BRENO BATISTA MACHADO

ADAPTAÇÃO DO MÉTODO DE MODELAGEM DE NEGÓCIOS ERIKSSON-PENKER UTILIZANDO A UML 2.0: ESTUDO DE CASO EM UMA FÁBRICA DE SOFTWARE

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do Curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

LAVRAS

MINAS GERAIS – BRASIL

2006

BRENO BATISTA MACHADO

ADAPTAÇÃO DO MÉTODO DE MODELAGEM DE NEGÓCIOS ERIKSSON-PENKER UTILIZANDO A UML 2.0: ESTUDO DE CASO EM UMA FÁBRICA DE SOFTWARE

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do Curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Área de concentração:

Engenharia de Software e Modelagem de Negócios

Orientadora:

Prof[a]. Olinda Nogueira Paes Cardoso

LAVRAS

MINAS GERAIS – BRASIL

2006

Ficha Catalográfica preparada pela Divisão de Processo Técnico da Biblioteca Central da UFLA

Machado, Breno Batista

Adaptação do Método de Modelagem de Negócios Eriksson-Penker Utilizando a UML 2.0: Estudo de Caso em uma Fábrica de Software / Breno Batista Machado – Minas Gerais, 2006.

Monografia de Graduação – Universidade Federal de Lavras. Departamento de Ciência da Computação.

1. Informática. 2. Engenharia Software. 3. Modelagem de Negócios. I. MACHADO, B. B. II. Universidade Federal de Lavras. III. Título.

BRENO BATISTA MACHADO

ADAPTAÇÃO DO MÉTODO DE MODELAGEM DE NEGÓCIOS ERIKSSON-PENKER UTILIZANDO A UML 2.0: ESTUDO DE CASO EM UMA FÁBRICA DE SOFTWARE

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do Curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Aprovada em 22 de Setembro de 2006.

_____

Prof. André Luiz Zambalde

_____

Prof. Antônio Maria Pereira de Resende

_____

Prof[a]. Olinda Nogueira Paes Cardoso

(Orientadora)

LAVRAS

MINAS GERAIS – BRASIL

2006

Aqueles que sempre acreditaram em mim, meus pais.

Agradecimentos

Primeiramente agradeço à Deus!

Agradeço também a todos, que de uma forma ou de outra, contribuíram para que eu me tornasse o que sou hoje (Bacharel em Ciência da Computação):

Aos meus pais, Benicio e Neumann, por serem os meus heróis, a quem sempre eu me espelhei. Muito obrigado por terem acreditado em mim e por terem me proporcionado esta oportunidade.

À minha irmã, Ninha, pelo carinho e atenção. Pelos brigadeiros, pipocas, filmes, bailes...

Aos meus avós, Quinca e Tália (que olham por mim lá de cima), Machadinho e Carminha, por todo o amor, carinho e “paparicos” de avós.

Aos meus tios, primos e parentes, pelas alegrias que sempre me proporcionaram.

Aos meus amigos de Campo do Meio, Duardo, Ernani, Coelhão e Francinho, pelas ótimas farras que fizemos e pelas que ainda virão.

A minha namorada, Renata (gatinha mais linda do mundo), pelo amor, carinho, atenção, ajuda, loucuras. Não teria conseguido sem você. Te amo!

A todos os colegas de república que morei, por me ensinarem a conviver com pessoas diferentes, sob o mesmo teto. E também não vou esquecer das várias festas e pórres.

Ao pessoal da Turma 11 de Computação. Vocês são feras. Vou sentir saudades.

Ao grupo de caneladas, Vanessa B., Vanessa M., Lelet’s e Felipe, por terem me ensinado a ser canelada (sendo que alguns destes, eu que levei para o mau caminho. hehehe). Ah, e várias noites em claro.

Aos inesquecíveis amigos que fiz aqui em Lavras, em especial Leo, Di, Wanda, Glasi, Almir. “Amigo é coisa para se guardar debaixo de sete chaves (...)”.

A família SW (SWQuality, SWFactory e SWLearning), por além de todo o carinho, por ter sido minha segunda faculdade, onde aprendi muitas coisas. A equipe SANF a mais de 3 anos na ativa.

Ao pessoal da Montreal BH, Lucia, Lucélia, Ricardinho e Leo.

À Ana Cristina, por ter me orientado maior parte da minha graduação, por sua amizade e por ter sido minha segunda mãe.

À Olinda, minha orientadora, por ter ajudado a turma sempre quando precisado, e por ter topado me orientar (me salvado).

A todos os meus professores por terem me ensinado a distinguir entre o certo e o errado.

E finalmente ao Google, por sempre responder as minhas perguntas e ter sido tão prestativo na ajuda do desenvolvimento de diversos trabalhos.

ADAPTAÇÃO DO MÉTODO DE MODELAGEM DE NEGÓCIOS ERIKSSON-PENKER UTILIZANDO A UML 2.0: ESTUDO DE CASO EM UMA FÁBRICA DE SOFTWARE

RESUMO

Este trabalho apresenta a adaptação do método de modelagem de negócios Eriksson-Penker, que utiliza a UML, para a UML 2.0. Um modelo de negócio é uma abstração de como um negócio funciona e traz uma visão simplificada de uma realidade complexa, eliminando detalhes e focando nos aspectos mais importantes do negócio. O objetivo da utilização da UML no trabalho é demonstrar um mecanismo de modelagem de negócio próximo aos mecanismos utilizados para modelagens de sistemas de software. A adaptação do método para utilização da UML 2.0 foi feita através da criação de *profile*, que é um mecanismo da UML 2.0 que possibilita o agrupamento dos elementos criados para poder realizar a modelagem. Além de proporcionar a realização deste trabalho, o *profile* criado pode ser utilizado em trabalhos futuros. Foi realizado um estudo de caso em uma fábrica de software, aplicando o método adaptado a fim de comprovar os resultados do trabalho. O ganho maior da modelagem de negócios para a fábrica de software, foi auxiliar na definição de seus objetivos e de sua estrutura.

Palavras-Chave: Engenharia de Software, Modelagem de Negócios, UML 2.0.

ADAPTATION OF THE ERIKSSON-PENKER BUSINESS MODELING METHOD USING THE UML 2.0: CASE STUDY IN A SOFTWARE FACTORY

ABSTRACT

This work presents an adaptation of the method of Erinksson-Penker business modeling which uses the UML to use UML 2.0. A business model is an abstraction of as a business runs and shows a simplified vision of a complex reality. It removes details and focusing in the most important aspects of the business. The objective to use UML at the work is demonstrating a mechanism of business modeling near to the mechanisms used to model software systems. The adaptation of the method to use UML 2.0 was done through the creation of a profile, which is a mechanism of the UML 2.0 that makes possible grouping the elements created to be able to carry out the modeling. As well providing the realization of this work, the created profile can be used in future works. It was carried out a case study in a software factory, applying the method adapted in order to prove the results of the work. The biggest profit of the business modeling to the software factory where the case of use was carried out went to help it at the definition of its objectives and structure.

Key-Words: Software Engineer, Business Modeling, UML 2.0.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

BSDM – Business System Development Method

EKD - Enterprise Knowledge Development

MOF - Meta-Object Facility

OCL - Object Constraint Language

OMG - Object Management Group

OMT - Object Modeling Technique

OOSE – Object Oriented Software Engineering

UML - Unified Modeling Language

1. INTRODUÇÃO

1.1. Contextualização e Motivação

Após a padronização da *Unified Modeling Language* (UML) pela *Object Management Group* (OMG) em novembro de 1997, houve um impacto muito grande no setor de desenvolvimento de software. A aceitação da UML estimulou a produção de uma nova geração de ferramentas e processos que usam a UML proporcionando o aparecimento de importantes conceitos e técnicas com regras de arquitetura, engenharia de requisitos e integração de ferramentas (Eriksson & Penker, 2000).

Entretanto, um problema que surge na área de desenvolvimento de software é o fato dos sistemas de software não suportarem corretamente os negócios[1] do qual eles estão inseridos por não atenderem totalmente as necessidades destes. Há varias razões para isto, por exemplo, não é feita uma correta especificação de requisitos do sistema, os desenvolvedores de sistema não têm um entendimento correto sobre o negócio ou o negócio muda frequentemente e o software não consegue acompanhar tais mudanças.

Identificar os requisitos corretos em sistemas de software não é a única razão para a modelagem de negócio. Modelagem de negócio cria uma abstração de um negócio que é complexo e estabelece um entendimento comum que pode ser transmitido para os envolvidos. Um melhor entendimento de como o negócio funciona facilita o seu aprimoramento e ajuda a identificar novas oportunidades.

Embora a UML em seus primeiros anos fosse usada principalmente, para modelagem de sistemas de software, ela pode ser aplicada a modelagem de negócio. Ela permite descrever os aspectos estruturais de um negócio (a organização, hierarquia de objetivos e as estruturas dos recursos) e os aspectos comportamentais (os processos e as regras que afetam a estrutura e o comportamento do negócio). Muitos desenvolvedores de software estão familiarizados com a UML, uma vez que eles a têm usado constantemente na modelagem de seus sistemas. Usando a mesma linguagem de modelagem para a modelagem de negócio e modelagem de sistemas de software, a UML assegura que a documentação seja consistente e facilita a comunicação entre os responsáveis pelas

[1] Negócio no contexto do presente trabalho deve ser entendido como qualquer tipo de atividade contínua que tem ou usa recursos e tem um ou vários objetivos (Eriksson & Penker, 2000).

modelagens de negócio e do software (Eriksson & Penker, 2000). Em adicional, existe uma vasta quantidade de ferramentas que suportam a UML, permitindo assim o uso da modelagem de negócio com a UML.

1.2. Objetivos

O presente trabalho tem como objetivo geral fazer a adaptação do método de modelagem de negócios Eriksson-Penker, que utiliza a UML para os novos conceitos trazidos pela UML 2.0. Além disso, é meta deste trabalho a realização de um estudo de caso fazendo modelagem de negócio, com o método adaptado da UML 2.0, em uma fábrica de software.

Tem-se com a UML 2.0 um mecanismo de modelagem de negócio próximo aos mecanismos utilizados para modelagens de sistemas de software. Dessa forma, a modelagem de negócio poderá ser utilizada como base para a especificação de sistemas de software e a transformação do modelo de negócio para o modelo de análise poderá ser feita com uma maior facilidade, pois são utilizadas a mesma linguagem para ambas as modelagens.

O estudo de caso em uma fábrica de software tem como finalidade facilitar o entendimento das estratégias propostas, nas definições de seus objetivos e na definição formal de sua estrutura, comprovando o trabalho realizado na adaptação.

1.3. Estrutura do Trabalho

O presente trabalho está estruturado da seguinte maneira:

No Capítulo 2, é feita uma revisão bibliográfica sobre a modelagem de negócios, trazendo a definição dos termos utilizados sobre o assunto.

No Capítulo 3, são apresentadas a UML 1.4, a UML 2.0 e discutindo as diferenças entre ambas.

No Capítulo 4, é apresentado o método de modelagem de negócio Eriksson-Penker que utiliza a UML e os elementos e a estrutura do método.

No Capítulo 5, é apresentada a metodologia que foi utilizada para a realização deste trabalho.

No Capítulo 6, é apresentada como foi realizada a adaptação do método Eriksson-Penker para os conceitos trazidos da UML 2.0.

No Capítulo 7, é apresentado o estudo de caso da modelagem de negócios de uma fábrica de software, mostrando como foi realizado e os resultados.

No Capítulo 8, é apresentada a conclusão deste trabalho, quais foram suas contribuições e o que poderá ser realizado para sua continuação.

2. MODELAGEM DE NEGÓCIOS

Neste capítulo, é introduzida uma noção sobre Modelagem de Negócios, apresentando uma definição, quais os aspectos que serão utilizados para focar a modelagem e algumas justificativas para fazer a modelagem de negócio.

Este capítulo tem como intuito de esclarecer os conceitos básicos utilizados na Modelagem de Negócio, necessários para o entendimento e realização do presente trabalho.

2.1. Introdução a Modelagem de Negócios

De acordo com (Eriksson & Penker, 2000), modelo de negócio é uma abstração de como um negócio funciona. É uma visão simplificada de uma complexa realidade, deixando eliminar detalhes e focando nos aspectos mais importantes do negócio.

O modelo de negócio cria um entendimento comum que facilita a discussão entre os envolvidos no negócio, ajudando-os a chegar a um acordo comum sobre os fundamentos chaves e trabalhar para atingir objetivos comuns.

Vernadat (1996) coloca os principais objetivos da modelagem de processos de negócio, sendo eles: i) uniformização do entendimento da forma de trabalho, gerando integração; ii) análise e melhoria do fluxo de informações; iii) explicitação do conhecimento sobre os processos, armazenando a competência organizacional; iv) realização de análises organizacionais e de indicadores (processos, financeiros e outros); v) realização de simulações, apoiando tomada de decisões; e vi) gestão da organização.

Muitos dos elementos importantes em um negócio – clientes, fornecedores, leis e regras – são externos, não fazendo parte da sua definição. Por isso, o sistema de negócio não pode ser considerado um sistema fechado, cujas artes são, na maioria das vezes, compartilhadas com outros modelos de negócios.

Os detalhes capturados pelo modelo diferem de acordo com a perspectiva do analista que o modela, visto que cada pessoa possui um ponto de vista diferente de como o negócio funciona e quais são seus objetivos. Isto é normal e o modelo de negócio não irá resolver completamente este problema. O modelo fornecerá uma visão simplificada da estrutura do negócio, que agirá como base para a comunicação entre os envolvidos,

aprimoramento ou inovações e também os requisitos necessários para um sistema de software que é necessário para apoio ao negócio.

2.2. A Regra dos Modelos

Segundo Eriksson & Penker (2000), o modelo de negócio funciona como um plano para conduzir um negócio. Ele atua como a base para a tomada de decisão e afeta nas decisões sobre a priorização de objetivos, obtenção dos recursos corretos ou negociações com os subcontratados. O modelo pode antecipar e prever mudanças necessárias para se manter competitivo. O modelo pode não prover todas as respostas, mas uma estratégia básica ou um plano para seguir.

Idealmente, um modelo de negócio consiste de um único diagrama que inclui os aspectos importantes de um negócio. Entretanto, isto nem sempre é possível, um único negócio é tão complexo e possui muitos aspectos que um diagrama somente pode não capturar toda a informação.

Assim sendo, um modelo de negócio é composto pelos seguintes itens (Eriksson & Penker, 2000):

- **Visões (*Views*)** – Um modelo de negócio é ilustrado com um número de diferentes visões, cada qual captura informações sobre um ou mais aspectos específicos do negócio. Uma visão é uma abstração de um ponto de vista específico, omitindo detalhes que são irrelevantes para este ponto de vista. Múltiplas visões são necessárias para separar propósitos e perspectivas em um modo controlado, sem perder informações importantes sobre o negócio;

- **Diagramas (*Diagrams*) –** Cada visão consiste de vários diagramas que mostram uma parte específica tanto da estrutura quanto de uma situação do negócio. Vários diagramas são necessários para enxergar uma única visão do modelo de negócio, visto que cada tipo de diagrama possui um propósito diferente e expressa um importante aspecto ou mecanismo com a visão do modelo de negócio; e

- **Objetos e Processos (*Objects and Processes*)** – Conceitos são expressos nos diagramas através do uso de diferentes objetos e processos. Os objetos na

modelagem podem ser: a) físico, como as pessoas, máquinas, produto e materiais; b) abstratos, como débitos, instruções e serviços. Objetos podem também representar outros objetos por conter informações sobre outras coisas no negócio. Processos são as funções no negócio que consome, refina ou usa objetos para afetar ou produzir outros objetos.

Um modelo é motivado por objetivos. O propósito de modelar um negócio pode ser entender ou melhorar sua funcionalidade.

2.3. Modelagem de Processo de Negócio

Como visto por Eriksson & Penker (2000), um negócio é um sistema complexo, consistindo da organização hierárquica de departamentos e suas funções. Entretanto, algumas destas funções não são restritas à um departamento e cruzam-se horizontalmente através de vários departamentos. O método tradicional para documentar um negócio é desenhar um gráfico organizacional, que divide o negócio em departamentos ou seções representados verticalmente. Este método de documentação é limitado para representar como o negócio é construído e organizado. O método tradicional não documenta os processos de negócios que fluem horizontalmente, afetando vários departamentos.

No campo de modelagem de processos, diversas teorias tentam explicar e melhorar como estruturar e executar um negócio. Poucos padrões e métodos existem nesta área, sendo que a maior parte da literatura concentra em como descrever um negócio e não em técnicas bem definidas para modelar um negócio. O conceito central usado para modelagem de processos é o processo de negócio, que descreve as atividades do negócio e como elas se relacionam para interagir com os recursos no negócio para alcançar um objetivo para o processo.

Os argumentos para a produção de modelos de negócio são (Eriksson & Penker, 2000):

- Entender melhor os mecanismos chaves de um negócio existente;
- Atuar como base para criação de um sistema de software adequado que suporta o negócio;
- Atuar como base para melhorar a atual estrutura do negócio e suas operações;

- Mostrar a estrutura de um negócio inovador; e
- Identificar a oportunidade para *outsourcing*.

Com isso, a modelagem em suas diversas aplicações pode gerar alguns ganhos importantes para as organizações (Santos et al., 2002), como:

- Uniformização de Entendimentos sobre a Forma de Trabalho: A modelagem de negócios, idealmente realizada com uma mesma ferramenta e notação, pode proporcionar para uma organização, através da visão por processos, a difusão das relações de trabalho entre as diversas unidades organizacionais que passam a entender melhor e a ter uma visão homogênea do negócio;
- Melhoria do Fluxo de Informações: Considerando-se que um dos principais objetivos da modelagem de processos de negócio é a identificação do fluxo de informações que corta uma organização, a importância da visão por processos é possibilitar e permitir o suporte pela TI na automatização desse fluxo e no melhor entendimento, principalmente, das informações trocadas nas interfaces organizacionais, podendo assim obter uma melhoria do fluxo de informações;
- Padronização dos Processos: Este ganho está diretamente relacionado à uniformização do entendimento da forma de trabalho, pois, diante de um referencial de conformidade com uma mesma linguagem de modelagem (modelos com uma mesma notação) compreendida, fica fácil definir um padrão de modelagem e representação dos processos de negócio;
- Melhoria da Gestão Organizacional: A associação dos processos modelados com indicadores de desempenho locais e globais proporciona melhorias na gestão organizacional em termos de monitoramento, controle e coordenação do trabalho;
- Redução de Tempo e Custos dos Processos: A modelagem proporciona a oportunidade de identificação e melhoria de problemas relacionados ao seqüenciamento das atividades, alocação de recursos e atividades que não agregam valor ao produto final que estão diretamente relacionados à redução de tempo e custos operacionais; e

- Aumento da Conceituação Organizacional sobre Processos: Este ganho está relacionado diretamente aos demais ganhos, pois é uma premissa básica para a consolidação da visão por processos dentro da organização que passa a trabalhar orientada total ou parcialmente a processos.

2.4. Considerações Finais

A modelagem de negócio pode ser considerada uma ferramenta importante para a organização, pois ela permite a visualização do negócio como todo, eliminando os detalhes relevantes e dando ênfase nos aspectos mais importantes do negócio.

O método tradicional de documentação de negócios em que é feita uma representação do negócio em departamentos e seções representadas verticalmente é incompleto. O processo de negócio flui horizontalmente agrupando muitas vezes mais de um departamento. Surgiu, assim, a necessidade de uma modelagem mais completa do negócio para fazer suas representação mais abrangente.

3. A UML (*Unified Modeling Language*)

O presente capítulo apresenta a UML, *Unified Modeling Language*, que é uma linguagem para construção de modelos.

Neste capítulo, são abordados os conceitos básicos da UML, explicando seus objetivos, sua estrutura e sua aplicação. Além disso, é apresentada uma comparação entre a UML 2.0 e UML 1.4, sendo explicado os novos conceitos introduzidos.

O intuito deste capítulo é fornecer o entendimento que foi necessário para realização das adaptações realizadas no método de modelagem de negócio realizada neste trabalho.

3.1. Introdução a UML

A UML é a linguagem proposta pela OMG que reúne as principais características das notações dos métodos Booch (Booch, 1991), *Object Modeling Technique* (OMT) (Rumbaugh et al., 1991) e *Object Oriented Software Engineering* (OOSE) (Jacobson et al., 1992). Tornou-se o padrão de mercado por ser intuitiva e fácil de usar.

A UML é amplamente utilizada para modelar sistemas. Um modelo de UML representa a descrição do conjunto de objetos que fazem parte da aplicação e de interações as quais eles estão sujeitos ao longo do tempo. Cada um desses conjuntos constitui uma configuração do sistema e a coleção das configurações possíveis denota a semântica do modelo. Assim, um modelo pode ser visto como a intenção do sistema (Booch et al., 2005).

A UML sugere a construção de vários diagramas que permitem expressar os aspectos estáticos e dinâmicos de uma aplicação. Aspectos estáticos estão relacionados à estrutura do sistema e independem do tempo. O objetivo é descrever as entidades de um sistema e como elas sempre vão interagir. Por outro lado, os aspectos dinâmicos dizem respeito à evolução da aplicação. O foco é contemplar a criação e a destruição de objetos e as interações ao longo do tempo; em outros termos, as transformações às quais o estado global do sistema está sujeito.

A UML é uma linguagem para visualizar, especificar, construir e documentar os artefatos de um sistema de software. As subseções a seguir explicam cada uma das características apresentadas.

3.1.1. A UML é uma Linguagem para Visualizar

De acordo com Booch et al. (2005), para muitos desenvolvedores, a distância entre pensar em uma implementação e colocar no código é perto de zero. Neste caso, o desenvolvedor está ainda fazendo alguns modelos, embora de forma mental, porém há vários problemas com isto: primeiro, passar aqueles modelos conceituais para outros pode acarretar erros desde que os envolvidos falem a mesma linguagem. Normalmente, organizações e projetos desenvolvem sua própria linguagem e é difícil de entender o que está acontecendo caso a pessoa seja nova no grupo ou não parte deste; segundo, existem algumas coisas sobre sistema de software que não podem ser entendidas desde que sejam construídos modelos que transcendam a linguagem de programação textual; e terceiro, se o desenvolvedor que escreveu o código não anotou os modelos que estão na sua cabeça, aquelas informações podem ser perdidas.

O uso da UML para elaborar modelos abrange uma terceira questão: Um modelo explícito para facilitar comunicação. Sendo uma linguagem gráfica, a UML resolve também o segundo problema descrito e, por último, a UML é mais do que um conjunto de símbolos gráficos. Atrás de cada símbolo da UML está uma semântica bem definida; dessa maneira, um desenvolvedor pode escrever um modelo em UML e outro desenvolvedor ou mesmo outra ferramenta pode interpretar o modelo de forma não ambígua. Isto resolve o primeiro problema.

3.1.2. A UML é a Linguagem para Especificar

Neste contexto, especificar significa construir modelos que são precisos, não ambíguos e completos (Booch et al., 2005). Em particular, a UML direciona a especificação da análise, *design* e decisão de implementação que deve ser feita no desenvolvimento e implantação do sistema de software.

3.1.3. A UML é uma Linguagem para Construir

Segundo (Booch et al., 2005), a UML não é uma linguagem de programação visual, mas seus modelos podem ser diretamente conectados a uma variedade de linguagem de

programação. Isto significa que é possível mapear de um modelo em UML para uma linguagem de programação como Java, c++ ou *Visual Basic* ou mesmo para tabelas em uma base de dados relacional ou armazenamento persistente de uma base de dados orientada a objetos.

Este mapeamento permite a geração de código de um modelo UML para uma linguagem de programação e vice-versa.

3.1.4. A UML é uma Linguagem para Documentar

Uma organização de software saudável produz diversos tipos de artefatos, além dos códigos executáveis em si (Booch et al., 2005). Estes artefatos incluem (mas não se limitam em): requisitos, arquitetura, projeto, código fonte, planos de projetos, testes, protótipos e *release*.

A UML dirige a documentação de uma arquitetura de sistema e todos os seus detalhes e prove uma linguagem para expressar requisitos e testes e para modelar atividades de planejamento de projeto e gerenciamento de *release*.

3.2. Um Modelo Conceitual de UML

Para entender a UML é preciso ter em mente o modelo conceitual da linguagem e isto requer aprender três elementos principais: blocos de construção básicos da UML, as regras que ditam como esses blocos podem ser reunidos e alguns mecanismos comuns que se aplicam ao longo da UML.

O vocabulário de UML inclui três tipos de blocos de construção: elementos, relacionamentos e diagramas.

Os elementos são as abstrações que são classes nativas em um modelo, os relacionamentos ligam esses elementos entre si e os; diagramas agrupam coleções de elementos e relacionamentos para representar uma abstração.

3.2.1. Elementos da UML

Segundo Booch et al. (2005), existem quatro tipos de elementos da UML:

- Elementos estruturais;

- Elementos comportamentais;
- Elementos agrupadores;
- Elementos de anotação.

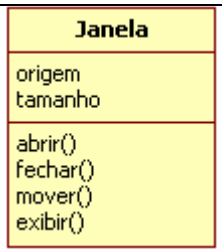
Esses elementos são os blocos de construção básicos orientados a objetos da UML que podem ser utilizados para escrever modelos bem formados.

Elementos estruturais

Elementos estruturais são os nomes dos elementos da linguagem UML. São geralmente partes estáticas de um modelo, representam elementos que são conceituais ou físicos. Coletivamente, elementos estruturais são chamados classificadores[2].

Na Tabela 3.1, são descritos os elementos estruturais da UML mais relevantes para realização deste trabalho.

Tabela 3.1 - Elementos Estruturais da UML.

ELEMENTOS ESTRUTURAIS	DESCRIÇÃO
 Classes	Uma classe é a descrição de um conjunto de objetos que compartilham atributos, operações, relacionamentos e semântica. A classe implementa uma ou mais interfaces.

Fonte: Adaptação de Booch et al. (2005).

