacionamentos entre si, ou fazem parte de um mesmo contexto.

O resultado é um diagrama com todas as classes, seus relacionamentos e suas multiplicidades. A característica deste diagrama é proporcionar uma visão geral dos *patterns* que endereçam um domínio específico. Este diagrama existe na metodologia original, porém é utilizado somente como forma de apresentação dos relacionamentos entre os *patterns*, não sendo referenciado como um diagrama a ser utilizado durante a modelagem dos sistemas.

Propõe-se fazer uso deste diagrama de forma mais extensiva e incluí-lo na documentação da análise dos aplicativos. A sua denominação, para esta dissertação, é Modelo de Objetos do Domínio – Visão Geral.

Como forma de complementação dessa documentação, propõe-se a criação de um novo diagrama, originário também dos *patterns*, porém contendo todos os serviços e os atributos estereotipados, denominado Modelo de Objetos do Domínio – Completo.

O principal objetivo da criação destes diagramas é auxiliar o projetista na fase da análise. Juntamente com os *patterns*, os Modelos de Objetos do Domínio auxiliam a criação de novos modelos, a partir dos já existentes.

Como resultado desta interação entre *pattern* e Modelos do Domínio, é possível obter reuso de elementos de análise, enfatizando, principalmente, a modelagem estática da aplicação. Os elementos para reuso da modelagem dinâmica são viabilizados através dos cenários e são apresentados na subseção 3.3.2.

Com base nessa estrutura de Modelos do Domínio proposta, foi elaborado um Modelo de Objetos do Domínio para cada modalidade (visão geral e completo). Para tal, levou-se em consideração o domínio[21] adotado por esta dissertação, objetivando tanto exemplificar a aplicação dos modelos, quanto servir como base para os estudos de caso.

São aplicados nos estudos de caso com o objetivo de serem validados e complementados, quando possível. Na seqüência os Modelos de Objetos do Domínio são apresentados em suas duas modalidades: visão geral na Figura 20 e completo na Figura 21.

Vale ressaltar que esses modelos estereotipados seguem uma estilo de modelagem sem a utilização de herança, porém, não impedem que os modelos resultantes implementem tal característica.

---

[21] Automação de um processo de negócio que trata da criação, aprovação e efetivação de uma solicitação de atividade.

Nome: Modelo de Objetos do Domínio - Visão Geral

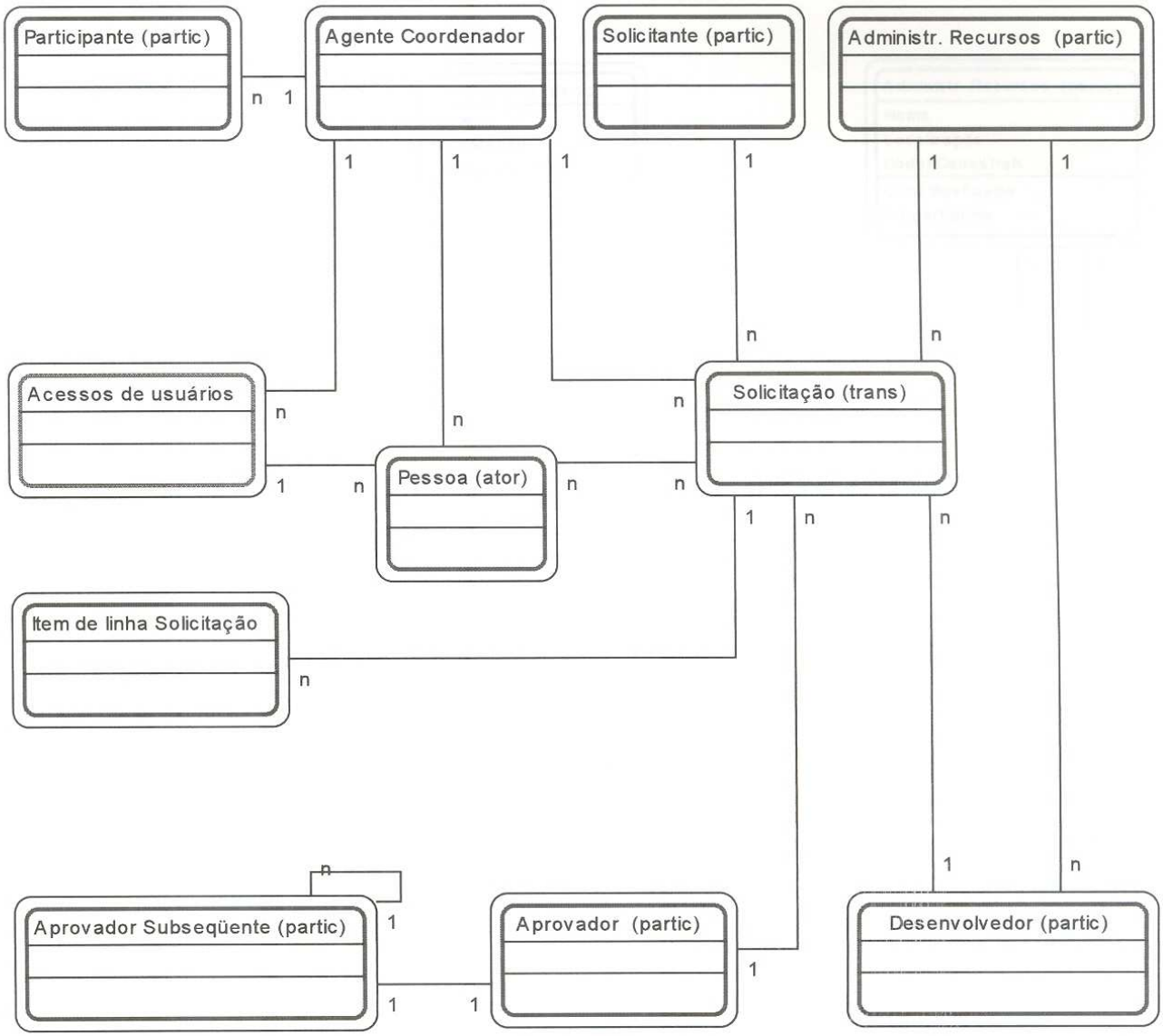


Figura 20 - Modelo de Objetos do Domínio - Visão Geral.

Nome: Modelo de Objetos do Domínio - Completo

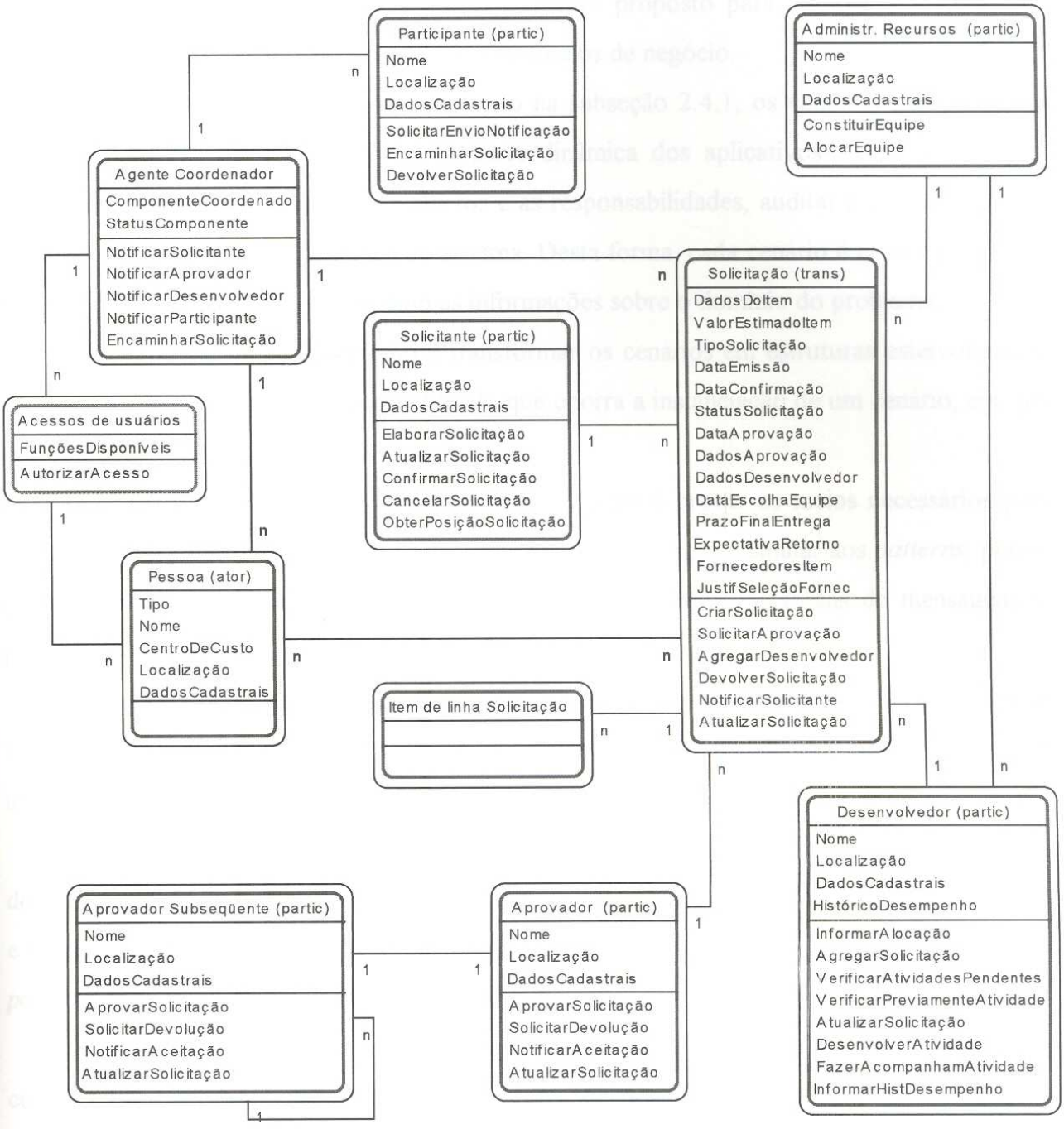


Figura 21 - Modelo de Objetos do Domínio - Completo.

3.3.2 Proposição dos Cenários do Domínio

O Cenário do Domínio é o segundo elemento proposto para compor a estrutura de documentação básica para a modelagem dos processos de negócio.

Na metodologia original, conforme visto na subseção 2.4.1, os cenários são encarados como ferramentas de documentação da parte dinâmica dos aplicativos. A elaboração dos cenários tem por objetivo refinar os objetos e as responsabilidades, auditar a completitude do modelo e melhorar a parte dinâmica do sistema. Desta forma, cada cenário é criado a partir da composição do modelo de objetos com as informações sobre o domínio do problema.

A proposição desta dissertação é transformar os cenários em estruturas estereotipadas, de forma similar aos *patterns*, possibilitando que ocorra a instanciação de um cenário, em um domínio específico.

O objetivo principal destes cenários estereotipados é prover os meios necessários para obter reuso da parte dinâmica de uma modelagem. A abordagem é similar aos *patterns*, porém os componentes reutilizados são as interações entre as classes, as trocas de mensagens e, principalmente, as regras e as rotas dos *workflows*.

Os agentes coordenadores, citados anteriormente, não foram incluídos nos Cenários do Domínio. A principal razão é a busca de uma maior independência da arquitetura de implementação, que é fortemente marcada por tais agentes.

Ainda aderente ao domínio adotado por esta dissertação, na seqüência são apresentados dois Cenários do Domínio. Como o objetivo é fazer complementações à metodologia original e validá-las, parte de um *workflow*, representante de um processo de negócio, é proposto nos *patterns* da subseção 3.2 e nos cenários seguintes.

Em um ambiente produtivo de desenvolvimento de *software*, o catálogo de *patterns* e cenários deve ser bem mais amplo. Ao serem aplicados de forma recorrente em modelos diferentes, novos *patterns* e novos cenários vão sendo incorporados ao catálogo.

Para ilustrar o uso dos Cenários do Domínio, a Figura 22 e a Figura 23 apresentam os cenários referenciados em cada um dos *Patterns* de Processos de Negócio, tal como apresentados na subseção 3.2.

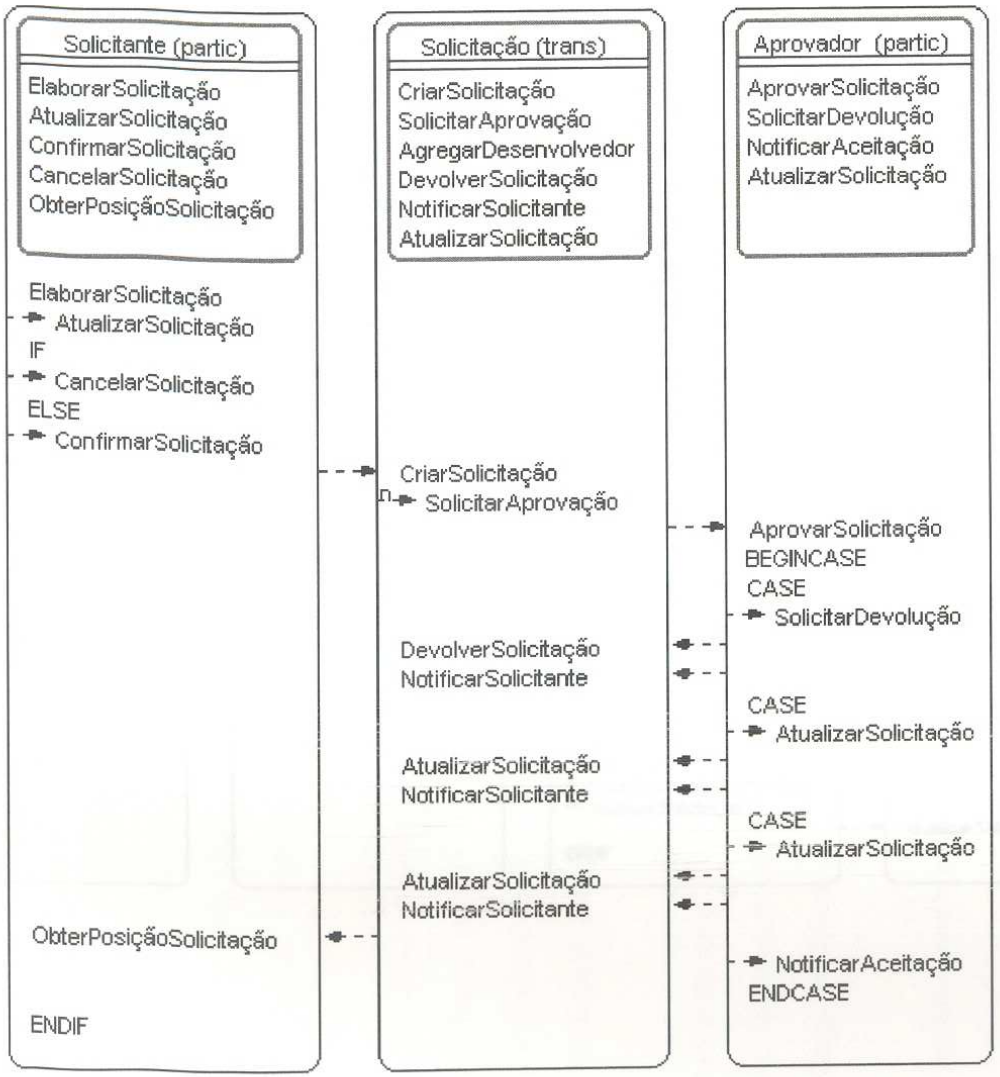


Figura 22 - Cenário do Domínio - Aprovação de Serviço

Name:

Cenário do Domínio - Aprovação de Serviço

Constraints:

Sem Restrições

Solicitante.ElaborarSolicitação(dados solicitação ;)
Solicitante.AtualizarSolicitação(alterações ;)
// Se Solicitante cancela
Solicitante.CancelarSolicitação(;)
// Senão
Solicitante.ConfirmarSolicitação(;)
Solicitação.CriarSolicitação(dados solicitação ; solicitação)
Solicitação.SolicitarAprovação(solicitação ; resultado)
Aprovador.AprovarSolicitação(solicitação ; resultado)
// Caso Solicitação APRESENTA RESSALVAS
Aprovador.SolicitarDevolução()
Solicitação.DevolverSolicitação(solicitação ;)
Solicitação.NotificarSolicitante(solicitante, solicitação, ressalvas ;)
// Caso Solicitação NÃO APROVADA
Aprovador.AtualizarSolicitação(solicitação, motivo recusa ;)
Solicitação.AtualizarSolicitação (solicitação , atualizações ;)
Solicitação.NotificarSolicitante(solicitante, solicitação, motivo recusa ;)
// Caso Solicitação APROVADA SEM RESSALVAS
Aprovador.AtualizarSolicitação(solicitação, confirmação ;)
Solicitação.AtualizarSolicitação (solicitação , atualizações)
Solicitação.NotificarSolicitante(solicitante, solicitação, confirmação ;)
Solicitante.ObterPosiçãoSolicitação (solicitação ; posição)
Aprovador.NotificarAceitação(solicitação aprovada ;)
// FimSe

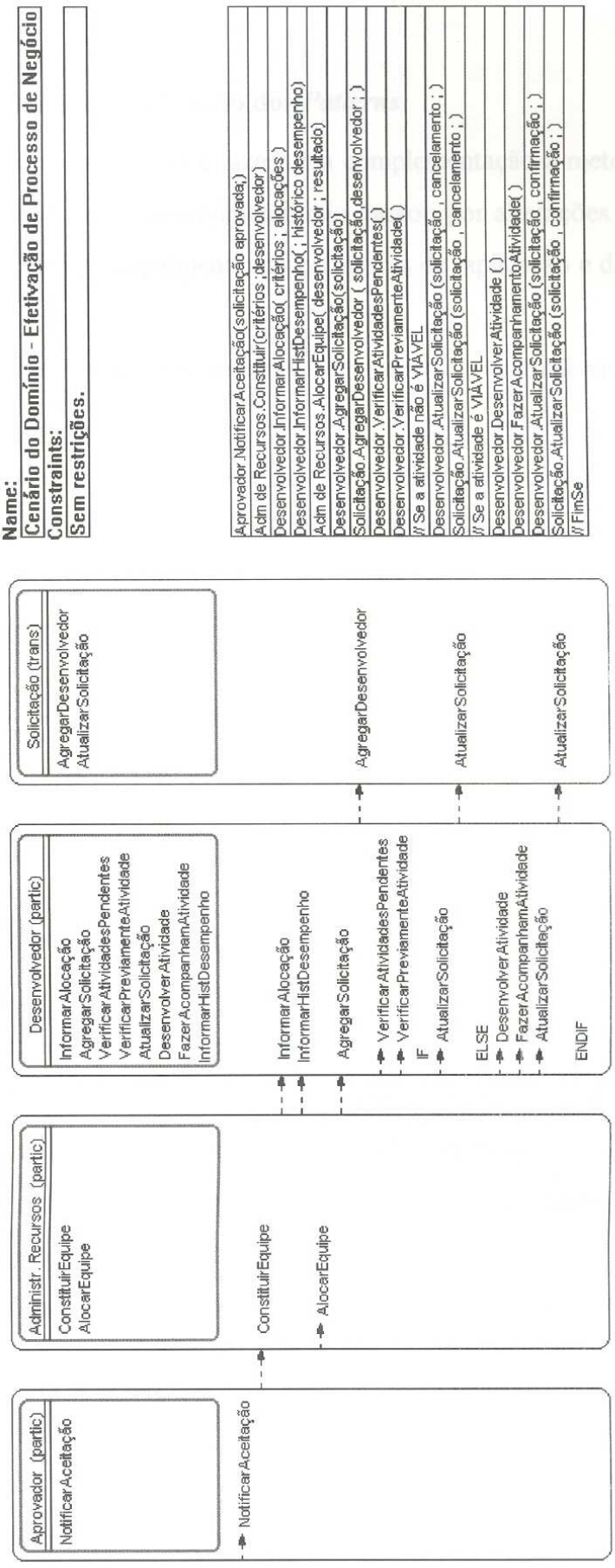


Figura 23 - Cenário do Domínio - Efetivação do Processo de Negócio

3.4 Adequação das Fases para Aplicação dos *Patterns*

Como o objetivo desta dissertação é fazer uma complementação à metodologia original, e não substituí-la, o processo de desenvolvimento não passou por alterações significativas. As alterações foram concentradas principalmente nas estruturas de aplicação e documentação dos *patterns*.

