UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE CIÊNCIAS
DEPARTAMENTO DE COMPUTAÇÃO
MESTRADO E DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

ISMAYLE DE SOUSA SANTOS

UM AMBIENTE PARA GERAÇÃO DE CENÁRIOS DE TESTES PARA LINHAS DE PRODUTO DE SOFTWARE SENSÍVEIS AO CONTEXTO

FORTALEZA, CEARÁ

2013

ISMAYLE DE SOUSA SANTOS

UM AMBIENTE PARA GERAÇÃO DE CENÁRIOS DE TESTES PARA LINHAS DE PRODUTO DE SOFTWARE SENSÍVEIS AO CONTEXTO

Dissertação submetida à Coordenação do Programa de Pós-Graduação em Ciência da Computação da Universidade Federal do Ceará, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientadora: Profa. Dra. Rossana Maria de Castro Andrade

Co-Orientador: Prof. Dr. Pedro de Alcântara dos Santos Neto

FORTALEZA, CEARÁ

2013

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca de Ciências e Tecnologia

S235a Santos, Ismayle de Sousa.
Um Ambiente para geração de cenários de testes para linhas de produto de software sensíveis ao contexto. / Ismayle de Sousa Santos. – 2013.
135 f. : il. , enc. ; 30 cm.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Departamento de Computação, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2013.
Área de Concentração: Ciência da Computação.
Orientação: Profa. Dra. Rossana Maria de Castro Andrade.
Coorientação: Prof. Dr. Pedro de Alcântara dos Santos Neto.

1. Engenharia de software. 2. Software - testes. I. Título.

CDD 005

ISMAYLE DE SOUSA SANTOS

UM AMBIENTE PARA GERAÇÃO DE CENÁRIOS DE TESTES PARA LINHAS DE PRODUTO DE SOFTWARE SENSÍVEIS AO CONTEXTO

Dissertação submetida à Coordenação do Programa de Pós-Graduação em Ciência da Computação da Universidade Federal do Ceará, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação. Área de concentração: Engenharia de Software

Aprovada em: __/__/____

BANCA EXAMINADORA

Profa. Dra. Rossana Maria de Castro Andrade
Universidade Federal do Ceará - UFC
Orientadora

Prof. Dr. Pedro de Alcântara dos Santos Neto
Universidade Federal do Piauí - UFPI
Co-orientador

Prof. Dr. Sérgio Castelo Branco Soares
Universidade Federal de Pernambuco - UFPE

Prof. Dr. Fernando Antônio Mota Trinta
Universidade Federal do Ceará - UFC

A minha eterna avó Elvina, aos meus maravilhosos pais, ao meu grande irmão e a minha futura esposa Paulinha

AGRADECIMENTOS

Na verdade não existem palavras no mundo para expressar tanta gratidão e mesmo que existissem eu precisaria de centenas de páginas. Dessa forma, tentarei fazer meu máximo com as palavras que conheço e com o espaço que tenho:

Agradeço primeiramente a DEUS, pois mesmo nas pequenas decisões ele sempre me guiou a tomar o melhor caminho. Sabe aquele ditado: "DEUS escreve certo por linhas tortas"? Não é verdade. Ele nunca usou linhas tortas. Nós em nossa plena imperfeição é que nunca o entendemos.

Agradeço aos meus pais por toda a atenção, amor e carinho. Eles são exemplos únicos e eternos para mim. Eles não são médicos, mas cuidam continuamente de mim. Eles não são professores, mas sabem me ensinar. Eles não são advogados, mas sempre me defendem. Não são engenheiros, mas construíram tudo o que há de bom em mim.

Também agradeço meu irmão Kádson que tanto me apoiou e me incentivou. Ele é a segunda pessoa mais determinada e corajosa que eu já vi. As vezes ele me diz que quer ser como eu, mas na verdade sou eu quem sempre quis ser como ele, pois na prova da vida ele sempre tira a nota máxima.

Agradeço também com muito carinho a minha futura esposa Paula por todo o amor concedido a mim. Além de estar sempre ao meu lado e sempre me incentivar, ela permanentemente exibe as melhores características que um ser humano pode ter. Só espero poder ser tão forte, tão amável e humano quanto ela é. Ela também sem sombras de dúvida é a pessoa mais determinada que eu conheço. É por isso que quem quiser saber a minha maior conquista, não precisa olhar no meu lattes, basta procurar pela minha doce Paula.

Agradeço aos meus grandes orientadores, a Professora Rossana e o Professor Pedro, por todo incentivo, apoio e pelas oportunidades. Foi com eles que percebi o real significado da frase "Se vi ao longe é porque estava nos ombros dos gigantes" (Aristóteles). O que sou profissionalmente (e até pessoalmente) hoje é resultado do trabalho dessas duas pessoas maravilhosas. Eu só espero algum dia mostrar ao menos um pouco do talento deles.

Agradeço aos meus amigos que compartilham comigo vários momentos da vida. Em especial agradeço aos meus meus amigos desde graduação David, Eugênio, Edmilson, Simone, Laiara, Júlio Cesar e Igor. Eles definitivamente fizeram a graduação se tornar bem mais divertida.

Agradeço a todos meus amigos do GREat, principalmente o pessoal da CTQs, professores, alunos e servidores. Vocês também são da minha família agora.

Agradeço a CAPES ao CNPq pelo suporte financeiro durante meus anos de pesquisa. Afinal, o que seria de mim sem as bolsas.

Por fim, agradeço a você que dedicou um tempo a ler esta dissertação :)

“Em Deus nós confiamos, todo resto nós testamos”

(Desconhecido)

RESUMO

Uma Linha de Produto de Software Sensível ao Contexto (LPSSC) é uma linha de produto para o desenvolvimento de aplicações sensíveis ao contexto, que alteram dinamicamente o comportamento ou que proveem serviços com base em informações de contexto. Nesse cenário, a atividade de testes precisa lidar ao mesmo tempo com as peculiaridades das aplicações finais, que são sensíveis ao contexto, e com o paradigma de desenvolvimento em linha de produto. Mediante a complexidade envolvida nos testes de uma LPSSC, é fundamental a existência de métodos ou ferramentas de suporte a essa atividade, especialmente com a intenção de criar testes a partir dos requisitos da linha. O objetivo dessa geração precoce dos testes é permitir a identificação e correção dos defeitos nos estágios iniciais de desenvolvimento. Dessa forma, esta dissertação tem por objetivo propor um ambiente de geração de cenários de testes para uma LPSSC que utiliza especificações textuais de casos de uso com informações de contexto e que possibilita a reutilização dos testes. Este ambiente é constituído pela proposta de um método de geração de cenários de testes, de um template para especificação textual de casos de uso de uma LPSSC e de uma ferramenta de apoio. O método utiliza como base especificações textuais de casos de uso com informações sobre: funcionalidade, variabilidade da linha, e como o contexto afeta os produtos finais. O ambiente também contém o template para caso de uso que fornece suporte ao uso do método e a ferramenta de apoio, que permite a modelagem de casos de uso segundo esse template e implementa o método proposto. Com a intenção de verificar os benefícios do ambiente, proposto nesta dissertação, quanto a geração de cenários de testes para uma LPSSC, conduziu-se uma avaliação preliminar na forma de experimento controlado. Baseado nos resultados coletados nessa avaliação percebeu-se que: o template favorece o entendimento de um caso de uso de uma LPSSC; o método favorece a criação de um conjunto de testes com uma maior cobertura para uma LPSSC, se comparado a criação de testes de forma não sistemática; e a ferramenta representa um suporte necessário.

Palavras-chave: Teste de Software. Caso de Uso. Linha de Produto de Software. Linha de Produto de Software Sensível ao Contexto. Cenário de Teste. Aplicações Sensíveis ao Contexto.

ABSTRACT

A Context-Aware Software Product Line (CASPL) is a product line for developing context-aware applications, which dynamically change their behavior or provide services based on context information. With this kind of line, the productivity can be increased and the development costs of the context-aware application can be cut down. In this scenario, the testing activity needs to deal with the peculiarities of both context-aware applications and the product line development paradigm. Through the complexity involved in testing a CASPL is essential to have methods or tools for supporting this activity, especially with the goal of creating tests from requirements. The aim of this early testing generation is to identify and fix the defects in the early stages of development. Therefore, this work aims to propose an environment for generating test scenarios for a CASPL that takes into account the presence of context information and seeks to maximize the testing reuse. This environment consists of a testing scenario generation method, a template for textual use case specification and a support tool. The method uses as input textual use cases especifications with information about: funcionality, variability, and how the context afect the final products. The environment also has a use case template that supports the use of the method and a tool support, which allows the modeling of use cases according to this template and implements the proposed method. In order to verify the benefits of the environment as a way to testing generation for a Context-Aware Software Product Line, an assessment in the form of controlled experiment is conducted. Based on the collected results, it is observed that: the use of the template makes easy the understanding of a Context-Aware Software Product Line use case; the method favors the creation of a set of tests with higher coverage than a non-systematic testing generation; and the tool is the necessary support for the method.

Keywords: Software Testing. Use Case. Software Product Line. Context-Aware Software Product Line. Test Scenario. Context-Aware Application

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

CAPLUC	***C**ontext **A**ware software **P**roduct **L**ine **U**se **C**ase template*
CASPL	*Context-Aware Software Product Line*
ChAPTER	***C**ontext **A**ware software **P**roduct line **TE**sting gene**R**ation method*
GREat	Grupo de Redes de Computadores, Engenharia de Software e Sistemas
IDE	*Integrated Development Environment*
LPS	Linha de Produto de Software
LPSSC	Linha de Produto de Software Sensível ao Contexto
UML	*Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

Esta dissertação apresenta um ambiente para geração de cenários de testes para Linhas de Produto de Software Sensíveis ao Contexto. Neste ambiente está inserida a proposta de um método de geração de testes, um template para especificação textual de casos de uso para esse tipo de linha e uma ferramenta de apoio ao uso de ambos, o método e o template.

Este capítulo está estruturado da seguinte forma: a Seção 1.1 contextualiza o presente trabalho; em seguida, na Seção 1.2, é apresentada a motivação para o seu desenvolvimento; na Seção 1.3 são expostos os objetivos e as contribuições deste trabalho; e por fim, na Seção 1.4 é descrita a organização desta dissertação.

1.1 Contextualização

O reuso de software pode ser definido como o uso de engenharia de conhecimentos ou de artefatos a partir de sistemas existentes para construir novos sistemas (FRAKES; KANG, 2005). Logo, por meio do reuso sistemático, as empresas desenvolvedoras de software podem obter benefícios como a melhoria da qualidade e a redução dos custos de desenvolvimento (ALMEIDA et al., 2007; FRAKES; KANG, 2005).

Com o intuito de maximizar o reuso de software no desenvolvimento de várias aplicações de um mesmo domínio, surgiu o conceito de Linha de Produto de Software (LPS). Segundo Northrop e Clements (2007), uma LPS pode ser definida como "um conjunto de sistemas de software que compartilham um conjunto de características comuns e que satisfazem as necessidades específicas de um segmento de mercado, sendo desenvolvidas a partir de um conjunto de artefatos comuns de forma sistemática" (tradução nossa). Dessa forma, como os produtos finais de uma LPS reutilizam artefatos comuns entre eles, a abordagem de desenvolvimento em forma de linha de produto maximiza os benefícios do reuso sistemático e planejado no desenvolvimento e teste dos produtos finais da linha.

Tratando-se do processo de desenvolvimento de uma LPS, diferentemente das aplicações tradicionais, existem três atividades essenciais (NORTHROP; CLEMENTS, 2007): i) atividade de desenvolvimento do núcleo de artefatos ou Engenharia de Domínio, que objetiva desenvolver os artefatos que serão reutilizados; ii) atividade de desenvolvimento do produto ou Engenharia da Aplicação, que se utiliza dos artefatos já desenvolvidos para a concretização dos produtos finais; iii) atividade de gerenciamento, que inclui a parte técnica e organizacional.

Já com relação à atividade de testes, ela é fundamental para garantir a qualidade dos produtos da LPS, assim como no caso de aplicações tradicionais. Contudo, no cenário de uma linha de produto essa atividade passa a ser mais complexa do que no caso de uma aplicação tradicional porque ela deve levar em consideração os pontos de variação da linha e os dois processos de desenvolvimento, Engenharia de Domínio e Engenharia de Aplicação (ALI; MOAWAD, 2010).

Ainda com relação a atividade de testes, mediante a presença dos dois processos

de desenvolvimento citados anteriormente, alguns autores consideram que são necessários dois processos de teste de uma LPS (EDWIN, 2007): o do *core assets*, que garante que os artefatos da Engenharia de Domínio foram desenvolvidos corretamente; e o dos produtos finais, no qual procura-se garantir que os produtos em desenvolvimento estão de acordo com os requisitos.

Vale destacar ainda que embora fundamental para a qualidade do produto, a atividade de testes é onerosa porque requer, dentre outras coisas, ferramentas e pessoal especializados (MYERS, 2004). Logo, da mesma forma que nas aplicações tradicionais, o teste dos artefatos da LPS nos estágios iniciais é importante para garantir a correção dos defeitos a um custo mínimo. Adicionalmente, como vários softwares são desenvolvidos compartilhando características comuns, a baixa qualidade de produtos do núcleo de artefatos da linha de produto pode ser propagada para todos os produtos dessa linha (ALI; MOAWAD, 2010).

No caso de uma Linha de Produto de Software Sensível ao Contexto (LPSSC), que é voltada ao desenvolvimento de aplicações sensíveis ao contexto, novos desafios emergem na atividade de testes. Isso ocorre porque as aplicações sensíveis ao contexto apresentam um novo espaço de entrada que pode afetar o comportamento delas em qualquer ponto durante a execução (WANG; ELBAUM; ROSENBLUM, 2007).

Esse novo espaço de entrada diz respeito as informações de contexto, que se referem a qualquer informação que possa ser utilizada para caracterizar a situação de uma entidade (DEY, 2001). Por exemplo, no caso de um guia de visitas móvel e sensível ao contexto de um laboratório de pesquisa, a localização do visitante pode ser a informação de contexto utilizada pela aplicação para fornecer as informações do laboratório adequadas a posição corrente do visitante. Dessa forma, as aplicações sensíveis ao contexto são aquelas que adaptam o seu comportamento com base em informações de contexto (WANG; ELBAUM; ROSENBLUM, 2007)

Por tudo que foi mencionado anteriormente, pode-se concluir que, de uma maneira geral, testar uma LPSSC envolve lidar ao mesmo tempo com os desafios inerentes aos testes de uma LPS, que precisam, por exemplo, maximizar a reutilização; e aos testes de aplicações sensíveis ao contexto, que devem, entre outras coisas, levar em consideração o contexto da aplicação.

1.2 Motivação

Na literatura, conforme é descrito no Capítulo 3, há vários trabalhos que apoiam a geração de teste a partir dos requisitos para aplicações únicas, que não são desenvolvidas por meio de uma LPS (CHATTERJEE; JOHARI, 2010; GUTIERREZ et al., 2008; ROUBTSOV; HECK, 2006), e também há opções para uma LPS (BERTOLINO; GNESI, 2003; NETO et al., 2011; ALI; MOAWAD, 2010; KAMSTIES et al., 2004). Em geral, essas abordagens adotam uma análise dos cenários de caso de uso e produzem algum modelo ou máquina de estados finita para guiar a geração dos testes.

No caso de uma linha de produto de software sensível ao contexto, a qual possui como desafio adicional a presença de informações de contexto influenciando a configuração e a

execução dos produtos da linha, não foi encontrada nenhuma abordagem . Desse modo, existe a necessidade da criação de novos métodos e ferramentas de apoio ao teste baseado em requisitos ou adaptação de propostas semelhantes existentes para a descoberta precoce de defeitos nos produtos de uma LPSSC. Além disso, ao contrário da variedade de templates existentes para especificação textual de casos de uso de uma LPS, que podem ser usados como base por algum método de geração de testes, não foi encontrada nenhuma proposta de template para caso de uso de uma LPSSC.

Vale destacar ainda que Engstrom e Runeson (ENGSTRöM; RUNESON, 2011) afirmam, como um dos resultados da revisão sistemática sobre testes de LPS, que "quase todas as estratégias propostas para teste de LPS são idealistas no sentido que elas colocam requisitos específicos em outras partes do processo de desenvolvimento, além do processo de teste" (tradução nossa). Dessa forma, é interessante que sejam propostas abordagens de geração de testes para LPS que não necessitem de conhecimentos em uma linguagem ou ferramenta específica. Seguindo essas características, existem poucas abordagens na literatura focando na geração de testes a partir de requisitos para uma linha de produto de software. Um exemplo é a proposta de Bertolino e Gnesi (2003), que gera cenários de testes diretamente a partir de especificações textuais de casos de uso.

Portanto, a principal motivação para este trabalho é a ausência de métodos ou abordagens que apoiem a geração de testes a partir de requisitos para uma LPSSC.

1.3 Objetivos e Contribuições

O presente trabalho tem como objetivo propor um ambiente de suporte a geração de testes para Linhas de Produto de Software Sensível ao Contexto. Este ambiente possibilita a geração de testes a partir de requisitos descritos em especificações textuais de casos de uso e não depende de conhecimento em uma linguagem ou ferramenta específica. Para atingir esse objetivo, as seguintes metas devem ser alcançadas:

- Propor um método para geração de testes de aceitação para linhas de produto de software sensíveis ao contexto;
- Propor um *template* de caso de uso para dar suporte ao método, próprio para a descrição de funcionalidades, variabilidades e informações de contexto de uma LPSSC;
- Implementar uma ferramenta de apoio a modelagem de casos de uso segundo o template e a geração dos testes de acordo com o método proposto; e
- Avaliar o ambiente quanto ao processo de geração de testes para uma LPSSC por meio de especificações textuais de casos de uso;

Com a pesquisa relatada nesta dissertação, espera-se como principais contribuições: a) um template para descrição textual de caso de uso próprio para uma LPSSC; b) um método de geração de teste a partir de casos de usos para LPSSC que permite o reuso de artefatos de teste

desde o levantamento de requisitos da linha; c) uma ferramenta de apoio ao uso do template e do método; e d) suporte a prática do *Test Driven Development* (TDD) (ASTELS, 2003) em LPSSC, pois os testes podem ser gerados antes mesmo da implementação dos artefatos da linha.

1.4 Organização da Dissertação

Esta dissertação está organizada em seis capítulos. O presente capítulo descreve uma breve introdução ao tema, contextualizando o assunto abordado, a motivação, os objetivos e as contribuições deste trabalho.

No Capítulo 2 são abordados os principais conceitos relacionados a linhas de produto de software, aplicações sensíveis ao contexto, teste de software e casos de uso. Nesse capítulo também são destacados os desafios de se testar uma linha de produto de software sensível ao contexto.

No Capítulo 3 são descritos os trabalhos relacionados a esta pesquisa. Para identificar tais trabalhos foram definidas previamente as *strings* de busca e os locais de pesquisa.

No Capítulo 4 é apresentado o ambiente de geração de testes para linhas de produto de software sensível ao contexto. Dessa maneira, a proposta do método de geração de testes, do template para casos de uso de LPSSC e a ferramenta de suporte são descritos em detalhes nesse capítulo.

No Capítulo 5 é descrita a avaliação preliminar do ambiente de geração de testes utilizando como alvo um guia de visita móvel e sensível ao contexto que é produto resultante de uma LPSSC.

Por fim, o Capítulo 6 descreve de forma sucinta os resultados alcançados, bem como as conclusões deste trabalho e apresenta possíveis linhas de pesquisa a serem consideradas em trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo descreve e relaciona conceitos relevantes para esta pesquisa. Ele é organizado como se segue: a Seção 2.1 apresenta conceitos sobre aplicações sensíveis ao contexto; na Seção 2.2 são introduzidas as definições acerca de linha de produto de software; a Seção 2.3 apresenta definições da engenharia de requisitos com destaque para as peculiaridades da engenharia de requisitos de uma LPS; a Seção 2.4 descreve em termos gerais a atividade de teste de software, bem como apresenta desafios relacionados a área; e, por fim, as considerações finais desse capítulo são feitas na Seção 2.5.

2.1 Aplicações Sensíveis ao Contexto

Uma aplicação sensível ao contexto é uma aplicação que utiliza o contexto para prover informação relevante e/ou serviços ao usuário, onde a relevância depende da tarefa do usuário (DEY, 2001). O contexto, por sua vez, pode ser definido como sendo "qualquer informação que possa ser utilizada para caracterizar a situação de uma entidade, sendo que uma entidade pode ser uma pessoa, um lugar ou um objeto considerado relevante para a interação entre um usuário e uma aplicação, incluindo o próprio usuário e aplicação" (DEY, 2001, p. 03, tradução nossa).

Dessa forma, a diferença entre aplicações tradicionais, não sensíveis ao contexto, e aplicações sensíveis ao contexto decorre do fato dessa última utilizar o contexto para configurar o seu comportamento. A Figura 2.1 ilustra essa diferença conforme apresentado em (VIEIRA; TEDESCO; SALGADO, 2009). Uma aplicação tradicional possui um processo de informação baseado apenas nas solicitações e entradas explícitas do usuário. Por outro lado, uma aplicação sensível ao contexto considera as informações explícitas fornecidas pelos usuários, aquelas armazenadas em bases de conhecimento contextuais, as inferidas por meio de raciocínio e, ainda, aquelas percebidas a partir do monitoramento do ambiente. Todas essas informações obtidas de forma não-explícita são chamadas de informações contextuais (VIEIRA; TEDESCO; SALGADO, 2009).

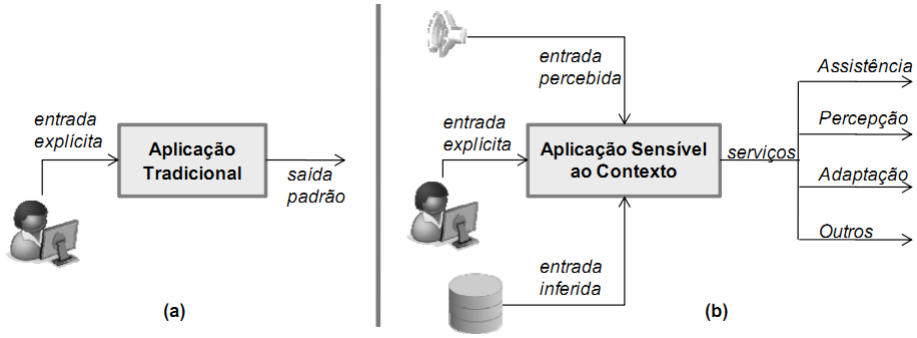


Figura 2.1: Aplicação Tradicional (a) versus Aplicação Sensível ao Contexto (b), Fonte: (VIEIRA; TEDESCO; SALGADO, 2009).

Com base nas informações obtidas, a aplicação sensível ao contexto pode então

executar serviços mais próximos as necessidades do usuário, como por exemplo (VIEIRA; TEDESCO; SALGADO, 2009): (1) assistência na execução da tarefa sendo realizada (e.g., recomendar recursos existentes relacionados à tarefa); (2) percepção do contexto, que se refere a notificar o usuário sobre o contexto associado a pessoas e interações do seu interesse, relativos à tarefa em execução; (3) adaptação, respondendo de forma oportuna às mudanças ocorridas no ambiente e às ações e definições dos usuários (e.g., personalização de interfaces e conteúdo); e (4) outros serviços, como o uso do contexto para enriquecer semanticamente o conhecimento gerenciado pela aplicação.

Em se tratando da história dos sistemas sensíveis ao contexto, essa se inicia com o *Active Badge Location System* (WANT et al., 1992) que é considerado a primeira aplicação sensível ao contexto (BALDAUF; DUSTDAR; ROSENBERG, 2007). Essa aplicação, que era baseada na tecnologia de infravermelho, determinava qual a posição atual do usuário para redirecionar as chamadas telefônicas para o telefone mais próximo ao usuário.

Outros exemplos de aplicações sensíveis ao contexto são os guias de visita sensíveis ao contexto tais como o GUIDE (CHEVERST et al., 2000), o Dynamic Tour Guide (KRAMER et al., 2005) e o GreatTour (MARINHO et al., 2010). O GUIDE fornecia informações sobre a cidade de Lancaster bem como um *tour* com base em informações de contexto do usuário (exemplo, localização e interesses do visitante) e do ambiente (exemplo: hora do dia e hora em que as atrações da cidade estavam funcionando). O Dynamic Tour Guide, por sua vez, é uma aplicação cujo objetivo é prover rotas personalizadas e um conjunto de informações ambientais ao usuário, considerando dentre outras coisas a data, os diferentes tipos de turista e as necessidades de passeio (KRAMER et al., 2005). Por fim, o GreatTour é uma aplicação móvel sensível ao contexto desenvolvida pelo GREat[1] para guiar os visitantes do laboratório deste grupo de pesquisa com base na localização deles.

2.1.1 Arquitetura de aplicações sensíveis ao contexto

No projeto de uma aplicação sensível ao contexto, existem desafios envolvidos, tais como (VIEIRA; TEDESCO; SALGADO, 2009): (i) a caracterização dos elementos contextuais para uso na aplicação e a sua representação em um modelo semântico; (ii) a aquisição dos elementos contextuais a partir de fontes heterogêneas (e.g., sensores físicos, bases de dados, agentes e aplicações); (iii) o processamento e interpretação das informações adquiridas; (iv) a disseminação e compartilhamento dos elementos contextuais entre diferentes aplicações; e (v) a adaptação da aplicação a variações no contexto processado.

Logo, devido a complexidade envolvida no desenvolvimento de uma aplicação sensível ao contexto, a seleção da abordagem de implementação está diretamente relacionada as peculiaridades da aplicação. A seleção da arquitetura, por exemplo, depende de vários fatores, tais como: método de aquisição dos dados do contexto, quantidade de possíveis usuários e recursos disponíveis (BALDAUF; DUSTDAR; ROSENBERG, 2007).

Em se tratando dos recursos disponíveis, a questão é se a aplicação sensível ao

[1] Grupo de Redes, Engenharia de Software e Sistemas

contexto faz uso de computadores de alto desempenho ou de dispositivos móveis. No caso da quantidade de usuários, um grande número de usuários pode exigir uma arquitetura diferente da necessária no caso da aplicação possuir somente um usuário. Por fim, com relação ao método de aquisição dos dados de contexto, existem basicamente três diferentes abordagens (CHEN, 2004):

- Acesso direto ao sensor. Essa abordagem é frequentemente usada quando os dispositivos apresentam os sensores embutidos, pois o software reune a informação desejada diretamente a partir dos sensores físicos. O beneficio desta abordagem é que aplicações de alto-nível tem um melhor conhecimento de como os dados são coletados e computados. Por outro lado, para acessar diretamente os sensores, a aplicação precisa conter implementações para se comunicar com os diferentes sensores, o que dificulta a manutenção da aplicação, principalmente dada a variedade de informações de contexto e sensores existentes;
- Acesso facilitado por uma infraestrutura de *middleware*. Essa abordagem introduz uma camada arquitetural para os sistemas sensíveis ao contexto com a intenção de esconder os detalhes de baixo nível. Com isso, a aplicação não precisa se preocupar em como adquirir o contexto e pode focar em como usar o sensor. Comparado ao acesso direto ao sensor, essa técnica facilita a extensibilidade porque o código do cliente não precisa ser modificado para coletar os dados (BALDAUF; DUSTDAR; ROSENBERG, 2007). A desvantagem desta abordagem é que ela impõe uma computação adicional, referente as operações do *middleware*; e
- Contexto adquirido a partir de um servidor de contexto. Permite múltiplos clientes para acessar recursos de dados remotamente. Logo, essa abordagem distribuida estende a arquitetura baseada em *middleware* introduzindo o componente de acesso remoto e permitindo o reuso dos dados dos sensores por vários clientes.