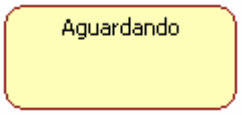
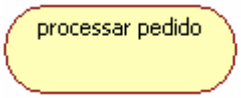
Elementos Comportamentais

Elementos comportamentais são as partes dinâmicas dos diagramas da UML. Há três tipos primários de elementos comportamentais: interações, máquina de estados e atividades.

A Tabela 3.2 apresenta os elementos comportamentais, que podem ser considerados os verbos de um modelo, representando comportamentos no tempo e espaço (Booch et al., 2005).

[2] Um mecanismo que descreve características estruturais e comportamentais. Classificadores incluem classes, interfaces, tipos de dados, símbolos, componentes, nós, casos de uso e subsistemas.

Tabela 3.2 - Elementos Comportamentais da UML.

ELEMENTOS ESTRUTURAIS	DESCRIÇÃO
 Mensagens	Uma interação é um comportamento que engloba um conjunto de mensagens trocadas entre um conjunto de objetos ou regras com um contexto particular para cumprir um propósito específico. Uma interação envolve um número de outros elementos, incluindo mensagens, ações e conectores.
 Estados	Uma máquina de estados é um comportamento que especifica as seqüências de estados que um objeto ou uma interação irão passar durante seu tempo de vida na resposta de eventos, junto com suas respostas ao evento. Uma máquina de estado envolve vários elementos, incluindo estados, transições, eventos e atividades.
 Ações	Uma atividade é um comportamento que especifica uma seqüência de passos que um processo computacional executa. Um passo de uma atividade é chamado de ação.

Fonte: Adaptação de Booch et al. (2005).

Elementos de Agrupamento

Elementos de agrupamento são as partes organizacionais dos diagramas da UML. Há um tipo primário de elemento de agrupamento chamado pacotes.

Um pacote é um mecanismo para organizar o próprio modelo, onde os elementos estruturais, elementos organizacionais e outros elementos de agrupamento podem ser colocados em um pacote (Booch et al., 2005) (Figura 3.1).

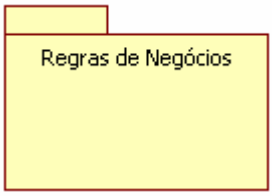


Figura 3.1 - Pacotes. Fonte: Adaptação de Booch et al. (2005).

Elementos de Notações

Elementos de notações são as partes explicativas dos diagramas da UML. O tipo principal de elementos de notações é a nota. A Figura 3.2 é a representação gráfica dos elementos de notações.

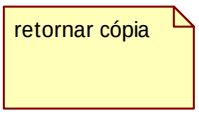


Figura 3.2 - Notas. Fonte: Adaptação de Booch et al. (2005).

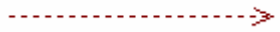
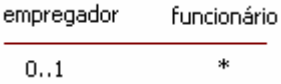
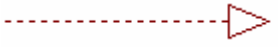
3.2.2. Relacionamentos em UML

Há quatro tipos de relacionamentos na UML: dependência, associação, generalização e realização.

Estes relacionamentos são os blocos de construção relacionais básicos da UML. São usados para escrever modelos bem formados.

A Tabela 3.3 apresenta graficamente os relacionamentos em UML e uma breve descrição de suas funções.

Tabela 3.3 - Relacionamentos da UML.

RELACIONAMENTOS	DESCRIÇÃO
 Dependências	Uma dependência é um relacionamento semântico entre dois elementos de modelagem onde uma modificação em um elemento pode afetar a semântica do outros elementos.
 Associações	Uma associação é um relacionamento estrutural entre classes que descrevem um conjunto de *links*. Agregação é um tipo especial de associação, representando um relacionamento estrutural entre um todo e suas partes.
 Generalizações	Uma generalização é um relacionamento de especialização generalização no qual o elemento especializado (o filho) herda a especificação do elemento generalizado (o pai). O filho compartilha a estrutura e o comportamento do pai.
 Realizações	Uma realização é uma relação semântica entre classificadores, em que um classificador especifica um contrato que outro classificador garante executar.

Fonte: Adaptação de Booch et al. (2005).

3.2.3. Diagramas no UML

Um diagrama é a representação gráfica de um conjunto de elementos. Muitas vezes um diagrama é definido como um gráfico conectado de vértices (elementos) e arestas (relacionamentos). Os diagramas são desenhados para visualizar um sistema de diferentes perspectivas; assim, um diagrama é uma projeção de uma perspectiva dentro de um sistema.

A UML inclui vários diagramas. A seguir é descrito os diagramas que foram utilizados no presente trabalho (Booch et al., 2005):

- **Diagrama de Classes -** representa um conjunto de classes, interfaces, e colaborações e seus relacionamentos. Diagramas de classe se enquadram na visão estática de um sistema;

- **Diagrama de Objetos -** representa um conjunto de objetos e seus relacionamentos. Diagramas de objetos representam uma foto momentânea dos elementos encontrados nos diagramas de classes. Diagramas de objeto se enquadram na visão estática de um sistema;

- **Diagrama de Seqüência e de Comunicação -** diagramas de seqüência e diagramas de comunicação são tipos de diagramas de interação. Um diagrama de interação mostra uma interação, consistindo de um conjunto de objetos ou regras, incluindo as mensagens que podem ser trocadas entre eles. Diagramas de interação se enquadram na visão dinâmica de um sistema. Um diagrama de seqüência enfatiza a ordem cronológica das mensagens, enquanto um diagrama de comunicação enfatiza a organização estrutural dos objetos e regras que enviam e recebem mensagens. Ambos representam conceitos similares, mas cada um tem enfoque em uma diferente visão de conceitos. Diagramas de Seqüência e de Comunicação se enquadram na visão dinâmica de um sistema;

- **Diagrama de Estado -** representa uma máquina de estados, consistindo de estados, transições, eventos e atividades. Diagramas de Estado se enquadram na visão dinâmica de um sistema;

- **Diagrama de Atividade –** representa uma estrutura de um processo ou outra computação como um fluxo de controle passo a passo na computação. Diagramas de atividades se enquadram na visão dinâmica de um sistema. Diagramas de Atividade se enquadram na visão dinâmica de um sistema;

3.3. Mecanismos de Extensão

A UML permite extensões, de modo controlado, para poder expressar de forma completa os inúmeros software possíveis. Os mecanismos de extensão são (Booch et al., 2005):

- **Estereótipos –** estendem o vocabulário da UML, permitindo criar novos tipos de blocos de construções que são derivados de existentes;
- **Valores Marcados** - estende as propriedades de um estereotipo da UML, permitindo incluir novas informações nas especificações de estereótipos;
- **Restrições** - estendem a semântica da UML, construindo blocos e permitindo você adicionar novas regras ou modificar uma existente.
- ***Profiles*** - um conjunto coerente de extensões adequadas a um dado domínio. Por sua natureza, *profiles* não são aplicáveis em todas as circunstancias e, diferentes *profiles* podem ou não ser compatíveis com outros (Rumbaugh et al., 2004).

3.4. Diferenças entre UML 2.0 e UML 1.4

A UML 1.4 combinou diferentes métodos de modelagem em um padrão, o que facilita a comunicação entre os elementos do modelo. A UML 2.0 inclui várias características que facilitam a comunicação entre máquina e os elementos do modelo, enquanto que, continua respondendo às questões e as necessidades das pessoas. A UML 2.0 adiciona características para descrever comportamentos e caminhos alternativos, apoiar desenvolvimento de componentes e sustentar o desenvolvimento distribuído complexo, além de buscar cobrir uma arquitetura dirigida por modelos com sofisticação bastante técnica para gerar código automaticamente e com bastante flexibilidade para evoluir com as mudanças contínuas.

Para este *upgrade* da UML, a OMG apontou metas bem definidas para UML 2.0 em um conjunto de requisitos que refletem questões encontradas por usuários da UML 1.4 . Da especificação de requisitos, quatro objetivos emergiram como importantes direções na evolução de UML (Eriksson et al., 2004):

- Fazer uma linguagem para modelar entidades de software mais executável;
- Prover mecanismos mais robustos para modelar *workflows* e ações;
- Criar um padrão para comunicação entre ferramentas diferentes; e
- Compor UML dentro de um *framework* de modelagem padrão.

Seguindo esse conjunto de metas, as principais mudanças da UML 2.0 em relação a UML 1.4 são (Eriksson et al., 2004):

- O diagrama de colaboração da UML1.4 foi transformado em um diagrama de comunicação. Os diagramas de comunicação são considerados um tipo de diagrama de interação;
- O diagrama de seqüência foi fragmentado, permitindo mostrar opções mais complexas em um diagrama como a área onde a ordem de mensagens é critica e áreas onde processamento paralelo é permitido. Os fragmentos também apóiam reuso de elementos do diagrama de seqüência;
- O diagrama de atividades representam um construtor de classe primária e não um agrupado a máquina de estado. O novo diagrama de atividades inclui apoio a características adicionais do fluxo e não tem enfoque em transições. Baseado em *Redes de Petri*, os novos diagramas de atividade modelam fluxos dinâmicos e podem mostrar facilmente atividades de interrupção, fluxos múltiplos, controle de exceção, e regiões protegidas;
- Surgiram dois novos diagramas de interação: o diagrama de tempo e o diagrama de visão da interação;
- O novo diagrama de tempo provê um modo para controlar o tempo ao longo de um eixo de um diagrama de interação. Este diagrama permite traçar estado e

comportamento para clarear unidades de tempo que ajudam na análise de assuntos relacionados à *runtime*;

- A máquina de estados de protocolos e as portas provêem várias características para apoiar desenvolvimento de componente fornecendo regras claras para as exigências ambientais do componente;

- Uma máquina de estados tem regras de extensão mais clara de forma que a parte da máquina de estados pode ser usada em vários lugares e pode trabalhar com um classificador que faz parte de uma hierarquia de herança;

- UML 2.0 inclui características padrões para observar e criar restrições baseadas no tempo e na duração de comportamentos;

- O uso de ocorrências de interação habilita decomposição de interações em blocos de modelagem reutilizáveis;

- A introdução do diagrama de estrutura composto permite mostrar a conexão entre um classificador de alto-nível, como um caso de uso, e os elementos de *runtime* necessários para aquele classificador;

- Na UML 2.0, as regras para definição da estrutura interna de um classificador são mais claras, facilitando a decomposição de um classificador;

- O conjunto de generalizações provê uma forma para clarificar os tipos diferentes de generalizações. Uma classe pode ter dois ou mais tipos de relacionamento de generalização para diferentes conjuntos de classes;

- Pacotes podem receber um estereótipo padrão <<*merge*>> para melhor mostrar o tipo de relacionamento de dependência em um diagrama de pacote;

- Muitos elementos têm características que, agora mais explicitamente além das classes, se aplicam a classificadores. Outros elementos de UML 2.0 podem melhorar de hierarquias de generalização e relacionamentos definidos. Por exemplo, sinais, como também máquinas de estado, podem ser redefinidos por herança; e

- O diagrama de estrutura composto é uma adição nova em UML 2.0 e é usado para mostrar a arquitetura de *runtime* de um classificador em particular. Os nós dos componentes neste diagrama, também são novos.

Finalmente, os conceitos de *profiles* na UML 2.0 e arquitetura dirigida a modelo podem produzir finalidades significativamente mais robustos, pois permite a reutilização de características utilizada em um modelo.

Em termos de extensões, a UML 2.0 enfatiza o uso de *profiles*, como também o uso de níveis de modelos múltiplos, para manter modelos focalizados no seu propósito principal, aproveitando abordagens comuns para problemas. Estas extensões fazem com que fique mais fácil para alguns níveis de modelo produzir código.

Um modelador indica qual o *profile* se aplica ao seu modelo, podendo qualquer outro usuário adquirir o mesmo perfil para trabalhar no sistema. Esta arquitetura de linguagem é projetada para apoiar uma proliferação de extensões do modelo para que o arquiteto possa estender os conceitos de UML e de domínios novos.

A UML 2.0 não usa mais a linguagem UML para definir seu próprio modelo. É definida em termos de uma linguagem de modelagem mais abstrata denominada *Meta-Object Facility* (MOF) (MOF, 2006). Isto permite a comunidade de modelagem habilidade para completar UML criando um primo de UML. Esta nova habilidade para criar um outro membro da família UML de linguagens resultou na especificação separada de uma linguagem de restrição com *Object Constraint Language* (OCL) (OCL, 2006), que não mais está definida na especificação de UML.

3.5. Considerações Finais

A diferença mais relevante entre a versão 2.0 para a 1.4 da UML, observada, para a execução do trabalho foi a maneira de realizar a extensão dos elementos de modelagem, em principal destaque o conceito de *profile*.

O presente capítulo trouxe o conhecimento para a realização das adaptações necessárias no método de modelagem Eriksson-Penker.

4. MODELAGEM DE NEGÓCIO COM UML: MÉTODO ERIKSSON-PENKER

Neste capítulo, é apresentada a estrutura do método de modelagem de negócios Eriksson-Penker. São discutidas as características de uma arquitetura de negócios, os conceitos que são usados na definição desta arquitetura e o conjunto de extensões para UML que são utilizados no método Eriksson-Penker.

Este capítulo mostra estrutura do método Eriksson-Penker que foi necessária adaptar para incorporar as características da UML 2.0.

Para a escolha do método em questão foi realizado um estudo em alguns métodos existentes na literatura (Alencar 1999, Bubenko et al., 2001, Persson, 2001, Chen-Burger, 2001) e também através da comparação realizada por Monteiro (2003) dos seguintes métodos e técnicas: o método *Business System Development Method* (BSDM) (IBM, 1992 Chen-Burger, 2001), a Técnica *i**[3] (Yu, 1995), o método *Enterprise Knowledge Development* (EKD) (Bubenko et al., 1997, Bubenko et al., 1998, Bubenko et al., 2001) e método Eriksson-Penker (Eriksson & Penker, 2000).

4.1. Arquitetura de Negócio

De acordo com Eriksson & Penker (2000), arquitetura de negócio é a base para descrever e entender uma empresa. A arquitetura de negócios lista as partes requeridas de um negócio, como estas partes são estruturadas e interagidas e como a arquitetura deve envolvê-las. Embora seja difícil definir o termo, pode-se usar a seguinte definição:

"...um conjunto organizado de elementos se relaciona claramente com outro, que juntos formam um todo definido por sua funcionalidade. ... Os elementos representam a estrutura organizacional e comportamental de um sistema de negócio, e mostra abstrações dos processos chaves e estrutura no negócio." (Vernadat, 1996)

Segundo Eriksson & Penker (2000), uma boa arquitetura permite que o modelador abstraia o negócio em diferentes aspectos e se concentre em um somente aspecto por vez.

[3] O acrônimo *i** refere-se a "*distributed intentionality*"

Isto, juntamente com desconsideração dos detalhes irrelevantes, é essencial para entender os sistemas complexos e suas relações.

De acordo com Eriksson & Penker (2000), uma boa arquitetura possui as seguintes características:

- Captura o real negócio como mais verdadeiro e correto possível;
- Focar nos processos e nas estruturas chaves do negócio com um nível apropriado de abstração;
- Representa um consenso entre as pessoas envolvidas no negócio;
- Adapta facilmente a mudanças e extensões;
- É fácil de entender e encoraja a comunicação entre os diferentes *stakeholders* do negócio.

Mas, atingir estas características não é simples. São requeridas várias coisas (Eriksson & Penker, 2000):

- Modeladores que têm um alto conhecimento do negócio ou, no mínino, acesso a pessoas com conhecimento suficiente que possam ser entrevistadas e possam participar na construção da arquitetura;
- A linguagem de modelagem deve capturar os importantes conceitos no negócio, junto com as relações entres estes capturados;
- Uma capacidade para organizar diagramas visuais em diferentes visões do negócio, onde cada visão ilustra um aspecto específico do negócio;
- Um projeto baseado na experiência, em que trabalhar e em que não trabalhar;
- Um processo de desenvolvimento que assegura a qualidade e precisão dos modelos produzidos.

4.2. Conceitos de Negócios

Os conceitos que são utilizados na definição de sistemas de negócios são:

- **Recursos –** são os objetos dentro do negócio, como pessoas, informações, e produtos que são usados ou produzidos no negócio. Os recursos podem ser categorizados como físico, abstrato e informacional;

- **Processos –** são as atividades executadas dentro do negócio durante o processo de mudança dos estados e recursos;

- **Objetivos –** são os propósito do negócio ou a conseqüência que o negócio como um todo está tentando realizar;

- **Regras –** são as afirmações que definem ou restringem algum aspecto do negócio, e representa o conhecimento do negócio. As regras governam como o negócio deve ser executado ou como os recursos podem ser estruturados e relacionados entre si. As regras podem ser classificadas como funcionais, comportamentais e estruturais.

4.3. Extensões de Negócios Eriksson-Penker

A UML foi definida para modelar a arquitetura de sistemas de software e tem similaridades e diferenças entre sistemas de software de negócios. Para fazer a modelagem de negócios com a UML, precisa-se estendê-la para identificar e visualizar os importantes conceitos de processos, objetivos, recursos e as regras do sistema de negócio. Por causa disto, é apresentado um conjunto de extensões para a UML chamado de Extensões de Negócio Eriksson-Penker (*Eriksson-Penker Business Extensions*), que provê os símbolos necessários para a modelagem de negócio.

4.3.1. Processos de Negócio

O processo de negócio é representado pela Extensão de Negócio Eriksson-Penker, que é descrito por um símbolo de acordo com a Figura 4.1. A simbologia da Extensão Eriksson-Penker é utilizado em muitas técnicas de modelagem de processos e não sobrepõe outro elemento existente na UML.

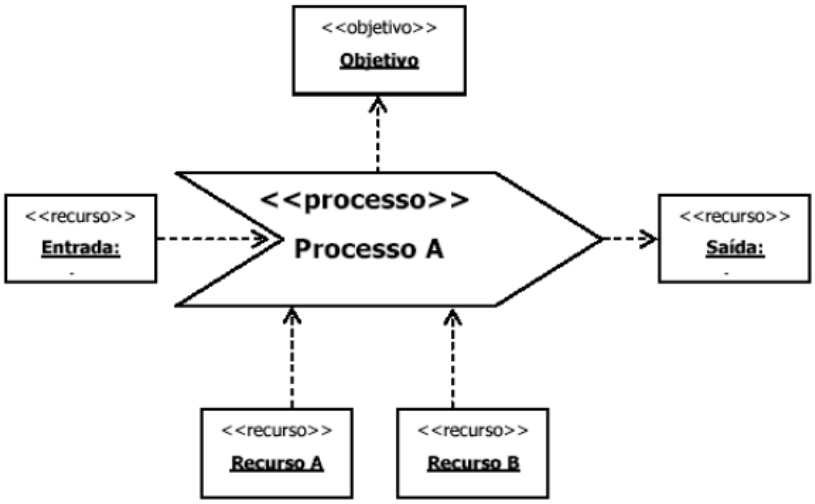


Figura 4.1 - Símbolo do processo de negócio, ilustrando um objetivo e entradas e saídas do objeto. Adaptado de Eriksson & Penker (2000).

Formalmente na UML, este símbolo é uma atividade estereotipada de um diagrama de atividade utilizado para ilustrar o processo de negócios. Um processo recebe recursos como entrada na sua parte esquerda e indica seus recursos de saída na sua parte direita. O objetivo do processo é ilustrado como um objeto objetivo acima do símbolo do processo.

Um processo pode ter Valores Marcados com informações adicionais sobre o processo. Os Valores Marcados utilizados na Extensão de Negócios Eriksson-Penker são:

- **Objetivo (*goal*) –** descreve o objetivo do processo;
- **Propósito (*purpose*) –** descreve o propósito do processo;
- **Documentação (*documentation*) –** uma descrição informal de como funciona o processo;
- **Dono do Processo (*process owner*) –** define o dono do processo, ou seja, representa na organização quem será responsável pelo processo;
- **Atores do Processo (*process actors*) –** define os atores necessários para executar o processo;
- **Prioridade (*priority*) –** descreve a prioridade do processo;
- **Riscos (*risks*) –** descreve os riscos do processo;

- **Possibilidades (*possibilities)* –** descreve as possibilidades do processo;
- **Tempo (*time*) –** tempo aproximado que deverá ser gasto na execução do processo;
- **Custo (*cost*) –** um valor aproximado do custo do processo.

4.3.2. Passos do Processo

O processo contém um número de passos ou atividades que são executados como uma parte do processo. As atividades são caracterizadas como o seguinte (Eriksson & Penker, 2000):

- **Diretas –** se envolve na criação do produto ou do serviço;
- **Indiretas –** se serve de suporte para as atividades diretas;
- **Garantia da qualidade –** se assegura a qualidade de outras atividades.

Os processos também podem ser caracterizados de outros modos, tal como processos de desenvolvimento, de melhoria ou de gerência.

Os passos do processo são ilustrados como atividades aninhadas. Entretanto, se um processo não possui subprocessos, é usado o símbolo padrão de atividade da UML.

Um processo pode se espalhar por mais de uma área da empresa.

4.3.3. Eventos de Negócio

Um processo é afetado por eventos que são ocorridos no ambiente que o cerca ou por eventos gerados por outros processos que fazem o processo ser ativado. Um evento de negócio pode ser definido como: um evento de negócio é um acontecimento externo do mundo real que requer certa ação (Gale & Eldred, 1996); um evento de negócio representa um registro de uma mudança no negócio em um particular instante no tempo (OMG, 1998).

Um evento pode: iniciar a execução de um processo, afetar o comportamento e execução do processo e concluir um processo gerando um evento.

Na notação de negócio Eriksson-Penker, um evento de negócio é representado como uma classe (o tipo evento) e objetos (instâncias do tipo evento). A classe evento é estereotipada como um evento de negócio (*business event*). Suas relações podem ser ilustradas em um diagrama de classes (Figura 4.2).

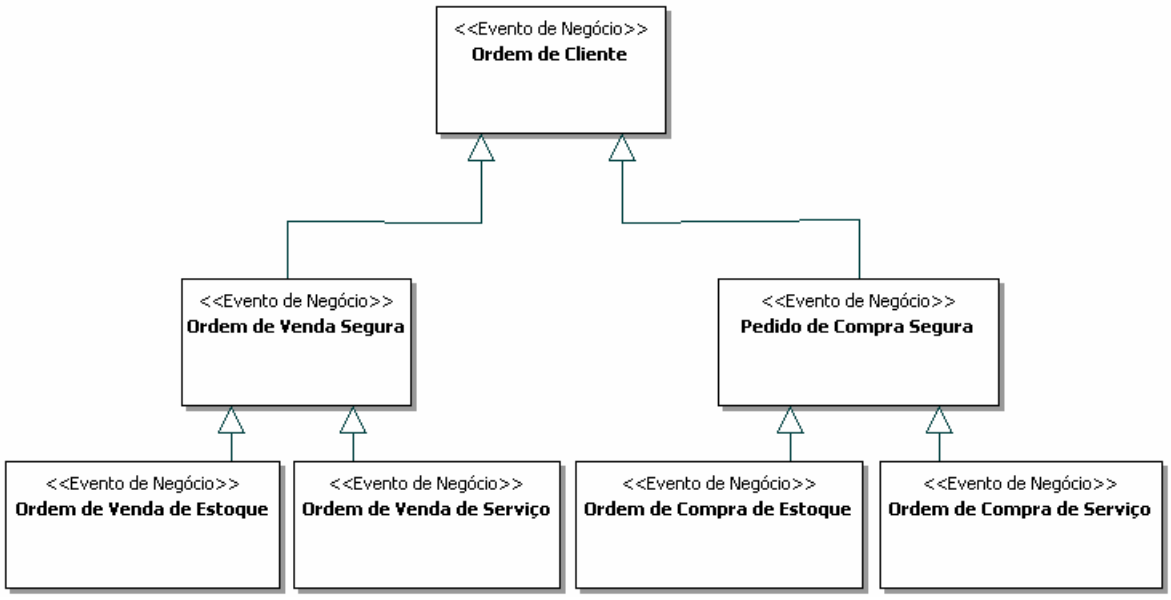


Figura 4.2 - Eventos de negócio são modelados como modelados como classes em uma hierarquia de classes. Fonte: Adaptação Eriksson & Penker, (2000).

O símbolo recebe (pentágono côncavo) e o símbolo envia (pentágono convexo) são usados em um processo para ilustrar os eventos enviados e recebidos (Figura 4.3). O símbolo recebido pode ser lido, o processo espera esse evento ocorrer, o símbolo indica que o processo gerou e enviou um evento entre duas atividades. Ambos, símbolos e eventos, podem ser ligados a um objeto por uma seta de dependência.

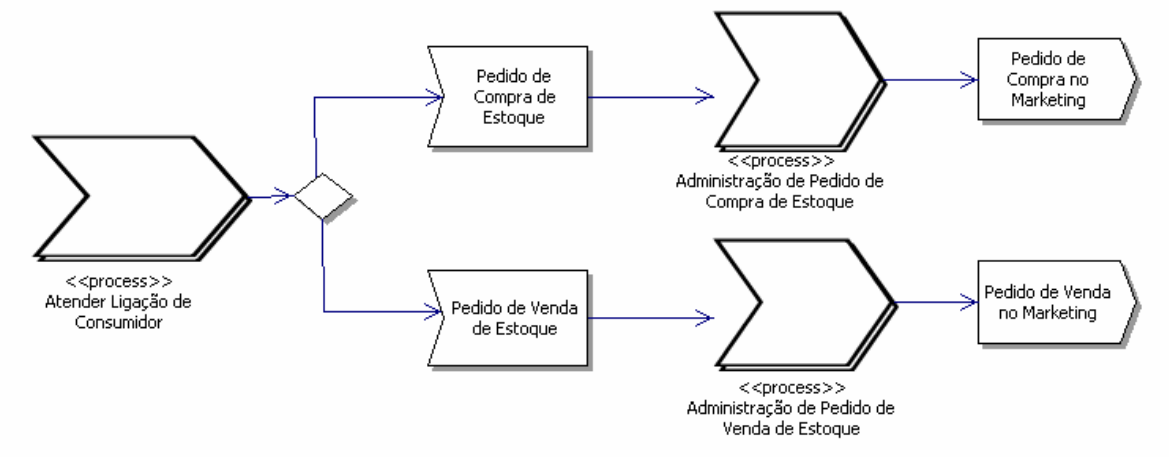


Figura 4.3 - Eventos de negócio enviados e recebidos durante um processo. Fonte: Adaptação de Eriksson & Penker (2000).