Das proposições feitas, as que inserem fases no processo original são: os *Patterns* de Processos de Negócio, os Modelos de Objetos do Domínio e os Cenários do Domínio. Dentre eles, o que causa maior impacto é a aplicação dos Cenários do Domínio, que passam a seguir a mesma filosofia dos *patterns*, ou seja, possuem uma estrutura estereotipada, que permite ser instanciada e adequada para o domínio onde são aplicados. Esta é a forma pela qual os cenários contribuem para obtenção do reuso de elementos de análise, principalmente na modelagem dinâmica.

Os demais elementos inseridos na metodologia somente complementam etapas no processo original.

Ressalta-se, novamente, que o processo é apresentado através das estratégias, diferindo das demais metodologias que apresentam o seu processo mais formalmente, através de etapas e atividades. A estrutura do processo de desenvolvimento original pode ser vista na subseção 2.4.

Na seqüência, são apresentadas as particularidades do processo que passaram por alterações, ou que simplesmente devem ter uma maior ênfase do que originalmente proposto.

A estratégia n.° 1, denominada "quatro atividades principais, quatro componentes principais", propõe uma ordem de aplicação das atividades [22] do processo e, dentro de cada uma das atividades, a ordem de aplicação dos componentes [23].

Essa estratégia apresenta uma ordem padrão e duas variações possíveis para aplicação, tanto das atividades, quanto dos componentes, atribuindo ao desenvolvedor a responsabilidade pela sua seleção. Tomando-se como base a modelagem dos processos de negócio, propõe-se

---

[22] Quatro atividades: identificar propósitos e requisitos, selecionar objetos, estabelecer responsabilidades e aprimorar a parte dinâmica com cenários.

[23] Quatro componentes: domínio do problema (PD), interação humana (HI), gerenciamento de dados (DM) e interação com sistemas (SI).

utilizar a segunda variação das atividades, ou seja, "... selecionar objetos de transações, agregações e plano. Após, utilizar os *patterns* correspondentes para guiar a seleção de objetos adicionais, estabelecendo responsabilidades na seqüência e, por último, aprimorar a parte dinâmica com cenários".

Ainda fazendo considerações sobre a modelagem de processos de negócio e a estratégia n.º 1, a ordem de utilização dos quatro componentes selecionada foi aquela proposta como padrão, ou seja, "domínio do problema (PD), interação humana (HI), gerenciamento de dados (DM) e interação com sistemas (SI)".

Há ainda um último componente, denominado NT (*not this time*) "não neste momento", não utilizado por não apresentar particularidades relevantes na modelagem de processos de negócio.

Na seqüência são propostos, em forma de tópicos, os principais passos a serem seguidos, sob a ótica das complementações para processos de negócio:

- selecionar as transações e agregações;
- aplicar os *patterns* originais para definir os relacionamentos e algumas responsabilidades;
- aplicar os *Patterns* de Processos de Negócio (subseção 3.2) para complementação dos relacionamentos e das responsabilidades;
- validar e complementar o modelo de objetos com os Modelos de Objetos do Domínio (subseção 3.3.1);
- aplicar os Cenários do Domínio (subseção 3.3.2);
- elaborar os demais cenários não atendidos pelos Cenários do Domínio;
- fazer uma avaliação sobre os *patterns* aplicados e complementá-los, se possível.

As fases iniciais de requisitos, propósitos e propriedades do sistema não são citadas, visto que não foram influenciadas pelas proposições para processos de negócio.

3.5 Considerações sobre a UML

A grande maioria dos trabalhos que endereçam orientação a objetos e *patterns* faz referência aos modelos da UML. Os diagramas e as notações mais utilizadas são os diagramas de classes, diagramas de objetos e os diagramas de interação. Para maiores detalhes, consultar as referências originais [UML 97a] e [UML 97b], ou alternativamente [FOW 97b], [FUR 98] e [ERI 98].

Martin Fowler, em [FOW 97b], considera que os Diagramas de Atividade "são particularmente úteis em conexão com *workflow* e em descrever o comportamento que possui muito processamento paralelo." Entende que os diagramas de atividade informam o que acontece, mas não sobre quem recaem as intervenções. Para efeito de implementação, isto significa que não está especificado qual classe faz cada atividade. Para a modelagem de domínio do problema, significa que o diagrama não aponta qual pessoa ou departamento é responsável por cada atividade. Para suprir esta deficiência, pode-se etiquetar cada atividade com a classe ou pessoa responsável, tal procedimento, contudo, não oferece a mesma clareza que os diagramas de interação.

Outra forma de contornar esta deficiência é arranjar as atividades em zonas verticais, separadas por linhas tracejadas, chamadas de *swimlanes* (raias de natação, em sua tradução literal) ou separadores. Cada zona representa uma classe, pessoa ou departamento com suas respectivas atividades.

Fowler [FOW 97b] sugere a utilização dos diagramas de atividade para analisar um *use case*, tratar aplicações *multi-threaded* e entender *workflow* entre muitos *use cases*. Em situações predominantemente de *workflow*, considera ele que estes diagramas são excelentes.

Entretanto, para a representação de *workflow* (ou processos de negócio), estes diagramas apresentam as deficiências citadas anteriormente, sem que hajam alternativas que não comprometam a modelagem ou a clareza do modelo. Desta forma, a sua utilização para representação de *workflow* é desaconselhada.

Durante a apresentação dos diagramas de atividades, foram citados os diagramas de interação como sendo alternativas para a representação e modelagem de *workflow*. Segundo a notação apresentada pela UML [UML 97a, UML 97b], estas interações são representadas pelos Diagramas de Seqüência e Diagramas de Colaboração.

A diferença básica entre os dois diagramas é a forma de apresentar a seqüência das mensagens trocadas entre os objetos. No diagrama de seqüência, por ser disposto de uma forma cronológica verticalmente, a ordem é determinada pela disposição das mensagens no diagrama, de cima para baixo. Sua vantagem é a simplicidade e clareza para representar as trocas de mensagens entre os objetos, assim como para endereçar concorrência de processo.

Já no diagrama de colaboração, as mensagens são identificadas por números e os objetos, representados por retângulos, dispostos conforme determinação do projetista. Dessa

forma, o diagrama evidencia como os objetos estão ligados e permite que, através do *layout*, sejam delimitados pacotes segundo algum critério de agrupamento. A desvantagem é a visualização da seqüência das mensagens que é prejudicada, visto que o seu objetivo principal não é este.

Independente do diagrama de interação utilizado, há dois tipos de informações de controle disponíveis: condição e marcador de iteração. A condição indica quando uma mensagem é enviada e a iteração indica que uma mensagem é enviada muitas vezes para diversos objetos recebedores.

Furlan, em [FUR 98], aponta os diagramas de colaboração como opção geral para problemas de interações. Os diagramas de seqüência somente devem ser selecionados se houver a necessidade de evidenciar a seqüência das interações. Desta forma, para a modelagem dos processos de negócio, descarta-se a aplicação dos diagramas de colaboração.

Pelas suas características, conclui-se que os diagramas de seqüência podem ser utilizados para a modelagem dos processos de negócio. Estes diagramas permitem o registro da seqüência das atividades, através da sua abordagem *top-down* denominada "linha da vida dos objetos" e permitem o registro de algumas regras, através das condições de guarda.

Uma vez apresentados alguns dos diagramas tradicionais da modelagem orientada a objetos, são feitas considerações envolvendo os diagramas de seqüência e os cenários de *patterns*. Tais cenários são apresentados nas subseções 2.4.1 e 3.3.2.

Traçando um paralelo entre os cenários de *patterns* e os diagramas de seqüência, observam-se várias similaridades: a disposição vertical dos objetos, a abordagem *top-down*, os fluxos representando as mensagens e as condições de guarda para disparo dos fluxos.

Porém, a motivação para adotar os cenários de *patterns* como as ferramentas de modelagem dinâmica, na presente dissertação, é apresentada a seguir:

- são parte integrante da metodologia e do processo de desenvolvimento orientado a *patterns*.
- permitem a criação de hierarquias, onde a abstração é o fator que determina a profundidade de cada cenário.
- os diagramas de seqüência somente apresentam condições de guarda, ao passo que os cenários possuem estruturas mais poderosas de controle e de laço, como *CASE-*

ENDCASE, WHILE-ENDWHILE, IF-ELSE-ENDIF, START_TASK-STOP_TASK, entre outras.

- possuem uma área específica para detalhamento das interações. Nesta, as estruturas de controle, os parâmetros das mensagens e as observações sobre as regras e as rotas podem ser registrados.

Dessa forma, as considerações feitas sobre a utilização de diagramas da modelagem orientada a objetos não invalidam a sua utilização para a modelagem de processos de negócio. Pelas características dos cenários de *patterns*, estes foram os selecionados para compor a estrutura de apoio (Cenários do Domínio), assim como a documentação resultante.

Os diagramas de classes e objetos, referenciados no início desta subseção, apresentam apenas diferenças de notação entre as metodologias, não demandando uma avaliação ou comparação entre as abordagens.

3.6 Considerações sobre o capítulo

Neste capítulo, são feitas proposições com o intuito de tornar a metodologia resultante capaz de suportar a modelagem de processos de negócio. Para tal, inicialmente é apresentada uma nova estrutura de documentação, que serve como base para o desenvolvimento de seis *Patterns* de Processos de Negócio. Em complemento, são propostos os modelos particulares do domínio.

A integração desses elementos é feita através de um roteiro que contém as fases para aplicação das estruturas de *patterns* e dos diagramas. Finalmente, são feitas referências aos diagramas que utilizam a notação UML.

4 ESTUDOS DE CASO

Ao longo deste capítulo, serão apresentados dois estudos de caso nos quais *patterns* são utilizados para fazer a modelagem de processos de negócio. Isto porque, como uma das premissas é a obtenção de reuso na análise de sistemas, a elaboração de somente um estudo de caso não seria suficiente para evidenciar este reuso.

Os dois estudos de caso foram modelados com base em dados obtidos de processos de negócio reais, pertencentes a contextos distintos, em organizações também distintas. O detalhamento da estrutura de documentação de cada sistema é apresentado nas subseções 4.1 e 4.4.

Os principais objetivos a serem atingidos com a elaboração desses estudos de caso são:

- obter reuso de elementos da análise;
- validar os *Patterns* de Processos de Negócio propostos;
- validar os Modelos de Objetos do Domínio e Cenários do Domínio propostos;
- retroalimentar os Modelos de Objetos do Domínio e Cenários do Domínio;
- validar a hipótese apresentada na subseção 1.2 e fornecer elementos para as conclusões.

A granularidade pretendida para os modelos deve atingir o nível da fase de análise. Podendo chegar, em alguns casos, a detalhes contemplados na fase de *design*, conforme o detalhamento das classes, atributos e serviços.

A sistemática para apresentação de cada estudo de caso segue a ordem: dados básicos e contexto do sistema, diagramas obtidos e, ao final, avaliação dos resultados observados durante a modelagem.

Ao apresentar os diagramas obtidos na modelagem [24], são evidenciados os resultados da aplicação dos *patterns*, quais sejam, os modelos de objetos e os cenários. Na metodologia

---

[24] Todos os diagramas foram confeccionados com o suporte da ferramenta Playground v2.0, da empresa Object International (www.oi.com).

original [COA 97], há estratégias que geram outras documentações, como: propósitos do sistema, glossário, fontes de estresse, relação agora e depois, entre outras.

Como o foco principal é o reuso de elementos de análise e como são os *patterns* e os cenários que desempenham esta função, as estratégias que geram apenas documentação não são consideradas, visto que não foram afetadas pelas propostas de alteração. Contudo, as estratégias que norteiam o processo são utilizadas, visando a aplicação dos *patterns*.

Para cada estudo de caso, é feito o mapeamento entre as classes provenientes dos *Patterns* de Processos de Negócio e as classes de cada um dos modelos.

Na última subseção (4.7), onde são tratadas as considerações finais, é apresentada uma tabela que relaciona a quantidade total de atributos e serviços estereotipados nos Modelos do Domínio com os atributos e serviços existentes nos modelos de cada estudo de caso. Desta forma, é evidenciado o percentual de reuso entre os componentes dos modelos. Enumera, ainda, as contribuições obtidas com o desenvolvimento deste trabalho.

4.1 Apresentação do Projeto Gerência de Compras

Para o desenvolvimento do primeiro estudo de caso, foi selecionado um sistema de gerenciamento de solicitações de compras ou serviços, referenciado a seguir por Gerência de Compras. Sua seleção se deve à característica marcante de *workflow* que o sistema deve contemplar.

Este *workflow* rege as rotas e as regras das atividades e documentos envolvidos no processo de negócio. Contempla a criação das solicitações, a aprovação e a efetivação dos procedimentos para a sua execução.

As informações necessárias para a modelagem deste sistema foram extraídas de uma documentação preexistente, resultante do levantamento preliminar de requisitos[25]. Na documentação constavam as funções principais, os tipos de usuários, o *workflow* e os dados dos documentos e visões necessários para o sistema.

Centrando o foco nos elementos da análise, abaixo estão relacionadas, de forma resumida, as principais informações necessárias para a confecção do estudo de caso. As informações abaixo foram extraídas da documentação original, sem complementações:

---

[25] Este estudo de caso foi desenvolvido com base em informações fornecidas pela empresa Productique.

Funções Principais:

- gerenciar todo o ciclo de vida de uma solicitação de compras ou serviço;
- realizar o mecanismo de *workflow* de todo o ciclo de vida de compras;
- verificar automaticamente a disponibilidade de saldo para efetuar contas de investimento e despesa;
- transferir para o sistema centralizador, em *mainframe*, as solicitações atendidas.

Tipos de Usuários:

- Solicitante: qualquer usuário cadastrado no sistema pode emitir uma solicitação de compras. Além dos dados da compra, o solicitante poderá fornecer uma cotação prévia de preços;
- Aprovador da Solicitação: responsável pela aprovação ou não das solicitações de compras geradas. O gestor de compras é geralmente o gerente da área que originou a solicitação (relacionado a um centro de custo específico);
- Gerenciador de Compras: responsável por analisar a solicitação e selecionar o comprador mais indicado para trabalhar com o item solicitado;
- Comprador: responsável por efetuar a cotação de preços junto a 5 possíveis fornecedores do produto solicitado para compra. Após, determina o preço definitivo da compra na negociação final junto ao fornecedor.

O *workflow*, ou fluxo de uma solicitação de compras, é composto por 7 etapas principais, enumeradas abaixo:

1. Emissão da solicitação.
2. Confirmação da solicitação.
3. Aprovação da solicitação.
4. Escolha do comprador.
5. Cotação com 5 fornecedores.
6. Negociação final do preço.
7. Transferência da solicitação para o sistema *mainframe*.

Esses foram os requisitos que nortearam o estudo de caso. A aplicação dos *patterns* propostos, apresentados na subseção 3.2, foi feita conforme o roteiro apresentado na subseção 3.4. Na seqüência são apresentados os resultados obtidos na modelagem.

4.2 Modelagem do Projeto Gerência de Compras

Os modelos apresentados nesta subseção são produtos finais da fase de análise do sistema de Gerência de Compras, descrita anteriormente. As considerações sobre a aplicação dos *patterns*, modelos de objetos e cenários, são feitas na subseção 4.3.