Visando propor uma arquitetura genérica para aplicações sensíveis ao contexto, Baldauf et al (2007) apresentam uma arquitetura conceitual que separa a detecção e o uso do contexto, melhorando a extensibilidade e o reuso. A Figura 2.2 contém essa arquitetura.

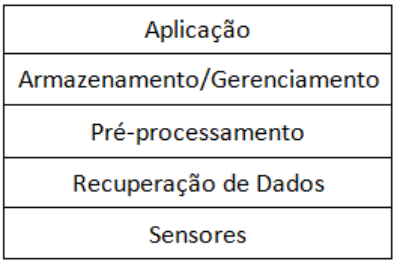


Figura 2.2: Arquitetura conceitual de sistemas sensíveis ao contexto, adaptado de (BALDAUF; DUSTDAR; ROSENBERG, 2007).

A primeira camada consiste de uma coleção de diferentes sensores, que neste caso se referem a qualquer fonte de dados que pode prover informações de contexto. Com relação

a forma como os dados são capturados, os sensores podem ser classificados em três tipos (INDULSKA; SUTTON, 2003): (i) sensores físicos: sensores de hardware que podem capturar dados físicos; (ii) sensores virtuais: proveem dados a partir de aplicações de software ou serviços; e (iii) sensores lógicos: usam várias fontes de informação e combinam sensores físicos e virtuais com informações adicionais de banco de dados ou de outras fontes.

A segunda camada é responsável por recuperar as informações de contexto. Ele utiliza *drivers* apropriados para sensores físicos e APIs (*Application Programming Interface*) para sensores lógicos e virtuais. A camada seguinte, a de pré-processamento, não está presente em todos os sistemas, mas pode ser útil caso os dados obtidos pela camada de recuperação de dados ainda não estejam adequados para uso. Essa camada é responsável por interpretar as informações de contexto. Por exemplo, as exatas coordenadas de uma pessoa obtidas com um GPS (*Global Positioning System*) podem não ter importância para uma determinada aplicação, mas, possivelmente, pode ser relevante o nome da rua ou do bairro no qual a pessoa se encontra.

A próxima camada é a de gerenciamento, que é responsável por armazenar e disponibilizar as informações obtidas para a camada superior.

Por fim, no topo da arquitetura temos a camada de aplicação, que é responsável por reagir aos diferentes eventos e as mudanças de contexto. Essas mudanças podem ocorrer, por exemplo, em função da localização do usuário ou do tipo do usuário.

2.1.2 Modelagem de Contexto

Para uma aplicação utilizar informações de contexto é preciso que seja estabelecido um modelo de contexto, ou seja, uma definição de como os dados de contexto serão armazenados de forma que possam ser processados por uma máquina (BALDAUF; DUSTDAR; ROSENBERG, 2007). Dessa forma, o modelo de contexto é um modelo que representa a estrutura das informações de contexto em uma dada notação (ROCHA; ANDRADE, 2012).

Baseado na estrutura de dados utilizada para representar as informações de contexto, Strang e Linnhoff-popien (STRANG; LINNHOFF-POPIEN, 2004) apresentam seis abordagens diferentes de modelagem de contexto: chave-valor, baseado em esquema de marcação, gráfico, orientado a objetos, baseado em lógica e baseado em ontologias.

O modelo chave-valor representa a estrutura de dados mais simples para se representar contexto. As informações de contexto são descritas por meio de pares compostos por uma chave que identifica a informação e um valor associado a essa chave. Em particular, tal estrutura chave-valor é fácil de gerenciar, mas dado a ausência de uma estrutura sofisticada, ele impossibilita o uso de algoritmos eficientes de recuperação de contexto.

Já o modelo baseado em esquema de marcação utiliza uma estrutura de dados hierárquica composta por *tags* com atributos e valores, baseadas no padrão XML (*eXtensible Markup Language*), para representar as informações de contexto. Os perfis (*profiles*) como o CSCP (*Comprehensive Structured Context Profiles*) (BUCHHOLZ; HAMANN; HUBSCH, 2004) são representantes típicos dessa classe de modelo.

O Modelo gráfico, por sua vez, utiliza elementos gráficos para a representação de contexto e inclui as abordagens baseadas em UML (*Unified Modeling Language*) e ORM (*Object Role Modeling*).

Oferecendo o uso de todo o poder da orientação a objetos (e.g., encapsulamento, herança e reusabilidade) existem os modelos orientados a objetos, nos quais o acesso às informações de contexto é feito apenas por meio de interfaces.

Os modelos baseados em lógica são os que possuem um maior grau de formalismo. Tais modelos são definidos por meio de fatos, expressões e regras. Um sistema baseado em lógica é então usado para gerenciar as informações de contexto e permite adicionar, atualizar ou remover fatos e um processo de inferência pode ser utilizado para derivar novos fatos com base nas regras modeladas (STRANG; LINNHOFF-POPIEN, 2004).

Por fim, existem os modelos que usam ontologias para representar as informações de contexto e, neste caso, é possível usufruir as possibilidades de reuso e os mecanismos de inferência existentes.

2.1.3 Situações de Contexto

As informações dos sensores físicos, apresentados na Seção 2.1.2, são adquiridas sem nenhuma camada de interpretação e podem ser sem significado, vulneráveis a pequenas mudanças ou incertezas (YE et al., 2007). Logo, dadas as limitações do contexto de baixo nível, as aplicações sensíveis ao contexto que usam tal contexto podem ter pouca utilidade (BETTINI et al., 2010).

Para minimizar esse problema, Bettini et al. (2010) recomendam o uso de situações de contexto, que são definidas como uma abstração semântica derivada a partir de informações de baixo nível. Nesse caso, as adaptações ocorrem com base nas mudanças de situações, logo a mudança de contexto só dispara a adaptação se ela provocar uma mudança de situação. As vantagens do uso de situações, é que estas são mais estáveis, e fáceis de definir e manter do que informações contextuais de baixo nível (YE et al., 2007; BETTINI et al., 2010).

A Figura 2.3 apresenta a hierarquia de abstrações sumarizada em (BETTINI et al., 2010). Informações de contexto de baixo nível são semanticamente interpretadas por uma camada de contexto de alto nível. Nesse ponto, situações são abstraídas das informações de baixo nível e são reutilizadas em diferentes ambientes e aplicações. Por fim, relacionamentos entre abstrações podem ser utilizados para reduzir consideravelmente o espaço de busca por situações.

Nesse cenário, é importante ressaltar a diferença entre atributos de contexto, estado de contexto e espaço de situação (PADOVITZ; LOKE; ZASLAVSKY, 2008):

- Atributo de Contexto: é definido como qualquer tipo de dado que é usado no processo de raciocinar sobre o contexto. Esse é associado com um sensor lógico, computado ou físico. Exemplo: o nível de luz pode ser representado por um atributo de contexto "a_1" que é "sentido" por um sensor de luz.

Figura 2.3: Diferentes camadas de abstração de contexto, adaptado de (BETTINI et al., 2010).

- Estado de Contexto: é uma coleção de valores de atributos de contexto que são usados para representar um estado específico do sistema em um tempo t. Um estado de contexto é $C^t = (a_1^t, a_2^t, ..., a_n^t)$ onde C^t denota uma tupla definida sobre uma coleção de n atributos, onde cada valor a_i^t corresponde ao valor do atributo a_i no tempo t. Exemplo: um estado de contexto C^t composto pelos atributos nível de luz a_1^t, nível de barulho a_2^t e nível de movimento a_3^t.

- Espaço de Situação: representa uma situação da vida real e é formado por uma tupla de regiões de valores de atributos que correspondem a alguma situação. Esse espaço de situação é denotado por $S_j = (A_1^j, A_2^j, ..., A_n^j)$, consistindo de n regiões aceitáveis (A) para os n atributos de contexto. Exemplo: Para a situação "pessoa saudável" a região de valores aceitáveis para o atributo de temperatura (a_1^t) do corpo é entre 36.2 °C e 36.9 °C, o que corresponde a região A_1^j.

Nesta seção foram apresentados os principais conceitos relacionados a aplicações sensíveis ao contexto. Como o trabalho proposto visa linhas de produto para aplicações sensíveis ao contexto, na próxima seção são detalhados os conceitos de uma LPS.

2.2 Linha de Produto de Software

Desde o início do desenvolvimento de software na indústria, pesquisadores e profissionais tem procurado por métodos, técnicas e ferramentas que possibilitassem melhorias em custos, *time-to-market* e qualidade (ALMEIDA et al., 2007). Dessa forma, desde o início do desenvolvimento de software tem sido explorado a ideia de reuso (FRAKES; KANG, 2005).

Existem diferentes definições para o reuso de software, mas esta dissertação adota a definição de Frakes e Isoda (2005), na qual o reuso de software é apresentado como "o uso de engenharia de conhecimento ou artefatos de sistemas existentes para construir novos sistemas" (tradução nossa). Tal definição é interessante porque destaca que o reuso não está relacionado

apenas ao reuso de código, mas também ao reuso de outros artefatos, como documentação e do próprio conhecimento envolvido no desenvolvimento de software.

Como uma das formas de maximizar os benefícios do reuso de software surgiram as linhas de produtos de software ou famílias de produto. Segundo a definição de Linha de Produto de Software (LPS) apresentada por Clements e Northrop (2002) na Introdução, a principal característica dos produtos de uma LPS é o fato deles serem desenvolvidos a partir de um conjunto comum de artefatos. Dessa forma, o desenvolvimento de aplicações do mesmo domínio pode se beneficiar desse paradigma para reduzir os custos de desenvolvimento e de teste envolvidos.

Uma vez que o desenvolvimento de linhas de produto já é uma prática industrial, organizações utilizam essa abordagem para, dentre outras coisas, (NORTHROP; CLEMENTS, 2007; ARAúJO, 2010):

- Alcançar ganhos de produtividade: o núcleo de artefatos é reutilizado para instanciar vários sistemas, logo, o custo total para criar cada sistema é reduzido;
- Reduzir o tempo de entrega do produto: 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 LPS;
- Melhorar a qualidade do produto: à medida que o núcleo de artefatos vai sendo reutilizado em novos sistemas, tais artefatos vão sendo revisados e testados. Desta forma, novos sistemas usando esses artefatos tendem a ser mais confiáveis e com maior qualidade do 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; e
- Aumentar a satisfação do usuário: a abordagem de LPS aumenta a qualidade dos produtos de software e possibilita o cumprimento de prazos para entrega.

Com relação ao desenvolvimento de uma LPS, existem duas estratégias principais: engenharia avante ou engenharia reversa. A engenharia avante é usada quando não há sistemas previstos para guiar o desenvolvimento da nova linha de produto. A engenharia reversa, por sua vez, é usada quando há sistemas disponíveis para análise e modelagem, sendo esses sistemas candidatos para modernização e inclusão em um projeto de desenvolvimento de uma LPS (GOMAA, 2004).

Segundo BOSCH (2000 apud ARAÚJO, 2010, pág 30), 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.

2.2.1 Atividades Essenciais

Para guiar o desenvolvimento de uma linha de produto de software, o SEI[2] estabeleceu três atividades essenciais (NORTHROP; CLEMENTS, 2007): desenvolvimento do núcleo de artefatos ou ativos centrais, desenvolvimento do produto e gerenciamento. A Figura 2.4 apresenta a relação entre essas três atividades.

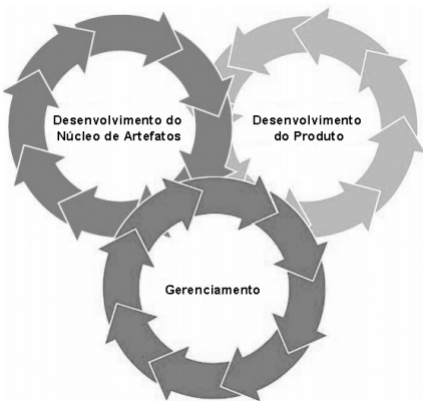


Figura 2.4: As três atividades essenciais para linha de produto de software, traduzida de (NORTHROP e CLEMENTS, 2007) por (ARAÚJO, 2010).

Na Figura 2.4, as flechas rotativas indicam não somente que os ativos centrais (*core assets*) são usados no desenvolvimento de produtos, mas também que revisões de *core assets* existentes ou mesmo de novos *core assets* podem evoluir o desenvolvimento do produto. O gerenciamento por sua vez é responsável por todas as atividades e processos para fazer essas duas atividades trabalharem juntas.

2.2.1.1 Engenharia de Domínio

A atividade de desenvolvimento do núcleo de artefatos, conhecida como Engenharia de Domínio, tem como objetivo o desenvolvimento para reutilização através da implementação de artefatos reutilizáveis para uma linha de produto. O objetivo desta atividade é definir as similaridades e variabilidades da linha, bem como a arquitetura e os componentes reutilizáveis, permitindo a definição do plano de produção dos produtos.

O processo de Engenharia de Domínio pode ser dividido nas seguintes atividades (LINDEN; SCHMID; ROMMES, 2007): (i) Análise do domínio, no qual é feita a coleta de informações e conhecimento sobre uma coleção de sistemas, visando explicitar seu conjunto de similaridades e diferenças; (ii) Projeto do domínio, com o desenvolvimento da arquitetura da LPS; (iii) Implementação do domínio, onde temos a implementação dos artefatos reutilizáveis; (iv) Testes do domínio, que é responsável por validar os componentes implementados na atividade anterior; e (v) Gerenciamento dos produtos, que possui como objetivo definir o escopo da LPS, identificando os produtos que irão constituir a LPS como um todo.

[2]Software Engineering Institute

Vale destacar que os *core assets* incluem, mas não são limitadas a arquitetura, documentação, especificações, componentes de software, modelos de desempenho, planos de testes, casos de testes, planos de trabalho e descrições de processos (NORTHROP, 2002). Logo, o sucesso dos produtos específicos gerados por uma LPS depende da qualidade dos artefatos criados na engenharia de domínio (EDWIN, 2007).

A Figura 2.5 apresenta os artefatos de entrada e saída envolvidos nessa atividade. Como artefatos de entrada tem-se:

- 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 as normas que serão seguidas para a produção dos produtos e os componentes que serão reutilizados;
- Estratégia de produção: descreve a estratégia geral da abordagem para produzir os artefatos da linha de produto; e
- 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.

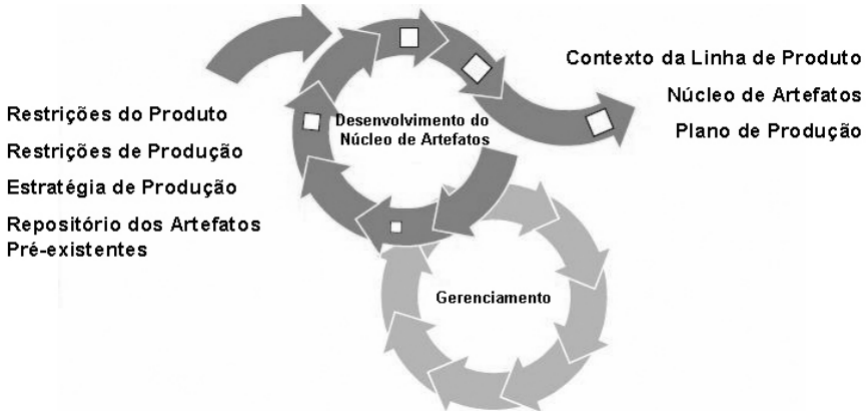


Figura 2.5: Desenvolvimento do Núcleo de artefatos, traduzida de (NORTHROP e CLEMENTS, 2007) por (ARAÚJO, 2010).

Por outro lado, como artefatos de saída, essa atividade possui:

- 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; e

- Plano de produção: descreve as decisões a serem tomadas para instanciar produtos específicos a partir do núcleo de artefatos da LPS.

2.2.1.2 Engenharia de Aplicação

A atividade de desenvolvimento do produto, conhecida como Engenharia da Aplicação, compreende o desenvolvimento com reutilização e tem por objetivo desenvolver os produtos da linha. A Figura 2.6 apresenta os artefatos de entrada e saída envolvidos na atividade de desenvolvimento da aplicação. Os requisitos de um produto específico, bem como 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. Já como saída desse processo tem-se os produtos da linha que são desenvolvidos usando-se o núcleo de artefatos de acordo com o plano de produção para satisfazer os requisitos definidos.

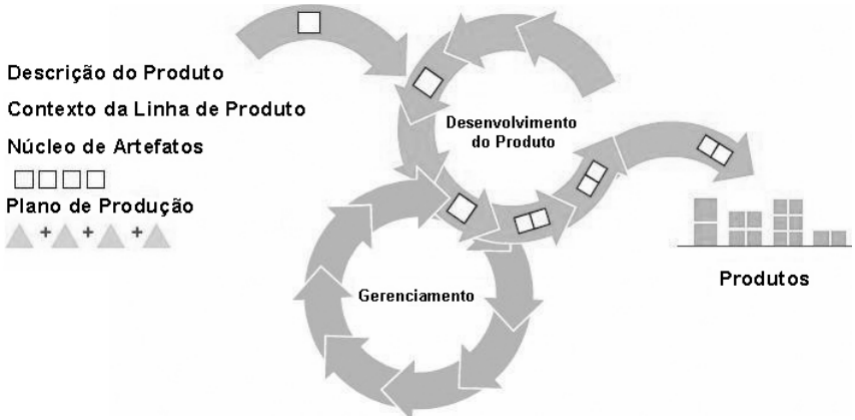


Figura 2.6: Desenvolvimento do produto, traduzido de (NORTHROP e CLEMENTS, 2007) por (ARAÚJO, 2010).

O processo de engenharia da aplicação compreende as seguintes atividades (LINDEN; SCHMID; ROMMES, 2007): (i) Análise da aplicação, que tem como objetivo identificar os requisitos específicos de um determinado produto, tendo como ponto de partida o modelo de domínio da LPS; (ii) Projeto da aplicação, responsável por instanciar e adaptar a arquitetura da LPS de acordo com os requisitos identificados na atividade anterior; (iii) Implementação da aplicação; e (iv) Testes da aplicação, que é responsável pela validação do produto.

Destaca-se que como as atividades são interligadas e iterativas, a atividade de desenvolvimento do produto pode afetar as outras atividades. Pode-se citar, por exemplo, a existência e a disponibilidade de um produto específico que podem afetar os requisitos de um produto subsequente. Assim, durante a execução dessa atividade é possível identificar requisitos que, inicialmente, não haviam sido especificados e, consequentemente, deve-se atualizar o núcleo de artefatos da linha de produto.

2.2.1.3 Gerenciamento

A terceira atividade essencial é o gerenciamento, que inclui tanto o gerenciamento técnico quanto o organizacional. Enquanto o gerenciamento organizacional é responsável pelas estruturas e recursos organizacionais (e.g., pessoal capacitado) e pelas estratégias de produção, o gerenciamento técnico é responsável por garantir que os *core assets* e os produtos estão sendo desenvolvidos conforme as atividades e o processo definido para a linha de produto, além de coletar dados para acompanhar o progresso.

Destaca-se como um dos artefatos mais importantes dessa atividade o plano de adoção (NORTHROP, 2002), que descreve o estado desejado da organização e uma estratégia para alcançar tal estado. Além disso, o gerenciamento técnico e organizacional também contribui para o núcleo de artefatos deixando disponível para reuso artefatos de gerenciamento como orçamentos e cronogramas usados no desenvolvimento dos produtos na linha de produto.

2.2.2 Modelo de Características

Quando se trata de LPS, *feature* (característica) e *feature model* (modelo de características) são dois conceitos fundamentais. Uma característica é um atributo, qualidade ou aspecto visível ao usuário (KANG et al., 1990). Já um modelo de características é uma abstração em alto nível que reune as similaridades e variabilidades de uma LPS em função das suas características (FERNANDES, 2009). Nesse modelo, as características são organizadas em árvores de nós E e OU que representam as similaridades e variabilidades na modelagem do domínio. Características mais gerais ficam no topo da árvore enquanto que aquelas mais específicas são localizadas abaixo. A raiz da árvore, por sua vez, representa o sistema completo. Desse modo, um modelo de características pode prover uma notação compacta para descrever relações complexas entre diferentes características dentro de um domínio (ERIKSSON; BÖRSTLER; BORG, 2004)

De acordo com a abordagem original proposta por Kang et al. (1990), as características podem ser "obrigatórias", "opcionais" e "alternativas" e podem ter relações do tipo "requer" ou "exclui" com outras características. Características obrigatórias são aquelas disponíveis em todos os sistemas construídos dentro da família. Características opcionais são aquelas características que podem ou não ser incluídas nos produtos. Características alternativas representam uma seleção "exatamente-um-de-muitos" que é feita dentro de um conjunto de características. O relacionamento "requer" indica que uma característica depende de outra, enquanto que a relação "exclui" entre duas características indica que ambas não podem ser incluídas no mesmo sistema.

A Figura 2.7 apresenta um excerto do modelo de características de uma linha de produto móvel e sensível ao contexto, destinada ao desenvolvimento de guias de visita móveis e sensíveis ao contexto. Conforme exibido nessa figura, as *features Show Environment Profile*, *List items*, *Show item Profile* e *Show Documents*, *Text* são obrigatórias. As demais características são opcionais, sendo que os pares *Polychromatic e Monochromatic* e *Text e Cloud Tags* são *features* alternativas.

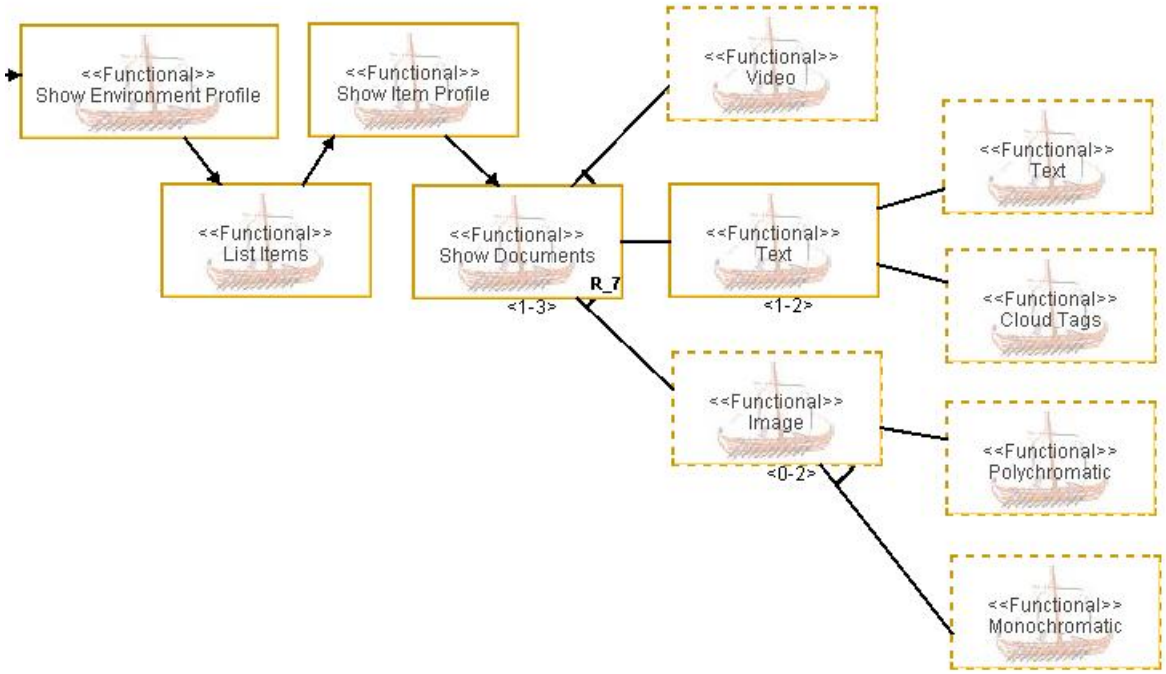


Figura 2.7: Um excerto de um modelo de características.

2.3 Engenharia de Requisitos

Um requisito é uma condição ou habilidade necessária para um sistema alcançar um determinado objetivo ou finalidade (LIMA, 2009). O IEEE (1990) define o termo requisito como:

1. Uma condição ou capacidade necessária por um usuário para resolver um problema ou alcançar um objetivo.
2. Uma condição ou capacidade que deve ser satisfeita ou possuída por um sistema ou componente do sistema para satisfazer um contrato, norma técnica, especificação ou outro documento formalmente imposto.
3. Uma representação documentada de uma condição ou capacidade como em (1) ou (2).

Logo, quando os requisitos de um sistema são definidos, tem-se o estabelecimento do que o sistema deve fazer. Esses requisitos são geralmente apresentados em dois níveis: i) em alto nível para os usuários finais e clientes; e ii) com uma especificação mais detalhada do sistema para os desenvolvedores.

Nesse cenário, o processo que envolve todas as atividades exigidas para criar e manter o documento de requisitos do sistema é chamado de engenharia de requisitos. A Figura 2.8 ilustra as quatro atividades genéricas do processo de engenharia de requisitos (SOMMERVILLE, 2003): estudo de viabilidade, levantamento e análise de requisitos, especificação de requisitos e validação de requisitos.

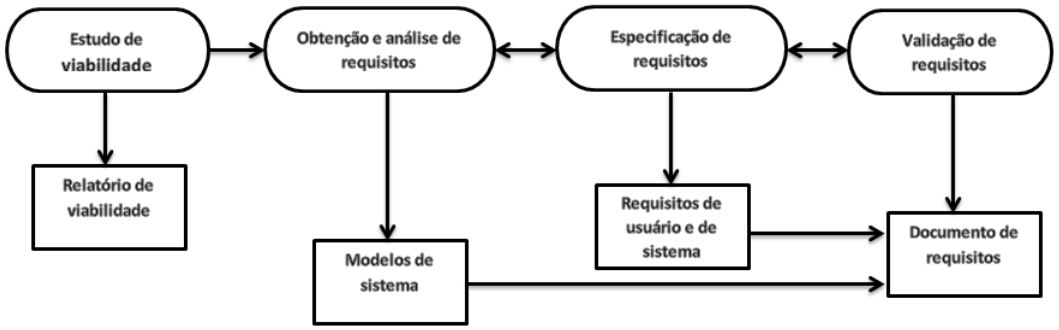


Figura 2.8: Processo de Engenharia de Requisitos, adaptado de (SOMMERVILLE, 2003).

Durante o Estudo de Viabilidade, é feita uma estimativa para verificar se as necessidades dos usuários que foram identificadas podem ser satisfeitas com a utilização das atuais tecnologias de software e hardware. Essa atividade é relativamente rápida e barata e o resultado deve informar se a decisão é a de prosseguir com uma análise mais detalhada.

A atividade seguinte, de Levantamento e Análise de Requisitos, corresponde ao processo de obter os requisitos do sistema pela observação de sistemas existentes, pela conversa com usuários e compradores em potencial, pela análise de tarefas e assim por diante. Além disso, essa etapa pode envolver o desenvolvimento de um ou mais diferentes modelos e protótipos de sistema para auxiliar o analista a compreender o sistema a ser especificado.

A atividade de Especificação de Requisitos, por sua vez, é responsável por traduzir as informações coletadas durante a atividade de análise em um documento que defina um conjunto de requisitos. Nesse documento são incluídos requisitos do usuário, que são declarações em alto nível, e requisitos do sistema, cuja descrição é mais detalhada.

