Elaboração de Especificações de Casos de Uso para Linhas de Produto de Software Baseada em Fragmentos

Diego O. Araújo[1], Eber A. Schmitz[1], Alexandre L. Correa[1], Antonio J. Alencar[1]

[1]Programa de Pós-Graduação em Informática
Instituto de Matemática e Núcleo de Computação Eletrônica
Universidade Federal do Rio de Janeiro
Caixa Postal 68530 – 21941-590 – Rio de Janeiro – RJ, Brasil

doaraujomsi@gmail.com, eber@nce.ufrj.br, alexcorr@yahoo.com, juarez@dcc.ufrj.br

Resumo. *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 Linhas de Produto de Software é uma tarefa trabalhosa e o modelo resultante pode se tornar sobrecarregado para o uso prático. Para superar esses problemas, é proposto um método que possui duas fases: na primeira fase, um conjunto de fragmentos de casos de uso do domínio é elaborado a partir da análise dos requisitos comuns e variáveis da linha de produto; na segunda fase, descrições textuais de casos de uso para uma dada aplicação podem ser rapidamente produzidas através da composição e personalização desses fragmentos.*

Abstract. *Use cases are widely used for functional requirements elicitation and specification of information systems. However, writing use case textual specifications for Software Product Lines is a laborious task and the resulting model can become overloaded for practical use. To overcome these issues, a two-phase method is proposed: in the first phase, a set of domain use case fragments is elaborated from the analysis of common and variable product line requirements; in the second phase, use case textual descriptions for a given application can be quickly produced by composing and customizing those fragments.*

1. Introdução

O reuso de software é o processo 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). Recentemente, desenvolvedores e gerentes perceberam que o reuso de requisitos e de arquiteturas de software pode trazer benefícios maiores do que os obtidos apenas com o reuso 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 e Northrop 2002). Dentre os benefícios da adoção de uma abordagem de LPS, destacam-se: redução do custo de desenvolvimento, aumento da qualidade e redução do tempo de entrega dos produtos (Pohl et al 2005).

Uma LPS consiste de algumas áreas práticas, como a engenharia de requisitos que 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 os requisitos de uma aplicação específica (Bertolino et al 2006).

A técnica de casos de uso tem ganhado uma ampla aceitação dentre as diversas abordagens disponíveis para a especificação de requisitos de sistemas de informação (Kulak e Guiney 2003). Casos de uso capturam os requisitos funcionais do sistema através de descrições textuais que devem especificar todas as interações entre os usuários e o sistema. O sucesso dessa técnica tem gerado idéias para sua utilização no paradigma de linhas de produto, como os trabalhos propostos em (Bertolino et al 2006) e (Bragança e Machado 2007). Entretanto, a elaboração de especificações de casos de uso para LPS é uma tarefa trabalhosa, uma vez que o comportamento do usuário e do sistema durante a troca de informações entre ambos pode variar entre os membros da linha de produto. 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 e Heymans 2003).

Este trabalho mostra como especificações textuais de casos de uso de uma dada aplicação, que é um membro de uma LPS, podem ser elaboradas através da composição de um conjunto de fragmentos pré-definidos. Cada fragmento representa uma seqüência de interações recorrentes coletadas a partir de sistemas de informação existentes (Dias et al 2008). Estes artefatos fornecem a base para os textos dos casos de uso, auxiliando os engenheiros de requisitos a instanciarem mais facilmente as descrições de casos de uso de acordo com as características da aplicação a ser desenvolvida. O método apresentado é baseado na abordagem para a engenharia de requisitos contida no processo ESPLEP (Gomaa 2004), e também inclui a identificação de requisitos funcionais comuns e variáveis de uma LPS.

Este artigo está organizado da seguinte forma: a seção 2 descreve os trabalhos relacionados que têm o objetivo de adaptar a técnica de casos de uso para linhas de produto, incluindo uma breve discussão sobre o processo ESPLEP, especialmente sua fase de modelagem de requisitos. A seção 3 apresenta o método proposto e um exemplo da sua aplicação. A última seção apresenta as conclusões e os planos para trabalhos futuros.

2. 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 representação da variabilidade nas especificações textuais dos casos de uso. Para solucionar este problema, várias abordagens têm sido propostas, como o método PLUS apresentado por Gomaa (2004). Trata-se de um método de modelagem de software orientado a objetos para LPS que estende os métodos de modelagem baseados na UML a fim de modelar explicitamente as similaridades e variabilidades entre as aplicações de uma linha de produto.

O autor propôs também um processo de desenvolvimento de software para LPS chamado ESPLEP, que incorpora o método PLUS. Seguindo o arcabouço apresentado em (SEI 2009), esse processo baseia-se em dois sub-processos principais: (a) engenharia da linha de produto, também conhecida como engenharia do domínio, que consiste em construir uma arquitetura genérica comum a todas as aplicações da linha de produto e (b) engenharia da aplicação, no qual essa arquitetura é instanciada durante a criação de um determinado produto. Ambos os sub-processos possuem uma fase de modelagem de requisitos, que consiste na elaboração de um modelo de casos de uso e de um modelo de *features*.