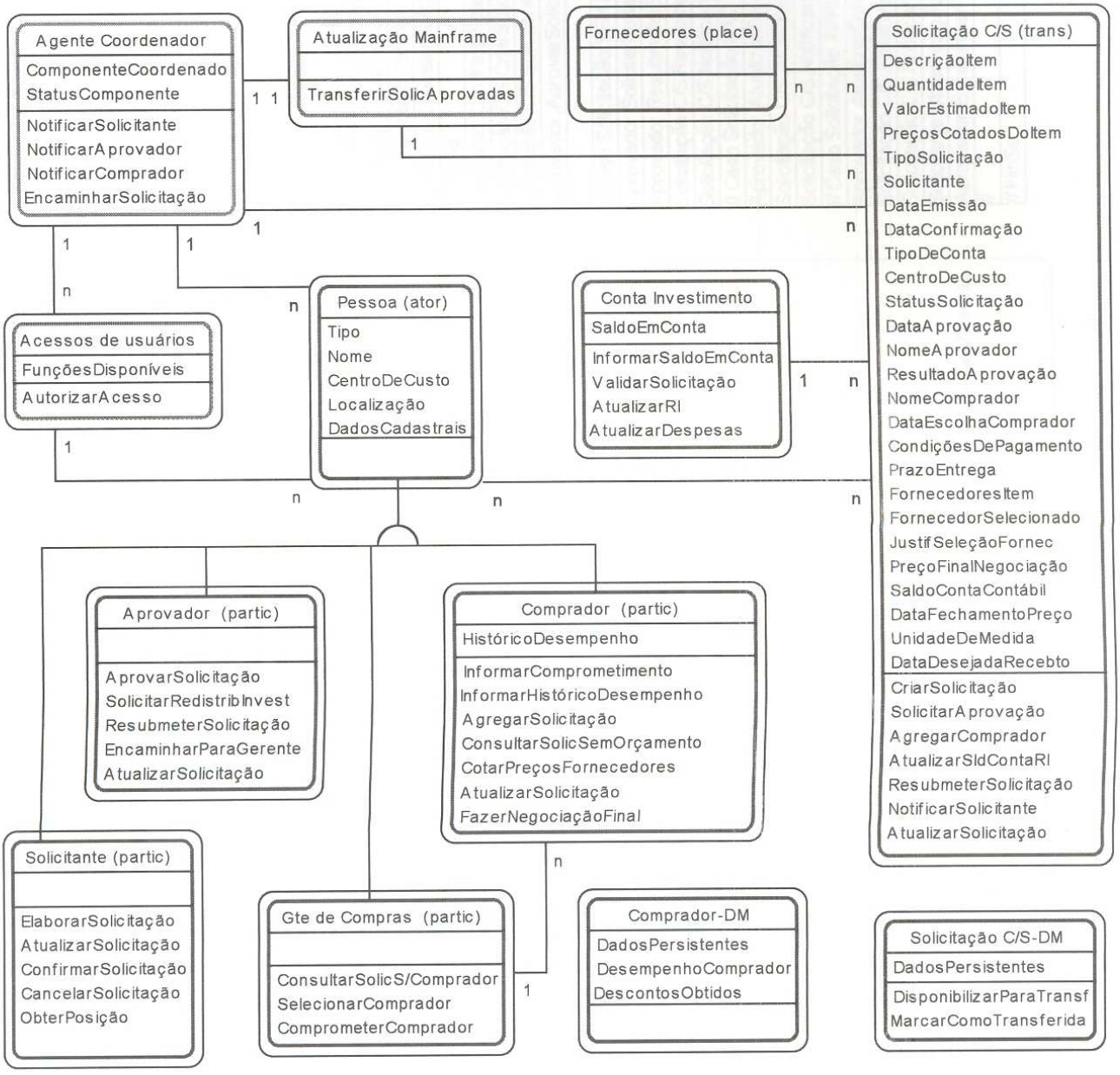


Figura 24 - Modelo de Objetos da Gerência de Compras

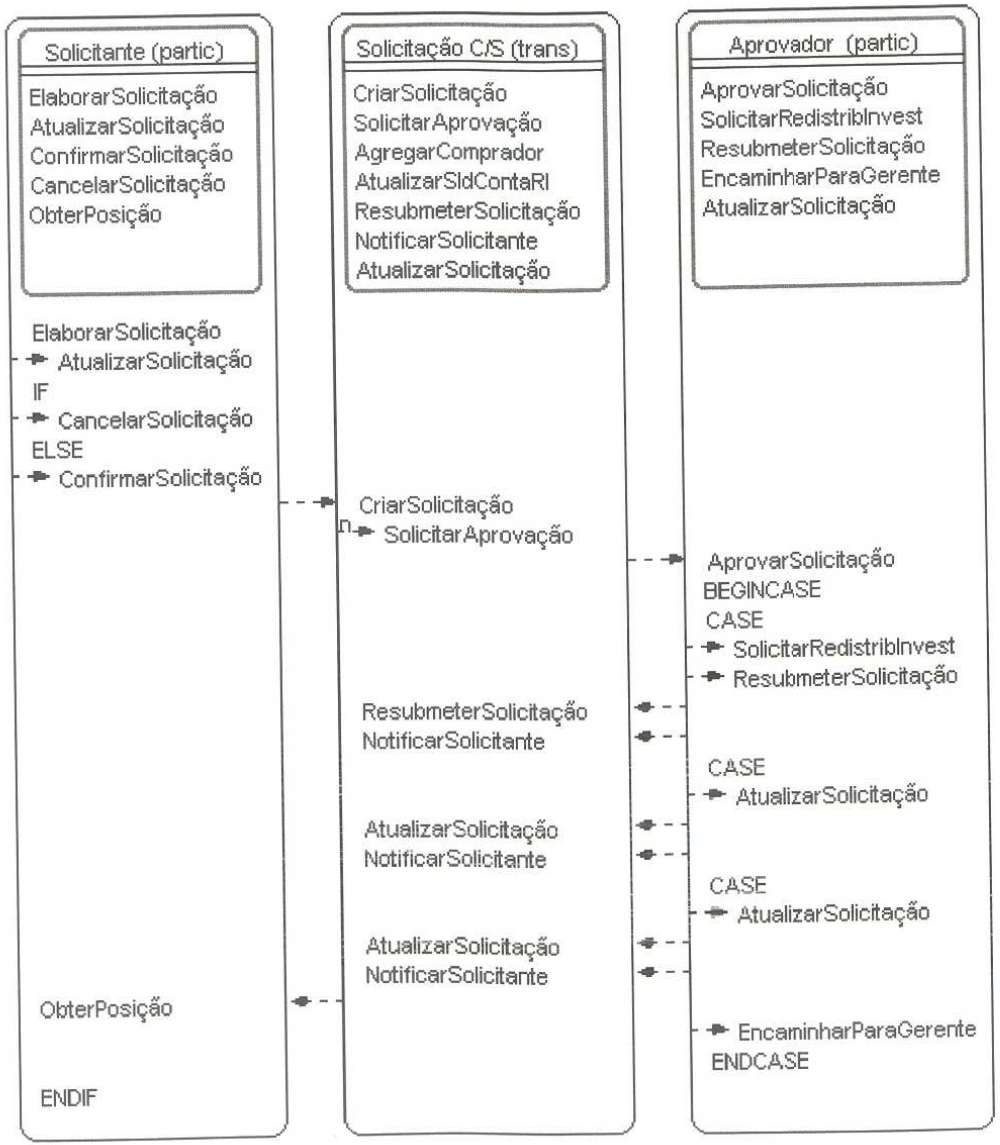


Figura 25 - Cenário de Aprovação de Compras

Name:

Cenário Aprovação de Compras

Constraints:

Sem Restrições

Solicitante.ElaborarSolicitação(dados solicitação ;)
Solicitante.AtualizarSolicitação(alterações ;)
// Se Solicitante cancela
Solicitante.CancelarSolicitação(;)
// Senão
Solicitante.ConfirmarSolicitação(;)
Solicitação C/S.CriarSolicitação(dados solicitação ; solicitação)
Solicitação C/S.SolicitarAprovação(solicitação ; resultado)
Aprovador.AprovarSolicitação(solicitação ; saldo investimento,resultado)
//
// Caso Solicitação SEM SALDO DE INVESTIMENTO
Aprovador.SolicitarRedistribInvest (saldo investimento, centro custo ;)
Aprovador.ResubmeterSolicitação (solicitação, motivo recusa ;)
Solicitação C/S.ResubmeterSolicitação(solicitação ;)
Solicitação C/S.NotificarSolicitante(solicitante, solicitação, motivo recusa ;)
// Caso Solicitação NÃO APROVADA
Aprovador.AtualizarSolicitação(solicitação, motivo recusa ;)
Solicitação C/S.AtualizarSolicitação (solicitação , atualizações)
Solicitação C/S.NotificarSolicitante(solicitante, solicitação, motivo recusa ;)
// Caso Solicitação APROVADA SEM RESSALVAS
Aprovador.AtualizarSolicitação(solicitação, confirmação ;)
Solicitação C/S.AtualizarSolicitação (solicitação , atualizações)
Solicitação C/S.NotificarSolicitante(solicitante, solicitação, confirmação ;)
Solicitante.ObterPosição(solicitação ; posição)
Aprovador.EncaminharParaGerente(solicitação aprovada ;)
//
// FimSe

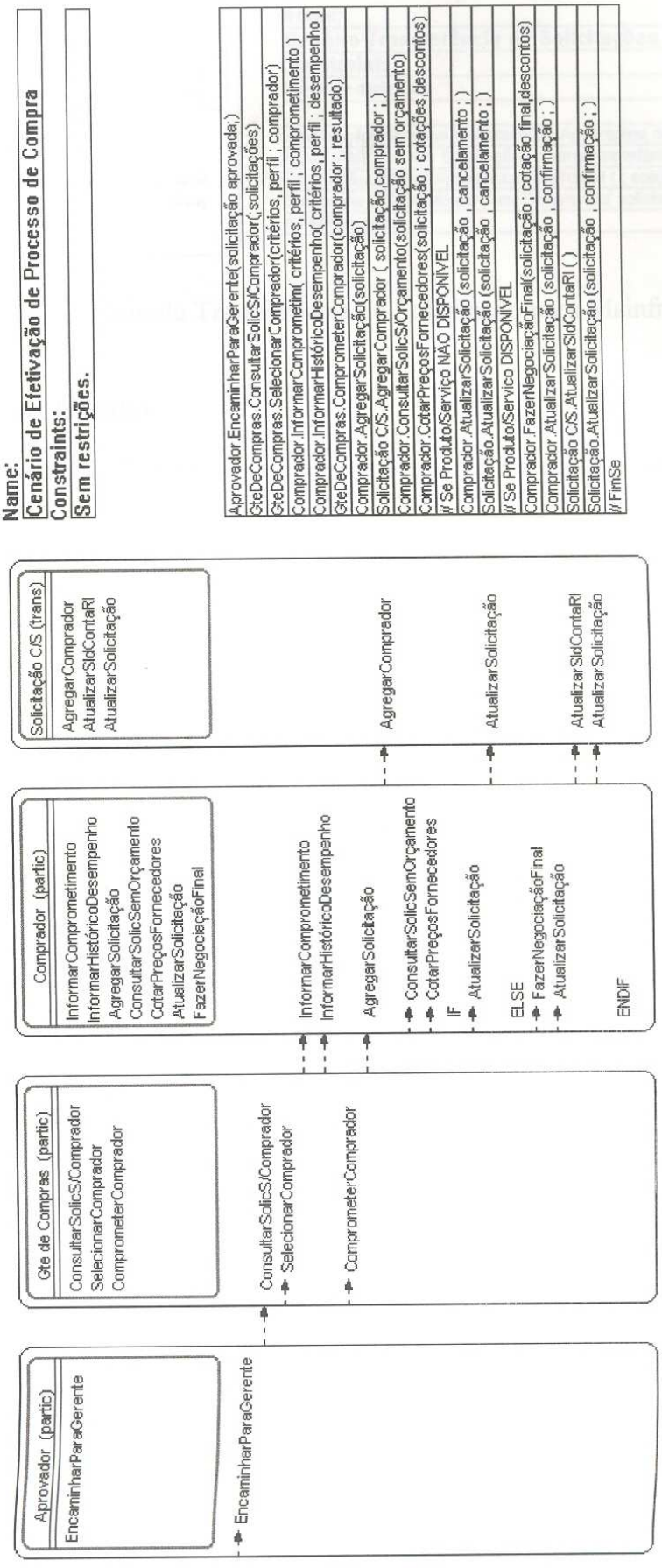


Figura 26 - Cenário de Efetivação de Processo de Compras

Figura 27 – Cenário de Transferência de Solicitação para Mainframe.

4.3 Resultados Parciais Obtidos

Ao propor os *Patterns* de Processos de Negócio desta dissertação, o domínio para sua futura aplicação era conhecido, e com base na metodologia original e no conhecimento deste domínio, os *patterns* foram elaborados.

Inicialmente utilizou-se a seguinte terminologia para o contexto dos *patterns*: Pedido, Aprovação e Aquisição. Após a aplicação destes *patterns* no primeiro estudo de caso, optou-se por alterar as suas denominações para torná-las mais genéricas e significativas. O termo Pedido foi alterado para Solicitação e Aquisição alterado para Desenvolvimento da Atividade.

Fazendo um mapeamento entre as classes dos *Patterns* de Processos de Negócio e as classes obtidas no modelo final deste estudo de caso, tem-se a relação apresentada na Tabela 1.

Patterns de Processo Negócio	Modelo de Gerência de Compras
Solicitante	Solicitante
Solicitação	Solicitação de compras / Serviço
Aprovador	Aprovador de Compras
Aprovador Subseqüente	-
Administrador de Recursos	Gerente de Compras
Desenvolvedor	Comprador
Agente Coordenador	Agente Coordenador
-	Conta Investimento
-	Fornecedores
-	Atualização *Mainframe*

Tabela 1 - Mapeamento entre *patterns* e modelo de Gerência de Compras.

Observa-se que a classe Aprovador Subseqüente não foi utilizada no modelo de Gerência de Compras, devido ao fato de a aprovação ser feita por apenas um participante, tornando-a dispensável. Já as classes criadas no modelo de Gerência de Compras que não são provenientes dos *Patterns* de Processos de Negócio, são tratadas a seguir, na apresentação dos resultados do Modelo de Objetos.

A Solicitação, neste estudo de caso, abrange o pedido de compra de itens ou contratação de serviços. Como o processo é o mesmo para os dois casos, por motivo de simplificação, foi referenciado na modelagem Solicitação de Compras e Processo de Compra.

Seguindo as proposições para os estudos de caso, a efetivação do processo de compra foi referenciada, porém não extensivamente detalhada. Os serviços representantes são: ConsultarSolicSemOrçamento, CotarPreçosFornecedores, AtualizarSolicitação e FazerNegociaçãoFinal. Pela complexidade, observa-se que estes serviços poderiam gerar um novo conjunto de Modelos de Objetos do Domínio e Cenários do Domínio, mais específico para atender ao processo de negócio de efetivação de compra.

Nas próximas subseções (4.3.1 a 4.3.4) são relatadas as particularidades observadas durante a execução desta fase, agrupadas por modelo envolvido. Observa-se que o resultado da modelagem apresentado é composto somente pelo Modelo de Objetos e pelos Cenários de *Patterns*, devido ao enfoque dado ao reuso dos elementos de análise. Dessa forma, informações como 'descrição de objetivos', 'funcionalidades' etc. não são apresentadas. Como referência para uma modelagem que apresente tais elementos pode ser considerado [PAL 98b].

4.3.1 Modelo de Objetos

O modelo de objetos, visto na Figura 24, foi obtido através da aplicação dos *Patterns* de Processos de Negócio. Posteriormente, foram validadas as multiplicidades confrontando-as com o Modelo de Objetos do Domínio - Visão Geral (Figura 20). A última verificação é feita através do Modelo de Objetos do Domínio - Completo (Figura 21), no qual todos os serviços e atributos estereotipados são novamente revisados.

Para facilitar a representação do relacionamento entre os participantes e a solicitação, foi criada a classe Pessoa, que é a generalização de Solicitante, Aprovador de Compras, Gerente de Compras e Comprador. Além dessa prática de modelagem, a metodologia original faz referência à criação de uma classe Pessoa no *pattern* n.º 2 Ator-Participante.

O serviço ConsultarSolicS/Comprador, da classe Gerente de Compras, difere da filosofia do restante do sistema, onde as informações e documentos são enviados ao participante interessado. Neste caso, foi criado um serviço para evitar que a caixa de entrada do gerente seja congestionada com solicitações. Desta forma, o gerente irá verificar as pendências quando julgar necessário.

Na classe Agente Coordenador, os serviços NotificarSolicitante e EncaminharSolicitação foram criados para exemplificar a modelagem considerando-se a arquitetura e plataforma de implementação[26]. Por esta razão, esta classe e estes serviços somente compõem o modelo de objetos, não sendo considerados nos cenários. O mesmo ocorre com a classe Acessos de Usuários. Tais classes, criadas com o objetivo único de exemplificação, não são completas e não atendem a todos os requisitos que demandariam para uma efetiva implementação.

Dentre as particularidades do sistema não atendidas pelos *patterns* e cenários, estão as classes Fornecedores, Contas de Investimento e Atualização *Mainframe*. Estas classes foram incluídas no modelo de objetos deste estudo de caso, contudo não foram incluídas no Modelo de Objetos do Domínio ou nos *Patterns* de Processos de Negócio. Tais classes são particularmente específicas dos requisitos da aplicação sendo modelada. Desta forma, a probabilidade de ocorrência em outras aplicações é pequena, não justificando a sua inclusão nos *patterns* e cenários, que objetivam diretamente o reuso de seus elementos.

Já as classes de gerenciamento de dados (DM), como Comprador-DM e Solicitação-DM, foram referenciadas somente com o objetivo de exemplificar a sua utilização, não alterando a estrutura e os elementos da análise. Foram mantidas sem relacionamento com os demais componentes do modelo, conforme proposição da metodologia original.

Na elaboração desse Modelo de Objetos, os *patterns* originais (primeiro nível de abstração) não foram diretamente utilizados, visto que os *Patterns* de Processos de Negócio (segundo nível) supriram tal necessidade. Vale ressaltar que indiretamente utilizou-se o catálogo original pois, como exemplo, o *Pattern* de Processo de Negócio nº 1 Solicitante-Solicitação é originário do *pattern* nº 3 Participante-Transação, da metodologia original.

Apesar de não ter sido identificado como necessário neste estudo de caso, a aplicação direta do catálogo original de *patterns* é incentivado, especialmente quando os demais níveis de *patterns* não suprirem os requisitos para a modelagem.