Por fim, a última atividade, de Validação de requisitos verifica os requisitos quanto a sua pertinência, consistência e integralidade. Erros no documento de requisitos podem ser descobertos, então os requisitos são modificados para corrigir tais erros.

2.3.1 Caso de Uso

Em geral, a linguagem natural é utilizada para especificar os requisitos do sistema, contudo, o uso desta pode causar alguns problemas, tais como ambiguidade e ausência de meio para padronizar os requisitos (SOMMERVILLE, 2003). Para evitar tais problemas, como alternativas ao uso da linguagem natural existem linguagens estruturadas, notações gráficas e especificações matemáticas.

Destas alternativas destacam-se os casos de uso, porque estes podem ser usados tanto para levantar requisitos quanto para especificar os mesmos. Além disso, o modelo de casos de uso apresenta uma visualização gráfica por meio de diagramas de casos de uso que facilitam o entendimento.

A técnica de casos de uso foi introduzida por Jacobson na década de 90 como parte da metodologia de desenvolvimento de software OOSE (Object-Oriented Software Enginee-

ring) (JACOBSON et al., 1992 apud ARAÚJO, 2010, pág 39). Um caso de uso define "um conjunto de interações orientado a objetivos entre atores externos e o sistema sob consideração" (FANTECHI et al., 2003, p. 02, tradução nossa). Eles se referem a serviços, tarefas ou funções oferecidas pelo sistema, como emitir um relatório ou cadastrar a venda de algum produto (GUEDES, 2007). Com isso, a visão de caso de uso mostra as funções que o sistema deve executar e serve como um contrato entre o cliente e o desenvolvedor (LIMA, 2009).

Casos de uso podem ser usados (ERIKSSON; BÖRSTLER; BORG, 2004): (i) como base para gerentes de projeto fazerem estimativas de tempo; (ii) por designers para relacionar designs específicos para requisitos; e (iii) por testadores, que podem criar cenários de teste baseado no comportamento e nas condições descritas nos casos de uso.

Em geral, casos de uso utilizam uma linguagem descritiva natural para descrever os requisitos do sistema. Ao definir um caso de uso para uma dada função do sistema, é possível "capturar quem (ator) faz o que (interação) com o sistema e para que propósito (objetivo)" (FANTECHI et al., 2003, p. 03, tradução nossa).

Um caso de uso pode possuir dois tipos de fluxo de eventos: o fluxo básico e um ou mais fluxos alternativos. O fluxo básico especifica o que normalmente acontece quando o caso de uso é executado. É o fluxo mais óbvio, no qual todas as ações são bem-sucedidas. Os fluxos alternativos descrevem comportamentos opcionais ou excepcionais, e são variações do fluxo básico (LIMA, 2009).

O conteúdo do caso de uso descreve um conjunto de cenários especificando sequências de interação entre o sistema e os atores do sistema. O ator define um conjunto de papéis que os usuários podem exercer ao interagir com o sistema. O ator é sempre externo ao sistema e pode representar papéis executados por usuários humanos, hardware externo ou outros sistemas (LIMA, 2009).

Com respeito aos atores do sistema, esses podem ser classificados de acordo com a interação com o sistema em dois níveis (LIMA, 2009): ator primário, que quase sempre é o que inicia um caso de uso e é o interessado que acessa o sistema para entregar-lhe diretamente um serviço; ator secundário (ou ator de suporte), que é um interessado que provê um serviço ao sistema, mas não diretamente, como por exemplo, uma impressora.

Um cenário, por sua vez, é um fluxo de eventos de um caso de uso, uma narração de uma sequência de solicitações e respostas entre um usuário (ator) e o sistema, que mostra como esse usuário usa o sistema para alcançar um objetivo significante (LIMA, 2009).

A Figura 2.9 apresenta um exemplo de caso de uso para efetuar uma compra em uma loja online fictícia. Neste exemplo, o caso de uso tem um fluxo básico de oito passos e um fluxo alternativo, que é referente a um consumidor regular. Por outro lado, a Figura 2.10 apresenta o cenário "Comprar um produto", que corresponde ao fluxo básico do caso de uso apresento na Figura 2.9.

Como um meio para representar graficamente os casos de uso tem-se os diagramas de casos de uso, que exibem os relacionamentos entre os atores e os casos de uso. Tais diagramas mostram apenas símbolos, mas, na verdade, os casos de uso são escritos na forma de texto

1. O consumidor navega através do catálogo e seleciona itens para comprar;
2. O consumidor faz o *check out*;
3. O consumidor preenche as informações para entrega dos produtos (endereço, forma de entrega);
4. O sistema apresenta informação dos preços, incluindo o frete;
5. O consumidor preenche as informações do cartão de crédito;
6. O sistema autoriza a compra;
7. O sistema confirma a venda imediatamente;
8. O sistema envia um e-mail de confirmação para o consumidor;

Alternativa: Consumidor regular

3a. O sistema exibe as informações de entrega atuais, de preço e os últimos quatro dígitos do cartão de crédito;
3b. O consumidor pode aceitar ou editar;
Retornar para o cenário primário no passo 6.

Figura 2.9: Exemplo de caso de uso (FOWLER e SCOTT, 2000, p. 40).

"Comprar um produto" de uma loja on-line: "O usuário navega o catálogo e adiciona os itens desejados a cesta de compras. Quando o cliente selecionar a forma de pagamento, ele descreve as informações do cartão de crédito e confirma a compra. O sistema verifica a autorização e confirma a venda".

Figura 2.10: Exemplo do cenário (FOWLER e SCOTT, 2000, p. 40).

simples e bastante compreensível (LIMA, 2009). A Figura 2.11 apresenta um exemplo de um diagrama de caso de uso, no qual tem-se três atores (Cliente, Vendedor, Supervisor) interagindo com três casos de uso (Colocar pedido, Verificar situação, Estabelecer pedido) de um sistema de "Vendas".

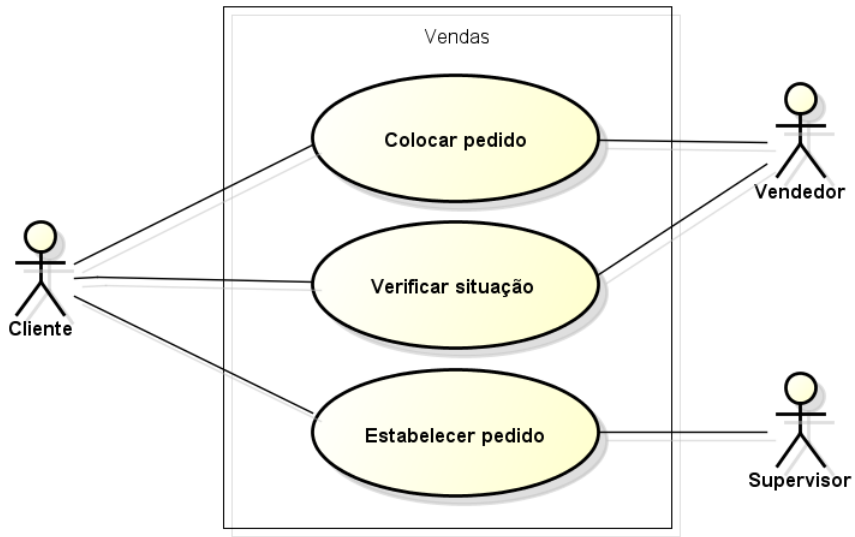


Figura 2.11: Exemplo de diagrama de caso de uso, adaptado de (LIMA, 2009, p. 59).

No diagrama de caso de uso, um caso de uso é representado por uma elipse. O retângulo (opcional) mostra os limites do sistema, contendo em seu interior o conjunto de casos de uso e, do lado externo, os atores, cujo ícone de estereótipo padrão é a figura de um "homem-palito", interagindo com o sistema. Cada linha entre um ator e um caso de uso mostra uma associação entre eles.

2.3.2 Engenharia de Requisitos de LPS

Em se tratando de LPS, dois diferentes problemas devem ser abordados durante a elicitação e especificação de requisitos: (a) como identificar os requisitos comuns e variáveis entre todos os membros da linha de produto; e (b) como especializar e instanciar requisitos genéricos da linha de produto para obter os requisitos de uma aplicação específica (NORTHROP; CLEMENTS, 2007; GALLINA et al., 2006)

Dados a complexidade e a natureza extensiva do desenvolvimento de linhas de produto, a engenharia de requisitos é muito importante para a prática da linha de produto de software (KULOOR e EBERLEIN, 2002). Em uma linha de produto de software, existem vários requisitos que são comuns através da família de produtos, enquanto que outros são únicos de produtos individuais. Além disso, tem-se vários consumidores e interessados envolvidos. Com isso, um processo de engenharia de requisitos bem estabelecido, é essencial para o desenvolvimento de uma linha de produto de software (KULOOR e EBERLEIN, 2002).

No contexto de LPS, os requisitos definem as aplicações e suas *features*, além das restrições sobre essas aplicações. As principais diferenças entre a engenharia de requisitos para LPS e a de sistemas únicos são (NORTHROP; CLEMENTS, 2007):

- 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;

- Especificação de Requisitos: essa atividade inclui não somente 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.

2.4 Teste de Software

Existem várias definições diferentes para teste de software. Myers (2004), por exemplo, define testes como sendo o processo de executar um programa com o objetivo de encontrar defeitos. Já de acordo com um relatório do Instituto de Engenharia de Software - SEI, teste é uma abordagem para validar e verificar os artefatos produzidos no desenvolvimento de software (MCGREGOR, 2001).

Uma definição mais extensa pode ser encontrada em (IEEE, 2004, p. 4-5, tradução nossa), na qual teste de software é definido como a "verificação dinâmica do funcionamento de um programa utilizando um conjunto finito de casos de teste, adequadamente escolhido dentro de um domínio de execução, em geral, infinito, contra seu comportamento esperado". Essa última é interessante por destacar que nos testes existe a comparação com resultados esperados e que, em geral, não é possível testar todos os casos de testes possíveis.

Um caso de teste é uma especificação de como o software deve ser testado através da definição das entradas e saídas esperadas, bem como das condições sob as quais o teste deve ocorrer (PRESSMAN, 2006). A Tabela 1 apresenta um exemplo de um caso de teste para o login da página de webmail do GREat - Grupo de Redes, Engenharia de Software e Sistemas da Universidade Federal do Ceará. O caso de teste da Tabela 1 utiliza o modelo de casos de testes proposto por Padua Filho (2003). Relacionados aos casos de testes existem os procedimentos de testes, que são sequências de instruções para configuração, execução e avaliação dos resultados de um dado caso de teste (IEEE, 1990). A Tabela 2 ilustra um exemplo de procedimento de teste para efetuar o login no GREat.

Identificação	UFC-ETF-LO-CT-AU1	
Itens a testar	Verificar se a tentativa de efetuar o login utilizando uma senha não cadastrada exibe mensagem apropriada	
Entradas	**Campo**	**Valor**
	Utilizador	Ismaylesantos
	Senha	Great123
Saídas Esperadas	**Campo**	**Valor**
	Nome	Ismaylesantos
	Senha	
	Mensagem	Erro ao entrar
Ambiente	Banco de dados de teste	
Procedimentos	Login – UFC-ETF-LO-PT-AU	
Dependências	Não aplicável	

Figura 2.12: Exemplo de caso de teste.

Embora os testes sejam importantes para a qualidade de um software, os custos associados a essa atividade podem chegar a mais de 50% do custo total de um projeto (PRESSMAN, 2006). Isso porque os testes requerem tempo, conhecimento, planejamento, infraestrutura e pessoal especializado, sendo, portanto, uma atividade inerentemente custosa (MYERS,

Identificação	UFC-ETF-LO-PT-AU
Objetivo	Efetuar o login no sistema
Requisitos especiais	Nenhum
Fluxo	1. Preencher o campo “Utilizador” 2. Preencher o campo “Senha” 3. Clicar em “Ok”

Figura 2.13: Exemplo de procedimento de teste.

2004). Dessa forma, ferramentas ou métodos que auxiliem de alguma forma essa atividade são de grande importância para reduzir os custos dessa atividade.

Os testes podem ser classificados quanto ao objetivo dele. Como por exemplo, testes funcionais são feitos para verificar se as especificações estão corretamente implementadas (PADUA FILHO, 2003). Também existem outros tipos de testes, tais como os testes de desempenho para verificar a adequação aos requisitos de desempenho e os testes de segurança que avaliam o nível de segurança da aplicação.

É importante destacar que os testes assumem um papel importante durante todo o ciclo de desenvolvimento de software. A Figura 2.14 apresenta o modelo-V, no qual as tarefas de desenvolvimento e as tarefas de testes são atividades correspondentes de igual importância (SPILLNER et al., 2006). Neste modelo, para os quatro estágios de desenvolvimento, existem quatro níveis de testes: testes de unidade, testes de integração, testes de sistema e testes de aceitação.

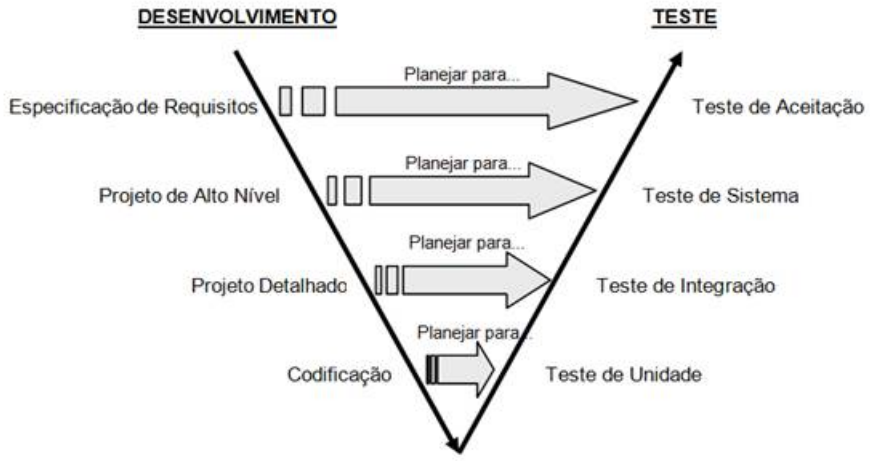


Figura 2.14: Modelo V, extraído de (DEVMEDIA, 2012).

Os testes de unidade são feitos para avaliar as unidades produzidas na fase de codificação (MACHADO, 2010). Tipicamente, testes de unidade ocorrem com acesso ao código que está sendo testado e no ambiente de desenvolvimento (MÜLLER et al., 2007). Os testes de integração, por sua vez, avaliam o quanto as interfaces entre as unidades de um dado subsistema tem consistência e se comunicam corretamente (MACHADO, 2010). Já os testes de sistema verificam o comportamento de todo o sistema. Por fim, os testes de aceitação são frequentemente de responsabilidade do usuário do sistema e tem por objetivo estabelecer a confiança no sistema.

2.4.1 Teste em aplicações sensíveis ao contexto

Diferentemente de aplicações tradicionais, onde o comportamento está incluso no código implementado, aplicações sensíveis ao contexto tem parte do seu comportamento determinado de acordo com o contexto (TSE et al., 2004). Assim, o problema em se testar aplicações sensíveis ao contexto está relacionado ao desafio de prever todas as mudanças de contexto relevantes e quando elas podem impactar o comportamento deste tipo de aplicação (WANG; ELBAUM; ROSENBLUM, 2007). Baseado neste contexto, Lu (LU, 2009) descreve em seu trabalho quatro desafios em se testar aplicações sensíveis ao contexto:

- Formatos ricos. Dados de contexto podem ser capturados a partir de uma variedade de diferentes fontes dinamicamente. Os dados obtidos são então representados em formatos ricos e hierarquias semânticas. Testadores precisam cuidadosamente replicar ou simular valores de contexto enquanto projetam entradas de teste. Além disso, Lu (2009) destaca que ao desenvolver uma ferramenta ou método de teste deve-se lembrar que contextos têm estruturas de representação interna que variam de simples modelo chave-valor para representações gráficas e ontologias;

- Contextos Voláteis. Contextos podem variar rapidamente (TSE et al., 2004; WANG; ELBAUM; ROSENBLUM, 2007). Quando a execução de um programa usa um contexto particular para derivar sua resposta ao usuário, o contexto já pode ter sido substituído por um novo contexto. A execução de um programa é então fragmentada em vários segmentos tais que diferentes segmentos podem se referir a diferentes contextos. Assim, diferentemente de um programa tradicional no qual é produzido o mesmo resultado para o mesmo valor de entrada, a entrada de uma aplicação sensível ao contexto pode induzir diferentes execuções resultantes de acordo com os diferentes contextos sob os quais a aplicação executou;

- Sensibilidade ao contexto egocêntrica. Segundo Lu (2009), entidades computacionais são usualmente modeladas como serviços e tem comportamento egocêntrico se preocupando apenas com as informações de contexto que lhe interessam. Dessa forma, o contexto observado, por exemplo, na presença de ruído, pode não ser exatamente o mesmo da situação esperada para disparar um dado serviço fazendo com que o serviço não seja invocado quando deveria; ou

- Soft Computing. Contextos físicos são imprecisos por natureza. Logo, quando a execução de um programa gera um resultado é difícil determinar quando ele é (i) correto; (ii) errado devido a falha de software; ou (iii) errado devido a imprecisão da informação de contexto.

2.4.2 Teste em linhas de produto de software

Testar uma LPS envolve examinar o *core assets* da linha, os produtos específicos e a interação entre esses produtos e o *core assets* (NORTHROP, 2002). Em se tratando de linhas de produto de software existem, portanto, dois processos de testes: dos *core assets* e da aplicação (EDWIN, 2007).

O principal objetivo do processo de teste dos *core assets* é garantir que os componentes que estão sendo desenvolvidos como *core assets* da linha de produto funcionam corretamente (EDWIN, 2007). Enquanto que o objetivo do teste do produto é garantir que o produto sendo desenvolvido é o produto especificado nos requisitos.

Dado as peculiaridades de uma LPS, o processo de teste dessa não segue o modelo-V apresentado no início dessa seção. Como alternativa ao modelo-V, para melhor se adequar a LPS foi proposto por Jin-hua at el. (2008) o modelo W, ilustrado na Figura 2.15.

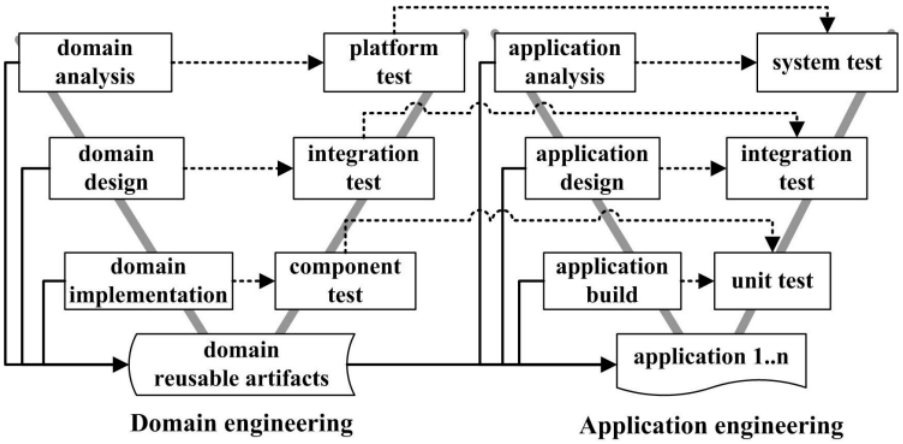


Figura 2.15: Modelo W, extraído de (JIN-HUA; QIONG; JING, 2008).

O modelo W aborda o teste durante a engenharia de domínio e durante a aplicação. Na engenharia de domínio, o teste de componentes, de integração e de plataforma são executados sobre os componentes reutilizáveis, a arquitetura da linha e a plataforma comum. No teste de componente do domínio podem ser usadas as técnicas tradicionais tais como teste de caixa branca. Devido a variabilidade, é impossível testar exaustivamente as interações entre os componentes, logo, em geral, são realizados casos de testes para as interações comuns e aquelas que contém poucas interações variáveis. Além disso, como na engenharia de domínio não existe um sistema completo não é possível executar um teste de sistema completo na engenharia de domínio e, com isso, é possível projetar os cenários de testes, mas não é possível executá-los (JIN-HUA; QIONG; JING, 2008).

No caso da engenharia de aplicação, para validar os produtos finais são executados testes de unidade, integração e sistema. Ressalta-se que nesta etapa são reutilizados os procedimentos de testes, casos de testes, dados de testes e outros artefatos de testes resultantes da engenharia de domínio para aumentar a qualidade dos testes da aplicação (JIN-HUA; QIONG; JING, 2008).

Dessa forma, dado o impacto da variabilidade da linha na atividade de testes, existem quatro diferentes estratégias de teste (JIN-HUA; QIONG; JING, 2008):

- Força-Bruta: ele executa todas as atividades de testes em todos os níveis para todas as possíveis aplicações na engenharia de domínio, logo, não é necessário realizar testes na engenharia da aplicação. Essa estratégia não é aplicável na prática devido a explosão de configurações possíveis.

- Estratégia de Aplicação Pura: na qual os testes são executados somente na engenharia da aplicação e, nesse caso, nenhum teste é executado na engenharia de domínio. Os problemas dessa abordagem são que o teste só executa quando a aplicação é totalmente finalizada, e para os testes da aplicação final os artefatos de testes são feitos do zero, não obtendo vantagem com uma reutilização.

- Estratégia de Aplicação de Exemplo: usa uma ou várias aplicações exemplo para testar os artefatos do domínio e na engenharia de aplicação cada aplicação é sistematicamente testada. A vantagem dessa abordagem é que ela garante a qualidade dos artefatos do domínio.

- Estratégia de Reuso: essa estratégia divide o teste da LPS em duas partes: teste de domínio e teste da aplicação. O de domínio tem por objetivo testar as partes comuns e preparar o teste das partes variáveis. O teste da aplicação objetiva reusar os artefatos do teste do domínio para testar uma aplicação específica. A vantagem dessa abordagem é o reuso sistemático dos artefatos de testes.

Com relação aos desafios de se testar LPS, segundo Edwin (2007), os maiores são a variabilidade dos *core assets* da linha de produto de software e os dois processos de desenvolvimentos envolvidos nas LPS. Edwin também afirma que não existem ferramentas de suporte específicas para todo o processo de teste de uma linha de produto de software. O que existe são ferramentas para teste de LPS que "são extensões de diferentes ferramentas usadas no processo de teste tradicional para acomodar o processo de teste de linha de produto de software" (Edwin, 2007, p.86, tradução nossa).

Além disso, são também desafios envolvidos no teste de uma linha de produto de software a criação dos casos de testes e a seleção dos casos de testes (LEE; KANG; LEE, 2012). Os casos de testes do domínio devem ter formas que podem ser eficientemente reutilizadas no teste das aplicações e precisam ser capazes de conter variabilidades. Por outro lado, deve existir um meio para testar os produtos da linha que reutilize os casos de testes do domínio e minimize o reteste para outras partes que já foram testadas na engenharia de domínio. Logo, o método de seleção dos casos de testes deve selecionar casos de testes do domínio para serem executados.

Tentando superar alguns desafios dessa área, autores tem proposto na literatura a reutilização de casos de testes e de outros artefatos de testes através da linha de produto (MCGREGOR, 2001; NEBUT et al., 2003; EDWIN, 2007).

2.5 Conclusão

Este capítulo introduziu os conceitos básicos sobre Aplicações Sensíveis ao Contexto, Linhas de Produto de Software, Engenharia de Requisitos e Teste de Software.

Relacionado a aplicações sensíveis ao contexto, alguns exemplos foram apresentados e foi detalhada a diferença para as aplicações tradicionais. Essa diferença provém da

presença de informações de contexto influenciando a configuração e o comportamento das aplicações sensíveis ao contexto. Além disso, foram apresentados alguns conceitos sobre modelagem de contexto e arquitetura de uma aplicação sensível ao contexto, bem como foi descrito a definição de situação de contexto. Essa última é importante porque como casos de uso são feitos durante a etapa de levantamento de requisitos, as informações sobre a LPS são descritas em alto nível nos casos de uso. Logo, o uso de situações no caso de uso se torna mais apropriado do que a definição em baixo nível do contexto.

Com respeito a linhas de produto de software, foi apresentado neste capitulo as vantagens associadas ao seu uso e foi descrito em detalhes as atividades essenciais de desenvolvimento: desenvolvimento do núcleo de artefatos, desenvolvimento do produto e gerenciamento. Vale ressaltar que quando a linha de produto é destinada ao desenvolvimento de aplicações sensíveis ao contexto tem-se uma Linha de Produto de Software Sensível ao Contexto.

Na seção sobre Engenharia de Requisitos, após a apresentação das atividades genéricas do processo de engenharia de requisitos, foram introduzidos os principais conceitos sobre casos de uso e cenários de uso. Além disso, as diferenças entre o processo de engenharia de requisitos de aplicações únicas e de LPS foram descritas nessa seção.

Finalmente, na seção sobre testes de software foi destacado os desafios relacionados ao teste de aplicações sensíveis ao contexto e ao teste de linhas de produto de software. No caso de uma linha de produto de software sensível ao contexto, os desafios associados a atividade de testes resultam da junção dos desafios relacionados ao teste de aplicações sensíveis ao contexto e ao teste de uma LPS.

3 TRABALHOS RELACIONADOS

Neste capítulo são apresentados os trabalhos encontrados durante a pesquisa que estão relacionados com a ideia dessa pesquisa. Ele é organizado como se segue: na Seção 3.1 é apresentada a metodologia de busca utilizada para identificar os trabalhos relacionados; na Seção 3.2 são apresentados os trabalhos encontrados para especificação de casos de uso para linhas de produto de software; a Seção 3.3 descreve especificações de casos de uso para aplicações sensíveis ao contexto; a Seção 3.4 apresenta alguns métodos para geração de testes a partir de casos de uso; e por fim, na Seção 3.5 são apresentadas as considerações finais deste capítulo.

3.1 Metodologia de busca

Para encontrar os trabalhos relacionados foi realizada uma busca na literatura por publicações que estivessem relacionados a pelo menos um dos seguintes tópicos: i) Especificação de casos de uso para LPS, procurando identificar como as variabilidades de uma LPS podem ser modeladas no caso de uso; ii) Especificação de casos de uso para aplicações sensíveis ao contexto, para levantar como o impacto do contexto nos produtos pode ser modelado em casos de uso; e iii) Teste baseado em casos de uso, para identificar as abordagens existentes para geração de testes a partir de descrições textuais de caso de uso.

É importante mencionar que esses tópicos foram definidos porque representam aspectos fundamentais dessa pesquisa. Isso porque o método proposto neste trabalho utiliza casos de uso para gerar testes e como não existia um template próprio para casos de uso de LPSSC, foi definido um template que permite modelar as variabilidades da linha e as influências do contexto no comportamento e configuração dos produtos finais.

Ainda com relação as buscas, embora não tenha sido executada uma revisão sistemática (KITCHENHAM, 2004), alguns princípios desse processo de pesquisa bibliográfica, tais como a definição de *strings* de busca e locais de pesquisa, foram adotados neste trabalho. Dessa forma, para cada um dos três tópicos foram definidas duas strings de busca, uma inglês e outra em português. Como locais de buscas utilizou-se as bases ACM DL Library [1] e o IEEE Explorer [2], bem como os anais do Simpósio Brasileiro de Engenharia de Software - SBES (desde 2000), do Simpósio Brasileiro de Componentes e Arquitetura de Sistemas - SBCARS (desde a primeira edição em 2007) e do Simpósio Brasileiro de Qualidade de Software - SBQS (desde a primeira edição em 2002, mas com exceção dos anos 2005, 2007 e 2008 por indisponibilidade dos anais referentes a tais anos).