Antes de começar a elaboração das especificações de casos de uso, 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 de membros da linha de produto. Uma LPS pode ser desenvolvida quando não existe nenhum sistema para orientar a sua definição ou com base na análise de sistemas existentes. Neste último caso, a identificação dos requisitos comuns e variáveis se inicia a partir da exploração simultânea dos requisitos para todas as aplicações alvo (Pohl et al 2005). Uma maneira simples de realizar esta tarefa é através do uso de matrizes aplicações x requisitos, onde é possível relacionar as aplicações e seus requisitos (Moon et al 2005).

Gomaa (2004) propôs três tipos de casos de uso para LPS: núcleo, os quais devem estar presentes em todos os membros da linha de produto; opcionais, os quais podem estar presentes em alguns produtos; e alternativos, que são usados quando casos de uso diferentes são requisitados por diferentes aplicações. Algumas abordagens têm sido propostas a fim de apoiar a elaboração de especificações de casos de uso baseando-se nos requisitos do domínio identificados. Em (Moon et al 2005), diagramas de casos de uso foram extraídos a partir das matrizes aplicações x requisitos usadas na identificação de requisitos. Kim et al (2006) propôs algumas regras para transformar os requisitos do domínio descritos por metas e cenários em especificações de casos de uso.

O método PLUS definiu algumas diretrizes para a elaboração de especificações textuais de casos de uso. Além disso, a variabilidade é expressa usando relacionamentos do tipo *include* e *extend* da UML nos diagramas de casos de uso e pontos de variação nos textos de casos de uso. Outros autores adaptaram o *template* de casos de uso sugerido por Cockburn (2001), acrescentando variabilidade a este formalismo: Bertolino et al (2006) introduziram *tags* (opcionais, alternativas ou paramétricas) nos cenários, para indicar partes do texto que podem ser instanciadas para uma aplicação específica. John e Muthig (2002) propuseram textos genéricos de casos de uso, onde fragmentos de texto variáveis nos cenários são descritos usando *tags* baseadas em XML.

Durante a fase de modelagem de requisitos, um modelo de *features* também deve ser desenvolvido (Gomaa 2004). Em LPS, uma *feature* é definida 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), Gomaa propôs três tipos de *features*: comuns, opcionais e alternativas. Dado duas *features* A e B, uma relação de dependência entre essas *features* (A $\rightarrow$ B) pode ser de três tipos: *requires* (a seleção de A implica na seleção de B), *mutually includes* (B só pode ser selecionada em conjunto com A) e *mutually exclusive* (a seleção de A implica na não seleção de B). Assim como na abordagem original proposta por Griss et al (1998), *features* representando requisitos

funcionais podem ser determinadas a partir do modelo de casos de uso do domínio. Cada *feature* pode ser modelada como um grupo de casos de uso ou os casos de uso podem ser mapeados para as *features*. Neste contexto, algumas outras abordagens surgiram recentemente. No método PLUSS (Eriksson et al 2005), cada *feature* pode estar associada a um caso de uso, um cenário alternativo ou um passo específico. Bragança et al (2007) apresentaram um método para automatizar a transformação de casos de uso da UML em modelo de *features*.

É importante notar que a maioria dessas abordagens foca principalmente em expressar a variabilidade nos casos de uso. Entretanto, nas descrições textuais de casos de uso, é difícil distinguir cenários alternativos clássicos, ou seja, provenientes das funcionalidades do domínio, de cenários alternativos originados da variabilidade (Trigaux e Heymans 2003). Além disso, a análise de similaridade e variabilidade voltada para casos de uso é uma tarefa não trivial, para a qual pouca atenção tem sido dada. O método proposto neste trabalho tem por objetivo auxiliar a elaboração de especificações de casos de uso a partir dos requisitos comuns e variáveis identificados no domínio e, com isto, agilizar a instanciação desses artefatos durante a especificação de requisitos de novas aplicações da linha de produto. Esta abordagem estende o método PLUS e inclui a identificação de requisitos através da modelagem de objetivos, a análise de similaridade e variabilidade através de matrizes aplicações x requisitos e a escrita de descrições textuais de casos de uso através da composição e personalização de fragmentos.

3. Desenvolvendo Especificações de Casos de Uso para LPS

O método para a especificação de casos de uso para LPS apresentado a seguir é baseado nos conceitos introduzidos na seção anterior e se insere nas duas atividades essenciais de uma LPS: engenharia do domínio e engenharia da aplicação. Para ilustrar este método, é apresentado um exemplo 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.

3.1. Engenharia do Domínio

No contexto da engenharia do domínio, o método é composto por cinco atividades: (a) elaborar um modelo conceitual do domínio a partir da análise das aplicações alvo no domínio; (b) identificar os requisitos do domínio através da análise dos objetivos e sub-objetivos comuns e variáveis das aplicações alvo; (c) elaborar descrições de fragmentos de casos de uso para cada sub-objetivo identificado; (d) construir um diagrama de *features* do domínio a partir dos objetivos e sub-objetivos identificados; (e) construir um diagrama de casos de uso do domínio a partir do diagrama de *features* do domínio.