O Modelo de Objetos resultante foi, em termos de separação por componentes da metodologia original, assim dividido: Classes PD: Aprovador, Comprador, Conta Investimento, Fornecedor, Gerente de Compras, Pessoa, Solicitação e Solicitante; Classes HI:

---

[26] Conforme definição inicial, a plataforma de implementação dos estudos de caso é Lotus Notes, porém faz-se a ressalva que a metodologia ora empregada pode criar modelos adequados para implementação em qualquer plataforma cliente-servidor, como por exemplo em ambiente web.

Acessos de Usuários; Classes SI: Agente Coordenador e Atualização Mainframe; e, por último Classes DM: Comprador DM e Solicitação DM.

4.3.2 Cenário de Aprovação de Compras

Inicialmente, o Cenário do Domínio - Aprovação de Serviço (Figura 22) contemplava somente duas opções para a solicitação: aprovada e não aprovada. Ao instanciar o cenário e aplicá-lo ao domínio do problema, gerando o Cenário de Aprovação de Compras (Figura 25), verificou-se a existência de uma situação intermediária, onde a solicitação não era recusada, porém, aguardava uma redistribuição de saldos. Para contemplar este caso, foi alterado o Cenário do Domínio (Figura 22) incluindo uma terceira opção, denominada Solicitação com Ressalvas.

4.3.3 Cenário Efetivar Processo de Compras

A possibilidade de um comprador fazer cotação de preços em fornecedores e verificar que não há disponibilidade para atender a solicitação, fez com que surgisse o serviço VerificarPreviamenteAtividade no Cenário do Domínio. Neste caso, conforme visto na Figura 26, o serviço permite que o próprio comprador cancele a solicitação, por falta de disponibilidade. Em outros, o desenvolvedor da atividade pode cancelar a solicitação por motivos diversos.

4.3.4 Cenário Transferência de Solicitações

Este cenário, visto na Figura 27, surgiu em função da necessidade específica da aplicação, não contemplada pelos *Patterns* de Processos de Negócio. Não foi oriundo de nenhum Cenário do Domínio, tampouco deu origem a um Cenário do Domínio, visto que é muito particular ao sistema e tem pouca probabilidade de ser reutilizado diretamente em outras aplicações.

Com base nas complementações evidenciadas em cada etapa deste primeiro estudo de caso, foram atualizados os *Patterns* de Processos de Negócio, os Modelos de Objetos do Domínio e os Cenários do Domínio. O segundo estudo de caso somente teve seu início após estas atualizações.

4.4 Apresentação do Projeto Sistemática de Trabalho

O segundo estudo de caso foi desenvolvido com base em processos de negócio que tratam demanda e aprovação de atividades de informática (anteprojeto), formação de equipes e posterior desenvolvimento das atividades. A referência a esse conjunto de atividades é feita por Sistemática de Trabalho. A sua seleção é devida à característica marcante de *workflow*, da mesma forma que para a Gerência de Compras, porém, pertencente a um contexto diferente.

As informações necessárias para a modelagem deste sistema foram extraídas de uma documentação predefinida, resultante do levantamento preliminar dos requisitos [27]. Da mesma forma que para o primeiro estudo de caso, o foco são os elementos de análise.

Na seqüência estão resumidas as informações utilizadas na modelagem do sistema. Devido à estrutura diferenciada e particular às organizações onde cada estudo de caso foi aplicado, foram mantidas as características dos requisitos de cada uma.

A primeira parte é a descrição funcional, que apresenta as principais funções, juntamente com os possíveis papéis que cada participante desempenha, enumeradas na seqüência.

1) Registrar Demanda:

Um analista de sistemas (com perfil voltado para o negócio) será o responsável por filtrar solicitações de serviços de informática. Uma vez detectada a oportunidade de um novo produto, que resultará em ganhos para a instituição, o analista elabora uma solicitação para estudo preliminar.

Serão registradas informações pertinentes à solicitação da área usuária, como descrição da atividade, retorno financeiro etc.

2) Priorizar Demanda:

Uma vez registrada a demanda, o gerente de informática responsável pela área envolvida priorizará ou devolverá a solicitação.

3) Orçar Projeto:

Neste momento, o analista deverá fazer a previsão orçamentária e estimativa de prazos para a atividade.

---

[27] Este estudo de caso foi desenvolvido com base em informações fornecidas pelo Banco do Estado do Paraná.

4) Enviar para aprovação:

Sendo a atividade devidamente priorizada e orçada, será enviada para aprovação da área usuária, via correio eletrônico.

5) Receber Parecer do Usuário:

A área usuária analisará os custos e prazos da solicitação que lhe é destinada. Após, pode retornar parecer favorável para início das atividades ou renegociar, juntamente com o analista que atende sua área, os prazos e custos da solicitação.

6) Constituir Equipe:

Através de consultas ao histórico do perfil e ao planejamento de atividades dos técnicos, será constituída equipe e alocados recursos humanos para a atividade. Se houver uma equipe predeterminada, esta será revista e efetivada.

7) Iniciar Atividades:

Uma vez que o usuário do sistema aprova a atividade e os técnicos já estão alocados, os dados da atividade são atualizados e esta é iniciada.

8) Registrar Acompanhamento:

Cada projeto pode ter seu prazo de acompanhamento diferenciado. O registro, feito pelo analista líder, deve estar de acordo com a periodicidade determinada.

Para a representação inicial do *workflow*, foi utilizado um diagrama específico da metodologia da organização, com notação particular[28], visualizado na Figura 28. Este diagrama somente é apresentado, nesta dissertação, com o objetivo de prover as informações utilizadas no estudo de caso. Tal diagrama não é avaliado, visto que os cenários de *patterns* irão desempenhar o seu papel de representar a seqüência de atividades do *workflow*.

---

[28] Resumo da notação. Quadrado: atividade; Lado esquerdo: entradas da atividade; Lado direito: saídas; Lado inferior: responsável pela atividade; [%]: ou; [&]: e. Tal notação é uma adaptação da proposta por Central Computer and Telecommunications Agency (CCTA), denominada SSADM e da técnica IDEF 0, desenvolvolvida por Doug Ross.

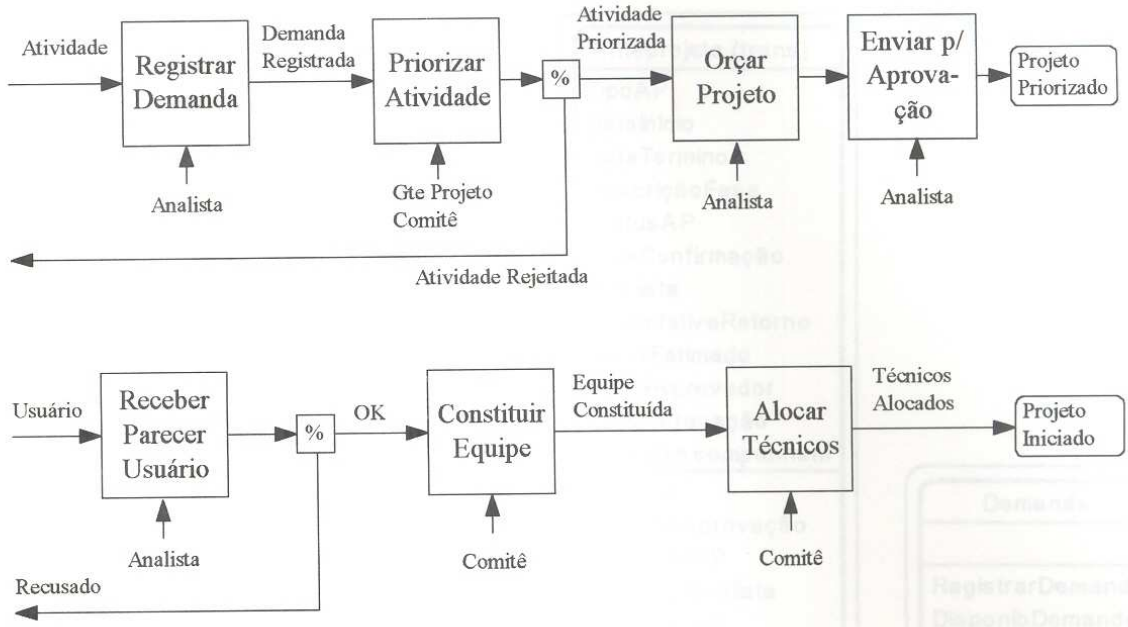


Figura 28 - *Workflow* original do aplicativo de Sistemática de Trabalho.

Este diagrama contempla somente os elementos considerados neste estudo de caso. No diagrama original há, ainda, um *workflow* para registro de anomalias, outro para a geração de relatórios de registros de acompanhamentos e, por último, um *workflow* para entrega, validação e acompanhamento da atividade encerrada.

Esses foram os requisitos que nortearam o segundo estudo de caso. De forma similar ao primeiro estudo de caso, a aplicação dos *patterns* propostos, subseção 3.2, foi feita conforme o roteiro apresentado na subseção 3.4. Na seqüência são apresentados os resultados obtidos na modelagem.

4.5 Modelagem do Projeto Sistemática de Trabalho

Os modelos apresentados nesta subseção são produtos finais da fase de análise do sistema de Sistemática de Trabalho, descrita anteriormente. As considerações sobre a aplicação dos *patterns*, modelos de objetos e cenários são feitas na subseção 4.6.

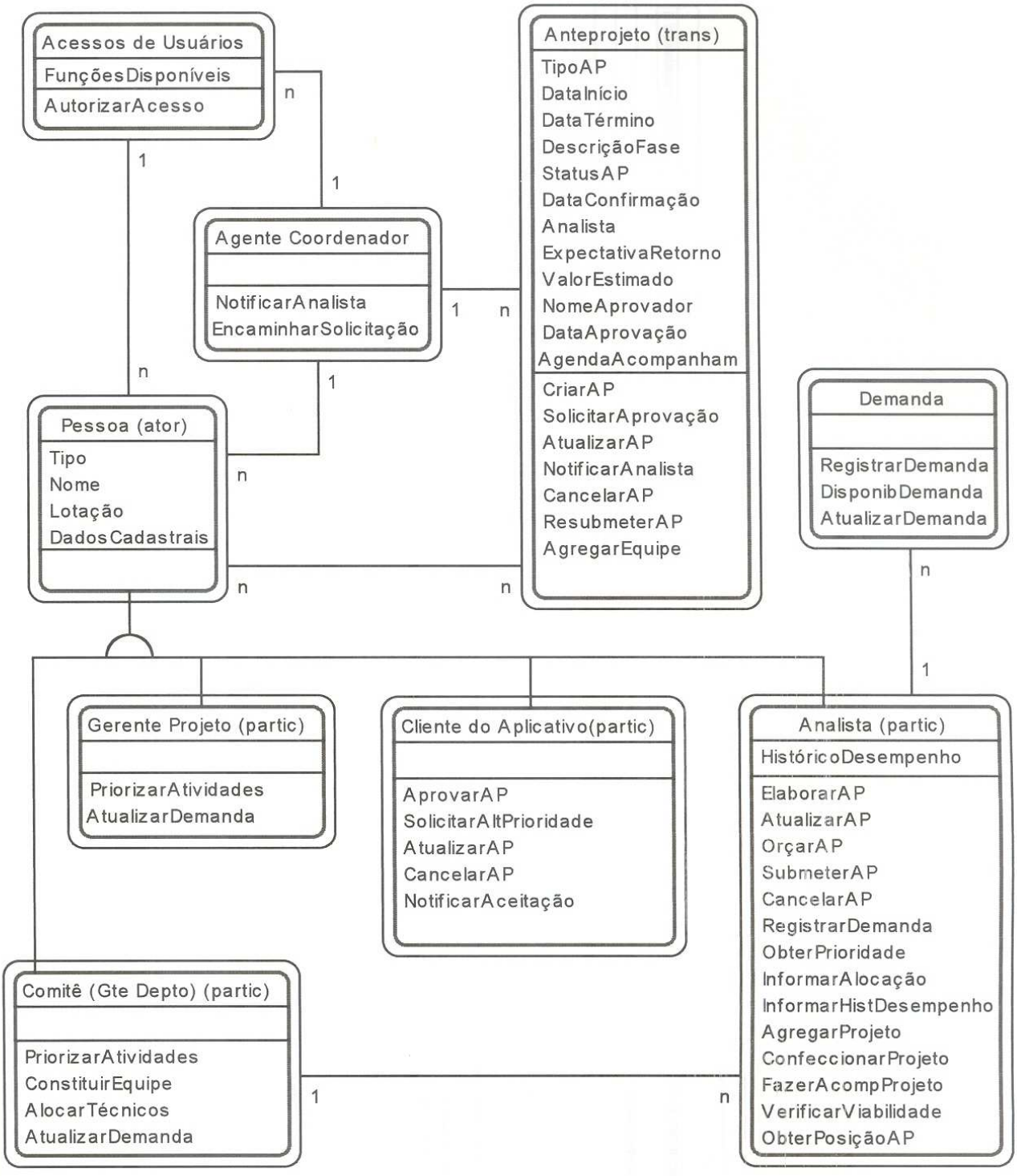


Figura 29 - Modelo de Objetos da Sistemática de Trabalho.

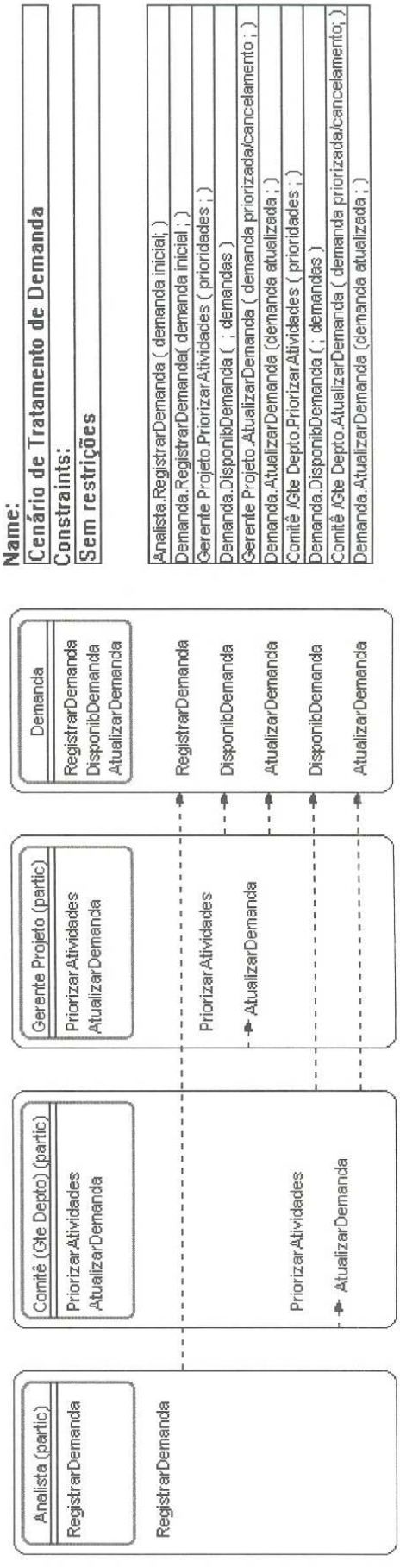


Figura 30 - Cenário de Tratamento de Demanda.

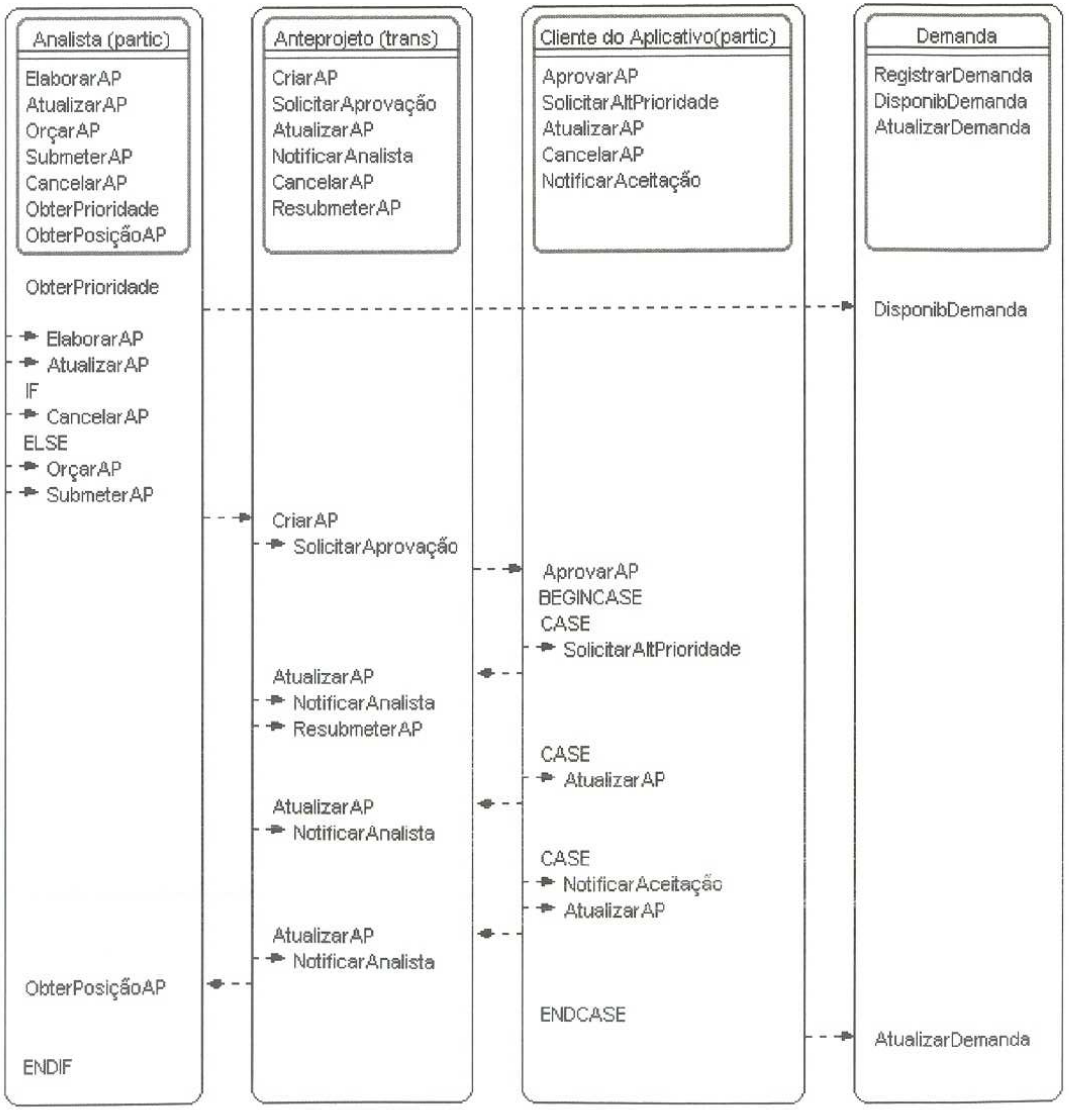


Figura 31 - Cenário de Aprovação de Anteprojeto.