4.3.4. Modularizando o Sistema de Negócio

Um sistema grande terá muitos processos. Para trabalhar com estes processos, eles precisam ser agrupados ou modularizados para que o modelador possa se concentrar somente em um conjunto de processos por vez. Os processos são modularizados, utilizando o mecanismo de pacotes (*package*) da UML (Figura 4.4).

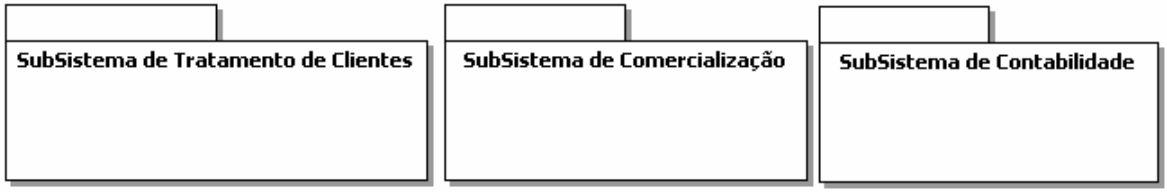


Figura 4.4 - Processos agrupados em pacotes. Fonte: Adaptação de Eriksson & Penker (2000).

4.3.5. Recursos

Recursos são objetos que agem ou são usados no processo. Vernadat (1996) propõem a seguinte definição de recursos: “Um recurso é uma entidade que pode desempenhar um papel na realização de certos tipos de tarefas." (Eriksson & Penker, 2000)

Uma outra definição é: “Um recurso é um conceito usado nos negócios e representa qualquer coisa que nós escolhemos para avaliar como um todo.” (Darnton & Darnton, 1997)

Os tipos recurso são apresentados como uma classe. Uma instância de recurso é representada como objetos.

A Extensão de Negócios Eriksson-Penker define alguns estereótipos indicados para categorias diferentes de recursos (Figura 4.5).

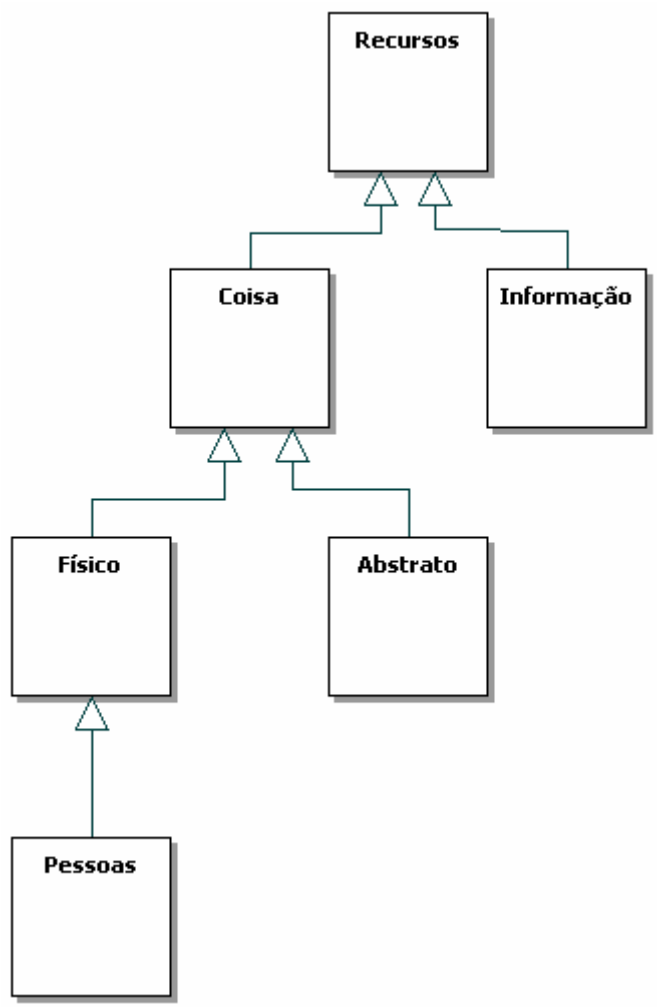


Figura 4.5 - Um meta-modelo mostrando uma hierarquia de diferentes tipos de recursos. Fonte: Adaptação de Eriksson & Penker (2000).

Gale & Eldred (1996) definem quatro tipos concretos de recursos:

- **Físico –** entidade que, com o seu real material, ocupa volume no espaço;
- **Abstrata –** uma idéia ou um conceito, frequentemente uma composição de outros objetos;
- **Objeto Informativo –** representa um conceito, coisa ou outro objeto de informação. Ele segura a informação sobre outros recursos. É muito importante separar objetos de informação do que representa conceito ou coisa, um objeto de informação segura fatos ou conhecimento sobre outros objetos no negócio;

- **Pessoas –** um humano interagindo com o processo.

4.3.6. Objetivos

Um objetivo descreve o estado desejado de um ou mais recursos. Objetivos são presos ao negócio e a processos de negócio individuais. O objetivo pode ser medido em ordem do seu progresso. A medida pode ser definida quantitativamente ou qualitativamente.

Um objetivo pode ser divido em sub-objetivos, onde para atingir o objetivo maior, deve-se atingir primeiro os sub-objetivos. A divisão de objetivos em sub-objetivos é chamada de modelagem de objetivo (Eriksson & Penker, 2000).

Junto com os objetivos, estão também os problemas, que é um obstáculo para atingir o objetivo. Entretanto, é comum criar novos objetivos com o intuito de evitar ou resolver os problemas encontrados.

O conjunto de Extensões de Negócio de Eriksson-Penker representa a modelagem de objetivos e problemas por meio do diagrama de objetos, mostrando as dependências entre os objetivos e os sub-objetivos. Um objeto objetivo é indicado como um objeto objetivo de uma classe de objetivo estereotipada. Neste método, existem dois tipos de classes para representar um objetivo: Objetivo Quantitativo e Objetivo Qualitativo, ambas com o estereótipo de Objetivo. Problemas são exibidos como notas com estereótipos de problema. A Figura 4.6, mostra como exemplo a descrição do objetivo “Maior Lucratividade”, que é um Objetivo Quantitativo sendo decomposto completamente em outros três sub-objetivos. Note que o objetivo “Vendas Efetivas” possui um problema.

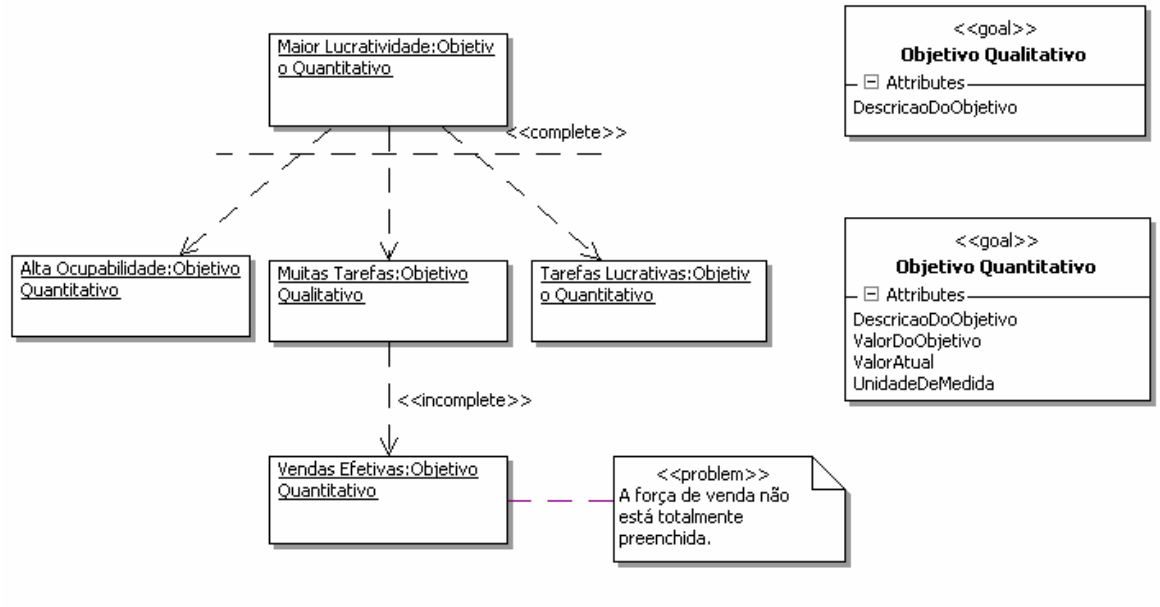


Figura 4.6 - Objetivos, sub-objetivos e problemas em um diagrama de objetos. Fonte: Adaptação de Eriksson & Penker (2000).

4.3.7. Regras de Negócio

Um modelo de negócio contém regras de negócios que define restrições, condições, e políticas de como o processo de negócio deve ser feito. Uma definição prática para regra é: "... afirmações que podem controlar ou afetar ambos a execução do processo de negócio como a estrutura dos recursos no negócio." (Eriksson & Penker, 2000)

A seguir, são apresentados os três tipos de regras de negócios:

- **Derivações –** define como um conhecimento em uma forma pode se transformar em outro conhecimento;
- **Restrições –** restringe as possíveis estruturas e o comportamento dos objetos e processos;
- **Existência –** define quando alguma coisa deve existir e quando ela pode ser destruída.

As Extensões de Negócio Eriksson-Penker usa uma nota estereotipada para definir as regras. Esta nota se une explicitamente a um elemento específico ou parte dele no diagrama e define informais regras suaves. Uma regra suave pode ser um axioma, um princípio, ou uma regra que implica o uso do juízo humano. As regras podem estar presentes nas visões e nos diagramas. Um exemplo pode ser observado na Figura 4.7 que

representa a regra para que uma pessoa possa alugar uma propriedade. Onde uma pessoa pode assinar nenhum ou vários contratos de aluguéis, e cada contrato de aluguel entrega uma propriedade, que é um objeto alugado. A regra do valor do aluguel e definida como equivale a 11,5% do valor da propriedade.

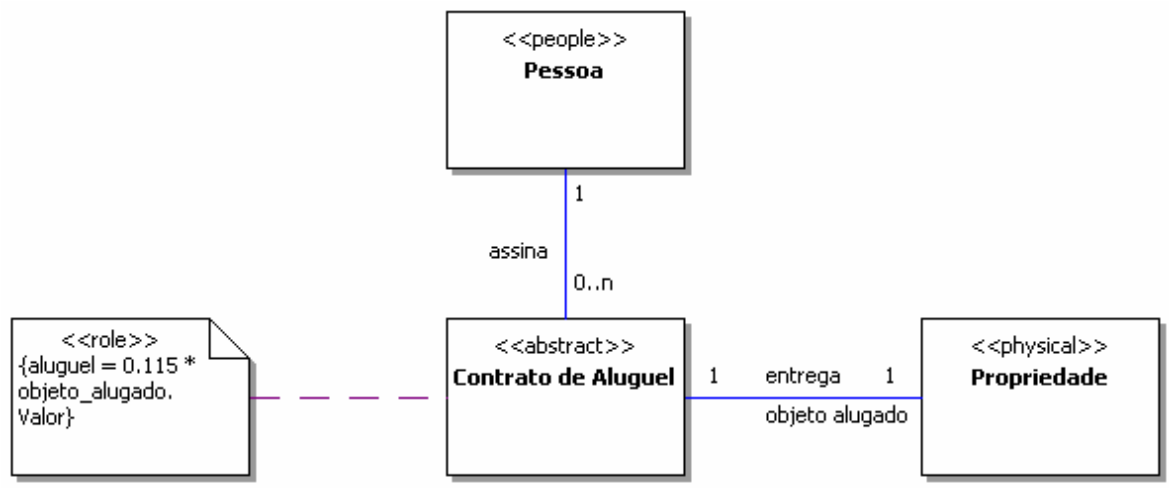


Figura 4.7 - Regras de negócio em um diagrama de classes como restrição, regra ou relacionamentos múltiplos. Fonte: adaptado de Eriksson & Penker (2000).

4.3.8. Relacionamentos

Em adicional aos relacionamentos e às estruturas dentro de cada categoria de recursos, há também relacionamentos entre diferentes categorias de conceitos.

Os relacionamentos mais comuns usados na UML são:

- **Transição de estado/atividade –** um relacionamento entre estados ou atividades indicando quem precede o outro, mostrando a ordem das atividades. Também é conhecido como Relacionamentos Temporais.

Generalização – um relacionamento de generalização/especialização, onde os objetos especializados são substituíveis pelos os objetos mais generalizados.

- **Associação/agregação –** um relacionamento que descreve um conjunto de *links*, e cada *link* é uma conexão entre os objetos.

- **Dependência –** um relacionamento entre dois elementos de modo que um depende do outro;

- **Refinamento –** uma relação entre elementos, onde um elemento é mais refinado do que o outro.

4.3.9. Mecanismos Gerais

Há somente um novo mecanismo geral na Extensão de Negócio Eriksson-Penker: a Nota de Referência. A nota de referência é uma nota estereotipada que contém uma referência para outro diagrama ou outro documento, como pode ser visto na Figura 4.8.

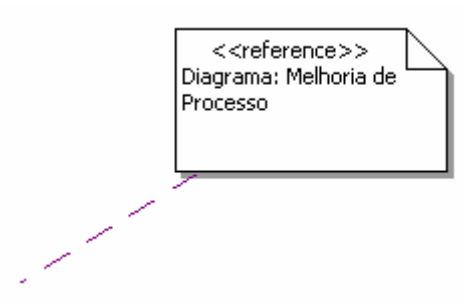


Figura 4.8 - Uma nota de referencia permite ao modelador visualizar referencia de um outro diagrama ou outro documento. Fonte: Adaptação de Eriksson & Penker (2000).

4.4. Visões de Negócios

O método Eriksson-Penker usa quatro diferentes visões de um negócio para poder realizar a modelagem de um negócio, e são elas: Visão dos Negócios, Processo dos Negócios, Estrutura dos Negócios, Comportamento dos Negócios.

4.4.1. Visão de Negócios

A Visão de Negócios descreve os objetivos da empresa. Esta visão forma a estratégia geral do negócio, define os objetivos do negócio, e atua como um guia para modelar as outras visões. A Visão de Negócios também pode ser usada como um instrumento motivacional. Ela deve ser compartilhada com os funcionários do negócio, e prover as informações e recursos, com também definir responsabilidades, para alcançar aquelas visões.

De acordo com Darnton & Darnton (1997), há alguns fatores importantes para serem considerados quando estamos criando a Visão de Negócios:

- Missão da Companhia – objetivo geral da companhia.

- Objetivos – objetivos mais específicos.
- Pontos Fortes – aspectos específicos no qual a companhia se sobre-sai.
- Pontos Fracos – aspectos específicos no qual a companhia deve melhorar.
- Oportunidades – áreas de crescimento potencial para o futuro da companhia.
- Ameaças – Condições externas que podem afetar negativamente a companhia.
- Fatores Críticos – Elementos que são requeridos para o crescimento ou sucesso do negócio.
- Estratégias – Plano de ações para alcançar os objetivos.
- Competências Principais – áreas do negócio que são de maior importância.
- Papéis – as funções específicas das pessoas que trabalham no negócio.
- Unidades Organizacionais – Grupos nos quais o negócio é dividido.

O resultado final da Visão de Negócios é a definição do estado futuro e desejado da companhia, e como alcançar este estado. O resultado primário é expresso em uma visão descritiva, e alguns modelos de objetivos/problemas. A visão descritiva é um documento textual curto que descreve a visão da companhia daqui alguns anos. O modelo de objetivos/problemas é um diagrama formal que descreve os principais objetivos e problemas da companhia.

Há três técnicas usadas na Visão de Negócios: Definição Estratégica, Modelo Conceitual e Modelo de objetivos/problemas.

Definição Estratégica

A estratégia é normalmente baseada em conclusões que resultaram de avaliar os processos principais do negócio ou por criação de novos processos. A estratégia mostra a direção na qual o negócio é conduzido, os processos do negócio, bem como a organização, então devem ser conseqüentemente adaptados.

A seguir segue as questões tipicamente consideradas em um plano estratégico:

- Clientes – Quem são os clientes e quais características eles têm? Como é a interação do cliente com a modificação de negócios?

- Competidores – Quem são os concorrentes e o que eles estão fazendo? De que maneira eles estão modificando o seu modelo de negócios?

- Tamanho e posição na Indústria – como o negócio é posicionado na indústria? Ele tem de expandir-se para aumentar a ação de mercado? Como a indústria se está modificando?

- Rentabilidade e crescimento – como está a rentabilidade e crescimento do negócio comparado com outros negócios no mesmo domínio?

- O mundo circundante - que modificações (especificamente políticas ou tecnológicas) estão acontecendo?

- Percepção pública – como o público percebe a companhia? Em que caminhos é desejável modificar esta percepção? A companhia precisa de uma nova imagem pública?

- Nível de serviço. Qual é o nível de serviço com o cliente? O serviço pode ser melhorado ou estendido?

Diagrama de Visão Descritiva

Uma técnica que pode ser utilizada é a visão descritiva, que é um documento textual que descreve o futuro desejado da companhia. Nesta visão é sumarizado a estratégia da companhia. Nele contém o contexto atual do negócio, as exigências do negócio, os problemas que podem ser visualizados, e os cenários possíveis de se acontecer. A visão descritiva é uma descrição de como a companhia deve ficar, como está, e que tipos de resultados são esperados. Esta visão deve conter os objetivos alto-níveis claramente definidos. Ela também apresenta meios de medir quando um objetivo alto-nível é atingido.

Modelo Conceitual

Um modelo conceitual define os conceitos mais importantes utilizados no negócio. Este modelo estabelece um vocabulário comum para todos os conceitos, e demonstra as relações entre os diferentes conceitos.

O modelo conceitual é apresentado como o diagrama de classes da UML. Os nomes das classes e associações usadas no modelo são importantes, uma vez que eles são os conceitos que foram definidos. As classes podem ter atributos significantes para ajudar a descrever os conceitos, bem como uma explicação textual definida na propriedade Documentação da UML. Este modelo pode ser incrementado com novos conceitos, de acordo com o desenvolvimento da modelagem, não se restringindo apenas aos conceitos definidos neste primeiro ponto.

Neste diagrama, os atributos e operações não são tão importantes como um diagrama usado para representar classes de um software, o mais importante neste diagrama é capturar os conceitos e como eles se relacionam. Um exemplo pode ser visto na Figura 4.9, que é a modelagem de conceitos elaborada no estudo de caso, onde está sendo explicada na seção 7.2.3.

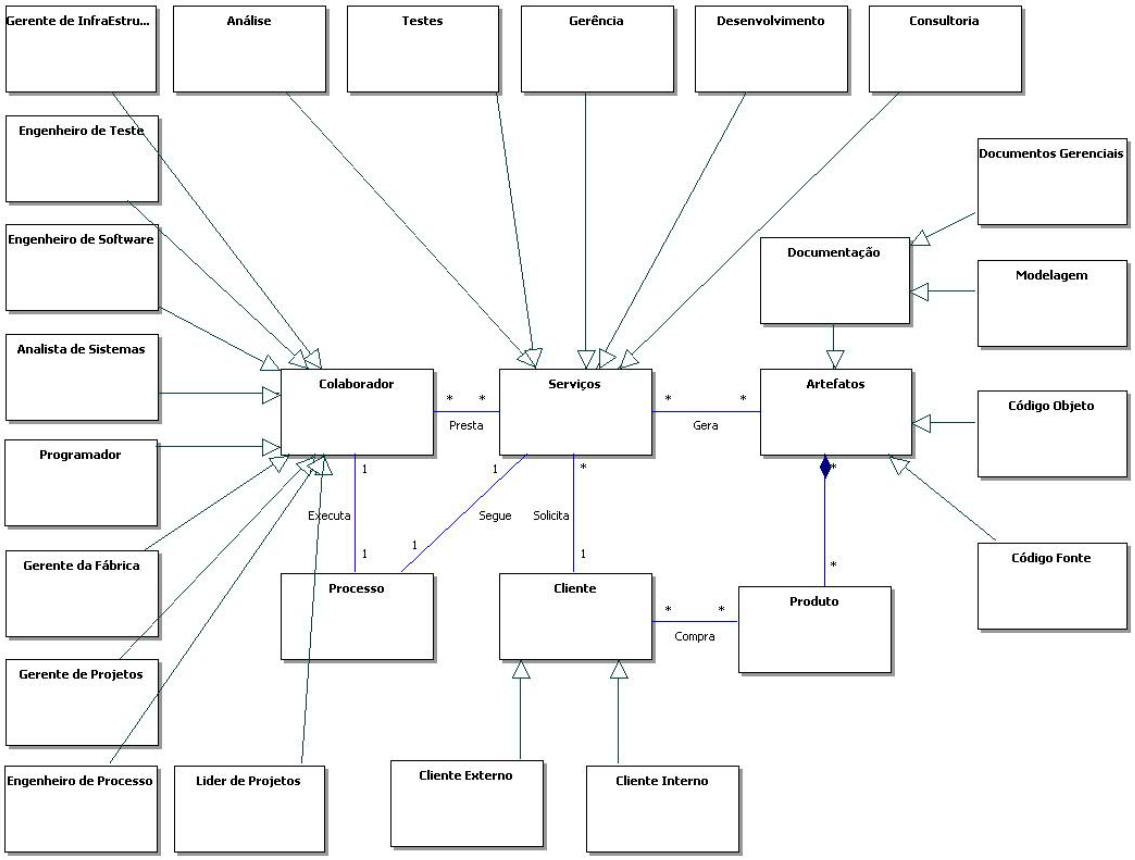


Figura 4.9 - Exemplo de Modelo Conceitual. Fonte: Elaborado pelo autor.

Modelagem de Objetivos/Problemas

Um modelo de objetivos descreve os objetivos do negócio e os problemas que impedem que se atinjam estes objetivos. Objetivos controlam o comportamento do negócio e mostra um estado desejado de alguns recursos no negócio. O modelo de objetivos estabelece por que o negócio existe, o que o negócio está tentando atingir, e o qual a estratégia para o negócio atingir estes objetivos.

O diagrama objetivos/problemas é representado pelo diagrama de objetos da UML. O modelo de objetivos completo é representado por vários diagramas de objetos, onde cada diagrama mostra um objetivo alto-nível específico decomposto em sub-objetivos. Objetivos são representados por objetos de classes que possuem o estereótipo <<objetivo>>.

No método Eriksson-Penker tem-se duas classes já definidas para representar objetivos, que são Objetivo Quantitativo e Objetivo Qualitativo. Um Objetivo Quantitativo pode ser facilmente medido através de algum valor que deve ser atingido. Um Objetivo Qualitativo é difícil de descrever em termos mensuráveis, e envolve julgamentos humanos. Ambos os tipos de objetivos possuem uma descrição do objetivo como um atributo. Um objetivo quantitativo possui também um valor de objetivo, um valor atual e uma unidade de medida. Um objetivo específico no negócio é descrito como um objeto da classe objetivo. As relações entre os objetivos são representadas por dependências e associações.

Uma dependência é representada por uma linha tracejada de um super-objetivo para um sub-objetivo, terminando com uma seta aberta. Pode-se interpretar, com esta dependência, que a realização completa do sub-objetivo contribui para a realização do super-objetivo, isto é, para completar o super-objetivo, deve-se completar todos os sub-objetivos. Se um objetivo pode ser completamente decomposto em sub-objetivos, uma linha pontilhada é desenhada através de todas as dependências e uma restrição é escrita ao lado da linha: {completo}. Se um objetivo não pode ser completamente decomposto em sub-objetivos, {incompleto} é escrito (este é o padrão, para caso nada seja escrito).

Um outro conceito chave no diagrama de objetivos/problemas é um problema. Um problema sempre é ligado a um objetivo. Um problema também pode ser decomposto em sub-problemas. Por ser ligado a objetivos, esta estrutura geralmente é mostrada indiretamente em uma hierarquia de objetivos. Um problema é representado com o

elemento nota da UML, com o estereótipo de <<problema>> ligado a seu objeto objetivo. Problemas são eliminados (resolvidos) por ações.

Um plano de ações pode ser formulado de um modelo de objetivos. Este contém uma lista de problemas, a causa de cada problema, a ação apropriada para cada problema, os pré-requisitos de cada ação e, finalmente, o recurso ou processo responsável para resolvê-lo. Isto pode ser mostrado visualmente no diagrama de objetivos/problemas através do uso de notas estereotipadas, como <<problema>>, <<causa>>, <<ação>> e <<pré-requisitos>>. Um problema é ligado a um objetivo, uma causa é ligada a um problema e assim por diante. Um exemplo de diagrama de objetivos/problemas pode ser visto na Figura 4.10, que representa o diagrama realizado no estudo de caso que é apresentado na seção 7.2.4.