Além disso, buscando por *grey literature* (e.g., relatório técnicos e trabalhos em progresso), foram executadas buscas no Google[3] e no Google Scholar [4]. Adicionalmente, su-

[1] http://http://dl.acm.org
[2] http://http://ieeexplore.ieee.org
[3] https://www.google.com/
[4] http://scholar.google.com/

gestões de especialistas das áreas correlatas aos tópicos de pesquisa também foram considerados na busca por trabalhos relacionados. A Tabela 3.1 apresenta as strings e locais de busca e sumariza os resultados obtidos. A quantidade de resultados obtidos representa o total de trabalhos retornados pelo local de busca. A quantidade de resultados classificados, por sua vez, indica quantos dos artigos obtidos foram selecionados, com base na leitura do título e resumo, para serem lidos por inteiro.

Tabela 3.1: Strings e Fontes de Buscas utilizadas para levantar os trabalhos relacionados.

Strings de Pesquisa	(“software product line” **OR** “software product lines”) **AND** (“use case” **OR** “use cases”)	(“context aware” **OR** “context awareness”) **AND** (“use case” **OR** “use cases”)	“test” **AND** (“use case” **OR** “use cases”)	(“linha de produto de software” **OU** “linhas de produto de software”) **E** (“caso de uso” **OU** “casos de uso”)	(“sensível ao contexto” **OU** “sensíveis ao contexto”) **E** (“caso de uso” **OU** “casos de uso”)	“teste” **E** (“caso de uso” **OU** “casos de uso”)
Fontes de Busca	**Classificados/ Obtidos**	**Classificados/ Obtidos**	**Classificados/ Obtidos**	**Classificados/ Obtidos**	**Classificados/ Obtidos**	**Classificados/ Obtidos**
ACM	**31 / 745**	**5 / 656**	**20 / 95**	-	-	-
IEEE	**5 / 27**	**4 / 137**	**19 / 540**	-	-	-
Google	**25 / 300**	**6 / 200**	**13 / 100**	**8 / 300**	**5 / 200**	**5 / 100**
Google Scholar	**8 / 200**	-	-	**5 / 162**	-	-
SBES[1]	-	-	-	**0 / 321**	**1 / 321**	**3 / 321**
SBCARS	-	-	-	**1 / 92**	**0 / 92**	**1 / 92**
SBQS	-	-	-	**0 / 222**	**0 / 222**	**4 / 222**
Especialista	-	**2/2**	-	-	-	**6 / 6**

Com respeito aos resultados, ilustrados da Tabela 3.1, algumas considerações precisam ser feitas, sobre como as buscas foram executadas:

- Na ACM foi executada busca em modo avançado para o uso da *string* de busca inteira.
- Na IEEE, devido a falta de um mecanismo de busca apropriada que permitisse o uso da *string* de busca completa, as buscas nessa base de dados foram realizadas dividindo-se a *string* de busca original em strings com apenas dois termos, respeitando a lógica original. As *strings* resultantes dessa fragmentação são ilustradas na Tabela 3.2. Dado essa abordagem fragmentada de busca, o número dos trabalhos obtidos, que foi dado como a soma dos resultados das buscas efetuadas, não representa a quantidade de artigos únicos obtidos, pois haviam resultados repetidos entre buscas fragmentadas de uma mesma string original. Como por exemplo, uma busca com a strings “use case” AND “software product line” (string 1 na Tabela 3.2) apresentava alguns resultados repetidos com a busca “use cases” AND “software product line” (string 2 na Tabela 3.2).
- No Google, de forma análoga ao caso da IEEE, não foi possível usar a *string* completa no sistema de busca em modo avançado. Contudo, foi possível fragmentar a *string* em apenas duas, uma com “use case” e outra com “use cases”. Vale ressaltar que no caso do Google, as buscas retornaram centenas de milhares de trabalhos, e com isso apenas os 100 primeiros, ordenados pela relevância, foram considerados como resultantes da busca. Além disso, no caso das buscas referentes a modelagem de caso de uso para LPS, outras duas buscas foram realizados, *strings* de busca de número 7 e 18, e por isso os resultados obtidos ficaram em 300.
- No Google Scholar as buscas procederam de forma semelhante ao caso do Google.

- Nas conferências nacionais (SBES, SBCARS e SBQS) as buscas foram feitas manualmente nos anais das mesmas quando estes estavam disponíveis fisicamente para consulta ou disponíveis na internet (e.g., na Biblioteca Digital Brasileira de Computação [5]).

Tabela 3.2: Strings de buscas resultantes da fragmentação das strings originais.

String de busca original	Número	Strings de busca resultantes	Fonte
(“software product line” **OR** “software product lines”) **AND** (“use case” **OR** “use cases”)	1	(“software product line”) **AND** (“use case”)	IEEE
	2	(“software product line”) **AND** (“use cases”)	
	3	(“software product lines”) **AND** (“use cases”)	
	4	(“software product lines”) **AND** (“use case”)	
	5	(“software product line” **OR** “software product lines”) **AND** (“use case”)	Google e Google Scholar
	6	(“software product line” **OR** “software product lines”) **AND** (“use cases”)	
	7	use case software product line	Google
(“context aware” **OR** “context awareness”) **AND** (“use case” **OR** “use cases”)	8	(“context aware”) **AND** (“use case”)	IEEE
	9	(“context aware”) **AND** (“use cases”)	
	10	(“context awareness”) **AND** (“use case”)	
	11	(“context awareness”) **AND** (“use cases”)	
	12	(“context aware” **OR** “context awareness”) **AND** (“use case”)	Google
	13	(“context aware” **OR** “context awareness”) **AND** (“use cases”)	
“test” **AND** (“use case” **OR** “use cases”)	14	“test” **AND** (“use case”)	IEEE
	15	“test” **AND** (“use cases”)	
(“linha de produto de software” **OU** “linhas de produto de software”) **E**	16	(“linha de produto de software” **OU** “linhas de produto de software”) **E** (“caso de uso”)	Google e Google Scholar
	17	(“linha de produto de software” **OU** “linhas de produto de software”) **E** (“casos de uso”)	
	18	caso de uso linha de produto de software	Google
(“sensível ao contexto” **OU** “sensíveis ao	19	(“sensível ao contexto” **OU** “sensíveis ao contexto”) **E** (“caso de uso”)	Google
	20	(“sensível ao contexto” **OU** “sensíveis ao contexto”) **E** (“casos de uso”)	

Após coletar os trabalhos classificados, foram excluídos os registros repetidos que são oriundos de trabalhos que são classificados em mais de uma busca. Os números finais dos artigos analisados, não repetidos, são apresentados na Tabela 3.3. Esses números refletem a quantidade de artigos que foi lida para encontrar os trabalhos relacionados a esta pesquisa.

Tabela 3.3: Quantidade de artigos analisados.

Tópico	Artigos Classificados	Artigos Analisados (não repetidos)
Modelagem de caso de uso para LPS	72	62
Modelagem de caso de uso para aplicações sensíveis ao contexto	23	23
Teste a partir de casos de uso	71	44

Nas subseções a seguir são apresentados mais detalhes sobre os trabalhos relacionados identificados. Ressalta-se que foram classificados como trabalhos relacionados aqueles que atendiam a pelo menos um dos seguintes requisitos: i) Apresenta uma descrição textual de caso de uso com suporte a especificação de variabilidades de uma LPS; ii) Apresenta uma descrição textual de caso de uso com suporte a especificação das influencias do contexto na execução ou configuração de uma aplicação; ou iii) Apresenta um método de geração de testes com base em descrições textuais de caso de uso.

3.2 Especificação de casos de uso para LPS

Conforme a definição de LPS, apresentada no Capítulo 2, a principal característica de uma LPS é a presença de variabilidades. Com isso, propostas foram feitas para tratar dessa variabilidade nos vários artefatos de desenvolvimento de software, incluindo a especificação

[5] http://www.lbd.dcc.ufmg.br/bdbcomp/bdbcomp.jsp

de casos de uso. Nesta subseção são apresentados os trabalhos relacionados que apresentam suporte a variabilidade na descrição textual de caso de uso.

Segundo John e Muthing (2002), expressar a variabilidade nos casos de uso traz os seguintes benefícios:

- Ajuda as pessoas envolvidas no estabelecimento da variabilidade e na obtenção de um melhor conhecimento do domínio;
- A variabilidade explícita nos casos de uso suporta a instanciação e a derivação de modelos exatos na engenharia de aplicação; e
- Casos de uso variáveis são um bom meio para a comunicação das possibilidades de produto entre engenheiros de requisitos e engenheiros da linha de produto de software.

Ao todo foram encontrados nove trabalhos com propostas para especificação de variabilidades em descrições textuais de casos de uso: John e Muthing (2002), Bertolino et al. (2002), Eriksson et al. (2004), Gomma (2004), Gallina et al. (2006), Anthonysamy e Somé (2008), Choi et al. (2008), Bonifácio e Borba (2009) e Araújo (2010). Nas subseções a seguir esses trabalhos são descritos em mais detalhes.

3.2.1 Casos de uso Genéricos

Segundo John e Muthing (JOHN; MUTHIG, 2002), em se tratando de variabilidades em descrições textuais de caso de uso, qualquer trecho pode ser variante. Logo, os autores propõem o uso de *tags* <*variant*> e </*variant*> para identificar pontos de variação dentro da descrição textual do caso de uso e denominam um caso de uso com essa característica de genérico, por permitir a especificação de variabilidades.

A Figura 3.1 apresenta um caso de uso genérico relacionado a um sistema que mantém a velocidade de um sistema de direção automática (*cruise control system*) com ajuda de um regulador de gás.

No exemplo da Figura 3.1, a variação está na presença (variant ALT 1) ou não (variant ALT 2) de um regulador de distância. Ressalta-se que as questões sublinhadas refletem partes do modelo de decisão geral (usado para instanciar o caso de uso do produto a partir do caso de uso da LPS). Segundo John e Muthing (2002), tais questões também ajudam a entender a variabilidade do caso de uso a partir do ponto de vista do caso de uso.

3.2.2 PLUC - Product Line Use Case

Na proposta de Bertolino et al. (BERTOLINO et al., 2002; BERTOLINO; GNESI, 2003) é apresentado o PLUC (*Product Line Use Case*) que é uma extensão do caso de uso do Cockburn‘s (COCKBURN, 2001) para lidar com a variabilidade de uma linhas de produto de software.

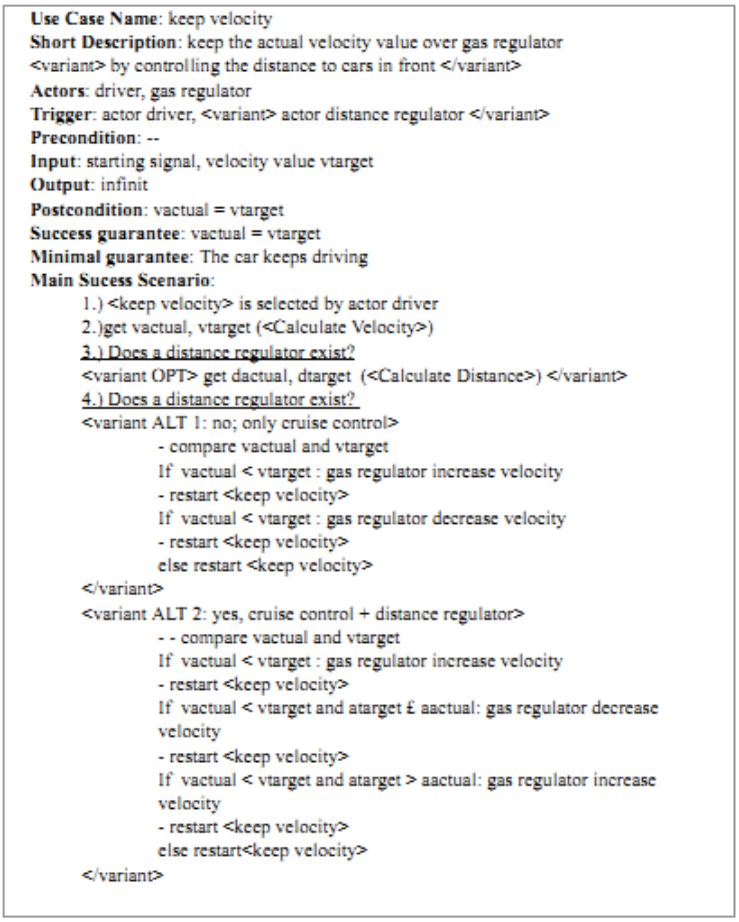
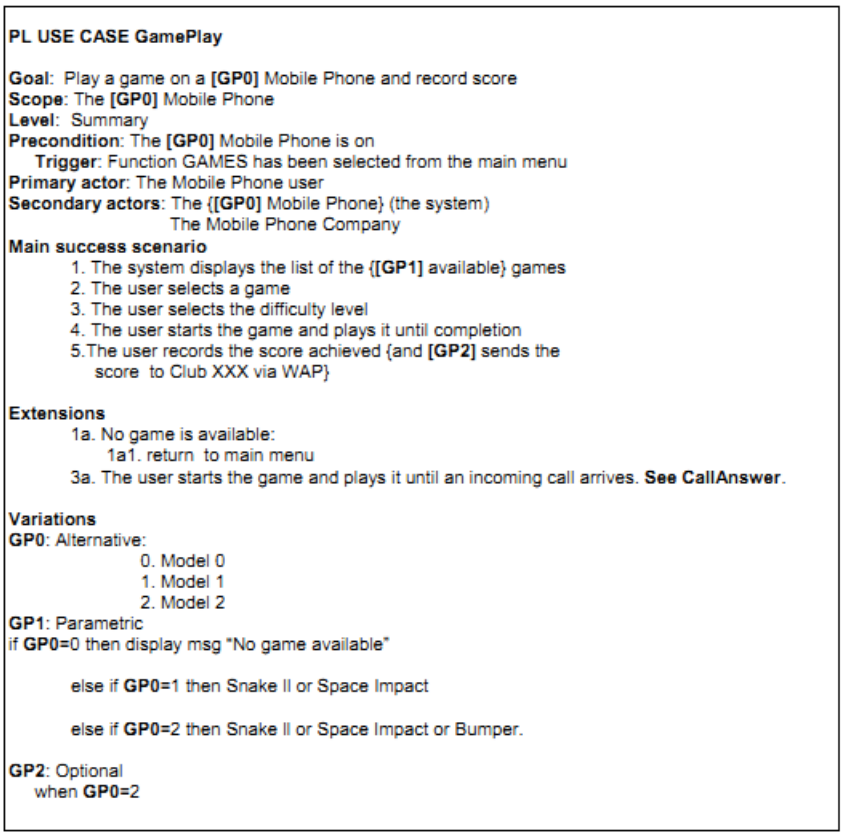


Figura 3.1: Exemplo de caso de uso genérico, fonte: (JOHN; MUTHIG, 2002).

Por meio de *tags*, o PLUC permite a descrição de três tipos de variações: a) Alternativa, que expressa a possibilidade de instanciar o requisito selecionando uma instância dentre um conjunto pré-definido de escolhas possíveis, cada uma dependendo da ocorrência de uma condição; b) Paramétrica, no qual a instanciação é conectada ao valor real do parâmetro no requisito para o produto específico; e c) Opcional, onde a instanciação pode ser feita selecionando indiferentemente dentre um conjunto de valores, que são *features* opcionais, para um produto derivado. Quando o PLUC é instanciado para o produto, tem-se o PUC (*Product Use Case*). A Figura 3.2 ilustra o exemplo de um caso de uso modelado neste template.

O exemplo da Figura 3.2 descreve o comportamento de um telefone pertencente a uma LPS quando um jogo é iniciado pelo usuário. As tags [GP0], [GP1] e [GP2] indicam os pontos de variação dentro do caso de uso. As variações, por sua vez, são definidas dentro da seção "Variações" dentro do PLUC.

Além disso, alguns cenários de um PLUC podem depender de cenários em outro PLUC, representando *features cross-cutting*. Logo, isso é representado com uma nota: "See PLUC name". No caso da Figura 8, tem-se a nota "See CallAnswer" dentro do PLUC GamePlay, o que significa que se uma chamada chega ao telefone enquanto o usuário está jogando, os passos que serão executados podem ser encontrados no PLUC CallAnswer.



PL USE CASE GamePlay

Goal: Play a game on a **[GP0]** Mobile Phone and record score
Scope: The **[GP0]** Mobile Phone
Level: Summary
Precondition: The **[GP0]** Mobile Phone is on
Trigger: Function GAMES has been selected from the main menu
Primary actor: The Mobile Phone user
Secondary actors: The {**[GP0]** Mobile Phone} (the system)
The Mobile Phone Company
Main success scenario
1. The system displays the list of the {**[GP1]** available} games
2. The user selects a game
3. The user selects the difficulty level
4. The user starts the game and plays it until completion
5. The user records the score achieved {and **[GP2]** sends the score to Club XXX via WAP}

Extensions
1a. No game is available:
1a1. return to main menu
3a. The user starts the game and plays it until an incoming call arrives. **See CallAnswer**.

Variations
GP0: Alternative:
0. Model 0
1. Model 1
2. Model 2
GP1: Parametric
if **GP0**=0 then display msg "No game available"

else if **GP0**=1 then Snake II or Space Impact

else if **GP0**=2 then Snake II or Space Impact or Bumper.

GP2: Optional
when **GP0**=2

Figura 3.2: Exemplo de caso de uso na notação PLUC, fonte: (BERTOLINO et al., 2002).

3.2.3 Uma extensão do Fluxo de Caixa Preta de Eventos

Eriksson et al. (ERIKSSON; BÖRSTLER; BORG, 2004) propõem a modelagem de variabilidades utilizando parâmetros por meio de uma versão estendida do Fluxo de Caixa Preta de Eventos, que trata-se de uma notação tabular para descrever cenários de casos de uso (RUP 2003).

A extensão do Fluxo de Caixa Preta de Eventos permitiu a especificação de comportamentos variantes utilizando-se uma notação compacta. Com essa notação, os autores abordam o problema de suporte fraco para rastreabilidade entre comportamento variante e a arquitetura do sistema. Isso porque os passos de caixa preta podem ser decompostos em passos de caixa branca, documentando como elementos arquiteturais colaboram para resolver cada passo caixa preta.

Além disso, o uso de parâmetros permitiu a diferenciação entre pontos de variação global e local. A abordagem utiliza o símbolo "$" para denotar um parâmetro local, cujo escopo é o caso de uso no qual ele reside, e o símbolo "@" para denotar um parâmetro global, cujo escopo é todo o modelo de domínio.

Para dar suporte a variabilidade na descrição textual de caso de uso, a proposta de Eriksson et al. (2004) é utilizar o identificador do passo do fluxo, que na notação original do RUP (2003) é utilizado apenas para numerar os passos, para especificar as variações como se segue (ERIKSSON; BÖRSTLER; BORG, 2004):

- O número identificador do passo, tal como “1”, identifica um passo mandatório no cenário, assim como acontece na notação original;
- Vários passos identificados com o mesmo número, tais como “4”, “4” e “4” , identifica um número de alternativas para um passo mandatório no cenário, no qual exatamente um deve ser selecionado;
- Vários passos identificados com o mesmo número e uma letra consecutiva, tais como “4a”, “4b” e “4c”, identifica um número de alternativas para um passo mandatório no cenário, no qual pelo menos um deve ser selecionado;
- Um identificador do número do passo dentro de parênteses, tais como “(2)”, identifica um passo opcional no cenário;
- Vários passos identificados com o mesmo número dentro de parênteses, tais como “(3)”, “(3)” e “(3)” identifica um número de alternativas para um passo opcional no cenário, no qual exatamente um deve ser selecionado; e
- Vários passos identificados com o mesmo número dentro de parênteses e uma letra consecutiva, tais como “(3)a”, “(3)b” e “(3)”, identifica um número de alternativas para um passo opcional no cenário, no qual pelo menos um pode ser selecionado.

A Figura 3.3 apresenta um exemplo de descrição de cenário de caso de uso usando a notação de Eriksson et al. (2004). O caso de uso da figura é de uma *Automatic Teller Machine* que utiliza um sistema de identificação pessoal. Os passos “1”, “4” e “5” descrevem um comportamento obrigatório, o passo “(2)” descreve um comportamento opcional, e os passos consecutivos identificados com um “(3)” descrevem um comportamento opcional, do qual exatamente um pode ser selecionado. O cenário de caso de uso do exemplo também usa 2 parâmetros, o parâmetro global @PIN_SIZE e o parâmetro local $MAX_CHIP_CARD.

3.2.4 Descrição de casos de uso segundo o PLUS

Gomaa (GOMAA, 2004) apresenta um método de projeto de LPS basedo em UML denominado PLUS (Product Line Uml-based Software Engineering). Nele, Gomma propôs três tipos de casos de uso para linhas de produto: (i) obrigatórios (*kernel*), que estão presentes em todos os membros da linha de produto; (ii) opcionais, que estão presentes em apenas alguns produtos; e (iii) alternativos, quando diferentes versões do caso de uso são requeridos por diferentes membros da linha de produto.

Além disso, Gomma (2004) propôs esteriótipos (*«kernel», «optional» e «alternative»*) para distinguir tais elementos na notação UML de casos de uso e apresentou um template para auxiliar a elaboração das descrições textuais dos casos de uso. Segundo Gomma (2004), para pequenas variações, os pontos de variação são descritos no próprio caso de uso identificando o lugar no caso de uso onde a mudança pode ocorrer. Quando o caso de uso passa a ficar muito complexo, as variabilidades podem ser modeladas como relacionamentos de inclusão e exclusão entre casos de uso (GOMAA, 2004).

Step	Actor Action	Black Box System Action	Black Box Budget Requirements
1	The use case begins when the **Customer** inserts the chip card into the ATM.	The **System** request that the customer should be identified.	Total response time is `$MAX_CHIP_CARD` *s*
(2)	The **Customer** presents its `@PIN_SIZE` character long PIN.	The System verifies the PIN.	Total response time is 0.1*s*
(3)	The **Customer** presents the fingerprint to the ATM.	The **System** verifies the fingerprint.	Total response time is 0.1*s*
(3)	The **Customer** presents the voice sample to the ATM.	The **System** verifies the voice sample.	Total response time is 0.1*s*
4	The **Customer** selects the amount of money to be withdrawn.	The **System** deducts the amount from the customer account and dispenses the cash to the Customer.	
5	The **Customer** takes the cash from the money dispenser.	The **System** is put into idle state.	

Figura 3.3: Exemplo de descrição de cenário de caso de uso na proposta de Eriksson et al (2004), fonte: (ERIKSSON; BÖRSTLER; BORG, 2004).

No template de Gomma (2004), para documentar pequenas variações as seguintes informações devem ser fornecidas:

- Nome do ponto de variação;
- Tipo de funcionalidade a ser inserida no ponto de variação, a qual pode ser opcional, alternativa obrigatória (no qual uma das alternativas deve sempre ser selecionada) ou alternativa opcional (é possível não selecionar nenhuma das alternativas);
- Linha na descrição do caso de uso onde a variabilidade pode ser introduzida; e
- Descrição da funcionalidade inserida no ponto de variação.

A Figura 3.4 apresenta um exemplo de descrição de caso de uso na proposta do Gomma. O caso de uso trata da utilização de um micro-ondas para cozinhar comida e como variações ele apresenta uma opcional, “Lâmpada”, que pode ou não estar presente e outra obrigatória alternativa, “Linguagem”, na qual uma seleção deve ser feita.

3.2.5 UCET - *Use Case Elicitation Template*

Em um relatório técnico, Gallina et al. (GALLINA et al., 2006) apresentam templates para elicitação de requisitos de LPS: UCET (*Use Case Elicitation Template*), REQET (*Requirements Elicitation Template*) e DOMET (*Domain Elicitation Template*). Segundo os

Nome do Caso de uso: Cozinhar Comida
Categoria de Reuso: *Kernel*
Resumo: O usuário põe a comida no forno e o micro-ondas cozinha a comida
Atores: Usuário (primário), Temporizador (secundário)
Pré-condição: Forno de micro-ondas 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. Temporizador decorre e notifica o sistema
8. Sistema para 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

Alternativas:
Linha 1: Usuário pressiona Iniciar quando a porta está aberta. Sistema não inicia o cozimento
Linha 4: Usuário pressiona Iniciar quando a porta está fechada e o tempo de cozimento é igual a zero. Sistema não inicia o cozimento
Pós-condição: O forno micro-ondas cozinhou a comida

Nome: Lâmpada
Tipo de funcionalidade: Opcional
Número de linhas: 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 para.

Nome: Linguagem
Tipo de funcionalidade: Obrigatória Alternativa
Número de linhas: 3,8
Descrição de funcionalidades: Existe uma escolha de linguagem para exibir as mensagens. O padrão é Inglês. Linguagens alternativas exclusivas mutualmente são: Francês, Espanhol, Alemão e Italiano.

Figura 3.4: Exemplo de descrição de cenário de caso de uso na proposta de Gomma (2004), adaptado de (GOMAA, 2004).

autores, o template de caso de uso proposto foi inspirado nas propostas de Cockburn‘s (COCKBURN, 2001), Bertolino (BERTOLINO; GNESI, 2003) e Gomma (GOMAA, 2004), e um dos diferenciais dessa proposta foi dar atenção aos requisitos não funcionais.

Os campos presentes no UCET são descritos a seguir:

- ID: identificador na forma “UCXX”, onde X é um digito;
- Nome do caso de uso: cada caso de uso tem um nome que deve representar o objetivo do mesmo;
- Categoria de seleção: especifica o quanto o caso de uso é obrigatório, opcional ou alternativo, especificando os casos de uso alternativos entre colchetes;
- Descrição: descreve o caso de uso com uma ou duas sentenças;
- Atores: especifica o nome de todos os atores que atuam no caso de uso;

- Dependência: descreve o quanto o caso de uso depende de outro caso de uso, ou seja, o quanto ele inclui ou estende outro caso de uso;
- Pré-condições: estabelece uma ou mais condições que devem ser verdadeiras no início do caso de uso;
- Pós-condições: identifica a condição que sempre é verdadeira no final do caso de uso se a sequência principal for seguida;
- Cenário Principal: descrição textual na forma de entrada dos autores, seguida de respostas do sistema. No caso, o sistema é tratado como uma caixa preta e detalhes internos não são descritos;
- Alternativas do cenário principal: essa seção provê a descrição de caminhos alternativos a sequência principal;
- Não funcionais: especifica propriedades não funcionais (e.g., segurança, eficiência) relacionados ao caso de uso;
- Descrição dos pontos de variação: descreve os pontos de variação presentes no caso de uso. As variantes tem um tipo (Obrigatório, Alternativo ou Opcional) e um alvo, que pode ser os dados utilizados no caso de uso ou comportamento.

Para um melhor entendimento do uso do template de Gallina et al. (2006), considere a aplicação baseada em "Hello World" (GALLINA et al., 2006). Dependendo da localização do consumidor, a aplicação deve exibir uma mensagem apropriada (que significa "Hello" em inglês): a) Bélgica : "Bondjoû"; b) França: "Bonjour/SalutHello"; e c) Luxemburgo: "Moien". Além disso, a mensagem deve ser exibida diretamente na tela do computador ou no *display* de um celular.

Considerando tais características, o caso de uso modelado na abordagem de Gallina et al. (2006) é apresentado na Figura 3.5.

