UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
NÚCLEO DE COMPUTAÇÃO ELETRÔNICA

DIEGO OLIVEIRA ARAÚJO

ELABORAÇÃO DE ESPECIFICAÇÕES DE CASOS DE USO PARA LINHAS DE PRODUTO DE SOFTWARE BASEADA EM FRAGMENTOS

Rio de Janeiro

2010

Diego Oliveira Araújo

ELABORAÇÃO DE ESPECIFICAÇÕES DE CASOS DE USO PARA LINHAS DE PRODUTO DE SOFTWARE BASEADA EM FRAGMENTOS

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Informática do Departamento de Ciência da Computação, Instituto de Matemática, Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Informática.

Orientador: Eber Assis Schmitz.

Co-orientador: Alexandre Luis Correa.

Rio de Janeiro

2010

A662 Araújo, Diego Oliveira
Elaboração de especificações de casos de uso para linhas de produto de software baseada em fragmentos. / Diego Oliveira Araújo. – Rio de Janeiro, 2010.
132 f.: il.

Dissertação (Mestrado em Informática) – Universidade Federal do Rio de Janeiro, Instituto de Matemática, Núcleo de Computação Eletrônica, 2010.

Orientador: Eber Assis Schmitz.
Co-orientador: Alexandre Luis Correa.

1. Linhas de Produto de Software – Teses. 2. Engenharia de Requisitos – Teses. 3. Casos de Uso – Teses. 4. Fragmentos de Casos de Uso – Teses. I. Eber Assis Schmitz (Orient.). II. Alexandre Luis Correa (Co-orient.). III. Universidade Federal do Rio de Janeiro. Instituto de Matemática. Núcleo de Computação Eletrônica. IV. Título.

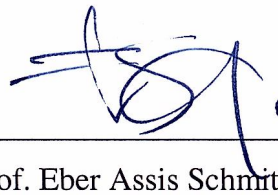
CDD

Diego Oliveira Araújo

ELABORAÇÃO DE ESPECIFICAÇÕES DE CASOS DE USO PARA LINHAS DE PRODUTO DE SOFTWARE BASEADA EM FRAGMENTOS

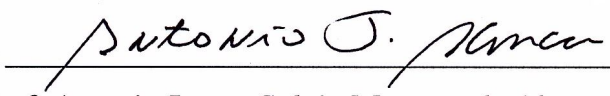
Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Informática do Departamento de Ciência da Computação, Instituto de Matemática e Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro como parte dos requisitos necessários à obtenção do título de Mestre em Informática.

Aprovada em: 25 de fevereiro de 2010.

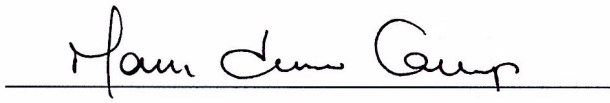


Prof. Eber Assis Schmitz, Ph.D. – Orientador – UFRJ/DCC/IM-NCE

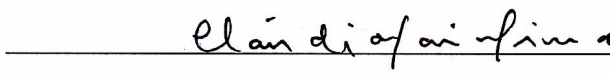
Prof. Alexandre Luis Correa, D.Sc. – Co-orientador – UFRJ/PPGI



Prof. Antonio Juarez Sylvio Menezes de Alencar, D.Phil. – UFRJ/NCE



Profa. Maria Luiza Machado Campos, Ph.D. – UFRJ/PPGI



Profa. Cláudia Maria Lima Werner, D.Sc. – UFRJ/COPPE

AGRADECIMENTOS

Acima de tudo, agradeço a Deus pelo dom da vida e por, em muitos momentos aflitivos, proporcionar-me a sua paz e a serenidade para enfrentar os obstáculos que atravessaram o meu caminho e superar os desafios.

Agradeço aos meus pais, José Mário e Eunice, pelo incentivo e por me proporcionarem a oportunidade de estudar. Agradeço a minha irmã, Juliana, por todo o apoio e incentivo que tenho recebido.

Agradeço a minha namorada, Vanessa Núúd, pelo companheirismo, respeito e compreensão dos inúmeros momentos dedicados a este trabalho.

Agradeço aos professores Eber Assis Schmitz e Alexandre Luis Correa pela orientação acadêmica e por compartilharem comigo os conhecimentos que possuem. Agradeço por acreditarem na minha capacidade e por estarem dispostos a me ajudar.

Agradeço aos professores Cláudia Maria Lima Werner, Maria Luiza Machado Campos e Antonio Juarez Sylvio Menezes de Alencar por, gentilmente, aceitarem o convite para participar da banca examinadora deste trabalho e pelas contribuições e sugestões dadas.

Agradeço ao professor Fernando Manso, aos amigos Felipe Dias e Carlos Alessandre, e aos demais colegas do curso de mestrado, pelas contribuições dadas ao longo das reuniões semanais do grupo de pesquisa.

Agradeço aos professores que participaram da avaliação deste trabalho nos diversos seminários do programa de mestrado e na sessão técnica do III Simpósio Brasileiro de Componentes, Arquiteturas e Reutilização de Software.

Agradeço aos amigos de trabalho, Glauco Santos e Maria Rute, pelo apoio e compreensão da realização deste curso de mestrado.

Agradeço à Universidade Federal do Rio de Janeiro pelo curso de mestrado que estou concluindo, a todos os professores, secretárias e funcionários do departamento.

Por fim, agradeço a todos aqueles que, de alguma forma, contribuíram na elaboração deste trabalho.

RESUMO

ARAÚJO, Diego Oliveira. **Elaboração de especificações de casos de uso para linhas de produto de software baseada em fragmentos**. 2010. 132 f. Dissertação (Mestrado em Informática) – Instituto de Matemática, Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2010.

A área de reúso de software tem evoluído a partir do reúso de componentes individuais em direção ao reúso em larga escala com as Linhas de Produtos de Software (LPS). Casos de uso são largamente utilizados na elicitação e especificação de requisitos funcionais de sistemas de informação. Entretanto, escrever especificações textuais de casos de uso para uma família de sistemas é uma tarefa trabalhosa, uma vez que o comportamento dos usuários e do sistema durante a troca de informações entre ambos pode variar entre os membros da linha de produto. Para superar estes problemas, este trabalho apresenta uma abordagem para a elaboração de especificações de casos de uso para LPS baseada em fragmentos. Um fragmento de caso de uso é uma seqüência de interações entre o ator e o sistema que pode ser reutilizado em diferentes partes dos documentos de requisitos. A abordagem consiste de duas atividades principais: engenharia do domínio e da aplicação. Na primeira atividade, um conjunto de fragmentos de casos de uso do domínio é elaborado a partir da análise dos objetivos comuns e variáveis da linha de produto. Na segunda atividade, descrições textuais de casos de uso para uma aplicação específica podem ser rapidamente produzidas através da composição e personalização desses fragmentos.

Palavras-chave: Linhas de Produto de Software. Engenharia de Requisitos. Casos de Uso. Fragmentos de Casos de Uso.

ABSTRACT

ARAÚJO, Diego Oliveira. **Elaboração de especificações de casos de uso para linhas de produto de software baseada em fragmentos**. 2010. 132 f. Dissertação (Mestrado em Informática) – Instituto de Matemática, Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2010.

The area of software reuse has evolved from the reuse of individual components to the large-scale reuse in Software Product Lines (SPL). Use cases are widely employed for functional requirements elicitation and specification of information systems. However, writing use case textual specifications for a SPL is a laborious task, since the information exchanging behavior between user and system may vary among the members of the SPL. To overcome this issue, this dissertation presents an approach for the elaboration of use case specifications for SPL based on fragments. A use case fragment is a sequence of interactions between the actor and the system that can be reused in different parts of the requirement documents. The approach consists of two main activities: domain and application engineering. In the first activity, a set of domain use case fragments is elaborated from the analysis of common and variable product line goals. In the second activity, use case textual descriptions for a specific application can be quickly produced by composition and customization of those fragments.

Keywords: Software Product Lines. Requirements Engineering. Use Cases. Use Case Fragments.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

COTS	*Commercial-Off-The-Shelf*
CREWS	*Co-operative Requirements Engineering With Scenarios*
ESPLEP	*Evolutionary Software Product Line Engineering Process*
FODA	*Feature-Oriented Domain Analysis*
GETI	Gestão Estratégica de Tecnologia da Informação
GPS	*Global Positioning System*
IEEE	*Institute of Electrical and Electronics Engineers*
LPS	Linha de Produto de Software
OO	*Object-Oriented*
OOSE	*Object-Oriented Software Engineering*
PLP	*Product Line Practice*
PLUC	*Product Line Use Cases*
PLUS	*Product Line UML-Based Software Engineering*
PLUSS	*Product Line Use Case Modeling for Systems and Software*
PU	Processo Unificado
SEI	*Software Engineering Institute*
UML	*Unified Modeling Language*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

1.1 Motivação

O processo de desenvolvimento de software tem mudado consideravelmente nas últimas décadas. À medida que a indústria de software eleva a sua capacidade de produção, ela também eleva a demanda por sistemas mais complexos e com melhor qualidade (KOTONYA, SOMMERVILLE, 1998). Assim, a competitividade entre elas aumenta de acordo com a habilidade para reagir rapidamente às mudanças do mercado e aos requisitos dos usuários (ATKINSON *et al.*, 2002). A forma tradicional de se desenvolver sistemas a partir do zero não atende mais essas demandas e a indústria de software tem passado por algumas mudanças para tentar solucionar tais questões.

O reúso de software é uma abordagem que tem como objetivo a criação de novos sistemas a partir de software existente, de forma a reduzir o esforço necessário para o desenvolvimento e manutenção desses novos sistemas (KRUEGER, 1992). A aplicação dessa abordagem em larga escala tem sido uma meta da indústria de software desde o surgimento do processo de engenharia de software nos finais dos anos sessenta. Várias tecnologias têm surgido para solucionar os problemas de reúso de software, desde rotinas para modularizar software até a programação baseada em componentes. Entretanto, muitas dessas tecnologias dão ênfase ao reúso em codificação, sendo que essa atividade representa apenas cerca de 20% do custo total de desenvolvimento de software (GOMAA, 2004).

Recentemente, desenvolvedores e gerentes perceberam que o reúso de requisitos e de arquiteturas de software pode trazer benefícios maiores do que os obtidos apenas com o reúso de componentes individuais de software. Nesse contexto, Linha de Produto de Software (LPS) surgiu como uma abordagem fundamental por focar no desenvolvimento de uma família de produtos visando um mercado específico e compartilhando um conjunto comum de artefatos

(CLEMENTS, NORTHROP, 2002). Dentre os benefícios da adoção de uma abordagem de LPS, destacam-se: redução do custo de desenvolvimento, aumento da qualidade do produto, redução do tempo de entrega dos produtos, utilização mais eficiente dos recursos humanos, dentre outros (POHL *et al*., 2005; SEI, 2010).

Uma LPS consiste de algumas áreas práticas (SEI, 2010), como, por exemplo, a Definição da Arquitetura, o Desenvolvimento de Componentes, a Integração entre Sistema e Software, Testes e a Engenharia de Requisitos, a qual requer um esforço significativo em projetos de software. Dois diferentes problemas devem ser abordados durante a elicitação e especificação de requisitos de uma LPS: (a) como identificar os requisitos comuns e variáveis entre todos os membros da linha de produto; (b) como especializar e instanciar requisitos genéricos da linha de produto para obter-se os requisitos de uma aplicação específica. Para tratar esses problemas, a relação entre requisitos do produto e da linha tem sido representada em abordagens de modelagem, que precisam suportar os conceitos de parametrização, especialização e generalização (BERTOLINO *et al.*, 2006).

A especificação de requisitos através de casos de uso tem se tornado uma das formas mais utilizadas dentre as diversas abordagens disponíveis no contexto de documentação de funcionalidades de software (KULAK, GUINEY, 2003). Os casos de uso capturam os requisitos funcionais do sistema através de descrições textuais que devem especificar todas as interações entre o sistema e seus usuários (ROSENBERG, STEPHENS, 2007). Essa técnica tornou-se muito popular devido ao seu estilo de escrita informal e simples, que é compreensível tanto para os técnicos como para os interessados não-técnicos que participam da elaboração dos documentos de requisitos de software.

O sucesso dos casos de uso tem gerado idéias para sua utilização no paradigma de linhas de produto, como os trabalhos apresentados em (KIM *et al.*, 2006) e (BRAGANÇA, MACHADO, 2007). Entretanto, a elaboração de especificações de casos de uso para LPS é

uma tarefa trabalhosa, uma vez que o comportamento dos usuários e do sistema durante a troca de informações entre ambos pode variar entre os membros da linha de produto. Além disso, é importante também ressaltar que o modelo de casos de uso resultante pode se tornar sobrecarregado para o uso prático, devido à presença de um número consideravelmente maior de cenários alternativos que são utilizados para representar a variabilidade (TRIGAUX, HEYMANS, 2003).

Assim, a motivação para este trabalho resulta da percepção da necessidade de uma forma de identificar e representar a variabilidade das interações entre usuários e sistemas existentes nas aplicações alvo de uma LPS, de forma a agilizar a personalização e reutilização das descrições textuais de casos de uso resultantes durante a especificação de requisitos de uma dada aplicação da linha de produto.

1.2 Caracterização do Problema

A elaboração de especificações de casos de uso para LPS requer a identificação de casos de uso comuns e variáveis entre os membros da linha de produto e a elaboração de descrições textuais genéricas que sejam de fácil utilização durante a especificação de requisitos de uma aplicação específica. Neste trabalho, estamos interessados em apoiar a produção de especificações de casos de uso para linhas de produto, no contexto de casos de uso elaborados a partir da análise de um conjunto de aplicações alvo. Acreditamos que possa haver um meio de agilizar a instanciação desses artefatos numa abordagem de LPS.

1.3 Objetivo

O objetivo do presente trabalho é apresentar uma abordagem para a elaboração de especificações de casos de uso para linhas de produto baseada no conceito de fragmentos.

Nessa abordagem, especificações textuais de casos de uso para uma dada aplicação podem ser rapidamente produzidas através da composição e personalização desses fragmentos. Cada fragmento de caso de uso representa uma seqüência de interações recorrentes entre usuários e sistema, coletadas a partir de aplicações existentes (DIAS *et al.*, 2008).

Além de auxiliar a instanciação de especificações de casos de uso, a abordagem apresenta também como um conjunto de fragmentos reutilizáveis do domínio pode ser elaborado a partir da análise dos objetivos comuns e variáveis da linha de produto.

1.4 Contribuições

A principal contribuição da abordagem proposta neste trabalho é auxiliar a elaboração de especificações de casos de uso para uma aplicação específica da LPS. Ao explicitar a forma como a reutilização das descrições textuais de casos de uso do domínio deverá ser realizada, ou seja, através da seleção, personalização e composição de fragmentos, será possível padronizar e agilizar a tarefa de instanciação desses artefatos durante a especificação de requisitos de um dado membro da linha de produto.

Além disso, a abordagem apresenta uma forma de modelagem de requisitos para LPS que visa identificar e descrever as informações sobre a variabilidade do comportamento dos usuários e do sistema entre os membros da linha de produto. O modelo de casos de uso do domínio resultante apresenta vantagem sobre os demais modelos disponíveis na literatura por não apresentar cenários alternativos adicionais, que seriam usados para representar as variações das funcionalidades, tornando-o mais simples e de fácil utilização.

Finalmente, destaca-se como uma contribuição adicional a produção de um catálogo de fragmentos de casos de uso para um domínio específico, permitindo que haja um maior conhecimento sobre o domínio considerado quando a abordagem proposta é utilizada.

Este trabalho originou um artigo que foi publicado em anais de congresso (ARAÚJO *et al.*, 2009), cuja apresentação possibilitou discussões que contribuíram positivamente para a continuidade da investigação. As idéias e definições utilizadas no artigo forneceram a base necessária para a elaboração desta dissertação.

1.5 Organização da Dissertação

Esta dissertação está estruturada em cinco capítulos. No presente capítulo, apresenta-se uma introdução à pesquisa realizada, considerando-se a motivação para o estudo do tema, a caracterização do problema, o objetivo do trabalho, suas contribuições e organização.

No segundo capítulo é apresentado o embasamento teórico utilizado nesse trabalho. Primeiramente, tratando das Linhas de Produto de Software, com ênfase na sua engenharia de requisitos, que engloba a modelagem de casos de uso. Em seguida, é abordado o estudo do conceito de fragmentos de casos de uso e sua utilização na especificação de requisitos de software. Por último, são apresentados os trabalhos relacionados com a pesquisa.

A abordagem proposta para a elaboração de especificações de casos de uso para LPS baseada em fragmentos é discutida em detalhes no capítulo três, onde também é incluído um exemplo de utilização dessa abordagem. No quarto capítulo, apresenta-se um cenário de uso para a abordagem, que visa mostrar o uso prático da mesma. Finalmente, o capítulo cinco apresenta as conclusões desse trabalho, destacando a comparação com outras abordagens, as limitações encontradas e apontando possíveis melhorias e trabalhos futuros.

Além dos capítulos apresentados, esta dissertação conta com um apêndice que apresenta parte do catálogo de fragmentos de casos de uso desenvolvido durante o cenário de uso apresentado.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Introdução

Esse capítulo apresenta algumas informações que são relevantes para a compreensão da abordagem proposta nesta dissertação. Primeiramente, será descrito o paradigma de LPS, apresentando a sua definição, as atividades essenciais, os benefícios obtidos e as estratégias existentes. Em seguida, a engenharia de requisitos para LPS é mostrada, apresentando as sub-atividades envolvidas nessa atividade e uma abordagem existente para a modelagem de requisitos para linhas de produto. Além disso, o conceito de fragmentos de casos de uso é descrito, apresentando a sua definição, um exemplo de fragmento e a forma de utilização dos mesmos. Por último, serão apresentados os trabalhos relacionados com a abordagem proposta.

2.2 Linhas de Produto de Software

2.2.1 Introdução

A engenharia de linhas de produto, ou simplesmente linhas de produto, é uma abordagem para reúso que provê métodos para planejar, controlar e melhorar a infra-estrutura de reúso para o desenvolvimento de uma família de produtos similares (JOHN, MUTHIG, 2002). O conceito de linhas de produto não é novo. Estima-se que o primeiro exemplo de linha de produto surgiu na história antiga com a construção das pirâmides do Egito. Atualmente, os exemplos mais modernos de linhas de produto se originam das linhas de aeronaves na indústria aeronáutica e dos modelos de telefones celulares de determinados fabricantes (SEI, 2010).

A abordagem de linhas de produto foi popularizada a partir do conceito de produção em massa, proposto originalmente na indústria automobilística por Henry Ford no início do

século 20. A produção em massa consiste em uma produção em larga escala de produtos que são adaptados às necessidades de clientes individuais (DAVIS, 1987). Ela permitiu a produção para um mercado de massa muito mais barato do que a criação de produtos individuais de forma artesanal. Décadas mais tarde, McIlroy (1968) levou a idéia da produção em massa para o desenvolvimento de software, como uma possível solução para a crise de software existente.

O conceito de famílias de software surgiu como um resultado da produção de software em massa e foi introduzido originalmente por Dijkstra (1972) e Parnas (1976). Dijkstra apresentou um modelo de desenvolvimento baseado em famílias onde as diferenças nas decisões de projeto distinguiam os membros da família. Parnas caracterizou famílias como grupos de itens fortemente relacionados por suas similaridades, onde estas são mais importantes que as variações entre os membros da família. Em contraste, o termo populações de produtos é aplicado quando um conjunto de produtos possui diversas similaridades e diversas diferenças (VAN OMMERING, 2002).

O uso do termo característica (*feature*) para direcionar a produção de software em massa foi proposto no começo dos anos 90 por Kang *et al*. (1990). Em seguida, as primeiras conferências sobre o assunto apareceram, tornando Linhas de Produto de Software (LPS) uma nova linha de pesquisa. Na última década, o conceito de linhas de produto alcançou um amplo reconhecimento na indústria de software. Muitas organizações já adotam ou pensam em adotar esse paradigma de reúso (VAN DER LINDEN *et al.*, 2007).

2.2.2 Definição

Linhas de produto de software (LPS) são definidas como um conjunto de sistemas de software que compartilham um conjunto de características comuns e gerenciadas, satisfazendo as necessidades específicas de um segmento particular de mercado ou de negócio, e que são

desenvolvidas de maneira pré-definida a partir de um conjunto comum de artefatos, chamados de ativos centrais (CLEMENTS, NORTHROP, 2002).

O processo de desenvolvimento de software no contexto de linhas de produto foca no desenvolvimento de um número de aplicações e não apenas no desenvolvimento de uma única aplicação. Para atender esse novo paradigma, uma distinção entre engenharia do domínio e da aplicação pode ser introduzida para auxiliar o processo. Desta maneira, a construção do conjunto de ativos reutilizáveis e a identificação de suas variações podem ficar separadas da produção das aplicações da linha de produto.

Uma linha de produto é criada dentro do escopo de um domínio, o qual pode ser um grupo de conhecimento, uma área de especialização ou um conjunto de funcionalidades relacionadas. Quando diferentes aplicações existentes são analisadas numa mesma linha de produto ou domínio de problema, elas são freqüentemente comparadas com base nas suas características (*features*). Segundo Griss (2001), uma *feature* pode ser definida como uma característica que usuários e clientes consideram como sendo de importância na descrição e distinção de membros de uma linha de produtos.

Um ativo central é um artefato ou recurso que é utilizado na produção de mais de um produto em uma LPS. Essas novas aplicações podem ser desenvolvidas seguindo um plano de produção, que define como cada produto é produzido. O plano não só descreve como os ativos centrais podem ser usados para desenvolver um novo produto, como também especifica como adicionar um novo produto à linha de produto existente. Portanto, o plano reúne todos os artefatos reutilizáveis para construir novos produtos, cuja síntese também é uma parte do plano. Na próxima seção, as atividades essenciais de uma LPS serão apresentadas.

2.2.3 Atividades Essenciais de LPS

O SEI (*Software Engineering Institute*) estabeleceu as atividades essenciais de uma abordagem de LPS, através da iniciativa PLP (*Product Line Practice*) (SEI, 2010). Essas atividades são: o Desenvolvimento do Núcleo de Artefatos, o Desenvolvimento do Produto e o Gerenciamento da Linha de Produto. Para a execução dessas atividades, é necessária a definição de áreas de trabalho mais gerenciáveis com conjuntos menores de atividades. Cada área de trabalho possui um plano de trabalho e métricas associadas que permitem acompanhar sua execução e avaliar o sucesso dos trabalhos realizados. Usualmente, as áreas de trabalho permitem produzir artefatos concretos que levam à criação de artefatos centrais utilizados na linha de produto. A Figura 1 apresenta as interações entre essas atividades.

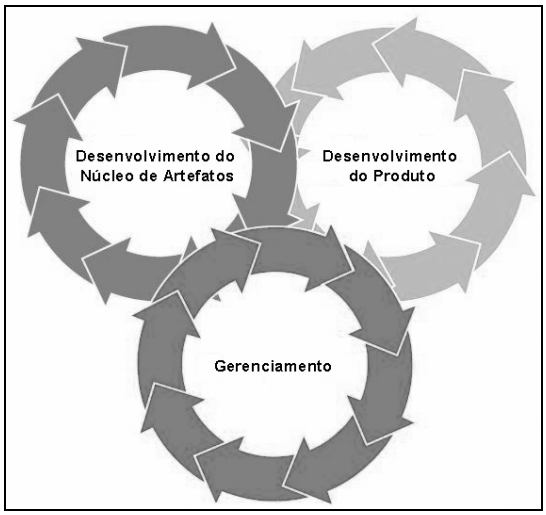


Figura 1 – Atividades Essenciais de LPS, traduzidas de (SEI, 2010)

Com essa figura, o SEI procura demonstrar que todas as atividades são altamente interligadas e iterativas, além de indicar que elas podem ocorrer em qualquer ordem, com uma atividade utilizando a outra. As flechas rotativas indicam que tanto os artefatos podem gerar produtos, como os produtos podem gerar artefatos. Como exemplo disso, pode-se citar a identificação de um novo requisito relevante ao domínio que inicialmente não fazia parte da especificação de requisitos da linha de produto. Desta forma, essa especificação evolui,

incorporando o novo requisito identificado, e o núcleo de artefatos é atualizado. Nas próximas seções, uma breve descrição de cada atividade será apresentada.

2.2.3.1 Desenvolvimento do Núcleo de Artefatos

A atividade de desenvolvimento do núcleo de artefatos, também conhecida como engenharia do domínio, tem como principal objetivo estabelecer uma infra-estrutura central, que será reutilizada pelos produtos gerados a partir da linha de produto. Nessa atividade é realizada uma análise do domínio, a fim de definir o contexto da LPS, a arquitetura e os componentes reutilizáveis, permitindo determinar o plano de produção dos produtos. A Figura 2 apresenta os artefatos de entrada e saída envolvidos nessa atividade.

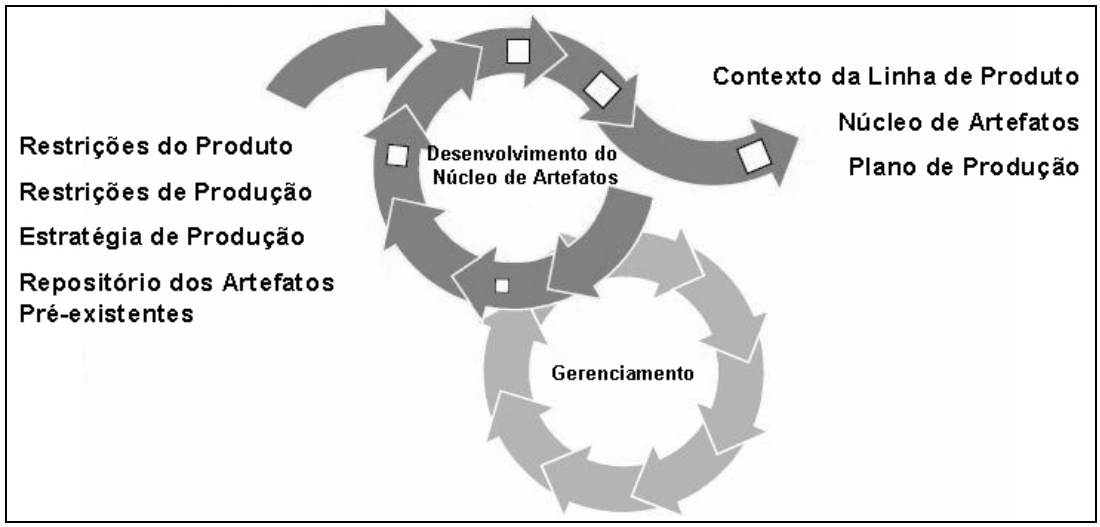


Figura 2 – Desenvolvimento do Núcleo de Artefatos, traduzido de (SEI, 2010)

Essa atividade tem como principais artefatos de entrada:

- Restrições do produto: descrevem as semelhanças e as variações entre produtos que constituirão a LPS, as normas que devem ser seguidas para definir as características dos produtos e os limites de desempenho;
- Restrições de produção: descrevem os componentes COTS (*Commercial-Off-The-Shelf*) (MEYERS, OBERNDORF, 2002) que podem ser usados, as normas que

serão seguidas para a produção dos produtos e os componentes legados que serão reutilizados;

- Estratégia de produção: descreve a estratégia geral da abordagem para produzir os artefatos da linha de produto;
- Repositório dos artefatos pré-existentes: contém os artefatos existentes, como por exemplo: componentes, especificações ou partes legadas do domínio catalogadas para a sua futura reutilização.

Essa atividade tem como principais artefatos de saída:

- Contexto da linha de produto: descreve os produtos que constituirão a linha de produto ou que essa abordagem é capaz de produzir. Essa descrição apresenta as semelhanças e as variações entre os produtos, além de incluir características e operações destes, tais como desempenho e atributos de qualidade;
- Núcleo de artefatos: fornece a base para a produção de produtos a partir da linha de produto. Normalmente, o núcleo de artefatos inclui uma arquitetura a ser reutilizada pelos produtos, bem como os seus componentes;
- Plano de produção: descreve as decisões a serem tomadas para instanciar produtos específicos a partir do núcleo de artefatos da LPS.

2.2.3.2 Desenvolvimento do Produto

A atividade de desenvolvimento do produto, também conhecida como engenharia da aplicação, tem como principal objetivo gerar produtos de uma LPS. Os artefatos de saída da atividade de desenvolvimento do núcleo de artefatos (contexto da linha de produto, núcleo de artefatos e plano de produção dos produtos) servem como entrada para esta atividade. Além desses artefatos, são necessários também os requisitos para um produto específico. A Figura 3 apresenta os artefatos de entrada e saída envolvidos nessa atividade.

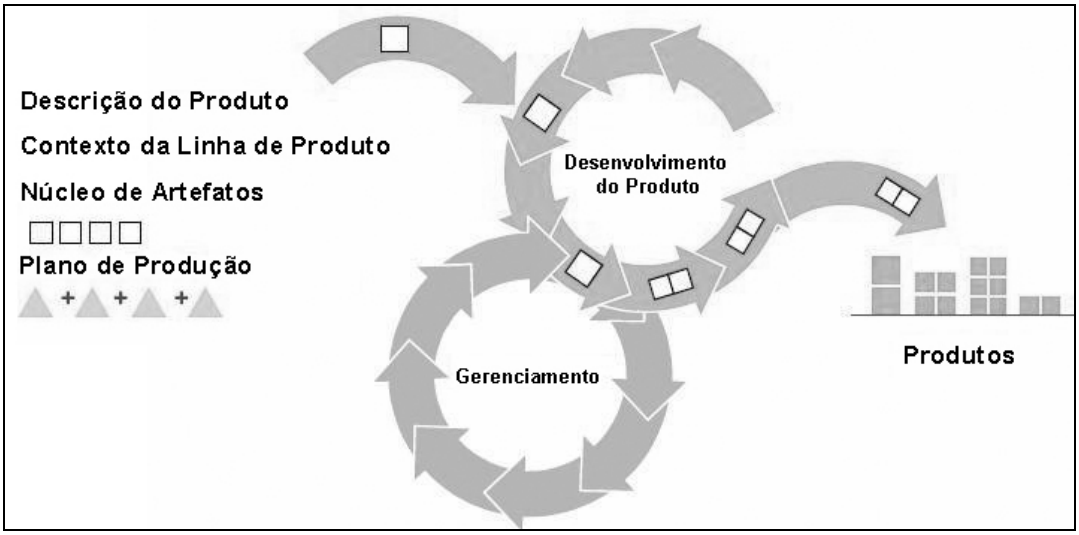


Figura 3 – Desenvolvimento do Produto, traduzido de (SEI, 2010)

As flechas rotativas indicam iteração e relacionamentos intrínsecos entre os produtos da LPS. Pode-se citar, por exemplo, a existência e a disponibilidade de um produto específico que podem afetar os requisitos de um produto subseqüente. Assim, durante a execução dessa atividade é possível identificar requisitos que, inicialmente, não haviam sido especificados e, conseqüentemente, deve-se atualizar o núcleo de artefatos da linha de produto.

2.2.3.3 Gerenciamento da Linha de Produto

A atividade de gerenciamento da linha de produto desempenha um papel crítico no êxito de uma abordagem de LPS. Essa atividade deve garantir que todas as atividades técnicas sejam realizadas de acordo com um planejamento coordenado, a fim de garantir sucesso na construção e manutenção da linha de produto (SEI, 2010). Destacam-se como artefatos mais importantes dessa atividade: o plano de adoção, que descreve o estado desejado da organização e uma estratégia para alcançar tal estado; e o plano de interação, que estabelece como as atividades de desenvolvimento do núcleo de artefatos e desenvolvimento do produto devem interagir a fim de permitir a evolução da LPS e o gerenciamento das semelhanças e das variabilidades de cada artefato da linha de produto.

O gerenciamento da linha de produto pode ser dividido em dois níveis: técnico e organizacional. O nível técnico é responsável por coordenar as atividades de desenvolvimento do núcleo de artefatos e desenvolvimento do produto para garantir que as equipes envolvidas no desenvolvimento sigam os processos definidos para a linha de produto e coletem dados suficientes para acompanhar o progresso desta. O nível organizacional é responsável por garantir os recursos corretos e em quantidades suficientes para as unidades organizacionais.