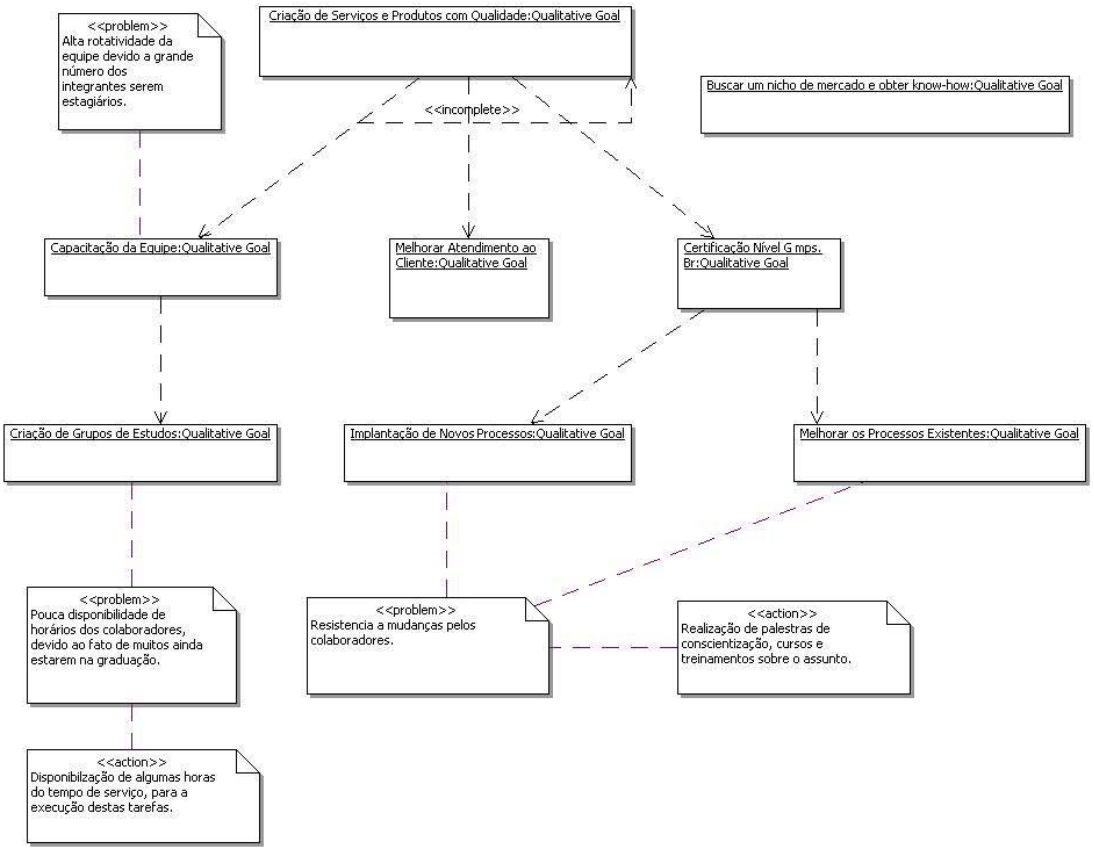


Figura 4.10 – Exemplo de diagrama de objetivos/problemas. Fonte: Elaborado pelo autor.

4.4.2. Visão de Processo de Negócios

A visão de Processo de Negócios é o centro da modelagem de negócios. Os processos mostram as atividades de devem ser feitas para atingir um objetivo explícito, junto de seus relacionamentos com os recursos participantes no processo. Os processos possuem um propósito e objetivos específicos, e todos os processos coletivamente tentam atingir os objetivos gerais do negócio.

A visão de Processo de Negócios é descrita com um diagrama de atividades da UML, junto com um conjunto de estereótipos disponibilizado pelo método Eriksson-Penker. Em adicional, uma variante do diagrama de processos, chamado de diagrama de linha de montagem, é usada para descrever mais claramente como o processo interage com recursos durante sua execução.

É essencial definir o seguinte quando está modelando negócios na ordem de identificar e especificar os processos do negócio:

- Quais atividades são requeridas?
- Quando uma atividade é executada, e em qual ordem?
- Porque as atividades são executadas; quais são os objetivos do processo?
- Como as atividades são executadas?
- Quem ou o quê está envolvido na execução das atividades?
- O que está sendo consumido ou produzido?
- Como as atividades devem ser executadas?
- Quem controla o processo?
- Como o processo está relacionado à organização do negócio?
- Como o processo se relaciona a outros processos?

Diagrama de Processo

Um diagrama de processo é representado pelo diagrama de atividades da UML, como um conjunto de estereótipos que descrevem as atividades executadas dentro dos

processos e como eles se interagem, os objetos de entrada e saída, o fornecimento e controle de recursos que participam do processo e o objetivo do processo. A Figura 4.11 mostra um diagrama de processos genérico.

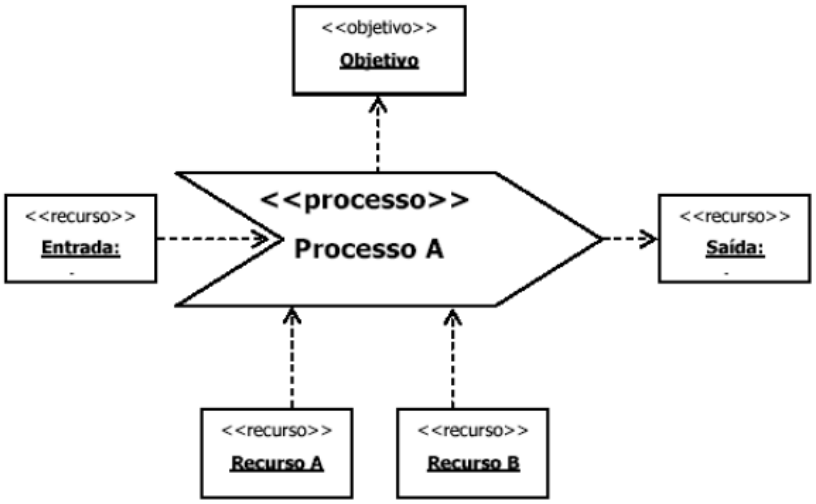


Figura 4.11 - Um genérico diagrama de processos. Fonte: Adaptado de Eriksson & Penker (2000).

Um processo é uma atividade estereotipada de <<processo>>. Um processo pode conter subprocessos. Um símbolo de atividade é utilizado para indicar processos que não podem ser decompostos, isto é, são atividades atômicas.

Objetos de recursos e objetos que são envolvidos no processo são colocados em volta do processo. Estes objetos são:

- **Objetos de Objetivos** – um objeto de objetivo representa um objetivo que o negócio tenta atingir. Este objeto é ligado com o processo através da dependência estereotipada de <<alcança>>, indicando que o processo tenta alcançar o objetivo.

- **Objetos de Entrada** – objetos que são consumidos ou refinados pelo processo. Estes objetos são recursos, e como tais podem ser estereotipados como <<físico>>, <<abstrato>>, <<pessoa>> ou <<informação>>. São ligados ao processo com uma linha pontilhada, e geralmente estes objetos são colocados do lado esquerdo do processo.

- **Objetos de Saída** – objetos que são produzidos pelo processo ou que são o resultado do refinamento de um ou mais objetos de entrada. Os objetos de saída

são ligados a partir do processo por uma linha pontilhada. Objetos de saídas são postos do lado direito do processo.

- **Objetos de Suprimento** – objetos que participam no processo, mas não são refinados ou consumidos. Estes objetos são desenhados debaixo do processo e são ligados ao processo por uma linha pontilhada. Esta dependência possui o estereótipo de <<fornece>>

- **Objetos de Controle** – recursos que controlam ou executam o processo. Estes objetos são geralmente colocados abaixo do processo, ligados por uma linha pontilhada, com a dependência estereotipada de <<controla>>.

Uma técnica utilizada é a inserção de *swimlanes* para descrever onde a atividade é executada quanto a organização do negócio.

Caso um modelo possua muitos processos, pode-se utilizar os pacotes disponíveis da UML para organizar os modelos.

O método Eriksson-Penker introduz um novo diagrama para ser utilizado, o Diagrama de Linha de Montagem. Este diagrama é baseado no diagrama de atividades da UML, e é utilizado, em particular, quando o objetivo da modelagem é a produção de sistemas de informações que apóiam os processos.

Na parte superior deste diagrama, se tem um diagrama de processo, e abaixo há um número de pacotes horizontais que representam um grupo de objetos. O propósito deste diagrama é mostrar como o processo na parte superior do diagrama, escreve e lê objetos na parte inferior. Um exemplo deste diagrama é apresentado na Figura 4.12.

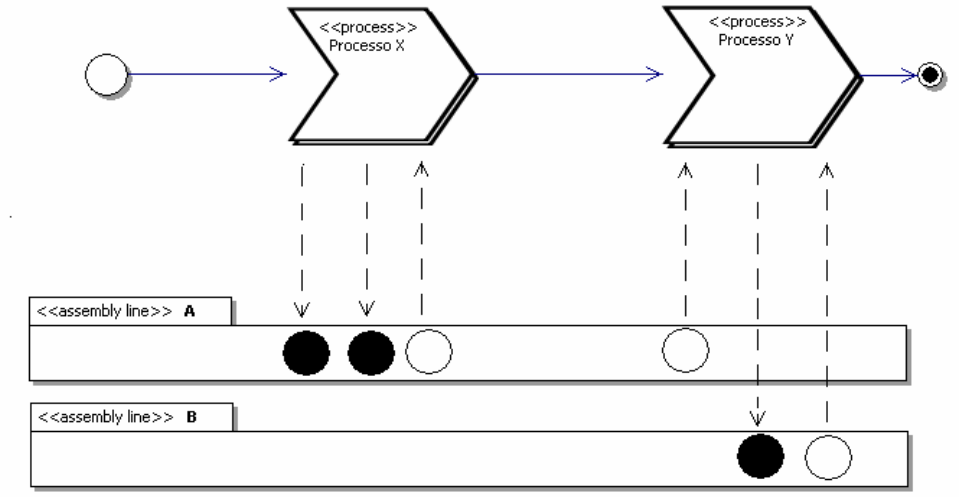


Figura 4.12 - Exemplo de diagrama de Linha de Montagem. Fonte: Adaptação de Eriksson & Penker (2000).

4.4.3. Visão de Estrutura de Negócios

A visão de Estrutura de Negócios mostra a estrutura dos recursos, dos produtos, ou dos serviços, e as informações no negócio, incluindo a tradicional organização da companhia (divisões, departamentos, seções, entre outros).

Os diagramas da UML utilizados para documentar esta visão são os diagramas de classe e os diagramas de objetos. Os diagramas de classes mostram a estrutura principal, e os diagramas de objetos mostra a atual configuração do diagrama de classe (ex. como uma organização está em um determinado momento).

Modelagem de Recurso

Modelos de recursos mostram a estrutura de diferentes recursos. O modelo genérico é descrito em um diagrama de classe, enquanto uma configuração atual da estrutura é mostrada no diagrama de objetos. A estrutura interna dos recursos, que são normalmente produtos ou serviços oferecidos pela companhia, pode ser apresentados em um modelo de recurso. A diferença entre um modelo de recurso e um modelo conceitual (como mostrado na visão de Visão de Negócios) é que o recurso se concentra na modelagem das estruturas mais concretas de recursos, como produtos ou serviços, enquanto o modelo conceptual se concentra na definição do significado e as relações dos conceitos importantes usados para definir o negócio. Um exemplo de modelo de recurso é exibido na Figura 4.13.

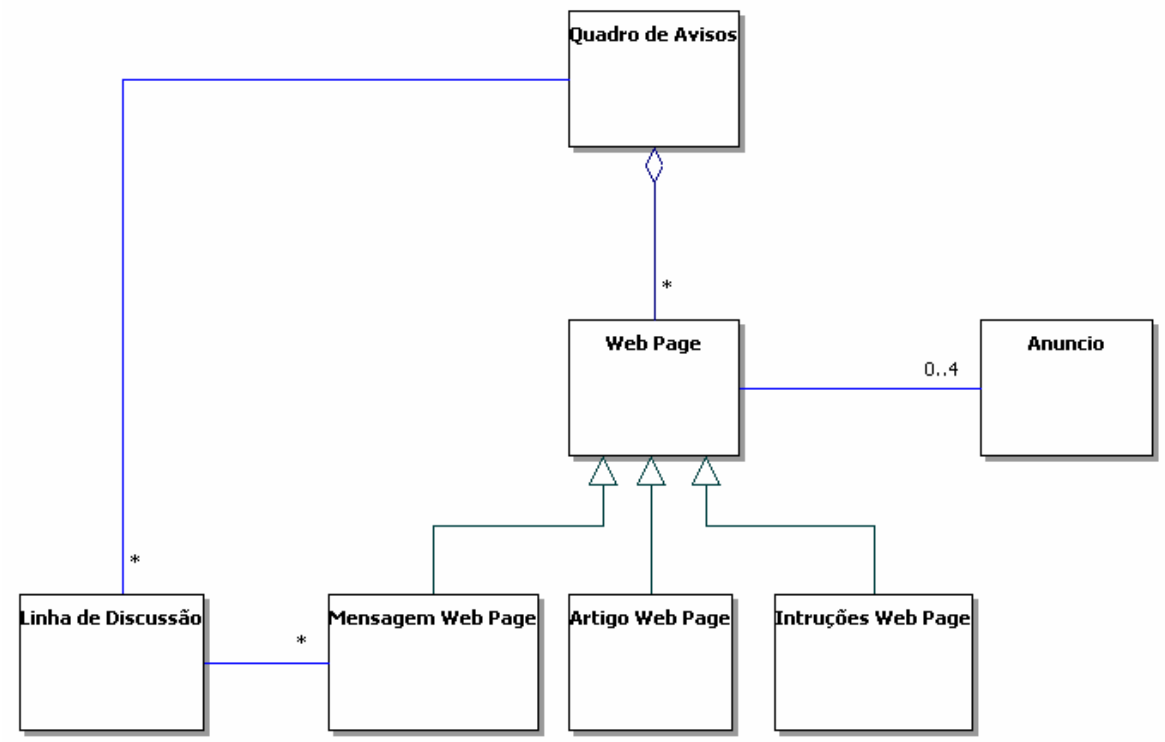


Figura 4.13 - Exemplo de Modelagem de Recursos. Fonte: Adaptado de Eriksson & Penker (2000).

Modelagem de Informações

Modelagem de informações cria modelos de informações estrategicamente importantes para o negócio. Embora a informação também seja um recurso, ela é modelada separadamente. A informação é o que servirá como entrada para os sistemas de informações que apóiam o negócio, a informação possui um valor estratégico para o negócio.

Modelagem Organizacional

A modelagem organizacional é outro caso especial da modelagem de recurso, onde os recursos são alocados em unidades organizacionais, que estão relacionadas uma com as outras de acordo com regras específicas.

A estrutura completa de uma organização por ser expressa por diagramas de classes e diagramas de objetos. O digrama de classe especifica a estrutura básica e as regras da organização e o diagrama de objeto mostra a situação real da organização. Os recursos são alocados em um diagrama de objeto no qual os objetos de recurso são ligados a objetos de organização, e os processos são ligados a organização por *swimlanes* em um diagrama de processo.

O diagrama de classe descreve os nomes das unidades organizacionais e as regras de negócios para arranjá-los e ligá-los um a outro. A Figura 4.14 mostra um diagrama de classe de uma companhia com uma equipe de gerência que é organizada em divisões. As divisões à sua vez são organizadas em seções.

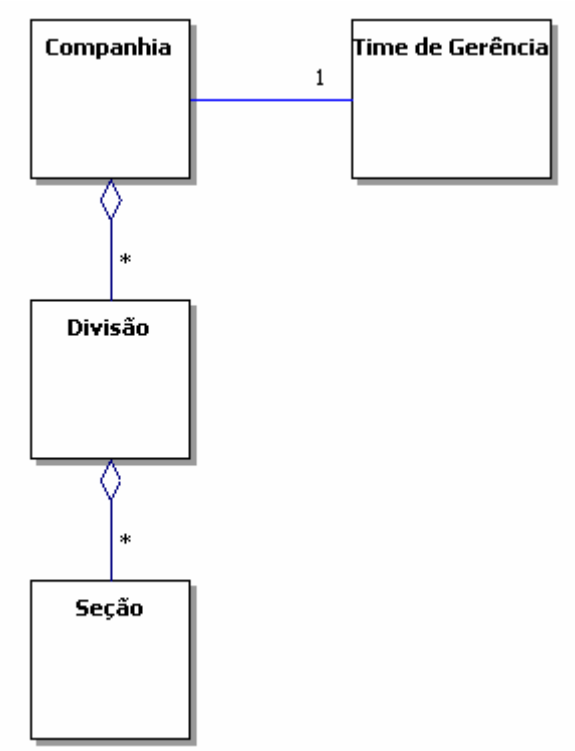


Figura 4.14 - Modelo Organizacional como Diagrama de Classe. Fonte: Adaptação de Eriksson & Penker (2000).

4.4.4. Visão de Comportamento de Negócios

A visão de comportamento de negócios mostra tanto o comportamento individual de recursos e processos no negócio bem como a interação destes entre os outros recursos e processos. O comportamento dos objetos de recurso é governado pela visão de processo de negócios, que mostra o fluxo de controle principal de todo o trabalho executado. A visão de Comportamento de Negócios investiga cada um dos objetos envolvidos mais detalhadamente: o seu estado, o seu comportamento em cada estado, e transições possíveis. A visão de comportamento de negócios pode ser um instrumento importante para definir as responsabilidades exatas de cada atividade, e definir o comportamento exato de cada recurso alocado em um processo.

Os estados combinados dos processos e objetos definem a condição atual do sistema. Os estados são modificados pela operação do sistema, isto é, pelos processos. Quando um estado é alterado, eventos são disparados para notificar os outros processos sobre a modificação. A visão de comportamento de processos é caracterizada pelos diagramas dinâmicos da UML: estado, seqüência, colaboração, processos e diagramas de linha de montagem.

A diferença entre a visão de comportamento de negócios e a visão de processo de negócios é que este último ilustra as atividades do sistema, as transformações e funcionalidades, enquanto a visão de comportamento de negócios ilustra o comportamento dinâmico de cada um dos objetos relacionados nessas atividades. Naturalmente, deve haver coerência entre essas duas visões.

Modelagem de Estados

A modelagem de negócios mostra o comportamento de um individual recurso pela identificação dos possíveis estados deste recurso e o comportamento do objeto deste recurso em cada estado. O comportamento de um recurso é descrito usando o digrama de estados a UML com os seguintes conceitos chaves:

- **Estados:** os diferentes estados que o objeto pode ter, incluindo os estados inicial e final.
- **Eventos:** a causa da mudança de estado. Os eventos que podem ser enviado a um recurso são mostrados como operações na classe do recurso.
- **Ações:** as atividades executadas em um estado específico ou na transição de um estado para outro. As ações executadas são modeladas como as ações tomadas dentro de uma operação na classe de recurso.

Normalmente, os estados de recurso, não de processos, são mostrados quando modelar os estados. Os estados diferentes em um processo são as atividades (ex., sub-processos) no processo. Um diagrama de estados para um processo é muito similar a um diagrama de processos e não adicionado nenhuma informação significativa.

Modelagem de interação

O comportamento e um sistema de negócio é também composto por interações entre processos e interações entre recursos. Estas interações podem ser mostradas nos diagramas dinâmicos da UML, como o diagrama de seqüência ou colaboração. Isto é chamado de modelagem de interação. Os diagramas de estados modelam o comportamento individual de um recurso especifico, enquanto os diagramas de seqüência e colaboração mostram o comportamento, a interação, que ocorrem entre diferentes recursos.

Diagramas de Seqüência e Colaboração

A técnica tradicional para descrever interações entre objetos na UML são os diagramas de seqüência e colaboração. Ambos os diagramas mostram como um conjunto de objetos interage através de operações chamadas em um cenário especifico. Diagramas de seqüência e colaboração podem ser usados para mostrar as cooperações detalhadas de um numero de recursos de objetos.

A interação descrita em um diagrama de seqüência ou colaboração é disparada por uma referência a um processo para um objeto em um diagrama de linha de montagem. Os diagramas de seqüência e colaboração mostram as interações detalhadas entre objetos de diferentes pacotes. A referencia de um processo para u pacote de linha de montagem em um diagrama de linha de montagem disparam a interação entre um numero de recursos em um diagrama de linha de montagem. A interação e modelada para permitir os recursos chamarem ações uns sobre os outros.

Diagrama de Processos

A interação entre processos pode ser mostrada em um diagrama de processo, o qual te faz lembrar um diagrama de atividades UML com estereótipos das extensões. Os objetos de saída de um processo são os objetos de entrada de outro processo. Este é o caso quando os sub-processos de um processo são mostrados, assim como quando dois processos independentes são mostrados no mesmo diagrama. Note que um processo não é modelado como uma classe única como um recurso; isto é na prática uma abstração de interações e atividades executadas por um número de recursos.

4.5. Considerações Finais

Neste capítulo foi apresentado o método de modelagem de negócio Eriksson-Penker que utiliza a UML. O capítulo mostra a estrutura do método que é divido em quatro visões, a visão Visão de Negócios, a visão Processos de negócios, a visão Estrutura de Negócios e a visão Comportamento de Negócios, os diagramas e modelos que compõem cada visão e os elementos utilizados da UML para a construção dos modelos e diagramas.

5. METODOLOGIA

A metodologia utilizada na realização deste trabalho foi dividida em três etapas, descritas nas seções a seguir:

5.1. Pesquisa sobre UML e Modelagem de Negócio

Esta primeira etapa teve como foco a pesquisa e estudo sobre a UML e sobre alguns métodos de modelagem de negócios, em destaque o Método Eriksson-Penker.

Esta etapa teve a característica exploratória, buscando conhecimento sobre os temas que seriam abordados no trabalho. Foi utilizada a pesquisa bibliográfica como abordagem metodológica desta fase.

De acordo com Jung (2004), a pesquisa bibliográfica tem por finalidade principal formar uma consistente base “mental” a partir daquilo que é existente. Portanto, proporciona uma ampla aquisição de conhecimentos para o entendimento substancial do assunto, viabilizando ao pesquisador “ousar” ao propor novos argumentos que justifiquem as descobertas.

A pesquisa bibliográfica foi baseada basicamente em trabalhos impressos disponíveis em bibliotecas e conteúdos disponíveis na *Internet*.

5.2. Adaptação do Método de Modelagem de Negócio

Esta etapa constituiu na adaptação do método de modelagem de negócio Eriksson-Penker que utiliza a UML para a UML 2.0.

A adaptação foi realizada utilizando com o auxílio da ferramenta *Borland® Together® Architect 2006 Service Pack 1 for Eclipse Trial* (Borland, 2006), que disponibiliza um ambiente para a adaptação dos elementos de modelagem contida no método Eriksson-Penker para UML 2.0.

5.3. Estudo de Caso em uma Fábrica de Software

Esta etapa constituiu do estudo de caso em uma fábrica de software, onde foi aplicada a adaptação do método. O estudo de caso, conforme Jung (2004), tem por finalidade entender “como” e “por que” funcionam as “coisas”.

Segundo Jung (2004), pode-se definir um estudo de caso como sendo um procedimento de pesquisa que investiga um fenômeno dentro do contexto local, real e especialmente quando os limites dos entre fenômeno e o contexto não estão claramente definidos.

Foi escolhido uma fábrica de software, devido a facilidade de se realizar o trabalho e necessidade da empresa em conhecer os seus objetivos e documentar sua estrutura de modo formal.

Para a realização da modelagem de negócios da fábrica, foram realizadas diversas reuniões com pessoas chave que possuíam um conhecimento de como é executado o negócio da fábrica de software e sua estrutura.

6. ADAPTAÇÃO DO MÉTODO ERIKSSON-PENKER PARA UML 2.0

A UML possui mecanismos para criar novos elementos de modelagem e ampliar e/ou acrescentar domínios[4]. Estes mecanismos são: estereótipos (*stereotype*), valores atribuídos (*tagged values*) e restrições (*constraints*).

Os estereótipos na UML 2.0 são desdobramentos de uma meta-classe ou de outro estereótipo, o que resulta uma nova meta-classe que permite um meio de estender o meta-modelo da UML, admissível.

Este capítulo apresenta a adaptação do método Eriksson-Penker para UML 2.0.

6.1. Elementos da Extensão de Negócios Eriksson-Penker

O conjunto de elementos adicionais para a UML no método Eriksson-Penker para a modelagem de negócios permite modelar negócios de maneira rápida e com qualidade.

A extensão de negócios Eriksson-Penker disponibiliza, no seu conjunto de extensões para UML, visões (*views*), diagramas (*diagrams*), modelos (*models*), restrições (*constraints*), valores atribuídos (*tagged values*) e estereótipos (*stereotypes*).

Estes elementos serão adaptados para que seja incorporaradas as características da UML 2.0.

A estrutura do método Eriksson-Penker é descrita a seguir:

Business Vision **(Visão de Negócios)** – mostra a visão geral, os conceitos-chave, os objetivos e os problemas a serem eliminados. Os diagramas e visões que compõem a visão de negócios são:

- *Vision Statement Diagram* (Diagrama de Visão Descritiva) - descreve a visão geral do negócio. Este diagrama é expresso em texto corrido.

[4] Domínio no contexto do presente trabalho, deve ser entendido como o "universo" (conjunto de tudo quanto existe)que a UML é capaz de representar.

- *Conceptual Model* (Modelo Conceitual) – da ênfase a definição dos conceitos-chave do negócio é expresso em um diagrama de classes da UML. O modelo conceitual não utiliza estereótipos, usa somente os elementos disponíveis no diagrama de classes, onde um conceito é representado por uma classe.

- *Goal/Problem Model* (Modelo de Objetivos/Problemas) – descreve os objetivos e os problemas que impedem atingir estes objetivos. O modelo de objeticos/problemas é expresso em um diagrama de objetos da UML. A Tabela 6.1 apresenta os estereótipos e restrições utilizados neste diagrama.

Tabela 6.1 - Elementos do Modelo *Goal/Problem Model* que serão adaptados.

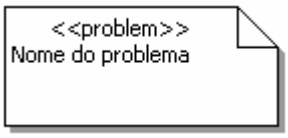
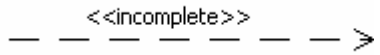
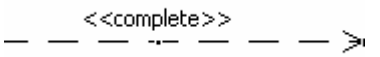
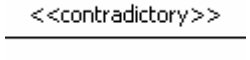
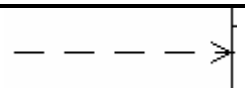
ELEMENTOS	DESCRIÇÃO
 Goal (Objetivo)	Representa os estados desejados, denota os objetivos que motivam as ações e levam as mudanças de estados em uma direção desejada.
 Problem (Problema)	Representa os problemas que podem impedir ou atrapalhar a atingir um objetivo definido.
 *Incomplete Goal Decomposition* (Decomposição Incompleta do Objetivo)	Representa os objetivos decompostos de modo incompleto em sub-objetivos. Isto é, quando é alcançado os sub-objetivos, não necessariamente atinge-se o objetivo.
 *Complete Goal Decomposition* (Decomposição Completa do Objetivo)	Representa os objetivos decompostos de modo completo em sub-objetivos. Isto é, quando é alcançado os sub-objetivos, necessariamente atinge-se o objetivo
 *Contradictory Goal* (Objetivo Contraditório)	Representa os objetivos que podem ser contraditórios, mas tem de ser cumpridos.