Conforme observado na Figura 3.5, na descrição das variabilidades, é feita referência a variáveis, tais como *WalMessage*, *FrMessage*, e *LuMessage*. Tais variáveis representam conceitos do domínio em questão e são descritas por um documento no template do DOMET, conforme ilustrado na Figura 3.6. Nesse caso, os conceitos do domínio são descritos por meio de um nome, do tipo de variação, de uma descrição textual do conceito e da descrição das dependências. Dessa forma, o template de caso de uso está relacionado ao template de descrição do domínio.

3.2.6 A proposta de modelagem orientada a aspectos de Anthonysamy e Somé

Anthonysamy e Somé (ANTHONYSAMY; SOMÉ, 2008) apresentam uma abordagem orientada a aspectos para modelagem de casos de uso de linhas de produto de software. Na abordagem orientada a aspectos para aplicações tradicionais (SOMé; ANTHONYSAMY,

Use case ID	UC01
Use case name.	Display “Hello” Message
Slection category.	Mand.
Description.	The application user is displayed a message meaning “hello” in a given language
Actors.	Primary actor: User
Dependency.	None
Preconditions.	HelloWord application is launched.
Postconditions.	A message has been displayed to the user.
Main scenario	The application displays [V1](LuMessage) on [V2](CompDisplay)
Alternatives of the main scenario	None.
Non-Functional.	Once the application launched the display of the message should take less than one second.
Variation points description	V1: Type:Alt Concerns: data. One of the following: WalMessage, FrMessage, LuMessage V2: Type: Alt Concerns: data. One of the following CompDisplay, MobileDisplay

Figura 3.5: Exemplo de descrição de cenário de caso de uso na proposta de Gallina et al., fonte: (GALLINA et al., 2006).

2008), esses autores definem interesses transversais como “advices use cases” que são conectados a um conjunto de casos de uso base por meio de um relacionamento “aspect”, que apresenta uma cardinalidade um-para-muitos. Segundo esses autores, o relacionamento um-para-muitos é adotado porque em geral interesses transversais tipicamente influenciam vários casos de uso.

Advices use cases são definidos da mesma forma que casos de uso, mas podem conter somente algumas seções do caso de uso, ser iniciados pelo sistema e descrever interações incompletas. A conexão entre os *advices use cases* e o caso de uso afetado é feita por meio da correspondência sintática dos *joinpoints*, que revelam ocorrências potenciais de interesses

Name	**Var Type**	**Description**	**Dependencies**
`Message`	Mand	Abstract Notion of Message	
`WalMessage`	Alt	Represents the Walloon message that will be displayed to the application user	Generalization: `Message`
`FrMessage`	Alt	Represents the French message that will be displayed to the application user	Generalization: `Message`
`LuMessage`	Alt	Represents the Luxembourgish message that will be displayed to the application user	Generalization: `Message`
`Display`	Mand	Abstract Notion of Display	
`CompDisplay`	Alt	Represents a computer screen	Generalization: `Display`
`MobileDisplay`	Alt	Represents a mobile phone’s screen	Generalization: `Display`

Figura 3.6: Exemplo de descrição de conceitos do domínio na proposta de Gallina et al., fonte: (GALLINA et al., 2006).

transversais no caso de uso base, e *pointcut expressions*, que são expressões parametrizadas baseadas em padrão que correspondem aos *joinpoints*. Um *pointcut*, por exemplo, definido como "passo 1,2" refere-se aos passos um e dois do caso de uso e "passo *" refere-se a todos os passos do caso de uso.

Com relação aos tipos de *advice weaving*, os autores Anthonysamy e Somé consideram quatro tipos de composição:

- *before*: requisitos transversais são aplicados antes do *joinpoint*;
- *after*: requisitos transversais são aplicados depois do *joinpoint*;
- *around*: requisitos transversais são aplicados no lugar do *joinpoint*; e
- *concurrent*: requisitos transversais são aplicados concorrentemente com um *joinpoint*;

No caso de uma LPS, a abordagem orientada a aspectos para modelagem de caso de uso (ANTHONYSAMY; SOMÉ, 2008) é estendida para tratar de variabilidades. Nessa última, é definida a relação <<variabilidade>> que é uma especialização da relação <aspecto>. O relacionamento "variabilidade" funciona de forma análoga ao relacionamento aspecto, contudo, ele apresenta uma condição que determina o quanto o membro da LPS provê a funcionalidade descrita no *advice use case*.

Para ilustrar a abordagem, os autores apresentam como exemplo a modelagem de caso de uso de uma linha de produto de microondas que apresenta *features* opcionais e alternativas. Mais detalhes sobre essa linha estão encontrados em (GOMAA, 2004). A Figura 3.7 apresenta o caso de uso Cozinhar Comida. Observa-se que tal caso de uso captura as funcionalidades comuns dos produtos da linha de microondas.

Como variabilidades alternativas dessa linha tem-se a unidade de *display* que é *one-line* ou *multi-line* e como variabilidades opcionais tem-se o *beeper* que indica quando o cozimento acabou. A Figura 3.8 apresenta a representação das variabilidades e similaridade da linha em questão na ferramenta desenvolvida pelos autores, a UCEd. Já a Figura 3.9 apresenta a descrição textual dos *advices use cases* relacionados as variabilidades unidade de *display* e *beeper*.

3.2.7 Casos de uso e o modelo de variabilidades ortogonal

Segundo Choi et al. (2008), as abordagens propostas para modelar variabilidades nos casos de uso falham porque: (i) não oferecem suporte completo para todos os tipos de variabilidade; (ii) o uso de *tags* para colocar informações de variabilidade no caso de uso, torna-o muito complexo diminuindo a legibilidade; e (iii) *tags* em artefatos particulares, como cenários de caso de uso, espalham as informações de variabilidade, reduzindo a rastreabilidade.

Logo, procurando solucionar esses problemas, os autores propõem um sistema de *tags* mais simples que a proposta de Bertolino et al. (2002) apresentada na SubSeção 3.1.2, e fazem um mapeamento dessas *tags* para o modelo OVM (*Orthogonal Variability Model*) (POHL;

Title: Cook Food
Primary Actor: User
Goal: Allows User to cook, heat food
Precondition: Microwave oven is idle
Post-condition: Microwave oven has cooked the food
Steps:
1.The User opens the oven door
2.The User puts the food in the oven
3.The User closes the oven door
4.The User presses the Cooking Time button
5.The Microwave System prompts the User for the cooking time
6.The User enters a cooking time on the numeric keypad and
presses Start button
7.The Microwave System starts cooking the food
8.The Microwave System continually cooks and displays the
cooking time remaining
9.The Timer elapses and notifies the Microwave System
10.The Microwave System stops cooking the food and
displays the end message
11.The User opens the oven door
12.The User removes the food from the oven
13.The User closes the door
14.The Microwave System clears the display
Alternatives:
6.a.The Cooking time is zero
6.a.1.The Microwave System notifies the User of an error

Figura 3.7: Exemplo de caso de uso base na proposta de Anthonysamy e Somé, fonte: (ANTHONYSAMY; SOMÉ, 2008).

G.; LINDEN, 2005). Dessa forma, a proposta de Choi et al. (2008) mantem as informações de variabilidade consistentes durante todo o desenvolvimento. Além disso, baseado no OVM e nos casos de uso com *tags*, os autores apresentam um algoritmo para geração de cenários de caso de uso dos produtos.

A diferença entre a abordagem de Choi et al. (CHOI et al., 2008) e de Bertolino et al (BERTOLINO; GNESI, 2003) está no fato da abordagem da primeira utilizar as *tags* somente para indicar alguma informação de variabilidade e mapear esta informação para algum Ponto de Variação ou Variante no OVM. Com isso, a abordagem de Choi et al. não utiliza os três tipos

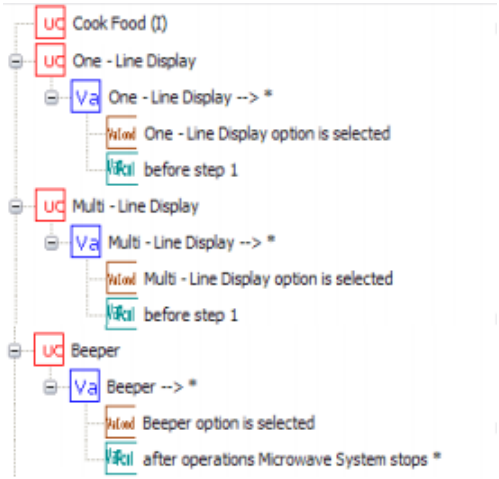


Figura 3.8: Representação das Variabilidades na UCEd, fonte: (ANTHONYSAMY; SOMÉ, 2008).

Title: One-Line Display Goal: To display messages and cooking time Steps: BI.The Microwave System enables one-line display panel
Title: Multi-Line Display Goal: To display messages and cooking time Steps: CI.The Microwave System enables multi-line display panel
Title: Beeper Goal: To indicate user when cooking stops Steps: EI.The Microwave System activates the beeper when cooking stops

Figura 3.9: Descrição dos *advices use cases* na proposta de Anthonysamy e Somé, fonte: (ANTHONYSAMY; SOMÉ, 2008).

de *tags* propostos na abordagem do Bertolino et al. (2003).

A Figura 3.10 apresenta um exemplo de modelagem de caso de uso na proposta de Choi et al.(2008). No caso do exemplo, trata-se do caso de uso Cozinhar Comida da linha de produtos de microondas descrita em (GOMAA, 2004).

Use case name: Cook Food.
...
Description:
1. User opens the door, puts food in the oven, and closes the door. Cooking is prohibited if [WST] detects no items in the oven.
2. User selects power level within [PLT]. {User presses the Cooking Time button following [MP] procedure.}
3. System prompts for cooking time.
4. User enters the cooking time on the numeric keypad and presses Start.
...
Variations:
[WST] WST1: Analog Sensor
WST2: Digital Sensor
[PLT] PLT1: One-level
PLT2: Multi-level
[MP] If selected, then it results in one minute being added to the cooking time. If the cooking time was previously zero, cooking is started.
[HET] HET1: One-level
HET2: Multi-level

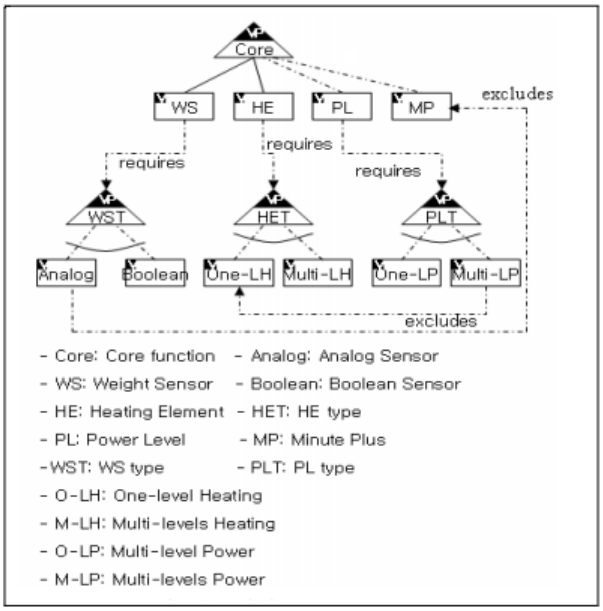


Figura 3.10: Do lado esquerdo o caso de uso Cozinhar Comida modelado na proposta de Choi et al, e do lado direito o modelo OVM associado, fonte: (CHOI et al., 2008).

3.2.8 Casos de uso na abordagem MSVCM

Bonifácio e Borba (2009) apresentam uma abordagem de modelagem de variabilidades em casos de uso denominada *Modeling Scenario Variability as Crosscutting Mechanisms* (MSVCM). Segundo os autores, essa abordagem melhora a separação de interesses entre gerenciamento da variabilidade e especificação dos cenários, definindo a variabilidade de cenários

como uma composição de diferentes artefatos: modelo de caso de uso da LPS, modelo de *feature*, modelo de configuração do produto e informação de conhecimento.

Com respeito ao modelo de caso de uso, este é composto por casos de uso e *aspectual use cases*. Um caso de uso tem um nome, uma descrição e uma lista de cenários que são compostos por uma sequência de passos (pares ação do Usuário x resposta do Sistema). Já um *aspectual use case* tem um nome e uma lista de *advices* que podem ser usados para estender o comportamento dos cenários existentes.

A Figura 3.11 apresenta um cenário obrigatório que especifica um comportamento comum que é requerido para confirmar uma compra. Note que o parâmetro SM referenciado no passo 2 permite o reuso do cenário "Procedimento de Compra" para diferentes configurações da *feature* associada a esse parâmetro, que no caso trata-se da *feature* "Shipping Method". Neste caso, o artefato conhecimento de configuração é responsável por relacionar parâmetros para as *features*. Além disso, a anotação "[CustomerPreferences]" é outro ponto de variação do cenário Procedimento de Compra.

Id: SC01
Description: Proceed to purchase

Id	User Action	System Response
P1	Fill in the requested information and select the proceed option.	Request the shipping method and address.
P2	Select one of the available shipping methods (<SM>), fill in the destination address and proceed.	Calculate the shipping costs.
P3	Confirm the purchase.	Execute the order and send a request to the Delivery System to dispatch the products. [CustomerPreferences]

Figura 3.11: Caso de uso "Procedimento de Compra", fonte: (BONIFáCIO; BORBA, 2009).

Na Figura 3.12, por sua vez, os *advices use cases* relacionados ao caso de uso "Procedimento de Compra" são descritos. Os *advices* ADV01 e ADV02 introduzem um comportamento antes do passo P1 definido no cenário "Procedimento de Compra". No caso, o que determina a escolha do comportamento a ser seguido é a presença das *features*. No exemplo da figura, se o produto tiver as *features* "*Shopping Cart*" e "*Bonus*" então o ADV02 vai ser executado, caso contrário o escolhido é o ADV01.

Por fim, o ADV03 atualiza as preferências do usuário baseado no histórico de buscas e compras do usuário. No caso, este comportamento é iniciado após qualquer passo com a anotação *[CustomerPreferences]*.

3.2.9 Modelagem de casos de uso baseada em fragmentos

A abordagem de Araújo (2010) é voltada para apoiar a especificação de casos de uso para uma aplicação específica de uma LPS baseado no conceito de fragmentos. Ela consiste

Id: ADV01
Description: Buy a specific product
Before: P1

Id	User Action	System Response
B1	Select the buy product option.	Present the selected product. The user can change the quantity of items he wants to buy. Calculate and show the amount to be paid.
B2	Select the confirm option.	Request payment information.

Id: ADV02
Description: Buy products using a shopping-cart
Before: P1

Id	User Action	System Response
C1	Select the checkout option.	Present the items in the shopping cart and the amount to be paid. The user can remove items from the shopping cart.
C2	Select the confirm option.	Request bonus and payment information.

Id: ADV03
Description: Register user preferences.
After: [CustomerPreferences]

Id	User Action	System Response
R1	-	Update the preferences based on the search results or purchased items.

Figura 3.12: *Advices use cases* relacionados ao caso de uso “Procedimento de Compra”, fonte: (BONIFáCIO; BORBA, 2009).

em duas atividades principais: engenharia do domínio e da aplicação. Na engenharia de domínio, 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 engenharia da aplicação, descrições textuais de casos de uso para uma aplicação específica podem ser rapidamente produzidas através da composição e personalização desses fragmentos.

Originalmente, os fragmentos de casos de uso não são de nenhum domínio específico, mas na proposta de Araújo (2010), os fragmentos são voltados para o domínio da LPS. Ressalta-se que nessa abordagem, uma *feature* pode estar associada a um ou mais fragmentos. A Figura 3.13 apresenta um exemplo de fragmento de caso de uso. Esse fragmento está relacionado a *feature* obrigatória “Reserva de Carro”, pertencente a LPS de reserva de aluguel de carros (ARAúJO, 2010).

A Figura 3.14 apresenta o fragmento da Figura 3.13 personalizado para atender a aplicação específica em questão, onde as substituições são indicadas em negrito e as partes descartadas são com um risco. Ressalta-se que na abordagem de Araújo (2010) um caso de uso também pode ser feito juntando-se várias partes de vários fragmentos de casos de uso.

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 3.13: Fragmento de caso de uso, fonte: (ARAúJO, 2010).

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 3.14: Fragmento de caso de uso personalizado para aplicação específica, fonte: (ARAúJO, 2010).

3.3 Especificação de casos de uso para Aplicações Sensíveis ao contexto

Conforme mencionado no início desde capítulo, também foram considerados como trabalhos relacionados aqueles que apresentaram uma descrição textual de caso de uso com suporte a especificação das influências do contexto na execução ou configuração de uma aplicação. Isso porque, uma vez que não foi encontrado nenhuma proposta de template para descrição textual de caso de uso para LPSSC, onde o contexto também está presente, também faz parte

desta pesquisa propor um template próprio para LPSSC, com informações de variabilidade e de contexto da linha.

Após a análise dos artigos encontrados, conforme metodologia apresentada na Seção 3.1, identificou-se que 03 trabalhos (ANJOS, 2006; PATRICIO, 2010; SILVA, 2011) relatavam o uso de informações de contexto dentro da descrição textual de caso de uso e somente 01 trabalho (YANG, 2010) apresentou uma forma para modelar o impacto das informações de contexto dentro de uma descrição textual de caso de uso.

Anjos (ANJOS, 2006) apresenta um estudo de aplicações cientes do contexto e uma implementação de uma aplicação móvel ciente de contexto utilizando o *middleware* MoCA (*Mobile Collaboration Architecture*). Na modelagem de casos de uso dessa aplicação, Anjos descreve os casos de uso por meio de elementos tradicionais: um identificador e um nome, uma descrição, entradas e pré-condições, saídas e pós-condições e fluxo de eventos. A Figura 3.15 apresenta um dos casos de uso dessa aplicação.

A ferramenta desenvolvida por Anjos (2006) tem por objetivo auxiliar o dia-a-dia de funcionários em uma empresa por meio da utilização de informações de contexto dos usuários para prover informações úteis. Dentre as funcionalidades, o software fornece informativos ligados às dependências da empresa.

[CU02] Atualizar informações sobre uma região

Caso de uso responsável por disponibilizar as informações (mensagens) existentes na região onde o usuário se encontra.

Entradas e pré-condições: O usuário deve estar logado no sistema.

Saídas e pós-condições: Nenhuma.

Fluxo Principal de Eventos

1. O sistema monitora as mudanças de localização do usuário.
2. Sempre que o usuário entra numa nova região, o servidor é avisado desse evento através de uma função do LIS. O servidor então envia essa informação para o cliente que está rodando no dispositivo móvel do usuário. Além dessa informação de mudança de local, o servidor envia as mensagens associadas à região onde o usuário acabou de entrar.
3. Na tela principal da aplicação uma caixa de texto disponibiliza as mensagens da região atual para o usuário.

Figura 3.15: Caso de uso “Atualizar Informações sobre uma Região”, fonte: (ANJOS, 2006).

Patricio (PATRICIO, 2010) descreve o uso das variações de contexto numa seção chamada “Fluxo de Contexto” de forma análoga como são descritos os fluxos alternativos na seção “Fluxos Alternativos”. A Figura 3.16 apresenta um exemplo de um caso de uso apresentado no trabalho de Patricio (2010) que apresenta o contexto influenciando a indicação de livros ao usuário.

Já Silva (SILVA, 2011), por sua vez, descreve uma estrutura de software de suporte ao desenvolvimento e execução de sistemas de recomendação sensíveis ao contexto para TV Digital. A Figura 3.17 apresenta um caso de uso do software desenvolvido por Silva, onde

```
1 Descrição
  Esse caso de uso descreve o empréstimo de um livro da Biblioteca por
      um Aluno.
2 Atores
  2.1 Aluno
  2.2 Biblioteca
3 Pré-Condições
  O usuário possui cadastro na Biblioteca.
4 Fluxo Básico de Eventos
  1. O aluno se autentica
  2. O aluno informa o código do livro a ser emprestado
  3. O sistema exibe um comprovante com o código do livro e a data de
     devolução
5 Fluxos Alternativos
  5.1 Usuário Inválido
    No passo 1 se o aluno não realizar autenticação, então
      1. O caso de uso Cadastrar Aluno é acionado
  5.2 Aluno possui multa de mora
    No passo 2 se o aluno tem multa de mora, então
      1. O caso de uso Calcular Multa é acionado
6 Fluxos de Contexto
  4.2 Indicação de outros Livros
    O sistema sugere outras obras que contribuam com o estudo do aluno
        baseando-se no seu perfil e em meta-informações do Livro
7 Pós-Condições
  Sucesso
    O aluno recebe o livro emprestado
```

Figura 3.16: Caso de uso “Emprestar livro”, fonte: (PATRICIO, 2010).

é possível observar que é feita a captura de informações de contexto para posteriormente tais informações serem utilizadas no sistema de recomendação desenvolvido.

Caso de Uso: 03	**Obter Informações Contextuais**
Ator	Usuário
Descrição	Diante da solicitação de recomendação ou realimentação de relevância é acionada a captura de forma implícita das informações contextuais do usuário tais como identificação do usuário, sua localização, dia e horário da interação e o tipo de dispositivo de acesso.
Evento Iniciador	Solicitação de recomendação personalizada de programas de TV ou realimentação de relevância realizada pelo usuário.
Pré-condição	Sistema de recomendação devidamente configurado em execução e programação de TV disponível.
Pós-condição	Armazenamento das informações de contexto, o que deverá permitir o acesso posterior a tais informações.
Extensões	Não há extensões
Inclusões	Não há extensões

Figura 3.17: Caso de uso “Obter Informações Contextuais”, fonte: (SILVA, 2011).

Finalmente, Yang (YANG, 2010) descreve as variações de contexto como fluxos alternativos, só que com restrições adicionadas aos fluxos, onde essas restrições são estados de contexto. Logo, aqui o fluxo alternativo só executa se a restrição de contexto for atendida. A Figura 3.18 apresenta um exemplo de caso de uso, apresentando no trabalho de Yang, para efetuar login em um sistema bancário. No exemplo dessa figura, as informações de contexto que impactam o caso de uso são o meio utilizado (*Channel*) e o tipo de usuário (*UserIdentify*).

Use Case Element	Description
Use Case Name	Login
Use Case Description	Get an authorization to do some transaction in the system
Primary Actor	Customer, Teller and Manager
Precondition	N/A
Basic Flow (**Context**: Channel = ATM, UserIdentify = Customer)	1. Insert the Debit/credit card; [Instead to Alt A, Instead to Alt B, Instead to Alt C] 2. Input PIN; 3. System welcomes; [Instead to Alt D]
Alternate Flow A (**Context:** Channel = ATM, UserIdentify = Employee)	1. Input the employee number; 2. Continue at step 2 of Basic Flow;
Alternate Flow B (**Context:** Channel = Branch, UserIdentify = Customer)	1.Show Passport and the debit/credit card or deposit book; 2.Continue at step 2 of Basic Flow;
Alternate Flow C (**Context:** Channel= Internet/Cell phone, UserIdentify = Customer)	1.Input the account number; 2.Continue at step 2 of Basic Flow;
Alternate Flow D (PIN is wrong)	2.System shows a message "PIN was wrong, please input again!" 3.Go to step 2 of Basic Flow or the customer gives up;

Figura 3.18: Descrição extendida do caso de uso "Login", fonte: (YANG, 2010).

3.4 Geração de Testes a partir de casos de uso

Quanto a geração de testes a partir de casos de uso, vários trabalhos apresentavam algo relacionado a geração de testes a partir de descrições textuais de caso de uso para aplicações únicas. A seguir, são apresentados sete destes trabalhos (GUTIERREZ et al., 2008; ROUBTSOV; HECK, 2006; BARNETT et al., 2003; BERTOLINI; MOTA, 2010; CHATTERJEE; JOHARI, 2010; SOMé; ANTHONYSAMY, 2008; GUTIÉRREZ et al., 2006).

Gutiérrez et al. (GUTIERREZ et al., 2008) não apresentam nenhuma técnica nova de geração de testes a partir de casos de uso, mas descrevem um estudo de caso sobre geração de casos de testes a partir de casos de uso. No estudo de caso, a partir de descrições textuais de caso de uso e por meio da técnica de análise de cenários, são gerados casos de testes, que são implementados e executados para verificar a sua efetividade.

Roubtsov e Heck (ROUBTSOV; HECK, 2006) descrevem o teste de aceitação de um sistema de larga escala por meio de uma abordagem de geração de testes a partir de casos de uso. Os autores enfatizam que, segundo a experiência deles, não é possível um mapeamento direto entre os cenários de caso de uso e os casos de testes, porque os cenários de casos de uso de grandes sistemas são escritos em alto nível de detalhes. Para superar esse problema, eles criam artefatos de teste em três níveis (cenários de teste, *scripts* de teste e casos de teste). A partir dos casos de uso eles extraem os cenários de testes que descrevem uma sequência de eventos. No segundo nível, *scripts* de testes são especificados contendo um procedimento de como executar os passos de testes dos cenários de testes. Por fim, são gerados casos de testes como instâncias dos *scripts* de testes.

Barnett el at. (BARNETT et al., 2003) apresentam o ambiente Asml tool que gera máquina de estados finita a partir de modelos de casos de uso que podem ser usados para

validação ou teste. Para isso, o caso de uso é modelado em AsmL (*Abstract State Machine Language*), uma linguagem de modelagem executável que é integrada com o *framework* .NET (.NET, 2012) e com as ferramentas de desenvolvimento da Microsoft.

Bertolini e Mota (BERTOLINI; MOTA, 2010) apresentam um *framework* para geração de caso de teste de interface gráfica baseado em projeto de caso de uso de GUI. O primeiro passo é a descrição do caso de uso em uma linguagem natural controlada. O segundo passo é o uso da ferramenta TaRGeT para seleção do algoritmo de geração de testes apropriado e uso desde algoritmo para gerar os casos de testes. Em seguida, casos de testes são gerados em uma linguagem natural controlada e são traduzidos em *scripts* de testes baseados em uma linguagem de *script* parametrizada. Por fim, o último passo finaliza os *scritps* de testes gerados, incorporando bibliotecas necessárias e código para iniciar e terminar os *scripts* de testes.

Chatterjee e Johari (CHATTERJEE; JOHARI, 2010) apresentam uma ferramenta, denominada STATEST 1.1.0, que gera suites de testes automatizadas a partir de requisitos descritos informalmente por meio de casos de uso. Para isso, os casos de uso são decompostos em descrições de cenários relevantes, onde cada descrição é transformada em diagrama de transição de estados que é utilizado para geração dos testes.

Somé e Anthonysamy (SOMé; ANTHONYSAMY, 2008) apresentam uma abordagem para geração de casos de testes de sistemas a partir de descrições textuais de casos de uso. Para isso, os casos de uso são descritos em uma linguagem natural controlada, e com eles são geradas máquinas de estado baseada em fluxos de controle. Em seguida, essas máquinas de estado são conectadas em uma máquina de estados no nível do sistema, e os testes baseados nessa máquina e com base em critérios de teste definidos.

Por fim, Gutierrez (GUTIÉRREZ et al., 2006) apresenta um processo de geração de casos de testes a partir de casos de uso para aplicações web. A partir dos casos de uso descritos em linguagem natural, eles geram um diagrama de atividades representando um modelo comportamental do caso de uso. Os valores dos testes são gerados usando o método de partição de categorias. Além disso, eles utilizam diagramas de sequência para refinar os casos de testes e implementar os *scripts* de testes.