2.2.4 Benefícios

Conforme discutido na seção anterior, a abordagem de LPS difere das formas de reúso de software tradicionais ao persistir que cada artefato reutilizável seja construído sob o escopo de uma família de software e que tenha um propósito claro neste escopo. Desta forma, podem-se alcançar as vantagens do reúso de maneira ainda melhor por se tratar de uma abordagem sistemática, onde os artefatos são necessários novamente não por acaso, mas sim por causa das semelhanças entre os produtos (JOHN, MUTHIG, 2002). Dentre os principais benefícios da adoção de uma abordagem de LPS, destacam-se (POHL *et al.*, 2005; SEI, 2010):

- Redução do custo de desenvolvimento: quando o núcleo de artefatos é reutilizado para instanciar vários sistemas, o custo total para criar cada sistema é reduzido. Todavia, são necessários altos investimentos para construir o núcleo de artefatos. A Figura 4 apresenta uma comparação do custo acumulado para desenvolver uma LPS com n instâncias e para desenvolver n produtos de software independentes. Quando o número de sistemas é pequeno, o custo inicial de desenvolvimento de uma LPS é mais alto, enquanto que para quantidades maiores de sistemas, ele se torna mais baixo. O ponto destacado na figura é o local onde o custo é o mesmo tanto para desenvolver sistemas da forma tradicional, quanto para desenvolvê-los através de uma abordagem de LPS. Estudos empíricos revelaram que os

investimentos para iniciar uma LPS, normalmente, são retornados após o desenvolvimento de pelo menos três sistemas (CLEMENTS, NORTHROP, 2002). Entretanto, muitos fatores podem influenciar a localização onde o ponto de equilíbrio é alcançado, como o domínio da linha de produto, a experiência da organização e a estratégia de adoção da LPS (McGREGOR *et al*., 2002);

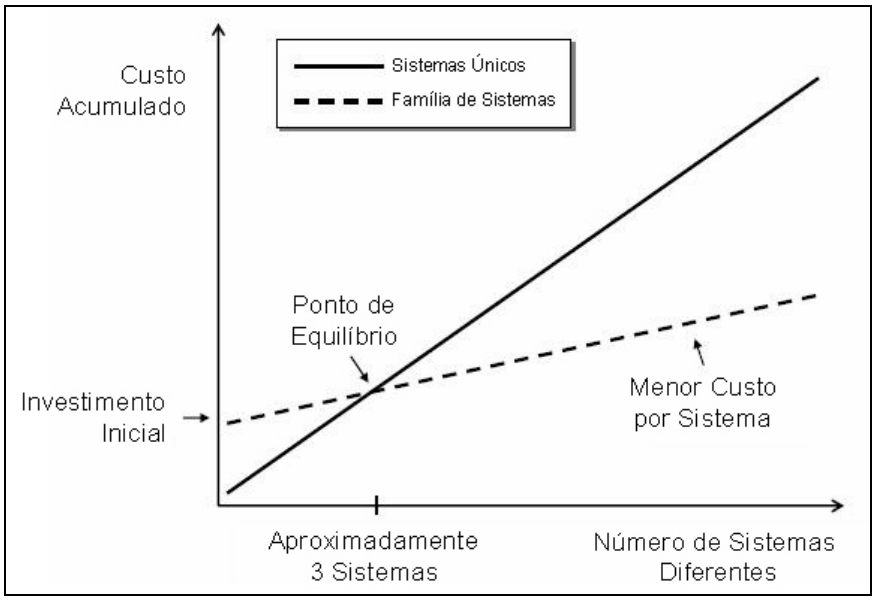


Figura 4 – Custo do Desenvolvimento de n Sistemas Independentes Comparado à Abordagem de LPS, traduzido de (POHL *et al*., 2005)

- Aumento da qualidade do produto: o núcleo de artefatos torna-se cada vez mais revisado e testado à medida que o número de sistemas aumenta numa LPS. Desta forma, novos sistemas usando esses artefatos tendem a ser mais confiáveis e com maior qualidade que os sistemas desenvolvidos a partir do zero, tendo em vista que os artefatos reutilizáveis já tiveram a sua qualidade assegurada em muitos produtos de software;
- Redução do tempo de entrega dos produtos: o esforço necessário para instanciar novos sistemas em uma linha de produtos é muito menor que o esforço necessário para desenvolver um sistema independente. Assim, o tempo necessário para

produzir novos sistemas pode ser reduzido de forma significativa com uma LPS. Entretanto, cabe ressaltar que o tempo de entrega dos primeiros produtos nesse tipo de abordagem é maior, por causa dos investimentos necessários para criar uma linha de produto;

- Diminuição dos riscos do produto: artefatos, como por exemplo, planos de testes, casos de testes e testes, que são desenvolvidos e revisados para os componentes, são reusados em muitos produtos, detectando e corrigindo possíveis falhas;
- Aumento da satisfação do cliente: a abordagem de LPS aumenta a qualidade dos produtos de software, possibilita o cumprimento de prazos para entrega e provê material de treinamento e documentação bem avaliados;
- Utilização mais eficiente dos recursos humanos: a experiência e produtividade aumentam, em razão da funcionalidade comum dos produtos e do processo de produção. Além disso, são gastos menos recursos para treinamento para o uso de ferramentas, processos e componentes do sistema.

2.2.5 Estratégias

Segundo Krueger (2002), a evolução de uma LPS pode ser de três tipos: pró-ativa, reativa ou extrativa. Na abordagem pró-ativa, a organização já visa inicialmente desenvolver a LPS para cobrir todo o escopo de produtos. Por outro lado, na abordagem reativa os produtos da LPS são desenvolvidos apenas sob demanda. Considerada uma abordagem intermediária, a extrativa ocorre quando partes dos artefatos dos produtos existentes são generalizados para uma LPS de tal forma que possam ser reutilizados para outros produtos.

Existem duas estratégias principais para o desenvolvimento de LPS: engenharia avante ou engenharia reversa. A engenharia avante é usada quando não há sistemas previstos para guiar o desenvolvimento da nova linha de produto, ou seja, as funcionalidades comuns podem

ser determinadas antes das funcionalidades variáveis. A engenharia reversa é usada quando há sistemas disponíveis para análise e modelagem, sendo esses sistemas candidatos para modernização e inclusão em um projeto de desenvolvimento de LPS (GOMAA, 2004).

Segundo Bosch (2000), a escolha do tipo de abordagem ou estratégia mais adequada para o desenvolvimento da linha de produto é uma tarefa fundamental, pois uma escolha mal realizada pode acarretar vários problemas, tais como: inconsistência entre os componentes desenvolvidos e as necessidades dos produtos especificadas; componentes mal projetados; dificuldades para a construção das interfaces dos componentes; gerenciamento incorreto do conhecimento; e evolução dos componentes sem modificar as interfaces.

A adoção de uma abordagem de LPS também deve levar em consideração fatores organizacionais (SEI, 2010), tais como: a natureza dos produtos, o mercado ou missão, os objetivos do negócio, a estrutura organizacional, a cultura e as políticas organizacionais, as disciplinas de processo de software, a maturidade dos artefatos legados e a distribuição geográfica da equipe de trabalho. Mesmo considerando esses fatores, uma organização pode encontrar dificuldades ao trabalhar com LPS (BERGEY *et al*., 2001), como por exemplo: a mudança cultural que ocorre devido ao fato dos desenvolvedores estarem acostumados a desenvolver um sistema por vez; o extenso mapeamento do domínio; a construção de uma arquitetura básica; e a definição de um conjunto de componentes.

2.3 Engenharia de Requisitos para Linhas de Produto de Software

2.3.1 Introdução

Requisitos são declarações do que o sistema deve fazer; como deve se comportar; as propriedades que deve apresentar; as qualidades que deve possuir; e as restrições que o

sistema e seu desenvolvimento devem satisfazer (KOTONYA, SOMMERVILLE, 1998). O IEEE (IEEE, 1990) define o termo requisito como:

- Uma condição ou capacidade necessária por um usuário para resolver um problema ou alcançar um objetivo;
- Uma condição ou capacidade que deve ser satisfeita ou possuída por um sistema ou componente do sistema para satisfazer um contrato, padrão, especificação ou outro documento formalmente imposto;
- Uma representação documentada de uma condição ou capacidade como na definição de 1 ou 2.

De acordo com Sommerville e Sawyer (1997), a engenharia de requisitos implica que técnicas sistemáticas e repetíveis devem ser usadas para garantir que os requisitos do software sejam completos, consistentes e relevantes. Para isso, a prática sistemática da engenharia de requisitos requer a realização do seguinte conjunto de atividades:

- Elicitação de Requisitos: é o processo de descobrir, analisar, documentar e entender as necessidades do usuário e as limitações para o sistema;
- Análise de Requisitos: é o processo de refinar as necessidades do usuário e limitações;
- Especificação de Requisitos: é o processo de documentar as necessidades do usuário e as limitações de forma clara e precisa;
- Verificação de Requisitos: é o processo de assegurar que os requisitos do sistema estão completos, corretos, consistentes e claros;
- Gerenciamento de Requisitos: é o processo de planejar, coordenar, e documentar as atividades de engenharia de requisitos (DORFMAN, 1997).

No contexto de LPS, os requisitos definem as aplicações e suas *features*, além das restrições sobre essas aplicações. Segundo Clements e Northrop (2002), os requisitos estão

intimamente ligados com a definição do escopo da linha de produto e evoluem juntos. Eles podem ser divididos em dois grupos: comuns e variáveis. Os requisitos comuns representam os requisitos de toda a linha de produto, enquanto que os requisitos variáveis são específicos para um subconjunto de produtos. As principais diferenças entre a engenharia de requisitos para LPS e a tradicional (sistemas únicos) são (POHL *et al.*, 2005; SEI, 2010):

- Elicitação de Requisitos: essa atividade foca na definição do escopo e deve capturar explicitamente as variações esperadas através da aplicação de técnicas de análise do domínio ao longo do ciclo de vida da linha de produto. A elicitação de requisitos para linha de produto geralmente envolve um número maior de *stakeholders* em comparação com a elicitação de requisitos para um sistema único, podendo incluir especialistas do domínio, especialistas do mercado, dentre outros profissionais;
- Análise de Requisitos: essa atividade tem o escopo da linha de produto como uma de suas entradas e engloba a identificação de variações e similaridades sobre os requisitos elicitados. A análise de requisitos também deve ser capaz de identificar possíveis conflitos quando novos requisitos são adicionados à linha de produto. Além disso, pode ser adicionado um mecanismo que aponte economias de custo na escolha de requisitos comuns e variáveis para uma aplicação específica;
- Especificação de Requisitos: essa atividade inclui não só a documentação de requisitos específicos de uma aplicação, como também a documentação de requisitos comuns entre aplicações. Neste último caso, a especificação dos requisitos deve apresentar partes genéricas que possam ser preenchidas, expandidas ou instanciadas para uma aplicação específica da linha de produto;
- Verificação de Requisitos: essa atividade pode ocorrer em duas fases. Na engenharia do domínio, os requisitos comuns da linha de produto devem ser

verificados. Na engenharia da aplicação, os requisitos específicos do produto devem ser verificados para garantir que eles fazem sentido para o produto;

- Gerenciamento de Requisitos: essa atividade deve incluir uma política que defina como as mudanças nos requisitos da linha de produto são propostas, analisadas e revisadas. Além disso, o gerenciamento de requisitos deve preconizar o acoplamento entre os requisitos da linha de produto e os artefatos do seu núcleo para identificar, por exemplo, o impacto nos artefatos após uma alteração de requisitos.

As atividades de engenharia de requisitos podem mudar de uma organização para outra. Dentre os fatores que contribuem diretamente para a variabilidade desse processo, pode-se destacar: a maturidade técnica da organização, a cultura organizacional e o domínio da aplicação (KOTONYA, SOMMERVILLE, 1998). No contexto de LPS, algumas situações também influenciam a engenharia de requisitos (BAYER *et al*., 1999; BIRK *et al*., 2003):

- Contexto inicial: uma LPS pode ser construída a partir do zero; ou enquanto alguns produtos estão sendo desenvolvidos; ou com base em sistemas existentes. Assim, o contexto inicial interfere na estratégia da elicitação de requisitos;
- Orientação do mercado: uma LPS pode ser desenvolvida para um dado segmento de mercado ou para projetos individuais de clientes. Desta forma, além de interferir na estratégia da elicitação de requisitos, a orientação do mercado interfere também na identificação dos *stakeholders* envolvidos na engenharia de requisitos;
- Tipo do produto: os produtos de LPS podem ser sistemas que agregam funcionalidade a outros sistemas para formar um sistema integrado (*suite*); ou sistemas que possuem funcionalidades comuns, voltados para um segmento de

mercado ou perfil de cliente. Assim, o tipo do produto também interfere na elicitação de requisitos;

- Maturidade do domínio: o domínio pode ser antigo ou novo. Em geral, um domínio antigo pode ser entendido mais facilmente que um domínio novo. Desta forma, o nível de maturidade do domínio interfere na elicitação de requisitos;
- Estabilidade do domínio: o domínio pode ser estável ou instável. Geralmente, um domínio estável demora mais tempo para sofrer alteração do que um domínio instável. Assim, a estabilidade do domínio interfere no gerenciamento de requisitos;
- Arquitetura: a arquitetura define a capacidade e as restrições da plataforma de reúso. Quando surgem novos requisitos, estes precisam ser consistentes com a plataforma. Desta forma, a arquitetura interfere na negociação de requisitos;
- Tamanho da organização: o número de empregados que trabalham com LPS interfere no nível de formalidade do processo de engenharia de requisitos;
- Distribuição geográfica da organização: quando uma LPS está sendo desenvolvida em lugares diferentes, o acesso aos requisitos deve ser permitido de qualquer local. Assim, a distribuição geográfica da organização interfere na estratégia do gerenciamento de requisitos e também na escolha da ferramenta de suporte.

2.3.2 Análise de Similaridades e Variabilidades

Como vimos na seção anterior, a engenharia de requisitos deve levar em consideração a variabilidade da linha de produto. Antes de começar a elaboração das especificações de requisitos, o escopo da linha de produto deve ser definido, a fim de identificar aspectos como, por exemplo: a sua funcionalidade; o grau de similaridade e variabilidade; e o número

desejável de membros da linha de produto (GOMAA, 2004). Segundo Pohl *et al*. (2005), a análise de similaridades e variabilidades tem como objetivo a identificação dos requisitos comuns a todas as aplicações da linha de produtos, assim como dos requisitos que diferem entre as aplicações, determinando as diferenças precisamente.

Uma LPS pode ser desenvolvida quando existe apenas uma aplicação para orientar a sua definição ou com base na análise de sistemas existentes, que devem possuir semelhança suficiente para garantir a formação da base da linha de produto. Neste último caso, o engenheiro de requisitos deve identificar sistemas em potencial que podem ser membros da linha de produtos. De maneira geral, recomenda-se que uma organização com o objetivo de desenvolver uma LPS tenha desenvolvido pelo menos três aplicações similares que pertençam a um mesmo domínio (ROBERTS *et al*., 1997; WEISS, LAI, 1999).

A análise adicional e o esforço necessário para se desenvolver uma LPS precisam ser justificados pela redução total de custos obtida a partir dos vários membros da linha de produtos, cujos custos de desenvolvimento podem ser compartilhados (POHL *et al*., 2005). As aplicações alvo devem possuir uma quantidade de funcionalidades comuns maior que a de diferenças. Entretanto, quando a quantidade de diferenças entre as aplicações supera a de semelhanças, o desenvolvimento da linha de produtos não deve ser prosseguido, pois os sistemas poderão não constituir uma linha de produtos adequada.

Após a definição do conjunto de aplicações alvo da linha de produtos, cada aplicação deve ser analisada para obter-se um conjunto de requisitos. É importante que o domínio da linha de produto esteja relativamente estável a fim de evitar uma volatilidade inesperada dos requisitos. A compreensão dos requisitos do domínio pode ser auxiliada por especialistas no domínio (GOMAA, 2004). Para facilitar a análise de similaridades e variabilidades, cada sistema poderia ser descrito usando uma notação comum, como a UML, mesmo que essa notação não tenha sido originalmente usada para descrever o sistema. No caso de sistemas

legados que possuem pouca ou nenhuma documentação, pode ser necessária a realização de uma engenharia reversa, na qual os requisitos dos sistemas são extraídos a partir da análise do código original usado nesses sistemas (VASCONCELOS, 2007).

A modelagem de objetivos é uma técnica eficaz para a identificação de requisitos na engenharia de requisitos. Os objetivos fornecem a razão para os requisitos, isto é, um requisito existe porque algum objetivo provê uma base para ele (SOMMERVILLE, SAWYER, 1997). No contexto de linhas de produto, os objetivos de negócio direcionam planos de marketing e de produtos, forçando produtos em uma família de produtos a possuírem requisitos comuns e variáveis. Com base nos objetivos de negócio, as características dos produtos podem ser determinadas e, portanto, os objetivos fornecem a fundamentação para os requisitos do domínio. Conseqüentemente, a modelagem de objetivos proporciona um gerenciamento adequado das variações entre aplicações de uma LPS (KIM *et al.*, 2006).

Semmak e Brunet (2006) apresentaram um arcabouço para contribuir na melhoria da elicitação de requisitos através do reúso de modelos de domínio. O modelo de domínio reúne objetivos, que são organizados de forma hierárquica, associados com regras do domínio e materializados em fragmentos conceituais. Cada fragmento conceitual representa uma visão abstrata da especificação que permite a realização de um dado objetivo. Para construir um novo sistema, o engenheiro de requisitos deverá extrair os requisitos do modelo de domínio e depois adaptar os fragmentos conceituais obtidos de acordo com o contexto do sistema. A Figura 5 apresenta um exemplo de hierarquia de objetivos no domínio de livrarias.

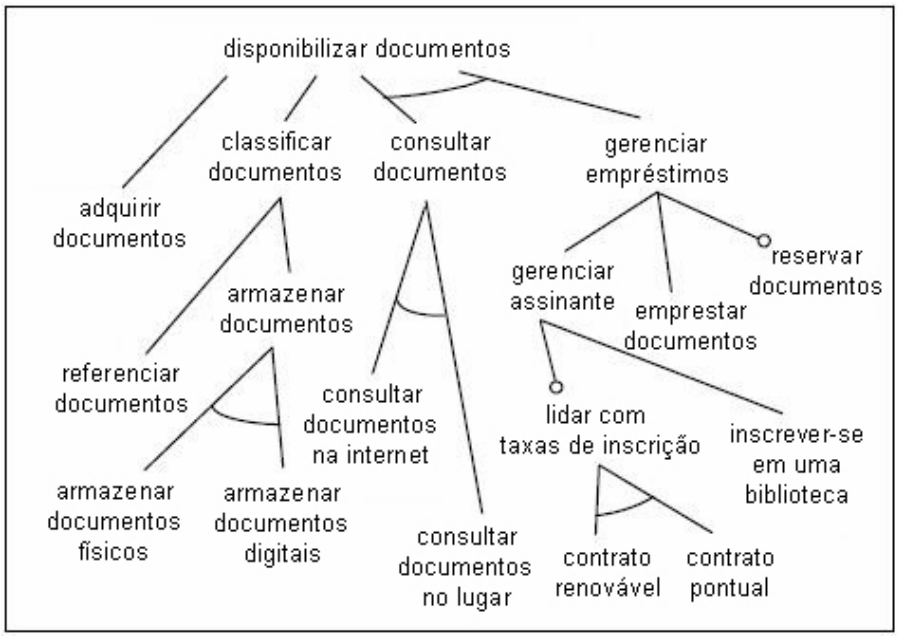


Figura 5 – Uma Hierarquia de Objetivos para o Domínio de Livrarias, traduzida de (SEMMAK, BRUNET, 2006)

Os objetivos formam uma estrutura de árvore, onde os nós folha representam objetivos operacionais para os quais os fragmentos conceituais estão associados, como o objetivo *Emprestar Documentos*, enquanto que os nós de nível superior representam objetivos abstratos, como o objetivo *Gerenciar Empréstimos*. Um arco entre dois objetivos indica que no mínimo um desses deve ser selecionado, como os objetivos *Armazenar Documentos Físicos* e *Armazenar Documentos Digitais*. Os nós opcionais são representados através de um círculo pequeno, como o objetivo *Reservar Documentos*.

Segundo Semmak e Brunet (2006), o conhecimento do domínio pode ser descrito e organizado na forma de fragmentos do conhecimento, onde cada fragmento é uma trinca <objetivo, regra de negócio e fragmento conceitual>. Os objetivos definem um requisito em potencial que os sistemas devem satisfazer; eles expressam o que o usuário final gostaria de fazer. Uma regra de negócio define uma regra do domínio para a qual o objetivo deve obedecer; ela é descrita por um nome e uma expressão. Um fragmento conceitual descreve algumas dependências comportamentais entre os elementos do domínio que interagem para

alcançar um objetivo. A variabilidade neste modelo de domínio pode ocorrer nos três níveis apresentados: objetivos, regras do domínio e fragmentos conceituais.

Como exemplo de fragmento do conhecimento, pode-se citar os seguintes elementos: o objetivo *Adquirir Documentos*, uma regra de negócio que define o número máximo de documentos que podem ser adquiridos e um fragmento conceitual que descreve a realização desse objetivo, no qual o bibliotecário da livraria deve fazer um pedido de documentos aos fornecedores. Desta forma, existe um ator (bibliotecário), alguns objetos do domínio (documento e fornecedor) e alguns eventos (pedido e recebimento de documentos).

Após a identificação dos requisitos de cada aplicação alvo, a análise de similaridades e variabilidades deve ser realizada para obter-se um conjunto de requisitos do domínio dentre todos os membros da linha de produto. A análise não deve abstrair as diferenças entre os sistemas e criar uma arquitetura genérica. Pelo contrário, as diferenças entre as aplicações precisam ser investigadas com vigor. De acordo com Pohl *et al*. (2005), a identificação dos requisitos comuns e variáveis tem início a partir da exploração simultânea dos requisitos de todas as aplicações alvo. Uma maneira simples de realizar essa tarefa é através da utilização de matrizes aplicações x requisitos, onde é possível relacionar as aplicações e seus requisitos. Essas matrizes fornecem uma síntese dos requisitos de mais alto nível das várias aplicações consideradas e, portanto, podem ser usadas na análise de similaridades e variabilidades. O Quadro 1 ilustra um exemplo de matriz aplicações x requisitos sugerida por Pohl *et al*. (2005).

Quadro 1 – Estrutura de uma Matriz Aplicações x Requisitos para Quatro Aplicações, traduzida de (POHL *et al*., 2005)

Requisitos	A1	A2	A3	A4
R1	mandatório	mandatório	mandatório	mandatório
R2	-	-	mandatório	mandatório
R3	mandatório	mandatório	mandatório	-
R4	-	mandatório	-	-
...	...	...	...	...

A coluna mais à esquerda da matriz lista os requisitos das aplicações consideradas, como o requisito R1. As aplicações são listadas na linha mais superior, como a aplicação A1. No corpo da matriz, há marcações que identificam para quais aplicações um dado requisito está presente. Como exemplo, pode-se citar o requisito R1, que está presente em todas as aplicações, sendo, então, um candidato a ser definido como um requisito comum da linha de produto. O requisito R2 não está disponível nas aplicações A1 e A2. Então, ele não é definido como um requisito comum da LPS. O mesmo acontece com os requisitos R3 e R4.

2.3.3 Modelagem de Casos de Uso

A técnica de casos de uso foi introduzida por Jacobson como parte da metodologia de desenvolvimento de software OOSE (*Object-Oriented Software Engineering*) (JACOBSON *et al.*, 1992). Segundo os autores, um caso de uso é uma maneira específica de usar um sistema usando alguma parte da sua funcionalidade, constituindo um curso completo de interações que acontecem entre atores e o sistema. A idéia geral de um caso de uso é representar seqüências de interações entre um sistema e o mundo externo ao sistema, ou seja, descreve maneiras de usar o sistema, mesmo que este ainda não esteja implementado.

Posteriormente, a metodologia OOSE e outras duas metodologias de desenvolvimento de software orientadas a objeto (OO), Booch (BOOCH, 1994) e OMT (RUMBAUGH, 1995), foram unidas e estendidas através de uma linguagem de modelagem unificada, denominada UML (*Unified Modeling Language*). Conseqüentemente, essa linguagem emergiu como um padrão entre desenvolvedores de software OO. Segundo Eriksson e Penker (2000), o interesse por esse padrão causou um grande impacto no mercado, o que motivou muitas empresas a investir em treinamento nessa representação para seus funcionários e em migração de outras linguagens de modelagem para o uso da UML.

De acordo com Svetinovic *et al*. (2007), um caso de uso é um modo particular de um ator utilizar o sistema para atender a um objetivo de uma parte interessada, podendo ser visto como uma coleção de seqüências de interações entre um sistema e seus atores, visando atingir um objetivo. O ator representa um agente externo ao sistema que, de alguma forma, participa de um caso de uso correspondendo a um papel específico que um usuário pode atuar. Como exemplo de atores, pode-se citar o papel que uma pessoa, um dispositivo externo ou mesmo outro sistema desempenha com o software.

Um caso de uso é uma descrição narrativa de um processo do domínio da aplicação, cujo objetivo é auxiliar o engenheiro de requisitos a identificar e descrever a maioria dos requisitos funcionais para um sistema em desenvolvimento. Eles capturam o comportamento pretendido do software sem haver a necessidade de especificar como esse comportamento é implementado, ou seja, descreve o que o sistema faz e não como é feito. Esses artefatos proporcionam uma visão externa do sistema, onde o sistema é visto como uma caixa preta (*black box*) (BOOCH *et al*., 2005).

A abordagem original de casos de uso, proposta por Jacobson e, posteriormente, adotada na UML, não define, de maneira precisa, nenhum formato ou modelo (*template)* específico para descrever o conteúdo de um caso de uso. Na literatura, podem-se encontrar diversas sugestões de modelos de casos de uso (COCKBURN, 2001; SCHNEIDER, WINTERS 2001; KULAK, GUINEY, 2003; HALMANS, POHL, 2003). De maneira geral, eles são definidos por uma linguagem natural estruturada, que pode ser adaptada de acordo com o domínio da aplicação, e descrevem informações tais como: nome e objetivo do caso de uso; atores envolvidos; pré-condições e pós-condições; fluxo de eventos principal e fluxos alternativos.

Além do conceito de casos de uso, Jacobson *et al*. (1992) também introduziram um modo de representá-los graficamente. O diagrama de casos de uso, o qual também é parte

integrante da UML, exibe uma coleção de casos de uso, atores e seus relacionamentos. Os casos de uso são representados por uma elipse com linhas contínuas, incluindo o nome que o diferencia dos demais; um ator é representado por uma figura esquematizada, um “boneco de palitos”; e as associações entre atores e casos de uso são representadas por linhas. A Figura 6 apresenta um exemplo de diagrama de casos de uso.

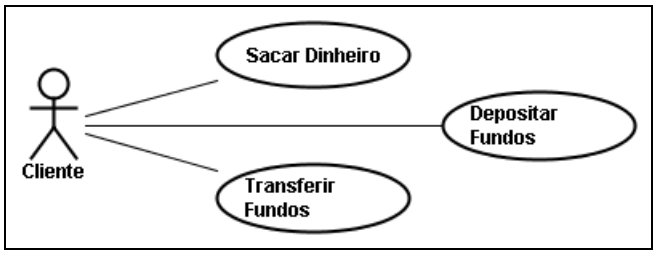


Figura 6 – Exemplo de Diagrama de Casos de Uso

No diagrama, a conexão entre um ator e um caso de uso indica que esses elementos se comunicam entre si, cada um com a possibilidade de enviar e receber mensagens. Na UML, essa conexão é denominada de associação. Por outro lado, os relacionamentos entre casos de uso são aplicados com a finalidade de fatorar comportamentos comuns e variantes, evitando redundâncias e aumentando o reúso. Na UML, esses relacionamentos podem ser de três tipos (BOOCH *et al*., 2005):

- Generalização: é um mecanismo usado para identificar comportamento reutilizáveis. Esse relacionamento indica que um caso de uso pode compartilhar o comportamento definido em um ou mais casos de uso;
- Inclusão: é um mecanismo utilizado para evitar descrever o mesmo fluxo de eventos por várias vezes, fatorando o comportamento comum em um caso de uso próprio. Esse relacionamento indica que um caso de uso incorpora explicitamente o comportamento de outro caso de uso. No diagrama, ele é identificado pelo estereótipo <<include>>. A Figura 7 apresenta um exemplo de relacionamento de inclusão entre casos de uso;

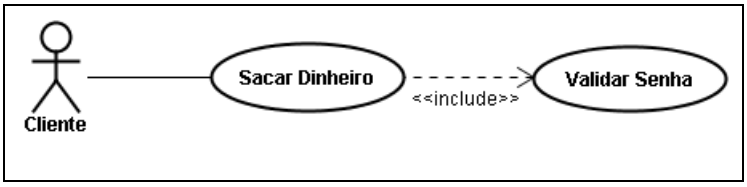


Figura 7 – Exemplo de Relacionamento de Inclusão

- Extensão: é um mecanismo utilizado para modelar um comportamento que ocorre em situações específicas, separando os comportamentos opcionais do comportamento obrigatório de um caso de uso. Esse relacionamento indica que um caso de uso incorpora implicitamente o comportamento de um outro caso de uso em um local especificado indiretamente pelo caso de uso estendido. No diagrama, ele é identificado pelo estereótipo <<extend>>. A Figura 8 apresenta um exemplo de relacionamento de extensão entre casos de uso.

Figura 8 – Exemplo de Relacionamento de Extensão

Nos últimos anos, a abordagem de casos de uso tem recebido muita atenção fora da área de engenharia de requisitos de software, sendo utilizada ao longo de todo o ciclo de vida do desenvolvimento do software. Como exemplo, pode-se citar a utilização dos casos de uso na verificação e validação da arquitetura de sistemas, na realização de testes de sistemas e na reengenharia de software. Esse sucesso tem gerado idéias para utilização dos casos de uso no desenvolvimento de famílias de software. Entretanto, ao especificar casos de uso para linhas de produto, torna-se necessário explicitar as variações das funcionalidades que são tratadas de forma diferente por diversos membros da linha de produtos.

Uma maneira de representar a variabilidade em casos de uso é através da definição de pontos de variação. De acordo com Jacobson *et al*. (1997), um ponto de variação é um local ou região na especificação de requisitos onde uma mudança pode ocorrer. Por exemplo, em descrições textuais, o ponto de variação pode ser um passo, um conjunto de passos ou um fluxo alternativo. A Figura 9 apresenta um exemplo de descrição textual de caso de uso, que foi adaptada para explicitar variabilidades (FANTECHI *et al*., 2004).

Ator Primário: o celular {[V0] série 33}
Objetivo: jogar um jogo no celular {[V0] série 33} e gravar a pontuação.
Pré-condições: a função JOGO foi selecionada no MENU principal.
Cenário Principal de Sucesso:
- O sistema apresenta a lista dos jogos {[V1] disponíveis}.
- O usuário seleciona um jogo.
- O sistema apresenta o logotipo do jogo selecionado.
- O usuário seleciona o nível de dificuldade, seguindo o procedimento {[V2] apropriado} e pressiona SIM.
- O sistema inicia o jogo e executa-o até o seu término.
- O usuário grava a pontuação obtida e {[V3] possivelmente} envia a pontuação para o Club Nokia através de WAP.
- O sistema apresenta a lista dos jogos {[V1] disponíveis}.
- O usuário pressiona NÃO.

V0: alternativo
1. modelo Nokia 3310.
2. modelo Nokia 3330.

V1: opcional
se V0=1 então jogo1 ou jogo2.
senão se V0=2 então jogo1 ou jogo2 ou jogo3.

V2: paramétrico
se V0=1 então procedimento-A:
- pressione Seleção.
- vá até Opções e pressione SIM.
- vá até Nível de Dificuldade e pressione SIM.
- selecione o nível de dificuldade desejado e pressione SIM.

senão se V0=2 então procedimento-B:
- pressione Seleção.
- vá até Nível e pressione SIM.
- selecione o nível de dificuldade desejado e pressione SIM.

V3: paramétrico
se V0=1 então função não disponível.
senão se V0=2 então função disponível.

Figura 9 – Uma Descrição Textual de Caso de Uso para LPS, traduzida de (FANTECHI *et al*., 2004)

Nessa abordagem, os pontos de variação estão representados por *tags* (VO, V1, V2, e V3) e suas explicações estão localizadas no final da descrição. Eles podem ser de três tipos: alternativos, opcionais ou paramétricos. Entretanto, como a informação está contida em uma representação textual, um novo desafio consiste em diferenciar cenários alternativos clássicos, ou seja, provenientes das funcionalidades do domínio, de cenários alternativos originados da variabilidade da linha de produto. Isso implica em descrições de casos de uso sobrecarregadas (*overloaded*) e muito pesadas para o uso prático (TRIGAUX, HEYMANS, 2003).

Quando o caso de uso se torna muito complexo devido à modelagem da variabilidade dentro da sua descrição textual, os pontos de variação podem ser representados através de relacionamentos de inclusão ou extensão (JACOBSON *et al*., 1997). Neste caso, o objetivo é maximizar a extensão e a reutilização dos casos de uso. Casos de uso abstratos são então determinados para identificar padrões comuns em vários casos de uso, os quais podem ser extraídos e reutilizados. Entretanto, as deficiências são reveladas quando se torna difícil a percepção de quais relacionamentos de extensão são oriundos das funcionalidades do domínio e quais são originados da variabilidade (TRIGAUX, HEYMANS, 2003).

2.3.4 Modelagem de *Features*

O termo *feature* é utilizado comumente pela área de marketing para descrever alguma função importante ou distintiva que um determinado produto possui. Na indústria telefônica, por exemplo, um telefone pode possuir várias *features*: espera de chamadas, acesso à internet, mensagem de voz, câmera, GPS, MP3 *Player*, dentre outras (GOMAA, 2004).

Em sistemas de informação, o trabalho de Kang *et al*. (1990) foi quem primeiro propôs utilizar modelos de *features* nesse contexto. Os autores apresentaram um método baseado no conceito de *features* para a atividade de análise do domínio de sistemas (PRIETO-DÍAZ, 1987), denominado FODA (*Feature-Oriented Domain Analysis*). Nesse método, uma *feature*

é definida como um aspecto, qualidade ou característica proeminente ou distintiva de um sistema ou conjunto de sistemas, que é visível ao usuário. A Figura 10 apresenta um exemplo de diagrama de *features* simples usando a notação do método FODA.