Tabela 6.1 - Elementos do Modelo *Goal/Problem Model* que serão adaptados.

 *Goal Dependency* (Dependência Objetivo)	Representa os objetivos que são organizados em hierarquias de dependências, ou seja, em que um objetivo depende de outros objetivos.

Fonte: Adaptado de Eriksson & Penker (2000).

Visão *Business Process* (Processos de Negócios) – da ênfase aos processos de negócios, ilustra a interação dos processos e o uso dos recursos para realizar os objetivos do negócio. Os diagramas e visões que compõem a visão de processos de negócios são:

- *Process Diagram* (Diagrama de Processos) – apresenta os processos do negócio e os colaboradores. O diagrama de processos é uma particularização do diagrama de atividades da UML. Os estereótipos e restrições utilizados neste diagrama são apresentados na Tabela 6.2.

Tabela 6.2 - Elementos do Diagrama *Process Diagram* que serão adaptados.

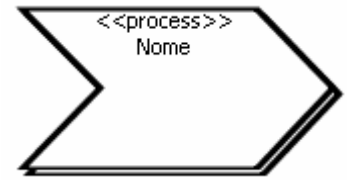
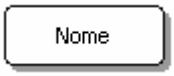
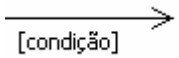
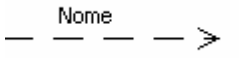
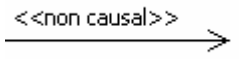
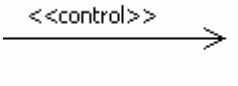
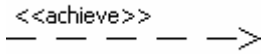
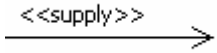
ELEMENTOS	DESCRIÇÃO
 *Process* (Processo)	Representa os processos compostos por um conjunto de atividades relatadas que quando corretamente executadas satisfazem um objetivo explícito.
 *Activity (Atomic Process)* (Atividade, Processo Atômico)	Representa um processo atômico, isto é, que não pode ser divido.
 *Process Flow* (Fluxo de Processo)	Representa o controle de fluxo de processos com uma condição.
 *Resource Flow* (Fluxo de Recurso)	Representa a utilização ou produção de um recurso por uma atividade.
 *Noncausal Resource Flow* (Fluxo de Recurso não obrigatório)	Representa a partir de quando um recurso pode ser utilizado ou produzido por uma atividade.

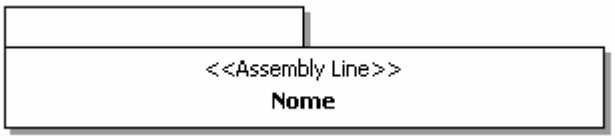
Tabela 6.2 - Elementos do Diagrama *Process Diagram* que serão adaptados.

 *Process Control* (Controle de Processo)	Representa a partir de quando um processo é controlado por um objeto.
 *Goal Connection* (Conexão do Objetivo)	Representa a ligação de um objetivo a um processo.
 *Process Supply* (Fornecedor de Processo)	Representa o processo fornecido por um objeto.

Fonte: Adaptado de Eriksson & Penker (2000).

- *Assembly Line Diagram* (Diagrama de Linha de Montagem) – da ênfase a conexão entre os processos e os objetos envolvidos. O diagrama de linha de montagem é uma particularização do diagrama de atividades da UML. Os estereótipos e restrições utilizados neste diagrama são apresentados na Tabela 6.3.

Tabela 6.3 - Elementos do Diagrama *Assembly Line Diagram.*

ELEMENTOS	DESCRIÇÃO
 *Assembly Line* (Linha de Montagem)	Representa a sincroniza dos processos com os objetos que eles utilizam.
 *Writed Object* (Objeto Escrito)	Representa o objeto escrito na linha de montagem.
 *Read Object* (Objeto Lido)	Representa o objeto que é lido da linha de montagem.

Fonte: Eriksson & Penker (2000).

- **Visão *Business Structure* (Estrutura de Negócios)** – da ênfase as estruturas de recursos utilizadas como unidades organizacionais, produtos, documentos, informações e conhecimentos.

    - *Resource Model* (Modelo de Recursos) – captura os recursos de um negócio que pode ser informações ou objetos que podem ser abstratas ou concretas. Objetos concretos incluem pessoas, máquinas e itens; coisas abstratas geralmente são unidades organizacionais, departamentos, entre outros. O *Resource Model* é representado em um diagrama de classe da UML. Os estereótipos e restrições utilizados neste diagrama são mostrados na Tabela 6.4.

Tabela 6.4 - Elementos do Modelo *Resource Model*.

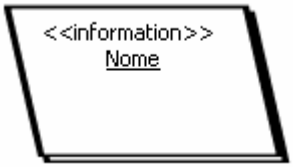
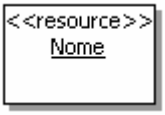
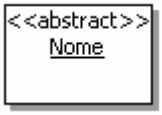
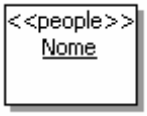
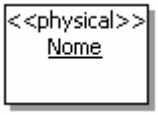
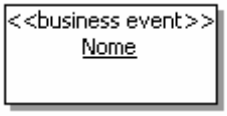
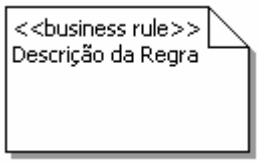
SÍMBOLO	DESCRIÇÃO
 *Information* (Informação)	Representa uma informação, que também pode ser considerada um recurso.
 *Resource* (Recurso)	Representa um recurso.
 *Abstract Resource* (Recurso Abstrato)	Representa um recurso abstrato.
 *People* (Pessoas)	Representa uma ou mais pessoas.
 *Physical Resource* (Recurso Físico)	Representa um recurso físico.
 *Business Event* (Evento de Negócios)	Representa um evento de negócios.

Tabela 6.4 - Elementos do Modelo *Resource Model* (Continuação).

 *Business Rule* (Regra de Negócios)	Representa uma regra de negócios.

Fonte: Eriksson & Penker (2000).

- *Organization Model* (Modelo Organizacional) – Apresenta as estruturas organizacionais de um negócio. É um caso especial do *Resource Model*. O *Organization Model* é representado em um diagrama de classe da UML e em alguns casos com o diagrama de objetos. Os estereótipos e restrições utilizados neste diagrama são os mesmos utilizados no *Resource Model*. A Tabela 6.4 apresenta os estereótipos e as restrições utilizados no diagrama *Resource Model*.

- *Information Model* (Modelo de Informações) – apresenta as informações de uma maneira estruturada para facilitar decisões. O *Information Model* é também um caso especial do *Resource Model* e é representado em um Diagrama de Classe ou Diagrama de Objetos da UML. Os estereótipos e restrições utilizados neste diagrama são mostrados na Tabela 6.4.

- **Visão *Business Behavior* (Comportamento de Negócios)** – dá ênfase aos comportamentos individuais e as interações de processos como recursos.

  - *Statechart Diagram* (Diagrama de Estados) – é um diagrama padrão da UML usado para exprimir o comportamento de recursos, usando os elementos deste diagrama.

  - *Interaction Diagrams* (Diagramas de Interação) – estes diagramas são usados para fazer a análise das interações entre os processos e recursos. São utilizados os Diagramas de Colaboração e Seqüência da UML.

Há dois estereótipos que não pertencem a nenhuma destas visões, isto é, estes estereótipos podem ser utilizados em todos os diagramas e modelos de todas as visões anteriormente apresentadas. A

Tabela 6.5 apresenta esses dois estereótipos.

Tabela 6.5 - Elementos Diversos.

ELEMENTOS	DESCRIÇÃO
 *Reference Note* (Nota de Referência)	Representa a referencia a outro diagrama ou documento.
 *Business Package* (Pacote de Negócio)	Representa o empacotamento modelos de negócios ou parte deles para uma melhor organização.

Fonte: Adaptação de Eriksson & Penker (2000).

6.2. A Adaptação

A adaptação dos elementos da extensão de negócios Eriksson-Penker foi realizada com a utilização da ferramenta *Borland® Together® Architect 2006 Service Pack 1 for Eclipse* (Borland, 2006), que é reconhecidas pela OMG e dá suporte a modelagem com UML 2.0 (OMG, 2006). Esta ferramenta foi escolhida devido à facilidade de criação de *profiles* da UML 2.0.

Para armazenar os estereótipos adaptados para a UML 2.0, será utilizado um *profile*. O *profile* é um mecanismo da UML 2.0 que agrupa um grupo de estereótipos que possuem um determinado objetivo (Eriksson et al., 2004) e possibilita o desenvolvimento de um modelo com UML 2.0 utilizando os estereótipos que ele agrupa.

Na organização do trabalho, os estereótipos foram agrupados em quatro pacotes (*packages)*, de acordo com suas aplicações, são eles: *Goal Extensions* (Extensões de Objetivos), *Miscellaneous Extensions* (Extensões Diversas), *Process Extensions* (Extensões de Processo) e *Resources and Rules Extensions* (Extensões de Recursos e Regras).

6.2.1. Criando *Profiles*

O *package Profile* contém mecanismos que permitem meta-classes de meta-modelos serem estendidos e adaptados a diferentes propósitos (Eriksson et al., 2004). Um

profile agrupa elementos que estendem de meta-classes de um meta-modelo. A relação entre o *package* do *profile* e o *package* do meta-modelo é representada por uma dependência com o estereótipo *import*, como é apresentado na Figura 6.1.

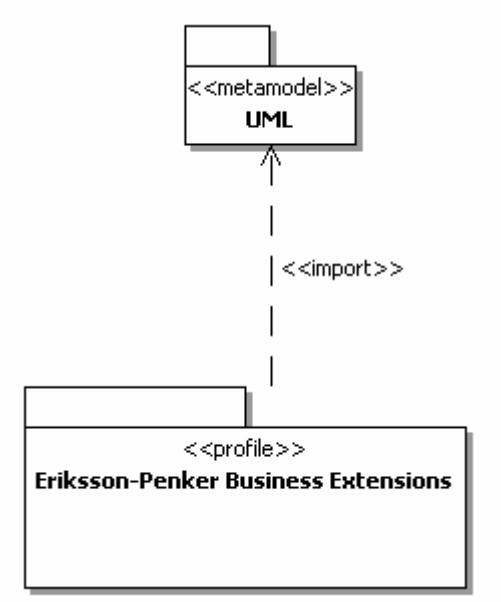


Figura 6.1 - Associação entre um *profile* e um metamodelo. Fonte: Adaptado de Eriksson et al. (2004).

Para criar um *profile* na ferramenta *Borland® Together®* é necessário criar um novo projeto do tipo "*Profile Definition Project*". Além da criação de um *profile*, a ferramenta *Borland® Together®* permite a geração de um *plug-in* deste *profile* para poder fazer a distribuição do mesmo.

No presente trabalho, os elementos foram divididos em quatro *packages*, dentro do *profile* criado. Os packages são: *Goal Extensions*, *Miscellaneous Extensions*, *Process Extensions* e *Resources and Rules Extensions*. Cada elemento foi alocado em cada pacote, de acordo com as visões onde eles aparecem, como foi mostrado na seção 6.1.

A ferramenta *Borland® Together®* permite a criação de uma barra de ferramentas para incluir os elementos criados na ferramenta. Para utilizar este recurso é necessário criar uma classe com o estereótipo *paletteContribution* que permite escolher na propriedade *diagrams*, em que diagramas disponíveis esta barra de ferramentas irá aparecer.

Para incluir os elementos criados na barra de ferramentas definida anteriormente é necessário fazer o relacionamento entre a classe *paletteContribution* e o elemento criado, que será um *stereotype*, com uma associação do tipo *contribution.*

6.2.2. Goal Extensions

Os estereótipos agrupados neste *package* são os utilizados na visão de negócios (*Business Vision)*. A seguir será descrito como foi a adaptação de cada elemento para a UML 2.0.

- ***Goal* (Objetivo) -** O elemento *goal* representa um objetivo que a organização está empenhada em atingir. No método Eriksson-Penker o elemento *goal* é estereotipado como uma classe (*class*). A adaptação para o UML 2.0 deste elemento faz a extensão de um elemento *stereotype* através do relacionamento *extension* da meta-classe *Class,*. que está no *package UML.Classes.Kernel* do *infrastructure* da UML 2.0.

  Para criar o estereótipo *goal* na ferramenta *Borland® Together®*, é necessário estender a classe *stereotype* da meta-classe *Class. A meta-classe Class* encontra-se no *package uml 20.classes*. A extensão do elemento *Goal* é apresentada na Figura 6.2.

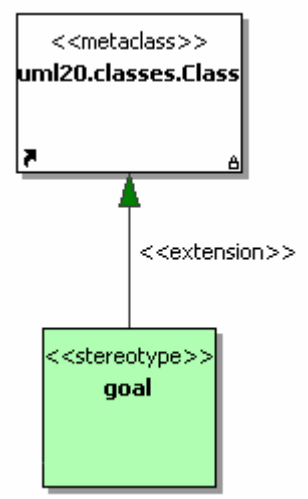


Figura 6.2 - Extensão do elemento *goal*. Fonte: Elaborado pelo autor.

- ***Problem* (Problema) -** O elemento *problem* representa um problema que pode dificultar ou impedir que um objetivo da empresa seja atingido. No método Eriksson-Penker, este elemento é estereotipado como uma nota (*note*). A adaptação para o UML 2.0 faz a extensão de um elemento *stereotype* através do

relacionamento *extension* da meta-classe *Comment*, que está no *package uml.Classes.Kernel* do *infrastructure* da UML 2.0.

Para criar o estereótipo *problem* na ferramenta *Borland® Together®* é necessário estender a classe stereotype da meta-classe *Note*. A meta-classe *Note* encontr-se no package *uml20.together*. A extensão do elemento goal é apresentado na Figura 6.3. O elemento *Note* definido pelo *Borland® Together®* é uma extensão da meta-classe *Comment* e é utilizado para realizar comentários (como o elemento *Note* do UML).

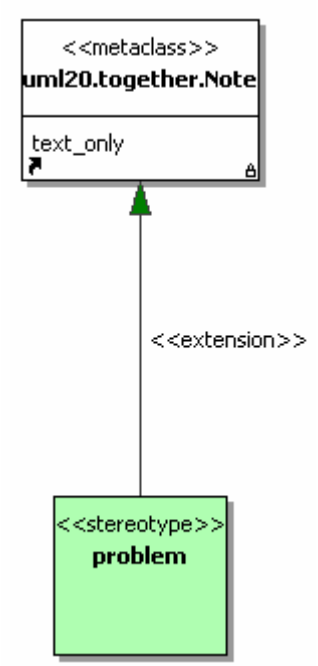


Figura 6.3 - Extensão do elemento *problem*. Fonte: Elaborado pelo autor.

- ***Contradictory Goal* (Objetivo Contraditório) -** O elemento *contradictory goal* representa uma relação entre objetivos e mostra que os objetivos podem ser contraditórios.

No método Eriksson-Penker, o elemento contem é estereotipado como uma associação (*association*). A adaptação para o UML 2.0 deste elemento consiste em fazer com que um elemento *stereotype* estenda, através do relacionamento *extension*, da meta-classse *Association*, que está no *package uml.Classes.Kernel* do *infrastructure* da UML 2.0.

Para criar este estereótipo na ferramenta *Borland® Together®*, a classe *stereotype* deve estender a meta-classe *Association* que se encontra no *package uml.kernel*. Esta extensão pode ser visualizada na Figura 6.4.

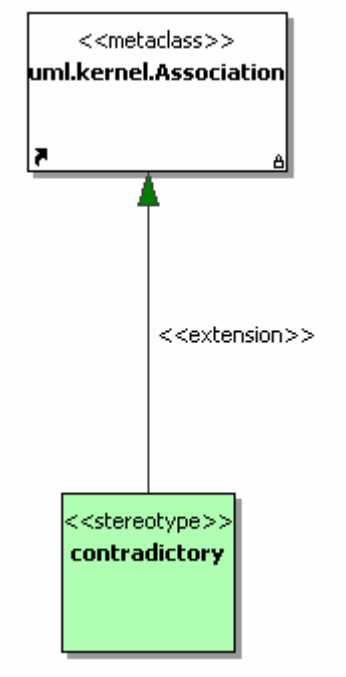


Figura 6.4 - Extensão do elemento *contradictory*. Fonte: Elaborado pelo autor.

- ***Incomplete Goal Decomposition* (Decomposição Incompleta do Objetivo) -** O elemento *incomplete* representa a não decomposição do objetivo, isto é, não foram atingidos os sub-objetivos em que o objetivo foi decomposto e não foi completado o objetivo.

  No método Eriksson-Penker o elemento *incomplete goal decomposition* é estereotipado como uma dependência (*dependency*). Sendo assim, sua adaptação para a UML 2.0 consiste em fazer com que um elemento *stereotype* estenda, através do relacionamento *extension*, da meta-classse *Dependency*, que está no *package UML.Classes.Dependencies* do *infrastructure* da UML 2.0.

  Para criar o estereótipo *incomplete goal decomposition* na ferramenta *Borland® Together®*, a classe *stereotype* deve estender a meta-classe *Dependency* que se encontra no package *uml20.classes*. A Figura 6.5 apresenta a extensão do elemento *incomplete goal decomposition*.

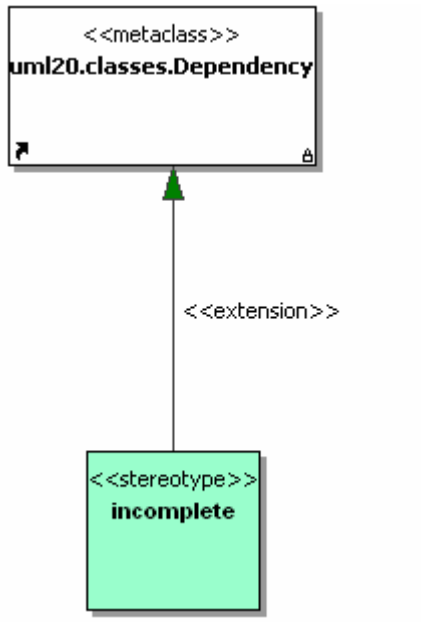


Figura 6.5 - Extensão do elemento *incomplete goal decomposition*. Fonte: Elaborado pelo autor.

- ***Complete Goal Decomposition* (Decomposição Completa do Objetivo) -** O elemento *complete goal decomposition* representa a decomposição completa do objetivo, isto é, se foram atingidos todos os sub-objetivos no qual o objetivo foi decomposto, terá sido completado o objetivo como um todo.

  No método Eriksson-Penker, este elemento é estereotipado como uma dependência (*dependency*). Sendo assim, sua adaptação para a UML 2.0 consiste em fazer com que um elemento *stereotype* estenda, através do relacionamento *extension*, da meta-classse *Dependency*, que está no *package UML.Classes.Dependencies* do *infrastructure* da UML 2.0.

  Para criar o estereótipo *complete goal decomposition* na ferramenta *Borland® Together®*, a classe *stereotype* deve estender a meta-classe *Dependency* que se encontra no package *uml20.classes*. A Figura 6.6 apresenta a extensão do elemento *complete goal decomposition*.

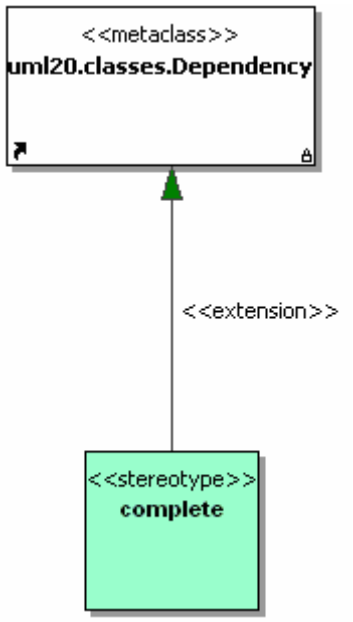


Figura 6.6 - Extensão do elemento *complete goal decomposition*. Fonte: Elaborado pelo autor.

6.2.3. Process Extensions

No *package process extensions* os estereótipos agrupados são utilizados na visão *Business Process*.

A adaptação de cada elemento para a UML 2.0 foi realizada da seguinte maneira:

- ***Process* (Processo) -** O elemento *process* representa um processo, isto é, um conjunto de atividades que são executadas no negócio.

  No método Eriksson-Penker o elemento *process* é estereotipado pelo elemento atividade (*activity*). Para adapta-lo a UML 2.0 o elemento activity será estendido do elemento *Action* do *package UML.Activities.Action* da *infrastructure* da UML 2.0.

  Para criar o estereótipo activity na ferramenta *Borland® Together®*, a classe stereotype estende-se a meta-classe *Action* que se encontra no *package uml20.activities*. A extensão do elemento *process* é apresentada na Figura 6.7. A adaptação do elemento será para o tipo *Action* ao invés do *Activity*, conforme definido para adaptação para a UML 2.0 devido a restrições da ferramenta.

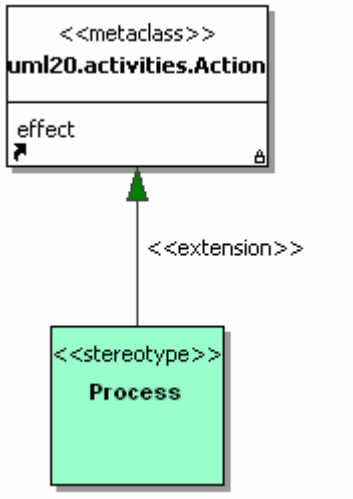


Figura 6.7 - Extensão do elemento *process*. Fonte: Elaborado pelo autor.

- ***Activity* (Atividade) -** O elemento *activity* representa uma atividade, isto é, uma atividade atômica de um processo.

No método Eriksson-Penker, o elemento *activity* é estereotipado pelo elemento atividade (*activity*). Para adapta-lo a UML 2.0 o elemento activity será estendido do elemento *Action* do *package UML.Activities.Action* da *infrastructure* da UML 2.0 que representa uma ação individual e torna coerente a relação à definição de atividade mostrada no método Eriksson-Penker.

Para criar o estereótipo activity na ferramenta *Borland® Together®*, a classe stereotype estende-se a meta-classe *Action* que se encontra no *package uml20.activities*. A extensão do elemento *activity* é apresentada na Figura 6.8.

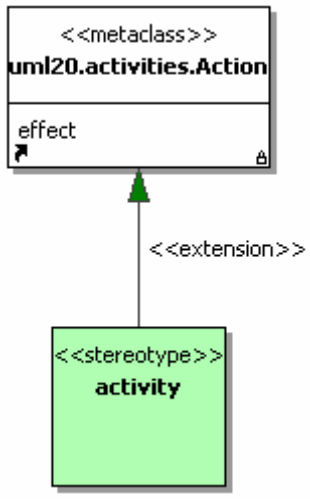


Figura 6.8 - Extensão do elemento *activity*. Fonte: Elaborado pelo autor.

- ***Noncausal Resource Flow* (Fluxo de Recurso não obrigatório) -** O elemento *noncausal resource flow* representa a relação entre um recurso e um processo e indica que o recurso pode ser utilizado pelo processo.

  No método Eriksson-Penker o elemento *noncausal resource flow* é estereotipado como um fluxo de objeto (*object flow*). A adaptação para UML 2.0 deste elemento faz com que um elemento *stereotype* estenda através do relacionamento *extension* da meta-classse *ObjectFlow* que está no *package UML.Activities.CompleteActivities* do *infrastructure* da UML 2.0.

  Para criar o estereótipo *noncausal resource flow* na ferramenta *Borland® Together®*, a classe *stereotype* estende a meta-classe *Dependency* que encontra-se no *package uml20.classes* e pode ser visualizada na Figura 6.9. Os relacionamentos entre os elementos de recursos serão adaptados para o tipo *Dependency* ao invés do *ObjectFlow*, conforme definido para adaptação para a UML 2.0 devido a restrições da ferramenta.

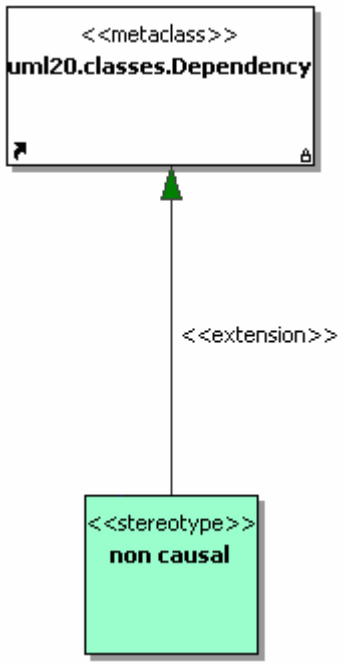


Figura 6.9 - Extensão do elemento *non causal resource flow*. Fonte: Elaborado pelo autor.

- ***Process Control* (Controle de Processo) -** O elemento *process control* representa um relacionamento de um recurso e um processo e indica que o processo é controlado por este recurso.

  Como acontece com o elemento *noncausal resource flow* o elemento *process control* é estereotipado como um fluxo de objeto (*object flow*). Sendo assim, sua adaptação para o UML 2.0 consiste em fazer com que um elemento *stereotype* estenda através do relacionamento *extension*, da meta-classse *ObjectFlow*, que está no *package UML.Activities.CompleteActivities* do *infrastructure* da UML 2.0.

  Para criar o estereótipo *process control* na ferramenta *Borland® Together®*, a classe *stereotype* deve estender a meta-classe *Dependency* que se encontra no package *uml20.classes*. A Figura 6.10 apresenta a extensão do elemento *process control*.