Name:

Cenário Aprovação de Anteprojeto

Constraints:

Sem restrições

Analista.ObterPrioridade (; demanda com prioridade)
Demanda.DisponibilizarDemanda (; demanda com prioridade)
Analista.ElaborarAP (dados AP ;)
Analista.AtualizarAP (alterações AP ;)
// Se Analista cancela AP (anteprojeto ;)
Analista.CancelarAP (anteprojeto ;)
// Senão
Analista.OrçarAP (orçamento AP ;)
Analista.Submeter AP (anteprojeto, dados AP ;)
Anteprojeto.CriarAP (anteprojeto, dados AP ;)
Anteprojeto.SolicitarAprovação (anteprojeto ; aprovação)
Cliente do Aplicativo.AprovarAP (anteprojeto ; aprovação)
//
// Caso anteprojeto tenha o INÍCIO ADIADO
Cliente do Aplicativo.SolicitarAltPrioridade(anteprojeto, adiamento ;)
Anteprojeto.AtualizarAP (anteprojeto, adiamento ;)
Anteprojeto.NotificarAnalista (analista, adiamento ;)
Anteprojeto.Resubmeter AP (anteprojeto, adiamento ;)
// Caso anteprojeto NÃO ACEITO
Cliente do Aplicativo.AtualizarAP (anteprojeto, motivo recusa ;)
Anteprojeto.AtualizarAP (anteprojeto, motivo recusa ;)
Anteprojeto.NotificarAnalista (analista, motivo recusa ;)
// Caso anterprojeto ACEITO SEM RESSALVAS
Cliente do Aplicativo.NotificarAceitação (anteprojeto, confirmação ;)
Cliente do Aplicativo.AtualizarAP (anteprojeto, confirmação ;)
Anteprojeto.AtualizarAP (anteprojeto, confirmação ;)
Anteprojeto.NotificarAnalista (analista, confirmação ;)
Analista.ObterPosiçãoAP (anteproejto ; andamento atual)
//
Demanda.AtualizarDemanda (anteprojeto , [cancelado, adiado,andamento, etc]
// FimSe

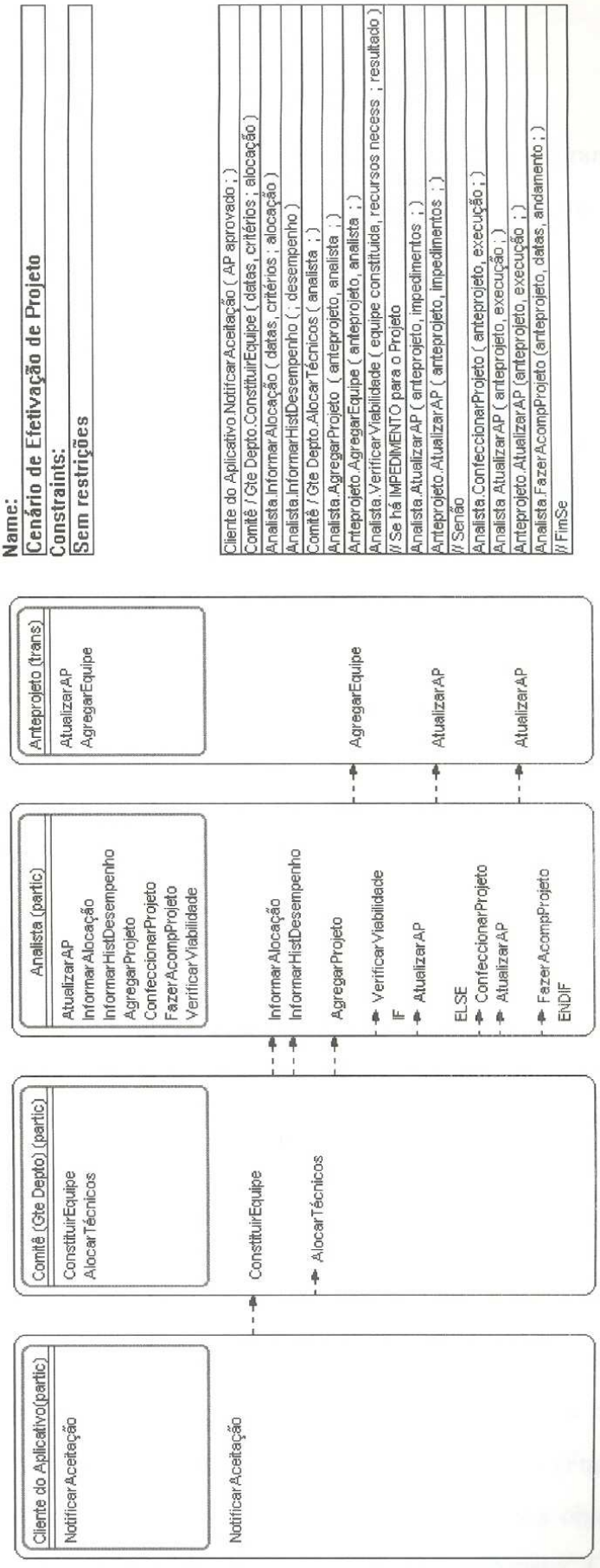


Figura 32 - Cenário de Efetivação de Projeto.

4.6 Resultados Obtidos

Algumas terminologias da documentação preliminar de requisitos foram alteradas para a modelagem dos processos. Dentre elas, a referência de Usuário passou a ser Cliente do Aplicativo e Projeto passou a ser referenciado por Anteprojeto.

Assim como no primeiro estudo de caso, é apresentado, na Tabela 2, o mapeamento entre as classes dos *patterns* e as classes obtidas no modelo final deste estudo de caso.

Patterns de Processo Negócio	Modelo de Sistemática de Trabalho
Solicitante	Analista
Solicitação	Anteprojeto
Aprovador	Cliente do Aplicativo
Aprovador Subseqüente	-
Administrador de Recursos	Comitê (Gerente Departamento)
Desenvolvedor	Analista
Agente Coordenador	Agente Coordenador
-	Gerente de Projetos
-	Demanda

Tabela 2 - Mapeamento entre *patterns* e modelo de Sistemática de Trabalho

De forma análoga ao primeiro estudo de caso, evidenciou-se que:

- a classe Aprovador Subseqüente não foi empregada nos modelos da Sistemática de Trabalho;
- algumas classes específicas do contexto de aplicação foram criadas e não retroalimentaram os *patterns* e cenários, quais sejam, Gerente de Projetos e Demanda;
- o processo de execução do anteprojeto não foi extensivamente detalhado, sendo representado pelos serviços ConfeccionarProjeto, AtualizarAP e FazerAcompProjeto.

Na seqüência, são relatadas as particularidades observadas durante a execução desta fase, agrupadas por modelo envolvido.

4.6.1 Modelo de Objetos

Seguindo a mesma diretriz do primeiro estudo de caso, o modelo de objetos, visto na Figura 29, foi obtido através da aplicação dos *Patterns* de Processos de Negócio e dos Modelos de Objetos do Domínio, de visão geral (Figura 20) e completo (Figura 21).

A classe Pessoa foi criada de forma similar e com os mesmos objetivos do primeiro estudo de caso, ou seja, generalizando as classes Gerente de Projeto, Comitê, Cliente do

Aplicativo e Analista. Ainda referenciando semelhanças ao primeiro estudo de caso, na classe Agente Coordenador, os serviços NotificarAnalista e EncaminharSolicitação foram criados para exemplificar a modelagem considerando-se a plataforma de implementação, assim como a classe Acessos de Usuários.

Gerente de Projeto e Demanda são as classes particulares deste sistema e não atendidas pelos *patterns* e cenários. A classe Gerente de Projeto foi criada especialmente para determinar a prioridade das atividades e demandas, função também desempenhada pelo o comitê.

Essas classes constam do modelo de objetos particular deste estudo de caso, entretanto, não foram incluídas no Modelo de Objetos do Domínio ou nos *Patterns* de Processos de Negócio. Assim como no primeiro estudo de caso, as classes particulares ao domínio têm pouca probabilidade de apresentar reuso em outras aplicações do domínio, não justificando a sua inclusão nos modelos estereotipados.

O atributo ExpectativaDeRetorno foi identificado na classe Anteprojeto, no momento em que esta última era detalhada. Mesmo sem que este atributo tenha sido evidenciado no primeiro estudo de caso, tornou-se um atributo estereotipado nos Modelos do Domínio e no *Pattern* de Processo de Negócio n.° 1, pela sua generalidade e probabilidade de reuso.

As classes de gerenciamento de dados, já referenciadas no primeiro estudo de caso somente com o objetivo de ilustração, não fizeram parte do modelo de objetos. Possíveis classes de gerenciamento de dados seriam Analista-DM, Solicitação-DM e Demanda-DM.

Ressalta-se novamente a possibilidade de aplicação dos *patterns* do catálogo original (primeiro nível de abstração) na modelagem, especialmente quando os demais níveis de *patterns* não suprirem os requisitos para a modelagem. Assim como ocorrido no primeiro, este estudo de caso foi atendido pelos *Patterns* de Processos de Negócio.

O Modelo de Objetos resultante foi, em termos de separação por componentes da metodologia original, assim dividido: Classes PD: Analista, Anteprojeto, Cliente do Aplicativo, Comitê, Demanda, Gerente de Projeto, Pessoa; Classes HI: Acessos de Usuários; Classes SI: Agente Coordenador.

4.6.2 Cenário Tratar Demanda

Este cenário, visto na Figura 30, surgiu em função da necessidade específica da aplicação, não contemplada pelos *Patterns* de Processos de Negócio ou pelos Cenários do

Domínio. Não foi oriundo de, tampouco convertido para, um Cenário do Domínio pois é particular ao sistema e tem pouca probabilidade de ser reutilizado.

4.6.3 Cenário Aprovação de Anteprojeto

O primeiro estudo de caso demandou uma alteração no Cenário do Domínio - Aprovação de Serviço (Figura 22), incluindo uma terceira opção para o resultado da aprovação. Neste cenário de aprovação de anteprojeto, visto na Figura 31, essa nova opção fez-se necessária e, ao seu término, as possíveis opções de aprovação do anteprojeto foram: aceito sem ressalvas, não aceito e início adiado.

4.6.4 Cenário Efetivação de Projeto:

O serviço estereotipado VerificarPreviamenteAtividade do Cenário do Domínio - Efetivação do Processo de Negócio (Figura 23), criado no primeiro estudo de caso, foi instanciado através do serviço VerificarViabilidade neste cenário, conforme visto na Figura 32. Esta atividade é feita pelo Analista, verificando se há equipe constituída e recursos necessários para o andamento do projeto.

A necessidade de fazer acompanhamento das atividades, fez surgir o serviço FazerAcompanhamAtividade na classe Desenvolvedor, no Cenário do Domínio. Um dos objetivos deste acompanhamento de atividades é a avaliar o desempenho e fornecer critérios para participação em anteprojetos. Essa necessidade não foi evidenciada na confecção dos *patterns* e, no primeiro estudo de caso, o histórico de desempenho do comprador é obtido em função dos descontos obtidos nas negociações, não exigindo uma intervenção direta do envolvido.

4.7 Considerações Gerais

Ao término do segundo estudo de caso, todos os modelos propostos foram utilizados e evoluídos, conforme a expectativa inicial. A utilização dos *patterns* e cenários, seguida da complementação, quando necessária, foram apresentadas nas subseções anteriores.

Os modelos de objetos dos estudos de caso, concebidos a partir dos *Patterns* de Processos de Negócio e dos Modelos de Objetos do Domínio, representam a modelagem estática do sistema. Evidenciam o relacionamento entre as classes, as multiplicidades, os atributos e os serviços necessários para compor a modelagem.

As regras e rotas para a automatização dos processos de negócio foram evidenciadas nos cenários dos estudos de caso. Tais cenários foram concebidos a partir da instanciação dos Cenários do Domínio e são os representantes da modelagem dinâmica do sistema.

Como forma de mensuração do reuso obtido com a aplicação dos *patterns* e Modelos do Domínio, é apresentada a Tabela 3. Ressalta-se que esta tabela tem por objetivo sumariar os resultados obtidos nos estudos de caso, sem atribuir graus ou escalas de eficiência de reuso, assim como não se propõe a fazer comparações com resultados de outros estudos de casos, haja vista a ênfase de reuso de análise ser pouco endereçada na comunidade.

Essa tabela faz um mapeamento entre a quantidade total de elementos oferecidos pelos Modelos do Domínio e a quantidade de elementos reutilizados ou criados. Para compor essa tabulação, os elementos considerados nos modelos de objetos são todos os atributos e os serviços. Já para os cenários, os elementos são os serviços invocados e as estruturas de controle e laço (*case, if, while* etc.). Observa-se que as classes de gerenciamento de dados (DM) e a classe Aprovador Subsequente não foram computadas nesta tabulação, devido ao fato de terem sido incluídas nos modelos somente com o intuito de exemplificar o seu uso.

Para melhor ilustrar a forma de leitura das informações dessa tabela, na seqüência é apresentado um exemplo. Tomando-se a primeira linha da tabela de Gerência de Compras, que faz referência ao modelo de objetos, considera-se que dos 72 elementos (atributos e serviços) oferecidos pelo Modelo de Objetos do Domínio, 49 foram reutilizados diretamente no modelo de objetos do primeiro estudo de caso.

Ou seja, 67,12% dos serviços e atributos do modelo de objetos, são provenientes da instanciação dos elementos estereotipados. Porém, 24 novos serviços ou atributos foram criados sem o auxílio dos Modelos do Domínio, representando 32,88% do total de elementos.

Vale ressaltar, novamente, que os Modelos de Objetos do Domínio representam uma síntese dos *Patterns* de Processos de Negócio. O reuso obtido nos modelos de objetos dos estudos de caso, se deve à aplicação dos *Patterns* de Processos de Negócio na modelagem. Já o reuso obtido nos cenários dos estudos de caso, se deve à instanciação dos Cenários do Domínio.

	Total	Gerência de Compras			
	Domínio	Qtd Reuso	% Reuso	Qtd Novos	% Novos
Modelo de Objeto do Domínio	72	49	67,12%	24	32,88%
Cenário do Domínio Aprovação Serviço	26	26	96,30%	1	3,70%
Cenário do Domínio Efetivação Proc. Negócio	18	17	89,47%	2	10,53%

	Total	Sistemática de Trabalho			
	Domínio	Qtd Reuso	% Reuso	Qtd Novos	% Novos
Modelo de Objeto do Domínio	72	38	66,67%	19	33,33%
Cenário do Domínio Aprovação Serviço	26	25	80,65%	6	19,35%
Cenário do Domínio Efetivação Proc. Negócio	18	17	94,44%	1	5,56%

Tabela 3 - Percentuais de reuso de elementos da análise

Nos dois estudos de caso, observou-se que houve complementação aos *Patterns* de Processos de Negócio e aos cenários. Desta forma, conclui-se que a sua consecutiva aplicação pode ocasionar:

- um refinamento dos *patterns* e cenários, tornando-os mais abrangentes e efetivos, como ocorrido nestes estudos de caso;
- um acréscimo ao catálogo de *patterns* e cenários, dentro de um domínio existente ou criando-se um novo domínio, como identificado nos processos de compra e anteprojeto, que justificariam a criação de um novo domínio de *patterns* e cenários.

Para tal, é importante a avaliação posterior à aplicação dos *patterns*, principalmente objetivando complementá-los para futuras utilizações em outros sistemas.

4.8 Contribuições

A principal contribuição deste trabalho é a apresentação de novas diretrizes para utilização e obtenção de *patterns* aplicados a processos de negócio.

Para alcançar este objetivo, várias etapas foram cumpridas. Estas, também trouxeram contribuições isoladas, apresentadas na seqüência:

- Apresentação do estado da arte dos assuntos tratados, através das pesquisas bibliográficas individuais, servindo como embasamento para a seleção de uma metodologia que utiliza *patterns* para prover reuso de elementos de análise;
- Proposta de alteração na estrutura de documentação dos *patterns*, criando-se mais um nível de abstração para atender a domínios específicos, neste caso, processos de negócio;
- Proposta de um catálogo de *patterns* destinado a atender processos de negócio, utilizado e evoluído nos estudos de caso;

- Proposta de utilização mais efetiva dos modelos de objetos do domínio, em suas duas versões: visão geral e completo. Estes modelos contribuem com a parte estática da modelagem;
- Proposta de elaboração e utilização de cenários pertencentes a um domínio específico. Dessa forma, estes cenários estereotipados podem ser reutilizados e passam a participar da modelagem dos sistemas, aprimorando a parte dinâmica;
- Roteiro para integração da metodologia original com as etapas e diagramas propostos por esta dissertação;
- Dois estudos de caso, que validaram as propostas e evidenciaram a aplicabilidade adequada dos *patterns* para modelar processos de negócio.

Essas contribuições procuram vir ao encontro das necessidades das organizações em modelar os seus processos de negócio essenciais. Esta modelagem, por conseqüência, será automatizada e tornar-se-á um *workflow*. Pretendem servir tanto aos desenvolvedores internos de *software*, quanto aos prestadores de serviços.

Além das contribuições diretas, apresentadas anteriormente, o resultado deste trabalho procura auxiliar, indiretamente, nos seguintes aspectos:

- transmitir o conhecimento entre os projetos e entre as equipes de desenvolvimento;
- aprimorar a comunicação entre projetistas, pois pode-se discutir todo um princípio com um simples nome de *pattern* [LAR 98];
- melhorar a documentação da fase de análise, principalmente a parte dinâmica;
- propiciar maior agilidade ao projetista e melhor qualidade para os modelos, visto que não partem do zero, mas sim de um modelo predeterminado [PAL 98c];
- padronizar as especificações para *design* e implementação, internamente na organização ou objetivando terceirização destas fases.