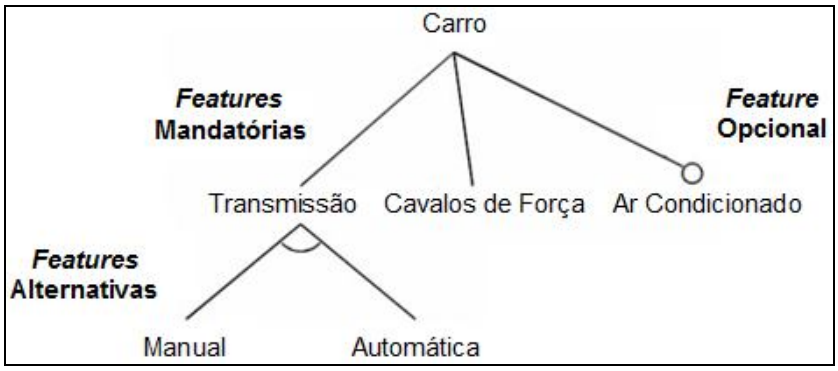


Figura 10 – Exemplo de Diagrama de *Features*, adaptado de (KANG *et al*.,1990)

O diagrama possui uma estrutura de árvore, com nós do tipo AND ou OR. As *features* mais gerais são alocadas no topo da árvore e as *features* mais refinadas são alocadas logo abaixo. Existem três tipos de *features*: mandatórias, opcionais e alternativas. As mandatórias são aquelas disponíveis em todas as aplicações de uma família de sistemas. As opcionais representam as variabilidades existentes na família de sistemas, as quais podem ser incluídas ou não nas aplicações. As alternativas possibilitam uma seleção onde apenas uma *feature* pode ser selecionada de um conjunto de *features*. Segundo esse diagrama, a *feature Carro* possui duas *features* mandatórias: *Transmissão* e *Cavalos de Força*, e uma *feature* opcional: *Ar Condicionado*. A *feature Transmissão* pode ser *Manual* ou *Automática*.

Além disso, Kang *et al*. (1990) definiram dois tipos de relacionamentos entre *features*: *requires* e *excludes*. O relacionamento *requires* indica que uma *feature* depende da outra para ser incluída em uma aplicação. O relacionamento *excludes* indica que ambas *features* não podem ser incluídas numa mesma aplicação, ou seja, apenas em uma delas.

Atualmente, a técnica de modelagem de *features* tem sido amplamente utilizada na análise de similaridades e variabilidades na engenharia de requisitos de LPS. As *features* são

usadas para diferenciar as aplicações de uma linha de produtos, descrevendo e definindo as funcionalidades comuns e variáveis. Na literatura, existem também abordagens de LPS que associaram a modelagem de *features* com a modelagem de casos de uso (GRISS *et al*., 1998; GOMAA, 2004; ERIKSSON *et al*., 2005; BRAGANÇA, MACHADO, 2007), conforme será apresentado na próxima seção. Entretanto, ainda não há um meta-modelo comum e aceito globalmente para *features*, ou seja, cada abordagem possui a sua própria notação.

2.3.5 Relacionamento entre *Features* e Casos de Uso

No desenvolvimento de sistemas únicos, os casos de uso são usados para determinar as funcionalidades de um sistema. Em linhas de produto, os casos de uso também servem para esse propósito. Segundo Griss *et al*. (1998), os casos de uso são direcionados ao uso e visam a obtenção de um claro entendimento dos requisitos funcionais, enquanto que as *features* são direcionadas ao reúso e visam a organização dos resultados da análise de similaridades e variabilidades. Além disso, os modelos de casos de uso fornecem uma completa descrição do que o sistema faz em um determinado domínio, enquanto que os modelos de *features* fornecem uma completa visão das funcionalidades do domínio, as quais podem ser selecionadas para o desenvolvimento de novas aplicações.

Os casos de uso e as *features* podem ser utilizados para complementar um ao outro. Segundo Gomaa (2004), os casos de uso podem ser associados às *features* com base nas suas propriedades de reúso. A estratégia de relacionar casos de uso e *features* não é nova, o trabalho de Griss *et al*. (1998) foi quem primeiro propôs essa abordagem. Nesse trabalho, foi apresentado um método no qual *features* funcionais são obtidas do modelo de casos de uso do domínio. Os autores também propuseram que a estrutura do diagrama de *features* pode ser construída de acordo com a estrutura do diagrama de casos de uso do domínio.

De um modo sucinto, o método proposto por Griss *et al*. (1998) possui dois passos principais. Inicialmente, (a) cria-se uma primeira versão de diagrama somente com as *features* funcionais derivadas do modelo de casos de uso do domínio, onde os nomes dos casos de uso são usados como ponto de partida para os nomes das *features*. Em seguida, (b) decompõem-se cada *feature* identificada de acordo com a estrutura do caso de uso ao qual está associada, baseando-se, por exemplo, nos relacionamentos de inclusão e extensão existentes. Nessa abordagem, os autores assumem que uma *feature* somente pode ser mapeada para um caso de uso, enquanto que um caso de uso pode corresponder a muitas *features*.

Os casos de uso estão relacionados ao alcance de um objetivo de negócio. Segundo Pohl *et al*. (2005), existe uma redundância entre as definições de objetivos e *features*: os objetivos descrevem as intenções das partes interessadas com relação ao sistema em questão, enquanto que as *features* descrevem as características que um sistema oferece aos seus clientes. Na maioria dos casos, objetivos e *features* definem informações semelhantes. Desta forma, para expressar as intenções de um sistema, tanto modelos de objetivos como modelos de *features* podem ser usados. No trabalho de Yu *et al*. (2008), foi apresentado um processo sistemático para a geração de modelos de *features* a partir de modelos de objetivos.

Na literatura existem outros métodos mais recentes que relacionam *features* e casos de uso. No método PLUSS (ERIKSSON *et al*. 2005), o diagrama de *features* é usado como a visão mais alta sobre a linha de produtos. Os autores utilizam esse diagrama como uma ferramenta para estruturar e instanciar um modelo de casos de uso para cada aplicação desenvolvida da família de produtos. Bragança e Machado (2007) ressaltam que a modelagem de *features* necessita de um extensivo conhecimento do domínio, especialmente para *features* funcionais, o qual só é possível após uma modelagem efetiva de tal domínio. Desta forma, assim como Griss *et al*. (1998), um diagrama de *features* é construído a partir do modelo de casos de uso do domínio, onde cada caso de uso é mapeado para uma *feature*.

2.3.6 Método PLUS

PLUS (GOMAA, 2004) trata-se de um método de projeto de software orientado a objetos para LPS que estende os métodos de modelagem baseados na UML para sistemas únicos a fim de modelar explicitamente as similaridades e variabilidades entre as aplicações de uma linha de produtos. Ele pode ser integrado a outros métodos e processos de software tradicionais para dar suporte ao desenvolvimento de linhas de produto. Nesse trabalho, também foi proposto um processo que apresenta uma perspectiva de desenvolvimento de LPS, chamado ESPLEP (*Evolutionary Software Product Line Engineering Process*). Ele incorpora o PLUS e é compatível com o PU (Processo Unificado) (JACOBSON *et al.*, 1999) e o modelo de desenvolvimento em espiral (BOEHM, 1986). Seguindo o arcabouço apresentado em (SEI, 2010), o ESPLEP baseia-se em dois sub-processos principais: a engenharia da linha de produtos (engenharia do domínio) e a engenharia da aplicação. A Figura 11 apresenta uma visão geral desse processo.

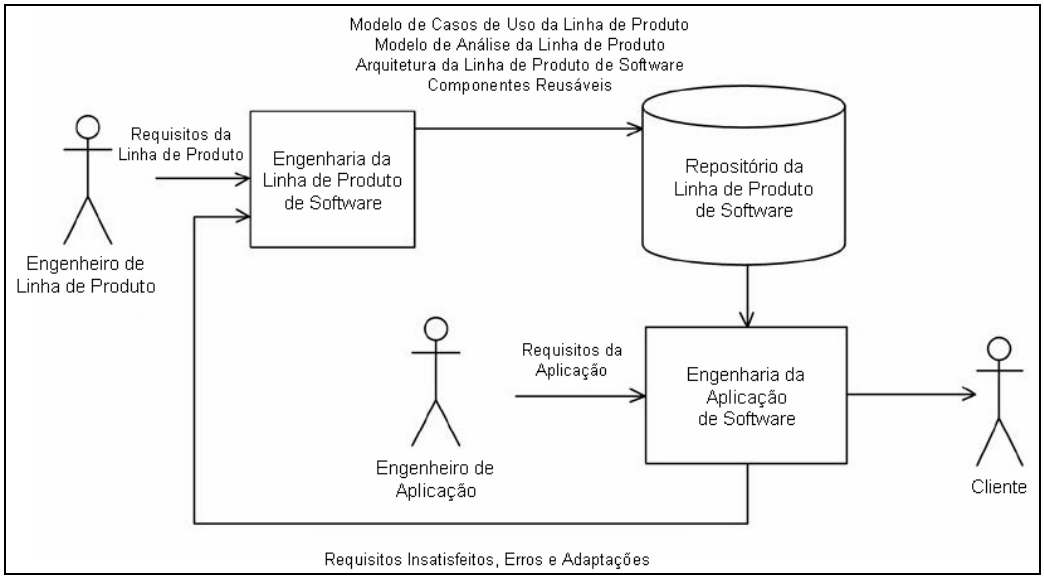


Figura 11 – Processo de Desenvolvimento de LPS do ESPLEP, traduzido de (GOMAA, 2004)

A atividade da engenharia da linha de produtos do ESPLEP produz vários artefatos reusáveis que modelam o domínio da linha de produtos. Ela consiste do desenvolvimento de

um modelo de casos de uso, um modelo de análise e da arquitetura da linha de produtos, além de componentes reusáveis, como o código e testes unitários de componentes, que serão armazenados em um repositório. Na atividade da engenharia da aplicação, uma aplicação individual, que é um membro da LPS, é desenvolvida. Dados os requisitos da aplicação, os modelos contidos no repositório são adaptados para derivar todos os modelos da aplicação. Além disso, com base na arquitetura instanciada para a aplicação e nos componentes selecionados do repositório, obtém-se a aplicação executável.

A Figura 12 apresenta a atividade da engenharia da linha de produtos do ESPLEP. Uma vez que o PLUS é baseado na UML, o autor propõe a adoção da maioria dos diagramas presentes nesse padrão: diagrama de casos de uso, classes, comunicação, seqüência, estados, estrutura compostas e implantação. Além desses diagramas, o método PLUS adota diagramas de *features* inspirados no método FODA (KANG *et al*., 1990), utilizando, entretanto, uma notação baseada no diagramas de classes da UML. A engenharia de linha de produtos é composta por três sub-atividades de modelagem (GOMAA, 2004):

- Modelagem de requisitos: modelagem de casos de uso e *features*;
- Modelagem de análise: modelagem estática; interação dinâmica; máquina de estados dinâmica; e dependências entre *features* e classes;
- Modelagem de projeto: padrões arquiteturais de software e modelagem de projeto de software baseado em componentes.

O ESPLEP é um processo de desenvolvimento de LPS baseado no conceito de casos de uso. Durante a modelagem de requisitos, os requisitos funcionais da linha de produtos são definidos em termos de atores e casos de uso. Durante a modelagem de análise, cada caso de uso da linha de produto passa a ser descrito por objetos que participam no caso de uso e por suas interações. Em seguida, a arquitetura da LPS baseada em componentes é desenvolvida. O

próximo passo é a implementação incremental de componentes, que consiste de um projeto detalhado, sua codificação e de seus testes unitários.

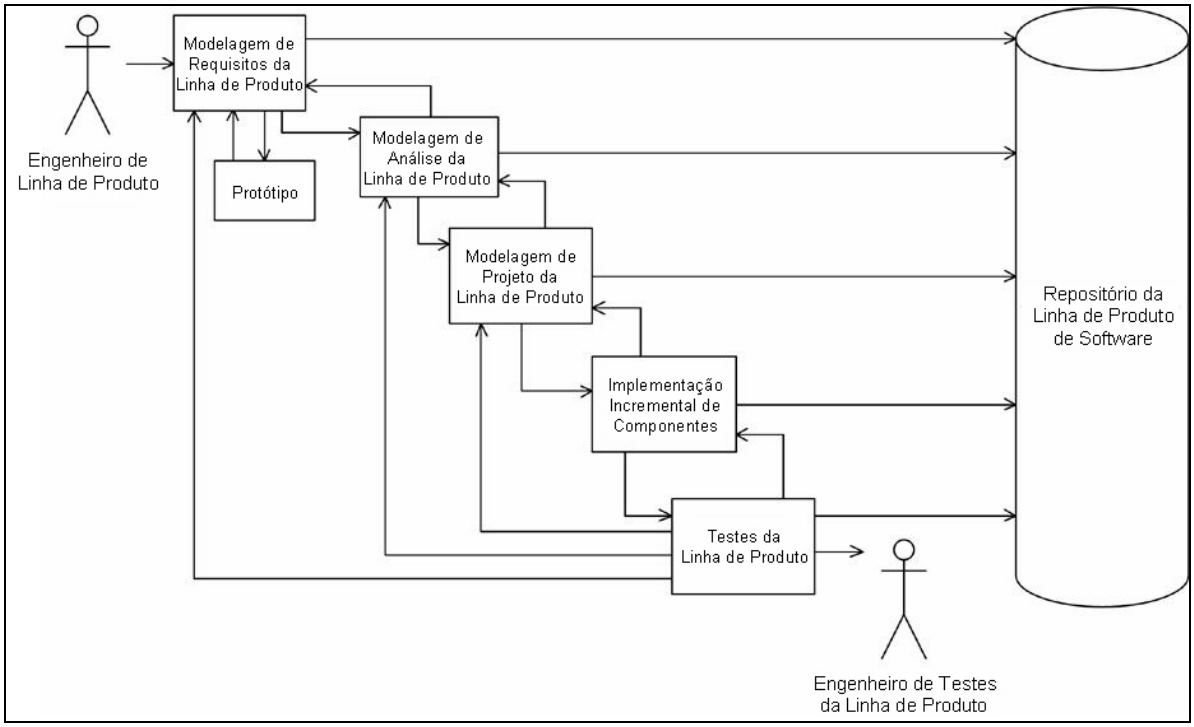


Figura 12 – Engenharia da Linha de Produtos do ESPLEP, traduzida de (GOMAA, 2004)

Dentre as sub-atividades da engenharia de linha de produtos do processo ESPLEP, a modelagem de requisitos é destacada neste trabalho como foco a ser analisado. Nessa sub-atividade, um modelo de requisitos composto por um modelo de casos de uso e um modelo de *features* deve ser desenvolvido. A variabilidade no modelo de casos de uso é tratada através do uso de estereótipos da UML, pontos de variação, relacionamentos de inclusão e extensão entre casos de uso, e generalização ou especialização de atores.

No desenvolvimento tradicional de sistemas, todos os casos de uso e atores existentes no diagrama são necessários. Entretanto, quando uma linha de produto está sendo modelada, somente alguns dos casos de uso e atores são requisitados por um dado membro da LPS. Gomaa propôs três tipos de casos de uso para linhas de produto: obrigatórios (*kernel)*, que estão presentes em todos os membros da linha de produto; opcionais, que estão presentes em

apenas alguns produtos; e alternativos, quando dois ou mais casos de uso não podem estar juntos numa mesma aplicação. Os estereótipos <<kernel>>, <<optional>> e <<alternative>> foram adotados para distinguir esses elementos. As Figuras 13 e 14 apresentam exemplos de diagrama de casos de uso seguindo a notação do método PLUS, elaborados para as LPS de fornos de microondas e de comércio eletrônico, respectivamente (GOMAA, 2004).

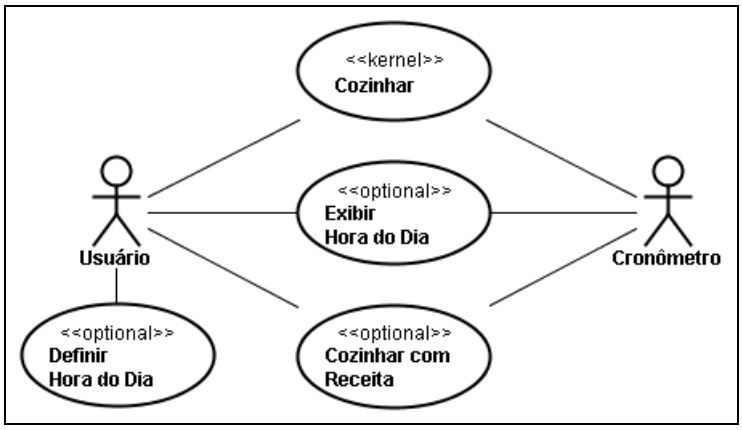


Figura 13 – Diagrama de Casos de Uso da LPS de Forno de Microondas, traduzido de (GOMAA, 2004)

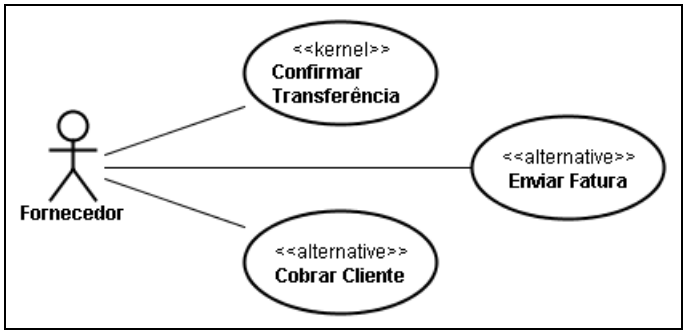


Figura 14 – Trecho do Diagrama de Casos de Uso da LPS de Comércio Eletrônico, traduzido de (GOMAA, 2004)

No primeiro diagrama, o caso de uso *Cozinhar* é *kernel* porque todos os fornos de microondas possuem a funcionalidade de cozinhar. Por outro lado, os casos de uso *Definir Hora do Dia*, *Exibir Hora do Dia* e *Cozinhar com Receita* são opcionais, pois nem todos os membros da linha de produto fornecem a hora do dia e possuem a capacidade de cozinhar com receita. No segundo diagrama, o caso de uso *Confirmar Transferência* é *kernel* porque o

Fornecedor sempre confirma a transferência em todos os sistemas de comércio eletrônico. Por outro lado, os casos de uso *Enviar Fatura* e *Cobrar Cliente* são alternativos, pois essas funcionalidades não estão presentes num mesmo membro da linha de produto.

Além de representar graficamente os casos de uso, Gomaa apresentou também um *template* para auxiliar a elaboração das descrições textuais dos casos de uso. Esse *template* possui uma estrutura similar à apresentada por Cockburn (2001), com a adição de uma seção específica para descrever pontos de variação que possam existir na descrição. Esse conceito de pontos de variação é similar ao apresentado por Jacobson *et al*. (1997), porém o método PLUS possui um escopo restrito, uma vez que somente é usado em modelos de casos de uso. A Figura 15 ilustra um trecho da descrição textual do caso de uso *Cook Food* da LPS de forno de microondas. Nessa descrição, o ponto de variação *Lâmpada* pode ocorrer nas linhas 1, 5, 8 e 9, acrescentando mais funcionalidade ao caso de uso opcionalmente.

Nome do Caso de Uso: Cozinhar.
Categoria de Reúso: *Kernel*.
Resumo: Usuário põe a comida no forno e o forno de microondas cozinha o alimento.
Atores: Usuário (primário), Cronômetro (secundário).
Pré-condição: Forno de Microondas está ocioso.
Descrição:

1. Usuário abre a porta, põe a comida no forno e fecha a porta.
2. Usuário pressiona o botão Tempo de Cozimento.
3. Sistema solicita o tempo de cozimento.
4. Usuário digita o tempo de cozimento no teclado numérico e pressiona Iniciar.
5. Sistema inicia o cozimento da comida.
6. Sistema apresenta o tempo de cozimento restante continuamente.
7. Cronômetro decorre e notifica o sistema.
8. Sistema pára de cozinhar a comida e apresenta uma mensagem de término.
9. Usuário abre a porta, remove a comida do forno e fecha a porta.
10. Sistema limpa o visor.

Nome: Lâmpada.
Tipo de funcionalidade: *Optional*.
Número(s) de linha(s): 1, 5, 8, 9.
Descrição da funcionalidade: Se a opção Lâmpada é selecionada, a lâmpada é ligada durante o cozimento e quando a porta está aberta. A lâmpada é desligada quando a porta está fechada e quando o cozimento pára.

Figura 15 – Trecho de Descrição Textual de Caso de Uso da LPS de Forno de Microondas, adaptado de (GOMAA, 2004)

Gomaa define *feature* como um requisito ou característica que está presente em um ou mais membros da linha de produto. Seguindo o método FODA (KANG *et al*. 1990), foram definidos três tipos de *features*: comuns, opcionais e alternativas. Os estereótipos <<common feature>>, <<optional feature>> e <<alternative feature>> foram adotados para distinguir esses elementos. A Figura 16 apresenta um trecho do diagrama de *features* elaborado para a LPS de forno de microondas, seguindo a notação do método PLUS (GOMAA, 2004).

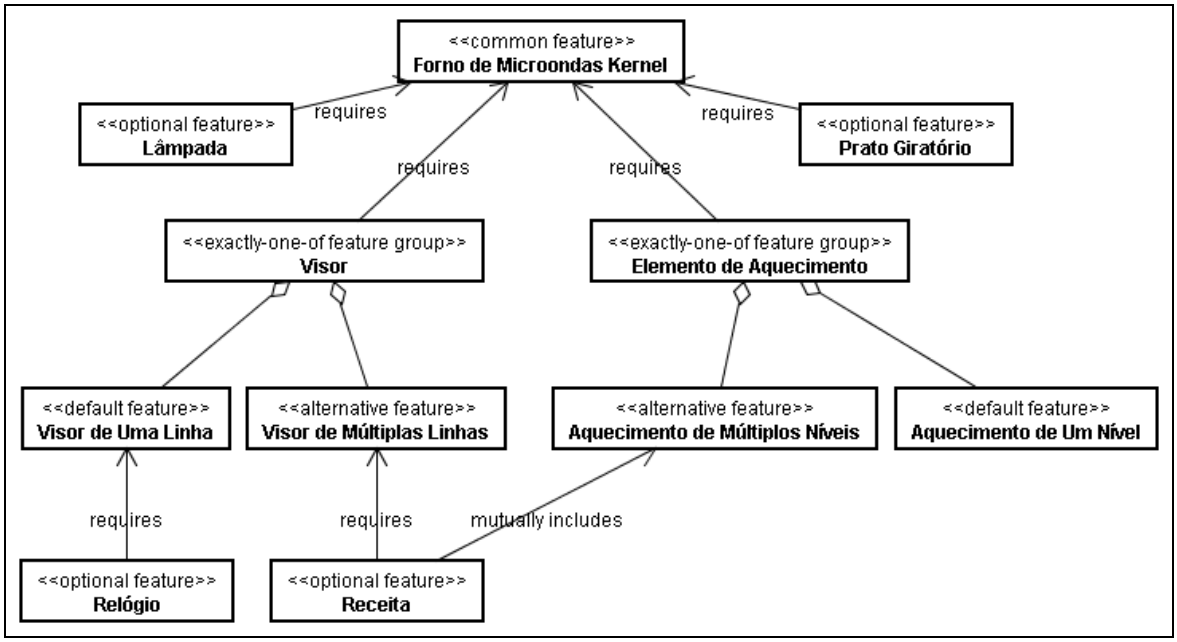


Figura 16 – Trecho do Diagrama de *Features* da LPS de Forno de Microondas, adaptado de (GOMAA, 2004)

Segundo esse diagrama, a linha de produto possui nove *features*, sendo uma comum representando o núcleo da linha de produto (*Forno de Microondas Kernel*), quatro opcionais (*Lâmpada*, *Prato Giratório*, *Relógio* e *Receita*) e quatro alternativas (*Visor de Uma Linha*, *Visor de Múltiplas Linhas*, *Aquecimento de Um Nível*, *Aquecimento de Múltiplos Níveis*, sendo *Visor de Uma Linha* e *Aquecimento de Um Nível* consideradas *default*). Dadas duas *features* A e B, uma relação de dependência entre essas *features* (A → B) pode ser de dois tipos: *requires* (a seleção de A implica na seleção de B), como a relação entre as *features*

Relógio e *Visor de Uma Linha* e *mutually includes* (B só pode ser selecionada em conjunto com A), como a relação entre as *features Receita* e *Aquecimento de Múltiplos Níveis.*

As *features* podem ser agrupadas, determinando uma restrição sobre a forma como elas podem ser usadas por um dado membro da LPS. O método PLUS propôs quatro tipos de grupos de *features*: *zero-or-one-of feature group* (a seleção de uma das *features* do grupo é opcional), *exactly-one-of feature group* (uma *feature* deve ser sempre selecionada do grupo), *at-least-one-of feature group* (uma ou mais *features* podem ser selecionadas do grupo) e *zero-or-more-of feature group* (todas as *features* do grupo podem ser selecionadas). Estereótipos também foram adotados para denotar os grupos de *features* nos diagramas, como é o caso dos grupos de *features Visor* e *Elemento de Aquecimento*, presentes na Figura 16.

O diagrama de *features* é desenvolvido para representar e integrar a variabilidade da linha de produto. Assim como na abordagem original proposta por Griss *et al*. (1998), sua construção é baseada no modelo de casos de uso do domínio. Nesse caso, as dependências entre casos de uso, assim como os pontos de variação existentes, são usados como fontes para representar as dependências entre *features*. No método PLUS, um caso de uso pode estar relacionado a uma ou mais *features*, assim como uma *feature* pode estar relacionada a um ou mais casos de uso. Na primeira situação, uma *feature* corresponde a uma funcionalidade menor que a funcionalidade do caso de uso em questão. Neste caso, ela pode estar relacionada a um ponto de variação ou a um parâmetro existente no caso de uso. Na segunda situação, dois ou mais casos de uso possuem uma estreita ligação entre eles e devem ser selecionados juntos. Neste caso, uma *feature* deve estar relacionada aos casos de uso, que também são denominados de pacotes de casos de uso.

Além disso, Gomaa propôs o uso de tabelas para descrever as relações entre *features*, casos de uso e pontos de variação. O Quadro 2 apresenta uma tabela com parte das relações entre esses três elementos da LPS de fornos de microondas (GOMAA, 2004), onde cada linha

identifica uma relação existente entre uma *feature* e um caso de uso ou entre uma *feature* e um ponto de variação. As colunas descrevem o nome e a categoria da *feature*, o nome do caso de uso do qual a *feature* está associada, a categoria do caso de uso ou o ponto de variação relacionado a este, e se for o caso, o nome do ponto de variação.

Quadro 2 – Parte das Relações entre *Features*, Casos de Uso e Pontos de Variação da LPS de Forno de Microondas, adaptado de (GOMAA, 2004)

Nome da *Feature*	Categoria da *Feature*	Nome do Caso de Uso	Categoria do Caso de Uso / Ponto de Variação (pv)	Nome do Ponto de Variação
Forno de Microondas *Kernel*	*common*	Cozinhar	*kernel*	
Lâmpada	*optional*	Cozinhar	pv	Lâmpada
Prato Giratório	*optional*	Cozinhar	pv	Prato Giratório
Visor de Uma Linha	*default*	Cozinhar	pv	Visor de Uma Linha
Visor de Múltiplas Linhas	*alternative*	Cozinhar	pv	Visor de Múltiplas Linhas
Aquecimento de Um Nível	*default*	Cozinhar	pv	Aquecimento de Um Nível
Aquecimento de Múltiplos Níveis	*alternative*	Cozinhar	pv	Aquecimento de Múltiplos Nível
Relógio	*Optional*	Definir Hora do Dia	*optional*	
		Exibir Hora do Dia	*optional*	
Receita	*optional*	Cozinhar com Receita	*optional*	

De acordo com essa tabela, o núcleo da linha de produto está relacionado ao caso de uso *kernel Cozinhar*, as *features* opcionais *Lâmpada* e *Prato Giratório*, assim como as *features* alternativas *Visor de Uma Linha, Visor de Múltiplas Linhas, Aquecimento de Um Nível* e *Aquecimento de Múltiplos Níveis*, estão relacionadas a pontos de variação que fazem parte desse caso de uso. A *feature* opcional *Relógio* está relacionada a dois casos de uso opcionais: *Definir Hora do Dia* e *Exibir Hora do Dia*. Por último, a *feature* opcional *Receita* está relacionada ao caso de uso opcional *Cozinhar com Receita*.

2.4 Fragmentos de Casos de Uso

2.4.1 Introdução

Dias *et al*. (2008) apresentou uma abordagem visando reduzir o esforço necessário para elaborar especificações de casos de uso. A estratégia dessa proposta consiste em escrever um texto de caso de uso através da composição de um conjunto pré-definido de fragmentos, os quais são conjuntos de interações recorrentes entre o sistema e seus atores necessários para atingir sub-objetivos genéricos. A qualidade dos fragmentos é decorrente da qualidade de cada passo do fragmento e das boas práticas adotadas dos trabalhos anteriores. Tal abordagem pode ser aplicada em qualquer especificação de requisitos de software que tenha sub-objetivos associáveis aos fragmentos.

Esta seção apresenta os principais conceitos relacionados com a abordagem de especificação de requisitos baseada em fragmentos de casos de uso. Primeiramente, serão apresentadas as idéias e definições utilizadas na elaboração da abordagem e, posteriormente, o uso dos fragmentos é explicado e exemplificado.

2.4.2 Definição de Fragmentos de Casos de Uso

Um objetivo de caso de uso pode ser decomposto recursivamente em sub-objetivos, como apresentado por Yu *et al*. (2008). Cada sub-objetivo corresponde a funcionalidades que estão abaixo do nível principal de interesse dos usuários interessados. Em sistemas de informação comerciais, essa decomposição costuma gerar diversos sub-objetivos semelhantes, que podem ser vistos como instâncias de um sub-objetivo mais genérico. Como exemplo de um sub-objetivo genérico, pode-se citar a seleção de um elemento a partir de um conjunto de elementos existentes para que o sistema apresente seus detalhes ao ator.

DIAS (2008) define um fragmento de caso de uso como sendo a estrutura comum de interações entre o sistema e seus atores, incluindo os fluxos alternativos, focados em atingir um sub-objetivo genérico. Esses fragmentos são recorrentemente encontrados durante a tarefa de descrição de requisitos de sistemas de informação e podem ser usados na maior parte das situações que se assemelhem em relação ao sub-objetivo específico. A escrita de um fragmento de caso de uso é composta por três tipos de texto (DIAS, 2008):

- *Templates*: contêm o texto necessário para a descrição das interações requeridas para atingir um sub-objetivo, sendo escrito de acordo com as boas práticas definidas para passos de casos de uso;
- Pontos de personalização: apresentados entre os sinais de ‘<’ e ‘>’. Aparecem no texto dos *templates* e devem ser substituídos por objetos do negócio ou por atores que interagem com o caso de uso em questão durante a personalização do fragmento;
- Pontos de explicação: são referências tanto às propriedades de cada objeto de negócio como à descrição das regras de negócio participantes do caso de uso.

Além disso, algumas partes dos fragmentos podem ser utilizadas opcionalmente. Essa estratégia permite a omissão de trechos desnecessários, o que torna o uso dos fragmentos menos rígido e, conseqüentemente, mais abrangente.

Os passos de caso de uso contidos nos fragmentos de casos de uso apresentados no trabalho de Dias (2008) foram escritos de acordo com a proposta do grupo CREWS (COX, PHALP, 2000) e também seguiram as orientações de qualidade de requisitos apresentadas em (SOMMERVILLE, SAWYER, 1997). O grupo CREWS propôs dois conjuntos de orientações para a escrita de passos individuais: estilo e conteúdo. As orientações de estilo provêem uma estrutura para que a escrita de um passo seja fácil de ser lida e compreendida. As orientações de conteúdo provêem *templates* para descrever passos de forma padronizada, como por

exemplo: <ator> confirma registro do <objeto>. Segundo Sommerville e Sawyer (1997), uma maneira de especificar requisitos com qualidade é através da definição de *templates* para descrever os requisitos e do uso de uma linguagem simples, concisa e fácil de ser lida.

2.4.3 Exemplo de Fragmento de Caso de Uso

Os fragmentos de casos de uso apresentados em DIAS (2008) possuem seis seções: o nome do fragmento, o sub-objetivo genérico, a descrição do fluxo básico, o conjunto dos fluxos alternativos relativos aos passos do fluxo básico, os detalhes das estruturas de dados e das regras de negócio. A Figura 17 apresenta um exemplo de fragmento de caso de uso que pode ser utilizado para refinar o sub-objetivo genérico *Obter Confirmação de Registro*. Esse fragmento pode ser usado na maioria das situações onde se deseja obter uma confirmação após a solicitação de registro de um dado objeto de negócio.