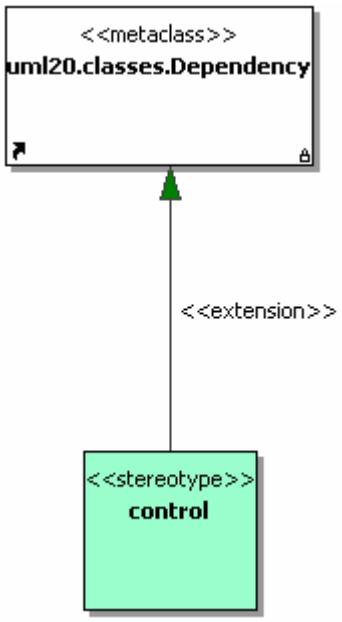


Figura 6.10 - Extensão do elemento *process control*. Fonte: Elaborado pelo autor.

- ***Goal Connection* (Conexão do Objetivo) -** O elemento *goal connection* representa a relação entre um objetivo e um processo e indica que um processo pretende atingir um objetivo.

  No método Eriksson-Penker o elemento *goal connection* é estereotipado como uma dependência (*dependency*). Sendo assim, sua adaptação para a UML 2.0 consiste em fazer com que um elemento *stereotype* estenda, através do

relacionamento *extension*, da meta-classse *Dependency*, que está no *package UML.Classes.Dependencies* do *infrastructure* da UML 2.0.

Para criar o estereótipo *goal connection* na ferramenta *Borland® Together®* é necessário que a classe *stereotype* estendea a meta-classe *Dependency* que encontra-se no package *uml20.classes*. A Figura 6.11 apresenta a visão desta extensão.

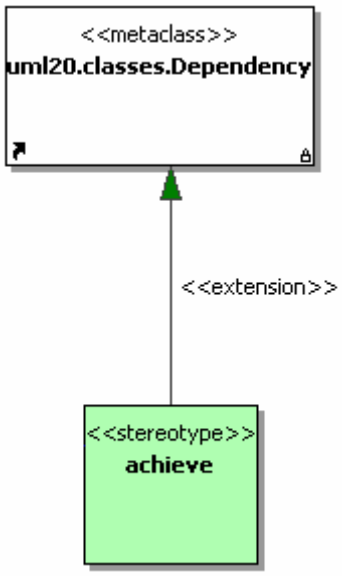


Figura 6.11 - Extensão do elemento *Goal Connection*. Fonte: Elaborado pelo autor.

- ***Process Supply* (Fornecedor de Processo) -** O elemento *process supply* representa um relacionamento entre um recurso e um processo e indica que o recurso fornece algo ao processo.

Como acontece com os elementos *noncausal resource flow* e *elemnet process control* o elemento *process supply* é estereotipado como um fluxo de objeto (*object flow*). Sendo assim, sua adaptação para UML 2.0 consiste em fazer com que um elemento *stereotype* estenda através do relacionamento *extension*, da meta-classse *ObjectFlow*, que está no *package UML.Activities.CompleteActivities* do *infrastructure* da UML 2.0.

Para criar o estereótipo process supply na ferramenta *Borland® Together®* a classe *stereotype* estende a meta-classe *Dependency* que encontra-se no package *uml20.classes*. Esta extensão pode ser visualizada na Figura 6.12.

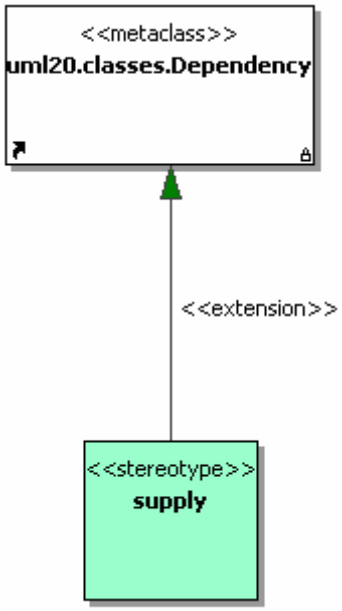


Figura 6.12 - Extensão do elemento p*rocess supply*. Fonte: Elaborado pelo autor.

- ***Assembly Line* (Linha de Montagem)**

O elemento *assembly line* representa a relação entre os processos e os objetos que eles utilizam.

No método Eriksson-Penker o elemento *assembly line* é estereotipado como um pacote (*package*). A sua adaptação para o UML 2.0 consiste em fazer a extensão elemento *stereotype* estenda, através do relacionamento *extension*, da meta-classse *Package*, que está no *package UML.Classes.Kernel* do *infrastructure* da UML 2.0.

Para criar este estereótipo na ferramenta *Borland® Together®*, a classe *stereotype* deve estender a meta-classe *Package* que se encontra no package *uml.kernel.packages*. Esta extensão pode ser visualizada na Figura 6.13.

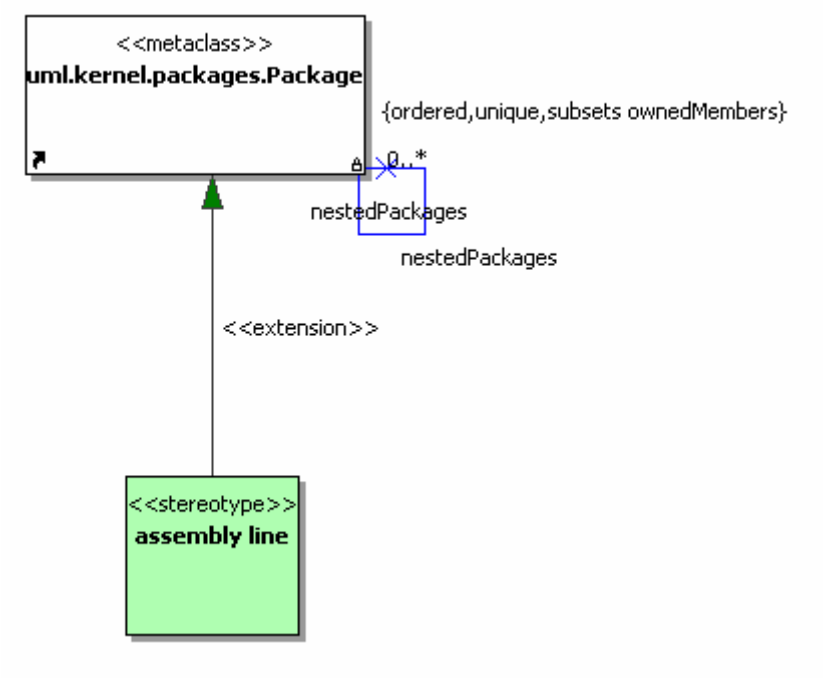


Figura 6.13 - Extensão do elemento *assembly line*. Fonte: Elaborado pelo autor.

- ***Writed Object* (Objeto Escrito) -** O elemento *writed object* representa um objeto escrito por um processo no objeto *assembly line*.

  No método Eriksson-Penker o elemento *writed object* é estereotipado como um objeto (*object*). Sendo assim, sua adaptação para o UML 2.0 consiste em fazer com que um elemento *stereotype* estenda, através do relacionamento *extension*, da meta-classse *InstanceSpecification*, que está no *package UML.Classes.Kernel* do *infrastructure* da UML 2.0.

  Para criar o estereótipo *writed object* na ferramenta *Borland® Together®* a classe *stereotype* deve estender a meta-classe *InstanceSpecification* que se encontra no package *uml20.kernel.instances*. A Figura 6.14 apresenta a extensão do estereótipo *writed object*.

  A representação da imagem do elemento visualizada no diagrama é inserida em um arquivo de imagem na propriedade *viewmap* da classe *stereotype*.

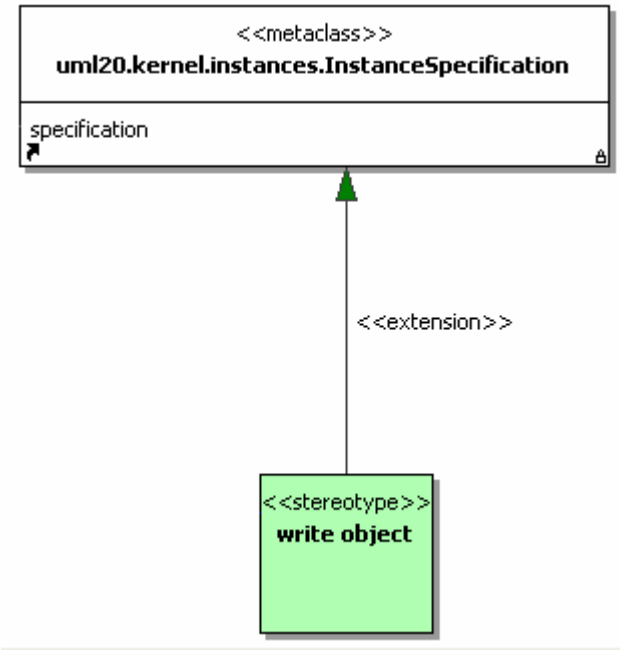


Figura 6.14 - Extensão do elemento *write object*. Fonte: Elaborado pelo autor.

- ***Read Object* (Objeto Lido) -** O elemento *read object* representa um objeto que é lido por um processo no objeto *assembly line*.

  Como acontece com o *writed object*, este elemento é estereotipado como um objeto (*object*) no método Eriksson-Penker. Sendo assim, sua adaptação para o UML 2.0 consiste em fazer com que um elemento *stereotype* estenda através do relacionamento *extension* da meta-classse *InstanceSpecification*, que está no *package UML.Classes.Kernel* do *infrastructure* da UML 2.0.

  Para criar o estereótipo *read object* na ferramenta *Borland® Together®*, a classe *stereotype* deve estender a meta-classe *InstanceSpecification* que se encontra no package *uml20.kernel.instances*. A Figura 6.15 apresenta a extensão do estereótipo *read object*.

  A representação da imagem do elemento será visualizada no diagrama inserido em um arquivo de imagem na propriedade *viewmap* da classe *stereotype.*

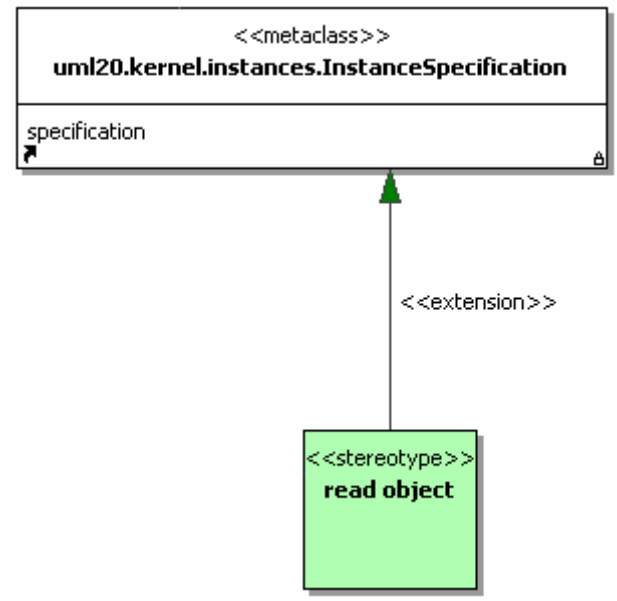


Figura 6.15 - Extensão do elemento *read object*. Fonte: Elaborado pelo autor.

6.2.4. Resourses and Rules Extensions

Os estereótipos agrupados no *package resources and rules extensions*são os utilizados na visão *Business Structure*.

A adaptação de cada elemento para a UML 2.0 foi realizada da seguinte maneira:

- ***Resource* (Recurso)** - O elemento *resource* representa um recurso do negócio. No método Eriksson-Penker o elemento *resource* é estereotipado como um objeto (*object*). Sendo assim, sua adaptação para UML 2.0 consiste em fazer com que um elemento *stereotype* estenda através do relacionamento *extension*, da meta-classse *InstanceSpecification* que está no *package UML.Classes.Kernel* do *infrastructure* da UML 2.0.

  Para criar o estereótipo *resource* na ferramenta *Borland® Together®*, a classe *stereotype* deve estender a meta-classe *InstanceSpecification* que se encontra no package *uml20.kernel.instances*. A Figura 6.16 apresenta a extensão do estereótipo *resource.*

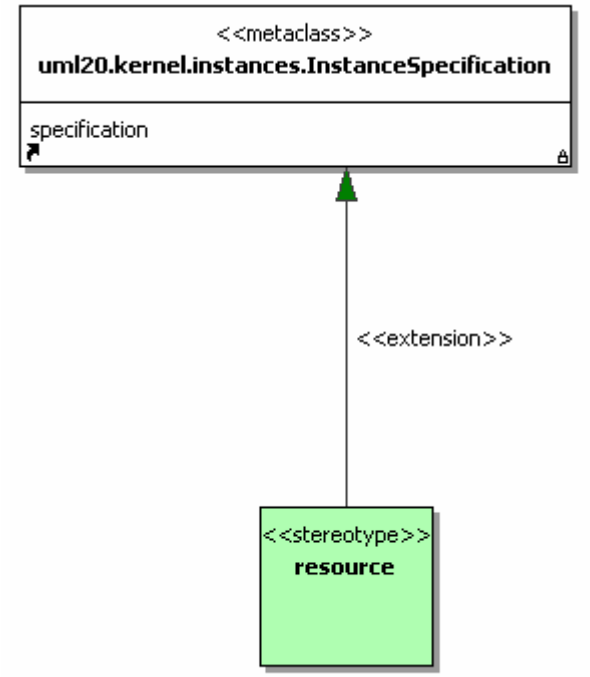


Figura 6.16 - Extensão do elemento *resource*. Fonte: Elaborado pelo autor.

- ***Information*** **(Informação)** - O elemento *information* representa uma informação no negócio, mas que também é considerada um recurso. No método Eriksson-Penker, o elemento *information* é estereotipado como um objeto (*object*). Sendo assim, sua adaptação para o UML 2.0 consiste em fazer com que um elemento *stereotype* estenda, através do relacionamento *extension*, da meta-classse *InstanceSpecification*, que está no *package UML.Classes.Kernel* do *infrastructure* da UML 2.0.

  Para criar este estereótipo na ferramenta *Borland® Together®*, a classe *stereotype* deve estender a meta-classe *InstanceSpecification* que se encontra no *package uml20.kernel.instances*. A Figura 6.17 apresenta a extensão do elemento *information*.

  Para representar a imagem do elemento que será visualizada no diagrama, foi inserido um arquivo de imagem na propriedade *viewmap* da classe *stereotype*.

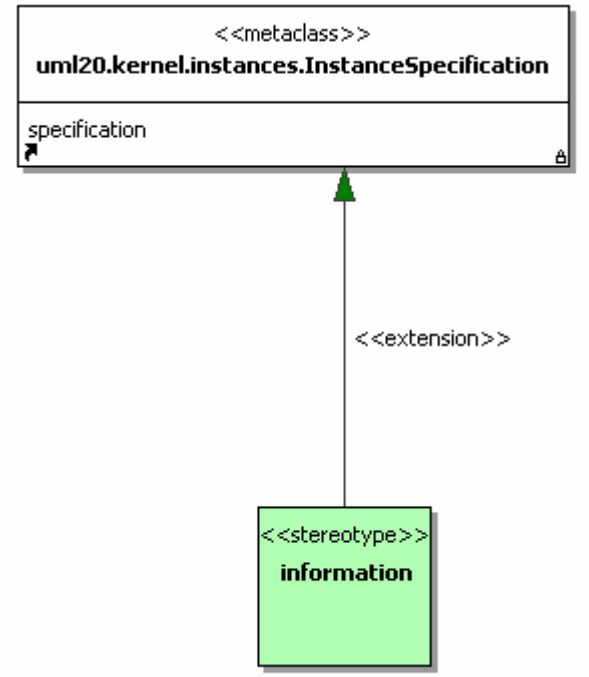


Figura 6.17 - Extensão do elemento *information*. Fonte: Elaborado pelo autor.

- ***Abstract Resource* (Recurso abstrato)** - O elemento *abstract resource* representa um recurso abstrato do negócio,ou seja, intangível. No método Eriksson-Penker o elemento *abstract resource* é estereotipado como um objeto (*object*). Dessa forma, sua adaptação para UML 2.0 consiste em fazer com que um elemento *stereotype* estenda, através do relacionamento *extension*, da meta-classe *InstanceSpecification*, que está no *package UML.Classes.Kernel* do *infrastructure* da UML 2.0.

  Para criar o estereótipo *abstract resourse* na ferramenta *Borland® Together®* a classe *stereotype será* estendedida a meta-classe *InstanceSpecification* que se encontra no package *uml20.kernel.instances*. A Figura 6.18 apresenta a visualização da extensão do estereótipo *abstract resource*.

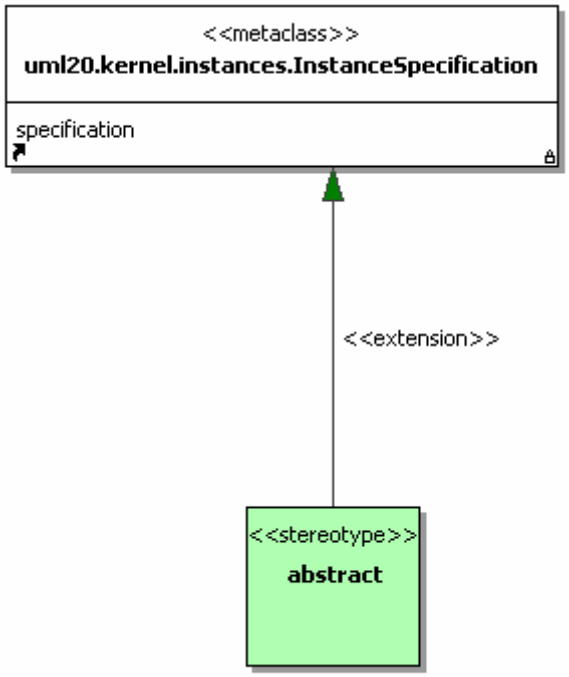


Figura 6.18 - Extensão do elemento *abstract resource*. Fonte: Elaborado pelo autor.

- ***People* (Pessoas)** - O elemento *abstract resource* representa uma pessoa ou um grupo de pessoas envolvidas no negócio. No método Eriksson-Penker **o** elemento people é estereotipado como um objeto (*object*). Sendo assim, sua adaptação para o UML 2.0 consiste em fazer com que um elemento *stereotype* estenda, através do relacionamento *extension*, da meta-classe *InstanceSpecification*, que está no *package UML.Classes.Kernel* do *infrastructure* da UML 2.0.

  Para criar o estereótipo *people* na ferramenta *Borland® Together®*, a classe *stereotype* deve estender a meta-classe *InstanceSpecification* que se encontra no *package uml20.kernel.instances*. A Figura 6.19 apresenta a extensão do estereótipo *people*.

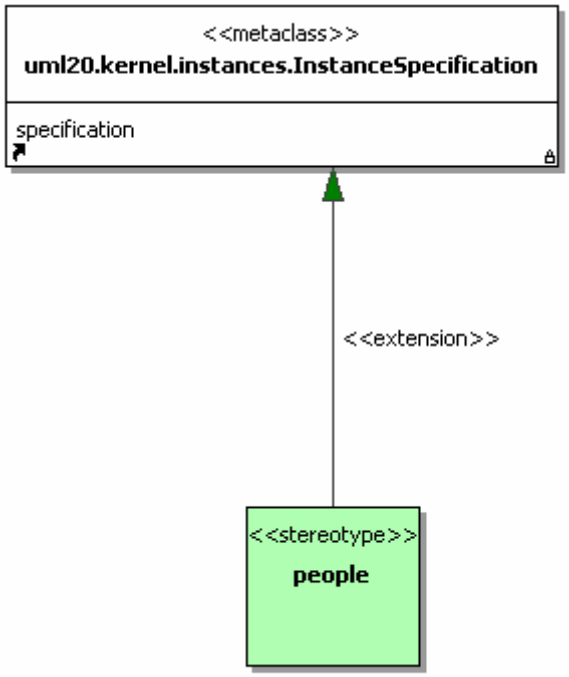


Figura 6.19 - Extensão do elemento *people*. Fonte: Elaborado pelo autor.

- ***Physical Resource*** **(Recurso Físico)** - O elemento ***p****hysical* ***r****esource* representa uma pessoa ou um grupo de pessoas envolvidas no negócio. No método Eriksson-Penker este elemento é estereotipado como um objeto (*object*). Sendo assim, sua adaptação para o UML 2.0 consiste em fazer com que um elemento *stereotype* estenda, através do relacionamento *extension*, da meta-classe *InstanceSpecification*, que está no *package UML.Classes.Kernel* do *infrastructure* da UML 2.0.

  Para criar o estereótipo *physical resource* na ferramenta *Borland® Together®*, a classe *stereotype* estende a meta-classe *InstanceSpecification* que encontra-se no package *uml20.kernel.instances*. A Figura 6.17 apresenta a extensão do estereótipo *physical resource*

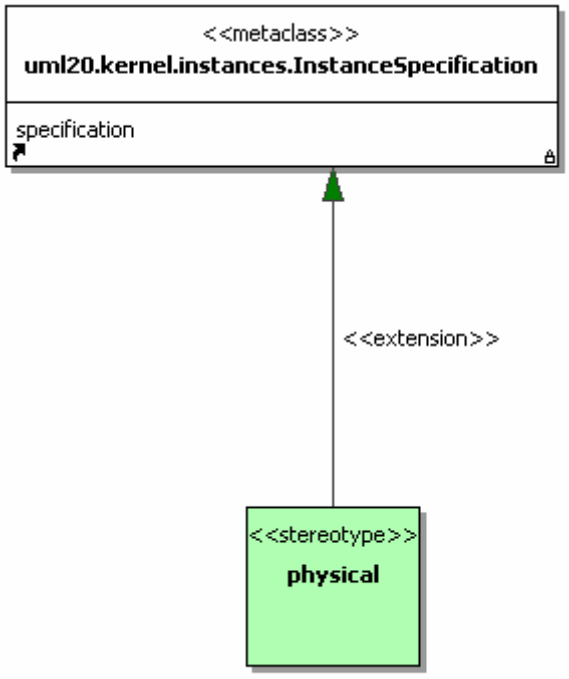


Figura 6.20 - Extensão do elemento *physical resource*. Fonte: Elaborado pelo autor.

- ***Business Event* (Evento de Negócios) -** O elemento *business event* representa um evento no espaço e no tempo que apresenta um impacto no negócio. No método Eriksson-Penker o elemento *business event* é estereotipado como um sinal (*signal*). Sendo assim, sua adaptação para o UML 2.0 consiste em fazer com que um elemento *stereotype* estenda, através do relacionamento *extension*, da meta-classe *InstanceSpecification*, que está no *package UML.CommonBehaviors.Communications* do *infrastructure* da UML 2.0.

  A meta-classe *Signal* não permite que ela seja estendida mas somente implementada. Portanto, para criar o estereótipo *business event* na ferramenta *Borland® Together®* foi feito com que a classe *stereotype* estendesse a meta-classe *NamedElement*, que representa um elemento e que no modelo pode ter um nome,e encontra-se no package *uml.kernel*, em seguida foi feito com que a classe *stereotype* implementasse pela associação *Generalization*, a meta-classe *Signal*, que encontra-se no *package uml2.0*. A Figura 6.21 apresenta a extensão do estereótipo *business event*.

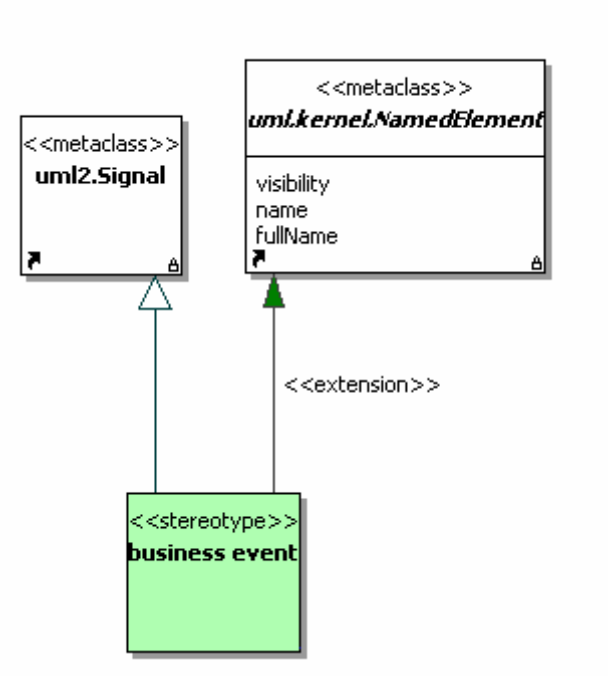


Figura 6.21 - Extensão do elemento *business event*. Fonte: Elaborado pelo autor.

- ***Business Rule* (Regra de Negócios)** - O elemento *business rule* representa uma regra aplicada ao negócio que pode restringir ou controlar a sua execução No método Eriksson-Penker o elemento *business rule* é estereotipado como uma nota (*note*). A adaptação para o UML 2.0 deste elemento consiste em fazer com que um elemento *stereotype seja* estendido, através do relacionamento *extension* da meta-classe *Comment* que está no *package uml.Classes.Kernel* do *infrastructure* da UML 2.0.

  Para a criar o estereótipo *business rule* na ferramenta *Borland® Together®*, a classe *stereotype* deve estender a meta-classe *Note* que se encontra no package *uml20.together*. A extensão do elemento *business rule* é apresentado na Figura 6.22.

  O elemento *Note* definido pelo *Borland® Together®* é uma extensão da meta-classe *Comment* e é utilizado para realizar comentários (como o elemento *Note* do UML).

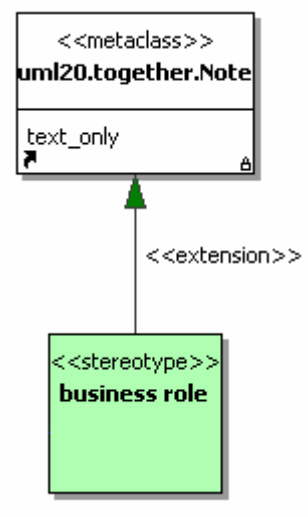


Figura 6.22 - Extensão do elemento *business role*. Fonte: Elaborado pelo autor.