4.9 Considerações sobre o capítulo

Neste capítulo são desenvolvidos dois estudos de caso que aplicam *patterns* a processos de negócio, objetivando, principalmente, promover reuso de elementos da análise e validar as proposições e complementações apresentadas nesta dissertação.

Ao apresentar os diagramas obtidos na modelagem, são evidenciados os resultados da aplicação dos *patterns*, quais sejam, os modelos de objetos e os cenários.

Para cada estudo de caso, é feito o mapeamento entre as classes provenientes dos *Patterns* de Processos de Negócio e as classes de cada um dos modelos. Nas considerações finais, é apresentada uma tabela que evidencia o percentual de reuso entre os componentes dos modelos. Este capítulo enumera, ainda, as contribuições obtidas com o desenvolvimento deste trabalho.

5 CONCLUSÕES

Neste trabalho são propostas diretrizes para integração de três aspectos que exercem influência no desenvolvimento de *software*: processos de negócio, reuso e *patterns*. Mais especificamente, a forma empregada para obter reuso, especialmente das fases iniciais do ciclo de vida do desenvolvimento de *software*, é através da aplicação de *patterns*. Estes, formam a base para documentação e modelagem dos processos de negócio.

Os processos de negócio, definidos pelo conjunto de atividades que realizam um objetivo de negócio ou uma meta política representam, para a engenharia de *software* um desafio, pois demandam uma fase de análise muito extensa e crítica. Como forma de automatização de processos de negócio é utilizado o conceito de *workflow*, por meio do qual documentos, informações ou tarefas transitam entre participantes, conforme regras procedimentais.

O reuso atua como um facilitador para desenvolver sistemas com maior qualidade e com menores custos, porém, para ser efetivo, demanda uma abordagem sistemática que atenda a todos os recursos do ciclo de vida do desenvolvimento. Isto somente é possível ao inserir as técnicas de reuso no processo, assegurando que o reuso seja planejado, avaliado e integrado, ao ponto de ser praticado naturalmente, conduzido pelo processo.

Em vista disso, a ênfase dada nesta dissertação é para o reuso de elementos da análise. Como exemplo, Jay Reddy em [RED 98], enfatiza que a implementação é apenas uma fração dos custos de construção e manutenção de um sistema, justificando o esforço na obtenção de reuso em todas as fases do ciclo.

Os *patterns*, integrados ao processo de desenvolvimento, são empregados para contemplar o requisito de reuso de alto nível. Atuam como estruturas a serem seguidas durante o desenvolvimento de *software*, abstraídas de práticas que foram desenvolvidas, evoluídas e que resultaram em sucesso.

Esta dissertação inicia pela exposição dos aspectos gerais de processos de negócio, reuso e *patterns*, assim como pelas considerações sobre a integração destes três elementos.

Após conceituados todos os elementos envolvidos neste trabalho, são feitas as delimitações do escopo de atuação de cada um, juntamente com a apresentação da hipótese a

ser comprovada ao término. Tal hipótese é definida como a busca do reuso de componentes de análise na modelagem de aplicativos que suportam processos de negócio.

Resultantes da revisão bibliográfica sobre *patterns*, são apresentadas as origens e definições. A origem é apontada como uma obra na área de arquitetura, proposta por Christopher Alexander em 1977, contudo não há um consenso na comunidade sobre a real origem. Já as definições variam conforme a linha de pesquisa e abstração de aplicação dos *patterns*, mas o traço comum entre elas evidencia que os *patterns* registram as idéias e alternativas de um projeto de *software* para que possam ser reutilizadas em outros, na forma de uma estrutura a ser seguida ou um plano de atuação.

Para a análise das metodologias de *patterns*, foram selecionadas as mais disseminadas na comunidade e mais referenciadas nas bibliografias. Como forma de diferenciação, foi utilizado o critério da abordagem da metodologia. O primeiro critério, metodologias dirigidas a exemplos, seleciona aquelas que aplicam os *patterns* ou *design pattern* a partir de um catálogo ou repositório, sem enfatizar o processo. Com esta abordagem, foram referenciadas: *Design pattern* [GAM 94] de Gamma et alii, POSA [BUS 96] de Buschmann et alii e Analisys *Patterns* [FOW 97a] de Fowler.

O segundo critério, metodologias dirigidas a processo, considera aquelas que apresentam os passos a serem seguidos para a concepção e elaboração dos sistemas utilizando *patterns*. Para este critério foram consideradas: Taligent [TAL 94a, TAL 94b] da empresa Taligent, *Hot spots* [PRE 95] de Pree e Estratégias e *Patterns* [COA 97] de Coad et alii. Esta última é apresentada com um grau de detalhamento muito superior ao das demais, por ter sido a metodologia selecionada para embasar as proposições.

A metodologia original [COA 97] é estendida com as seguintes propostas:

- Complementação da estrutura de documentação dos *patterns*, através da criação e modificação de alguns campos. Desta forma, é implementado um segundo nível de *patterns*, com granularidade para atender a um domínio mais específico, no caso, processos de negócio;
- Criação de um catálogo inicial com seis *Patterns* de Processos de Negócio, já no formato proposto no item anterior, com o objetivo de atender a um domínio de processos de negócio específico;

- Criação dos Modelos de Objetos do Domínio, que são resultantes da integração de todos os *Patterns* de Processos de Negócio de um domínio específico, com suas classes, seus relacionamentos e atributos. Estes modelos foram propostos em duas versões: visão geral e completo, com objetivos distintos;
- Criação dos Cenários do Domínio, que passam a receber atribuições semelhantes aos *patterns*, ao serem instanciados de um catálogo de cenários pertencentes a um domínio específico. Tornam-se estruturas genéricas onde as interações entre as classes, as trocas de mensagens, as rotas e as regras são estereotipadas;
- Por último, é proposto um roteiro para integração das novas etapas e diagramas à metodologia original.

Para validar as complementações à metodologia original, são elaborados dois estudos de caso, com base em dados obtidos de processos de negócio reais. O primeiro, gerencia o ciclo de vida de solicitações de compras ou serviços e o segundo, controla demanda e aprovação de atividades de informática.

Para cada estudo de caso são apresentados os dados básicos e escopo do sistema, os diagramas obtidos (modelos de objetos e cenários) e a avaliação dos resultados observados durante a modelagem. Nesta, é feito o mapeamento entre as classes provenientes dos *Patterns* de Processos de Negócio com as classes de cada um dos modelos.

Após os dois estudos de caso, uma tabela evidencia o percentual de reuso obtido em cada estudo de caso, juntamente com as observações feitas durante a elaboração. Com base nestas informações, evidencia-se que os *Patterns* de Processos de Negócio foram evoluídos após a sua utilização, proporcionaram reuso de elementos de análise e, juntamente com as demais propostas de complementação à metodologia original, possuem aplicabilidade para modelar processos de negócio.

A principal contribuição deste trabalho pretende ser a proposição de novas diretrizes para utilização e obtenção de *patterns* aplicados a processos de negócio. As demais contribuições diretas e indiretas também são enumeradas.

A presente dissertação cumpre uma etapa parcial do objetivo de obter reuso de elementos da fase de análise, nas aplicações de processos de negócio. As diretrizes propostas contemplam um domínio restrito, suficiente para ilustrá-las e validá-las nos estudos de caso. Entretanto, para aplicação efetiva em grandes ambientes produtivos, necessitam ser estendidas

e validadas em ciclos evolutivos de maturação, até o instante que atendam a uma parcela significativa dos processos de negócio das organizações e propiciem grandes percentuais de reuso.

Na seqüência são relacionadas as perspectivas de futuras aplicações e complementações aos assuntos abordados neste trabalho.

5.1 Trabalhos Futuros

Nesta subseção são feitas propostas para extensão desta dissertação, seja com o objetivo de dar continuidade ao trabalho atual, seja para preencher lacunas não contempladas.

A primeira proposta é a complementação dos *patterns* para o domínio ora endereçado, assim como a criação de novos Modelos de Objetos do Domínio e Cenários do Domínio. Como exemplo, poderiam ser mais profundamente abordados os modelos e cenários para Efetivação de Atividade. No estudo de caso Gerência de Compras, tal efetivação representa o processo de compra ou contratação de serviços, ao passo que para Sistemática de Trabalho representa o desenvolvimento até a conclusão de um projeto de *software*. Como conseqüência desta complementação, os percentuais de reuso podem ser melhor aferidos, visto que a amostragem dos dados tende a ser maior e pode ser mais extensivamente analisada.

Como estratégia de disseminação dos *patterns* em grandes ambientes produtivos, outra proposta de trabalho futuro seria a abordagem de outros domínio de processos de negócio, como atendimento a clientes, controle de operações, gerência de projetos etc. Dessa forma, a crescente gama de opções de modelos oferecida ao desenvolvedor deve progressivamente suprir suas necessidades, até que se atinjam bons índices de reuso e modelos de análise com qualidade. Como exemplo de processo de negócio pertencente a outro domínio, pode ser considerada a modelagem apresentada em [PAL 98a], referente a uma aplicação de gerência de projetos. Tal modelagem pode ser adaptada para adequar-se às propostas de complementação feitas por esta dissertação, dando origem aos modelos de um novo domínio.

Para ampliar as fronteiras delimitadas neste trabalho pode-se propor, também, fazer o mapeamento entre a fase de análise ora apresentada, com as fases de *design* e implementação. Para tal, deve-se levar em consideração a estrutura particular dos modelos de análise, inicialmente proposta em [COA 97]. Um trabalho que pode servir como base para esses estudos é apresentado por Adriana Thomé, em [THO 98]. Neste, a subseção "4.5-Projetando o Modelo de Objetos", faz referência ao projeto das classes e aos dados que devem persistir. Já a

subseção "4.6-Definindo um Plano de Implementação e Testes" prossegue no ciclo de desenvolvimento de *software* até as suas últimas etapas.

Como exemplo dos elementos da fase de *design*, são apresentados modelos para especificação de classes, atributos e serviços, respectivamente na Figura 35, Figura 36 e Figura 37 no Anexo 2 - Especificações de Classes, Atributos e Serviços.

Ainda fazendo referência ao mapeamento entre as fases de análise, *design* e implementação, Graig Larman, em [LAR 98], apresenta na subseção 23 ("Mapeando *designs* para código") uma proposta para mapear os diagramas obtidos na análise e *design* para a implementação. Propõe como entrada para a codificação "os artefatos da UML... – os diagramas de colaboração e os diagramas de classe" [LAR 98].

Outro trabalho futuro que complementaria o atual poderia ser o estudo do impacto da utilização dos Diagramas de Transição de Estados em conjunto com *patterns*, visto que tais diagramas raramente são referenciados quando a abordagem para modelagem dos sistemas utiliza *patterns*. Ao evidenciar as vantagens e desvantagens da sua aplicação em parceria com os *patterns*, pode-se optar por incluí-los ou não no processo de desenvolvimento, ou mesmo por utilizá-los para endereçar um subconjunto específico de elementos.

Como última extensão a este trabalho, propõe-se uma melhoria na forma de documentação e recuperação dos *patterns*. O desenvolvimento de uma ferramenta automatizada para registro, guarda e recuperação dos *patterns* poderá aprimorar o seu processo de seleção, refletindo em um projeto de desenvolvimento mais efetivo. Da mesma forma, poderiam ser endereçados os Modelos de Objetos do Domínio e os Cenários do Domínio, que também passam a demandar necessidades de armazenamento e recuperação para que possam suprir a expectativa de reuso de seus elementos.

Ainda considerando ferramentas de apoio e documentação, uma proposta feita em [CAM 97] possibilita a visualização de *design patterns* de uma maneira tridimensional, onde os eixos representam as categorias básicas de classificação. Tal estrutura de visualização pode ser adaptada para os *patterns* considerados por esta dissertação, de maneira que os eixos poderiam ser representados pelos *patterns* do catálogo original ou pelos *Patterns* de Processos de Negócio. Já a forma de visualização de cada classe pode ser em função do componente a que pertence (PD, DM etc.). Tal ferramenta pode contemplar integralmente a necessidade de

documentação ou atuar em parceria com outra especializada na guarda e recuperação dos *patterns.*

As propostas anteriores pretendem nortear as extensões a esta dissertação, sendo aderentes às diretrizes e metodologias utilizadas, às proposições feitas e aos processos de negócio.

ANEXO 1 - METODOLOGIAS DE *PATTERNS*

No capítulo 2 desta dissertação, são caracterizadas metodologias dirigidas a exemplos aquelas que possuem uma abordagem de aplicação de *patterns* a partir de um catálogo ou repositório e as dirigidas a processos aquelas que apresentam os passos a serem seguidos para a elaboração dos sistemas.

Os principais aspectos considerados como diferenciais entre as metodologiais são apresentados na Tabela 4, onde são evidenciadas as seguintes características: metodologia dirigida a processo, dirigida a exemplos, ênfase na fase de análise ou ênfase nas fases de *design* ou implementação.

Metodologias	Dirigida a Processo	Dirigida a Exemplos	Ênfase na Análise	Ênfase no *Design* ou Implement.
Design Patterns [GAM 94]		X		X
POSA [BUS 96]		X	X	X
Analysis Patterns [FOW 97a]		X	X	X
Taligent [TAL 94a, TAL 94b]	X			X
Hot Spots [PRE 95]	X			X
Estratégias e *Patterns* [COA 97]	X	X	X	

Tabela 4 – Características de Metodologias de *Patterns*.

Com base nessa tabulação, foi selecionada a metodologia Estratégias e *Patterns* [COA 97], com o objetivo de aprofundamento e complementação. A justificativa para esta seleção é mais detalhadamente apresentada na subseção 2.3, porém pode ser sumarizada em dois aspectos principais:

- possuir forte ênfase na fase de análise;
- ser dirigida tanto a exemplos, quanto a processo.

Esses aspectos vieram ao encontro das premissas apresentadas para o desenvolvimento deste trabalho, quais sejam, obtenção de reuso de elementos da fase de análise, desenvolvimento de *patterns* para atender a processos de negócio e inserção destes *patterns* no processo de desenvolvimento.

A metodologia Estratégias e *Patterns* foi selecionada por ter contemplado os requisitos específicos propostos para o desenvolvimento deste trabalho. Dessa forma, as demais metodologias citadas não são descartadas para uso na modelagem de processos de negócio. Tampouco objetivou-se, com esse levantamento, classificar a melhor metodologia sob algum aspecto ou eleger uma como sendo a mais completa.

Na seqüência, este anexo apresenta a estrutura básica e algumas características de cada uma das metodologias consideradas na Tabela 4, com exceção da Estratégia e *Patterns,* que é mais extensivamente apresentada na subseção 2.4.

1) Elementos Reutilizáveis de *Software* Orientado a Objetos [GAM 94]

A maior representante das metodologias dirigidas a exemplos é a apresentada por Gamma et alii em [GAM 94], desenvolvida a partir do conhecimento já adquirido de um determinado domínio de aplicação, que devidamente moldado e documentado, serve como embasamento para reuso em futuros sistemas ou subsistemas. Nela, são amplamente discutidas as características e composições de *design patterns,* em uma primeira etapa, seguida por um conjunto de *design patterns* prontos, testados e implementados, objetivando o seu reuso imediato em diversos domínios de aplicação.

A metodologia preocupa-se em catalogar os *design patterns*, de forma a alcançar uma padronização na análise e no *design* dos sistemas, até atingir um nível em que a nomenclatura dos *design patterns* será adotada como linguagem padrão para comunicação dentro das equipes de desenvolvimento, assim como entre os desenvolvedores e os usuários externos.

Os *design patterns* são grupados segundo dois critérios. O primeiro diz respeito ao "propósito", que diferencia o *design pattern* conforme sua destinação. O segundo é o "escopo", que separa os *design patterns* conforme a sua abrangência a objetos ou classes.

No primeiro critério, propósito, os *design patterns* são subdivididos em três tipos, quais sejam:

- Criadores (*Creational Patterns*): abstraem o processo de instanciação, auxiliando a independência do sistema, em relação à criação, composição e representação de seus objetos. Para sua formação, além de encapsularem o conhecimento relativo às classes concretas utilizadas pelo sistema, também escondem como as instâncias dessas classes são criadas e agrupadas.
- Estruturais (*Structural Patterns*): são os responsáveis por definir como as classes e objetos são compostos para formar uma estrutura maior. Utilizam herança para compor interfaces ou implementações (métodos). Exemplo de sua utilização é o agrupamento de duas ou mais classes em uma, por meio de herança múltipla. O resultado é uma classe que combina propriedades das superclasses, facilitando a utilização de bibliotecas de classes independentes.
- Comportamentais (*Behavioral Patterns*): são os responsáveis pelos algoritmos e delegação de responsabilidades entre os objetos, definindo não somente os padrões das classes e objetos, mas também os padrões de comunicação entre eles. Estes *design patterns* caracterizam fluxos de controle complexos, difíceis de serem seguidos em tempo de execução. Eles desviam a atenção do fluxo de controle, liberando-a para a interconexão entre os objetos.

Estrutura completa

Gamma [GAM 94] complementa a estrutura de *design patterns* apresentada anteriormente, e justifica que as notações gráficas são importantes mas não suficientes. Elas simplesmente capturam o produto final do processo de desenvolvimento, como o relacionamento entre as classes e objetos. Para reutilizar projetos, deve-se guardar as decisões, alternativas e soluções. Para atender a essas necessidades, projetada a seguinte estrutura, com a respectiva explicação do que cada item deve conter:

Nome do *design pattern* e classificação
Exprime a essência do *design pattern*, sendo muito importante a sua correta definição, pois irá fazer parte do vocabulário dos projetos. A classificação segue a apresentada anteriormente (propósito e escopo).

Intenção
Uma declaração que responde às seguintes perguntas: O que este *design pattern* faz? Sobre qual assunto do projeto ou problema este *pattern* trata?

Também conhecido como
Outros nomes também conhecidos para este *design pattern*, se existirem.

Motivação
Um cenário que ilustra o problema do projeto e como as estruturas de classes e objetos o solucionam. O cenário ajuda a entender as descrições mais abstratas que surgirão.