Dentro da descrição do fragmento, os pontos de personalização devem ser instanciados com a correta informação relacionada ao contexto do caso de uso. Por exemplo, os pontos de personalização <ator> e <objeto> devem ser preenchidos com um ator e o próprio objeto de negócio definido no modelo de domínio, respectivamente. No contexto de sistemas de reserva de aluguel de carros, por exemplo, o passo "<ator> solicita o registro do <objeto>" poderia ser instanciado como "Recepcionista solicita o registro da reserva".

Apesar de ter interações pré-definidas, alguns passos e fluxos alternativos podem ser indicados como opcionais. O escritor do caso de uso deve determinar se eles são necessários ou não durante a instanciação do fragmento. Na Figura 17, o fluxo alternativo b é opcional porque pode ser necessário aplicar uma regra de registro antes de finalizar a interação. Além disso, os detalhes de estrutura de dados e de regras também precisam ser personalizados. A lista de propriedades deve descrever ordenadamente todas as informações extraídas do objeto

de negócio que serão apresentadas ao ator. Os detalhes de regras descrevem as restrições necessárias e tarefas que o sistema requer para manipular os objetos.

Nome do Fragmento: Obter Confirmação de Registro
Sub-objetivo: Obter confirmação a respeito do registro de um conjunto de informações no sistema.
Fluxo Básico:
1. <ator> solicita o registro de <objeto>.
2. Sistema apresenta detalhes de <objeto>.
3. <ator> confirma registro de <objeto>.
4. Sistema apresenta os detalhes do recibo de registro.

Fluxos Alternativos:
a) *registro cancelado*
No passo 3 do Fluxo Básico, <ator> decide cancelar o registro.
1. Sistema apresenta a mensagem: "Registro cancelado".
2. Fluxo de eventos retorna ao passo 1 do Fluxo Básico.

b) *regra de registro violada*
No passo 4 do Fluxo Básico, o Sistema não permite o registro de <objeto> devido à violação da <regra_de_registro>.
1. Sistema apresenta a mensagem de acordo com as regras que tenham sido violadas.
2. Fluxo de eventos retorna ao passo 1 do Fluxo Básico.

Fluxos opcionais: b.
Detalhes de Estrutura de Dados:
a) Detalhes de <objeto> = <conjunto_de_propriedades> a ser registrado.
b) Detalhes do recibo de registro = lista de <propriedades>.

Detalhes de Regras:
a) <regra_de_registro> = definição de todas as regras que devem ser aplicadas para validar os parâmetros informados para o registro de um dado objeto.

Regras opcionais: a.

Figura 17 – Fragmento de Caso de Uso *Obter Confirmação de Registro* (DIAS, 2008)

2.4.4 Utilização dos Fragmentos de Casos de Uso

Além de apresentar o conceito de fragmentos de casos de uso, Dias (2008) descreveu como utilizar esses artefatos. O macro-processo possui como entradas: atores, objetivos, casos de uso identificados e definição dos termos do negócio, e como saídas: os casos de uso com suas descrições completas. Essa abordagem é composta por cinco etapas: obter sub-objetivos,

associar a fragmentos, personalizar fragmentos, compor descrição e completar descrição. As atividades de cada etapa são descritas brevemente a seguir.

- Obter Sub-Objetivos: nesta etapa, o objetivo principal do caso de uso deve ser descomposto em seus respectivos sub-objetivos ordenadamente;
- Associar a Fragmentos: nesta etapa, deve-se buscar o fragmento do catálogo que seja mais adequado a cada sub-objetivo identificado;
- Personalizar Fragmentos: nesta etapa, cada fragmento deve ser personalizado através da substituição dos pontos de personalização por objetos do negócio e da expansão dos pontos de explicação;
- Compor Descrição: nesta etapa, os trechos obtidos com cada fragmento devem ser agrupados na seqüência lógica necessária para tal e cada parte dos trechos deve ser agrupada com a sua finalidade;
- Completar Descrição: nesta etapa, a descrição do caso de uso composta por trechos deve ser completada com informações adicionais que não tenham sido contempladas na personalização dos fragmentos.

A Figura 18 apresenta um exemplo de personalização do fragmento de caso de uso *Obter Confirmação de Registro*, ilustrado na Figura 17. O fragmento foi personalizado para atender ao sub-objetivo *Obter Confirmação de Registro do Aluguel*, que faz parte do domínio de Aluguel de Carros. Os pontos <ator> e <objeto> foram substituídos, respectivamente, por um ator participante do caso de uso e por objetos que fazem parte da interação em cada passo dos fluxos básico e alternativos, e estão indicados em negrito. O detalhamento das estruturas de dados e das regras também consideraram o sub-objetivo em questão. As partes opcionais descartadas foram indicadas com um risco.

Fluxo Básico:

1. **Recepcionista** solicita o registro do **aluguel**.
2. Sistema apresenta detalhes do **aluguel**.
3. **Recepcionista** confirma registro do **aluguel**.
4. Sistema apresenta os detalhes do recibo de registro.

Fluxos Alternativos:

a) *registro cancelado*

No passo 3 do Fluxo Básico, **Recepcionista** decide cancelar o registro.

1. Sistema apresenta a mensagem: "Registro cancelado".
2. Fluxo de eventos retorna ao passo 1 do Fluxo Básico.

~~b) *regra de registro violada*~~

~~No passo 4 do Fluxo Básico, o Sistema não permite o registro de <objeto> devido à violação da <regra_de_registro>.~~

~~1. Sistema apresenta a mensagem de acordo com as regras que tenham sido violadas.~~

~~2. Fluxo de eventos retorna ao passo 1 do Fluxo Básico.~~

Fluxos opcionais: b.

Detalhes de Estrutura de Dados:

a) Detalhes do **aluguel** = lista de motorista.nome, responsável.número, data de início, data de término, "carro alugado".placa.

b) Detalhes do recibo de registro = lista de motorista.nome, responsável.número, data de início, data de término, "carro alugado".placa, aluguel.data de início e aluguel.data de término.

Detalhes de Regras:

~~a) <regra_de_registro> = definição de todas as regras que devem ser aplicadas para validar os parâmetros informados para o registro de um dado objeto.~~

Regras opcionais: a.

Figura 18 – Exemplo de Personalização do Fragmento de Caso de Uso *Obter Confirmação de Registro* (DIAS, 2008)

A Figura 19 apresenta um exemplo de fluxo básico que foi elaborado com o auxílio de trechos de fluxos básicos originados de fragmentos. O caso de uso possui como objetivo o registro de um aluguel de carro, cuja descrição foi elaborada com o auxílio dos seguintes fragmentos: *Obter Detalhes de um Objeto, Seleção Única de uma Lista* e o fragmento *Obter Confirmação de Registro*, ilustrado na Figura 18. O primeiro fragmento deu origem aos passos 2 e 3 do fluxo básico; o segundo deu origem aos passos 4-8; o terceiro deu origem aos passos 9-11; o último deu origem aos passos 12-15. Os passos 1 e 16 do fluxo básico não foram contemplados na personalização dos fragmentos e foram adicionados posteriormente.

Fluxo Básico:

1. O caso de uso começa quando a **Recepcionista** solicita o registro de um aluguel.
2. **Recepcionista** informa **licença do motorista**.
3. Sistema apresenta detalhes do **motorista**.
4. **Recepcionista** informa **data de término**.
5. Sistema certifica que **data de término** é valida de acordo com a **regra de data**.
6. Sistema apresenta uma lista de **modelos de carro**.
7. **Recepcionista** seleciona um **modelo de carro**.
8. Sistema apresenta detalhes do **modelo de carro** selecionado.
9. Sistema apresenta uma lista de **cartões de crédito**.
10. Recepcionista seleciona um **cartão de crédito**.
11. Sistema apresenta detalhes do **cartão de crédito** selecionado.
12. **Recepcionista** solicita o registro do **aluguel**.
13. Sistema apresenta detalhes do **aluguel**.
14. **Recepcionista** confirma registro do **aluguel**.
15. Sistema apresenta os detalhes do recibo de registro.
16. O caso de uso termina.

Figura 19 – Fluxo Básico Resultante da Composição dos Trechos de Fluxos Básicos (DIAS, 2008)

2.5 Trabalhos Relacionados

Embora a técnica de casos de uso seja largamente utilizada na indústria, sua utilização em linhas de produto ainda apresenta questões relevantes, principalmente em relação à análise de similaridades e variabilidades voltada para casos de uso e a representação da variabilidade nas especificações textuais desses artefatos.

Algumas abordagens têm sido propostas a fim de apoiar a especificação de casos de uso para linhas de produto baseando-se nos requisitos do domínio identificados. Moon *et al*. (2005) sugeriram um método para especificar requisitos em LPS, onde as similaridades e variabilidades no domínio são explicitamente consideradas. Os autores introduziram várias matrizes para identificar os requisitos comuns e variáveis do domínio e definiram diretrizes para a construção do diagrama de casos de uso da linha de produto a partir desses artefatos. No trabalho de Kim *et al*. (2006), foi proposta uma modelagem de metas e cenários para

identificar os requisitos do domínio e fornecer a razão para eles. Os autores definiram quatro níveis de abstração de requisitos em uma família de produtos (nível de negócio, nível de serviço, nível de interação e nível interno) e ainda introduziram regras para transformar tais requisitos em cenários de casos de uso.

Outros autores adaptaram o *template* de casos de uso sugerido por Cockburn (2001), acrescentando variabilidade a esse formalismo e inserindo-o no contexto de linhas de produto. Esses trabalhos baseiam-se na inclusão de *tags* para indicar quais partes dos requisitos da LPS precisam ser instanciados para uma aplicação específica. John e Muthig (2002) sugeriram o uso de diagramas e textos genéricos para representar a variabilidade nos casos de uso de linhas de produto. Esses autores descreveram fragmentos de texto variáveis e variantes em cenários de casos de uso através de *tags* baseadas em XML (<variant> e </variant>). No trabalho de Fantechi *et al*. (2004), foi proposta uma técnica para representar requisitos de LPS, denominada PLUC (*Product Line Use Cases*). Os autores introduziram três tipos de *tags* (opcionais, alternativas ou paramétricas) dentro das seções de caso de uso (fluxo principal, fluxos alternativos e etc.) com o objetivo de identificar e especificar as variações.

Assim como na abordagem original proposta por Griss *et al*. (1998), alguns autores adotaram a idéia de extrair *features* funcionais a partir do modelo de casos de uso da linha de produto. Eriksson *et al*. (2005) apresentaram um método para a modelagem de casos de uso para LPS chamado PLUSS (*Product Line Use Case Modeling for Systems and Software*). Os autores utilizaram um diagrama de *features*, que adiciona restrições na seleção de *features* alternativas, e uma descrição textual de casos de uso em formato tabular, que representa a variabilidade da linha de produto nos requisitos funcionais. Cada *feature* pode estar associada a um caso de uso, um cenário alternativo, um passo específico ou parâmetros existentes nos passos. No trabalho de Bragança e Machado (2007), foi proposta uma abordagem direcionada a modelos para LPS, onde foi possível automatizar a transformação do modelo de casos de

uso utilizado em um modelo de *features* funcionais da linha de produto. Tal modelo de casos de uso segue uma descrição formal baseada no meta-modelo de atividades da UML, que foi proposta pelos mesmos autores em (BRAGANÇA, MACHADO, 2006).

Por fim, o trabalho proposto por Bonifácio e Borba (2009) apresenta uma abordagem para o gerenciamento da variabilidade em cenários de casos de uso, com o objetivo de separar melhor os interesses entre as linguagens usadas para gerenciar variabilidades e as linguagens usadas para especificar cenários de casos de uso. Para isso, os autores apresentaram um *framework* para modelar a variabilidade como um fenômeno transversal (*crosscutting*), cujo modelo de casos de uso é composto por casos de uso e *aspectual use cases* (JACOBSON, NG, 2004), que são associados a *features* da linha de produto.

Os dois primeiros trabalhos comentados nesta seção (MOON *et al.*, 2005; KIM *et al.*, 2006) focam em auxiliar a elaboração de especificações de requisitos para linhas de produto, mais especificamente, os casos de uso. No primeiro, apesar de deixar clara a forma como a variabilidade da linha de produto foi identificada, não foi descrito como as descrições textuais de casos de uso foram elaboradas. No segundo, os autores explicitaram a origem dos textos dos casos de uso, porém não oferecem suporte à análise de similaridades e variabilidades.

Os dois trabalhos citados em seguida (JOHN, MUTHIG, 2002; FANTECHI *et al.*, 2004) focam na representação da variabilidade em descrições textuais de casos de uso. Entretanto, conforme já mencionado na literatura, é difícil distinguir cenários alternativos clássicos, ou seja, provenientes das funcionalidades do domínio, de cenários alternativos originados da variabilidade (TRIGAUX, HEYMANS, 2003). Além disso, essas abordagens não dão suporte à identificação das partes comuns e variáveis que foram representadas por *tags* e à elaboração das descrições textuais desses artefatos.

Os outros trabalhos comentados nesta seção (ERIKSSON *et al.*, 2005; BRAGANÇA, MACHADO, 2007; BONIFÁCIO, BORBA, 2009), além de apresentarem também sugestões

para a representação da variabilidade em casos de uso, focam na instanciação desses artefatos a partir da seleção das *features* do domínio para uma aplicação específica. Entretanto, essas abordagens também não oferecem suporte à identificação dos requisitos comuns e variáveis do domínio analisado, assim como não fornecem auxílio na elaboração do modelo de *features* e do modelo de casos da linha de produto.

Os trabalhos citados foram importantes fontes de informação, construindo essenciais fundamentos para o desenvolvimento desta dissertação. As lacunas identificadas nos trabalhos apresentados encontram-se justamente na definição de uma abordagem de LPS que auxilie a análise de similaridades e variabilidades do domínio, a elaboração das especificações de casos de uso e agilize a instanciação desses artefatos durante a especificação de requisitos de uma dada aplicação. É nesse contexto que está inserida a abordagem para a elaboração de especificações de casos de uso para LPS que será mostrada no próximo capítulo.

3 ABORDAGEM PARA A ELABORAÇÃO DE ESPECIFICAÇÕES DE CASOS DE USO PARA LPS

3.1 Introdução

Este capítulo apresenta a abordagem para a elaboração de especificações de casos de uso para linhas de produto baseada em fragmentos. A abordagem proposta estende a atividade de modelagem de requisitos contida no processo ESPLEP (GOMAA, 2004), que se insere nas duas atividades essenciais de uma LPS: engenharia do domínio e engenharia da aplicação. Primeiramente, os principais conceitos utilizados na construção da abordagem serão apresentados e, posteriormente, a abordagem será explicada e exemplificada, através de uma linha de produto simples, que corresponde a um domínio de locação de carros que oferece reservas de aluguel de carros através da Internet.

3.2 Conceitos Utilizados

O processo ESPLEP propõe que o modelo de requisitos de uma LPS seja composto por um modelo de *features* e um modelo de casos de uso, que contém um diagrama de casos de uso e especificações textuais. Na abordagem aqui proposta, esse diagrama é construído com base no conjunto de objetivos e seus respectivos sub-objetivos do domínio. Para auxiliar a elaboração das especificações textuais, optou-se por elaborar uma descrição de fragmento de caso de uso para cada sub-objetivo identificado. A Figura 20 ilustra um modelo dos principais conceitos utilizados na construção da abordagem proposta.

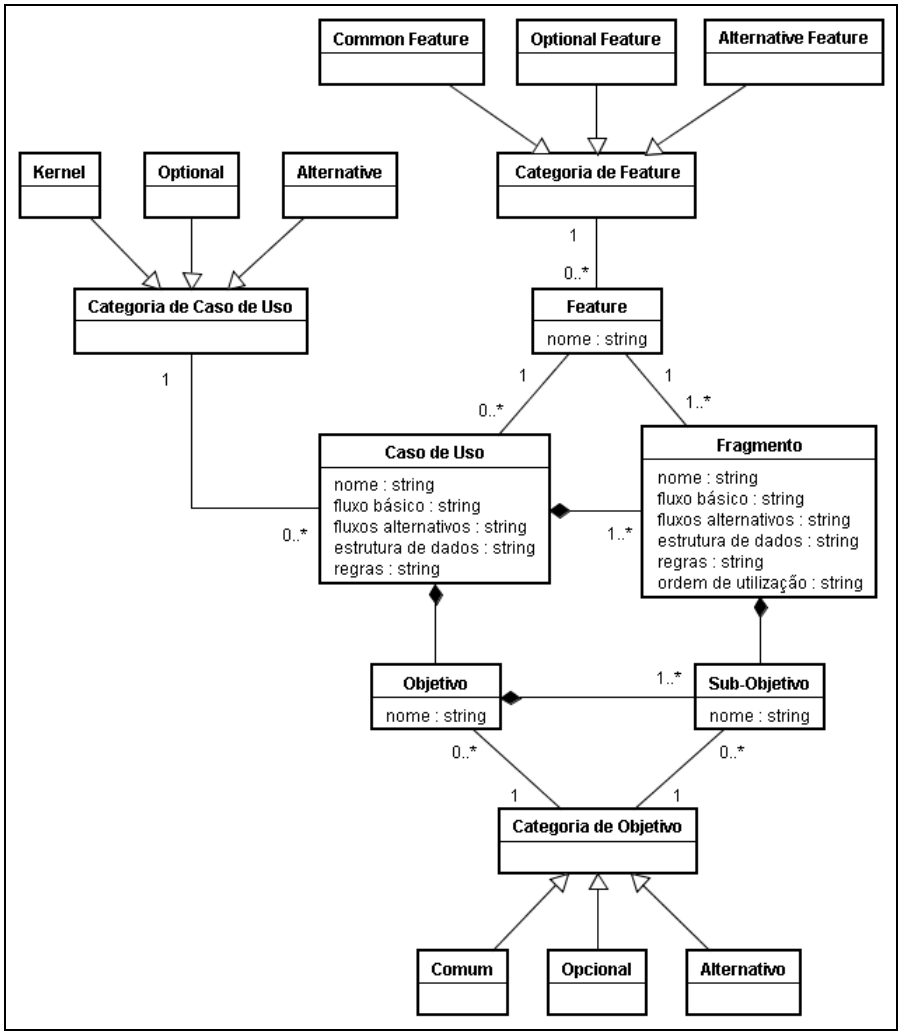


Figura 20 – Modelo dos Principais Conceitos Utilizados na Abordagem Proposta

Um objetivo contém um nome, um conjunto de sub-objetivos e uma categoria, que pode ser comum, opcional ou alternativo. Um objetivo comum é aquele que está presente em todas as aplicações alvo, caso contrário, ele é definido como opcional. Dois ou mais objetivos opcionais são considerados alternativos quando não estão presentes numa mesma aplicação. Os sub-objetivos contêm também um nome e uma categoria como os objetivos. Um caso de uso contém um nome, um objetivo, um fluxo básico, um conjunto de fluxos alternativos, um conjunto de estruturas de dados, um conjunto de regras, um conjunto de fragmentos e uma categoria, que pode ser *kernel*, *optional* ou *alternative*, seguindo o método PLUS (GOMAA, 2004). Os fragmentos contêm também um nome, um fluxo básico, um conjunto de fluxos

alternativos, um conjunto de estruturas de dados e um conjunto de regras como os casos de uso, porém também são compostos por um sub-objetivo e contém a ordem de utilização dos mesmos. Uma *feature* contém um nome, um conjunto de casos de uso, um conjunto de fragmentos e uma categoria, que pode ser *common feature*, *optional feature* e *alternative feature*, também seguindo o trabalho de Gomaa (2004). As multiplicidades utilizadas nas associações de composição indicam: i) "1..*" significa pelo menos um ou muitos associados; ii) "sem cardinalidade" significa um e somente um associado.

3.3 Engenharia do Domínio

No contexto da engenharia do domínio, a abordagem é composta por seis atividades e possui como foco principal a elaboração de especificações de casos de uso da LPS. As entradas dessa etapa são as aplicações alvo da linha de produto. As principais saídas dessa etapa são: o diagrama de *features* do domínio, o diagrama de casos de uso do domínio e um catálogo de fragmentos de casos de uso do domínio. A Figura 21 apresenta o diagrama da engenharia do domínio nesta abordagem, contendo a ordem de execução e as principais atividades dessa etapa, que serão descritas nas próximas seções.

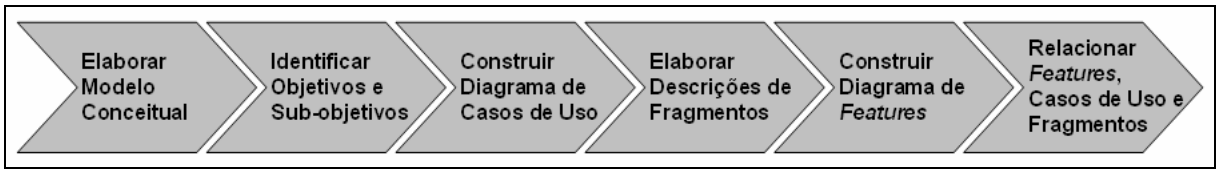


Figura 21 - Atividades da Engenharia do Domínio na Abordagem Proposta

3.3.1 Elaborar Modelo Conceitual do Domínio

- Entrada: aplicações alvo.
- Saída: modelo conceitual do domínio.

A abordagem aplicada na engenharia do domínio começa com a elaboração de um modelo conceitual do domínio comum para as aplicações alvo. Para que o engenheiro de requisitos possa identificar e especificar requisitos do domínio de forma coerente e precisa, os conceitos básicos e termos utilizados no domínio devem ser definidos (MOON *et al.*, 2005). Termos de negócio que são utilizados pelas partes interessadas, bem como no processo de desenvolvimento, podem ser descritos no modelo conceitual do domínio. Nesta abordagem, tais termos são usados na descrição dos casos de uso, especialmente na seção de estrutura de dados. A Figura 22 apresenta um trecho do modelo conceitual do domínio de reserva de aluguel de carros elaborado após a análise de três aplicações pertencentes a este domínio.

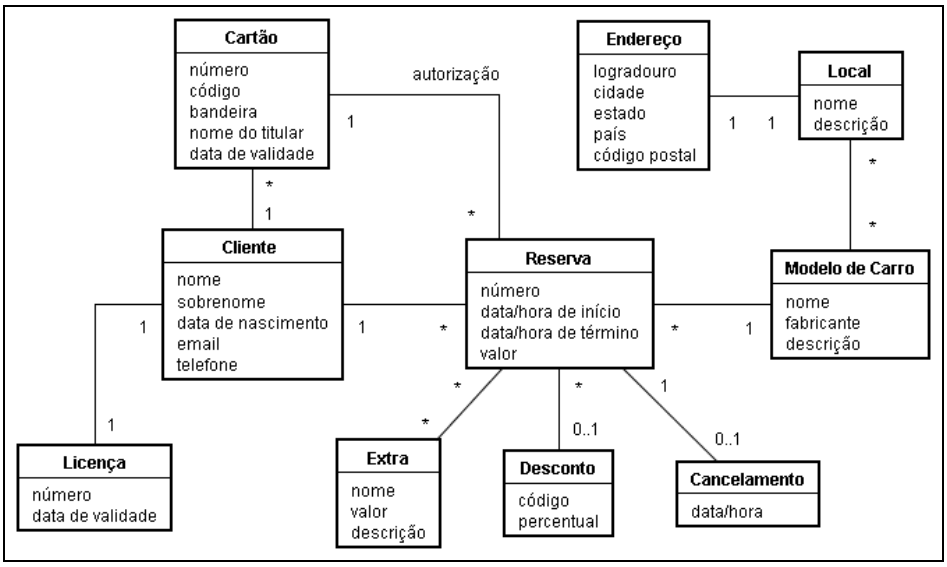


Figura 22 – Trecho do Modelo Conceitual do Domínio de Reserva de Aluguel de Carros

Neste exemplo, a reserva de um carro envolve a seleção de um modelo de carro, que se dá a partir da escolha de um local de retirada e da definição de um período de locação. Tal modelo de carro deve estar disponível para reserva no local indicado e no período informado. Além disso, extras (opcionais e proteções) podem ser adicionados e um desconto sobre o valor da reserva pode ser solicitado. As informações pessoais do cliente também devem ser fornecidas. Para garantir a reserva, uma autorização de pagamento no cartão de crédito do cliente é exigida. Eventualmente, a reserva pode ser cancelada após a sua confirmação.

3.3.2 Identificar Objetivos e Sub-objetivos do Domínio

- Entrada: aplicações alvo.
- Saída: lista de objetivos e sub-objetivos do domínio.

Os casos de uso e os fragmentos de casos de uso do domínio estão relacionados a objetivos e sub-objetivos do domínio, respectivamente. Por isso, a próxima atividade a ser executada na engenharia do domínio consiste na identificação dos objetivos e sub-objetivos comuns e variáveis entre as aplicações alvo da linha de produto.

A análise de cada aplicação pode ser realizada através de simulações de uso das mesmas ou com base na experiência de profissionais. Na abordagem aqui proposta, os objetivos de cada aplicação são identificados através da modelagem de objetivos do ponto de vista do usuário (SEMMAK, BRUNET, 2006). Para essa tarefa, não há restrição quanto à técnica utilizada para realizar a decomposição dos objetivos em sub-objetivos, cabendo ao engenheiro de requisitos da linha de produto escolher o nível de granularidade dos objetivos mais adequado para as suas necessidades. O trabalho de Yu *et al.* (2008) apresenta uma dessas possíveis técnicas para a modelagem de objetivos e sub-objetivos do domínio.

A lista de objetivos e seus respectivos sub-objetivos associados a cada aplicação devem ser agrupadas em uma lista única, retirando-se possíveis redundâncias. Para auxiliar a identificação de quais desses elementos são comuns e quais são variáveis no domínio, uma matriz relacionando todos os objetivos e sub-objetivos identificados com todas as aplicações analisadas pode ser construída (POHL *et al.*, 2005). O Quadro 3 apresenta a matriz aplicações x objetivos elaborada para o domínio de reserva de aluguel de carros do exemplo.

Quadro 3 – Matriz Aplicações x Objetivos da LPS de Reserva de Aluguel de Carros

Objetivos/Sub-objetivos		Categoria	A1	A2	A3
1	**Reservar Carro**	**comum**	**x**	**x**	**x**
1.1	Selecionar Modelo de Carro	comum	x	x	x
1.2	Adicionar Extras	opcional	x	x	
1.3	Aplicar Desconto	opcional		x	x
1.4	Cadastrar Dados Pessoais	comum	x	x	x
1.5	Autorizar Pagamento em Cartão	opcional			x
1.6	Obter Confirmação da Reserva	comum	x	x	x
1.7	Enviar Comprovante	alternativo	x		
1.8	Imprimir Comprovante	alternativo			x
2	**Alterar Reserva**	**comum**	**x**	**x**	**x**
2.1	Consultar Reserva	comum	x	x	x
2.2	Solicitar Alteração da Reserva	comum	x	x	x
2.3	Obter Confirmação da Alteração	comum	x	x	x
3	**Cancelar Reserva**	**opcional**	**x**	**x**	
3.1	Consultar Reserva	comum	x	x	x
3.2	Solicitar Cancelamento da Reserva	opcional	x	x	

A matriz apresenta no lado esquerdo a lista de objetivos e seus respectivos sub-objetivos que foram extraídos a partir da análise de três aplicações reais do domínio, as quais foram aqui denominadas A1, A2 e A3 e estão situadas no topo. No corpo da matriz, as células marcadas com ‘x’ identificam em quais aplicações um dado objetivo ou sub-objetivo está presente. Há também uma coluna que descreve a categoria de cada objetivo ou sub-objetivo listado (comum, opcional ou alternativo). Neste exemplo, o objetivo *Reservar Carro* está presente em todas as aplicações sendo, então, definido como um objetivo comum. Por outro lado, o sub-objetivo *Adicionar Extras* não está presente na aplicação A3 e, portanto, ele é considerado um sub-objetivo opcional. Após a confirmação da reserva, um comprovante pode ser gerado e isto pode ser realizado de duas formas: envio ou impressão do comprovante. Entretanto, os sub-objetivos *Enviar Comprovante* e *Imprimir Comprovante* não estão presentes numa mesma aplicação sendo, então, definidos como alternativos.

3.3.3 Construir Diagrama de Casos de Uso do Domínio

- Entrada: lista de objetivos e sub-objetivos do domínio.
- Saída: diagrama de casos de uso do domínio.

Após a análise de similaridades e variabilidades, os objetivos e sub-objetivos do domínio devem ser analisados para extrair o diagrama de casos de uso do domínio. A construção desse artefato começa com a elaboração de um diagrama inicial, onde se adiciona um caso de uso para cada objetivo identificado. A Figura 23 apresenta o diagrama de casos de uso construído para o domínio de reserva de aluguel de carros do exemplo. Neste caso, os objetivos *Reservar Carro*, *Alterar Reserva* e *Cancelar Reserva* deram origem aos casos de uso presentes no diagrama.

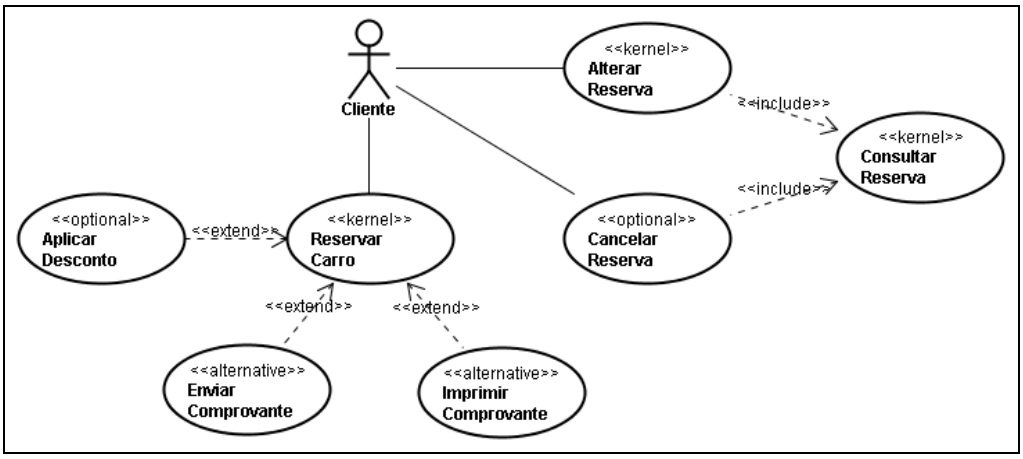


Figura 23 – Diagrama de Casos de Uso da LPS de Reserva de Aluguel de Carros

O diagrama inicialmente obtido pode ser refinado adicionando-se relacionamentos de inclusão ou extensão entre os casos de uso. Caso um sub-objetivo apareça em diferentes casos de uso, ele é considerado comum a esses elementos, dando origem, nesse caso, a um caso de uso relacionado com o caso de uso original através de um relacionamento de inclusão. No exemplo, o sub-objetivo *Consultar Reserva* tornou-se um caso de uso do tipo *include* relacionado com os casos de uso *Alterar Reserva* e *Cancelar Reserva*. Caso um sub-objetivo não seja necessário para o alcance de um dado objetivo, ele pode ser separado como um novo

caso de uso relacionado com o caso de uso original através de um relacionamento de extensão. No exemplo, os sub-objetivos *Aplicar Desconto*, *Enviar Comprovante* e *Imprimir Comprovante* tornaram-se três casos de uso do tipo *extend* relacionados com o caso de uso *Reservar Carro*.

Por último, as categorias dos casos de uso dependem das categorias identificadas para os objetivos e sub-objetivos do domínio. Casos de uso originados de objetivos ou sub-objetivos comuns, opcionais e alternativos são chamados de caso de uso *kernel*, *optional* e *alternative*, respectivamente. Seguindo o método PLUS (GOMAA, 2004), os estereótipos <<kernel>>, <<optional>> e <<alternative>> foram adotados para distinguir esses elementos.

3.3.4 Elaborar Descrições de Fragmentos de Casos de Uso

- Entrada: aplicações alvo, modelo conceitual e lista de sub-objetivos do domínio.
- Saída: catálogo de fragmentos de casos de uso do domínio.

Os fragmentos de casos de uso propostos em (DIAS *et al*. 2008) não pertencem a nenhum domínio específico, pois estão relacionados a sub-objetivos genéricos. Entretanto, uma vez que esses fragmentos são construídos para atender diversos domínios, eles podem não ser tão ricos em detalhes ou não estar tão alinhados a um domínio específico. Na abordagem aqui proposta, fragmentos de casos de uso específicos de um domínio são elaborados e utilizados para auxiliar a especificação textual de casos de uso de novas aplicações num contexto de linhas de produto. Esses artefatos podem ser considerados como a materialização do conceito de fragmentos conceituais proposto por Semmak e Brunet (2006).

Desta forma, para cada sub-objetivo identificado do domínio, uma descrição detalhada na forma de especificação de fragmento de caso de uso deve ser elaborada, baseando-se nas simulações de uso das aplicações alvo. No exemplo, doze fragmentos de casos de uso foram elaborados, como os fragmentos *Selecionar Modelo de Carro*, *Aplicar Desconto* e *Enviar*

Comprovante. O modelo conceitual do domínio construído na primeira atividade pode ser personalizado de acordo com a aplicação em desenvolvimento e utilizado para identificar os detalhes de estrutura de dados. Além disso, detalhes de regras de negócio também podem ser adicionados. A Figura 24 apresenta a descrição de um fragmento de caso de uso específico resultante do refinamento do sub-objetivo *Obter Confirmação da Reserva*, que só pode ser utilizado após o uso de um dos seguintes fragmentos: *Cadastrar Dados Pessoais*, *Adicionar Extras* e *Autorizar Pagamento em Cartão*.