3.1.1. Elaborar 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 que são utilizados pelas partes interessadas, bem como no processo de desenvolvimento, podem ser descritos no modelo conceitual do domínio. A Figura 1

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.

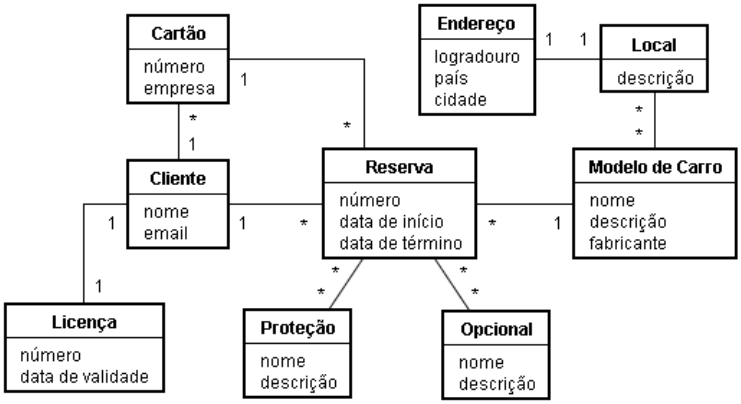

Figura 1. Trecho do Modelo Conceitual do Domínio de Reserva de Aluguel de Carros

Neste exemplo, a reserva de um carro envolve a seleção de um local, a definição de um período de locação e a escolha de um modelo de carro. Este modelo deve estar disponível no local selecionado e no período informado. Além disso, opcionais podem ser adicionados e proteções podem ser contratadas. Para garantir a reserva, um cartão de crédito do cliente pode ser exigido também. As informações pessoais também devem ser fornecidas antes da confirmação da reserva.

3.1.2. Identificar Requisitos do Domínio

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 e Sawyer 1997). No contexto de linhas de produto, os objetivos de negócio direcionam planos de marketing e de produtos, e forçam produtos em uma família de produtos a possuírem requisitos comuns e variáveis. Com base nos objetivos de negócio, as características dos produtos são determinadas e, portanto, os objetivos fornecem a fundamentação para os requisitos do domínio (Kim et al 2006).

Para obter-se um conjunto de requisitos do domínio dentre todos os membros da linha de produto, deve-se identificar os requisitos comuns a todas as aplicações, e os requisitos que diferem entre as aplicações. De acordo com Pohl et al (2005), as matrizes aplicações x requisitos fornecem uma síntese dos requisitos de mais alto nível das várias aplicações e, portanto, podem ser usadas na análise de similaridade e variabilidade. No método aqui proposto, os requisitos de cada produto são identificados através da modelagem de objetivos do ponto de vista do usuário. Segundo Cockburn (2001), o objetivo do caso de uso pode ser recursivamente decomposto em sub-objetivos, os quais correspondem a sub-funções que estão sob o principal nível de interesse dos usuários.

A Tabela 1 apresenta a matriz elaborada para o domínio de reserva de aluguel de carros do exemplo, contendo um conjunto de objetivos e seus respectivos sub-objetivos que foram extraídos a partir da análise de três aplicações reais, as quais foram aqui denominadas A1, A2 e A3. Seguindo a abordagem apresentada em (Moon et al 2005),

na análise de similaridade e variabilidade (S/V), para cada requisito identificado pode ser calculada uma taxa de similaridade (Taxa) indicando quanto o requisito é utilizado no domínio. Ela é definida como a razão entre a quantidade de sistemas que possuem o requisito e o número total de sistemas. Quando a taxa é considerada alta, o objetivo ou sub-objetivo é definido como comum (C), como o sub-objetivo *Escolher um Local*; caso contrário, opcional (O), como o sub-objetivo *Escolher Opcionais/Proteções*. Assim como em (Moon et al 2005), adotou-se o valor de 0,5 para separação de objetivos comuns dos opcionais.

Tabela 1. Matriz Aplicações x Requisitos

Objetivos/Sub-objetivos		S/V	Taxa	A1	A2	A3
1	**Reservar um Carro**	**C**	**1,0**	**1**	**1**	**1**
1.1	Escolher um Local	C	1,0	1	1	1
1.2	Selecionar um Modelo de Carro	C	1,0	1	1	1
1.3	Escolher Opcionais/Proteções	O	0,33	1	0	0
1.4	Informar Dados Pessoais	C	1,0	1	1	1
1.5	Informar Cartão de Crédito	O	0,33	0	0	1
1.6	Obter Confirmação da Reserva	C	1,0	1	1	1
2	**Alterar uma Reserva**	**C**	**1,0**	**1**	**1**	**1**
2.1	Obter Detalhes da Reserva	C	1,0	1	1	1
2.2	Informar Dados Pessoais	C	1,0	1	1	1
2.3	Obter Confirmação da Alteração	C	1,0	1	1	1

3.1.3. Elaborar Descrições de Fragmentos de Casos de Uso

Dias et al (2008) apresentou uma abordagem visando reduzir o esforço necessário para elaborar especificações de casos de uso de alta qualidade. 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 necessários para atingir sub-objetivos genéricos. Cada fragmento pode, então, ser personalizado para satisfazer a um objetivo de caso de uso. Tal abordagem pode ser aplicada em qualquer especificação de requisitos de software que tenha sub-objetivos associáveis aos fragmentos.