Aplicabilidade
Em quais situações o *design pattern* pode ser aplicado.

Estrutura
Uma representação gráfica das classes, utilizando a notação OMT [RUM 94]. Também utilizam-se os diagramas de interações para ilustrar as seqüências de requisições e colaborações entre os objetos.

Participantes
Relaciona as classes e/ou objetos participantes deste *design patterns* e suas responsabilidades.

Colaborações
Define como os participantes colaboram para desempenhar suas responsabilidades.

Conseqüências
Relaciona como o *design pattern* alcança seus objetivos, quais são os resultados obtidos com sua utilização e quais aspectos da estrutura do sistema podem ser utilizados independentemente.

Implementação
Lista as armadilhas, as sugestões e técnicas a serem consideradas na implementação. Também trata dos assuntos específicos de linguagens, se houver.

Código exemplo
Fragmentos de código de como implementar o *design pattern* em C++ ou *Smalltalk*.

Usos conhecidos
Exemplos de pelo menos duas implementações em sistemas reais em diferentes domínios.

***Patterns* relacionados**
Lista os *design patterns* que são intimamente ligados com este, quais as diferenças e como eles devem ser compostos.

São apresentados 23 exemplos de *design patterns*, abrangendo todos os critérios de agrupamento que propõe inicialmente. Com esses exemplos genéricos e bastante abstratos, procura atender à necessidade de todos os domínios de aplicação.

Ciclo de vida

Avaliando o ciclo de vida dos itens do catálogo de *design patterns*, [GAM 94] propõe um ciclo de construção, abrangendo as seguintes fases: Prototipação, 'Mais requisitos', Expansão, 'Mais reuso' e Consolidação, conforme caracterização a seguir.

A Prototipação é a fase do desenvolvimento em que o *software* é concebido e passa por incrementos constantes, até que atenda aos requisitos iniciais. Neste ponto, o *software* atinge a

'adolescência' sendo composto basicamente por hierarquias de classes que refletem as entidades iniciais do domínio. O principal tipo de reuso é o de caixa-branca[29].

'Mais requisitos, Expansão e Mais reuso'. 'Mais requisitos' juntamente com 'Mais reuso' alavancam a fase de expansão, na qual, após a implementação do *software*, este é forçado a atender a novos requisitos exigindo-se que produza mais reuso. Após concluída essa fase, as hierarquias de classes não mais representarão um problema do domínio, mas sim vários, e as classes irão definir muitas operações não relacionadas e variáveis instanciáveis.

A fase de Consolidação é justificada pois muitas vezes, a fase anterior torna a aplicação muito inflexível para manutenções futuras, e é onde são feitos ajustes, dando origem aos *frameworks*. Dentre os ajustes, tem-se como exemplos: separação de classes em componentes de propósito geral e específico; adequação do posicionamento dos métodos nas classes, movendo-os para cima ou para baixo na hierarquia; e racionalização das interfaces das classes. Esta fase produz diversos objetos novos, freqüentemente por decomposição de objetos existentes, sendo mais comum ocorrer o reuso do tipo caixa-preta [30], pois ele ocorre mais por composição do que por herança.

Esta metodologia é a precursora das orientadas a exemplo e uma das mais difundidas. Outras, com a mesma fundamentação, são relacionadas na seqüência.

A análise da estrutura desta metodologia resultou na evidência de dois requisitos específicos da presente dissertação que não foram atendidos: a metodologia possui forte ênfase na fase de *design* e não apresenta um processo de desenvolvimento definido e delineado.

---

[29] Gamma et alii, em [GAM 94], define reuso caixa-branca como sendo aquele obtido através da herança de classes. Define, pois, caixa-branca em termos de visão, devido ao fato do interior (métodos e atributos) das superclasses serem visíveis para as subclasses.

[30] Gamma et alii, em [GAM 94] define reuso caixa-preta aquele em que novas e mais complexas funcionalidades são obtidas através da composição de objetos e que para tal, é requerido que os objetos tenham interfaces bem definidas. O termo caixa-preta advém do desconhecimento do interior dos objetos.

2) Pattern-Oriented Software Architecture – POSA [BUS 96]

Pattern-Oriented Software Architecture (POSA) (arquitetura de *software* orientada a *patterns*) [BUS 96] propõe uma grande relação de *patterns*, de uma forma similar a [GAM 94], porém concentrando-se principalmente nos *patterns* de *design*, sendo consideravelmente mais abstrata.

Esta metodologia propõe os seus *patterns* conforme a estrutura resumida a seguir: Contexto, que descreve as situações onde o *pattern* pode ser aplicado; o Problema, que captura a essência do problema que ocorre repetidamente e a Solução, mostrando como resolver os problemas recorrentes e lidar com as forças nele envolvidas. A solução é composta por uma parte estática, que contempla a estrutura dos componentes e seus relacionamentos, e por uma parte dinâmica com o comportamento em tempo de execução.

O critério de agrupamento dos *patterns* é o seguinte:

- *Patterns* Arquiteturais, que expressam um esquema de organização estrutural e fundamental para um sistema de *software*. São conjuntos de subsistemas predefinidos, especificando seus relacionamentos e as regras para sua organização;
- *Design patterns* refinam os subsistemas ou componentes de um sistema de *software*. Descrevem uma estrutura que constitui uma solução para um problema de projeto geral dentro de um contexto particular;
- Idiomas, que são *patterns* de um nível mais baixo, específicos para uma determinada linguagem de programação. Descrevem a forma de implementar um aspecto particular de um componente, ou o relacionamento entre componentes, usando características de uma linguagem específica.

Os *patterns* não são aplicados isoladamente, pois um único *pattern* não é capaz de resolver todos os problemas de *software*, em todos os níveis de abstração. Portanto, existem relacionamentos entre os *patterns* que os tornam fortemente conectados. A utilização de um *pattern* para solucionar um problema mais genérico pode originar um outro problema que também pode ser solucionado por um *pattern* mais específico, ou por uma combinação deles.

O relacionamento entre os *patterns* pode ocorrer da seguinte forma:

- Refinamento, dando suporte na implementação de um *pattern*;
- Combinação, ajudando na composição de estruturas complexas de projeto;
- Variações, auxiliando na seleção de um padrão mais adequado dentro de uma situação específica de projeto.

A análise da estrutura desta metodologia evidenciou a ausência de um dos requisitos específicos da presente dissertação, qual seja, a presença de um processo de desenvolvimento definido. Outro enfoque que não condiz com a proposta de reuso de análise é a forte característica que a metodologia possui em endereçar *design* e arquitetura.

3) Analysis Patterns [FOW 97a]

Analysis Patterns [FOW 97a] endereçam os objetos do domínio de forma paralela a POSA, ou seja, mais abstrata que mostrado em [GAM 94], porém a separação de seus *patterns* é feita em função do domínio, como inventário, contabilidade e finanças.

Esses *patterns* são conceituais, com o objetivo de representar a forma de pensar do usuário em detrimento da forma utilizada pelo computador. Para demonstrar como os *patterns* de análise se encaixam nas arquiteturas dos sistemas de informação, são propostos *patterns* de suporte. Estes também auxiliam a construção de modelos conceituais para se tornarem interfaces de *software* e implementações, assim como estabelecer o relacionamento entre as estruturas mais simples e as estruturas mais avançadas.

A análise da estrutura desta metodologia evidenciou um enfoque mais próximo da fase de análise do que [BUS 96], contudo não apresenta um processo de desenvolvimento definido para integração dos *patterns*, conforme requisito apresentado pela presente dissertação.

4) Metodologia Taligent [TAL 94a, TAL 94b]

Esta metodologia é apresentada através de dois artigos [TAL 94a] e [TAL 94b], onde são observados dois aspectos característicos. O primeiro está relacionado ao fato de que a metodologia direciona à construção de *frameworks* pequenos, que por sua vez formarão outros *frameworks* maiores. O objetivo é ampliar as opções de reuso, pois tanto os *frameworks*

pequenos quanto os maiores podem ser reutilizados. O segundo aspecto é que esta metodologia apresenta uma linha bem definida dos passos a serem seguidos para construção do *framework*, assim como sugere a utilização de um ambiente de desenvolvimento particular e otimizado para a metodologia Taligent [31].

Segundo [TAL 94a], para que um projeto tenha sucesso, o *framework* deve ter sido desenvolvido considerando-se os seguintes aspectos:

- Completo - o *framework* deve suprir todos os requisitos da aplicação e conter implementações embutidas sempre que possível. Deve ter implementadas derivações concretas das classes abstratas e funções *default* [32], para facilitar o entendimento do desenvolvedor e liberá-lo para concentrar o seu foco nas áreas que realmente necessita customizar.
- Flexível - as abstrações podem ser utilizadas em diversos contextos.
- Extensível - o *framework* deve permitir a adição ou modificação fácil de funcionalidades. Deve permitir customização do comportamento através da derivação de novas classes.
- Compreensível - o desenvolvedor deve ter uma interação clara com o *framework*, que deve ser bem documentado, seguindo padrões e provendo exemplos da utilização e implementação. Este é um ponto importante, principalmente se as equipes que desenvolvem e utilizam o *framework* são diferentes.

Esta metodologia propõe uma hierarquia dos elementos de um *framework*, onde a raiz é a aplicação obtida através do reuso, sendo formada por *frameworks*, que são formados por *patterns*, que, por sua vez, são formados por classes abstratas, conforme representado na Figura 33.

---

[31] A metodologia pertence a Taligent, Inc., uma joint venture independente da Apple Computer, Inc., IBM Corporation e Hewlett-Packard Company. Possui um caráter mais comercial que as demais.
[32] Default - valor básico, predefinido, implícito ou padrão.

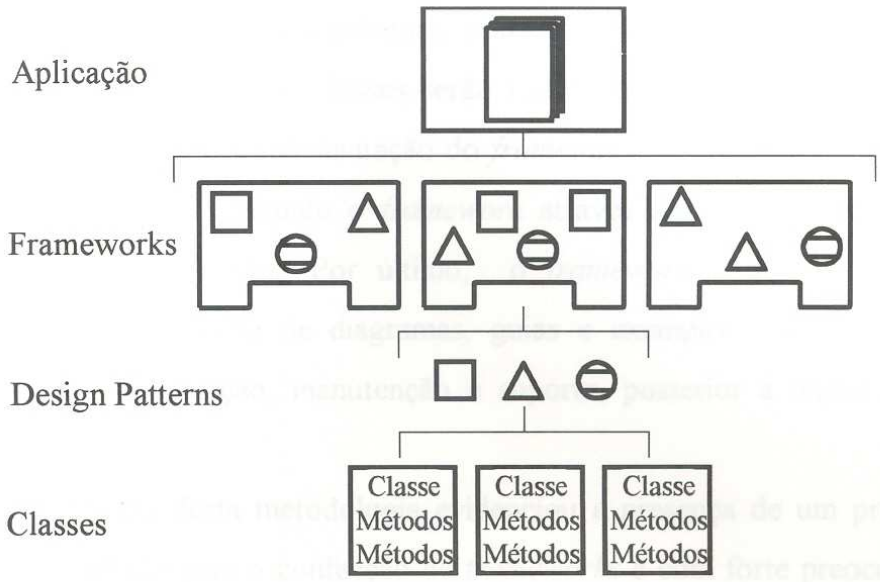


Figura 33 - Esquema de decomposição dos *Frameworks* [TAL 94a].

Em [TAL 94a] é proposta uma representação mais simplificada para *design patterns* orientados a objetos, composta de quatro itens:

1) Precondições: define quais *design patterns* devem ter sido satisfeitos antes do atual ser utilizado. A seqüência de utilização dos *design patterns* é a habilidade mais importante para o especialista desenvolvedor.

2) Problema: sumariza os problemas atendidos pelo *design pattern*. Este item auxilia a determinar se é aplicável no domínio pretendido ou não.

3) Restrições (*Constraints*): descreve as forças conflitantes atuando na solução do problema. Exemplos típicos são tempo de execução e espaço para execução, ou tempo de desenvolvimento e complexidade do programa. É importante definir as prioridades entre os conflitos para facilitar a escolha dos *design patterns*.

4) Solução: contém o sumário da solução para o problema apresentado anteriormente. Esta normalmente acompanha um diagrama demonstrando as atividades requeridas para transformar o sistema que não satisfaz em um que satisfaça os requisitos.

Os passos a serem seguidos são quatro: identificar e caracterizar o domínio do problema, definir a arquitetura e o projeto, implementar o *framework* e desmembrar o *framework*.

No primeiro passo, o de identificação do domínio do problema, são definidas as abstrações chave, levantadas as soluções existentes e é examinado o processo, definindo em quais de suas partes o *framework* irá atuar.

No passo seguinte, definição da arquitetura, é identificada a forma de interação com o *framework*, como, por exemplo, quais classes serão instanciadas, quais serão derivadas ou quais funções serão invocadas. A implementação do *framework* ocorre com a implementação das classes-núcleo, testando e validando o *framework* através das iterações que refinam o projeto e adicionam funcionalidades. Por último, o *framework* é desmembrado com a documentação detalhada em forma de diagramas, guias e exemplos. Nesta fase, ainda é definido o processo de distribuição, manutenção e suporte, posterior à implementação do *framework*.

A análise da estrutura desta metodologia evidenciou a presença de um processo bem definido, porém mais voltado para a confecção de *frameworks* e com forte preocupação com implementação. A ausência de um catálogo de *patterns* não satisfaz este requisito específico imposto pela presente dissertação.

5) Metodologia Dirigida a *Hot Spots* [PRE 95]

Outra metodologia expressiva é a apresentada por Wolfgang Pree, em [PRE 95], que utiliza a abordagem de *hot spot*[33]. Seu núcleo é a utilização de *metapatterns*, representando um conjunto de *design patterns* que descrevem a forma de construir *frameworks*, independentemente de um domínio específico, objetivando categorizar e descrever qualquer *design pattern* formador de um *framework* em um metanível (*metalevel*). Estes não substituem a abordagem que utiliza *design pattern*, mas complementam-na. Esta metodologia pode ser resumida em quatro etapas, apresentadas na Figura 34 e descritas na seqüência:

- Identificação das classes e objetos: desempenhada pelo especialista do domínio em conjunto com o engenheiro de *software*, valendo-se das metodologias tradicionais orientadas a objetos.
- Identificação dos *hot spots*[34]: desempenhada pela mesma equipe responsável pela etapa anterior.

---

[33] A metodologia chama-se *hot spot driven*, mas *hot spot* não é utilizado com a sua tradução literal, e sim significa "parte a ser flexibilizada".

[34] [PRE 95] "A dificuldade de um bom projeto orientado a objetos é a identificação dos *hot spots* em um *framework*, ou seja, aqueles aspectos do domínio da aplicação que devem ser mantidos flexíveis. Nós consideramos que um *framework* tem o atributo de qualidade 'bem projetado' se ele provê *hot spots* adequados para adaptações."

- Projeto/reprojeto do *framework*: nesta etapa o engenheiro de *software* reprojeta as classes para torná-las flexíveis, de acordo com os *hot spots*. É a etapa onde os *metapatterns* são aplicados.
- Adaptação do *framework*: nesta etapa o especialista do domínio retorna para fazer refinamentos. Ao final há um ponto de verificação, onde é evidenciado se os *hot spots* foram satisfatoriamente projetados, encerrando o processo se houve sucesso e retornando para a segunda fase, no caso de fracasso.

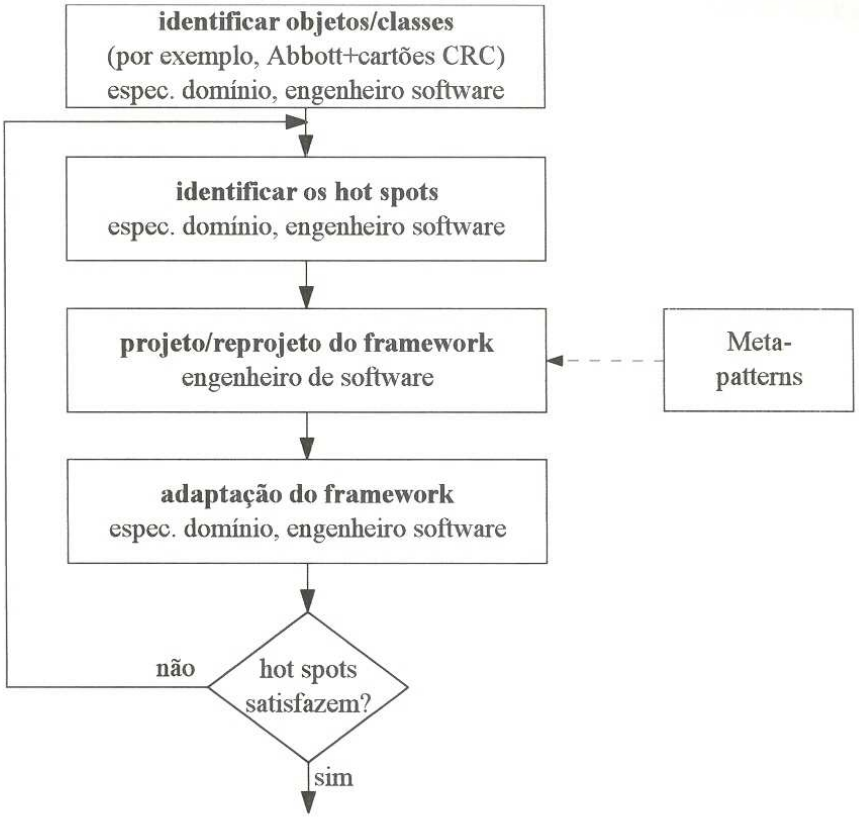


Figura 34 – Abordagem dirigida a *hot spots* [PRE 95].

A análise da estrutura desta metodologia possui similaridades com a análise feita sobre [TAL 94a, TAL 94b]. Destaca-se, também, o forte embasamento do processo e a ausência de um catálogo de *patterns,* vindo de encontro aos requisitos específicos desta dissertação.

ANEXO 2 - ESPECIFICAÇÕES DE CLASSES, ATRIBUTOS E SERVIÇOS.

Este anexo objetiva fazer uma proposição para o mapeamento entre a fase de análise, enfatizada por esta dissertação, com as fases de *design* e implementação, levando-se em consideração a estrutura particular dos modelos de análise proposta em [COA 97].