Nome do Fragmento: Obter Confirmação da Reserva
Sub-objetivo: Obter confirmação a respeito do registro da reserva.
Ordem de Utilização: Após Cadastrar Dados Pessoais ou Adicionar Extras ou Autorizar Pagamento em Cartão.
Fluxo Básico:
1. <cliente> solicita o registro da reserva.
2. Sistema apresenta detalhes da reserva.
3. <cliente> confirma registro da reserva.
4. Sistema registra a reserva.
5. Sistema apresenta recibo de registro.

Fluxos Alternativos:
a) *registro da reserva cancelado*
No passo 3 do Fluxo Básico, <cliente> decide cancelar o registro da reserva.
1. Sistema apresenta a mensagem: “Registro da reserva cancelado”.
2. Fluxo de eventos retorna ao passo 1 do Fluxo Básico.

Detalhes de Estrutura de Dados:
a) Detalhes da reserva = <reserva.data/hora de início, reserva.data/hora de término, reserva.valor, cliente.nome, cliente.sobrenome, local.nome, modelo de carro.nome, lista de extra.nome e extra.valor, desconto.código, desconto.percentual, cartão.número e cartão.bandeira>.
b) Detalhes do recibo de registro = <reserva.número e “detalhes da reserva”>.

Detalhes de Regras:
Não há.

Figura 24 – Descrição do Fragmento de Caso de Uso *Obter Confirmação da Reserva*

3.3.5 Construir Diagrama de *Features* do Domínio

- Entrada: diagrama de casos de uso do domínio e catálogo de fragmentos de casos de uso do domínio.

- Saída: diagrama de *features* do domínio.

No processo ESPLEP apresentado em (GOMAA, 2004), a modelagem de casos de uso antecede a modelagem de *features.* Assim como na abordagem original proposta por Griss *et al*. (1998), *features* funcionais são extraídos a partir do modelo de casos de uso do domínio. Seguindo esse último trabalho, na abordagem aqui proposta, o diagrama de *features* do domínio é criado de acordo com a estrutura do modelo de casos de uso do domínio, que é formado pelo diagrama de casos de uso e pelo catálogo de fragmentos. Assim, o diagrama de *features* do domínio é construído para representar o maior nível de abstração para o desenvolvimento do domínio.

Segundo Gomaa (2004), existem duas estratégias para elaborar o diagrama de *features* a partir dos requisitos identificados. A primeira é agrupar todos os requisitos comuns em uma única *feature* e explicitar os requisitos variáveis em *features* separadas. A outra estratégia é explicitar todos os requisitos em *features* separadamente. Na presente abordagem adotou-se a primeira estratégia, pois os requisitos comuns da linha de produto são bem definidos devido à análise de similaridades e variabilidades feita posteriormente. Neste caso, a construção do diagrama de *features* do domínio começa com a adição de uma *feature* comum que representa o núcleo da linha de produto e está relacionada com todos os casos de uso *kernel* e seus respectivos fragmentos.

A elaboração do diagrama segue com a adição de *features* opcionais e alternativas para cada caso de uso e/ou fragmento de caso de uso opcional e alternativo, respectivamente. Após esta etapa, as *features* podem ser agrupadas conforme as funcionalidades da qual fazem parte. Seguindo o método PLUS (GOMAA, 2004), os estereótipos <<common feature>>, <<optional feature>> e <<alternative feature>> foram adotados para distinguir essas três categorias de *features* e os estereótipos <<zero-or-one feature group>>, <<exactly-one-of feature group>>, <<at-least-one-of feature group>> e <<zero-or-more feature group>> foram

adotados para distinguir os quatro tipos de grupo de *features*. A Figura 25 apresenta o diagrama de *features* construído para o domínio de reserva de aluguel de carros.

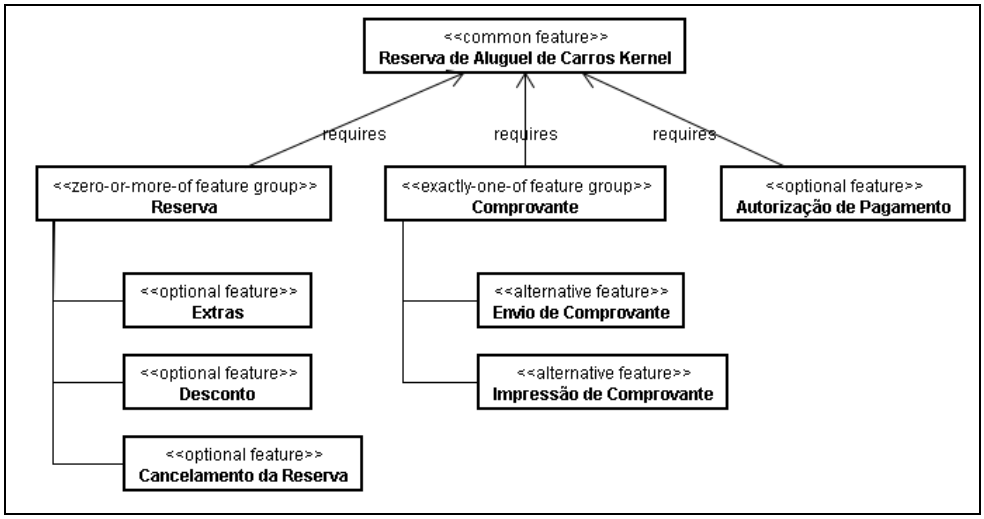


Figura 25 – Diagrama de *Features* da LPS de Reserva de Aluguel de Carros

De acordo com esse diagrama, a linha de produto possui sete *features*, sendo uma comum representando o núcleo da linha de produto (*Reserva de Aluguel de Carros Kernel*), quatro opcionais (*Extras*, *Desconto*, *Cancelamento da Reserva* e *Autorização de Pagamento*) e duas alternativas (*Envio de Comprovante* e *Impressão de Comprovante*). Há também dois grupos de *features*: *Reserva* e *Comprovante*. O grupo *Reserva* reúne as *features Extras*, *Desconto* e *Autorização de Pagamento*, que podem ser selecionadas sem restrição. O grupo *Comprovante* reúne as *features Envio de Comprovante* e *Impressão de Comprovante*, onde apenas uma dessas deve ser selecionada para uma nova aplicação.

3.3.6 Relacionar *Features*, Casos de Uso e Fragmentos

- Entrada: diagrama de *features* do domínio, diagrama de casos de uso do domínio e catálogo de fragmentos de casos de uso do domínio.
- Saída: relações entre *features*, casos de uso e fragmentos do domínio.

Segundo Gomaa (2004), as *features* da linha de produto podem ser concisamente descritas em uma representação tabular. Originalmente, a tabela descreve as relações entre *features*, casos de uso e pontos de variação e, portanto, facilita a instanciação do modelo de casos de uso a partir da seleção das *features* do domínio na engenharia da aplicação. Na abordagem aqui proposta, os pontos de variação nas descrições textuais de casos de uso são tratados através da seleção e personalização de fragmentos. Neste caso, a tabela relacionará fragmentos de casos de uso ao invés de pontos de variação. O Quadro 4 apresenta a tabela com as relações entre *features*, casos de uso e fragmentos elaborada para o domínio de reserva de aluguel de carros.

Quadro 4 – Relações entre *Features*, Casos de Uso e Fragmentos da LPS de Reserva de Aluguel de Carros

Feature		**Caso de Uso**		**Fragmento**
Nome	**Categoria**	**Nome**	**Categoria/ Fragmento**	**Nome**
Reserva de Carro Kernel	Common	Reservar Carro	Kernel	
			Fragmento	Selecionar Modelo de Carro
			Fragmento	Cadastrar Dados Pessoais
			Fragmento	Obter Confirmação da Reserva
		Consultar Reserva	Kernel	
			Fragmento	Consultar Reserva
		Alterar Reserva	Kernel	
			Fragmento	Solicitar Alteração da Reserva
			Fragmento	Obter Confirmação da Alteração
Extras	Optional	Reservar Carro	Fragmento	Adicionar Extras
Desconto	Optional	Aplicar Desconto	Optional	
			Fragmento	Aplicar Desconto
Cancelamento da Reserva	Optional	Cancelar Reserva	Optional	
			Fragmento	Solicitar Cancelamento da Reserva
Envio de Comprovante	Alternative	Enviar Comprovante	Alternative	
			Fragmento	Enviar Comprovante
Impressão de Comprovante	Alternative	Imprimir Comprovante	Alternative	
			Fragmento	Imprimir Comprovante
Autorização de Pagamento	Optional	Reservar Carro	Fragmento	Autorizar Pagamento em Cartão

De acordo com essa tabela, o núcleo está relacionado a três casos de uso *kernel*, como o caso de uso *Reservar Carro*, e a seis fragmentos, como o fragmento *Consultar Reserva*. Algumas *features* opcionais estão relacionadas a apenas um fragmento, como é o caso da *feature Extras* que está relacionada ao fragmento *Adicionar Extras*. Outras *features* opcionais estão relacionadas a um caso de uso e seus respectivos fragmentos, como é o caso da *feature Desconto* que está relacionada ao caso de uso *Aplicar Desconto* e ao fragmento de mesmo nome. Da mesma forma, as *features* alternativas *Envio de Comprovante* e *Impressão de Comprovante* estão relacionadas aos casos de uso *Enviar Comprovante* e *Imprimir Comprovante*, além dos seus respectivos fragmentos de mesmo nome, respectivamente.

3.4 Engenharia da Aplicação

No contexto da engenharia da aplicação, a abordagem é composta por quatro atividades e possui como foco principal a elaboração de especificações de casos de uso para uma dada aplicação da linha de produto. As entradas dessa etapa são as características da aplicação e os principais artefatos obtidos na engenharia do domínio. As principais saídas dessa etapa são: o diagrama de *features* da aplicação, o diagrama de casos de uso da aplicação e a descrição textual completa dos casos de uso da aplicação. A Figura 26 apresenta o diagrama da engenharia da aplicação nesta abordagem, contendo a ordem de execução e as principais atividades dessa etapa, que serão descritas nas próximas seções.

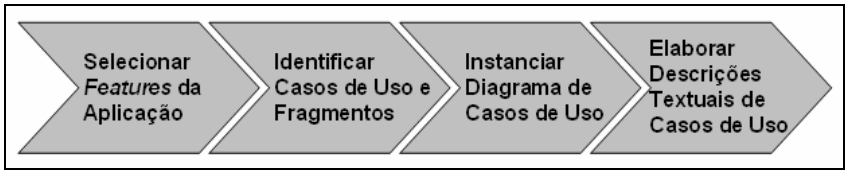


Figura 26 - Atividades da Engenharia da Aplicação na Abordagem Proposta

3.4.1 Selecionar *Features* da Aplicação

- Entrada: características da aplicação e diagrama de *features* do domínio.
- Saída: diagrama de *features* da aplicação.

No contexto da engenharia da aplicação, a abordagem proposta começa com a seleção das características da aplicação a partir do diagrama de *features* do domínio. Uma aplicação típica possui a *feature* comum que representa o núcleo da linha de produto e uma seleção das opcionais e alternativas. Considere, por exemplo, uma aplicação de reserva de aluguel de carros que não ofereça aos clientes extras (opcionais e proteções), não permita a aplicação de qualquer desconto sobre o valor da reserva e tampouco envie um comprovante da reserva após a sua confirmação. A Figura 27 apresenta o diagrama de *features* do domínio, onde as *features* selecionadas para essa aplicação estão destacadas na cor cinza.

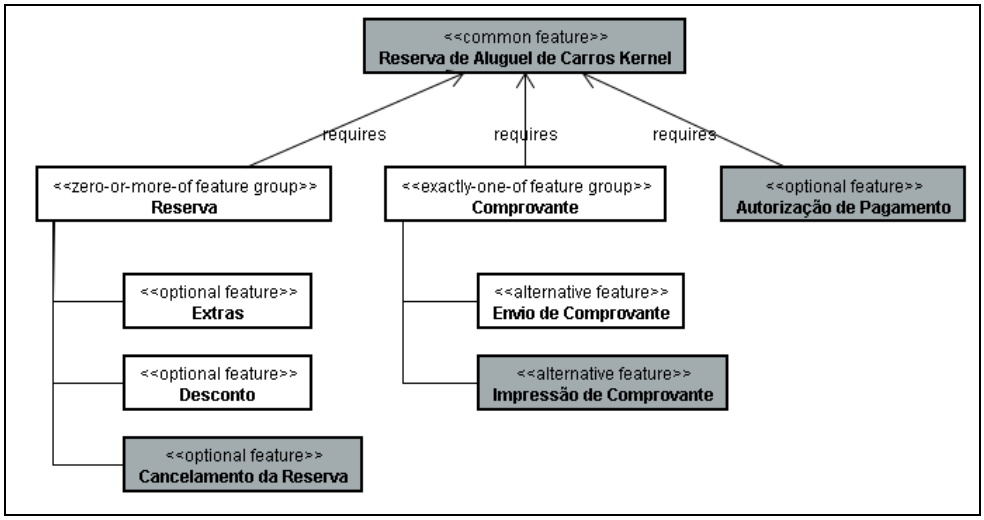


Figura 27 – *Features* Selecionadas para a Aplicação Específica

De acordo com este diagrama, a aplicação de reserva de aluguel de carros em questão é composta pela *feature* comum *Reserva de Aluguel de Carros Kernel,* pelas *features* opcionais *Cancelamento da Reserva* e *Autorização de Pagamento*, e pela *feature* alternativa *Impressão de Comprovante.*

3.4.2 Identificar Casos de Uso e Fragmentos da Aplicação

- Entrada: diagrama de *features* da aplicação e relações entre *features*, casos de uso e fragmentos do domínio.
- Saída: lista de casos de uso e fragmentos da aplicação.

De posse das *features* da aplicação, as relações entre *features*, casos de uso e fragmentos do domínio devem ser selecionadas. O resultado dessa tarefa é a identificação dos casos de uso e dos fragmentos que fazem parte da aplicação. Essa informação é importante para auxiliar a instanciação do diagrama de casos de uso do domínio e a posterior especificação textual desses casos de uso. O Quadro 5 apresenta a tabela com as relações entre *features*, casos de uso e fragmentos do domínio, onde as relações selecionadas para essa aplicação estão destacadas na cor cinza.

Quadro 5 – Relações Selecionadas para a Aplicação Específica

Feature		**Caso de Uso**		**Fragmento**
Nome	**Categoria**	**Nome**	**Categoria/ Fragmento**	**Nome**
Reserva de Carro Kernel	Common	Reservar Carro	Kernel	
			Fragmento	Selecionar Modelo de Carro
			Fragmento	Cadastrar Dados Pessoais
			Fragmento	Obter Confirmação da Reserva
		Consultar Reserva	Kernel	
			Fragmento	Consultar Reserva
		Alterar Reserva	Kernel	
			Fragmento	Solicitar Alteração da Reserva
			Fragmento	Obter Confirmação da Alteração
Extras	Optional	Reservar Carro	Fragmento	Adicionar Extras
Desconto	Optional	Aplicar Desconto	Optional	
			Fragmento	Aplicar Desconto
Cancelamento da Reserva	Optional	Cancelar Reserva	Optional	
			Fragmento	Solicitar Cancelamento da Reserva
Envio de Comprovante	Alternative	Enviar Comprovante	Alternative	
			Fragmento	Enviar Comprovante
Impressão de Comprovante	Alternative	Imprimir Comprovante	Alternative	
			Fragmento	Imprimir Comprovante
Autorização de Pagamento	Optional	Reservar Carro	Fragmento	Autorizar Pagamento em Cartão

Segundo esta tabela, a aplicação possui os seguintes casos de uso: *Reservar Carro*, *Consultar Reserva, Alterar Reserva, Imprimir Comprovante* e *Cancelar Reserva*. O caso de uso *Reservar Carro* pode ser composto pelos fragmentos *Selecionar Modelo de Carro*, *Cadastrar Dados Pessoais, Autorizar Pagamento em Cartão* e *Obter Confirmação da Reserva*. A especificação textual do caso de uso *Alterar Reserva* pode ser auxiliada pela utilização dos fragmentos *Solicitar Alteração da Reserva* e *Obter Confirmação da Alteração*. Os casos de uso *Consultar Reserva, Imprimir Comprovante* e *Cancelar Reserva* podem ser compostos pelos seguintes fragmentos: *Consultar Reserva, Imprimir Comprovante* e *Solicitar Cancelamento da Reserva*, respectivamente.

3.4.3 Instanciar Diagrama de Casos de Uso da Aplicação

- Entrada: lista de casos de uso da aplicação.
- Saída: diagrama de casos de uso da aplicação.

Com base na lista de casos de uso da aplicação, deve-se instanciar o diagrama de casos de uso do domínio. O diagrama de casos de uso da aplicação será constituído por todos os casos de uso *kernel* (identificados pelo estereótipo <<kernel>>) e uma seleção dos opcionais e alternativos (identificados pelos estereótipos <<optional>> e <<alternative>>). A Figura 28 ilustra o diagrama de casos de uso do domínio, onde os casos de uso selecionados para a aplicação estão destacados na cor cinza. Neste diagrama, o ator *Cliente* pode reservar um carro, alterar ou cancelar uma reserva de carro. Entretanto, ele não pode aplicar qualquer desconto sobre o valor da reserva e tampouco enviar um comprovante da reserva.

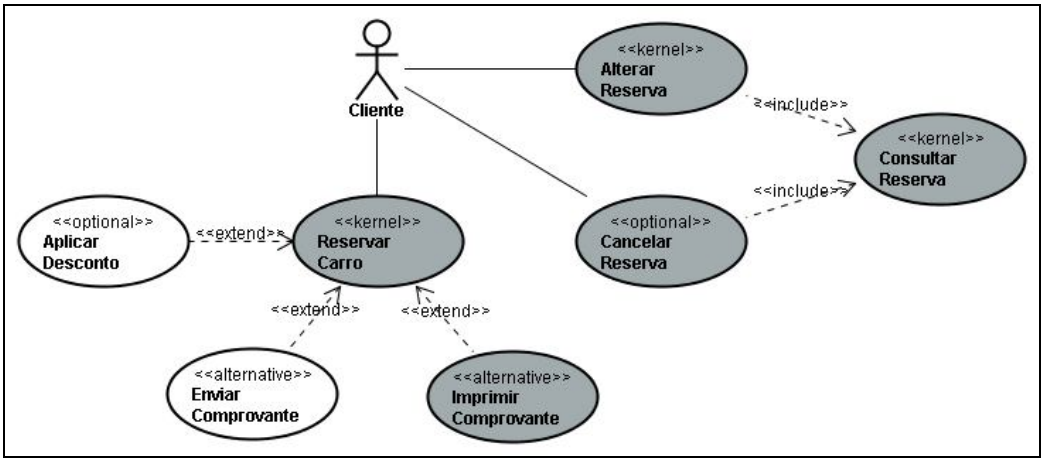


Figura 28 – Casos de Uso Selecionados para a Aplicação Específica

3.4.4 Elaborar Descrição Textual dos Casos de Uso

- Entrada: lista de casos de uso e fragmentos da aplicação.
- Saída: descrição textual completa dos casos de uso da aplicação.

Com base na lista de casos de uso da aplicação e de seus respectivos fragmentos, cada caso de uso deve ser escrito através da composição e personalização de fragmentos. Esta tarefa começa com a incorporação de cada fragmento na descrição textual do caso de uso correspondente. Caso seja necessário, o fragmento pode ser personalizado de acordo com as necessidades específicas da aplicação. A personalização dos fragmentos segue a estratégia proposta por Dias *et al*. (2008), que corresponde a um possível refinamento da definição dos atores relacionados, de todos os fluxos de dados envolvidos nas interações dos atores com o sistema, bem como das regras relacionadas à validação de passos ou à geração de resultados. A Figura 29 apresenta o fragmento *Obter Confirmação da Reserva* personalizado para atender a aplicação específica em questão, onde as substituições são indicadas em negrito e as partes descartadas são indicadas com um risco.

Fluxo Básico:
1. **Cliente** solicita o registro da reserva.
2. Sistema apresenta detalhes da reserva.
3. **Cliente** confirma registro da reserva.
4. Sistema registra a reserva.
5. Sistema apresenta recibo de registro.

Fluxos Alternativos:

a) *registro da reserva cancelado*

No passo 3 do Fluxo Básico, **Cliente** decide cancelar o registro da reserva.
1. Sistema apresenta a mensagem: "Registro da reserva cancelado".
2. Fluxo de eventos retorna ao passo 1 do Fluxo Básico.

Detalhes de Estrutura de Dados:

a) Detalhes da reserva = reserva.data/hora de início, reserva.data/hora de término, reserva.valor, cliente.nome, cliente.sobrenome, local.nome, modelo de carro.nome, ~~lista de extra.nome e extra.valor~~, ~~desconto.código~~, ~~desconto.percentual~~, cartão.número e cartão.bandeira.

b) Detalhes do recibo de registro = reserva.número e "detalhes da reserva".

Detalhes de Regras:

Não há.

Figura 29 – Fragmento de Caso de Uso *Obter Confirmação da Reserva* Personalizado para a Aplicação Específica

Segundo essa personalização, o *Cliente* é o ator que interage com o sistema, nos dados de saída foram retirados os atributos de opcionais, proteções e desconto, tendo em vista que a aplicação não oferece aos clientes nem opcionais, nem proteções e tampouco permite a aplicação de qualquer desconto sobre o valor da reserva do carro.

Para exemplificar uma descrição textual de caso de uso composta por fragmentos, a descrição do caso de uso *Reservar Carro* é apresentada a seguir: Figura 30, Figura 31, Figura 32 e Figura 33, contendo o detalhamento dos passos do fluxo básico e dos fluxos alternativos como também os detalhes de estrutura de dados e de regras de negócio. Esse caso de uso foi elaborado através da composição dos seguintes fragmentos: (1) *Selecionar Modelo de Carro*, (2) *Cadastrar Dados Pessoais*, (3) *Autorizar Pagamento em Cartão* e (4) *Obter Confirmação da Reserva*. O primeiro fragmento deu origem aos passos 2-7 do fluxo básico; aos fluxos alternativos *período de locação inválido* e *nenhum modelo de carro satisfaz a seleção*; à entrada *período de locação*; às saídas *lista de locais de retirada* e *lista de modelos de carro*; e

à *regra de período de locação*. O segundo fragmento deu origem aos passos 8-9 do fluxo básico; ao fluxo alternativo *dados pessoais inválidos*; à entrada *dados pessoais*; e à *regra de dados pessoais*. O terceiro fragmento deu origem aos passos 10-13 do fluxo básico; ao fluxo alternativo *cartão de crédito inválido*; à entrada *dados do cartão de crédito*; à saída *lista de bandeiras de cartão de crédito* e à *regra de cartão de crédito*. O quarto fragmento deu origem aos passos 14-18 do fluxo básico; ao fluxo alternativo *registro da reserva cancelado*; e às saídas *detalhes da reserva* e *detalhes do recibo de registro*.

No exemplo mostrado, os fragmentos selecionados foram personalizados e utilizados para compor a descrição do caso de uso. Entretanto, como os fragmentos são elaborados para uso em diversas aplicações da linha de produto, há situações específicas em que algumas adaptações e acréscimo de informação são necessários à descrição para que o objetivo do caso de uso seja atingido corretamente. Neste exemplo, houve a necessidade apenas de inserir o primeiro e o último passos, para explicitar o início e o fim do caso de uso.

Fluxo Básico:

1. O caso de uso começa quando o Cliente solicita o registro de uma reserva.
2. Sistema apresenta uma lista de locais de retirada.
3. Cliente seleciona um local de retirada.
4. Cliente informa período de locação.
5. Sistema certifica que período de locação é válido de acordo com a regra de período de locação.
6. Sistema apresenta uma lista de modelos de carro.
7. Cliente seleciona um modelo de carro.
8. Cliente informa dados pessoais.
9. Sistema certifica que dados pessoais são válidos de acordo com a regra de dados pessoais.
10. Sistema apresenta uma lista de bandeiras de cartão de crédito.
11. Cliente seleciona uma bandeira de cartão de crédito.
12. Cliente informa dados do cartão de crédito.
13. Sistema certifica que dados do cartão de crédito são válidos de acordo com a regra de cartão de crédito.
14. Cliente solicita o registro da reserva.
15. Sistema apresenta detalhes da reserva.
16. Cliente confirma registro da reserva.
17. Sistema registra a reserva.
18. Sistema apresenta recibo de registro.
19. O caso de uso termina.

Figura 30 – Fluxo Básico Resultante da Composição dos Trechos de Fluxos Básicos

Fluxos Alternativos:

a) *período de locação inválido*

No passo 5 do Fluxo Básico, o Sistema identifica que o período de locação informado viola a regra de período de locação.

1. Sistema apresenta a mensagem: “A data de término informada deve obrigatoriamente ser maior ou igual à data de registro da reserva”.
2. Fluxo de eventos retorna ao passo 4 do Fluxo Básico.

b) *nenhum modelo de carro satisfaz a seleção*

No passo 6 do Fluxo Básico, o Sistema não encontra nenhum modelo de carro para seleção.

1. Sistema apresenta a mensagem: “Nenhum modelo de carro encontrado”.
2. Fluxo de eventos retorna ao passo 4 do Fluxo Básico.

c) *dados pessoais inválidos*

No passo 9 do Fluxo Básico, o Sistema identifica que os dados pessoais informados violam a regra de dados pessoais.

1. Sistema apresenta a mensagem: “O cliente deve possuir idade maior que 18 anos”.
2. Fluxo de eventos retorna ao passo 8 do Fluxo Básico.

d) *cartão de crédito inválido*

No passo 13 do Fluxo Básico, o Sistema identifica que o cartão de crédito informado viola a regra de cartão de crédito.

1. Sistema apresenta a mensagem: “A data de validade do cartão de crédito informado deve obrigatoriamente ser maior que a data atual”.
2. Fluxo de eventos retorna ao passo 12 do Fluxo Básico.

e) *registro da reserva cancelado*

No passo 16 do Fluxo Básico, Cliente decide cancelar o registro da reserva.

1. Sistema apresenta a mensagem: “Registro da reserva cancelado”.
2. Fluxo de eventos retorna ao passo 14 do Fluxo Básico.

Figura 31 – Fluxos Alternativos Associados aos Passos do Fluxo Básico

Detalhes de Estrutura de Dados:

a) Lista de locais de retirada = lista de local.nome e endereço.cidade, ordenados alfabeticamente.
b) Período de locação = data/hora de início e data/hora de término da locação com a data no formato dia/mês/ano.
c) Lista de modelos de carro = lista de modelo de carro.fabricante e modelo de carro.nome, ordenados alfabeticamente. Esta lista contém somente os modelos de carro que estão disponíveis para locação no local de retirada indicado, a partir da data/hora de início até a data/hora de término informada.
d) Dados pessoais = nome, sobrenome, data de nascimento, e-mail, telefone e licença (número e data de validade).
e) Lista de bandeiras de cartão de crédito = lista de cartão.bandeira, ordenadas alfabeticamente.
f) Dados do cartão de crédito = número, código, bandeira, nome do titular e data de validade.
g) Detalhes da reserva = reserva.data/hora de início, reserva.data/hora de término, reserva.valor, cliente.nome, cliente.sobrenome, local.nome, modelo de carro.nome, cartão.número e cartão.bandeira.
h) Detalhes do recibo de registro = reserva.número e “detalhes da reserva”.

Figura 32 – Detalhes das Estruturas de Dados de todos os Termos de Negócio participantes do Caso de Uso

Detalhes de Regras:

a) Regra de período de locação = a data de término informada deve obrigatoriamente ser maior ou igual à data de registro da reserva.
b) Regra de dados pessoais = o cliente deve possuir idade maior ou igual a 18 anos.
c) Regra de cartão de crédito = a data de validade do cartão de crédito informado deve obrigatoriamente ser maior que a data de término da reserva.

Figura 33 – Detalhes das Regras de Negócio que Restringem os Termos de Negócio

3.5 Considerações Finais

Neste capítulo, foi apresentada uma abordagem para a elaboração de especificações de casos de uso para LPS. Na engenharia do domínio, realizou-se uma análise de similaridades e variabilidades do domínio, onde uma modelagem de objetivos associada ao uso de matrizes foi utilizada para identificar o domínio. Para auxiliar a elaboração das especificações de casos de uso, diretrizes foram definidas para a construção desses artefatos a partir do conjunto de objetivos e sub-objetivos do domínio. Para representar a variabilidade nas especificações de casos de uso, a notação do método PLUS (GOMAA, 2004) foi adotada para representar as variações ao nível de diagrama e fragmentos de casos de uso específicos do domínio foram elaborados para representar as variações ao nível de descrições textuais. Na engenharia da aplicação, a instanciação das especificações de casos de uso para uma dada aplicação foi orientada pelas relações pré-definidas entre *features*, casos de uso e fragmentos. As descrições textuais dos casos de uso foram elaboradas através da seleção, personalização e composição de fragmentos.

4 EXEMPLO DE APLICAÇÃO

4.1 Introdução

O capítulo anterior apresentou a abordagem para a elaboração de especificações de casos de uso para LPS baseada em fragmentos e um exemplo simples de ilustração. Para avaliar e analisar o uso prático da abordagem proposta, uma linha de produto de sistemas de aluguel de carros será apresentada neste capítulo. Primeiramente, será descrita a engenharia do domínio, apresentando os principais artefatos obtidos nesta etapa. Em seguida, a engenharia da aplicação é mostrada, apresentando três exemplos de aplicações desenvolvidas com o auxílio deste arcabouço. Por último, será apresentada a discussão sobre a aplicação da abordagem.

4.2 Engenharia do Domínio

A linha de produto descrita neste exemplo corresponde a um domínio de aluguel de carros, onde foram analisadas quatro aplicações reais: *HireMate*[1], *RentalPlanner*[2], *RentCentric*[3] e *EasyRentPro*[4]. A escolha de tais aplicações levou em consideração: o domínio ao qual elas pertencem, tendo em vista que parte deste domínio, isto é, o sub-domínio de reserva de aluguel de carros, já era conhecido devido ao exemplo elaborado e mostrado no Capítulo 3; e a disponibilidade de versões de teste gratuitas dessas aplicações. Esta seção apresenta os principais artefatos obtidos na engenharia do domínio: modelo conceitual do

[1] http://www.grensoft.com/HireMate/index.htm

[2] http://www.rental-planner.com

[3] http://www.rentcentric.com

[4] http://www.easyrentpro.com

domínio, diagrama de casos de uso do domínio, catálogo de fragmentos de casos de uso e diagrama de *features* do domínio.

4.2.1 Modelo Conceitual do Domínio

Cada aplicação foi analisada do ponto de vista do usuário, realizando-se simulações de várias funcionalidades existentes na mesma por meio de sua interface com o usuário. O resultado da análise foi o entendimento do domínio, onde os principais termos de negócio envolvidos e os seus relacionamentos foram identificados e documentados. A Figura 34 apresenta um trecho do modelo conceitual do domínio de aluguel de carros elaborado para o exemplo.