A Tabela 2 apresenta um trecho de um fragmento de caso de uso que pode ser utilizado para refinar o sub-objetivo genérico *Obter Detalhes de um Objeto*. Esse fragmento pode ser usado na maioria das situações onde se deseja obter detalhes de um objeto. Sua estrutura inclui não só as interações presentes no fluxo básico de uma descrição de caso de uso, como também os fluxos alternativos usuais, detalhes de entrada e saída de dados e de regras de negócio. Dentro da descrição, os pontos de personalização são marcados por '<' e '>'. Esses pontos devem ser instanciados com a correta informação relacionada ao contexto do caso de uso.

Os fragmentos de casos de uso apresentados em (Dias et al 2008) não pertencem a nenhum domínio específico, por estarem 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. No método aqui proposto, acredita-se que fragmentos de casos de

uso para um domínio específico podem ser encontrados e utilizados para auxiliar a especificação textual de casos de uso de novas aplicações num contexto de LPS.

Tabela 2. Uma Descrição de Fragmento de Caso de Uso

Nome do Fragmento de Caso de Uso: Obter Detalhes de um Objeto. **Sub-Objetivo:** Obter os detalhes de um objeto fornecendo o identificador desse objeto. **Propósito:** Este fragmento deve ser usado quando o ator deseja obter os detalhes de um objeto de negócio existente. **Fluxo Básico:** 1. <ator> informa <identificador> de <objeto>. 2. Sistema apresenta detalhes de <objeto>. **Fluxos Alternativos:** a) <objeto> não encontrado No passo 2 do Fluxo Básico, o Sistema não encontra nenhum <objeto> relacionado ao <identificador> fornecido. 1. Sistema apresenta a mensagem: "Nenhum <objeto> encontrado". 2. Fluxo de eventos retorna ao passo 1 do Fluxo Básico. **Detalhes de Entrada e Saída:** a) Detalhes de <objeto> = lista de <propriedades>.

Desta forma, para cada sub-objetivo identificado do domínio, uma descrição detalhada na forma de especificação de fragmento de caso de uso é então elaborada. O modelo conceitual do domínio construído na atividade 1 pode ser personalizado de acordo com a aplicação em desenvolvimento e utilizado para identificar os detalhes de entrada e saída de dados. Além disso, detalhes de regras de negócio também podem ser adicionados. A Tabela 3 apresenta um trecho de um fragmento de caso de uso específico que pode ser usado para refinar o sub-objetivo *Obter Confirmação da Reserva*. É importante esclarecer que este fragmento não foi produzido a partir do fragmento apresentado na Tabela 2 e está sendo mostrado para ilustrar a diferença entre um fragmento genérico e um fragmento voltado para um domínio específico.

Tabela 3. Fragmento de Caso de Uso Obter Confirmação da Reserva

Nome do Fragmento de Caso de Uso: Obter Confirmação da Reserva **Sub-objetivo:** Obter confirmação a respeito do registro da reserva. **Propósito:** Este fragmento deve ser usado quando o ator deseja obter a confirmação de uma reserva de carro corrente. **Fluxo Básico:** 1. <cliente> solicita o registro da reserva. 2. Sistema registra dados da reserva. 3. Sistema apresenta detalhes da reserva. 4. <cliente> confirma registro da reserva. 5. Sistema apresenta recibo de registro. **Fluxos Alternativos:** a) *registro cancelado* No passo 4 do Fluxo Básico, <cliente> decide cancelar o registro. 1. Sistema apresenta a mensagem: "Registro cancelado".	2. Fluxo de eventos retorna ao passo 1 do Fluxo Básico. **Detalhes de Entrada e Saída:** a) Dados da reserva = <local, data de início, data de término, modelo de carro, lista de opcionais, lista de proteções, cliente, cartão>. b) Detalhes da reserva = <local.nome, data de início, data de término, modelo de carro.nome, lista de opcional.nome, lista de proteção.nome, cliente.nome, cartão.número>. c) Detalhes do recibo de registro = <reserva.número, local.nome, reserva.data de início, reserva.data de término, modelo de carro.nome, lista de opcional.nome, lista de proteção.nome, cliente.nome, cartão.número>. **Detalhes de Regras:** *Não há*.

3.1.4. Construir Diagrama de *Features* do Domínio

Os objetivos descrevem as intenções das partes interessadas com relação ao sistema em consideração, enquanto que as *features* descrevem as características que um sistema oferece aos seus clientes (Pohl et al 2005). Existe uma redundância entre as definições de objetivos e *features*. Na maioria dos casos, modelos de *features* e objetivos definem informações semelhantes. Para expressar as intenções de um sistema, tanto modelos de objetivos como modelos de *features* podem ser utilizados. No trabalho apresentado por Yu et al (2008), objetivos foram associados a features para garantir a rastreabilidade dos requisitos do domínio.