Com respeito a geração de testes a partir de casos de uso para LPS, apenas cinco trabalhos foram identificados (KAMSTIES et al., 2004; ALI; MOAWAD, 2010; ZORZO et al., 2011; BERTOLINO; GNESI, 2003; NETO, 2011), os quais são descritos a seguir. Contudo, ressalta-se que não foi encontrado nenhum trabalho desse tipo voltado para linhas de produto de software sensíveis ao contexto.

Kamsties et al. (KAMSTIES et al., 2004) afirmaram que a derivação dos casos de testes do sistema para a família de produtos é difícil, devido a variabilidade nos requisitos, pois cada ponto de variação multiplica o número de possíveis comportamentos para serem testados. O trabalho propõe então novas estratégias para desenvolver casos de testes do domínio a partir de casos de uso com variabilidade e derivar casos de testes da aplicação a partir dos mesmos. O diferencial desta abordagem é que a variabilidade é introduzida nos casos de testes do domínio e utiliza diagramas de sequências para representar os casos de testes. Uma das estratégias propostas é a de segmentação que define que um ponto de variação é refletido em um caso de

teste por um conjunto de segmentos de casos de testes, onde cada segmento reflete uma variante do ponto de variação.

Segundo Mohamed e Moawad (ALI; MOAWAD, 2010), gerenciar os pontos de variação é um dos maiores desafios quando se trata de teste de LPS, bem como o fato de haver dois processos de desenvolvimento (Engenharia de Domínio e Engenharia de Aplicação). Os autores então investigam diferentes abordagens que iniciam o processo de teste de LPS a partir de requisitos, analisam duas técnicas e propõem uma técnica que usa o PLUC (BERTOLINO et al., 2002) e diagramas de atividades para permitir a geração de testes a partir de casos de uso.

Em (ZORZO et al., 2011), os autores propõem uma abordagem incremental, onde ocorre o teste individual do primeiro produto da LPS e, para os demais produtos que são gerados para a LPS, ocorre a aplicação de técnicas de teste de regressão, nas quais é feito o reaproveitamento de casos de teste dos produtos anteriormente testados. Nesse trabalho, o método de geração de testes seguiu três etapas: geração dos cenários dos casos de uso, identificação dos elementos necessários para executar os cenários, e definição de valores para os referidos elementos.

Bertolino et al. (BERTOLINO; GNESI, 2003) apresenta o PLUTO (*Product Lines Use case Test Optimization*), que é uma metodologia para derivação de testes a partir de requisitos da LPS descritos como PLUCs (BERTOLINO et al., 2002). Essa metodologia utiliza uma extensão do método de Partição de Categoria para lidar com as variabilidade da linha.

Finalmente, a ferramenta SPLMT-TE (NETO, 2011) gera cenários de testes combinando os cenários dos casos de uso. Ela gera casos de testes para o caminho principal, alternativo e excepcionais. Com essa ferramenta, os casos de testes gerados tem os seguintes elementos: identificador, nome, resumo, passos, pré-condição e pós-condição, tipo de variabilidade, referencia do caso de uso e caminho do sistema que deve ser testado.

3.5 Conclusão

Neste capítulo foram apresentados os trabalhos relacionados a esta pesquisa. Para identificar tais trabalhos foi executada uma busca nas principais fontes de pesquisa por trabalhos que tratassem da modelagem de variabilidade em descrições textuais de casos de uso ou da modelagem da influência do contexto no comportamento/configuração do software ou da geração de testes a partir da descrição textual de casos de uso.

Ao todo, como resultado das buscas e execução do primeiro filtro, lendo-se o resumo e título de cada trabalho, foram selecionados 129 trabalhos não repetidos. Destes foi possível identificar nove templates para a descrição textual de caso de uso para LPS, quatro formas de descrever a influência do contexto na aplicação dentro dos casos de uso, sete trabalhos relacionados a geração de testes a partir de casos de uso para aplicações únicas e quatro no cenário de LPS.

Com relação a descrições textuais de casos de uso para LPS, na literatura pesquisadores tem proposto adaptações dos templates de casos de uso existentes para lidar com va-

riabilidades. Nesse cenário, alguns trabalhos defendem a modelagem das variabilidades e similaridades de forma conjugada, enquanto outros trabalhos alegam que o melhor é separar o comportamento comum do variável.

Em se tratando das descrições textuais para aplicações sensíveis ao contexto, somente um trabalho foi encontrado modelando o impacto das mudanças de contexto no comportamento da aplicação. Nesse caso, restrições de contexto foram associadas aos fluxos alternativos do caso de uso, e com isso o que determinava se o fluxo alternativo executaria ou não era a satisfação da restrição de contexto.

No caso dos trabalhos voltados a geração de testes, foram identificados vários trabalhos direcionados a aplicações únicas (que não seguiam uma abordagem de LPS) e apenas quatro eram voltados para LPS. Em geral, esses trabalhos adotavam uma das três opções: i) analisar os cenários do caso de uso para gerar os cenários de testes; ii) gerar algum modelo (diagrama de atividades, estados ou sequência) a partir do caso de uso para só então gerar os testes; e iii) criar uma máquina de estados finita a partir dos casos de uso para guiar a geração dos testes.

Por tudo que foi apresentado neste capítulo, é possível observar que embora tenham sido encontrados vários trabalhos relacionados, nenhum deles tratava de uma LPSSC, que é uma linha de produto de software destinada ao desenvolvimento de aplicações sensíveis ao contexto. Vale ressaltar que, relacionado a geração de testes a partir de casos de uso para uma LPS, apenas quatro trabalhos foram identificados. Dessa forma, o Capítulo 4 apresenta a proposta de geração de testes a partir de casos de uso para uma LPSSC, procurando apoiar a identificação precoce dos defeitos.

4 PROPOSTA

Neste capítulo é descrita a proposta desta dissertação, que é um ambiente de geração de cenários de testes, constituído por um template para descrição textual de caso de uso, um método de geração de testes e uma ferramenta de apoio.

O restante deste capítulo está então organizado da seguinte forma: na Seção 4.2 é descrita a metodologia adotada para o desenvolvimento desta pesquisa; na Seção 4.2, a proposta de template para especificação textual de casos de uso para Linhas de Produto de Software Sensíveis ao Contexto é apresentada; na Seção 4.3, o método de geração de testes é descrito em detalhes; na Seção 4.4, a ferramenta de apoio é detalhada; e por fim, na Seção 4.5 são dadas as considerações finais sobre as contribuições deste trabalho.

4.1 Metodologia para o desenvolvimento da pesquisa

Para a proposta do ambiente de geração de cenários de testes para LPSSC, este trabalho seguiu a metodologia apresentada na Figura 4.1 por meio de um diagrama de atividades.

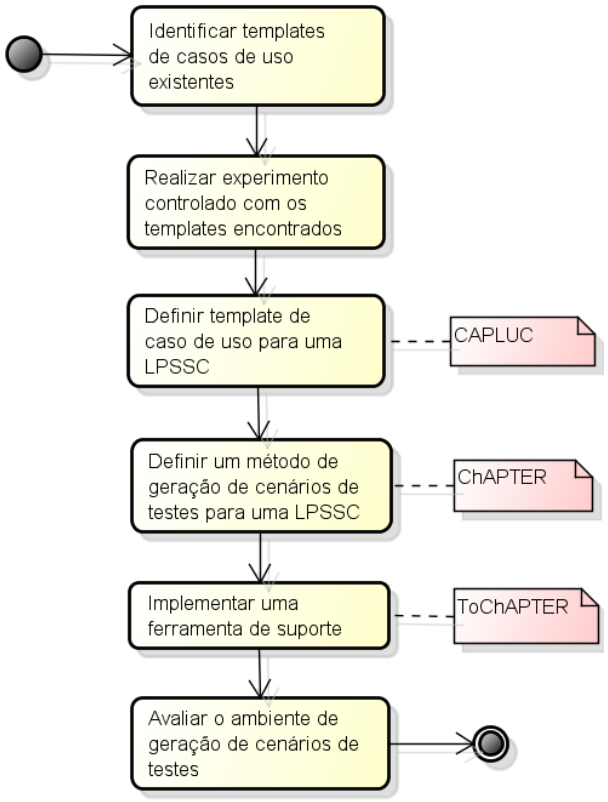


Figura 4.1: Metodologia para o desenvolvimento desta pesquisa.

Após uma extensa pesquisa bibliográfica nos temas envolvidos nesta dissertação e a conclusão de que não existia um template de caso de uso para LPSSC, o primeiro passo da metodologia seguida consiste em identificar propostas de templates de caso de uso para uma LPS tradicional, não sensível ao contexto. O resultado desta pesquisa bibliográfica está descrita

no Capítulo 2 e as abordagens identificadas para especificar casos de uso para uma LPS são apresentadas no Capítulo 3.

O segundo passo consiste em realizar um experimento controlado com os templates identificados afim de obter conhecimento empírico que auxiliasse na proposta de um template de caso de uso para uma LPSSC. O resultado desse experimento é descrito na Seção 4.1. Com base nesses resultados, o terceiro passo é a proposta, também apresentada na Seção 4.1, do CAPLUC (**C**ontext **A**ware software **P**roduct **L**ine **U**se **C**ase template), que especifica como um caso de uso de uma LPSSC pode ser descrito textualmente.

O quarto passo diz respeito a proposta de um método para geração de cenários de testes para uma LPSSC com base em casos de usos textuais. Esse método é apresentado na Seção 4.2 e é denominado de ChAPTER (**C**ontext **A**ware software **P**roduct line **TE**sting gene**R**ation method).

O quinto passo foi a implementação da ToChAPTER, ferramenta *web* que permite a modelagem de casos de uso de uma LPSSC segundo o CAPLUC e a aplicação do método ChAPTER. Detalhes sobre essa ferramenta são apresentados na Seção 4.3.

Por fim, o último passo dessa pesquisa foi a avaliação do ambiente de geração de cenários de testes, que é composto pelas propostas do CAPLUC e do ChAPTER e pela ferramenta ToChAPTER. Essa avaliação foi conduzida como um experimento controlado e é descrita no Capítulo 5.

4.2 Template de Caso de Uso para Linhas de Produto Sensíveis ao Contexto

Conforme mencionado no Capítulo 1, o principal objetivo deste trabalho é a proposta de um método de geração de testes para linhas de produto de software sensíveis ao contexto. Como este método utiliza casos de uso de uma LPSSC descritos em linguagem natural, foi realizada uma busca na literatura por propostas de templates direcionados para especificação textual de casos de uso, como apresentada em detalhes no Capítulo 3.

Pelos resultados dessa pesquisa bibliográfica, não foi encontrado nenhum template para descrição textual de casos de uso para uma LPSSC. Sendo assim, optou-se por propor um template que levasse em conta as informações de contexto e de variabilidade da linha.

Para fundamentar a proposta de template foi executado um experimento controlado com alguns dos templates encontrados na literatura para descrição textual de casos de uso de uma LPS. Nas subseções seguintes são apresentados detalhes do experimento controlado executado, a proposta de template de descrição textual de caso de uso para LPSSC e exemplos de uso do mesmo.

4.2.1 Análise dos templates existentes

Para guiar a proposta de um template para descrição textual de caso de uso para LPSSC, foram analisados os templates de caso de uso para LPS obtidos na literatura mediante a

pesquisa bibliográfica. Uma vez analisados os templates e identificadas as principais características de cada proposta, decidiu-se realizar um experimento controlado para comparar o impacto do formato desses templates na interpretação dos casos de uso. O objetivo desse experimento era extrair conhecimento empírico para fundamentar a proposta de template para caso de uso em LPSSC.

Dessa forma, o propósito do experimento realizado foi avaliar, com respeito ao esforço, do ponto de vista do pesquisador, no contexto de estudantes de Ciência da Computação e desenvolvedores, o impacto dos templates no entendimento dos casos de uso de uma LPS.

Vale ressaltar que esse experimento seguiu o guia apresentado por Wohlin et al. (WOHLIN et al., 2000) para realizar experimentos controlados, e foi baseado em outros experimentos (HADAR et al., 2010; MUSTAFA, 2010) que apresentam objetivos similares, pois avaliam a compreensão fornecida por casos de uso em diferentes formatos.

Como foram identificados nove templates (descritos em detalhes no Capítulo 3), o custo para fazer o experimento com todos seria oneroso, uma vez que cada template exige um treinamento apropriado e participantes que o utilizem. Dessa forma, três critérios foram estabelecidos para selecionar quais templates participariam do experimento: a) não modelar os produtos finais na especificação de casos de uso, pois isso reduz a manutenibilidade (BONIFÁCIO; BORBA, 2009); b) não depender de outro modelo ou ferramenta, pois limita o seu uso; e c) apresentar uma relação clara entre o cenário principal e os cenários que são variações deste cenário principal.

Esses critérios foram definidos como requisitos para o template proposto e foram estabelecidos com base na análise dos templates, na qual foram identificados alguns pontos negativos dos mesmos. Após a avaliação dos templates mediante os critérios de seleção, o template de Bertolino et al. (2003) foi excluído pelo item a, e o de Choi et al. (2008), Gallina et al.(2006) e Anthonysamy e Somé (2008) foram excluídos pelo item b. Além disso, o template proposto na abordagem de especificação baseada em fragmentos de Araújo (2010) foi excluída do experimento pelo item c.

Logo, quatro templates foram selecionados para o experimento: a proposta de caso de uso genérico de John e Muthing (2002), o template de Eriksson et al. (2004) que é uma extensão do Fluxo de Caixa Preta, o template apresentado por Gomma (2004) como parte do PLUS e a proposta de template presente na abordagem MSVCM apresentada por Bonifácio e Borba (2009).

4.2.1.1 Variáveis e Hipóteses

O estudo experimental foi projetado para responder as seguintes questões de pesquisa (QP):

- QP1: Qual dos templates avaliados favorece o entendimento dos casos de uso descritos com base no seu uso?

- QP2: Qual dos templates requer menos tempo para o entendimento dos casos de uso descritos com base no seu uso?
- QP3: O uso de templates que modelam variabilidades e similaridades no mesmo caso de uso facilita o entendimento quando comparado a templates que prescrevem a modelagem destes em lugares diferentes?

Essas questões foram então traduzidas em várias hipóteses (H):

- $H_{0\ accuracy}$: não existe diferença na acurácia obtida a partir da avaliação de casos de uso descritos utilizando diferentes templates. $H_{1\ accuracy}$: existe uma diferença na acurácia.
- $H_{0\ time}$: não existe diferença em termos de tempo requerido para entender os casos de uso descritos em diferentes templates. $H_{1\ time}$: existe uma diferença no tempo.
- $H_{0\ accuracygroup}$: não há diferença na acurácia quando se compara o uso dos templates com variabilidades e similaridades juntas na descrição do caso de uso contra os templates que prescrevem a descrição destes em lugares diferentes. $H_{1\ accuracygroup}$: existe uma diferença entre os resultados de acurácia quando se compara esses dois tipos de template.
- $H_{0\ timegroup}$: não há diferença no tempo gasto quando se compara o uso de templates com variabilidades e similaridades juntas na descrição do caso de uso contra os templates que prescrevem a descrição destes em lugares diferentes. $H_{1\ timegroup}$: existe uma diferença entre o tempo gasto quando se compara esses dois tipos de template.

As variáveis independentes do experimento são a ordem dos casos de uso utilizados, que é fixa, e o template de descrição textual de caso de uso, que pode assumir um dos quatro valores: “John e Muthing”, “Eriksson et al.”, “Gomma” ou “Bonifácio e Borba”.

As variáveis dependentes, por outro lado, são a acurácia e o tempo em minutos para responder as questões das tarefas de compreensão do experimento. Tais questões estão relacionadas a informações contidas nas especificações dos casos de uso utilizados no experimento e foram feitas para tentar medir o entendimento desses. Dessa forma, para cada caso de uso utilizado no experimento havia um questionário (Apêndice A) com duas questões sobre as variações (alternativas ou opcionais) presentes no caso de uso e uma questão relacionada ao entendimento da sequência de ações do caso de uso. Um exemplo desse último tipo de questão é exibida abaixo:

“Antes de selecionar a quantidade de dinheiro, o usuário precisa pelo menos:
a) Pelo menos fornecer o PIN
b) Pelo menos inserir o cartão com chip na ATM
c) Pelo menos fornecer uma impressão digital ou um exemplo de voz
d) Pelo menos fornecer o PIN e uma impressão digital ou o PIN e um exemplo de voz
e) Todas alternativas acima estão incorretas"

É importante mencionar que o objetivo das questões, como a descrita acima, era verificar se o voluntário conseguia interpretar o funcionamento do caso de uso, levando em conta as *features* opcionais e alternativas presentes no mesmo. No caso do exemplo, que se refere ao caso de uso apresentado na Figura 3.3, como o passo "fornecer o PIN" esta associado a uma *feature* opcional e os passos "fornecer uma impressão digital" e "fornecer um exemplo de voz", estão associados a uma *feature* alternativa opcional, tais passos podem estar presentes ou não. Dessa forma, a resposta correta seria a letra b, que se refere ao único passo obrigatório ("inserir cartão com chip na ATM"), que é apresentado nas alternativas dessa questão.

Com relação ao tempo gasto para analisar os casos de uso, esse foi medido em minutos. Os participantes, mediante supervisão do autor desta dissertação que atuava no experimento como moderador, registraram o tempo inicial e o tempo final de cada tarefa, e a subtração entre esses dois tempos revela o tempo gasto na atividade. Com isso, para cada voluntário do experimento, quatro registros de tempo gasto foram coletados.

Para avaliar a acurácia, por sua vez, os participantes responderam os questionários de acordo com o comportamento dos casos de uso. Como as questões relacionadas as variações alternativas e opcionais eram duas, os valores possíveis para a acurácia foram 0%, 50% ou 100%.

4.2.1.2 Projeto do estudo experimental

A Figura 4.2 ilustra os níveis dos participantes do experimento. O estudo experimental contou com a participação de 48 pessoas sendo 21 alunos de graduação, 20 alunos de pós-graduação (16 mestrandos e 4 doutorandos) e 7 desenvolvedores. Os alunos foram estudantes de graduação e de pós-graduação em Ciência da Computação da Universidade Federal do Ceará (UFC) e da Universidade Federal do Piauí (UFPI). Os desenvolvedores, por sua vez, trabalhavam em projetos de desenvolvimento do GREat - Grupo de Redes, Engenharia de Software e Sistemas[1].

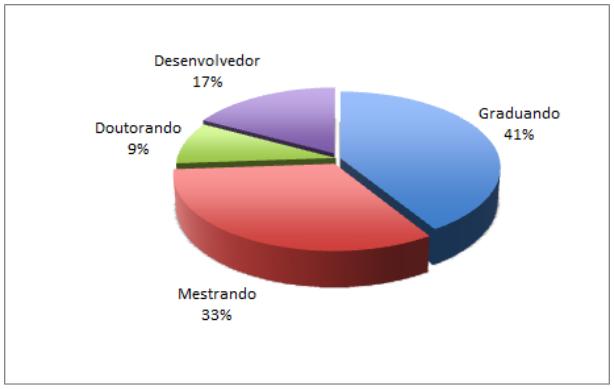


Figura 4.2: Nível dos participantes do experimento.

A Figura 4.3 exibe as atividades do estudo experimental. A primeira atividade correspondeu à execução de um questionário pré-experimento (Apêndice B). Esse questionário foi

[1]http://great.ufc.br

aplicado para caracterizar os voluntários do experimento e identificar os seus conhecimentos prévios com respeito aos tratamentos do estudo experimental.

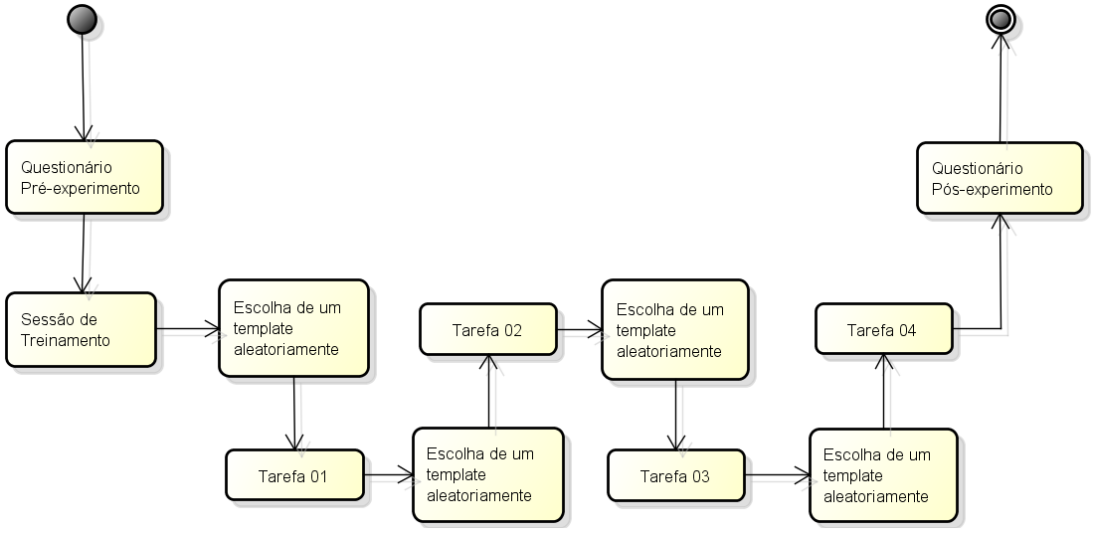


Figura 4.3: Atividades do estudo experimental.

Somente três dos vinte alunos de graduação afirmaram nunca ter estudado descrições textuais de caso de uso. Adicionalmente, nove estudantes de graduação nunca estudaram conceitos de LPS antes do experimento. No caso dos alunos de pós-graduação, somente dois alegaram nunca ter estudado descrição textual de caso de uso enquanto oito alunos não tinham estudado conceitos de LPS. Por fim, todos os desenvolvedores já utilizaram descrições textuais de caso de uso profissionalmente, mas três deles afirmaram nunca ter estudado LPS.

Com o objetivo de minimizar os efeitos da falta de conhecimento sobre os fatores do estudo experimental, uma sessão de treinamento foi conduzida com todos os participantes tentando explorar os conceitos básicos de LPS e de descrições textuais de casos de uso (segunda atividade da Figura 4.3).

Depois do treinamento, os participantes executaram as tarefas de compreensão (Tarefa 01, Tarefa 02, Tarefa 03 e Tarefa 04). Em cada tarefa o participante respondia um questionário, ao tempo que analisava uma descrição textual de caso de uso que era especificada em um dos quatro templates (John e Muthing, Eriksson, Gomma, Bonifacio e Borba) escolhido via sorteio. Os quatro casos de uso utilizados no experimento foram extraídos dos próprios artigos dos templates (JOHN; MUTHIG, 2002; GOMAA, 2004; ERIKSSON; BÖRSTLER; BORG, 2004; BONIFáCIO; BORBA, 2009).

Com relação ao questionário da tarefa, ele foi diretamente relacionado ao entendimento do caso de uso. O tempo gasto para responder o questionário, bem como com a pontuação obtida com as respostas corretas foram utilizadas para avaliar o desempenho dos templates. Por fim, após as quatro tarefas os participantes responderam um questionário pós-experimento (Apêndice C) indicando dentre outras coisas qual template ele considerou mais útil em cada tarefa de compreensão.

4.2.1.3 Resultados

Nesta subseção são apresentados os resultados do experimento. A Figura 4.4 apresenta um sumário dos dados coletados. O estudo experimental teve um total de 48 voluntários e como cada um executou quatro atividades, o resultado foi de 192 tarefas. Contudo, as tarefas cuja última questão não foi respondida corretamente foram reprovadas. Isso porque a última questão do questionário realizado em cada tarefa estava associada ao entendimento do funcionamento do caso de uso. Logo, uma resposta incorreta nesta questão sinaliza um erro no uso do template. Como resultado, somente 134 tarefas foram avaliadas, pois as 58 restantes foram reprovadas.

Nível	Template	Amostras (#)	Média Tempo (m)	Acurácia (%)
Estudantes de graduação	Bonifacio and Borba	10	8,20	50,00
	Eriksson	9	5,55	88,88
	Gomma	21	6,04	73,80
	John	15	7,13	53,33
	Subtotal	**55**	**6,65**	**66,36**
Estudantes de pós-graduação	Bonifacio and Borba	12	8,41	50,00
	Eriksson	22	4,50	95,45
	Gomma	15	5,40	96,67
	John	7	7,71	85,71
	Subtotal	**56**	**5,98**	**84,82**
Desenvolvedores	Bonifacio and Borba	1	6,00	50,00
	Eriksson	8	4,50	100,00
	Gomma	5	5,20	70,00
	John	9	5,00	100,00
	Subtotal	**23**	**4,91**	**91,30**
Total	Bonifacio and Borba	23	8,21	50,00
	Eriksson	39	4,74	94,87
	Gomma	41	5,70	81,70
	John	31	6,64	74,19
	Subtotal	**134**	**6,07**	**78,35**

Figura 4.4: Sumário dos resultados do experimento.

Os resultados mostraram que a acurácia e o tempo gasto para executar as atividades foram melhores para os desenvolvedores do que para os alunos de graduação e pós-graduação. Os alunos de pós-graduação tiveram um resultado melhor do que os alunos de graduação. Esse foi um resultado esperado, já que, em geral, desenvolvedores tem um contato maior com descrições de casos de uso do que os alunos de pós-graduação, os quais, por sua vez, geralmente são mais experientes que os alunos de graduação.

Os participantes do experimento que utilizaram o template proposto por Eriksson et al. (2004) tiveram os melhores resultados, pois eles gastaram menos tempo e tiveram mais acurácia se comparado aos participantes que usaram os outros templates. O segundo melhor resultado foi obtido pelos voluntários que usaram o template de Gomma (2004), embora o grupo de desenvolvedores tenha tido melhor resultado usando o template proposto por John e Muthing. Por fim, os participantes que utilizaram o template apresentado no trabalho de Bonifácio e Borba (2009) atingiram os piores resultados em todos os grupos.

Análise dos Dados

Para analisar os resultados foi aplicado o teste *Kolmogorov-Smirnov* (HOLLANDER; WOLFE, 1999) para avaliar se a distribuição seguia a distribuição normal. Como a distribuição da amostra não seguia a distribuição normal, foram utilizados testes não paramétricos para avaliar os dados.

Foi então utilizado o teste *Kruskal Wallis* (HOLLANDER; WOLFE, 1999), método não paramétrico para teste de hipóteses quando amostras são originadas da mesma distribuição. O teste indicou que o tempo gasto e a acurácia dos grupos tem diferenças estatisticamente significativas.

Uma vez detectada a diferença no tempo e na acurácia associado com o uso dos templates, uma análise foi feita cruzando dados template a template, tentando identificar o que causava as diferenças. Nesse caso, nós utilizamos o teste *Mann-Whitney* para análise dos dados.

Os resultados associados ao uso do template de "Bonifácio e Borba" foram os piores para tempo e acurácia. O tempo gasto usando esse template foi estatisticamente maior que o tempo gasto usando os outros templates. A acurácia foi estatisticamente menor que a acurácia relacionada ao uso dos outros templates. Além disso, o template foi apontado como preferido por apenas 3% dos voluntários.

O uso dos templates de Gomma e de "John e Muthing", por sua vez, exibiram resultados estatisticamente iguais tanto no tempo para executar as tarefas quanto na acurácia dos resultados. Os dados destes templates indicaram uma diferença positiva estatisticamente significante quando comparada ao template de "Bonifácio e Borba". O template do Gomma teve preferência de 29.1% dos voluntários do experimento, enquanto que o template de "John e Muthing" teve uma preferência de 21.6%.