6.2.5. *Miscellaneous Extensions*

Os estereótipos agrupados no *package miscellaneous extensions* não pertencem a uma visão especifica, mas podem aparecer em diversos diagramas e modelos.

A adaptação de cada elemento para a UML 2.0 foi realizada da seguinte maneira:

- ***Reference Note*** **(Nota de Referência)** - O elemento *reference note* representa uma referência para outro modelo ou diagrama. No método Eriksson-Penker **o** elemento *reference note* é estereotipado como uma nota (*note*). A adaptação para o UML 2.0 deste elemento consiste em fazer com que um elemento *stereotype* estenda, através do relacionamento *extension*, da meta-classe *Comment* que está no *package uml.Classes.Kernel* do *infrastructure* da UML 2.0.

  Para a criação deste estereótipo na ferramenta *Borland® Together®*, a classe *stereotype* deve estender a meta-classe *Note* que se encontra no package *uml20.together*. A Figura 6.23 apresenta a extensão do elemento *reference note*.

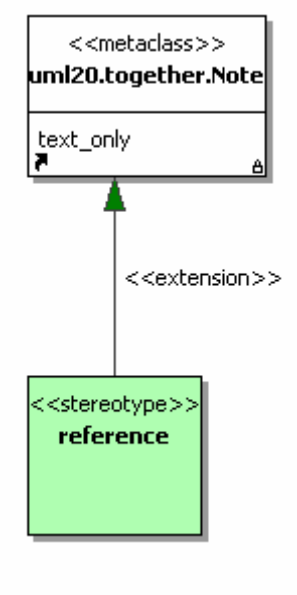


Figura 6.23 - Extensão do elemento *reference note*. Fonte: Elaborado pelo autor.

- ***Business Package* (Pacote de Negócio)** - O elemento *business package* representa um container no qual é usado para armazenar modelos e diagramas de modo que a modelagem total fique mais organizada. No método Eriksson-Penker **o** elemento*business package* é estereotipado como um pacote (*package*). A adaptação para o UML 2.0 deste elemento consiste em fazer com que um elemento *stereotype* estenda através do relacionamento *extension* da meta-classse *Package*, que está no *package UML.Classes.Kernel* do *infrastructure* da UML 2.0.

  Para criar o estereótipo *business package* na ferramenta *Borland® Together®* a classe *stereotype* estende a meta-classe *Package* que encontra-se no package *uml.kernel.packages*. A Figura 6.21 apresenta a extensão do elemento *business package*.

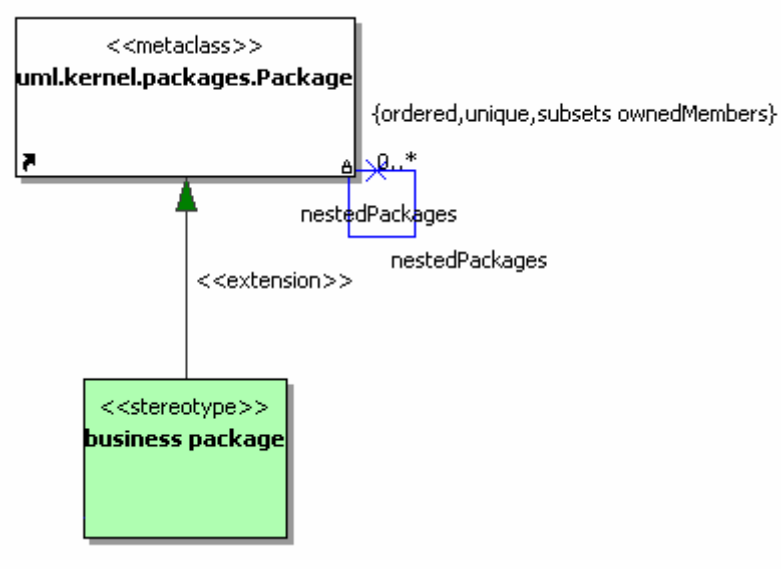


Figura 6.24 - Extensão do elemento *business package*. Fonte: Elaborado pelo autor.

6.2.6. Utilização do *Profile*

Para utilizar um *profile* é necessário aplicá-lo no modelo que está sendo desenvolvido. O uso de um *profile* por um modelo é representado por uma dependência com o estereótipo de *apply* indicando que o *profile* foi aplicado ao modelo. A Figura 6.25 apresenta a representação de um *profile* (*Eriksson-Penker Business Extensions*) que utiliza o metamodelo UML, sendo aplicado em uma modelagem (Modelagem de Negócios) . Isto possibilita a utilização, no modelo, dos estereótipos e outros elementos contidos no *profile*.

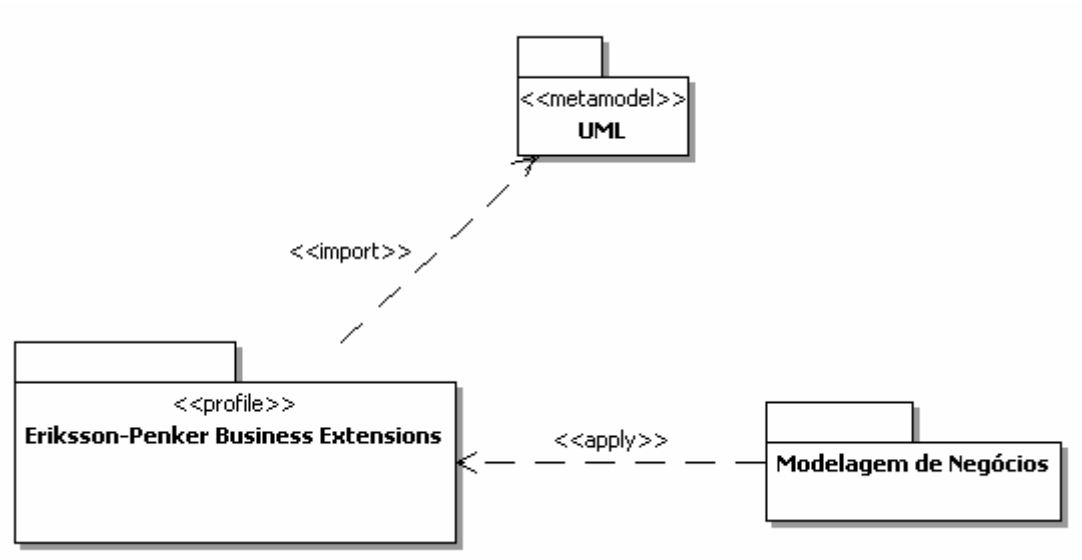


Figura 6.25 - Aplicação de um *profile* a um modelo. Fonte: Adaptado de Eriksson et al. (2004).

O uso do *profile* criado, na ferramenta *Borland® Together®*, é permitido após sua compilação gerando o *plug-in* através do comando "*Profile Deploy...*", que encontra-se no menu de comandos "*Model/Profile*" da ferramenta *Borland® Together®*.

A ferramenta executará a compilação do *plug-in* instalando-o na ferramenta *Borland® Together®* bastando somente fecha - lá e abri-lá novamente. O resultado da geração do *plug-in* é a criação de um diretório com o nome que foi configurado quando gerado. Este diretório encontra-se no "*Architect2006/eclipse/plugins*", que fica no diretório de instalação da ferramenta.

Para efetuar a distribuição deste *plug-in* é necessário disponibilizar o diretório criado. E para instalar o *plug-in* em outro computador é necessário copiá-lo para o diretório "*Architect2006/eclipse/plugins*", que fica no diretório de instalação da ferramenta.

6.3. Considerações Finais

Neste capítulo foi apresentada a adaptação do método Eriksson-Penker para UML 2.0 através da criação de um *profile* que agrupou os elementos criados. Para auxiliar a adaptação foi utilizada a ferramenta *Borland® Together®* que oferece um mecanismo de visual para a criação de *profiles* e de distribuição do *profile* através de *plug-in*.

Para a validação desta adaptação foi realizado um estudo de caso em uma Fábrica de Software e que será apresentado no Capítulo 7.

7. ESTUDO DE CASO – MODELAGEM DE NEGÓCIO À FÁBRICA DE SOFTWARE

O estudo de caso foi realizado em uma fábrica de software em que foi verificada a necessidade da realização da modelagem de negócios, uma vez observado que a fábrica não tinha bem definido os seus objetivos.

Conforme Siy et al. (2001), uma fábrica de software é uma organização que provê serviços de desenvolvimento de sistemas com alta qualidade, a baixo custo e de forma rápida, utilizando um processo de desenvolvimento de software bem definido e tecnologia de ponta, além de algumas formas de *feedback* para reconhecer e lidar com oportunidades de melhoria do processo.

Neste capítulo será apresentada a Fábrica de Software e os procedimentos realizados e resultados obtidos na modelagem de negócio do estudo de caso.

7.1. Apresentação da Fábrica de Software

A Fábrica de Software em que foi realizado o estudo de caso está localizada no sul do estado de Minas Gerais e é uma das áreas de uma empresa de pequeno porte, que possui, ainda as áreas de ensino à distância e de consultoria.

Do quadro atual de funcionários e estagiários desta empresa, a Fábrica de Software consta atualmente com cinco profissionais formados em ciência da computação e doze cursando a graduação em ciência da computação.

A Fábrica de Software possui mais de três anos de atuação, mesmo diante das diversas mudanças estruturais na empresa que ela pertence. Atualmente ela possui projetos e produtos nas áreas de gerência de documentos fiscais, gestão imobiliária, administração escolar, automação comercial, customização de ferramentas para ensino à distância e controle cafeeiro.

7.2. Modelagem de Visão *Business Vision*

Na visão *Business Vision* foi realizado a definição de objetivos e conceitos mais importantes para a Fábrica de Software, e a modelagem destes conceitos e objetivos.

7.2.1. Definições dos Princípios

Para definir os princípios da empresa, foi utilizado o formulário apresentado na seção 4.4.1. Para a definição foi realizado reuniões com pessoas chaves na Fábrica de Software.

Os princípios da Fábrica de Software não estavam claros, o que acarretou em um período de tempo razoável para sua definição e, muita discussão entre os participantes da reunião.

Como resultado da reunião obteve-se a definição dos princípios da Fábrica de Software, que é apresentada na Tabela 7.1.

Tabela 7.1 - Princípios da Fábrica de Software

1. Objetivos: Criar produtos e soluções de qualidade para atendimento de um nicho específico do mercado, consolidando um *know-how* em uma área especifica. Especialização da equipe através de treinamentos e palestras sobre as diversas áreas da computação, especialmente: gerência de projetos, melhoria de processo, testes, programação e novas tecnologias.
2. Pontos Fortes: • Mão de obra com potencial de especialização; • Mão de obra com um alto valor custo/benefício; • Localização Geográfica; • Treinamento especializado de alguns membros na área de engenharia de processo e gerência de projetos
3. Pontos Fracos: • A maior parte dos colaboradores serem estagiários com disponibilidade média de 20 horas semanais; • Baixo nível de realização de cursos externos, principalmente em novas tecnologias; • Falta de pessoas qualificadas nas áreas: financeira, marketing, RH e administrativa.

Tabela 7.1 - Princípios da Fábrica de Software (Continuação)

4. Oportunidades: • Produtos voltados para as necessidades da universidade parceira; • Exploração do mercado de sistemas acadêmicos; • Exploração do mercado de sistemas de gestão de construtoras; • Exploração do mercado de sistemas de GED/Workflow (Gerência Eletrônica de Documentos); • Projetos em prospecção junto com a empresa parceira.
5. Ameaças: • Após a conclusão do curso de graduação dos estagiários ocorrer impossibilidade de mantê-los na equipe devido ao baixo valor de mercado da região, comparado às cidades maiores e capitais; • Falta de planejamento e metas a curto, médio e longo prazo; • Instabilidade no relacionamento da Fábrica de Software, com as duas outras áreas de negócio da empresa; • Instabilidade no relacionamento entre os sócios.
6. Fatores Críticos: • Treinamento para toda a equipe; • Motivação da equipe; • Manutenção dos Estagiários que concluírem seus cursos de graduação.
7. Estratégias: • Investimento em treinamentos e certificações para os colaboradores; • Contratação de alguns funcionários qualificados em administração de empresas; • Divulgação de produtos já existentes na empresa.

Tabela 7.1 - Princípios da Fábrica de Software (Continuação)

8. Competências Principais: • Projetos de GED/Workflow para soluções fiscais; • Projetos de Gestão de Imobiliárias; • Projetos de Sistemas de Automação Comercial; • Projetos de Sistemas de Administração Escolar.
9. Papéis: Por ser uma Fábrica de Software de pequeno porte, algumas pessoas possuem diversos papéis. Foram identificados três tipos colaboradores na fábrica: • *Colaborador 1:* Gerente de Infra-Estrutura; Engenheiro de Software. • *Colaborador 2:* Gerente da Fábrica; Analista de Tecnologia / Processo / Arquitetura; Engenheiro de Software; Líder de Projetos. • *Colaborador 3:* Gerente da Fábrica; Gerente de Projetos; Analista de Sistemas / Processo; Engenheiro de Software. Os outros colaboradores se encaixam nos seguintes perfis, sendo que algumas vezes, ocorre sobreposição de papéis. • Programador; • Engenheiro de Software; • Engenheiro de Testes; • Líder de Projetos.
10. Unidades Organizacionais: Atualmente a Fábrica de Software não está dividida em unidades organizacionais.

Fonte: Fábrica de Software.

7.2.2. Visão Descritiva

Para a definição do mesmo, foram utilizadas como base, as perguntas que foram apresentadas na seção 4.4.1. Para formular as respostas destas perguntas, foi feita uma outra reunião com as duas pessoas chaves que participaram da reunião anterior.

O resultado obtido desta reunião foi a formulação do plano estratégico apresentado na Tabela 7.2.

Tabela 7.2 - Visão Descritiva da Fábrica de Software.

Clientes:

- Empresas do ramo imobiliário, com soluções para a gestão das atividades comerciais, financeiras e contábeis;
- Empresas do ramo de telecomunicações, com soluções para a automatização e controle do processo de aprovação, verificação e pagamento de documentos fiscais;
- Empresas de pequeno e médio porte a varejo e atacado, com soluções para automação comercial;
- Escolas de ensino fundamental e médio, com soluções para controle de cadastro de alunos, professores e funcionários. Com controle de lançamento de séries e notas para alunos, contagem de tempo para professores, controle de alocação de horários e salas e, controle de papeletas;
- Universidades com soluções para alocação de horários, salas e, disponibilidade de professores;
- Cooperativas e cafeicultores com soluções para controle para a pós-colheita do café.

Competidores:

- Atualmente a empresa não possui nenhum competidor por causa de seu nicho de mercado restrito. Mas visando a expansão do nicho de mercado da empresa, os competidores são empresas de grande porte que possuem soluções já consolidadas em gestão imobiliária e automação comercial.

Tamanho e posição na Indústria:

Atualmente a empresa se enquadrada como empresa de pequeno porte, visando um nicho de pequenos clientes, com exceção dos projetos que são em parceria com a Montreal Informática.

A expectativa é conseguir entrar no mercado regional de sistemas para automação comercial de pequenas e médias empresas. A empresa também almeja conseguir entrar no mercado de gestão de imobiliárias e,criar soluções em *GED/workflow* para controle de documentos fiscais de empresas de médio e grande porte.

Tabela 7.2 - Visão Descritiva da Fábrica de Software (Continuação).

Rentabilidade e Crescimento: • Atualmente a rentabilidade da empresa é baixa devido à falta de produtos para a geração de renda recorrente. E os projetos sobre demanda possuem um valor/hora baixo em relação ao mercado devido ao fato da maior parte da equipe dos projetos ser composta por estagiários.
Mundo Circundante: • Atualmente a empresa não é afetada pelas modificações políticas, mas ela necessita acompanhar as modificações tecnológicas.
Percepção pública: Atualmente a empresa não tem grande visibilidade no público em geral. Está sendo necessário um plano de divulgação da empresa para o público, com participação em eventos e reestruturação do site.
Nível de Serviço: Atualmente o nível de serviço para o cliente pode ser considerado relativamente bom, mas existem vários pontos a serem melhorados, a saber: • Rapidez em responder as solicitações ao cliente. • Melhorar o processo de implantação de produtos no cliente. • Melhorar o controle de versões bases do sistema. Melhorar a qualidade técnica dos colaboradores envolvidos.

Fonte: Fábrica de Software.

7.2.3. Criação do Modelo Conceitual

Para o desenvolvimento deste modelo, é utilizado o diagrama de classes da UML. Para representar os conceitos, são utilizadas classes e para as relações entre as classes, são utilizadas as associações existentes como generalização, composição ou mesmo a associação. Os elementos citados, do diagrama de classes da UML, foram os elementos utilizados na modelagem descrita a seguir.

Para realização deste modelo, foi realizada uma reunião com uma das pessoas chave,da Fabrica de Software, que possuía o conhecimento do negócio como um todo e por isso, a capacidade de identificar os conceitos existentes.

Os conceitos mais importantes que foram identificados, de um modo geral, foram: as pessoas que atuam na Fábrica de Software, chamadas internamente como colaboradores, explicitando os papéis existentes; o conceito do que é e o que é feito na Fábrica, os serviços prestados (foram explicitados os tipos de serviços que são oferecidos para os clientes como também os serviços executados internamente na Fábrica); os conceitos dos resultados gerados pelos serviços executados, como por exemplo, um produto final e do que ele é composto e por ultimo; os conceitos de clientes, que podem ser internos ou externos. O modelo gerado pode ser visualizado na Figura 7.1.

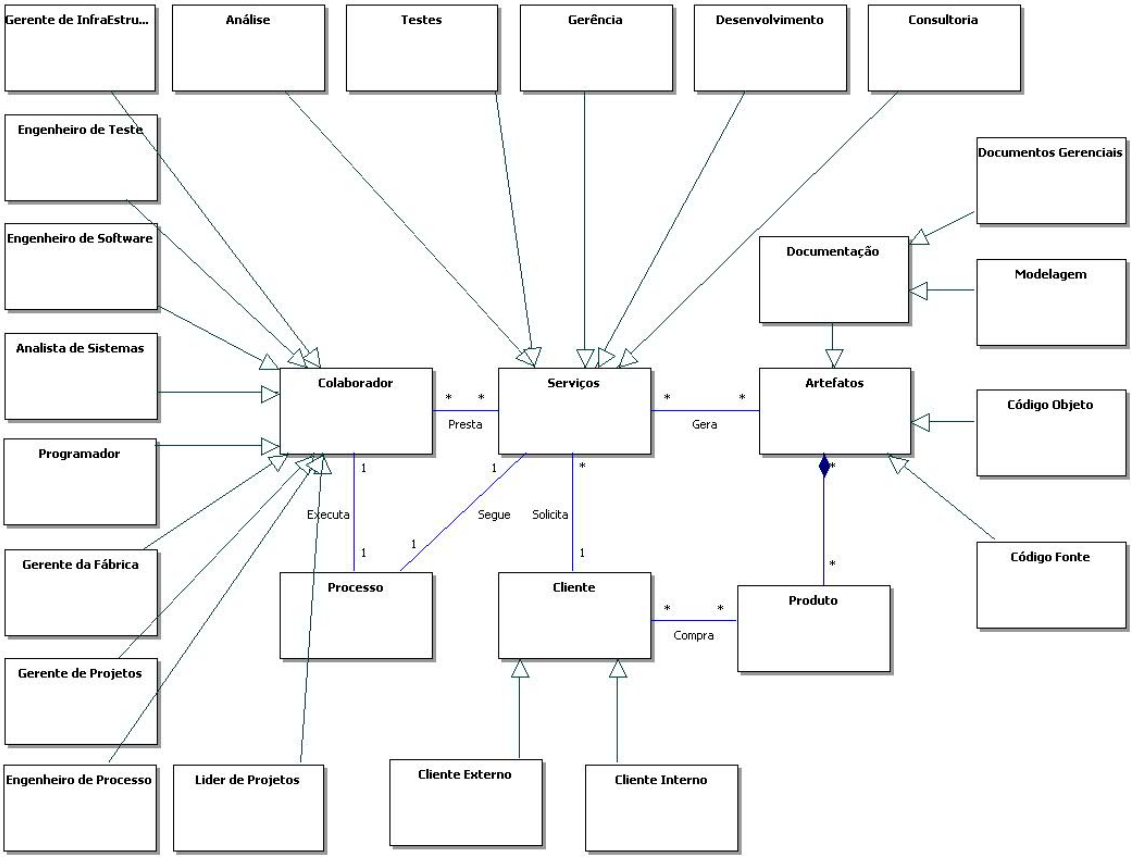


Figura 7.1 - Modelo Conceitual da Fábrica de Software. Fonte: Elaborado pelo autor.

Os conceitos relacionados às pessoas envolvidas na execução dos processos da Fábrica, ou colaboradores, que foram identificados, foram os papéis dos quais estes colaboradores poderiam exercer. Um detalhe importante, que é característico de

organizações de pequeno porte, e que ocorre nesta Fábrica de Software é de colaboradores assumirem diversos papéis. Os papéis que foram identificados foram:

- **Analista de Sistemas –** pessoa responsável pela análise dos sistemas que serão desenvolvidos, incluindo a parte de levantamento dos requisitos, análise e modelagem do sistema;

- **Engenheiro de Processos –** pessoa responsável pela definição e implantação de processos de apoio ao desenvolvimento de software na organização;

- **Engenheiro de Software –** pessoa que executa os papéis tanto de analista de sistemas, como de programador;

- **Engenheiro de Testes –** pessoa responsável em definir e executar os testes sobre o software desenvolvido, a fim de garantir a qualidade do produto;

- **Gerente da Fábrica –** responsável pela gerência da fábrica, cabendo a ele tomar as decisões estratégicas, aprovar ou reprovar a entrada de novos projetos e tomar as decisões internas;

- **Gerente de Infra-Estrutura –** responsável pela toda a infra-estrutura da fábrica, tanto como o funcionamento dos equipamentos, rede *intranet*, como também de algumas funções que seriam de um setor de RH (recursos humanos) como alocação dos equipamentos para os colaboradores, controle de faltas e horários dos colaboradores;

- **Gerente de Projetos –** responsável pela gerência de um ou mais projetos, elaboração de plano de projetos, cronogramas, alocação de colaboradores em projetos e responsável pela análise de riscos dos projetos. É sua responsabilidade a definição dos responsáveis pelas tarefas a serem executadas no projeto e acompanhamento do cronograma, de acordo com os dados que o líder de projetos repassa;

- **Líder de Projetos –** responsável pela distribuição, entre os participantes do projeto, das tarefas definidas pelo gerente de projetos, acompanhamento do desenvolvimento das tarefas pelos colaboradores, e também é de sua

responsabilidade resolver problemas técnicos que podem ocorrer no decorrer do desenvolvimento;

- **Programador –** responsável pela implementação dos sistemas a serem desenvolvidos, escrevendo o software propriamente dito, de acordo com as modelagens e requisitos gerados pelo analista de sistema.

Em relação aos serviços executados na Fábrica de Software, foi identificado que para um colaborador prestar um serviço dentro da Fábrica, ele deve executar um processo já definido para realização do serviço. O processo diz quais são as etapas a serem seguidas para a realização de um serviço. Uma definição melhor de serviço, que é tida dentro da Fábrica, é um conjunto de tarefas definidas que ao final é produzido um produto específico. Os serviços podem ser internos, para a realização de um produto, ou podem ser serviços oferecidos a clientes. Os tipos de serviços identificados foram:

- **Análise –** consiste em realizar o levantamento de um sistema e desenvolver a modelagem de análise do mesmo. Atualmente é um serviço interno e também é oferecido como um serviço para clientes;

- **Consultoria –** atualmente a consultoria é um serviço que é oferecido a clientes. A consultoria oferecida é na área de melhoria de processos;

- **Desenvolvimento –** consiste na implementação de um sistema, de acordo com uma análise já realizada. Este serviço é prestado internamente e também é oferecido para o cliente;

- **Gerência –** consiste em fazer a gerência de projetos de desenvolvimento de softwares. Este serviço, também, é oferecido para clientes e executado internamente;

- **Testes –** consiste na realização de testes sobre o sistema desenvolvido, a fim de garantir a qualidade do mesmo. Atualmente este serviço é oferecido somente internamente.

No serviço prestado de desenvolvimento de sistemas, realizado pela Fábrica de Software, geralmente é executado internamente os serviços de análise, gerência e testes para poder gerar o produto final requerido pelo cliente.

Os serviços são executados dentro da Fábrica para a geração de alguns produtos específicos. Geralmente no intuito da geração de um produto final que possa ser comercializado. Os produtos gerados pelos serviços são denominados, internamente na Fábrica, como artefatos. Os artefatos são:

- **Documentos Gerenciais –** todo documento gerencial gerado e necessário para realização da gerência de um projeto. Documento de Requisitos, Plano de Projeto e Cronograma são exemplos deste tipo de documento;
- **Modelagem –** todo documento gerado referente à modelagem de um sistema. São exemplos destes documentos Modelo de Análise, Modelo de Projeto, Documento de Caso de Uso;
- **Código-Fonte –** arquivos com código alto nível que será transformados em código-objeto;
- **Código-Objeto –** a parte executável do sistema, que geralmente é distribuído e implantado nos clientes.

Um serviço é executado por alguma solicitação de clientes. Estes clientes podem ser clientes externos, que pagam pelo serviço ou clientes internos, que surgem de acordo com a necessidade de execução de algum serviço dentro da organização. Além da solicitação de um serviço, um cliente pode também comprar um produto já existente na Fábrica de Software. Estes produtos são compostos por artefatos, como um software composto de código-objeto e documentação, ou mesmo a compra de modelos de documentos gerenciais.