No método apresentado neste trabalho, o diagrama de *features* do domínio é construído para representar o maior nível de abstração para o desenvolvimento do domínio. O diagrama de *features* do domínio é obtido a partir da análise dos objetivos e sub-objetivos identificados. Basicamente, os objetivos comuns tornam-se *features* comuns e objetivos opcionais tornam-se *features* opcionais. A mesma estratégia é aplicada para os sub-objetivos. Seguindo o método PLUS, os estereótipos <<common feature>> e <<optional feature>> foram adotados para distinguir os dois tipos de *features*. O relacionamento entre *features* segue o relacionamento entre objetivos e seus sub-objetivos descritos na matriz construída na atividade 2. A Figura 2 apresenta o diagrama de *features* construído para o domínio de reserva de aluguel de carros.

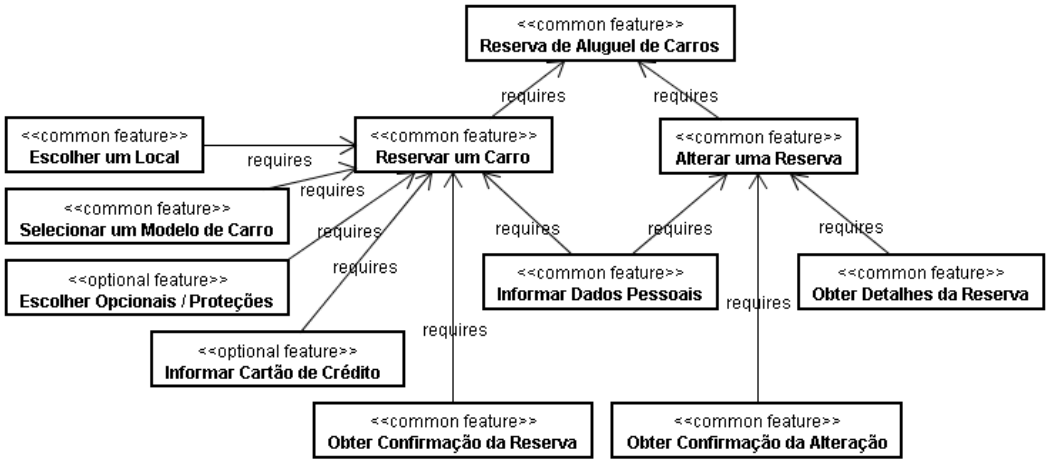

Figura 2. Diagrama de *Features* da Reserva de Aluguel de Carros

Neste diagrama, o domínio de reserva de aluguel de carros possui um núcleo que é composto por duas *features* principais: *Reservar um Carro* e *Alterar uma Reserva*. A *feature Reservar um Carro* é composta por algumas *features* comuns, como a *feature Selecionar um Modelo de Carro*, e por algumas opcionais, como a *feature Escolher Opcionais/Proteções*. A *feature Alterar uma Reserva* é composta por três *features* comuns: *Obter Detalhes da Reserva*, *Obter Confirmação da Alteração* e *Informar Dados Pessoais*, que é compartilhada com a *feature Reservar um Carro*.

3.1.5. Construir Diagrama de Casos de Uso do Domínio

No processo ESPLEP, a modelagem de casos de uso antecede a modelagem de *features*. Entretanto, no método aqui proposto, as *features* representam objetivos ou sub-objetivos

do ponto de vista do usuário e devem ser analisadas para extrair-se o diagrama de casos de uso do domínio. Portanto, uma vez que as *features* do domínio auxiliam na elaboração de especificações de casos de uso da LPS, a modelagem de *features* deve ser realizada antes da modelagem de casos de uso.

A construção do diagrama de casos de uso do domínio começa com a elaboração de um diagrama inicial, onde se adiciona um caso de uso para cada *feature* que represente um objetivo de primeiro nível do diagrama de *features*. As *features* que representem sub-objetivos (níveis mais abaixo no diagrama de *features*) originam fragmentos de caso de uso. A Figura 3 apresenta o diagrama de casos de uso construído para o domínio de reserva de aluguel de carros do exemplo. As *features Reservar um Carro* e *Alterar uma Reserva* deram origem aos casos de uso presentes no diagrama e oito fragmentos de casos de uso foram mapeados, como *Selecionar um Modelo de Carro* e *Obter Detalhes da Reserva.*