Por fim, o uso do template do "Eriksson et al." mostrou os melhores resultados de tempo e acurácia no experimento. O tempo para executar as tarefas foi o menor e foi possível registrar uma acurácia dos resultados estatisticamente superior quando comparado aos outros templates. Em termos de preferências dos voluntários, ele também foi o favorito com 46.3% de indicação. Dessa forma, o template do Eriksson et al. (2004) além de ser o de melhores resultados em termos de tempo e acurácia, foi também o favorito entre os voluntários.

Destaca-se que o número de tarefas consideradas incorretas gerou um alerta. No total, em cerca de 30% das tarefas registradas a questão final foi respondida de maneira incorreta. Por esse motivo, outras análises foram executadas tentando identificar a fonte do problema. Com base nos dados do questionário pós-experimento, foi possível analisar os dados relacionados a adequabilidade do tempo para completar a tarefa junto com o entendimento do objetivo das tarefas e o entendimento do problema. Os resultados dessa análise indicaram que não há diferença estatisticamente significativa entre os participantes que responderam corretamente a questão final e os que erraram. Logo, isso sinaliza que o erro na questão final não está relacionado ao tempo para completar as tarefas.

Devido aos resultados obtidos, outra análise foi executada sobre os dados relacionados ao perfil de cada voluntário. Esse dados estavam associados com as experiências anteriores e o conhecimento dos participantes sobre as tecnologias que iriam ser utilizadas no experimento.

A análise mostrou indícios de que a experiência em diagramas de casos de uso e em LPS não impactou nos resultados, ou seja, os resultados dos participantes com experiência prévia contra os participantes sem experiência foram estatisticamente iguais. Contudo, usando o teste *Mann Whitney* foi detectada uma diferença estatisticamente significativa em relação a experiência com casos de uso expressos em linguagem natural. Logo, o conhecimento prévio em casos de uso descritos em linguagem natural parece ser fortemente associado com a causa dos erros.

A Figura 4.5 ilustra as frequências das respostas relacionadas a experiência em casos de uso descritos em linguagem natural. O grupo com resposta errada na última questão das tarefas tem mais participantes com zero ou baixo conhecimento. Logo, a falta de conhecimento nesse tema tornou mais difícil responder as questões do caso de uso.

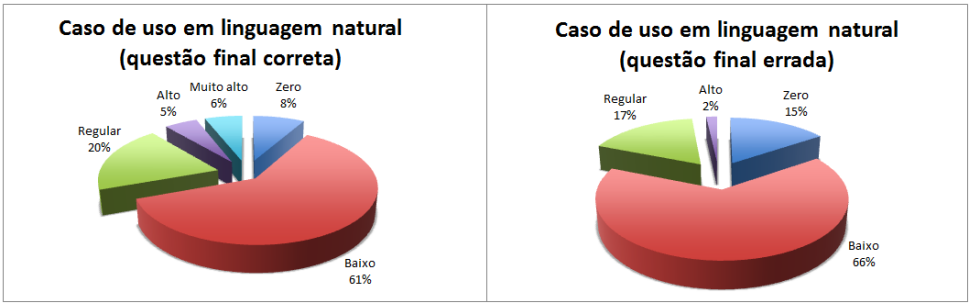


Figura 4.5: Conhecimento dos participantes.

Questões de Pesquisa

Depois da execução do experimento foi possível responder as questões de pesquisa propostas (QP):

- QP1: Qual dos templates avaliados favorece o entendimento dos casos de uso descritos com base no seu uso? O template de Eriksson et al. (2004) melhor favorece o entendimento dos casos de uso. Os participantes usando esse template tiveram melhores resultados em termos de tempo gasto e acurácia;

- QP2: Qual dos templates requer menos tempo para o entendimento dos casos de uso descritos com base no seu uso? O template de Eriksson et al. (2004) exige menos tempo para o entendimento do caso de uso. Os participantes usando esse template tiveram melhores resultados em termos de tempo gasto.

- QP3: O uso de templates que modelam variabilidades e similaridades no mesmo caso de uso facilita o entendimento quando comparado a templates que prescrevem a modelagem destes em lugares diferentes? Foram coletadas duas evidências para responder essa questão: existe uma diferença estatisticamente significativa quando comparado o uso dos templates que modelam variabilidades e similaridades juntas (“Eriksson et al.” e “John e Muthing”) quando comparado a templates que as modelam separadamente (”Bonifácio e Borba“ e Gomma). Utilizando o teste *Kruskal Wallis* percebeu-se que o tempo e

a acurácia são melhores em favor dos templates que modelam variabilidades e similaridades juntas. Além disso, foi registrado uma preferência maior dos participantes pelos templates com variabilidades e similaridades juntas (68%) contra separadamente (32%).

Discussão

Com a realização do experimento foi observado que as características de cada template geram impacto no entendimento dos casos de uso. Os voluntários que selecionaram o template do "Eriksson et al." como o melhor template, por exemplo, reportaram que ele possui uma descrição simples e objetiva, além de uma estrutura clara, organizada e compacta. Essas características tornaram fácil e intuitivo identificar o quanto o passo do caso de uso era obrigatório, opcional ou alternativo.

Com relação ao template do Gomma (2004), o seu diferencial está no fato de deixar de forma clara, no final do caso de uso, quais *features* opcionais e alternativas afetam o caso de uso e em que passo do caso de uso elas atuam. Contudo, entrevistas informais com os voluntários e os resultados do experimento indicaram que a desvantagem deste template é que é preciso ir e voltar várias vezes do final do caso de uso para o fluxo principal para entender como funciona de fato o caso de uso.

Também vale mencionar que a desaprovação relacionada ao template de "Bonifácio e Borba" pode ser justificada devido a separação entre o cenário principal e as variações sem uma definição explícita nos casos de uso do tipo de variação, tornando difícil entender se a variação é opcional ou alternativa.

Além disso, conforme observado durante o experimento e comentado por um dos voluntários, o caso de uso que descreve as variabilidades em conjunto com as similaridades torna fácil o entendimento da operação do caso de uso, enquanto que a descrição separada das variabilidades e similaridades torna fácil o reconhecimento das variações.

4.2.1.4 Ameaças a validade

Nesta subseção são discutidas as ameaças do experimento relacionadas a validade de conclusão, validade interna e a validade externa.

As ameaças a validade de conclusão são preocupadas com questões que afetam a correta conclusão sobre os relacionamentos entre os tratamentos e as saídas do experimento (WOHLIN et al., 2000). No caso do experimento, uma vez que houve a participação de 48 voluntários e os testes estatísticos foram escolhidos respeitando a distribuição probabilística da amostra, acredita-se que esse tipo de ameaça foi superado.

Com relação as ameaças à validade interna, essa está relacionada a fatores que podem indicar que o tratamento causa a saída quando na verdade isso não ocorre (WOHLIN et al., 2000). No experimento medidas foram tomadas para minimizar os efeitos dessas ameaças. Os problemas da instrumentação, por exemplo, que estão relacionados aos artefatos utilizados

no experimento foram minimizados porque tanto os casos de uso quanto os questionários das tarefas de compreensão foram entregues impresso aos participantes.

Também foram utilizados quatro casos de uso, de baixa complexidade, com diferentes características para minimizar os efeitos de Maturação. Esses casos de usos tem poucos aspectos em comum e com isso eles minimizam uma influência positiva no tratamento usado. Também acredita-se que isso ajudou a evitar a influência negativa causada pela repetição de uma atividade similar.

Como mencionado anteriormente, durante o experimento foi feito o uso das questões para avaliar o correto entendimento dos casos de uso. Assim, a questão final foi utilizada para cortar as execuções sem qualidade.

Finalmente, para evitar ameaças relacionadas ao uso de um único grupo, os quatro tratamentos (templates) foram usados por todos os sujeitos e a ordem dos templates utilizada foi totalmente aleatória. A ordem não foi definida para evitar os problemas de *compensatory equalization of treatments , compensatory rivalry e resentful demoralization* (WOHLIN et al., 2000). Entretanto, é importante mencionar que a ordem dos casos de uso era fixa e, com isso, em geral, os voluntários foram mais lentos no entendimento do primeiro caso de uso, correspondente a Tarefa 01.

As ameaças a validade externa, por sua vez, se preocupam com a habilidade de generalizar os resultados (WOHLIN et al., 2000). O uso de estudantes como voluntários é uma ameaça. Entretanto, o estudo foi executado com a presença de estudantes de graduação, de pós-graduação e desenvolvedores. Logo, acredita-se que a amostra é representativa.

Além disso, os quatro casos de uso usados no estudo são apresentados nos artigos que descrevem esses templates. Eles são pequenos e similares em termos de complexidade e tamanho, mas tem todas as características de casos de uso comuns presentes em vários sistemas de software.

Dessa forma, acredita-se que com mais experimentos, com diferentes tipos de usuários e com casos de uso de diferentes complexidades, as conclusões que foram obtidas no experimento descrito nesta seção podem ser estendidas.

4.2.2 CAPLUC: Context Aware software Product Line Use Case template

Mediante a pesquisa bibliográfica executada e os resultados do experimento, foi proposto como um resultado parcial deste trabalho um template para LPSSC que permite a definição de informações sobre a funcionalidade dos produtos da linha, a variabilidade dos produtos e a influência do contexto nos produtos da linha. O template foi proposto incorporando as principais vantagens dos templates com os melhores resultados do experimento, Gomma (2004) e Eriksson et al. (2004). Esse template para descrição textual de casos de uso de LPSSC, apresentado nessa subseção, é denominado de CAPLUC (acrônimo de **C**ontext **A**ware software **P**roduct **L**ine **U**se **C**ase template).

É importante notar que para grandes variações é recomendado o uso de mecanismos

de extensões dos casos de uso, conforme proposto por Gomma (GOMAA, 2004). Por outro lado, pequenas variações podem ser descritas como pontos de variação dentro dos casos de uso e nesse caso, os templates apresentados no Capítulo 2 para LPS e o CAPLUC para LPSSC podem ser utilizados.

Segundo o CAPLUC, o caso de uso da LPSSC deve ser definido em formato tabular por uma tupla de doze elementos: [Nome, Caso de Uso Estendido, Ponto de Extensão, Categoria de Reuso, Restrição de Contexto, Resumo, Atores, Pre-condição, Pos-Condição, Passos, Fluxos Alternativos, Sumário de Variações]. Além disso, é previsto no CAPLUC o uso de variáveis globais e locais da mesma forma como proposto por Eriksson et al. (2004). As variáveis globais são precedidas do simbolo @ e afetam todos os casos de uso da linha, enquanto que as variáveis locais, precedidas de $ afetam apenas o caso de uso em que elas aparecem.

A Figura 4.6 sumariza a proposta do CAPLUC. A maioria dos elementos do template são comuns a templates de caso de uso de aplicações tradicionais e funcionam da mesma forma. A seguir, são descritos mais detalhes dos elementos específicos para tratar das informações de variabilidade e de contexto de uma LPSSC:

- *Categoria de Reuso*. Este elemento foi extraído do template do Gomma (2004) e serve para identificar se o caso de uso é obrigatório, opcional ou alternativo.

- *Restrição de Contexto*. O propósito desse elemento é permitir a especificação de em qual contexto este caso de uso é aplicável. Ele foi proposto no CAPLUC porque em LPSSC mudanças de contexto podem influenciar a configuração da aplicação em tempo de execução, e com isso, casos de uso passam a ser "habilitados" ou "desabilitados" para execução de acordo com as *features* que permanecem ativadas.

- *Passos*. O CAPLUC segue o estilo proposto por Eriksson et al. (2004) apresentado na Seção 3.1.3, mas além de indicar as variabilidades pelos índices dos passos, propõe-se especificar o ponto de variação a que o passo se refere. Nesse caso, deve-se indicar o nome do Ponto de Variação e o Nome da Variante associado aos passos que representam uma variação. Se for um ponto de variação alternativo pode-se usar o prefixo "Alt" para simbolizar a escolha da alternativa. Além disso, o símbolo * deve ser utilizado para indicar passos que estão relacionados a restrições de contexto.

- *Sumário de Variações*. Neste elemento são especificadas as variações (pontos de variação) que afetam o caso de uso. O objetivo deste elemento é tornar a identificação das variações fácil e direta. Cada ponto de variação representa o local onde há uma variabilidade e é composto por uma tupla [Tipo de Variabilidade, Questão Descritiva (opcional), Variantes]. O tipo de variabilidade indica se o ponto de variação é de *features* alternativas ou opcionais. A ideia da questão descritiva foi extraída do template do "John e Muthing" (2002) e serve tanto para auxiliar o entendimento do caso de uso quanto para apoiar a instanciação dos produtos (ao responder as perguntas obtém-se quais as *features* selecionadas, e com isso é possível instanciar os produtos respondendo questões associadas aos pontos de variação). As variantes, por sua vez, são uma tupla [Nome, Local, Restrição de Contexto], onde o local indica em qual passo do caso de uso essa variação está presente

Elemento		Descrição	
Nome		*Nome do caso de uso. O nome deve refletir o objetivo do caso de uso*	
Caso de Uso Estendido		*Nome do caso de uso cujo comportamento é estendido por este caso de uso*	
Ponto de Extensão		*Local do caso de uso estendido onde este caso de uso atua*	
Categoria de Reuso		*Especificar o quanto o caso de uso é Obrigatório, Opcional ou Alternativo {colocar o nome dos casos de uso alternativos aqui}*	
Restrição de Contexto		*Descrição da condição de contexto que precisa ser verdadeira para que este caso de uso seja passível de execução*	
Resumo		*Resume o objetivo do caso de uso*	
Atores		*Descreve os atores do caso de uso, tanto os primários (que iniciam o caso de uso) quanto os secundários (que podem participar do caso de uso)*	
Pré-condição		*Especifica uma ou mais condições que devem ser verdadeiras no início do caso de uso*	
Pós-condição		*Especifica a condição que sempre é verdadeira ao final do caso de uso se a sequencia principal foi seguida*	
Passo		**Usuário**	**Sistema**
[Nome do Ponto de Variação] (Alt) [Nome da Variante]	*Número do Passo*	*Ação do Usuário*	*Resposta do Sistema*
Fluxos Alternativos			
Restrição para a execução do fluxo alternativo. No caso do produto, aqui também podem ser colocadas restrições de contexto para influenciar o comportamento da aplicação		*Descrição dos passos alternativos*	
Sumário das Variações Alternativas			
[Nome do ponto de variação alternativo]: *Questão descritiva*		***[Nome da Variante Alternativa 1]*** *(Restrição de Contexto: contexto que deve ser verdadeiro para a variante ser passível de escolha)*	*Identificação do passo na forma "Passo X"*
		[Nome da Variante Alternativa 2] *(Restrição de Contexto: contexto que deve ser verdadeiro para a variante ser passível de escolha)*	*Identificação do passo na forma "Passo X"*
		[Nome da Variante Alternativa N] *(Restrição de Contexto: contexto que deve ser verdadeiro para a variante ser passível de escolha)*	*Identificação do passo na forma "Passo X"*
Sumário das Variações Opcionais			
[Nome do ponto de variação opcional]: *Questão descritiva*		***[COM Nome da Variante Opcional]*** *(Restrição de Contexto: contexto que deve ser verdadeiro para a variante ser passível de escolha)*	*Identificação do passo na forma "Passo X"*
		[SEM Nome da Variante Opcional] *(Restrição de Contexto: contexto que deve ser verdadeiro para a variante ser passível de escolha)*	*Identificação do passo na forma "Passo X"*

Figura 4.6: CAPLUC, um template para descrição textual de casos de uso de LPSSC.

e a restrição de contexto especifica em que contexto esta variação é passível de ocorrer. Destaca-se que no caso das variações opcionais, um comportamento específico ocorre na presença da *feature* opcional (COM) e outro pode ocorrer na ausência dessa *feature* opcional (SEM).

Com a consolidação desse template espera-se produzir um meio que pode trazer os seguintes benefícios: a) melhor entendimento sobre a influência do contexto nos produtos da linha; b) Suporte a rastreabilidade entre requisitos e contexto; e c) Suporte às abordagens de apoio ao desenvolvimento de LPSSC existentes (FERNANDES; WERNER; TEIXEIRA, 2011; MARINHO et al., 2010).

4.2.3 Exemplo de Uso do CAPLUC

Nesta seção são apresentados três casos de usos modelados com o CAPLUC: “Acesso ao Contexto”, “Captura Contexto” e “Mostrar Documentos”. Esses casos uso foram feitos para a linha de guias de visita móvel e sensível ao contexto do projeto Mobiline (MARINHO et al., 2010; MOBILINE, 2012).

O primeiro caso de uso trata da forma de acesso as informações de contexto, ilustrado na Figura 4.7. Nesse caso uso está presente apenas um ponto de variação alternativo, *Acesso*, que possui duas variantes: Síncrono e Assíncrono. Ambas as variantes estão ligadas ao passo 02 do caso de uso, logo apenas uma deve ser escolhida. Além disso, nesse caso de uso não existe nenhuma restrição de contexto influenciando o funcionamento ou a configuração dos produtos finais, mas existe um fluxo alternativo especificando que o sistema deve esperar no máximo por 10s as informações de contexto.

Elemento do Caso de Uso		**Descrição**	
Nome		Acesso ao contexto	
Caso de Uso Estendido		--	
Ponto de Extensão		--	
Categoria de Reuso		Obrigatório	
Restrição de Contexto		--	
Resumo		Acesso às informações de contexto	
Atores		Gerenciador de Contexto (secundário)	
Pré-condição		O Usuário deve estar autenticado no sistema	
Pós-Condição		O Sistema obteve acesso a informação de contexto	
Passo		**Ações do Usuário**	**Ações do Sistema**
	1	--	O Sistema solicita a informação de contexto
[Acesso] Alt [Síncrono]	**2**	--	O Sistema fica bloqueado aguardando a informação de contexto
[Acesso] Alt [Assíncrono]	**2**	--	O Sistema efetua outras operações enquanto aguarda a informação de contexto
	3	--	O Gerenciador de Contexto notifica o Sistema indicando que o contexto está disponível
Fluxos Alternativos			
Fluxo Alternativo A (1a: Aguardando contexto)		2a. O Sistema deve aguardar no máximo por um período de 10s.	
Sumário de Variações Alternativas			
[Acesso]: “Qual forma de acesso ao contexto?”		**[Síncrono]** (Restrição de Contexto:)	Passo 2
		[Assíncrono] (Restrição de Contexto:)	Passo 2
Sumário de Variações Opcionais			

Figura 4.7: Caso de uso Acesso ao Contexto modelado no CAPLUC.

O próximo caso de uso apresentado nesta seção é o “Captura de Contexto”, que detalha como são capturadas as informações de contexto. Na LPSSC Mobiline, existiam quatro opções: a) Sensor; b) Memória; c) Qr_code; e d) Serviço Externo. Cada opção representava uma *feature* que está associado ao ponto de variação *Origem*, conforme apresentado na Figura 4.8. Note que as variantes “Sensor” e “Qr_code” apresentam restrições de contexto e por isso os passos que elas estão associadas apresentam o símbolo *.

<table>
<tr><th>Elemento do Caso de Uso</th><th colspan="4">Descrição</th></tr>
<tr><td>Nome</td><td colspan="4">Captura de Contexto</td></tr>
<tr><td>Caso de Uso Estendido</td><td colspan="4">--</td></tr>
<tr><td>Ponto de Extensão</td><td colspan="4">--</td></tr>
<tr><td>Categoria de Reuso</td><td colspan="4">Obrigatório</td></tr>
<tr><td>Restrição de Contexto</td><td colspan="4">--</td></tr>
<tr><td>Resumo</td><td colspan="4">Captura informações de contexto</td></tr>
<tr><td>Atores</td><td colspan="4">Gerenciador de Contexto (secundário)</td></tr>
<tr><td>Pré-condição</td><td colspan="4">O Usuário deve estar autenticado no sistema</td></tr>
<tr><td>Pós-Condição</td><td colspan="4">Informações de contexto foram capturadas</td></tr>
<tr><td colspan="2">Passo</td><td>Ações do Usuário</td><td colspan="2">Ações do Sistema</td></tr>
<tr><td></td><td>1</td><td>--</td><td colspan="2">O Sistema solicita para o Gerenciador de Contexto alguma informação de contexto informando o tipo de informação que ele precisa</td></tr>
<tr><td>[Origem]
Alt [Sensor]</td><td>2*</td><td>--</td><td colspan="2">O Sensor retorna a informação de contexto</td></tr>
<tr><td>[Origem]
Alt [Memória]</td><td>2</td><td>--</td><td colspan="2">A Memória retorna a informação de contexto</td></tr>
<tr><td>[Origem]
Alt [Qr_code]</td><td>2*</td><td>--</td><td colspan="2">O leitor de QR Code retorna a informação de contexto</td></tr>
<tr><td>[Origem]
Alt [Serviço Externo]</td><td>2</td><td>--</td><td colspan="2">O Serviço Externo retorna a informação de contexto</td></tr>
<tr><td colspan="5">Fluxos Alternativos</td></tr>
<tr><td colspan="2"></td><td colspan="3"></td></tr>
<tr><td colspan="5">Sumário de Variações Alternativas</td></tr>
<tr><td colspan="2" rowspan="4">[Origem]: “Qual meio utilizado para captura de contexto?”</td><td colspan="2">[Sensor]
(Restrição de Contexto: SENSOR AND SENSORES_AMBIENTE)</td><td>Passo 2</td></tr>
<tr><td colspan="2">[Memória]
(Restrição de Contexto:)</td><td>Passo 2</td></tr>
<tr><td colspan="2">[Qr_code]
(Restrição de Contexto: NOT_SENSOR OR NOT_SENSORES_AMBIENTE)</td><td>Passo 2</td></tr>
<tr><td colspan="2">[Serviço Externo]
(Restrição de Contexto:)</td><td>Passo 2</td></tr>
</table>

Figura 4.8: Caso de uso Captura de Contexto modelado no CAPLUC.

A variante “Sensor” só pode ser selecionada se os contextos *SENSOR*, indicando que o dispositivo que executa o guia móvel apresenta um sensor, e *SENSORES_AMBIENTE*, indicando que há sensores no ambiente, forem verdadeiros. Por outro lado, a variante “Qr_-code” só pode ser escolhida se os contextos *SENSOR* e *SENSORES_AMBIENTE* forem falsos.

Por fim, como uma das funcionalidades dos guias de visita modeladas na linha Mobiline havia a exibição dos documentos associados a cada ambiente. Tais documentos podiam ser do tipo texto, imagem ou vídeo. No caso do GREatTour, por exemplo, que é um produto dessa LPSSC, quando o visitante chega a sala de um pesquisador do laboratório GREat, é exibido um texto descrevendo as principais pesquisas deste. Já no caso do visitante se encontrar na sala de seminários são exibidas algumas imagem e vídeos de algumas apresentações feita nessa sala. O caso de uso que especifica essa funcionalidade foi denominado de “Mostra Documentos” e é ilustrado na Figura 4.9.

Conforme observado na Figura 4.9, o caso de uso “Mostrar Documentos” é composto por 2 pontos de variações: a) Imagem e b)Vídeo. O ponto de variação opcional *Imagem*

Elemento do Caso de Uso		**Descrição**	
Nome		Mostra Documentos	
Caso de Uso Estendido		--	
Ponto de Extensão		--	
Categoria de Reuso		Obrigatório	
Restrição de Contexto		--	
Resumo		Exibe os documentos associados a um ambiente	
Atores		Gerenciador de Contexto, Servidor (secundário)	
Pré-condição		O Usuário deve estar autenticado no sistema	
Pós-Condição		Informações sobre o ambiente são exibidas ao usuário da aplicação	
Passo		**Ações do Usuário**	**Ações do Sistema**
	1	--	O Gerenciador de Contexto é notificado de que o Usuário mudou de ambiente
	2	--	O Sistema solicita a algum WebService do Servidor o mapa e os arquivos relacionados ao ambiente em que se encontra o Usuário
	3	--	O Sistema verifica quais arquivos pode exibir ao Usuário com base no perfil dele
	4	--	O Sistema exibe ao Usuário os arquivos no formato de Texto
[Imagem]	**(5*)**	--	O Sistema exibe ao Usuário os arquivos de Imagem
[Video] [COM Video]	**(6*)**	--	O Sistema exibe ao Usuário os arquivos de Vídeo
[Video] [SEM Video]	**(7*)**	--	O Sistema informa ao usuário que vídeos não são exibidos devido a baixa bateria do celular ou ausência de rede Wifi
Fluxos Alternativos			
***Restrição de Contexto**: NOT_3G AND NOT_WIFI*		4. A aplicação retorna ao usuário uma mensagem indicando que os arquivos estão indisponíveis porque nenhuma rede foi localizada	
Sumário de Variações Alternativas			
Sumário de Variações Opcionais			
[Imagem]: "Imagens serão exibidas ao usuário?"		**[Imagem]** (Restrição de Contexto: WIFI)	Passo 5
[Video]: "Vídeos serão exibidos ao usuário?"		**[COM Video]** (Restrição de Contexto: WIFI AND NOT_LOW_BATTERY)	Passo 6
		[SEM Video] (Restrição de Contexto: NOT_WIFI OR LOW_BATTERY)	Passo 7

Figura 4.9: Caso de uso Mostra Documentos modelado no CAPLUC.

só tem uma variante que possui o mesmo nome. Ressalva-se que a variante "Imagem" só pode estar disponível se for verdadeiro o contexto *WIFI*, que se refere a disponibilidade de uma rede *wifi* para uso pelo dispositivo com o guia de visita móvel. Além disso, essa variante opcional está associada ao passo 5 do caso de uso, e com isso o índice desse passo fica entre parênteses e esse passo só é executado se essa variante for selecionada.

O ponto de variação *Vídeo*, diferentemente do *Imagem*, apresenta um comportamento na presença da variante opcional "Video" e outro comportamento para a ausência dessa variante. Por isso, duas variantes ("COM Video" e "SEM Video") estão relacionadas ao ponto de variação *Vídeo*. Cada uma dessas variantes apresenta uma restrição de contexto. No caso da variante "COM Video" é preciso que a rede *wifi* esteja disponível (*WIFI*) e o dispositivo móvel no qual o guia de visitas está executando não pode estar com bateria baixa (*NOT_LOW_BATTERY*). Já no caso do dispositivo estar com bateria baixa (*LOW_BATTERY*) ou sem rede *wifi*

(*NOT_WIFI*) a opção é não selecionar a variante Vídeo.

Finalmente, vale notar que no caso de não haver rede wifi (NOT_WIFI) ou rede 3G (NOT_3G), a aplicação não exibe nenhum documento e retorna uma mensagem ao usuário, conforme especificado nos fluxos alternativos.

Sobre os exemplos apresentados nessa seção, uma consideração deve ser feita sobre os contextos apresentados nos casos de uso. Como a especificação dos casos de uso acontece nos estágios iniciais do desenvolvimento da linha, as informações de contexto ainda estão em alto nível (*WIFI, 3G, LOW_BATTERY, SENSOR, SENSORES_AMBIENTE*). Por isso, pode-se dizer que nesse momento utiliza-se situações de contexto, como apresentado na Seção 2.1.3. Na engenharia de aplicação, quando mais detalhes sobre os produtos finais são conhecidos, essas situações de contexto podem ser detalhadas por meio de informações de contexto de baixo nível. Por exemplo, no caso da Engenharia de Aplicação do GREatTour, a situação de contexto *LOW_BATTERY* é descrita, em um documento sobre essa aplicação, como sendo o estado em que a bateria do dispositivo que executa o GREatTour está abaixo de 5%.