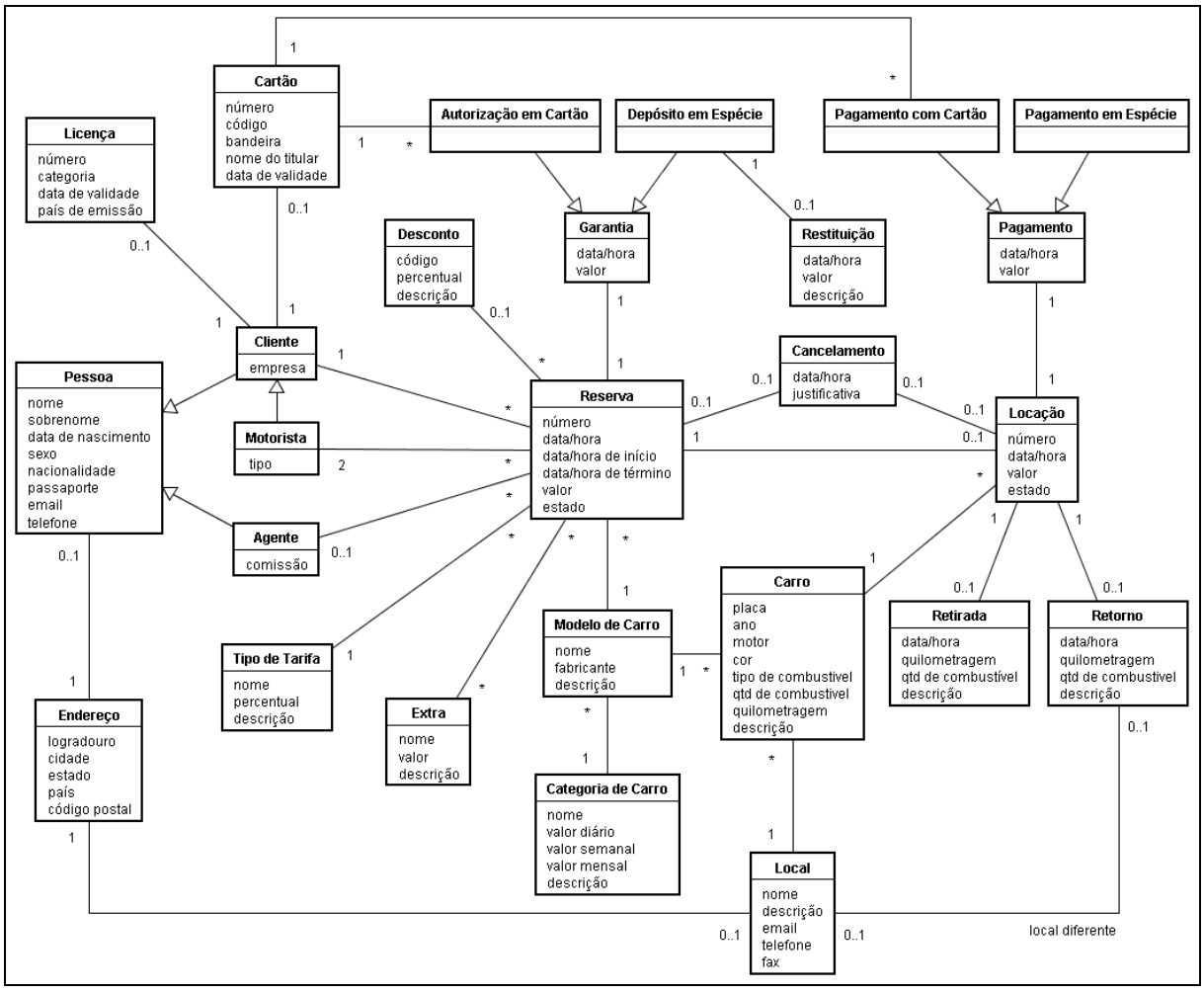


Figura 34 – Trecho do Modelo Conceitual do Domínio de Aluguel de Carros

Neste exemplo, a reserva de um carro envolve a seleção de um modelo de carro, que se dá a partir da escolha de um local de retirada e uma categoria de carro, e da definição de um período de locação. Tal modelo de carro deve ser pertencente à categoria escolhida e estar disponível para reserva no local indicado e no período informado. O cliente solicitante da reserva deve ser identificado, o motorista principal do carro alugado deve ser cadastrado e um agente de referência pode ser selecionado. Além disso, um tipo de tarifa pode ser escolhido, extras (opcionais ou proteções) podem ser adicionados, um desconto sobre o valor da reserva pode ser solicitado e um motorista adicional pode ser cadastrado. Para assegurar a reserva, uma garantia é exigida e esta pode ser realizada de duas formas: depósito em espécie ou autorização de pagamento em cartão de crédito do cliente. Eventualmente, a reserva do carro pode ser cancelada após a sua confirmação e esta ação pode implicar na restituição do valor do depósito ou parte deste, caso a garantia tenha sido feita por depósito em espécie.

Na retirada do carro, a reserva torna-se uma locação, um carro referente ao modelo escolhido é disponibilizado e o seu estado atual deve ser fornecido. No retorno do carro, o pagamento da locação deve ser efetuado e este pode ser realizado de duas formas: pagamento em espécie ou pagamento com cartão de crédito do cliente. Assim como na retirada, o estado atual do carro também deve ser fornecido e caso o local de retorno seja diferente do local de retirada, esse local também deve ser indicado. Eventualmente, a locação do carro pode ser cancelada após a sua confirmação.

4.2.2 Matriz Aplicações x Objetivos

O Quadro 6 apresenta a matriz aplicações x objetivos elaborada para o domínio de aluguel de carros do exemplo. Para facilitar a visualização desse artefato, as aplicações referências, *HireMate*, *RentalPlanner*, *RentCentric* e *EasyRentPro*, foram aqui denominadas A1, A2, A3 e A4, respectivamente.

Quadro 6 – Matriz Aplicações x Objetivos da LPS de Aluguel de Carros

Objetivos/Sub-objetivos		Categoria	A1	A2	A3	A4
1	**Reservar Carro**	**comum**	**x**	**x**	**x**	**x**
1.1	Identificar Cliente	comum	x	x	x	x
1.2	Selecionar Modelo de Carro	comum	x	x	x	x
1.3	Adicionar Extras	opcional	x		x	x
1.4	Selecionar Tipo de Tarifa	opcional			x	x
1.5	Aplicar Desconto	opcional	x		x	x
1.6	Cadastrar Motorista	comum	x	x	x	x
1.7	Identificar Agente	opcional	x	x		x
1.8	Depositar em Espécie	comum	x	x	x	x
1.9	Autorizar Pagamento em Cartão	comum	x	x	x	x
1.10	Obter Confirmação da Reserva	comum	x	x	x	x
2	**Retirar Carro**	**comum**	**x**	**x**	**x**	**x**
2.1	Consultar Reserva	comum	x	x	x	x
2.2	Solicitar Retirada do Carro	comum	x	x	x	x
2.3	Imprimir Contrato	opcional	x			x
2.4	Enviar Contrato	opcional			x	x
3	**Retornar Carro**	**comum**	**x**	**x**	**x**	**x**
3.1	Consultar Locação	comum	x	x	x	x
3.2	Solicitar Retorno do Carro	comum	x	x	x	x
3.3	Pagar em Espécie	comum	x	x	x	x
3.4	Pagar com Cartão	comum	x	x	x	x
3.5	Imprimir Recibo	opcional	x	x		
4	**Cancelar Reserva**	**comum**	**x**	**x**	**x**	**x**
4.1	Consultar Reserva	comum	x	x	x	x
4.2	Solicitar Cancelamento da Reserva	comum	x	x	x	x
4.3	Restituir Depósito	opcional	x		x	
5	**Cancelar Locação**	**opcional**	**x**		**x**	**x**
5.1	Consultar Locação	comum	x	x	x	x
5.2	Solicitar Cancelamento da Locação	opcional	x		x	x
5.3	Imprimir Comprovante	opcional	x			x
6	**Cadastrar Cliente**	**comum**	**x**	**x**	**x**	**x**
6.1	Cadastrar Dados Pessoais	comum	x	x	x	x

Segundo a análise de similaridades e variabilidades realizada, foram identificados 6 objetivos, sendo 5 comuns, como o objetivo *Reservar Carro* e um opcional, o objetivo *Cancelar Locação*. Em relação aos sub-objetivos, foram identificados 24, sendo 14 comuns, como o sub-objetivo *Selecionar Modelo de Carro*, e 10 opcionais, como o sub-objetivo *Adicionar Extras*. Também foi observado que dois sub-objetivos pertencem a dois objetivos

diferentes: *Consultar Reserva*, que compõe os objetivos *Retirar Carro* e *Cancelar Reserva*; e *Consultar Locação*, que compõe os objetivos *Retornar Carro* e *Cancelar Locação*.

4.2.3 Diagrama de Casos de Uso do Domínio

A Figura 35 apresenta o diagrama de casos de uso construído para o domínio de aluguel de carros do exemplo. Segundo esse diagrama, a linha de produto possui 17 casos de uso, sendo 12 *kernel*, como o caso de uso *Retirar Carro*, e 5 opcionais, como o caso de uso *Aplicar Desconto*. O ator *Recepcionista* pode cadastrar um cliente, reservar um carro, cancelar uma reserva, retirar ou retornar um carro. O ator *Gerente* pode realizar as mesmas funções que a *Recepcionista* realiza, além de poder cancelar uma locação de carro. O ator *Administradora de Cartão* é responsável por validar e confirmar o pagamento com cartão.

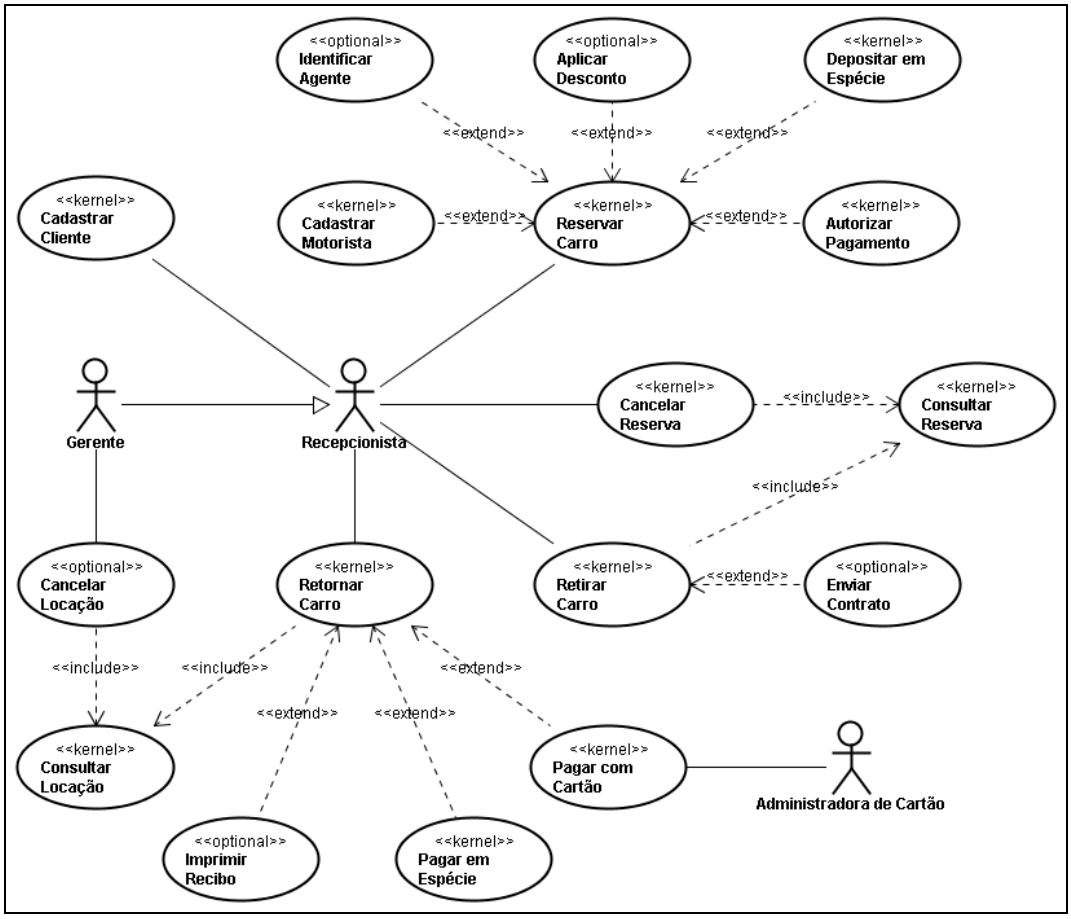


Figura 35 – Diagrama de Casos de Uso da LPS de Aluguel de Carros

4.2.4 Catálogo de Fragmentos de Casos de Uso

Um catálogo de fragmentos de casos de uso foi desenvolvido para o domínio de aluguel de carros do exemplo. Seguindo a quantidade de sub-objetivos encontrados, foram elaborados 24 fragmentos. Os fragmentos, juntamente com seus sub-objetivos, são:

- Identificar Cliente: identificar um dado cliente;
- Selecionar Modelo de Carro: selecionar um modelo de carro para a reserva;
- Adicionar Extras: adicionar extras à reserva;
- Selecionar Tipo de Tarifa: selecionar um tipo de tarifa para a reserva;
- Aplicar Desconto: aplicar desconto sobre o valor da reserva;
- Cadastrar Motorista: cadastrar um motorista para a reserva;
- Identificar Agente: identificar um dado agente de referência;
- Depositar em Espécie: garantir a reserva através de depósito em espécie;
- Autorizar Pagamento em Cartão: garantir a reserva através de autorização de pagamento em cartão de crédito;
- Obter Confirmação da Reserva: obter confirmação a respeito do registro da reserva;
- Consultar Reserva: obter detalhes de uma dada reserva;
- Solicitar Retirada do Carro: solicitar a retirada do carro reservado;
- Imprimir Contrato: imprimir o contrato de locação do carro;
- Enviar Contrato: enviar o contrato de locação do carro;
- Consultar Locação: obter detalhes de uma dada locação;
- Solicitar Retorno do Carro: solicitar o retorno do carro alugado;
- Pagar em Espécie: pagar a locação em espécie;
- Pagar com Cartão: pagar a locação com cartão de crédito;

- Imprimir Recibo: imprimir o recibo de pagamento da locação;
- Solicitar Cancelamento da Reserva: solicitar o cancelamento da reserva;
- Restituir Depósito: restituir o depósito em espécie da reserva;
- Solicitar Cancelamento da Locação: solicitar o cancelamento da locação;
- Imprimir Comprovante: imprimir o comprovante de cancelamento da locação;
- Cadastrar Dados Pessoais: cadastrar os dados pessoais do cliente.

Os dez primeiros fragmentos de casos de uso listados acima estão relacionados ao objetivo *Reservar Carro*. O catálogo contendo todos os detalhes a respeito desses fragmentos encontra-se no Apêndice A. A descrição completa do catálogo elaborado para o domínio de aluguel de carros pode ser encontrada no *site* do grupo de pesquisa[5].

4.2.5 Diagrama de *Features* do Domínio

A Figura 36 apresenta o diagrama de *features* construído para o domínio de aluguel de carros do exemplo. De acordo com esse diagrama, a linha de produto possui 11 *features*, sendo uma comum representando o núcleo da linha de produto (*Aluguel de Carros Kernel*) e os restantes opcionais, como a *feature Tipo de Tarifa*. Há também dois grupos de *features*: *Reserva* e *Contrato de Locação*, cujas *features* podem ser selecionadas sem restrição.

[5] Grupo de Pesquisa GETI: http://geti.dcc.ufrj.br

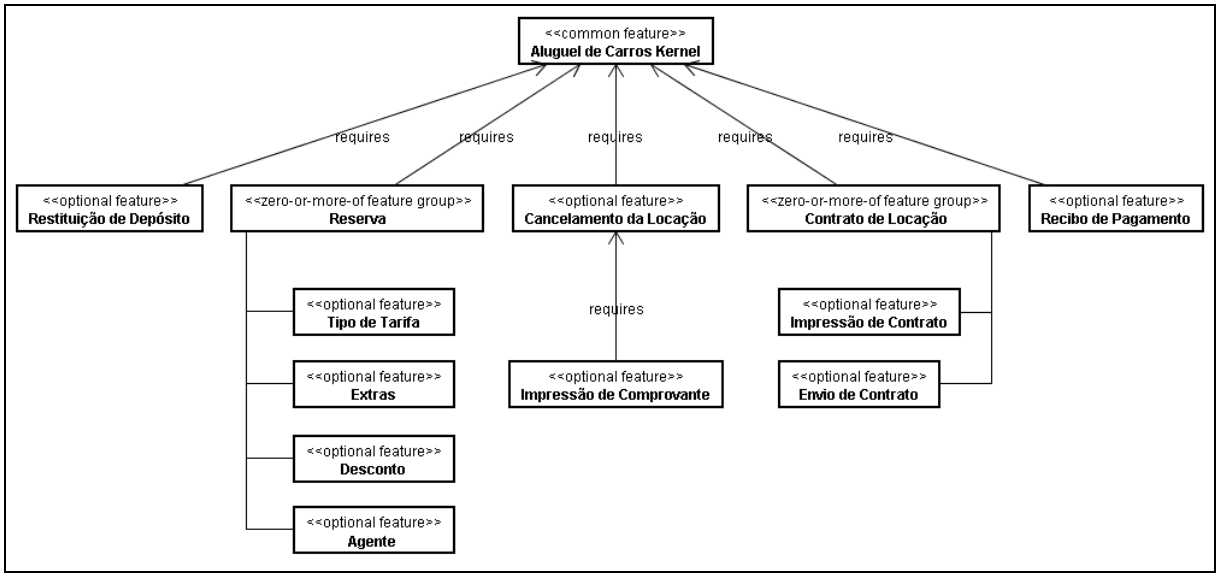


Figura 36 – Diagrama de *Features* da LPS de Aluguel de Carros

4.2.6 Relações entre *Features*, Casos de Uso e Fragmentos

O Quadro 7 apresenta a tabela com as relações existentes entre *features*, casos de uso e fragmentos para o domínio de aluguel de carros do exemplo. Segundo essa tabela, o núcleo está relacionado a 12 casos de uso *kernel*, como o caso de uso *Retirar Carro*, e a 14 fragmentos, como o fragmento *Consultar Reserva*. Algumas *features* opcionais estão relacionadas a apenas um fragmento, como é o caso da *feature Extras* que está relacionada ao fragmento *Adicionar Extras*. Outras *features* opcionais estão relacionadas a um caso de uso e seus respectivos fragmentos, como é o caso da *feature Recibo de Pagamento* que está relacionada ao caso de uso *Imprimir Recibo* e ao fragmento de mesmo nome.

Quadro 7 – Relações entre *Features*, Casos de Uso e Fragmentos da LPS de Aluguel de Carros

Feature		Caso de Uso		Fragmento
Nome	**Categoria**	**Nome**	**Categoria/ Fragmento**	**Nome**
Aluguel de Carro Kernel	Common	Reservar Carro	Kernel	
			Fragmento	Identificar Cliente
			Fragmento	Selecionar Modelo de Carro
			Fragmento	Obter Confirmação da Reserva
		Cadastrar Motorista	Kernel	
			Fragmento	Cadastrar Motorista
		Depositar em Espécie	Kernel	
			Fragmento	Depositar em Espécie
		Autorizar Pagamento	Kernel	
			Fragmento	Autorizar Pagamento em Cartão
		Consultar Reserva	Kernel	
			Fragmento	Consultar Reserva
		Retirar Carro	Kernel	
			Fragmento	Solicitar Retirada do Carro
		Consultar Locação	Kernel	
			Fragmento	Consultar Locação
		Retornar Carro	Kernel	
			Fragmento	Solicitar Retorno do Carro
		Pagar em Espécie	Kernel	
			Fragmento	Pagar em Espécie
		Pagar com Cartão	Kernel	
			Fragmento	Pagar com Cartão
		Cancelar Reserva	Kernel	
			Fragmento	Solicitar Cancelamento da Reserva
		Cadastrar Cliente	Kernel	
			Fragmento	Cadastrar Dados Pessoais
Extras	Optional	Reservar Carro	Fragmento	Adicionar Extras
Tipo de Tarifa	Optional	Reservar Carro	Fragmento	Selecionar Tipo de Tarifa
Agente	Optional	Identificar Agente	Optional	
			Fragmento	Identificar Agente
Desconto	Optional	Aplicar Desconto	Optional	
			Fragmento	Aplicar Desconto
Impressão de Contrato	Optional	Retirar Carro	Fragmento	Imprimir Contrato
Envio de Contrato	Optional	Enviar Contrato	Optional	
			Fragmento	Enviar Contrato
Recibo de Pagamento	Optional	Imprimir Recibo	Optional	
			Fragmento	Imprimir Recibo
Restituição de Depósito	Optional	Cancelar Reserva	Fragmento	Restituir Depósito
Cancelamento da Locação	Optional	Cancelar Locação	Optional	
			Fragmento	Solicitar Cancelamento da Locação
Impressão de Comprovante	Optional	Cancelar Locação	Fragmento	Imprimir Comprovante

4.3 Engenharia da Aplicação

Esta seção descreve a engenharia da aplicação da linha de produto de aluguel de carros. Seguindo a sugestão apresentada no trabalho de Donegan (2008), serão apresentados três exemplos de aplicações desenvolvidas com o auxílio deste arcabouço. Primeiramente, será descrito o desenvolvimento de uma das aplicações usada como referência para a linha de produto. Em seguida, o desenvolvimento do núcleo operacional é mostrado. Por último, será apresentado o desenvolvimento de uma nova aplicação, que é diferente das aplicações alvo e do núcleo operacional.

4.3.1 Aplicação Referência

A seguir é detalhado o processo de engenharia da aplicação *HireMate*. Como essa aplicação foi previamente avaliada na análise da linha de produto como uma aplicação referência, já se tem idéia dos requisitos e das características que a aplicação deve possuir. A Figura 37 apresenta o diagrama de *features* do domínio, onde as *features* selecionadas para essa aplicação estão destacadas na cor cinza.

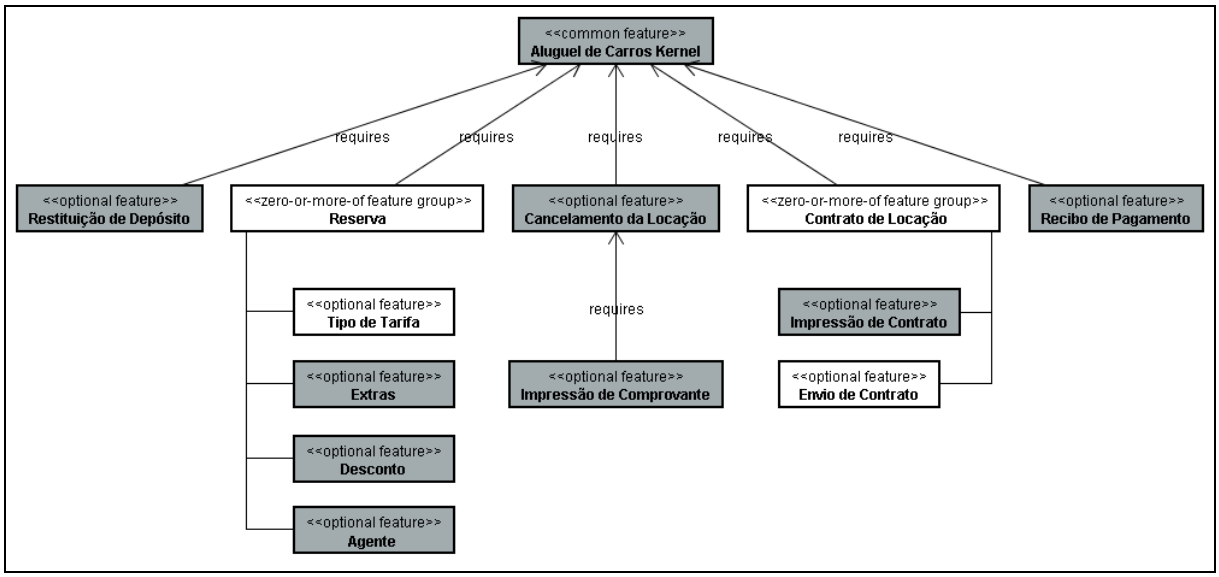


Figura 37 – *Features* Selecionadas para a Aplicação *HireMate*

De acordo com este diagrama, a aplicação de reserva de aluguel de carros em questão é composta pela *feature* comum *Aluguel de Carros Kernel* e pelas *features* opcionais *Restituição de Depósito*, *Extras*, *Desconto*, *Agente*, *Cancelamento da Locação*, *Impressão de Comprovante*, *Impressão de Contrato* e *Recibo de Pagamento*. As *features* opcionais *Tipo de Tarifa* e *Envio de Contrato* não fazem parte da aplicação *HireMate*.

O Quadro 8 ilustra a tabela com as relações entre *features*, casos de uso e fragmentos do domínio, onde as relações selecionadas para essa aplicação estão destacadas na cor cinza. Por simplicidade, não serão mostradas todas as relações associadas à *feature* que representa o núcleo da linha de produto.

Quadro 8 – Relações Selecionadas para a Aplicação *HireMate*

Feature		**Caso de Uso**		**Fragmento**
Nome	**Categoria**	**Nome**	**Categoria/ Fragmento**	**Nome**
Aluguel de Carro Kernel	Common	Reservar Carro	Kernel	
			Fragmento	Identificar Cliente
			Fragmento	Selecionar Modelo de Carro
			Fragmento	Depositar em Espécie
...	...	...	...	...
Extras	Optional	Reservar Carro	Fragmento	Adicionar Extras
Tipo de Tarifa	Optional	Reservar Carro	Fragmento	Selecionar Tipo de Tarifa
Agente	Optional	Identificar Agente	Optional	
			Fragmento	Identificar Agente
Desconto	Optional	Aplicar Desconto	Optional	
			Fragmento	Aplicar Desconto
Impressão de Contrato	Optional	Retirar Carro	Fragmento	Imprimir Contrato
Envio de Contrato	Optional	Enviar Contrato	Optional	
			Fragmento	Enviar Contrato
Recibo de Pagamento	Optional	Imprimir Recibo	Optional	
			Fragmento	Imprimir Recibo
Restituição de Depósito	Optional	Cancelar Reserva	Fragmento	Restituir Depósito
Cancelamento da Locação	Optional	Cancelar Locação	Optional	
			Fragmento	Solicitar Cancelamento da Locação
Impressão de Comprovante	Optional	Cancelar Locação	Fragmento	Imprimir Comprovante

Segundo esta tabela, a aplicação é composta por todos os casos de uso *kernel*, como o caso de uso *Reservar Carro* e todos os opcionais, exceto o caso de uso *Enviar Contrato*. Da

mesma forma, fazem parte da aplicação todos os fragmentos cujos sub-objetivos são comuns, como o fragmento *Identificar Cliente*, e alguns dos fragmentos cujos sub-objetivos são opcionais, como o fragmento *Adicionar Extras*. Neste caso, os fragmentos *Selecionar Tipo de Tarifa* e *Enviar Contrato* não pertencem à aplicação.

A Figura 38 apresenta o diagrama de casos de uso do domínio, onde os casos de uso selecionados para essa aplicação estão destacados na cor cinza. Neste diagrama, o ator *Recepcionista* pode cadastrar um cliente, reservar um carro, cancelar uma reserva, retirar ou retornar um carro. Entretanto, ela não pode enviar o contrato durante a retirada de um carro. O ator *Gerente* pode realizar as mesmas funções que a *Recepcionista* pode realizar, além de poder cancelar uma locação. O ator *Administradora de Cartão* é responsável por validar e confirmar o pagamento com cartão de crédito.

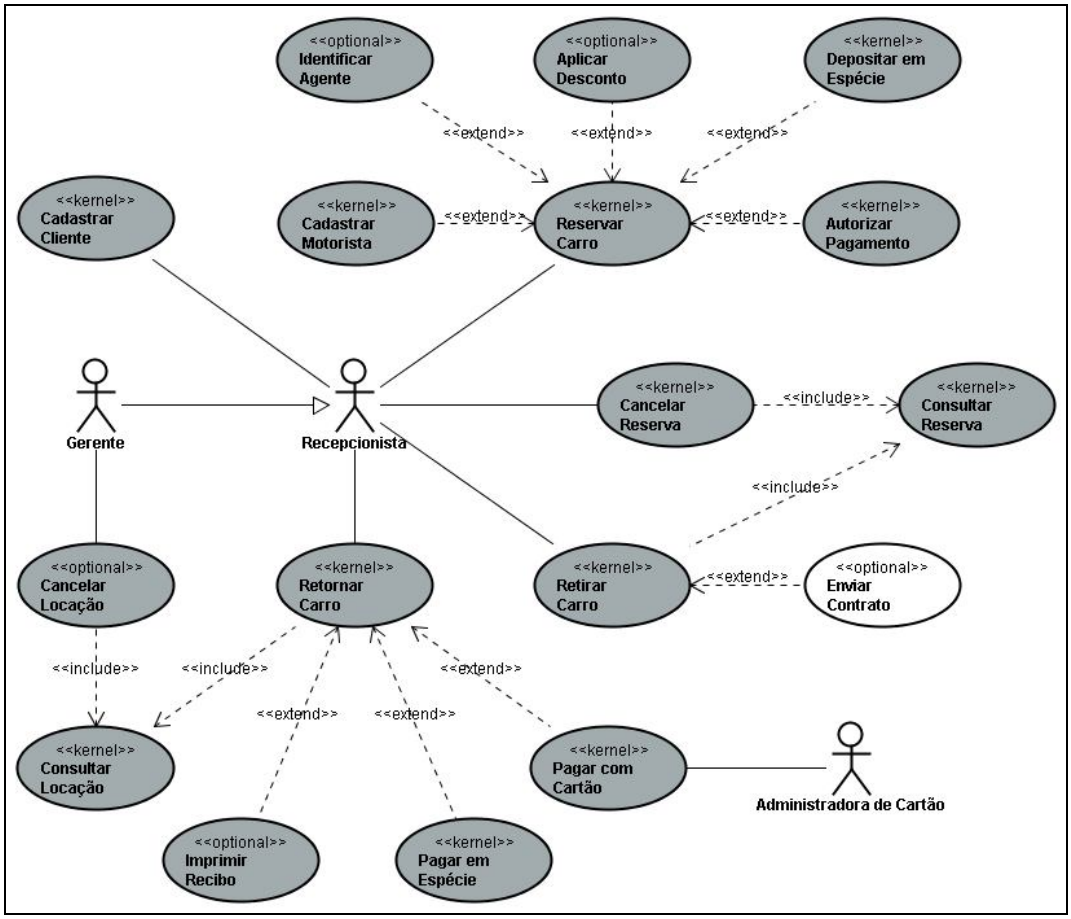


Figura 38 – Casos de Uso Selecionados para a Aplicação *HireMate*

A Figura 39 apresenta o fragmento *Selecionar Modelo de Carro* personalizado para atender a aplicação específica em questão, onde as substituições são indicadas em negrito e as partes descartadas são indicadas com um risco. De acordo com essa personalização, a *Recepcionista* é o ator que interage com o sistema, no fluxo principal foi retirado o passo 9 e nos dados de saída foram retirados os detalhes do modelo de carro, tendo em vista que essa aplicação não apresenta os detalhes do modelo de carro após a seleção deste para a reserva.

Fluxo Básico:
1. Sistema apresenta uma lista de locais de retirada.
2. **Recepcionista** seleciona um local de retirada.
3. Sistema apresenta uma lista de categorias de carro.
4. **Recepcionista** seleciona uma categoria de carro.
5. **Recepcionista** informa período de locação.
6. Sistema certifica que período de locação é válido de acordo com a regra de período de locação.
7. Sistema apresenta uma lista de modelos de carro.
8. **Recepcionista** seleciona um modelo de carro.
9. ~~Sistema apresenta detalhes do modelo de carro selecionado.~~

Fluxos Alternativos:

a) *período de locação inválido*

No passo 6 do Fluxo Básico, o Sistema identifica que o período de locação informado viola a regra de período de locação.
1. Sistema apresenta a mensagem: "A data de término informada deve obrigatoriamente ser maior ou igual à data de registro da reserva".
2. Fluxo de eventos retorna ao passo 5 do Fluxo Básico.

b) *nenhum modelo de carro satisfaz a seleção*

No passo 7 do Fluxo Básico, o Sistema não encontra nenhum modelo de carro para seleção.
1. Sistema apresenta a mensagem: "Nenhum modelo de carro encontrado".
2. Fluxo de eventos retorna ao passo 5 do Fluxo Básico.

Detalhes de Estrutura de Dados:
a) Lista de locais de retirada = lista de local.nome e endereço.cidade, ordenados alfabeticamente.
b) Lista de categorias de carro = lista de categoria de carro.nome, ordenados alfabeticamente.
c) Período de locação = data/hora de início e data/hora de término da locação com a data no formato dia/mês/ano.
d) Lista de modelos de carro = lista de categoria.nome, modelo de carro.fabricante, modelo de carro.nome, ordenados alfabeticamente. Esta lista contém somente os modelos de carro que estão disponíveis para locação no local de retirada indicado, a partir da data/hora de início até a data/hora de término informada.
e) ~~Detalhes do modelo de carro = <categoria de carro.nome, modelo de carro.nome, modelo de carro.fabricante e modelo de carro.descrição>.~~

Detalhes de Regras:
a) Regra de período de locação = a data de término informada deve obrigatoriamente ser maior ou igual à data de registro da reserva.

Figura 39 – Fragmento *Selecionar Modelo de Carro* Personalizado para a Aplicação *HireMate*

A Figura 40 apresenta o fluxo básico do caso de uso *Reservar Carro* para a aplicação *HireMate*, que foi elaborado com o auxílio de trechos de fluxos básicos originados dos seguintes fragmentos: (1) *Identificar Cliente*, (2) *Selecionar Modelo de Carro*, (3) *Adicionar Extras* e (4) *Obter Confirmação da Reserva*. O primeiro fragmento deu origem aos passos 2-4 do fluxo básico; o segundo deu origem aos passos 6-13; o terceiro deu origem aos passos 14-17; e o último deu origem aos passos 21-25. Os passos 1, 5, 18-20 e 26 do fluxo básico não foram contemplados na personalização dos fragmentos e foram adicionados posteriormente.

Fluxo Básico:

1. O caso de uso começa quando a Recepcionista solicita o registro de uma reserva.
2. Recepcionista informa parâmetros para busca de clientes.
3. Sistema apresenta uma lista de clientes.
4. Recepcionista seleciona um cliente.
5. Recepcionista informa que o cliente é o motorista principal.
6. Sistema apresenta uma lista de locais de retirada.
7. Recepcionista seleciona um local de retirada.
8. Sistema apresenta uma lista de categorias de carro.
9. Recepcionista seleciona uma categoria de carro.
10. Recepcionista informa período de locação.
11. Sistema certifica que período de locação é válido de acordo com a regra de período de locação.
12. Sistema apresenta uma lista de modelos de carro.
13. Recepcionista seleciona um modelo de carro.
14. Sistema apresenta uma lista de extras.
15. Recepcionista seleciona um ou mais extras.
16. Recepcionista confirma adição de extras.
17. Sistema atualiza valor da reserva.
18. Sistema apresenta uma lista de formas de garantia da reserva.
19. Recepcionista seleciona Depósito em Espécie como forma de garantia da reserva.
20. Executar o caso de uso “Depositar em Espécie”.
21. Recepcionista solicita o registro da reserva.
22. Sistema apresenta detalhes da reserva.
23. Recepcionista confirma registro da reserva.
24. Sistema registra a reserva.
25. Sistema apresenta recibo de registro.
26. O caso de uso termina.