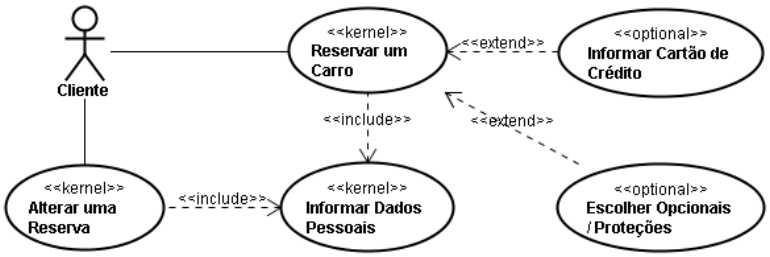

Figura 3. Diagrama de Casos de Uso da Reserva de Aluguel de Carros

O diagrama de casos de uso inicialmente obtido pode ser refinado adicionando-se relacionamentos entre casos de uso do tipo *include* e *extend* da UML. Caso um sub-objetivo apareça em diferentes casos de uso, ele é considerado comum a esses casos de uso, dando origem, nesse caso, a um caso de uso relacionado com o caso de uso original através de um relacionamento do tipo *include*. No exemplo, a *feature Informar Dados Pessoais* tornou-se um caso de uso do tipo *include* relacionado com os casos de uso *Reservar um Carro* e *Alterar uma Reserva*. Caso um sub-objetivo seja opcional no domínio, ele pode ser separado como um novo caso relacionado com o caso de uso original através de um relacionamento do tipo *extend* ou *include*. No exemplo, as *feature Escolher Opcionais/Proteções* e *Informar Cartão de Crédito* tornaram-se dois casos de uso do tipo *extend* relacionados com o caso de uso *Reservar um Carro*.

Por último, os tipos dos casos de uso dependem dos tipos identificados para as *features*. Um caso de uso originado de uma *feature* comum é chamado de caso de uso núcleo; caso contrário, opcional. Seguindo o método PLUS, os estereótipos <<kernel>> e <<optional>> foram adotados para distinguir estes dois tipos de casos de uso.

3.2. Engenharia da Aplicação

No contexto da engenharia da aplicação, o método é composto por três atividades: (a) selecionar as *features* da aplicação a partir do diagrama de *features* do domínio; (b) com base nas *features* selecionadas, instanciar o diagrama de casos de uso da aplicação a partir do diagrama de casos de uso do domínio; (c) elaborar as descrições textuais dos casos de uso através da composição e personalização dos fragmentos de casos de uso associados aos sub-objetivos selecionados.

3.2.1. Selecionar *Features* da Aplicação

A abordagem aplicada na engenharia da aplicação começa com a seleção das características da aplicação a partir do diagrama de *features* do domínio. Uma aplicação típica possui todas as *features* comuns e uma seleção das opcionais. Considere, por exemplo, uma aplicação de reserva de aluguel de carros que não ofereça aos clientes nem opcionais, nem proteções e tampouco exija qualquer cartão de crédito durante a reserva do carro. A Figura 4 apresenta o diagrama de *features* da aplicação desse exemplo, instanciado a partir do diagrama de *features* do domínio apresentado na Figura 2.

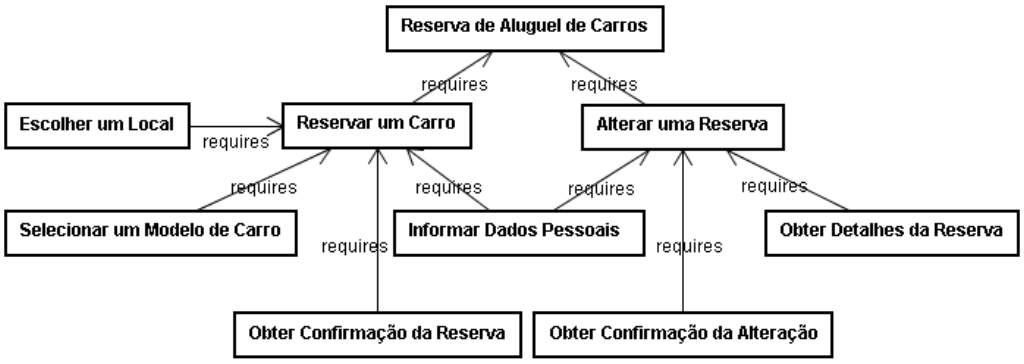

Figura 4. Diagrama de *Features* para uma Aplicação Específica

De acordo com esse diagrama, a aplicação de reserva de aluguel de carros em questão é composta por duas *features* principais: *Reservar um Carro* e *Alterar uma Reserva*. A *feature Reservar um Carro* é composta por quatro *features*: *Escolher um Local*, *Selecionar um Modelo de Carro, Informar Dados Pessoais* e *Obter Confirmação da Reserva*. A *feature Alterar uma Reserva* é composta pelas seguintes *features*: *Obter Detalhes da Reserva*, *Informar Dados Pessoais* e *Obter Confirmação da Alteração*.

3.2.2. Instanciar Diagrama de Casos de Uso da Aplicação

Com base no diagrama de *features* da aplicação, o diagrama de casos de uso do domínio é analisado para determinar os casos de uso que farão parte da aplicação específica. O diagrama de casos de uso da aplicação será constituído por todos os casos de uso núcleo (identificados pelo estereótipo <<kernel>> no diagrama de casos de uso do domínio) e uma seleção dos opcionais (identificados pelo estereótipo <<optional>>). A Figura 5 apresenta o diagrama de casos de uso da aplicação do exemplo, instanciado a partir do diagrama de casos de uso do domínio apresentado na Figura 3.