Tomando-se como base o trabalho desenvolvido por Adriana Thomé, em [THO 98], são apresentadas especificações de classe, atributo e serviço, respectivamente na Figura 35, Figura 36 e

Figura 37.

Nome da Classe : ContadorInteiro
Componente à que Pertence : CDP (Objeto PD ou Componente PD nesta dissertação)
Descrição : Classe que representa um contador inteiro, que é capaz de se incrementar, se decrementar, se inicializar, e de mostrar seu valor na base hexadecimal. É uma especialização da classe "Contador".
Restrições de Criação : Só pode existir "1" objeto ContadorInteiro.
Atributos : valor (Herdado de "Contador") valorInicial (Herdado de "Contador")
Serviços : incrementar (Redefinido de "Contador") decrementar (Redefinido de "Contador") inicializar (Herdado de "Contador") mostrarBase criarContadorInteiro (p) liberarContadorInteiro (p) acessarValor (p) (Herdado de "Contador") acessarValorInicial (p) (Herdado de "Contador") setarValor (p) (Herdado de "Contador") setarValorInicial (p) (Herdado de "Contador")

Figura 35 - Especificação para uma Classe [THO 98].

Nome do Atributo : valor				
Descrição : Atributo que contém o valor do contador.				
Tipo : Dependente das classes de especialização.				
Valor *Default* : ---------				
Atributo Discreto : Sim, seu domínio de valores se restringe aos números inteiros.				
Atributo Contínuo : ----------				
Atributo Monovalorado	X	Atributo Multivalorado :		
Detalhes Adicionais : É um atributo obrigatório, não determinante, público para leitura, privado para escrita.				

Figura 36 - Especificação para um Atributo [THO 98].

Nome do Serviço: incrementar
Função: Incrementa de 1 o valor do contador.
Parâmetros de Entrada: valor
Parâmetros de Saída: valor
Sub-Serviços Referenciados: ----------
Diagrama de Estrutura dos Módulos: ----------
Descrição em Português Estruturado: INCREMENTAR incrementar de 1 o "valor" do contador FIM-INCREMENTAR

Figura 37 - Especificação para um Serviço [THO 98].

GLOSSÁRIO

BPR – *Business Process Re-engineering*, Re-engenharia de Processos de Negócio: Atividades de verificação, análise, modelagem, definição e subseqüente implementação operacional dos processos de negócio essenciais de uma organização [WFM 96, LAW 97].

Cenário: uma seqüência de eventos que ocorrem durante uma determinada execução de um sistema. Cada evento transmite informações de um objeto para outro. Os cenários dão origem ao Diagrama de Eventos. Estas definições são apresentadas por [RUM 91] e diferem dos conceitos dos Cenários de *Patterns* utilizados nesta dissertação.

Cenário de *Pattern*: uma seqüência temporalmente ordenada de interações entre objetos, contendo as mensagens que os objetos emitem e recebem, assim como os serviços invocados em sua decorrência [COA 97].

Cenário do Domínio: é um cenário de *pattern* com uma estruturas estereotipadas que permite a sua instanciação, em um domínio de processo de negócio específico. Tal cenário, proposto por esta dissertação, propicia o reuso dos seguintes elementos: interações entre as classes, as trocas de mensagens e as regras e rotas dos processos de negócio.

Design: fase posterior à análise, também denominada fase de projeto, dentro do ciclo de vida de desenvolvimento de *software*. Tem como foco a estrutura de dados, arquitetura do *software*, detalhes procedimentais e caracterização de interfaces. Traduz os requisitos em uma representação do *software* que pode ser avaliada, anteriormente à fase de codificação [PRE 92].

Domínio do problema, *Problem-Domain* ou PD: componente da metodologia que contém os objetos que correspondem diretamente ao domínio do problema sendo modelado [COA 97].

***Framework*:** projeto para um conjunto de objetos que cooperam para alcançar um conjunto de responsabilidades. *Frameworks* são projetos de maior escala do que classes abstratas e promovem uma forma de reuso do conteúdo intelectual de um *software*. [JOH 92]

Gerenciamento de Dados, *Data-Management* ou DM: objetos que fornecem a interface entre os objetos do domínio do problema que necessitam persistência e as bases de dados e gerenciadores de arquivos [COA 97].

GoF: forma comumente utilizada para fazer referência ao livro *Design patterns, Elements of Reusable Object-Oriented Software* [GAM 94], devido aos seus quatro autores.

Groupware: termo utilizado para as tecnologias que suportam a colaboração entre pessoas. Provê ferramentas para solucionar problemas de negócios orientados a colaboração e, como exemplo, pode ser considerado desde *e-mail*, passando por sistemas de reuniões eletrônicas até chegar a *workflow* [COL 95].

Interação com Sistemas, *System-Interaction* ou SI: objetos que fornecem interface entre objetos do domínio do problema com outros sistemas ou equipamentos, encapsulando os protocolos de comunicação e os detalhes de baixo nível [COA 97].

Interação Humana, *Human-Interaction* ou HI: componente que contém os objetos que disponibilizam a interface entre o domínio do problema e as pessoas (normalmente objetos tela e relatório) [COA 97].

Método (no contexto da engenharia de *software*): provê a forma de construir um *software*, agrupando um vasto conjunto de tarefas que incluem estimativa e planejamento de projetos, análise dos requisitos do sistema e do *software*, codificação, teste e manutenção. Geralmente inclui uma notação gráfica ou textual e um conjunto de critérios de qualidade. [PRE 92]

Modelo de Objetos (*object model*): descrevem os tipos de objetos do sistema e seus relacionamentos estáticos. Os relacionamentos podem ser do tipo associação ou subtipo [FOW 97b]. Esta definição refere-se ao Diagrama de Classe da UML, referenciado como Modelo de Objetos na presente dissertação.

Modelo de Objetos do Domínio: é um Modelo de Objetos específico para um determinado domínio. Origina-se do agrupamento dos *Patterns* de Processos de Negócio e representa um diagrama com todas as classes, seus relacionamentos e suas multiplicidades. Tal diagrama, proposto por esta dissertação, é apresentado em duas modalidades: visão geral e completo.

Papéis (*roles*): possibilitam vincular as pessoas com as atividade que desempenham, em *workflows*. Uma pessoa pode desempenhar diversos papéis em uma estrutura, dependendo da rota e das informações sobre as atividades. Assim como um papel pode ser designado para diversas pessoas. [COL 95].

Pattern: é um plano ou uma estrutura para ser seguida durante a construção de *software*, uma abstração oriunda de observações da realidade e um exemplo digno de ser emulado [COA 97].

***Patterns* de Processos de Negócio:** são *patterns* que possuem abrangência delimitada, são documentados de forma padronizada e endereçam problemas específicos no contexto de processos de negócio. Estes *patterns* são propostos por esta dissertação e formam um catálogo inicial para, juntamente com os modelos do domínio, atender aos processos de negócio.

POSA: *Pattern-Oriented Software Architecture* ou Arquitetura de *software* orientada a *patterns*, metodologia proposta por Buschmann et alii [BUS 96].

Processo de Desenvolvimento de *Software*: conjunto de atividades, métodos, práticas e transformações empregadas pelas pessoas para desenvolver e manter *software*, assim como seus produtos associados (planos de projeto, documentos de *design*, código, casos de teste e manuais) [CMU 94].

Processo de Negócio ou *Business Process*: Conjunto de um ou mais procedimentos ou atividades que coletivamente realizam um objetivo político ou do negócio, normalmente no contexto de uma estrutura organizacional, definindo papéis funcionais e relacionamentos [WFM 96, LAW 97].

Regras (*rules*): determinam para quem e quais informações serão roteadas em um *workflow*. Desempenham a função de capturar as suposições sobre como conduzir o negócio. [COL 95].

Rotas (*routs*): representa o fluxo percorrido por objetos dentro de um *workflow*, conforme a determinação de condições e regras. As rotas podem ser seqüenciais, paralelas, *broadcast* ou *ad hoc*. [COL 95].

WfMS *Workflow Management System* (Sistema de Gerenciamento de *Workflow*): Um sistema que define, cria e gerencia a execução de um *workflow* através do uso de *software*, que é capaz de interpretar as definições de processo, interagir com os participantes e invocar recursos e ferramentas da tecnologia de informação. [WFM 96, LAW 97].

Workflow: a automação de um processo de negócio, em parte ou totalmente, durante o qual documentos, informações ou atividades são passadas de um participante para outro, de acordo com regras procedimentais [WFM 96, LAW 97].

REFERÊNCIAS BIBLIOGRÁFICAS

[BEC 94] BECK, KENT; JOHNSON, RALPH. **Patterns Generate Architecture**. In: European Conference on Object-Oriented Programming (ECOOP), 1994, Bologna. Proceedings... Bologna, 1994.

[BUS 96] BUSCHMANN, FRANK; MEUNIER, REGINE; ROHNERT, HANS; SOMMERLAD, PETER; STAL, MICHAEL. **Pattern-Oriented Software Architecture: A System of Patterns**. Chichester: John Wiley & Sons, 1996. 457 p.

[CAM 91] CAMPBELL, ROY; ISLAM, NAYEEM, JOHNSON, RALPH; KOUGIOURIS, PANOS; MADANY, PETER. **Choices, Frameworks and Refinement**. University of Illinois at Urbana-Champaign. Department of Computer Science, 1991.

[CAM 97] CAMPO, MARCELO; PRICE, ROBERTO; TEYSEYRE, ALFREDO. **Uma Abordagem 3D para a Visualização de Padrões de Projeto.** In: Simpósio Brasileiro de Engenharia de Software, 11., 1997, Fortaleza. Anais. Fortaleza - Brasil, SBC, 1997. p. 81-96.

[COA 91a] COAD, PETER;YOURDON, EDWARD (1991) **Projeto Baseado em Objetos**. Tradução de Vanderberg Dantas de Souza. Rio de Janeiro: Campus, 1993. 195 p.

[COA 91b] COAD, PETER; YOURDON, EDWARD (1991) **Análise Baseada em Objetos**. Tradução CT Informática. Rio de Janeiro: Campus, 1996. 225 p.

[COA 97] COAD, PETER; MAYFIELD, MARK. **Object Models: Strategies, Patterns & Applications - Second Edition**. New Jersey: Yourdon Press, 1997. 515 p.

[COL 95] COLEMAN, DAVID; KHANNA, RAMAN. **Groupware: Technology and Applications.** Upper Saddle River: Prentice Hall , 1995. 576 p.

[ERI 98] ERIKSSON, HANS-ERIK; PENKER, MAGNUS; **UML Toolkit.** New York: John Wiley & Sons, 1998. 397 p.

[FER 98] FERREIRA, LUCIANE LAMOUR; RUBIRA, CECÍLIA MARY FISCHER. **Padrão State Reflexivo: Refinamento do Padrão de Projeto State para uma Arquitetura Reflexiva**. In: Simpósio Brasileiro de Engenharia de Software, 12., 1998, Maringá. Anais. Maringá, Brasil, 1998, p. 139-153.

[FOW 97a] FOWLER, MARTIN. **Analysis Patterns. Reusable object models.** Menlo Park: Addison-Wesley, 1997. 357 p.

[FOW 97b] FOWLER, MARTIN; SCOTT, KENDALL. **UML Distilled. Applying the Standard Object Modeling Language.** Reading: Addison-Wesley, 1997. 179 p.

[FUR 98] FURLAN, JOSÉ DAVI; **Modelagem de Objetos através da UML.** São Paulo: Makron Books, 1998. 329 p.

[GAM 94] GAMMA, ERICH; HELM, RICHARD; JOHNSON, RALPH; VLISSIDES, JOHN. Design patterns: **Elements of Reusable Object-Oriented Software**. Reading: Addison-Wesley, 1994. 395 p.

[JOH 88] JOHNSON, RALPH; FOOTE, BRIAN. **Designing Reusable Classes**. Journal of Object-Oriented Programming. June/July, 1988.

[JOH 92] JOHNSON, RALPH. **Documenting Frameworks using Patterns**. In: Object-Oriented Programming Systems, Languages, and Applications Conference (OOPSLA), 1992, Vancouver. Proceedings... Vancouver - Canada, 1992.

[KOR 96a] KORSON, TIM. **Introduction to Reuse**. Object Magazine, v. 6, n. 1, April 1996. p. 32-33.

[KOR 96b] KORSON, TIM. **Managing Reuse: Applying the law of gravity.** Object Magazine, v. 6, n. 1, April 1996. p. 34-36.

[KRU 92] KRUEGER, CHARLES. **Software Reuse.** ACM Computing Surveys, v. 24, n. 2, June 1992. p. 131-183.

[LAJ 94] LAJOIE, RICHARD; KELLER, RUDOLF. **Design and Reuse in Object-Oriented Frameworks: Patterns, Contracts, and Motifs in Concert**. In: 62nd Congress of the Association Canadienne Française pour lÁvancement des Sciences (ACFAS), 1994, Montreal. Proceedings... Montreal - Canada, 1994.

[LAR 98] LARMAN, CRAIG. **Applying UML and Patterns – An Introduction to Object-Oriented Analysis and Design.** New Jersey: Prentice Hall PTR, 1998. 507 p.

[LAW 97] LAWRENCE, PETER (Ed.). **Workflow Handbook 1997.** Chichester: John Wiley & Sons, 1997. 501 p.

[LOT 95] LOTUS DEVELOPMENT CORPORATION. **Groupware: Communication, Collaboration and Coordination**. Cambridge, 1995. 59 p.

[MCG 96] MCGREGOR, JOHN; DODLE, JIM; KEDDY, ASHA. **A Pattern for Reuse – Let Architectural reuse guide component reuse.** Object Magazine, v. 6, n. 1, April 1996. p. 38-47.

[MAR 98] MARTIN, R.; RIEHLE, D.; FUSCHMANN, F. (Eds.). **Pattern Language of Program Design 3.** Reading: Addison-Wesley, 1998.

[PAL 98a] PALUDO, MARCO; BURNETT, ROBERT; JAMHOUR, EDGARD. **A Project Management modeling with pattern-oriented approach.** In: Simposio Argentino de Orientación a Objetos (ASSO98), 2., 1998, Buenos Aires. Proceedings... Buenos Aires – Argentina, SADIO, 1998, p. 23-33.

[PAL 98b] PALUDO, MARCO; BURNETT, ROBERT; JAMHOUR, EDGARD. **Patterns in CSCW Modeling**. In: Simpósio Brasileiro de Engenharia de Software, 12., 1998, Maringá. Anais. Maringá - Brasil, SBC, 1998. p. 37-52.

[PAL 98c] PALUDO, MARCO; BURNETT, ROBERT; JAMHOUR, EDGARD. **Patterns in CSCW Modeling**. In: California Software Symposium, 8., 1998, Irvine. Proceedings... Irvine - United States of America, 1998. p. 3 - 13.

[PAU 94] PAULK, MARK; WEBER, CHARLES; CURTIS, BILL; CHRISSIS, MARY. EDS. **The Capability Maturity Model: Guidelines for Improving the Software Process.** Reading: Addison-Wesley, 1994. 441 p.

[POM 98] POMPERMAIER, LEANDRO; PRICE, ROBERTO. **Um Editor Diagramático para Desenvolvimento de Sistemas de Informação na Internet.**. In: Simpósio Brasileiro de Engenharia de Software, 12., 1998, Maringá. Anais. Maringá - Brasil, SBC, 1998. p. 265-277.

[PRE 95] PREE, WOLFGANG. **Design patterns for object oriented software development**. Wokingham: Addison-Wesley, 1995. 268 p.

[PRE 96] PREE, WOLFGANG. **Frameworks - Past, Present, Future**. Object Magazine, v. 6, n. 2, May 1996. p. 24-26.

[PRE 92] PRESSMAN, ROGER. **Software Engineering: A Practitioner's Approach.** Singapore: McGraw-Hill, 1992. 703 p.

[RED 98] REDDY, JAY. **Making Reuse Work**. In: 5th International Conference on Software Reuse (ICSR), 1998, Victoria. Proceedings... Victoria - Canada, 1998, p. 362-363.

[RUM 94] RUMBAUGH, JAMES; BLAHA, MICHAEL; PREMERLANI, WILLIAM; FREDERICK, EDDY; LORESEN, WILLIAM. **Modelagem e Projetos Baseados em Objetos**. Tradução de Dalton Conde de Alencar. Rio de Janeiro: Campus, 1994. 652p.

[SCH 98] SCHREYJAK, STEFAN. **Using Components in Workflow Activities.** In: Second and Third International Workshop on Business Object (OOPSLA), 1998, Springer Verlag. Proceedings... Springer Verlag, 1998.

[TAL 94a] TALIGENT. **Building Object-Oriented Frameworks**. Taligent, Inc White Paper, 1994. 20 p. Disponível na internet em http://www.taligent.com.

[TAL 94b] TALIGENT. **Leveraging Object-Oriented Frameworks**. Taligent, Inc White Paper, 1994. 16 p. Disponível na internet em http://www.taligent.com.

[THO 98] THOMÉ, ADRIANA CURSINO. **Desenvolvimento de Software por Engenheiros: Diretrizes e Metodologias.** São José dos Campos, 1998. Dissertação (Mestrado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais. 333 p.

[UFP 95] UNIVERSIDADE FEDERAL DO PARANÁ. **Normas para apresentação de trabalhos - 5. ed.** Curitiba: Ed. da UFPR, 1995.

[UML 97a] Rational Software et alii. **Unified Modeling Language – UML Semantics version 1.1**. September, 1997. Disponível na internet em http://www.rational.com/uml.

[UML 97b] Rational Software et alii. **Unified Modeling Language – UML Notation Guide version 1.1**. September, 1997. Disponível na internet em http://www.rational.com/uml.

[WFM 95] WORKFLOW MANAGEMENT COALITION. **Workflow Management Coalition: The Workflow Reference Model**. Document number TC-00-1003. January, 1995.Disponível na internet em http://www.aiim.org/wfmc/standards/docs/tc003v11.pdf.

[WFM 96] WORKFLOW MANAGEMENT COALITION. **Workflow Management Coalition: Terminology & Glossary**. Document number WFMC-TC-1011. June, 1996. Disponível na internet em http://www.aiim.org/wfmc/standards/docs/glossary.pdf.