Figura 40 – Fluxo Básico do Caso de Uso *Reservar Carro* para a Aplicação *HireMate*

4.3.2 Núcleo Operacional

A engenharia da aplicação também pode ser usada para gerar o núcleo da linha de produto, pois ele é operacional. O mesmo processo explicado para a engenharia da aplicação *HireMate* deve ser seguido, com a diferença de que as *features* opcionais não devem ser selecionadas, pois o núcleo possui apenas o que há de comum numa linha de produto. Desta forma, a aplicação gerada possui apenas a *feature* comum *Aluguel de Carros Kernel*, o que implica num diagrama de casos de uso composto por apenas os casos de uso *kernel* e um catálogo de fragmentos também reduzido.

A Figura 41 apresenta o diagrama de casos de uso do domínio, onde os casos de uso selecionados para essa aplicação estão destacados na cor cinza. Neste diagrama, o ator *Recepcionista* pode cadastrar um cliente, reservar um carro, cancelar uma reserva, retirar ou retornar um carro. Entretanto, ela não pode identificar um agente de referência para uma reserva, aplicar um desconto sobre o valor de uma reserva, enviar o contrato na retirada de um carro e imprimir o recibo de pagamento no retorno de um carro. O ator *Gerente* pode realizar as mesmas funções que a *Recepcionista* pode realizar, entretanto não pode cancelar uma locação. O ator *Administradora de Cartão* é responsável por validar e confirmar o pagamento com cartão de crédito.

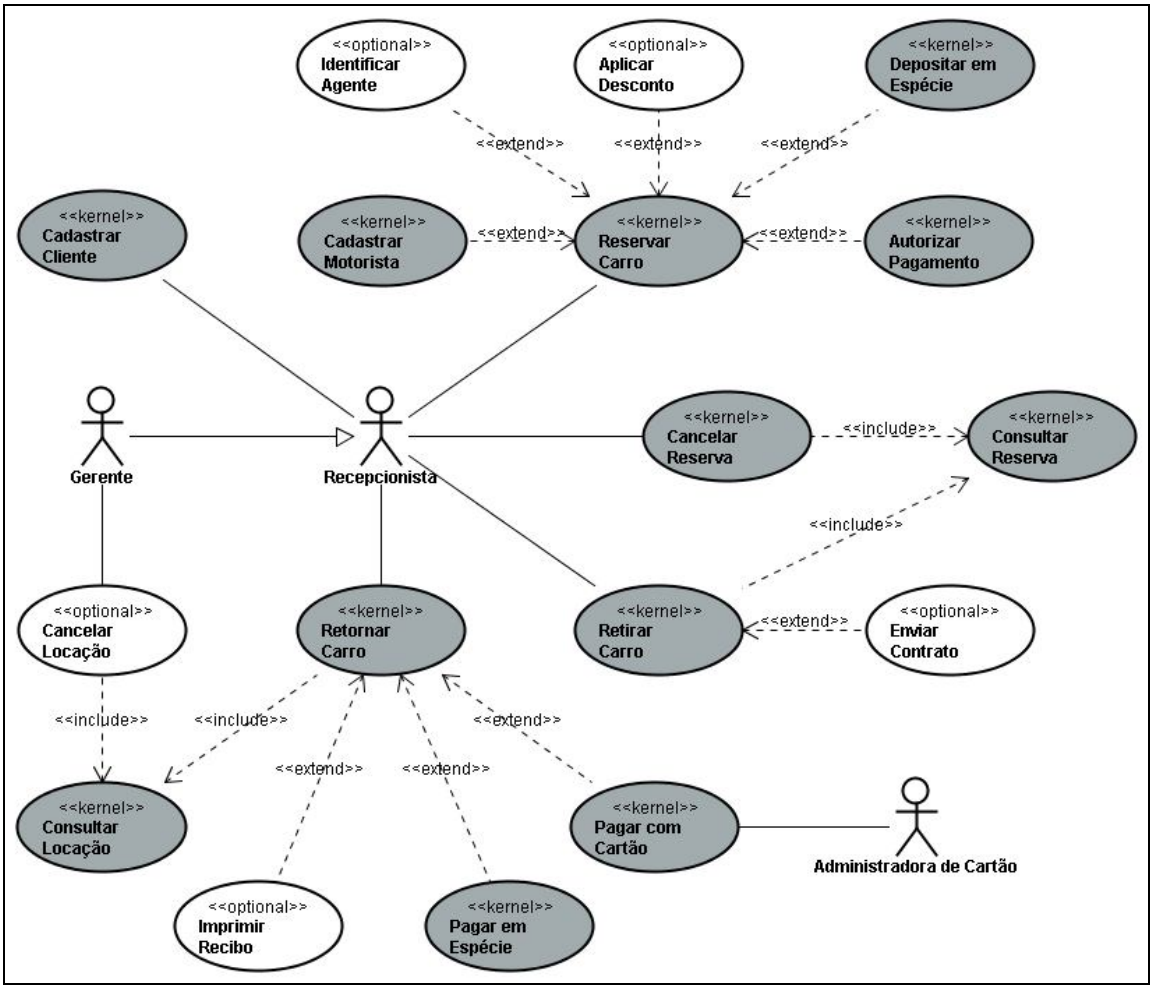


Figura 41 – Casos de Uso Selecionados para o Núcleo Operacional

A Figura 42 apresenta o fragmento *Selecionar Modelo de Carro* personalizado para atender a aplicação específica em questão, onde as substituições são indicadas em negrito e as partes descartadas são indicadas com um risco. De acordo com essa personalização, a *Recepcionista* é o ator que interage com o sistema, no fluxo principal foram retirados os passos 3, 4 e 9, e nos dados de saída foram retirados a lista de categorias de carro e os detalhes do modelo de carro, tendo em vista que essa aplicação possui apenas as partes obrigatórias dos fragmentos.

Fluxo Básico:

1. Sistema apresenta uma lista de locais de retirada.
2. **Recepcionista** seleciona um local de retirada.
3. ~~Sistema apresenta uma lista de categorias de carro.~~
4. ~~<recepcionista> seleciona uma categoria de carro.~~
5. **Recepcionista** informa período de locação.
6. Sistema certifica que período de locação é válido de acordo com a regra de período de locação.
7. Sistema apresenta uma lista de modelos de carro.
8. **Recepcionista** seleciona um modelo de carro.
9. ~~Sistema apresenta detalhes do modelo de carro selecionado.~~

Fluxos Alternativos:

a) *período de locação inválido*

No passo 6 do Fluxo Básico, o Sistema identifica que o período de locação informado viola a regra de período de locação.

1. Sistema apresenta a mensagem: "A data de término informada deve obrigatoriamente ser maior ou igual à data de registro da reserva".
2. Fluxo de eventos retorna ao passo 5 do Fluxo Básico.

b) *nenhum modelo de carro satisfaz a seleção*

No passo 7 do Fluxo Básico, o Sistema não encontra nenhum modelo de carro para seleção.

1. Sistema apresenta a mensagem: "Nenhum modelo de carro encontrado".
2. Fluxo de eventos retorna ao passo 5 do Fluxo Básico.

Detalhes de Estrutura de Dados:

a) Lista de locais de retirada = lista de local.nome e endereço.cidade, ordenados alfabeticamente.

b) ~~Lista de categorias de carro = lista de categoria de carro.nome, ordenados alfabeticamente.~~

c) Período de locação = data/hora de início e data/hora de término da locação com a data no formato dia/mês/ano.

d) Lista de modelos de carro = lista de categoria.nome, modelo de carro.fabricante, modelo de carro.nome, ordenados alfabeticamente. Esta lista contém somente os modelos de carro que estão disponíveis para locação no local de retirada indicado, a partir da data/hora de início até a data/hora de término informada.

e) ~~Detalhes do modelo de carro = <categoria de carro.nome, modelo de carro.nome, modelo de carro.fabricante e modelo de carro.descrição>.~~

Detalhes de Regras:

a) Regra de período de locação = a data de término informada deve obrigatoriamente ser maior ou igual à data de registro da reserva.

Figura 42 – Fragmento *Selecionar Modelo de Carro* Personalizado para o Núcleo Operacional

A Figura 43 apresenta o fluxo básico do caso de uso *Reservar Carro* para o núcleo operacional, que foi elaborado com o auxílio de trechos de fluxos básicos originados dos seguintes fragmentos: (1) *Identificar Cliente*, (2) *Selecionar Modelo de Carro* e (3) *Obter Confirmação da Reserva*. O primeiro fragmento deu origem aos passos 2-4 do fluxo básico; o

segundo deu origem aos passos 6-11; e o último deu origem aos passos 15-19. Os passos 1, 5, 12-14 e 20 do fluxo básico não foram contemplados na personalização dos fragmentos e foram adicionados posteriormente.

Fluxo Básico:

1. O caso de uso começa quando a Recepcionista solicita o registro de uma reserva.
2. Recepcionista informa parâmetros para busca de clientes.
3. Sistema apresenta uma lista de clientes.
4. Recepcionista seleciona um cliente.
5. Recepcionista informa que o cliente é o motorista principal.
6. Sistema apresenta uma lista de locais de retirada.
7. Recepcionista seleciona um local de retirada.
8. Recepcionista informa período de locação.
9. Sistema certifica que período de locação é válido de acordo com a regra de período de locação.
10. Sistema apresenta uma lista de modelos de carro.
11. Recepcionista seleciona um modelo de carro.
12. Sistema apresenta uma lista de formas de garantia da reserva.
13. Recepcionista seleciona Depósito em Espécie como forma de garantia da reserva.
14. Executar o caso de uso “Depositar em Espécie”.
15. Recepcionista solicita o registro da reserva.
16. Sistema apresenta detalhes da reserva.
17. Recepcionista confirma registro da reserva.
18. Sistema registra a reserva.
19. Sistema apresenta recibo de registro.
20. O caso de uso termina.

Figura 43 – Fluxo Básico do Caso de Uso *Reservar Carro* para o Núcleo Operacional

4.3.3 Nova Aplicação

Caso fosse necessário produzir uma aplicação do domínio de aluguel de carros diferente das aplicações alvo e do núcleo operacional, com base no levantamento de requisitos e nas características da aplicação, seria preciso verificar se a aplicação possui os requisitos existentes na linha de produto. Para possuir uma configuração válida da linha é preciso inicialmente analisar o diagrama de *features* do domínio. Já em relação ao fluxo do negócio

em que essas *features* são implementadas, pode-se analisar o diagrama de casos de uso e o catálogo de fragmentos. Outra opção é consultar um especialista da linha de produto.

Como exemplo, é apresentada a engenharia da aplicação considerando algumas *features* existentes e que sejam válidas para a linha de produto. A Figura 44 apresenta o diagrama de *features* do domínio, onde as *features* selecionadas para a nova aplicação estão destacadas na cor cinza. De acordo com esse diagrama, a aplicação de aluguel de carros em questão é composta pela *feature* comum *Aluguel de Carros Kernel* e pelas *features* opcionais *Tipo de Tarifa*, *Extras*, *Agente*, *Cancelamento da Locação* e *Impressão de Contrato*.

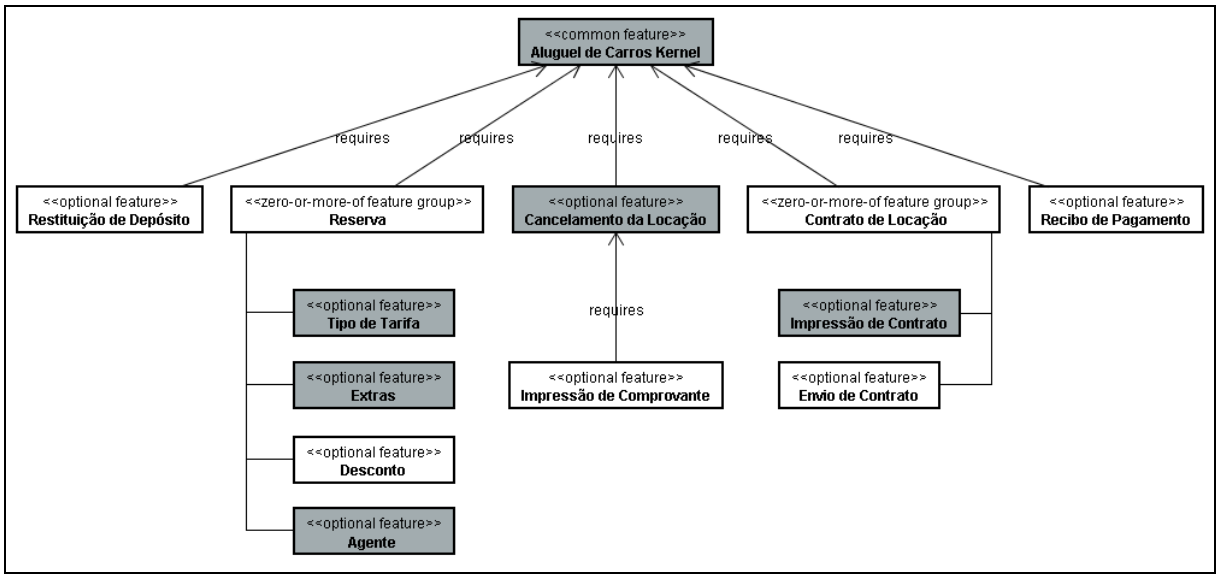


Figura 44 – *Features* Selecionadas para a Nova Aplicação

O Quadro 9 ilustra a tabela com as relações entre *features*, casos de uso e fragmentos do domínio, onde as relações selecionadas para essa aplicação estão destacadas na cor cinza. Assim como na Seção 4.3.1, por simplicidade, não serão mostradas todas as relações associadas à *feature* que representa o núcleo da linha de produto.

Quadro 9 – Relações Selecionadas para a Nova Aplicação

Feature		Caso de Uso		Fragmento
Nome	**Categoria**	**Nome**	**Categoria/ Fragmento**	**Nome**
Aluguel de Carro Kernel	Common	Reservar Carro	Kernel	
			Fragmento	Identificar Cliente
			Fragmento	Selecionar Modelo de Carro
			Fragmento	Depositar em Espécie
...	...	...	...	...
Extras	Optional	Reservar Carro	Fragmento	Adicionar Extras
Tipo de Tarifa	Optional	Reservar Carro	Fragmento	Selecionar Tipo de Tarifa
Agente	Optional	Identificar Agente	Optional	
			Fragmento	Identificar Agente
Desconto	Optional	Aplicar Desconto	Optional	
			Fragmento	Aplicar Desconto
Impressão de Contrato	Optional	Retirar Carro	Fragmento	Imprimir Contrato
Envio de Contrato	Optional	Enviar Contrato	Optional	
			Fragmento	Enviar Contrato
Recibo de Pagamento	Optional	Imprimir Recibo	Optional	
			Fragmento	Imprimir Recibo
Restituição de Depósito	Optional	Cancelar Reserva	Fragmento	Restituir Depósito
Cancelamento da Locação	Optional	Cancelar Locação	Optional	
			Fragmento	Solicitar Cancelamento da Locação
Impressão de Comprovante	Optional	Cancelar Locação	Fragmento	Imprimir Comprovante

Segundo esta tabela, a aplicação é composta por todos os casos de uso *kernel*, como o caso de uso *Reservar Carro* e os dois opcionais: *Identificar Agente* e *Cancelar Locação*. Os casos de uso *Aplicar Desconto*, *Enviar Contrato* e *Imprimir Recibo* não pertencem à aplicação. Da mesma forma, fazem parte da aplicação todos os fragmentos cujos sub-objetivos são comuns, como o fragmento *Identificar Cliente*, e alguns dos fragmentos cujos sub-objetivos são opcionais, como o fragmento *Adicionar Extras*.

A Figura 45 apresenta o diagrama de casos de uso do domínio, onde os casos de uso selecionados para essa aplicação estão destacados na cor cinza. Neste diagrama, o ator *Recepcionista* pode cadastrar um cliente, reservar um carro, cancelar uma reserva, retirar ou retornar um carro. Entretanto, ela não pode aplicar um desconto sobre o valor de uma reserva, enviar o contrato na retirada de um carro ou imprimir um recibo de pagamento. O ator *Gerente* pode realizar as mesmas funções que a *Recepcionista* pode realizar, além de poder

cancelar uma locação. O ator *Administradora de Cartão* é responsável por validar e confirmar o pagamento com cartão de crédito.

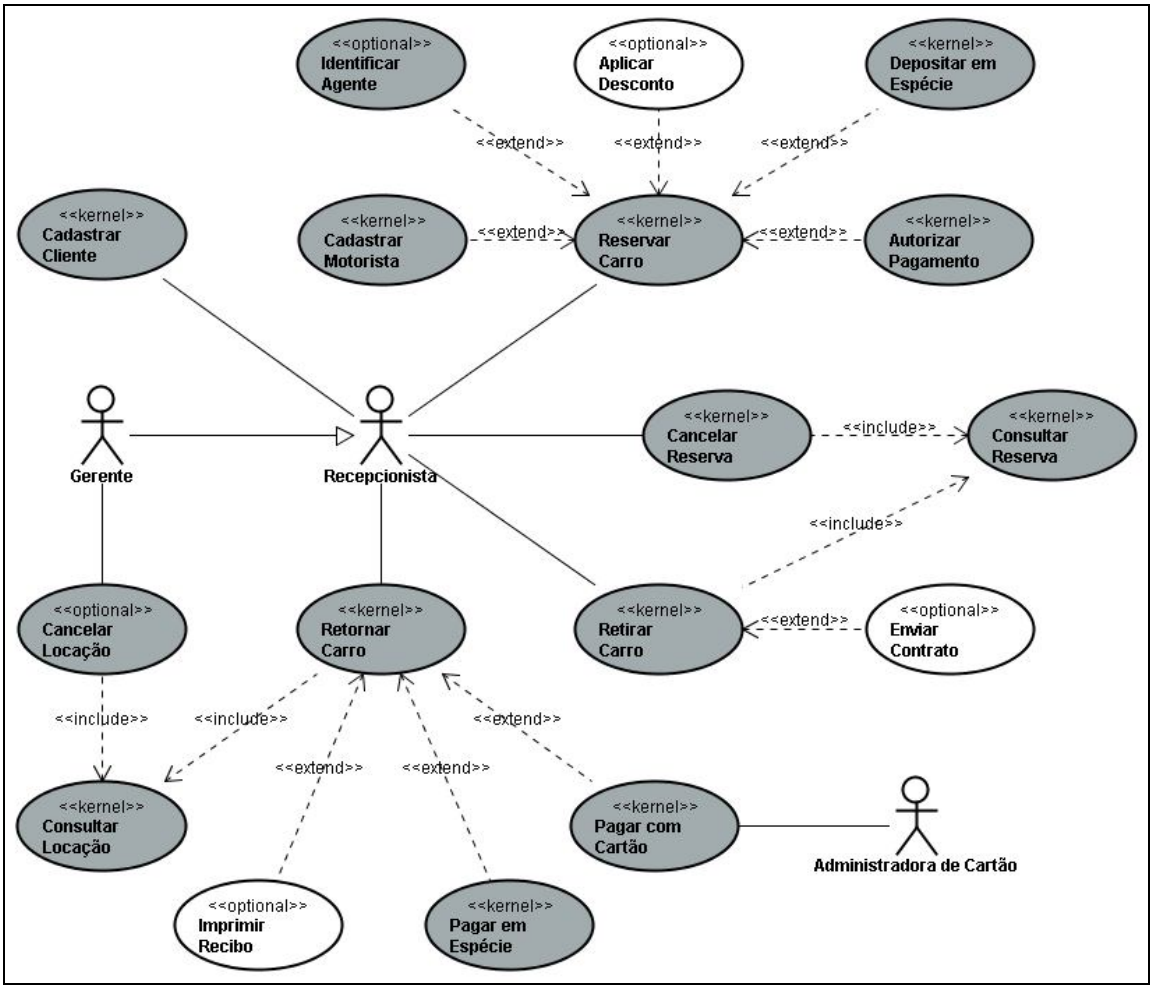


Figura 45 – Casos de Uso Selecionados para a Nova Aplicação

A Figura 46 apresenta o fragmento *Selecionar Modelo de Carro* personalizado para atender a aplicação específica em questão, onde as substituições são indicadas em negrito e as partes descartadas são indicadas com um risco. De acordo com essa personalização, a *Recepcionista* é o ator que interage com o sistema, no fluxo principal foram retirados os passos 3 e 4 e nos dados de saída foi retirada a lista de categorias de carro, tendo em vista que essa aplicação não exigirá a seleção de uma categoria de carro para a busca de modelos de carro disponíveis.

Fluxo Básico:

1. Sistema apresenta uma lista de locais de retirada.
2. **Recepcionista** seleciona um local de retirada.
3. ~~Sistema apresenta uma lista de categorias de carro.~~
4. ~~<recepcionista> seleciona uma categoria de carro.~~
5. **Recepcionista** informa período de locação.
6. Sistema certifica que período de locação é válido de acordo com a regra de período de locação.
7. Sistema apresenta uma lista de modelos de carro.
8. **Recepcionista** seleciona um modelo de carro.
9. Sistema apresenta detalhes do modelo de carro selecionado.

Fluxos Alternativos:

a) *período de locação inválido*

No passo 6 do Fluxo Básico, o Sistema identifica que o período de locação informado viola a regra de período de locação.

1. Sistema apresenta a mensagem: "A data de término informada deve obrigatoriamente ser maior ou igual à data de registro da reserva".
2. Fluxo de eventos retorna ao passo 5 do Fluxo Básico.

b) *nenhum modelo de carro satisfaz a seleção*

No passo 7 do Fluxo Básico, o Sistema não encontra nenhum modelo de carro para seleção.

1. Sistema apresenta a mensagem: "Nenhum modelo de carro encontrado".
2. Fluxo de eventos retorna ao passo 5 do Fluxo Básico.

Detalhes de Estrutura de Dados:

a) Lista de locais de retirada = lista de local.nome e endereço.cidade, ordenados alfabeticamente.

b) ~~Lista de categorias de carro = <lista de categoria de carro.nome, ordenados alfabeticamente>.~~

c) Período de locação = data/hora de início e data/hora de término da locação com a data no formato dia/mês/ano.

d) Lista de modelos de carro = lista de categoria.nome, modelo de carro.fabricante, modelo de carro.nome, ordenados alfabeticamente. Esta lista contém somente os modelos de carro que estão disponíveis para locação no local de retirada indicado, a partir da data/hora de início até a data/hora de término informada.

e) Detalhes do modelo de carro = categoria de carro.nome, modelo de carro.nome, modelo de carro.fabricante e modelo de carro.descrição.

Detalhes de Regras:

a) Regra de período de locação = a data de término informada deve obrigatoriamente ser maior ou igual à data de registro da reserva.

Figura 46 – Fragmento *Selecionar Modelo de Carro* Personalizado para a Nova Aplicação

A Figura 47 apresenta o fluxo básico do caso de uso *Reservar Carro* para a nova aplicação, que foi elaborado com o auxílio de trechos de fluxos básicos originados dos seguintes fragmentos: (1) *Identificar Cliente*, (2) *Selecionar Modelo de Carro*, (3) *Adicionar Extras*, (4) *Selecionar Tipo de Tarifa* e (5) *Obter Confirmação da Reserva*. O primeiro

fragmento deu origem aos passos 2-4 do fluxo básico; o segundo deu origem aos passos 6-11; o terceiro deu origem aos passos 12-15; o quarto deu origem aos passos 16-20; e o último deu origem aos passos 24-28. Os passos 1, 5, 21-23 e 29 do fluxo básico não foram contemplados na personalização dos fragmentos e foram adicionados posteriormente.

Fluxo Básico:

1. O caso de uso começa quando a Recepcionista solicita o registro de uma reserva.
2. Recepcionista informa parâmetros para busca de clientes.
3. Sistema apresenta uma lista de clientes.
4. Recepcionista seleciona um cliente.
5. Recepcionista informa que o cliente é o motorista principal.
6. Sistema apresenta uma lista de locais de retirada.
7. Recepcionista seleciona um local de retirada.
8. Recepcionista informa período de locação.
9. Sistema certifica que período de locação é válido de acordo com a regra de período de locação.
10. Sistema apresenta uma lista de modelos de carro.
11. Recepcionista seleciona um modelo de carro.
12. Sistema apresenta uma lista de extras.
13. Recepcionista seleciona um ou mais extras.
14. Recepcionista confirma adição de extras.
15. Sistema atualiza valor da reserva.
16. Sistema apresenta uma lista de tipos de tarifa.
17. Recepcionista seleciona um tipo de tarifa.
18. Sistema apresenta novo valor da reserva.
19. Recepcionista confirma seleção da tarifa.
20. Sistema atualiza valor da reserva.
21. Sistema apresenta uma lista de formas de garantia da reserva.
22. Recepcionista seleciona Depósito em Espécie como forma de garantia da reserva.
23. Executar o caso de uso “Depositar em Espécie”.
24. Recepcionista solicita o registro da reserva.
25. Sistema apresenta detalhes da reserva.
26. Recepcionista confirma registro da reserva.
27. Sistema registra a reserva.
28. Sistema apresenta recibo de registro.
29. O caso de uso termina.

Figura 47 – Fluxo Básico do Caso de Uso *Reservar Carro* para a Nova Aplicação

4.4 Discussão sobre o Exemplo de Aplicação

Na engenharia do domínio, a modelagem de objetivos proporcionou uma maneira natural de identificar os requisitos do domínio de cada aplicação alvo, capturando os diversos caminhos pelos quais os usuários do sistema podem seguir para atingir seus objetivos. Tais objetivos foram recursivamente decompostos em sub-objetivos, auxiliando no entendimento do domínio do qual fazem parte.

A análise de similaridades e variabilidades entre os membros da linha de produto através da utilização de matrizes permitiu a identificação de quais objetivos e seus respectivos sub-objetivos são comuns e variáveis de maneira clara e objetiva. Entretanto, o domínio estudado foi limitado em função da análise ter considerado apenas a interface das aplicações alvo. Desta forma, outras possíveis funcionalidades, como por exemplo, a análise de crédito para o aluguel de carro, não foram contempladas no exemplo apresentado.

A lista de objetivos e sub-objetivos do domínio foi fundamental para a identificação dos casos de uso da linha de produto e de seus relacionamentos, orientando a construção do diagrama de casos de uso. Além disso, foi possível elaborar um catálogo de fragmentos de casos de uso específicos do domínio, a partir dos sub-objetivos identificados e das simulações de uso das aplicações alvo. Por último, *features* funcionais da linha de produto e seus relacionamentos foram extraídos do modelo de casos de uso do domínio, que neste trabalho é composto pelo diagrama de casos de uso e pelo catálogo de fragmentos.

Na engenharia da aplicação, a seleção das *features* indicou com precisão quais casos de uso e fragmentos fazem parte da aplicação. Isso foi possível devido às relações existentes entre *features*, casos de uso e fragmentos. As descrições textuais dos casos de uso foram elaboradas com o auxílio do catálogo de fragmentos, através da seleção, personalização e composição desses artefatos. Entretanto, nem todos os passos dos casos de uso foram

contemplados com esta abordagem, como por exemplo, os passos relativos ao início e fim dos casos de uso, assim como chamadas de execução de casos de uso *extends* ou *includes*.

Neste contexto, pode-se destacar como ponto positivo que os fragmentos de casos de uso aparecem como um elemento intermediário entre *features* e especificações textuais de casos de uso. Uma vez que cada fragmento está associado a um sub-objetivo específico do domínio, esse artefato pode ser mais facilmente personalizado do que uma descrição textual completa de caso de uso, normalmente na forma de um *template*, pois o trecho associado ao fragmento é menor. A combinação destas características permitiu instanciar as especificações de casos de uso do domínio mais rapidamente e alinhadas às características da aplicação em desenvolvimento.

Finalmente, pode-se dizer que a abordagem de elaboração de especificações de casos de uso proposta estabelece uma ponte entre a análise de similaridades e variabilidades do domínio e a especificação de casos de uso para linhas de produto, contribuindo com o conhecimento atualmente disponível nessa área.

5 CONCLUSÃO

O desenvolvimento de sistemas de informação tem mudado nas últimas décadas. A demanda por sistemas mais complexos e com melhor qualidade tem aumentado na mesma proporção em que a indústria de software eleva a sua capacidade de produção. Assim, a organização que se destaca entre os concorrentes é aquela que reage rapidamente às mudanças do mercado e aos requisitos do usuário.

O reúso de software na construção de novos sistemas de informação permite lidar com essa mudança e apresenta-se como um importante aliado para reduzir o esforço necessário para o desenvolvimento e manutenção desses novos sistemas. Neste contexto, as Linhas de Produto de Software (LPS) inserem-se estabelecendo um arcabouço de reúso de software que gera benefícios maiores que os das abordagens de reúso tradicionais, pois permitem não só o reúso de componentes individuais de software, como também o de requisitos e arquiteturas.

A técnica de casos de uso tem ganhado uma ampla aceitação dentre as abordagens disponíveis para a especificação de requisitos de sistemas de informação. Os casos de uso capturam os requisitos funcionais do sistema através de descrições textuais que devem especificar todas as interações entre o sistema e seus usuários. Em um contexto de LPS, a variação do comportamento dos usuários e do sistema durante a troca de informações entre ambos deve ser considerada na elaboração de especificações de casos de uso.

Considerando a inserção da técnica de casos de uso em linhas de produto, a presente dissertação apresentou uma abordagem para a elaboração de especificações de casos de uso para LPS, que abrange desde a análise de similaridades e variabilidades do domínio até a escrita de descrições textuais de casos de uso. Tal abordagem foi inserida nas duas atividades essenciais de uma LPS: engenharia do domínio e da aplicação. Na primeira atividade, um diagrama de casos de uso, um catálogo de fragmentos de casos de uso e um diagrama de

features são elaborados. Na segunda atividade, o diagrama de casos de uso é instanciado e os fragmentos são selecionados e personalizados para uma aplicação específica. As descrições textuais dos casos de uso são produzidas através da composição desses fragmentos.

5.1 Comparações com Outras Abordagens

Conforme apresentado na Seção 2.4, existem diversas abordagens que introduziram o conceito de casos de uso para Linhas de Produto de Software. O Quadro 10 apresenta um sumário da atividade de especificação de casos de uso para LPS nessas abordagens.

Quadro 10 – Sumário da Especificação de Casos de Uso para LPS nas Abordagens Existentes

Característica	Moon *et al.* (2005)	Kim *et al.* (2006)	John e Muthig (2002)	Fantechi *et al.* (2004)	Eriksson *et al.* (2005)	Bragança e Machado (2007)	Bonifácio e Borba (2009)
Análise de similaridades e variabilidades do domínio.	x			x			
Auxílio à elaboração de diagramas de casos de uso.	x	x					
Auxílio à elaboração de descrições textuais de casos de uso.		x		x			
Representação da variabilidade em diagramas de casos de uso.	x	x	x			x	
Representação da variabilidade em descrições textuais de casos de uso.		x	x	x	x	x	x
Auxílio à instanciação de especificações de casos de uso para uma dada aplicação.					x	x	x

A coluna *Característica* apresenta as características consideradas nessa comparação: (1) análise de similaridades e variabilidades do domínio; (2) auxílio à elaboração de diagramas de casos de uso; (3) auxílio à elaboração de descrições textuais de casos de uso; (4) representação da variabilidade em diagramas de casos de uso; (5) representação da variabilidade em descrições textuais de casos de uso; e por último, (6) auxílio à instanciação de especificações de casos de uso para uma dada aplicação. As demais colunas referenciam as

abordagens analisadas, que foram identificadas pelos nomes dos autores. As células marcadas com ‘x’ indicam que a característica em questão está presente na abordagem analisada.

Segundo este quadro comparativo, a maioria das abordagens analisadas enfatizou a representação da variabilidade nas especificações de casos de uso e uma minoria focou na análise de similaridades e variabilidades do domínio e também no auxílio à elaboração e instanciação de especificações de casos de uso. Os problemas de identificação de casos de uso comuns e variáveis entre os membros da linha de produto e de elaboração de descrições genéricas de casos de uso que sejam de fácil utilização durante a especificação de requisitos de uma dada aplicação permeiam várias das propostas analisadas. Neste sentido, a construção de uma abordagem que auxilie a especificação e facilite a instanciação de casos de uso no contexto de LPS é um dos caminhos disponíveis para o tratamento desses problemas.

A abordagem proposta nesta dissertação está inserida nesse cenário e possui como foco principal auxiliar a instanciação de especificações textuais de casos de uso para uma dada aplicação. Para isso, foi realizada uma análise de similaridades e variabilidades do domínio, foram definidas diretrizes para auxiliar a elaboração das especificações de casos de uso e foram adotadas notações para representar a variabilidade nessas descrições.

5.2 Contribuições

Uma das principais contribuições da abordagem proposta foi estabelecer um auxílio à geração de descrições textuais de casos de uso para uma aplicação específica da linha de produto, usando um catálogo de fragmentos. Essa estratégia permitiu uma instanciação semi-automática dos textos de casos de uso a partir das características da aplicação durante a especificação de requisitos de um dado membro da LPS.