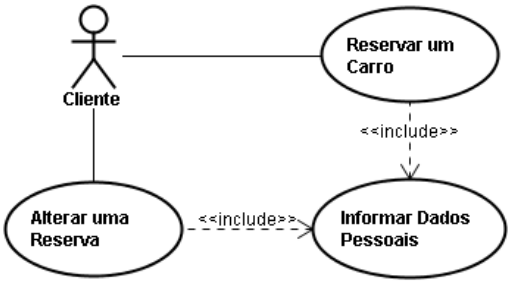

Figura 5. Diagrama de Caso de Uso para uma Aplicação Específica

Neste diagrama, o ator *Cliente* pode reservar um carro ou alterar uma reserva de carro. Como a aplicação não oferece aos clientes nem opcionais, nem proteções e tampouco exige qualquer cartão de crédito para garantir a reserva do carro, os casos de uso *Escolher Opcionais/Proteções* e *Informar Cartão de Crédito* não aparecem no diagrama.

3.2.3. Elaborar Especificação Textual dos Casos de Uso

A elaboração da especificação textual de cada caso de uso é realizada através da composição e personalização de um conjunto de fragmentos de casos de uso. Para cada caso de uso que se deseja especificar, deve-se identificar quais são as *features* associadas aos seus sub-objetivos com base no diagrama de *features* da aplicação. Depois, deve-se incorporar na descrição textual do caso de uso, se houver, o fragmento de caso de uso correspondente a cada *feature* identificada. Caso seja necessário, o fragmento pode ser personalizado de acordo com as necessidades específicas da aplicação. A personalização 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.

O apêndice apresenta a descrição do caso de uso *Reservar um Carro* elaborado através da composição dos seguintes fragmentos: (1) *Escolher um Local*, (2) *Selecionar um Modelo de Carro* e (3) *Obter Confirmação da Reserva*. O primeiro fragmento deu origem aos passos 2-5 do fluxo básico; ao fluxo alternativo *nenhum local satisfaz a seleção*; e às saídas *lista de locais* e *detalhes do local*. O segundo fragmento deu origem aos passos 6-10 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 modelos de carros* e *detalhes do modelo de carro*; e à *regra de período de locação*. O terceiro fragmento deu origem aos passos 12-16 do fluxo básico; ao fluxo alternativo *registro cancelado*; à entrada *dados da reserva*; e às saídas *detalhes da reserva* e *detalhes do recibo de registro*. Neste último caso, nos dados de entrada e de saída foram retirados os atributos de opcionais, proteções e cartão de crédito, tendo em vista que a aplicação não oferece aos clientes nem opcionais, nem proteções e tampouco exige qualquer cartão de crédito para garantir a reserva do carro.

4. Conclusões e Trabalhos Futuros

Este trabalho apresentou um método para a elaboração de especificações de casos de uso para LPS baseado em fragmentos. Esta abordagem foi inserida nas duas atividades essenciais de uma LPS: 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 requisitos comuns e variáveis da linha de produto, onde cada fragmento representa um conjunto de interações recorrentes necessárias para atender um sub-objetivo. O conjunto de artefatos elaborado é composto ainda por um diagrama de *features* e um diagrama de casos de uso do domínio. Na segunda atividade, descrições textuais de casos de uso para uma dada aplicação podem ser produzidas através da composição e personalização desses fragmentos.

A identificação de requisitos através da modelagem de objetivos e a análise de similaridades e variabilidades através do uso de matrizes fornecem uma maneira natural

de identificar a variabilidade num contexto de LPS, capturando os diversos caminhos pelos quais os usuários do sistema podem seguir para atingir seus objetivos. Os fragmentos de casos de uso aparecem neste contexto 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, este artefato pode ser mais facilmente personalizado do que um *template* de caso de uso. A combinação destas características auxilia os engenheiros de requisitos a instanciarem descrições de casos de uso mais rapidamente e alinhadas às características da aplicação em desenvolvimento. No momento, estamos trabalhando no desenvolvimento de uma ferramenta para apoiar a busca e personalização dos fragmentos.

Referências

Bragança A., Machado R.J. (2007) "Automating Mappings between Use Case Diagrams and Feature Models for Software Product Lines". Em: *International Conference on Software Product Lines* (SPLC), Kyoto, Japan.

Bertolino, A., Fantechi, A., Gnesi, S., Lami, G. (2006) "Product Line Use Cases: Scenario-Based Specification and Testing of Requirements". Em: *Software Product Lines: Research Issues in Engineering and Management*. LNCS, p. 425-445. Springer, Heidelberg.

Clements P., Northrop L. (2002) "Software Product Lines: Practices and Patterns". Boston, MA, USA, Addison-Wesley.

Cockburn A. (2001) "Writing Effective Use Cases". Addison-Wesley.

Dias, F., Schmitz, E., Campos, M. L., Correa, A., Alencar, A. (2008) "Elaboration of Use Case Specifications: an Approach Based on Use Case Fragments". Em: *23rd Annual ACM Symposium on Applied Computing* (SAC'08), Fortaleza, Ceará, Brazil, p. 614-618.

Eriksson, M., Börstler, J., Borg, K. (2005) "The PLUSS Approach - Domain Modeling with Features, Use Cases and Use Case Realizations". Em: *International Conference on Software Product Lines* (SPLC), LNCS, vol. 3714, Springer-Verlag, p. 33-44.