7.2.4. Criação do Diagrama Objetivos/Problemas

Para o desenvolvimento deste diagrama, foi utilizado o diagrama de classes da UML, onde os objetivos são representados por instâncias de classes com o estereótipo de *goal*. Um objetivo pode ser Quantitativo, quando pode ser facilmente mensurado o quanto foi atingido, ou Qualitativo, que não é facilmente mensurado, necessitando o julgamento humano para ser mensurado o quanto foi atingido. Os objetivos podem ser decompostos em sub-objetivos, tendo uma relação representada pela associação de dependência (*dependency*) da UML. Os objetivos podem ser decompostos completamente ou não, a

representação destas duas situações é dada pela associação de dependência estereotipada de *complete* e *incomplete*, respectivamente.

Os problemas são representados pelo elemento nota (*note*) da UML com o estereótipo de *problem*. Um problema sempre está associado a um objetivo, pois um problema atrapalha a realização do objetivo no qual ele está associado.

Pode ser identificada uma ação a ser tomada em relação a um problema. Esta ação é representada pelo elemento nota (*note*) da UML estereotipada de *action*. Uma ação sempre está ligada ao problema, no qual ela pretende sanar.

Para realização deste diagrama, também foi realizado uma reunião com duas pessoas chaves da Fábrica de Software. Uma destas pessoas era um dos Gerentes da Fábrica, que tinha uma melhor visão dos objetivos que a Fábrica pretende atingir. O modelo de objetivos/problemas criado, como resultado da reunião pode ser visualizado na Figura 7.2.

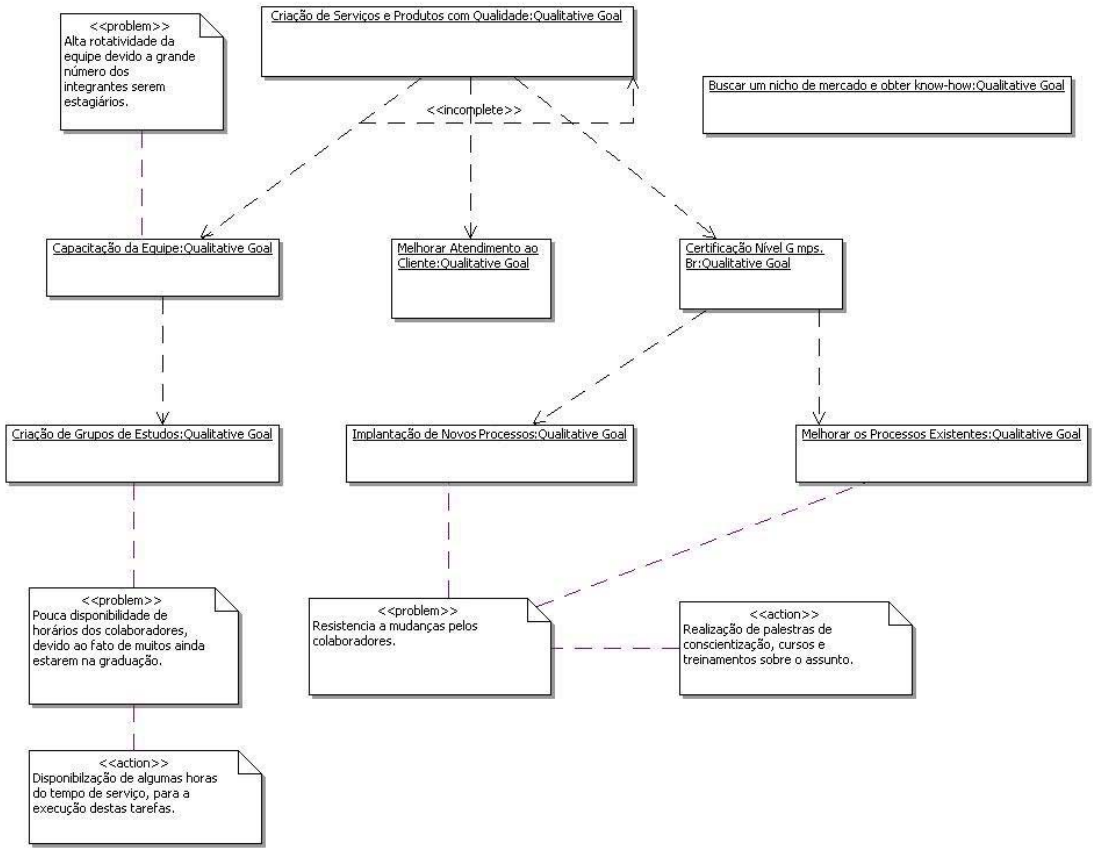


Figura 7.2 - Diagrama de Objetivos/Problemas da Fábrica de Software. Fonte: Elaborado pelo autor.

Os principais objetivos identificados para a Fábrica de Software foram três:

- Criação de serviços e produtos com qualidade;
- Capacitação da equipe; e
- Buscar um nicho de mercado e obter um *Know-How* nesta área.

O primeiro, dos três objetivos identificados, e também considerado o mais importante para a Fábrica de Software é visando a conquista da qualidade nos serviços prestados e produtos oferecidos. Este objetivo pôde ser decomposto, não completamente, em outros objetivos. O primeiro deles, e que foi mais facilmente identificado pelos responsáveis da Fábrica, foi “Melhorar o Atendimento ao Cliente”, visto que isto já é uma necessidade atual da Fábrica. O segundo, é a obtenção da certificação Nível G do modelo

MPS.BR, no qual a empresa já está trabalhando a algum tempo. E o terceiro, e também considerado com um dos principais objetivos da empresa, a qualificação técnica e intelectual da equipe que compõe a Fábrica de Software.

O objetivo de obter a certificação Nível G do modelo MPS.BR também foi decomposto em dois outros sub-objetivos. O primeiro é a melhoria constante dos processos existentes na Fábrica de Software e o segundo é a implantação de novos processos para algumas áreas que ainda não possuem processos institucionalizados.

Foi identificado que para a realização destes dois objetivos, a Fábrica teria que enfrentar a resistência a mudanças pelos colaboradores na Fábrica. Para contornar este problema, uma ação que pode ser tomada é a realização de palestras de conscientização, cursos e treinamentos sobre os processos a serem implantados ou melhorados.

O objetivo de capacitação técnica e intelectual da equipe que compõe a Fábrica, também foi decomposta em outro objetivo, o de criação de grupos de estudos de diversos temas de interesse dos colaboradores. Mas foi identificado o problema de disponibilidade de horários dos colaboradores, devido ao fato da maioria ainda estarem fazendo o curso de graduação. Uma possível ação que a poderia ser tomada, para contornar este problema, seria a disponibilização de algumas horas de serviço do colaborador para sua participação nestes grupos de estudos.

Outro problema identificado em relação à capacitação das equipes, foi o fato da alta rotatividade que existe na Fábrica, devido muitos dos colaboradores serem estagiários. Isto faz com que o colaborador não fique tempo suficiente na Fábrica para se qualificar, ou mesmo se desliga da organização após já ter sido bem qualificado.

O último objetivo relatado pelos responsáveis pela Fábrica de Software, foi a busca de um nicho específico de mercado para a especialização e obtenção de um *know-how* nesta área. Dos três objetivos de maior relevância para a organização, este foi identificado como o de menor prioridade, visto que a Fábrica já está atuando em determinados nichos.

7.3. Modelagem de Visão Business Process

A visão “Processos de Negócio” (*Business Process*) tem como objetivo mostrar as atividades que são executadas para atingir os objetivos, e mostras as relações entre as

atividades e os recursos que nela participam. A definição dos processos são de grande utilidade para o entendimento do negócio, para poder visualizar as oportunidades de melhoria ou inovação.

O diagrama de processos é utilizado para descrever os processos que são executados dentro de uma empresa. O objetivo principal dele é descrever como estes processos são executados, mostrando as relações entre os processos e os recursos envolvidos.

Para o desenvolvimento deste diagrama é utilizado o diagrama de atividades da UML. Neste diagrama para representar um processo, foi utilizado um elemento ação (*action*) da UML, estereotipado como *process*. Por causa de restrições da ferramenta utilizada, toda a modelagem foi inserida dentro de um único elemento atividade (*activity*). O fluxo entre os processos é representado pelo elemento fluxo de controle (*control flow*), que indica a ordem na qual os processos são executados. A representação dos recursos é feita a partir da extensão de um objeto, o elemento *Instance Specification* da UML. Os estereótipos que irão representar os recursos são *abstract*, *physical*, *people*, *information* e *resource*. A ligação destes recursos a processos é feito com a extensão da dependência (*dependency*) da UML. Os estereótipos criados para mostrar as relações dos recursos com os processos são *control*, *supply* e *non casual*. Estes recursos podem ser objetos de entrada a um processo ou objetos de saída. As pessoas (*people*) possuem a relação de controle (*control*) com um processo.

Para a definição do diagrama de processos, foi utilizada a documentação existente na Fábrica de Software sobre os processos institucionalizados. Os processos existentes na Fábrica foram definidos baseados em diversos modelos de qualidade, tendo gerado dois trabalhos de conclusão de curso e uma bolsa de iniciação científica (Aguiar, 2004, Liebmam, 2006, Machado, 2005).

O processo de desenvolvimento de software da empresa pode ser expresso em quatro grandes processos, Prospecção, Planejamento, Desenvolvimento e Fechamento, como é ilustrado na Figura 7.3.

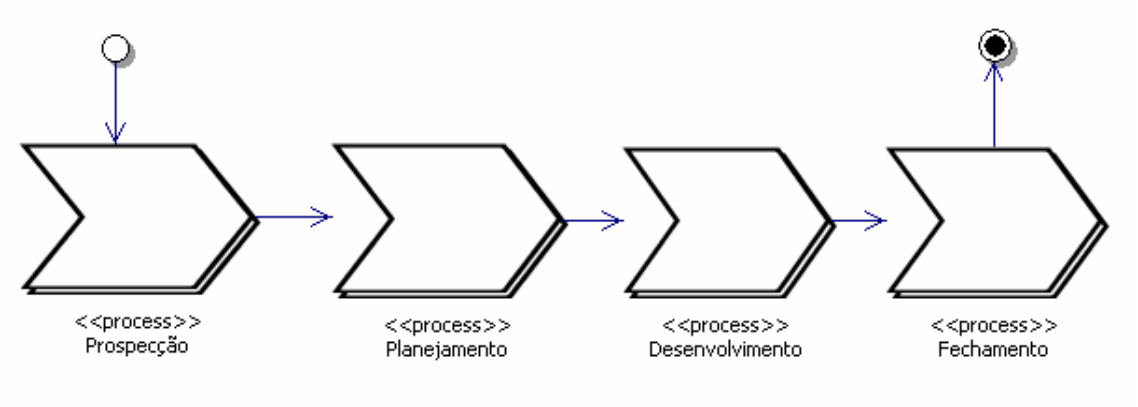


Figura 7.3 - Diagrama de Processos Geral da Fábrica de Software. Fonte: Elaborado pelo autor.

Cada processo deste é subdivido em inúmeros outros sub-processos. Este trabalho não irá descrever cada atividades contidas nos processos, pois isto fugiria do escopo do trabalho. Ao invés disto, foi feito a modelagem mais alto nível dos processos, como é mostrado na Figura 7.4. Esta modelagem retrata os processos principais da Fábrica de Software, sem entrar em muitos detalhes.

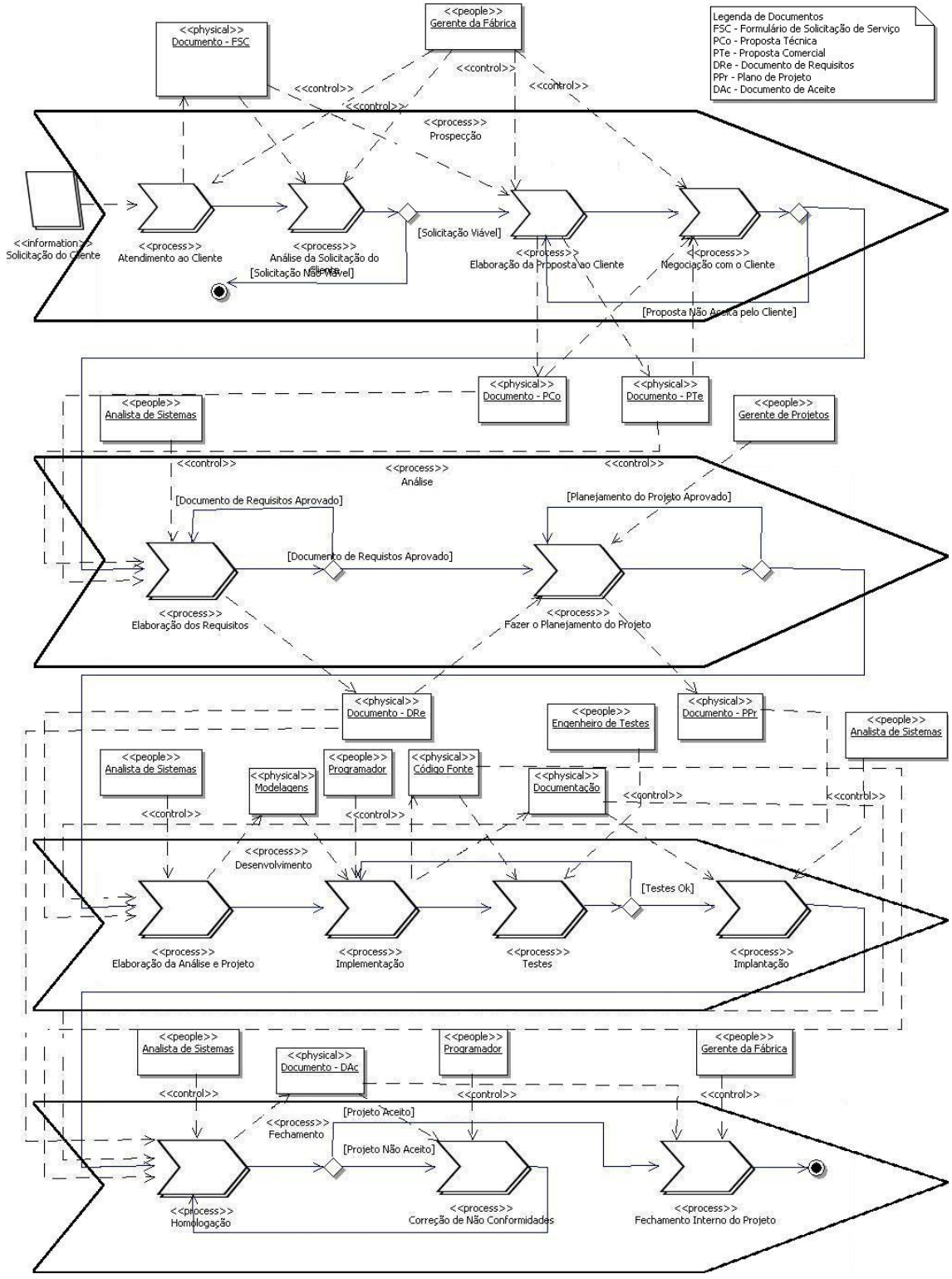


Figura 7.4 - Diagrama de Processos da Fábrica de Software. Fonte: Elaborado pelo autor.

O processo de Prospecção é iniciado quando algum cliente faz a solicitação de realização de algum serviço de desenvolvimento de algum sistema de software. O Gerente da Fábrica que é o responsável pela execução deste processo. Após ser feito o atendimento ao cliente, é preenchido o Formulário de Solicitação de Serviço, que será analisado a

viabilidade da solicitação, caso seja inviável, o projeto em prospecção é encerrado, caso contrário é elaborada uma proposta para o cliente. A proposta consiste nos documentos de Proposta Técnica e Proposta Comercial, que é levado para negociação com o cliente. Caso o cliente não aceite a proposta, ela reformulada, caso contrário passa para o processo de Análise.

No processo de Análise, um Analista de Sistemas faz o levantamento de requisitos do cliente gerando o Documento de Requisitos, de acordo com o que foi proposto na Proposta Técnica e Comercial. Após a elaboração do Documento de Requisitos, ele é submetido para a aprovação do cliente, que caso aprove será servirá como base para a elaboração de Plano de Projeto pelo Gerente de Projetos. Caso o Plano de Projeto seja aprovado, iniciará o processo de Desenvolvimento.

O processo de Desenvolvimento é iniciado com a elaboração da análise e projeto do sistema em questão. O responsável por este sub-processo é um Analista de Sistema que deverá elaborar as modelagens e documentos referentes do sistema, que servirá de base para que os programadores possam implementar o sistema. Após a implementação, o código gerado irá ser testado para por garantir a qualidade do produto, onde caso seja encontrado erros, são reportados para os programadores corrigirem e caso contrário já feito a implantação do sistema.

Após o processo de Desenvolvimento, é iniciado o processo de Fechamento, onde o Analista de Sistemas faz a homologação do sistema implantado com o cliente. Caso o projeto não seja aceito, ele é repassado para o programador corrigir as não conformidades. Caso o projeto seja aceito, o Gerente de Fábrica faz o fechamento interno do projeto realocando os recursos que participavam deste projeto em outros projetos em andamento na Fábrica de Software.

7.4. Considerações Finais

O estudo de caso foi realizado usando as visões Visão de Negócios e Processos de Negócios. O desenvolvimento da modelagem da Visão de Negócios enfrentou como obstáculo a falta de definição dos temas que ela aborda, isto é, não havia uma visão clara dos objetivos, dos conceitos e dos princípios da Fábrica de Software, o que resultou em reuniões demoradas o que resultou em um atraso no cronograma.

Em contra partida, a elaboração da visão de Processos de Negócios teve como facilidade o fato da Fábrica de Software ter o processo de desenvolvimento de software definidos e institucionalizados, resultado do constante esforço desempenhado com o intuito de uma certificação em um modelo de qualidade.

A maior contribuição que o desenvolvimento da modelagem de negócio para a Fábrica de Software foi a definição dos objetivos, dos conceitos e dos princípios apresentados na seção 7.2.

A indisponibilidade das pessoas chaves devido a sua alocação em projetos da Fábrica de Software impossibilitou a realização da modelagem das visões Estrutura de Negócio e Comportamento de Negócio, ficando a execução destas como trabalho futuro, uma vez que é de interesse da Fábrica de Software.

8. Considerações

Neste trabalho foi realizada uma adaptação do método de modelagem de negócios Eriksson-Penker que utiliza UML para incorporar as características inovadoras da UML 2.0. Após a adaptação foi realizado um estudo de caso da modelagem de negócios de uma fábrica de software utilizando o resultado da adaptação.

A adaptação consistiu redefinir os estereótipos existentes no método atual, de acordo com os novos conceitos de extensão da UML 2.0. Entretanto, não foi notado uma grande diferença na utilização dos elementos adaptados, em relação ao já existente. A diferença mais relevante foi verificada no modo de criação dos estereótipos e não na utilização do mesmo.

A característica mais interessante identificada na UML 2.0 foi o conceito de *profiles* que permite com que as características usadas num modelo possam ser utilizadas em outros facilmente. Dessa forma, os estereótipos criados para o método Eriksson-Penker podem ser utilizados em outros modelos que poderão ser desenvolvidos.

A modelagem de negócio da Fábrica de Software como estudo de caso mostrou ser possível utilizar os elementos adaptados do método Eriksson-Penker sem maiores dificuldades.

Observou-se que a modelagem de negócios permite um conhecimento maior do negócio em questão e no caso da Fábrica de Software, melhorar a definição dos objetivos e fazer modificações nos processos para melhor atingi-los.

A modelagem de negócio realizada na Fábrica de Software foi aprovada pelos responsáveis que concluíram que ela retrata a realidade na qual foi desenvolvida.

8.1. Contribuições e Trabalho Futuros

As principais contribuições do trabalho foram: o estudo sobre a UML 2.0 dando ênfase no uso de *profiles* visto que o tema ainda é algo novo e sem muitos trabalhos publicados; o estudo e utilização de um método de modelagem de negócio, o Eriksson-Penker, mostrando detalhadamente como foi realizada a modelagem; e a criação de um *profile* referente às extensões utilizadas no método Eriksson-Penker, a descrição de como foi realizada e a aplicação do uso em uma fábrica de software.

Podem ser realizados como trabalhos futuros, a partir deste: a modelagens das visões de Estrutura de Negócio e Comportamento de Negócio da Fábrica de Software; disponibilização do *profile* desenvolvido em forma de *plug-in* da ferramenta *Borland*® *Together*® na *internet*; criação do *profile* referente à modelagem de negócio pelo método Eriksson-Penker, baseada na adaptação realizada neste trabalho, para outras ferramentas que suportam UML 2.0; e formalização dos objetivos e conceitos identificados na modelagem dentro da Fábrica de Software.

REFERÊNCIAS BIBLIOGRÁFICAS

AGUIAR, H. V. **PEPP**: Processo de Software para Empresas de Pequeno Porte Baseado no Modelo CMMI. Monografia de Graduação, Departamento de Ciência da Computação, Universidade Federal de Lavras. 2004

ALENCAR, F., **Mapeando a Modelagem Organizacional em Especificações Precisas**. Tese de Doutorado, Centro de Informática, Universidade Federal de Pernambuco, 1999.

BOOCH, G. **Object-Oriented Analysis and Design with Applications**. Benjamin–Cummings, Redwood City, Calif., 1st edition. 1991.

BOOCH G., RUMBAUGH, J., JACOBSON, I. **The Unified Modeling Language User Guide**. 2 ed. Addison Wesley Professional. 2005.

BORLAND. **Tecnologias Together** Disponível em: <http://www.borland.com/br/products/together/index.html>. Acesso em: 10 de jul. de 2006.

BUBENKO, J. A. Jr., PERSSON, A., STIRNA, J., **D3**: EKD User Guide. Royal Institute of Technology (KTH) and Stockholm University, Stockholm, Sweden, 2001.

BUBENKO, J. A. Jr., STIRNA, J., BRASH, D. **EKD User Guide**. Department of Computer and Systems Sciences, Royal Institute of Technology, Stockholm, Sweden. 1997.

BUBENKO, J. A. Jr., STIRNA, J., BRASH, D. **EKD User Guide**. ESPRIT Program 7.1, project number 22927, Electrical Knowledge for Transforming Applications, Royal Institute for Technology, Stockholm. 1998.

CHEN-BURGER, Y., Robertson, D., Stader, J. **Formal Support For An Informal Business Modelling Method**. PhD thesis. Department of Artificial Intelligence, The University of Edinburgh, 80 South Bridge, Edinburgh, UK, 2001.

DARNTON, G., DARNTON, M. **Business Process Analysis**. Cambridge, U.K.: Thomson Business Press, 1997.

ERIKSSON, H. E., PENKER, M., LYONS, B., FADO, D. **UML™ 2 Toolkit.** 1 ed. Wiley Publishing, Inc., Indianapolis, Indiana. 2004.

ERIKSSON, H. E., PENKER, M. **Business Modeling with UML**: Business Patterns at Work. 1 ed. John Wiley & Sons, Inc., 2000.

GALE, T., Eldred, J. **Getting Results with the Object-Oriented Enterprise Model**. New York: SIGS Books. 1996.

IBM, **Business System Development Method:** Introducing BSDM, second ed., London: IBM UK, 1992.

JACOBSON, I., CHRISTERSON, M., JONSONN, P., OVERGAARD, G. **Object-Oriented Software Engineering**: A Use Case Driven Approach. Addison-Wesley, 1992.

JUNG, C. **Metodologia Para Pesquisa & Desenvolvimento** –Aplicada a Novas Tecnologias, Produtos e Processos. Rio de Janeiro: Axcel Books, 162 p. 2004.

LIEBMAM, A. **Melhoria no Processo de Software**: Implantação do MPS.BR Nível G em uma Empresa de Pequeno Porte. Monografia de Graduação, Departamento de Ciência da Computação, Universidade Federal de Lavras. 2006

MACHADO, B. B. **Metodologia para Gestão do Processo de Qualidade de Software**. Relatório de Iniciação Científica, Departamento de Ciência da Computação, Universidade Federal de Lavras. 2005.

MOF, OMG. **OMG's MetaObject Facility (MOF)**. Disponível em: < http://www.omg.org/mof/>. Acesso em: 15 de jul. 2006.

MONTEIRO, A. A. N. S. **Modelagem de Negócio na Prática**: Um Método para Suportar a Compreensão e Comunicação das Necessidades dos Negócios. Recife: UFPE, 2003

OCL, OMG. **Object Constraint Language:** OMG Avaliable Specification. Disponível em: < http://www.omg.org/docs/formal/06-05-01.pdf>. Acesso em: 12 de jul. 2006.

OMG. **Business Object Architecture Proposal**. Framingham, MA: OMG. 1998.

PERSSON, A., **Enterprise Modeling in Practice**: Situational Factors and their Influence on Adopting a Participative Approach. PhD Thesis, Department of Computer and Systems Sciences, Stockholm University/Royal Institute of Technology, Sweden, 2001

RUMBAUGH, J., BLAHA, M., PREMERLANI, W., EDDY, F., LORENSEN, W. **Object-oriented Modeling and Design**. Prentice Hall. 1991.

RUMBAUGH, J., JACOBSON, I., BOOCH, G**. The Unified Modeling Language Reference Manual**. 2 ed. Addison Wesley. 2004.

SANTOS, R., CAMEIRA, R., CLEMENTE, A., CLEMENTA, R. **Engenharia de processos de negócios**: aplicações e metodologias. In: ENCONTRO NACIONAL DE

ENGENHARIA DE PRODUÇÃO, 22, Curitiba. Anais Eletrônicos. Porto Alegre: ABEPRO. 2002. 1 CD.

SIY, H. P., MOCKUS, A., HERBSLEB, J. D., KRISHNAN, M., TUCKER, G. T. **Making The Software Factory Work**: Lessons from a decade of experience, In the Seventh International Symposium on Software Metrics, London, England, April 2001.

UML, OMG. **UML® Resource Page**. Disponível em: <http://www.uml.org/>. Acesso em: 18 de jul. 2006.

VERNADAT, F. **Enterprise Modeling and Integration**. London, England: Chapman & Hall, 1996.

YU, E., **Modeling Strategic Relationship for Process Reengineering**. PhD thesis, Computer Science Department, University of Toronto. Toronto, Canada. 1995.