Além disso, a abordagem apresenta uma modelagem de requisitos para LPS cujo foco principal é auxiliar a elaboração das especificações de casos de uso do domínio a partir da

análise das aplicações alvo. Nesta abordagem, o modelo de casos de uso da linha de produto é composto por um diagrama e um catálogo de fragmentos que são originados a partir do conjunto de objetivos e sub-objetivos do domínio.

Por fim, destaca-se a elaboração dos requisitos de uma LPS para o domínio de aluguel de carros a partir da análise das similaridades e variabilidades existentes de quatro aplicações desse domínio, onde foi elaborado um catálogo de fragmentos de casos de uso. Além de permitir que haja um maior conhecimento sobre o domínio considerado, os artefatos produzidos podem ser usados para ensino em cursos de graduação e pós-graduação referentes ao desenvolvimento de LPS. Os estudantes podem realizar trabalhos tais como instanciar os requisitos das aplicações da linha de produto e implementar novas variabilidades na mesma.

5.3 Limitações e Trabalhos Futuros

O uso da abordagem no exemplo de aplicação apresentado serviu para mostrar a utilização prática da proposta. Tal exemplo permitiu a análise da forma de elaboração e instanciação das especificações de casos de uso para linhas de produto e serviu também para a identificação de algumas limitações da abordagem e da solução desenvolvida.

Dentre as limitações identificadas, destacam-se: as descrições textuais de casos de uso não incluem pré e pós-condições que possam existir no domínio; a análise de similaridades e variabilidades voltada para a modelagem conceitual do domínio e para regras de negócio não é abordada; possíveis regras para a composição de *features* não são contempladas por esta abordagem; e questões relacionadas à evolução dos artefatos da LPS, mais especificamente, os fragmentos de casos de uso, não são tratadas. Além disso, os textos obtidos através da composição dos fragmentos podem não cobrir todas as especificações textuais de casos de uso, cabendo ao engenheiro de requisitos completar tais descrições posteriormente.

Melhorias na solução proposta certamente facilitarão o uso da abordagem. Dentre as características que podem ser desenvolvidas, destacam-se: suporte a modelagem conceitual do domínio e modelagem de objetivos; auxílio ao rastreamento dos casos de uso e fragmentos a partir das *features*; auxílio à evolução dos artefatos da linha de produto; gerenciamento dos fragmentos (inclusão, alteração, exclusão e busca); suporte a elaboração das especificações textuais de casos de uso a partir dos fragmentos selecionados (personalização, composição e completude), entre outras.

Na abordagem apresentada, as variações dos detalhes de estrutura de dados e de regras de negócio poderiam ser identificadas e representadas nas descrições textuais de casos de uso. Um estudo específico pode ser realizado para a avaliação dessas questões e da viabilidade da abordagem proposta em um processo de desenvolvimento de linhas de produto.

Finalmente, considerando-se as limitações apresentadas e as perspectivas de novas pesquisas, apontam-se como trabalhos futuros:

- Desenvolvimento de ferramentas para apoiar o gerenciamento dos fragmentos e a produção dos textos de casos de uso a partir desses artefatos;
- Tratamento da evolução dos artefatos existentes na abordagem proposta, devido a uma possível incorporação de novas aplicações alvo na LPS;
- Utilização da abordagem proposta para outros domínios;
- Outros trabalhos já realizados na classificação e seleção de componentes poderiam ser aplicados à abordagem proposta;
- Integração da abordagem proposta com processos de desenvolvimento de sistemas de informação baseados em LPS.
- Especificação de abordagens para a modelagem conceitual do domínio e para a modelagem de regras de negócio para linhas de produto.

REFERÊNCIAS BIBLIOGRÁFICAS

ATKINSON, C. *et al*. **Component-based product line engineering with UML**. Boston: Addison-Wesley, 2002.

ARAÚJO, D. O. *et al*. Elaboração de especificações de casos de uso para linhas de produto de software baseada em fragmentos. Anais do III SIMPÓSIO BRASILEIRO DE COMPONENTES, ARQUITETURAS E REUTILIZAÇÃO DE SOFTWARE, 3., 2009. Natal. **Anais ...** , Natal: SBC, 2009. p. 24-37.

BAYER, J. *et al*. PuLSE: a methodology to develop software product lines. In: ACM SIGSOFT SYMPOSIUM ON SOFTWARE REUSABILITY, 5., 1999, Los Angeles. **Proceedings …** New York, 1999. p. 122-131.

BERGEY, J. *et al*. **Fourth DoD product line practice Workshop Report**. Pittsburg: Software Engineering Institute, Carnegie Mellon University, 2001. (Technical Report, CMU/SEI-2001-TR-017, ESC-TR 2001-017).

BERTOLINO, A. *et al*. Product line use cases: scenario-based specification and testing of requirements. In: KAKOLA, T. ; DUEÑAS, J. C. (Ed.). **Software product lines**. Berlin: Springer, 2006. p. 425-445. (Computer Science).

BIRK, A. *et al*. **Report of the GI Work Group Requirements Engineering for Product Lines**, Munich: Fraunhofer , 2003. IESE Eprints. (Technical Report).

BOEHM, B. A Spiral model of software development and enhancement. **ACM Software Engineering Note**, New York, v. 11, n. 4, p. 22-42, 1986.

BONIFÁCIO, R. ; BORBA, P. Modeling scenario variability as crosscutting mechanisms. In: ACM INTERNATIONAL CONFERENCE ON ASPECT-ORIENTED SOFTWARE DEVELOPMENT, 8., 2009, Charlottesville, Virginia. **Proceedings …** New York, 2009. p. 125-136

BOOCH, G. **Object-oriented analysis and design with applications**. 2 ed. Reading: Addison-Wesley, 1994.

BOOCH, G. ; RUMBAUGH, J. ; JACOBSON, I. **The Unified modeling language user guide**. 2 ed. Reading: Addison-Wesley, 2005.

BOSCH, J. **Design and use of software architectures**: adopting and evolving a product line approach. Reading: Addison-Wesley, 2000.

BRAGANÇA, A. ; MACHADO, R. J. Extending UML 2.0 metamodel for complementary usages of <<extend>> relationship within use case variability specification. In: INTERNATIONAL CONFERENCE ON SOFTWARE PRODUCT LINES. 10., 2006, Baltimore, **Proceedings …** Washington: IEEE , 2006. p. 123-130.

_____. Automating mappings between use case diagrams and feature models for software product lines. In: INTERNATIONAL CONFERENCE ON SOFTWARE PRODUCT LINES. 11., 2007. Kyoto, **Proceedings …** Los Alamitos: IEEE, 2007, p. 3-12.

CLEMENTS, P. ; NORTHROP, L. **Software product lines**: practices and patterns. Boston: Addison-Wesley, 2002.

COCKBURN, A. **Writing effective use cases**. Reading: Addison-Wesley, 2001.

COX, K. ; PHALP, K. Replicating the CREWS use case authoring guidelines experiment. **Empirical Software Engineering Journal**, Chicago, v. 5, n. 3, p. 245-267, 2000.

DAVIS, S. M. **Future perfect**, Boston: Addison-Wesley, 1987.

DIAS, F. G. *et al*. Elaboration of use case specifications: an approach based on use case fragments. In: ANNUAL ACM SYMPOSIUM ON APPLIED COMPUTING, 23., 2008, Fortaleza. REQUIREMENTS ENGINEERING TRACK, 1., 2008, Fortaleza. **Proceedings ...** New York: ACM, 2008. p. 614-618, 2008.

DIAS, F. G. **Elaboração de requisitos de software**: uma abordagem baseada em fragmentos de casos de uso. 2008. Dissertação (Mestrado em Informática) – Instituto de Matemática, Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2008.

DIJKSTRA, E. W. Notes on structured programming. In: DAHL, O. J. *et al*. **Structured programming**, London: Academic Press, 1972. (A.P.I.C. Studies in Data Processing, 8).

DONEGAN, P. M. **Geração de famílias de produtos de software com arquitetura baseada em componentes**. 2008. Dissertação (Mestrado em Ciências de Computação e Matemática Computacional) – Universidade de São Paulo, São Carlos, 2008.

DORFMAN, M. ; THAYER, R. H. **Software requirements engineering**. Los Alamitos: IEEE Computer Society Press, 1997.

ERIKSSON, M. ; BÖRSTLER, J. ; BORG, K. The PLUSS approach - domain modeling with features, use cases and use case realizations. In: OBBINK, H. ; POHL, K. (Eds). **Software Product Lines**. Berlin: Springer-Verlag, 2005. p. 33-44, 2005. 9th International Conference, SPLC 2005. (Lecture Notes in Computer Science, vol. 3714).

ERIKSSON, H. ; PENKER, M. **Business modeling with UML:** business patterns at work. New York: John Wiley & Sons, 2000. 459 p.

FANTECHI, A. *et al.* A methodology for the derivation and verification of use cases for product lines. In: NORD, R. L. (Ed.). **Software Product Lines**. Berlin: Springer-Verlag, 2004. p. 255-265. 3rd International Conference, SPLC 2004. (Lecture Notes in Computer Science, vol. 3154).

GOMAA, H. **Designing software product lines with UML: from use cases to pattern-based software architectures**. Boston: Addison-Wesley, 2004.

GRISS, M. ; FAVARO, J. ; D'ALESSANDRO, M. Integrating feature modeling with the RSEB. INTERNATIONAL CONFERENCE ON SOFTWARE REUSE, 5., 1998. Victoria, Canada. **Proceedings …** Los Alamitos: IEEE Computer Society Press, 1998. p. 76-85.

GRISS, M. **Product-line architectures**. Component-based software engineering: putting the pieces together. Boston: Addison-Wesley, 2001. p. 405-420.

HALMANS, G. ; POHL, K. Communicating the variability of a software-product family to customers. **Software and Systems Modeling**, Berlin, v. 2, n. 1, p. 15-36, Mar. 2003.

IEEE - Institute of Electrical and Electronic Engineers. **IEEE Standard glossary of software engineering terminology**. New York, 1990. (IEEE Std 610.12-1990).

JACOBSON, I. ; GRISS, M. ; JONSSON, P. **Software reuse**: architecture, process, and organization for business success. Reading: Addison-Wesley, 1997.

JACOBSON, I. ; BOOCH, G. ; RUMBAUGH, J. **The Unified software development process**. Boston: Addison-Wesley, 1999. 512 p.

JACOBSON, I. ; NG, P. W. **Aspect-oriented software development with use cases**. Reading: Addison-Wesley, 2004.

JOHN, I. ; MUTHIG, D. Product line modeling with generic use cases. In: WORKSHOP ON TECHNIQUES FOR EXPLOITING COMMONALITY THROUGH VARIABILITY MANAGEMENT, 2002, San Diego. **Proceedings …** San Diego: SEI, 2002.

KANG, K. *et al*. **Feature-oriented domain analysis (FODA) feasibility study**. Pittsburgh: Software Engineering Institute, Carnegie Mellon University, 1990. (CMU/SEI-90-TR-021, ADA235785).

KIM, J. ; KIM, M. ; PARK, S. Goal and scenario based domain requirements analysis environment. **Journal of System and Software**, New York, v. 79, n. 7, p. 926-938, Jul. 2006.

KOTONYA, G. ; SOMMERVILLE, I. **Requirements engineering**: processes and techniques. New York: John Wiley & Sons 1998.

KRUEGER, C. W. Software reuse. **ACM Computing Surveys**, New York, v. 24, n. 2, p. 131-183, Jun. 1992.

KRUEGER, C. Easing the transaction to software mass customization. In: INTERNATIONAL WORKSHOP SOFTWARE PRODUCT-FAMILY ENGINEERING, 4., 2002, Bilbao. **Proceedings …** Berlim: Springer, 2002. p. 282-293.

KULAK, D. ; GUINEY, E. **Use cases**: requirements in context. 2 ed. Reading: Addison-Wesley, 2003.

McGREGOR, J. D. *et al*. Initiating software product lines. **IEEE Software**, New York, v. 19, n. 4, p. 24-27, Jul. 2002.

MCILROY, M. D. Mass produced software components. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGEENERING, 1., 1968, Garmisch Pattenkirchen: **Proceedings …** Brussels, NATO Science Committee, 1968.

MEYERS, B. C. ; OBERNDORF, P. **Managing software acquisition**. Open systems and COTS products. Reading: Addison-Wesley, 2002. (SEI Series in Software Engineering).

MOON, M .; YEOM, K. ; CHAE, H. S. An Approach to developing domain requirements as a core asset. **IEEE Transactions on Software Engineering**, New York, v. 31, n. 7, p. 551-569, Jul., 2005.

ÖVERGAARD, G. ; PALMKVIST, K. **Use cases**: patterns and blueprints. Reading: Addison-Wesley, 2004.

PARNAS, D. L. On the design and development of program families. **IEEE Transactions on Software Engineering**, New York, v. 2, n. 1, p. 1-9, Mar. 1976.

POHL, K. ; BOCKLE, G. ; VAN DER LINDEN, F. **Software product line engineering**: foundations, principles, and techniques. Berlin: Springer, 2005.

PRIETO-DÍAZ, R. Classifying software for reuse, **IEEE Software**, New York, v. 4, n. 1, p. 6-16, Jan. 1987.

ROBERTS, D. ; JOHNSON, R. Evolving frameworks: a pattern language for developing object-oriented frameworks. In: CONFERENCE ON PATTERN LANGUAGES AND PROGRAMMING, 3., 1996, Monticello, IL. **Proceedings …** Monticello, 1996. v. 3, p. 15.

ROSENBERG, D. ; STEPHENS, M. **Use case driven object modeling with UML**. New York: Apress, 2007. 438 p.

RUMBAUGH, J. OMT: the development process. **Journal of Object Oriented Programming**, Brentwood, v. 8, n. 2, p. 8-16, May, 1995.

SCHNEIDER, G, ; WINTERS, J. **Applying use cases**: a practical guide. 2 ed. Reading: Addison Wesley Professional, 2001.

SEI – SOFTWARE ENGINEERING INSTITUTE. **A Framework for software product line practice**. Pittsburgh. Disponível em: <http://www.sei.cmu.edu/productlines/frame_report/index.html>, acesso em 06 jan. 2010, 16:30:00.

SEMMAK, F. ; BRUNET, J. Variability in goal-oriented domain requirements. In: MORISIO, M. (Ed). **Reuse of off-the shelf components**. Heidelberg: Springer, 2006. p. 390-394. 9th International Conference on Software Reuse. (Lecture Notes in Computer Science, 4039).

SOMMERVILLE, I. ; SAWYER, P. **Requirements engineering**: a good practice guide. New York: John Wiley & Sons, 1997.

SVETINOVIC, D. *et al*. Unified use case statecharts: case studies. **Requirements Engineering,** London, v. 12, n. 4, p. 245-264, Oct. 2007.

TRIGAUX, J. C. ; HEYMANS, P. **Modelling variability requirements in software product lines: a comparative survey**. Namur: FUNDP - Equipe LIEL Institut d'Informatique, 2003.

VAN DER LINDEN, F. ; SCHMID, K. ; ROMMES E. **Software product lines in action**. The best industrial practice in product line engineering. Berlin: Springer, 2007.

VAN OMMERING, R. ; BOSCH, J. Components in product-line architectures. In: CRNKOVIC, I. ; LARSSON, M. (Ed.). **Building reliable component-based software systems**. Norwood: Artech House, 2002. p. 207-221.

VASCONCELOS, A. **Uma abordagem de apoio à criação de arquiteturas de referência de domínio baseada na análise de sistemas legados**. 2007. Tese (Doutorado em Engenharia de Sistemas e Computação) - COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2007.

WEISS, D. ; LAI, C. **Software product-line engineering**: a family-based software development process. Boston: Addison-Wesley, 1999. 448 p.

YU, Y. *et al*. Configuring features with stakeholder goals. In: ACM SYMPOSIUM ON APPLIED COMPUTING PROCEEDINGS, 23., 2008, Fortaleza. **Proceedings …** New York, 2008. p. 645-649.

APÊNDICE A – CATÁLOGO DE FRAGMENTOS DE CASOS DE USO

Este apêndice apresenta parte do catálogo de fragmentos de casos de uso elaborado para o domínio de aluguel de carros do exemplo de aplicação, que foi apresentado no Capítulo 4. Tais fragmentos estão relacionados somente ao objetivo *Reservar Carro*.

A.1 Fragmento Identificar Cliente

Nome do Fragmento: Identificar Cliente
Sub-objetivo: Identificar um dado cliente.
Ordem de Utilização: *Não há.*
Fluxo Básico:
1. <recepcionista> informa parâmetros para busca de clientes.
2. Sistema apresenta uma lista de clientes.
3. <recepcionista> seleciona um cliente.
4. Sistema apresenta detalhes do cliente selecionado.

Passos opcionais: 4.
Fluxos Alternativos:
a) *nenhum cliente satisfaz a busca*
No passo 2 do Fluxo Básico, o Sistema não encontra nenhum cliente para seleção.
1. Sistema apresenta a mensagem: “Nenhum cliente encontrado”.
2. Fluxo de eventos retorna ao passo 1 do Fluxo Básico.

Detalhes de Estrutura de Dados:
a) Parâmetros para busca de clientes = <nome, sobrenome, licença, telefone, código postal ou empresa>.
b) Lista de clientes = <lista de cliente.nome, cliente.sobrenome e cliente.empresa, ordenados alfabeticamente. Esta lista contém somente os clientes que possuem algum parâmetro informado para busca>.
c) Detalhes do cliente = <cliente.nome, cliente.sobrenome, cliente.data de nascimento, cliente.sexo, cliente.nacionalidade, cliente.passaporte, cliente.e-mail, cliente.telefone, cliente.empresa, endereço.logradouro, endereço.cidade, endereço.estado, endereço.país, endereço.código postal, licença.número, licença.categoria, licença.data de validade, licença.país de emissão, cartão.número, cartão.código, cartão.bandeira, cartão.nome do titular e cartão.data de validade>.

Detalhes opcionais: c.
Detalhes de Regras:
Não há.

Figura 48 – Fragmento de Caso de Uso *Identificar Cliente*

A.2 Fragmento Selecionar Modelo de Carro

Nome do Fragmento: Selecionar Modelo de Carro
Sub-objetivo: Selecionar um modelo de carro para a reserva.
Ordem de Utilização: Após Buscar Cliente.
Fluxo Básico:

1. Sistema apresenta uma lista de locais de retirada.
2. <recepcionista> seleciona um local de retirada.
3. Sistema apresenta uma lista de categorias de carro.
4. <recepcionista> seleciona uma categoria de carro.
5. <recepcionista> informa período de locação.
6. Sistema certifica que período de locação é válido de acordo com a regra de período de locação.
7. Sistema apresenta uma lista de modelos de carro.
8. <recepcionista> seleciona um modelo de carro.
9. Sistema apresenta detalhes do modelo de carro selecionado.

Passos opcionais: 3, 4, 9.
Fluxos Alternativos:
a) *período de locação inválido*
No passo 6 do Fluxo Básico, o Sistema identifica que o período de locação informado viola a regra de período de locação.

1. Sistema apresenta a mensagem: “A data de término informada deve obrigatoriamente ser maior ou igual à data de registro da reserva”.
2. Fluxo de eventos retorna ao passo 5 do Fluxo Básico.

b) *nenhum modelo de carro satisfaz a seleção*
No passo 7 do Fluxo Básico, o Sistema não encontra nenhum modelo de carro para seleção.

1. Sistema apresenta a mensagem: “Nenhum modelo de carro encontrado”.
2. Fluxo de eventos retorna ao passo 5 do Fluxo Básico.

Detalhes de Estrutura de Dados:

a) Lista de locais de retirada = <lista de local.nome e endereço.cidade, ordenados alfabeticamente>.
b) Lista de categorias de carro = <lista de categoria de carro.nome, ordenados alfabeticamente>.
c) Período de locação = <data/hora de início e data/hora de término da locação com a data no formato dia/mês/ano>.
d) Lista de modelos de carro = <lista de categoria.nome, modelo de carro.fabricante e modelo de carro.nome, ordenados alfabeticamente. Esta lista contém somente os modelos de carro que estão disponíveis para locação no local de retirada indicado, a partir da data/hora de início até a data/hora de término informada>.
e) Detalhes do modelo de carro = <categoria de carro.nome, modelo de carro.nome, modelo de carro.fabricante e modelo de carro.descrição>.

Detalhes opcionais: b, e.
Detalhes de Regras:

a) Regra de período de locação = <a data de término informada deve obrigatoriamente ser maior ou igual à data de registro da reserva>.

Figura 49 – Fragmento de Caso de Uso *Selecionar Modelo de Carro*

A.3 Fragmento Adicionar Extras

Nome do Fragmento: Adicionar Extras
Sub-objetivo: Adicionar extras à reserva.
Ordem de Utilização: Após Selecionar Modelo de Carro.
Fluxo Básico:
1. Sistema apresenta uma lista de extras.
2. <recepcionista> seleciona um ou mais extras.
3. Sistema apresenta detalhes dos extras selecionados.
4. <recepcionista> confirma adição de extras.
5. Sistema atualiza valor da reserva.

Passos opcionais: 3.
Fluxos Alternativos:
a) *nenhum extra satisfaz a seleção*
No passo 1 do Fluxo Básico, o Sistema não encontra nenhum extra para seleção.
1. Sistema apresenta a mensagem: “Nenhum extra encontrado”.
2. Fluxo de eventos retorna ao passo 1 do Fluxo Básico.

b) *adição de extras cancelada*
No passo 4 do Fluxo Básico, <recepcionista> decide cancelar a adição de extras.
1. Sistema apresenta a mensagem: “Adição de extras cancelada”.
2. Fluxo de eventos retorna ao passo 1 do Fluxo Básico.

Detalhes de Estrutura de Dados:
a) Lista de extras = <lista de extra.nome e extra.valor, ordenados alfabeticamente. Esta lista contém somente os extras que podem ser adicionados à reserva corrente>.
b) Detalhes de extras = <extra.nome, extra.valor e extra.descrição>.

Detalhes opcionais: b.
Detalhes de Regras:
Não há.

Figura 50 – Fragmento de Caso de Uso *Adicionar Extras*

A.4 Fragmento Selecionar Tipo de Tarifa

Nome do Fragmento: Selecionar Tipo de Tarifa
Sub-objetivo: Selecionar um tipo de tarifa para a reserva.
Ordem de Utilização: Após Selecionar Modelo de Carro ou Adicionar Extras.
Fluxo Básico:
1. Sistema apresenta uma lista de tipos de tarifa.
2. <recepcionista> seleciona um tipo de tarifa.
3. Sistema apresenta novo valor da reserva.
4. <recepcionista> confirma seleção do tipo de tarifa.
5. Sistema atualiza valor da reserva.

Fluxos Alternativos:
a) *seleção do tipo de tarifa cancelada*
No passo 4 do Fluxo Básico, <recepcionista> decide cancelar a seleção do tipo de tarifa.
1. Sistema apresenta a mensagem: "Seleção do tipo de tarifa cancelada".
2. Fluxo de eventos retorna ao passo 1 do Fluxo Básico.

Detalhes de Estrutura de Dados:
a) Lista de tipos de tarifa = <lista de tarifa.nome, ordenados alfabeticamente>.
Detalhes de Regras:
Não há.

Figura 51 – Fragmento de Caso de Uso *Selecionar Tipo de Tarifa*

A.5 Fragmento Aplicar Desconto

Nome do Fragmento: Aplicar Desconto
Sub-objetivo: Aplicar desconto sobre o valor da reserva.
Ordem de Utilização: Após Selecionar Modelo de Carro ou Adicionar Extras ou Selecionar Tipo de Tarifa.
Fluxo Básico:
1. Sistema apresenta uma lista de descontos.
2. <recepcionista> seleciona um desconto.
3. Sistema apresenta novo valor da reserva.
4. <recepcionista> confirma aplicação do desconto.
5. Sistema atualiza valor da reserva.

Fluxos Alternativos:
a) *aplicação do desconto cancelada*
No passo 4 do Fluxo Básico, <recepcionista> decide cancelar a aplicação do desconto.
1. Sistema apresenta a mensagem: “Aplicação do desconto cancelada”.
2. Fluxo de eventos retorna ao passo 1 do Fluxo Básico.

Detalhes de Estrutura de Dados:
a) Lista de descontos = <lista de desconto.código e desconto.percentual, ordenados alfabeticamente>.

Detalhes de Regras:
Não há.

Figura 52 – Fragmento de Caso de Uso *Aplicar Desconto*

A.6 Fragmento Cadastrar Motorista

Nome do Fragmento: Cadastrar Motorista
Sub-objetivo: Cadastrar um motorista para a reserva.
Ordem de Utilização: Após Selecionar Modelo de Carro ou Adicionar Extras ou Selecionar Tipo de Tarifa ou Aplicar Desconto.
Fluxo Básico:

1. <recepcionista> informa dados do motorista.
2. Sistema certifica que dados do motorista são válidos de acordo com a regra de motorista.
3. Sistema apresenta detalhes do motorista.
4. <recepcionista> confirma cadastro do motorista.
5. Sistema apresenta mensagem de sucesso.

Fluxos Alternativos:

a) *dados do motorista inválidos*

No passo 2 do Fluxo Básico, o Sistema identifica que os dados do motorista violam a regra de motorista.

1. Sistema apresenta a mensagem: “Os dados do motorista são inválidos”.
2. Fluxo de eventos retorna ao passo 1 do Fluxo Básico.

b) *cadastro do motorista cancelado*

No passo 4 do Fluxo Básico, <recepcionista> decide cancelar o cadastro do motorista.

1. Sistema apresenta a mensagem: “Cadastro do motorista cancelado”.
2. Fluxo de eventos retorna ao passo 1 do Fluxo Básico.

Detalhes de Estrutura de Dados:

a) Dados do motorista = <nome, sobrenome, data de nascimento, sexo, nacionalidade, passaporte, e-mail, telefone, tipo, endereço (logradouro, cidade, estado, país e código postal) e licença (número, categoria, data de validade e país de emissão)>.
b) Detalhes do motorista = <motorista.nome, motorista.sobrenome, motorista.data de nascimento, motorista.sexo, motorista.nacionalidade, motorista.passaporte, motorista.e-mail, motorista.telefone, motorista.tipo, endereço.logradouro, endereço.cidade, endereço.estado, endereço.país, endereço.código postal, licença.número, licença.categoria, licença.data de validade e licença.país de emissão>.

Detalhes de Regras:

a) Regra de motorista = <a data de validade da licença do motorista deve obrigatoriamente ser maior que a data de término da locação; o motorista deve obrigatoriamente possuir idade maior ou igual a 18 anos>.

Figura 53 – Fragmento de Caso de Uso *Cadastrar Motorista*

A.7 Fragmento Identificar Agente

Nome do Fragmento: Identificar Agente
Sub-objetivo: Identificar um dado agente de referência.
Ordem de Utilização: Após Cadastrar Motorista.
Fluxo Básico:
1. <recepcionista> informa parâmetros para busca de agentes.
2. Sistema apresenta uma lista de agentes.
3. <recepcionista> seleciona um agente.
4. Sistema apresenta detalhes do agente selecionado.

Passos opcionais: 4.
Fluxos Alternativos:
a) *nenhum agente satisfaz a busca*
No passo 2 do Fluxo Básico, o Sistema não encontra nenhum agente para seleção.
1. Sistema apresenta a mensagem: “Nenhum agente encontrado”.
2. Fluxo de eventos retorna ao passo 1 do Fluxo Básico.

Detalhes de Estrutura de Dados:
a) Parâmetros para busca de agentes = <nome, sobrenome, telefone ou código postal>.
b) Lista de agentes = <lista de agente.nome e agente.sobrenome, ordenados alfabeticamente. Esta lista contém somente os agentes que possuem algum parâmetro informado para busca>.
c) Detalhes do agente = <agente.nome, agente.sobrenome, agente.data de nascimento, agente.sexo, agente.nacionalidade, agente.passaporte, agente.e-mail, agente.telefone, endereço.logradouro, endereço.cidade, endereço.estado, endereço.país e endereço.código postal>.

Detalhes opcionais: c.
Detalhes de Regras:
Não há.

Figura 54 – Fragmento de Caso de Uso *Identificar Agente*

A.8 Fragmento Depositar em Espécie

Nome do Fragmento: Depositar em Espécie
Sub-objetivo: Garantir a reserva através de depósito em espécie.
Ordem de Utilização: Após Cadastrar Motorista ou Identificar Agente.
Fluxo Básico:
1. Sistema apresenta valor de depósito.
2. <recepcionista> confirma depósito em espécie.
3. Sistema apresenta detalhes do depósito.

Fluxos Alternativos:
a) *depósito em espécie cancelado*
No passo 2 do Fluxo Básico, <recepcionista> decide cancelar o depósito em espécie.
1. Sistema apresenta a mensagem: “Depósito em espécie cancelado”.
2. Fluxo de eventos retorna ao passo 1 do Fluxo Básico.

Detalhes de Estrutura de Dados:
a) Detalhes do depósito = <depósito em espécie.data/hora e depósito em espécie.valor>.

Detalhes de Regras:
Não há.

Figura 55 – Fragmento de Caso de Uso *Depositar em Espécie*

A.9 Fragmento Autorizar Pagamento em Cartão

Nome do Fragmento: Autorizar Pagamento em Cartão
Sub-objetivo: Garantir a reserva através de autorização de pagamento em cartão de crédito.
Ordem de Utilização: Após Cadastrar Motorista ou Identificar Agente.
Fluxo Básico:

1. Sistema apresenta valor de depósito.
2. Sistema apresenta uma lista de bandeiras de cartão de crédito.
3. <recepcionista> seleciona uma bandeira de cartão de crédito.
4. <recepcionista> informa dados do cartão de crédito.
5. Sistema certifica que dados do cartão de crédito são válidos de acordo com a regra de cartão de crédito.
6. Sistema apresenta detalhes do cartão de crédito.
7. <recepcionista> confirma autorização de pagamento em cartão.
8. Sistema apresenta detalhes da autorização.

Passos opcionais: 2, 3, 6.
Fluxos Alternativos:
a) *cartão de crédito inválido*
No passo 5 do Fluxo Básico, o Sistema identifica que o cartão de crédito informado viola a regra de cartão de crédito.

1. Sistema apresenta a mensagem: “A data de validade do cartão de crédito informado deve obrigatoriamente ser maior que a data de término da reserva”.
2. Fluxo de eventos retorna ao passo 4 do Fluxo Básico.

b) *autorização de pagamento em cartão cancelada*
No passo 7 do Fluxo Básico, <recepcionista> decide cancelar a autorização de pagamento em cartão.

1. Sistema apresenta a mensagem: “Autorização de pagamento em cartão de crédito cancelada”.
2. Fluxo de eventos retorna ao passo 1 do Fluxo Básico.

Detalhes de Estrutura de Dados:

a) Lista de bandeiras de cartão de crédito = <lista de cartão.bandeira, ordenadas alfabeticamente>.
b) Dados do cartão de crédito = <número, código, bandeira, nome do titular e data de validade>.
c) Detalhes do cartão de crédito = <cartão.número, cartão.código, cartão.bandeira, cartão.nome do titular e cartão.data de validade>.
d) Detalhes da autorização = <autorização em cartão.data/hora, autorização em cartão.valor, cartão.número, cartão.código, cartão.bandeira, cartão.nome do titular e cartão.data de validade>.

Detalhes opcionais: a, c.
Detalhes de Regras:

a) Regra de cartão de crédito = <a data de validade do cartão de crédito informado deve obrigatoriamente ser maior que a data de término da reserva>.

Figura 56 – Fragmento de Caso de Uso *Autorizar Pagamento em Cartão*

A.10 Fragmento Obter Confirmação da Reserva

Nome do Fragmento: Obter Confirmação da Reserva
Sub-objetivo: Obter confirmação a respeito do registro da reserva.
Ordem de Utilização: Após Depositar em Espécie ou Autorizar Pagamento em Cartão
Fluxo Básico:
1. <recepcionista> solicita o registro da reserva.
2. Sistema apresenta detalhes da reserva.
3. <recepcionista> confirma registro da reserva.
4. Sistema registra a reserva.
5. Sistema apresenta recibo de registro.

Fluxos Alternativos:
a) *registro da reserva cancelado*
No passo 3 do Fluxo Básico, <recepcionista> decide cancelar o registro da reserva.
1. Sistema apresenta a mensagem: “Registro da reserva cancelado”.
2. Fluxo de eventos retorna ao passo 1 do Fluxo Básico.

Detalhes de Estrutura de Dados:
a) Detalhes da reserva = <reserva.data/hora de início, reserva.data/hora de término, reserva.valor, cliente.nome, cliente.sobrenome, local.nome, categoria de carro.nome, modelo de carro.nome, lista de extra.nome e extra.valor, tipo de tarifa.nome, desconto.código, desconto.percentual, motorista.nome, motorista.sobrenome, motorista.licença, agente.nome, agente.sobrenome, garantia.data/hora e garantia.valor>.
b) Detalhes do recibo de registro = <reserva.número e “detalhes da reserva”>.

Detalhes de Regras:
Não há.

Figura 57 – Fragmento de Caso de Uso *Obter Confirmação da Reserva*