Gomaa, H. (2004) "Designing Software Product Lines with UML: from Use Cases to Pattern-Based Software Architectures". Addison-Wesley.

Griss, M., Favaro, J., D'Alessandro, M. (1998) "Integrating Feature Modeling with the RSEB". Em: *5th International Conference on Software Reuse*, Vancouver, Canada.

John, I., Muthig, D. (2002) "Product Line Modeling with Generic Use Cases". Em: *SPLC2 Workshop on Techniques for Exploiting Commonality Through Variability Management*, San Diego, California, USA.

Kang, K., Cohen, S., Hess, J., Novak, W., Peterson, A. A. (1990) "Feature-Oriented Domain Analysis (FODA) Feasibility Study" (CMU/SEI-90-TR-021, ADA235785). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

Kim, J., Kim, M., Park, S. (2006) "Goal and Scenario based Domain Requirements Analysis Environment". Elsevier Science Inc., New York, NY, USA, p. 926-938.

Krueger C. W. (1992) “Software Reuse”. ACM Computing Surveys, v. 24 (2), p. 131-183.

Kulak, D., Guiney, E. (2003) “Use Cases: Requirements in Context”. 2nd ed., Addison-Wesley.

Moon, M., Yeom, K., Chae, H. (2005) “An Approach to Developing Domain Requirements as a Core Asset-Based on Commonality and Variability Analysis in a Product Line”. IEEE Transactions on Software Engineering 31, 7, p. 551-569.

Pohl. K., Bockle. G., Van der Linden, F. (2005) “Software Product Line Engineering: Foundations, Principles, and Techniques”. Springer.

SEI - Software Engineering Institute. (2009) “A Framework for Software Product Line Practice”. Pittsburgh, http://www.sei.cmu.edu/plp/framework.html, Junho.

Sommerville, I., Sawyer, P. (1997) “Requirements Engineering: A Good Practice Guide”. John Wiley & Sons.

Trigaux, J. C., Heymans, P. (2003) “Modelling variability requirements in Software Product Lines: a Comparative Survey”. FUNDP - Equipe LIEL Institut d’Informatique.

Yu, Y., Leite, J. C. P., Lapouchnian, A., Mylopoulos, J. (2008) “Configuring Features with Stakeholder Goals”. Em: *23rd Annual ACM Symposium on Applied Computing* (SAC’08), Fortaleza, Ceará, Brazil, p. 645-649.

Apêndice. Uma Descrição de Casos de Uso Composta por Fragmentos

Caso de Uso: Reservar um Carro
Ator: Cliente
Fluxo Básico:
1. O caso de uso começa quando o Cliente solicita o registro de uma reserva.
2. Cliente informa país e cidade.
3. Sistema apresenta uma lista de locais.
4. Cliente seleciona um local.
5. Sistema apresenta detalhes do local selecionado.
6. Cliente informa período de locação.
7. Sistema certifica que período de locação é válido de acordo com a regra de período de locação.
8. Sistema apresenta uma lista de modelos de carro.
9. Cliente seleciona um modelo de carro.
10. Sistema apresenta detalhes do modelo de carro selecionado.
11. Executar o caso de uso "Informar Dados Pessoais".
12. Cliente solicita o registro da reserva.
13. Sistema registra dados da reserva.
14. Sistema apresenta detalhes da reserva.
15. Cliente confirma registro da reserva.
16. Sistema apresenta recibo de registro.
17. O caso de uso termina.

Fluxos Alternativos:

a) *nenhum local satisfaz a seleção*
No passo 3 do Fluxo Básico, o Sistema não encontra nenhum local para seleção.
1. Sistema apresenta a mensagem: "Nenhum local encontrado".
2. Fluxo de eventos retorna ao passo 2 do Fluxo Básico.

b) *período de locação inválido*
No passo 7 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 6 do Fluxo Básico.

c) *nenhum modelo de carro satisfaz a seleção*
No passo 8 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 6 do Fluxo Básico.

d) *registro cancelado*
No passo 15 do Fluxo Básico, Cliente decide cancelar o registro.
1. Sistema apresenta a mensagem: "Registro cancelado".
2. Fluxo de eventos retorna ao passo 12 do Fluxo Básico.

Detalhes de Entrada e Saída:
a) Lista de locais = lista de logradouros. Esta lista contém somente os locais do país e cidade informados.
b) Detalhes do local = descrição, endereço.logradouro, endereço.cidade, endereço.país.
c) Período de locação = data de início e data de término da locação no formato dia/mês/ano.
d) Lista de modelos de carros = lista de nomes, ordenados alfabeticamente. Esta lista contém somente os modelos de carros que podem ser alugados no local indicado, a partir da data de início até a data de término informada.
e) Detalhes do modelo de carro = nome, descrição, fabricante.
f) Dados da reserva = local, data de início, data de término, modelo de carro, cliente.
g) Detalhes da reserva = local.nome, data de início, data de término, modelo de carro.nome, cliente.nome.
h) Detalhes do recibo de registro = reserva.número, local.nome, reserva.data de início, reserva.data de término, modelo de carro.nome, cliente.nome.

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.