4.3 Método de Geração de teste para Linhas de Produto Sensíveis ao Contexto

A principal contribuição deste trabalho é a proposta do ChAPTER (**C**ontext **A**ware software **P**roduct line **TE**sting gene**R**ation method), que é descrito em detalhes neste capítulo. Esse método consiste na geração de testes para LPSSC a partir de descrições textuais de casos de uso.

Ressalta-se que como o método proposto utiliza descrições textuais de casos de uso em linguagem natural, ele se mantém simples em sua essência porque não exige conhecimentos em uma ferramenta ou linguagem de modelagem específica. Isso pode ser uma característica relevante, pois segundo Engstrom e Runeson (2011) quase todas as abordagens de teste de LPS existentes são idealistas porque são difíceis de usar, uma vez que requerem maiores mudanças no processo de desenvolvimento, como por exemplo, o uso de modelos formais para requisitos e variabilidade.

Com relação a proposta do método, ela foi concebida após análise das abordagens existentes, onde percebeu-se que uma extensão ou adaptação do método PLUTO (*Product Line Use Case Test Optimisation*) (BERTOLINO; GNESI, 2003) para tratar de LPSSC poderia ser interessante porque o PLUTO gera cenários de testes diretamente dos casos de uso sem a necessidade de um modelo intermediário. Dessa forma, o método ChAPTER foi desenvolvido tendo como base o método PLUTO (BERTOLINO; GNESI, 2003) proposto para Linhas de Produto de Software tradicionais.

O PLUTO é uma metodologia para gerenciar o processo de teste de linhas de produto, onde cenários de teste são gerados a partir de descrições de caso de uso especificados como PLUCs (*Product Lines Use Cases*) (FANTECHI et al., 2003). Para tornar esta geração possível, Bertolino e Gnesi (2003) propõem uma extensão do método de Partição de Categorias (OSTRAND; BALCER, 1988), que é uma abordagem de teste de caixa-preta criada originalmente para derivar testes funcionais de especificações escritas em linguagem natural.

Assim como o PLUTO, o ChAPTER também aborda no problema de teste baseado em especificações de requisitos, mas como o foco do método proposto neste trabalho são linhas sensíveis ao contexto, o ChAPTER traz diferenciais peculiares nos seguintes pontos:

- *Template do Caso de uso utilizado*. O PLUTO utiliza PLUCs que são casos de usos que descrevem por meio de tags as variações dos produtos. No caso do ChAPTER, por lidar com uma linha na qual informações de contexto afetam o comportamento das aplicações, ele utiliza como entrada casos de uso que descrevem funcionalidade, variabilidade e informações de contexto. Logo, o ChAPTER usa casos de uso denominados no método de SLICEs (***S**oftware product **LI**ne **C**ontextual use cas**E***). Na Seção 4.1, por exemplo, é apresentada uma proposta de template para um SLICE;

- *Uso da técnica de Partição de Categorias*. O PLUTO estende o método de partição de categorias para tratar das variabilidades. O ChAPTER, por sua vez, estende a adaptação desse método feita no PLUTO para tratar de informações de contexto. A Tabela 4.1, apresenta uma comparação detalhada entre os passos da técnica de Partição de Categorias original, da extensão do PLUTO e do ChAPTER. As diferenças são consequências do ChAPTER usar SLICES que contém informações de variabilidade e contexto; e

- *Etapas do Método*. O PLUTO é dividido em duas etapas: na primeira etapa, o método consegue todos os cenários de todos os produtos da linha por meio das combinações possíveis de escolhas; e na segunda etapa, por meio da instanciação das *tags* e com isso a definição do produto final, é possível derivar do conjunto de cenários gerados na fase anterior uma lista de testes para o produto específico. Já no ChAPTER, por lidar com uma linha na qual seus produtos podem se reconfigurar, são propostas três etapas de geração de testes, conforme será descrito na próxima seção.

Tabela 4.1: Comparação entre o Método de Partição de Categorias, PLUTO e ChAPTER

	Partição de Categorias Original	**PLUTO**	**ChAPTER**
Passo 01	Análise dos requisitos funcionais para identificar "unidades funcionais" que podem ser testadas separadamente	As unidades funcionais são os PLUCs	As unidades funcionais são os SLICES
Passo 02	Para cada unidade funcional, os testadores identificam as categorias: a) condições do ambiente (propriedades do sistema necessárias para uma certa unidade funcional) e b) os parâmetros (entradas explícitas para a unidade) que são relevantes testar.	Como nos casos de uso são definidos os cenários, no PLUTO os autores sugerem que os cenários sejam uma das categorias	Além de também sugerir os cenários como uma categoria, os pontos de variação também devem ser definidos como categorias
Passo 03	Definição das *escolhas* : valores significantes (do ponto de vista do testador) para cada categoria. As escolhas podem ser anotadas com restrições para evitar combinações contraditórias de escolhas ou contruções de testes não significantes	As tags de variabilidade são associadas as *escolhas* de forma a indicar para qual produto aquela escolha é válida, pois algumas *escolhas* dependem do produto que está sendo testado	As escolhas são associadas a restrições de contexto
Passo 04	Determinar as restrições entre as *escolhas* de diferentes categorias (exemplo: uma *escolha* numa dada categoria pode requerer uma determinada *escolha* em outra categoria)	Idem	Idem
Passo 05	Os cenários de teste são gerados combinando todas as possíveis *escolhas* para todas as categorias, respeitando as restrições definidas	Idem	Idem

4.3.1 ChAPTER

A Figura 4.10 apresenta um diagrama de atividades do método proposto, que tem três etapas e gera cenários de testes em três níveis. A primeira etapa inicia com a modelagem dos SLICES (*Software product LIne Contextual use casE*). Esses SLICES são casos de uso que descrevem além de funcionalidades, informações de variabilidade e de contexto.

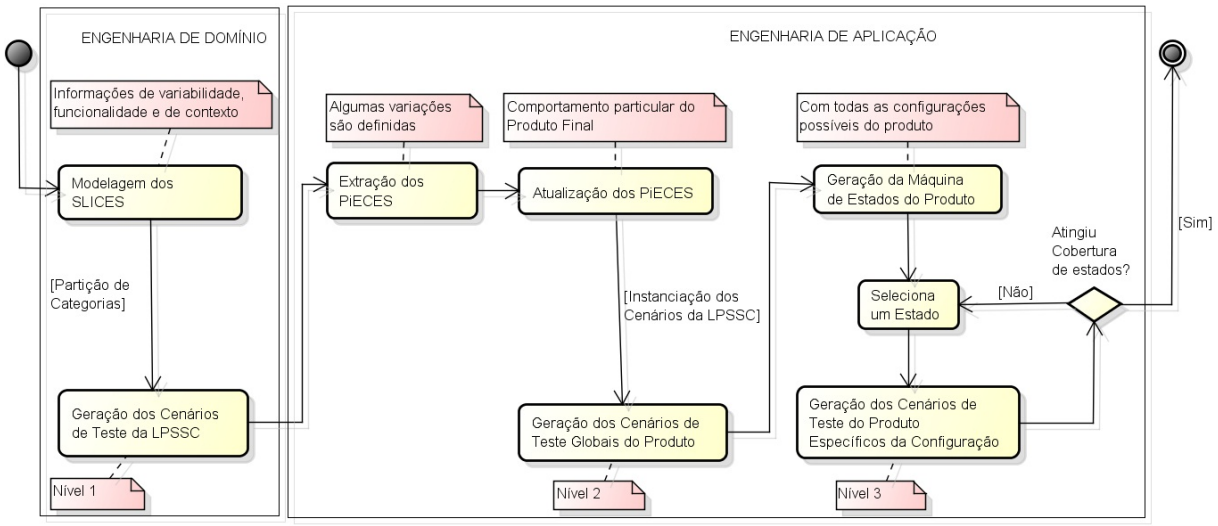


Figura 4.10: Proposta do ChAPTER.

Após a especificação dos SLICES, eles podem ser mapeados com as *features* de forma que uma *feature* pode ser um conjunto de SLICES, um único SLICE ou um ponto de variação dentro do SLICE. Ainda na Engenharia de Domínio, a partir dos SLICES da LPSSC são gerados os cenários de teste da linha do Nível 1, conforme uma adaptação do método de Partição de Categorias (ver Tabela 4.1). Tais cenários de testes englobam todas as configurações de todos os produtos da LPSSC em questão.

Visto que o número de cenários do Nível 1 explode exponencialmente com o número de pontos de variação, o recomendado é utilizar esses cenários para guiar o planejamento dos testes na Engenharia da Aplicação ou para implementar os cenários comuns visando a reutilização dos mesmos durante os testes dos produtos finais.

A segunda etapa do método inicia na Engenharia de Aplicação quando PiECES (***P**roduct contextual us**E** **C**as**E**s*) são gerados a partir dos SLICES por meio da definição dos pontos de variação. Isso significa, por exemplo, selecionar uma *feature* alternativa dentro das possibilidades de um ponto de variação alternativo. Ressalta-se que os PiECES seguem o mesmo template dos SLICES e que ao definir os PiECES o engenheiro da linha está especificando uma configuração do produto final.

Em seguida, uma vez que o contexto pode implicar em comportamentos bem específicos para a aplicação final, é previsto no método uma etapa de atualização dos PiECES na qual tais informações são especificadas. Logo, variações comportamentais que dependem das mudanças de contexto são especificadas nos fluxos alternativos ou como fluxos básicos de acordo com a aplicação em desenvolvimento.

Com os PiECES atualizados, é feito um recorte no conjunto de cenários de testes gerados no Nível 1 para extrair os cenários do produto relativos aos PiECES que somente tiveram variações definidas. Para os PiECES que foram atualizados com informações específicas do produto final, novos cenários de testes são gerados. Como resultado, nessa etapa obtém-se os cenários de testes globais do produto (Nível 2). Tais testes são chamados de "globais" porque incluem testes para todas as configurações possíveis de um mesmo produto. Nesse ponto, é interessante implementar os cenários de testes comuns, que não variam conforme a configuração da aplicação, para reutilizá-los durante o teste das configurações do produto.

Por fim, a última etapa corresponde a geração de uma máquina de estados para o produto, onde cada estado representa quais *features* estão ativas naquela configuração e a transição entre estados é feita por meio de mudanças de contexto. Dessa forma, essa máquina apresenta todas as configurações possíveis do produto. Em seguida, escolhe-se um desses estados e com base nas *features* ativas é feito um recorte do conjunto de testes globais do Produto dando origem aos testes do produto específicos da configuração (Nível 3). Logo, cada configuração passa a ter um conjunto de cenários de testes associado. Esse processo de selecionar um estado e realizar um recorte no conjunto de testes do Nível 2 prossegue até se alcançar a cobertura de estados desejada.

Destaca-se que esses testes específicos da configuração são interessantes para verificar se o produto se configura corretamente de acordo com as mudanças de contexto. Isso é possível pois a máquina de estados possui o registro de qual contexto é necessário para ativar as configurações do produto e como o contexto está especificado nos casos de uso pode-se identificar quais casos de uso, e por consequência quais cenários de testes, devem estar ativos em uma dada configuração do produto. Além disso, os cenários de teste do produto específicos da configuração também podem ser utilizados para garantir que as funcionalidades que funcionam isoladamente também funcionam no produto configurado.

Também é importante mencionar que como o método utiliza casos de uso que descrevem o comportamento do sistema (do ponto de vista do usuário), os cenários de teste gerados podem ser utilizados para realização de testes funcionais ou de aceitação. Além disso, o ChAPTER atua tanto na Engenharia de Domínio quanto na Engenharia de Aplicação, o que favorece o reuso dos artefatos de teste, pois os testes gerados na Engenharia de Domínio são refinados na Engenharia de Aplicação. Esse reuso dos artefatos de testes é importante porque possibilita a redução dos custos dos testes de uma Linhas de Produto (NETO et al., 2011).

Como principais contribuições esperadas com este método estão: a) Suporte ao teste de LPSSC; b) Apoio ao planejamento dos testes desde os requisitos; c) Redução dos custos dos testes, devido ao reuso dos artefatos de teste; e d) Meios para rastreabilidade entre requisitos, contexto, código, *features* e testes. Com relação à última contribuição, isso acontece porque o contexto vai estar modelado nos casos de uso e com a geração dos testes a partir destes casos de uso, tem-se uma rastreabilidade entre requisitos, contexto e testes. A partir desse ponto, com o mapeamento entre *features* e casos de uso e a execução dos testes, o que vai permitir identificar qual código executa qual caso de uso, é possível ter uma rastreabilidade entre esses cinco elementos.

4.3.2 Exemplo de uso do ChAPTER

Para ilustrar como funciona o método, esta seção apresenta detalhes de como funciona o ChAPTER a partir do caso de uso "Captura Contexto", que foi descrito na Seção 4.1. A diferença é que nesse exemplo vamos considerar que esse caso de uso também apresenta um fluxo alternativo, conforme apresentado na Figura 4.11.

Elemento do Caso de Uso		**Descrição**	
Nome		Captura de Contexto	
Caso de Uso Estendido		--	
Ponto de Extensão		--	
Categoria de Reuso		Obrigatório	
Restrição de Contexto		--	
Resumo		Captura informações de contexto	
Atores		Gerenciador de Contexto (secundário)	
Pré-condição		O Usuário deve estar autenticado no sistema	
Pós-Condição		Informações de contexto foram capturadas	
Passo		**Ações do Usuário**	**Ações do Sistema**
	1	--	O Sistema solicita para o Gerenciador de Contexto alguma informação de contexto informando o tipo de informação que ele precisa
[Origem] Alt [Sensor]	**2***	--	O Sensor retorna a informação de contexto
[Origem] Alt [Memória]	**2**	--	A Memória retorna a informação de contexto
[Origem] Alt [Qr_code]	**2***	--	O leitor de QR Code retorna a informação de contexto
[Origem] Alt [Serviço Externo]	**2**	--	O Serviço Externo retorna a informação de contexto
Fluxos Alternativos			
Fluxo Alternativo 01 (nenhuma meio disponível)		2. O Sistema não retorna a informação de contexto solicitada porque nenhum meio para isso está disponível 2.1 O Sistema exibe para o Usuário uma mensagem indicando o problema	
Sumário de Variações Alternativas			
[Origem]: "Qual meio utilizado para captura de contexto?"		**[Sensor]** (Restrição de Contexto: SENSOR AND SENSORES_AMBIENTE)	Passo 2
		[Memória] (Restrição de Contexto:)	Passo 2
		[Qr_code] (Restrição de Contexto: NOT_SENSOR OR NOT_SENSORES_AMBIENTE)	Passo 2
		[Serviço Externo] (Restrição de Contexto:)	Passo 2

Figura 4.11: Caso de uso Captura de Contexto modelado no CAPLUC.

Como pode ser observado na Figura 4.11, o caso de uso "Captura Contexto" foi modelado no template de um SLICE porque apresenta suporte para modelagem de funcionalidade, variabilidade e de informações de contexto.

Com o SLICE, o próximo passo do método consiste na geração dos cenários de testes da LPSSC (Nível 1) usando uma adaptação do método de Partição de Categorias apresentado na Tabela 4.1. Conforme estabelecido no ChAPTER, os cenários e pontos de variação são colocados como categorias, logo para a primeira categoria temos duas escolhas, o Cenário Principal e o Fluxo Alternativo 01, enquanto que para a segunda categoria existem quatro variantes:

Sensor, Memória, Qr_Code e Serviço Externo.

Outras categorias também podem ser incluídas para os testes. Supondo que o testador da linha considere o tipo de informação de contexto como relevante para o teste do SLICE "Captura Contexto", ele define informação de contexto como uma categoria. Nesse caso, as escolhas dessa categoria poderiam ser informações de localização do visitante ou informações do celular. Como resultado, tem-se três categorias no total, ilustradas na Figura 4.12 (a categoria supostamente definida pelo testador está identificada com o símbolo *).

Especificação de Teste do SLICE Captura Contexto
Cenários Principal Alternativo 01
Origem Sensor (SENSOR AND SENSORES_AMBIENTE) Memória Qr_code (NOT_SENSOR OR NOT_SENSORES_AMBIENTE) Serviço externo
Informação de Contexto* Localização do visitante Informações do celular (**SE** Memória)

Figura 4.12: Categorias e escolhas do exemplo.

Observa-se na Figura 4.12 que as restrições de contexto são descritas entre parênteses do lado das variantes que elas afetam. Além disso, no exemplo em questão, uma restrição foi associada entre as escolhas "Informações do celular" e "Memória", indicando que a primeira só pode ser feita se a segunda tiver sido escolhida. Esse tipo de restrição entre escolhas vem do método de Partição de Categorias. O objetivo de inserir essas restrições é evitar a geração de estados inconsistentes. No caso de uso "Captura de Contexto", por exemplo, um cenário de teste com "Informações do celular" e uma variante diferente de "Memória" não é um cenário válido da aplicação porque no exemplo esse tipo de informações é capturado pela aplicação e fica armazenado na memória.

Com a especificação dos testes (definição de categorias e escolhas) e com todas as combinações possíveis de escolhas, tem-se uma lista de todos os potenciais casos de testes para todos os produtos da linha relativo ao SLICE em consideração. No exemplo em questão, com a especificação de teste apresentada na Figura 4.12 é possível gerar 10 cenários de testes, considerando a restrição de que as "Informações do celular" só podem ser obtidas se a variante "Memória" estiver selecionada.

A Figura 4.13 apresenta os cenários de teste do Nível 1, de acordo com o ChAPTER. Vale ressaltar que como ocorre uma explosão de cenários no nível da linha, não é viável testar os cenários, mesmo porque os produtos finais ainda não estão executando nesse momento. Contudo, a partir desses cenários da linha podem ser implementados os procedimentos e casos de testes passíveis de serem reutilizados durante os testes dos produtos finais da linha. Além disso, com essa geração dos cenários de teste para a linha é possível analisar, dentre outras

coisas, o impacto de cada variação na atividade de testes.

CT 01 **Contexto:** SENSOR AND SENSORES_AMBIENTE **Cenário:** Principal **Origem:** Sensor **Informação de Contexto:** Localização do visitante	**CT 02** **Contexto:** NOT_SENSOR OR NOT_SENSORES_AMBIENTE **Cenário:** Principal **Origem:** Qr_code **Informação de Contexto:** Localização do visitante
CT 03 **Contexto:** **Cenário:** Principal **Origem:** Memória **Informação de Contexto:** Localização do visitante	**CT 04** **Contexto:** **Cenário:** Principal **Origem:** Serviço Externo **Informação de Contexto:** Localização do visitante
CT 05 **Contexto:** **Cenário:** Principal **Origem:** Memória **Informação de Contexto:** Informações do celular	**CT 06** **Contexto:** **Cenário:** Alternativo 01 **Origem:** Memória **Informação de Contexto:** Informações do celular
CT 07 **Contexto:** SENSOR AND SENSORES_AMBIENTE **Cenário:** Alternativo 01 **Origem:** Sensor **Informação de Contexto:** Localização do visitante	**CT 08** **Contexto:** NOT_SENSOR OR NOT_SENSORES_AMBIENTE **Cenário:** Alternativo 01 **Origem:** Qr_code **Informação de Contexto:** Localização do visitante
CT 09 **Contexto:** **Cenário:** Alternativo 01 **Origem:** Memória **Informação de Contexto:** Localização do visitante	**CT 10** **Contexto:** **Cenário:** Alternativo 01 **Origem:** Serviço Externo **Informação de Contexto:** Localização do visitante

Figura 4.13: Exemplo de cenários de testes do Nível 1.

A etapa seguinte é de instanciação dos PiECES a partir dos SLICES. Esse processo representa definir algumas variações e sair do nível da linha para uma configuração inicial do produto final. No exemplo dessa seção, o ponto de variação “Origem” tem quatro opções, as quais uma ou mais podem ser excluídas em tempo de projeto do produto final. Considerando que um Produto A, por exemplo, apresenta somente as *features* alternativas “Memória”, “Sensor” e “Qr_Code”, o resultado seria o SLICE “Captura Contexto”, retirando-se os trechos relacionados as *features* não selecionadas para o produto, ou seja, a “Serviço Externo”. A Figura 4.14 exibe o PiECE “Captura Contexto” do Produto A.

Logo, no caso do Produto A, a *feature* “Memória” está sempre disponível, mas as *features* “Sensor” e “Qr_Code” são selecionadas de acordo com o contexto corrente. Se tiver sensor no celular e sensores no ambiente, a *feature* “Sensor” será ativada para capturar a localização do visitante por meio dos sensores, caso contrário a *feature* “Qr_code” é ativada e a captura da posição do visitante é feita por meio da leitura dos Qr codes presentes no ambiente. Com isso, essa seleção dinâmica torna o Produto A sensível ao contexto, pois ele se reconfigura com base nas informações de contexto.

O próximo passo é de atualização dos PiECES gerados. Essa atualização refere-se ao acréscimo nos PiECEs do comportamento específico dos produto gerados. Um Produto B, por exemplo, poderia exibir a mensagem “Bom dia” se o visitante estiver usando a aplicação entre 06h e 12h e “Boa tarde” se estiver entre 12h e 18h. Esse comportamento pode então ser adicionado em algum PiECE desse Produto B e com essas novas informações outros cenários são gerados.

Voltando ao Produto A, supondo que não há nenhum comportamento adicional a ser acrescentado ao SLICE gerado, é possível então obter os cenários de testes globais do Produto A (Nível 2) por meio de um recorte dos cenários de testes da linha. Nesse caso, como a escolha

Elemento do Caso de Uso	**Descrição**	
Nome	Captura de Contexto	
Caso de Uso Estendido	--	
Ponto de Extensão	--	
Categoria de Reuso	Obrigatório	
Restrição de Contexto	--	
Resumo	Captura informações de contexto	
Atores	Gerenciador de Contexto (secundário)	
Pré-condição	O Usuário deve estar autenticado no sistema	
Pós-Condição	Informações de contexto foram capturadas	
Passo	**Ações do Usuário**	**Ações do Sistema**
1	--	O Sistema solicita para o Gerenciador de Contexto alguma informação de contexto informando o tipo de informação que ele precisa
[Origem] Alt [Sensor] **2**	--	O Sensor retorna a informação de contexto
[Origem] Alt [Memória] **2**	--	A Memória retorna a informação de contexto
[Origem] Alt [Qr_code] **2**	--	O leitor de QR Code retorna a informação de contexto
Fluxos Alternativos		
Fluxo Alternativo 01 (nenhuma meio disponível)	2. O Sistema não retorna a informação de contexto solicitada porque nenhum meio para isso está disponível 2.1 O Sistema exibe para o Usuário uma mensagem indicando o problema	
Sumário de Variações Alternativas		
[Origem]: "Qual meio utilizado para captura de contexto?"	**[Sensor]** (Restrição de Contexto: SENSOR AND SENSORES_AMBIENTE)	Passo 2
	[Memória] (Restrição de Contexto:)	Passo 2
	[Qr_code] (Restrição de Contexto: NOT_SENSOR OR NOT_SENSORES_AMBIENTE)	Passo 2
Sumário de Variações Opcionais		

Figura 4.14: Exemplo de PiECE gerado a partir do SLICE Captura de Contexto.

"Serviço Externo" não está disponível no Produto A, os cenários de testes da linha (ver Figura 4.13) que apresentam essa variante são descartados. Com isso, tem-se oito cenários de teste globais do Produto A, que são apresentados na Figura 4.15

CT 01 **Contexto:** SENSOR AND SENSORES_AMBIENTE **Cenário:** Principal **Origem:** Sensor **Informação de Contexto:** Localização do visitante	**CT 02** **Contexto:** NOT_SENSOR OR NOT_SENSORES_AMBIENTE **Cenário:** Principal **Origem:** Qr_code **Informação de Contexto:** Localização do visitante
CT 03 **Contexto:** **Cenário:** Principal **Origem:** Memória **Informação de Contexto:** Localização do visitante	**CT 04** **Contexto:** **Cenário:** Principal **Origem:** Memória **Informação de Contexto:** Informações do celular
CT 05 **Contexto:** **Cenário:** Alternativo 01 **Origem:** Memória **Informação de Contexto:** Localização do visitante	**CT 06** **Contexto:** **Cenário:** Alternativo 01 **Origem:** Memória **Informação de Contexto:** Informações do celular
CT 07 **Contexto:** SENSOR AND SENSORES_AMBIENTE **Cenário:** Alternativo 01 **Origem:** Sensor **Informação de Contexto:** Localização do visitante	**CT 08** **Contexto:** NOT_SENSOR OR NOT_SENSORES_AMBIENTE **Cenário:** Alternativo 01 **Origem:** Qr_code **Informação de Contexto:** Localização do visitante

Figura 4.15: Cenários de teste globais do Produto A (Nível 2).

Esses cenários globais estão relacionados a todas as configurações do Produto A com relação ao caso de uso "Captura de Contexto". Nesta etapa, como o produto final já está definido ele pode ser testado utilizando os cenários como teste de sistema ou de aceitação. Contudo, como esse produto sofre reconfigurações com as mudanças de contexto, não é possível

testar todos os cenários de teste em uma única configuração da aplicação e por isso existe a necessidade da terceira etapa do ChAPTER.

A última etapa do método inicia com a geração da máquina de estados finita do produto, onde cada estado representa um conjunto de *features* ativas e as transições entre os estados são mudanças de contexto. Com isso, essa máquina de estados apresenta todas as configurações possíveis do produto. A Figura 4.16 apresenta a máquina de estado no exemplo em questão.

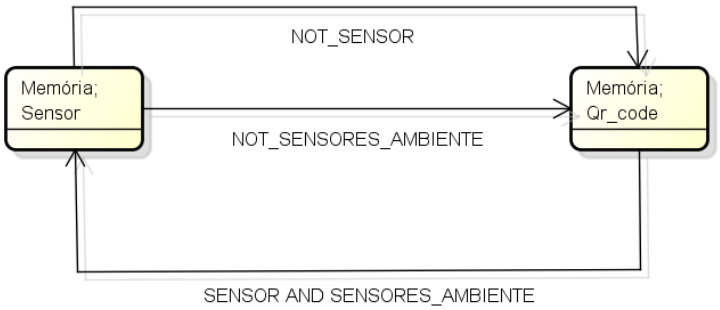


Figura 4.16: Máquina de Estados do Produto A.

Com a máquina de estados, escolhe-se um estado e são selecionados a partir dos cenários globais do produto, os cenário de teste do Produto Específicos da Configuração. Ao escolher o estado "Memória e Sensor", por exemplo, são selecionados seis cenários do conjunto de cenários de testes globais do produto que envolvem a *feature* "Memória" ou a *feature* "Sensor". A Figura 4.17 apresenta os conjuntos de cenários de testes para cada estado da Figura 4.16, destacando os cenários que são diferentes entre os dois estados.

Estado "Memória; Sensor"	**Estado "Memória, Qr_code"**
CT 01 **Contexto:** SENSOR AND SENSORES_AMBIENTE **Cenário:** Principal **Origem:** Sensor **Informação de Contexto:** Localização do visitante	**CT 01** **Contexto:** NOT_SENSOR OR NOT_SENSORES_AMBIENTE **Cenário:** Principal **Origem:** Qr_code **Informação de Contexto:** Localização do visitante
CT 02 **Contexto:** **Cenário:** Principal **Origem:** Memória **Informação de Contexto:** Localização do visitante	**CT 02** **Contexto:** **Cenário:** Principal **Origem:** Memória **Informação de Contexto:** Localização do visitante
CT 03 **Contexto:** **Cenário:** Principal **Origem:** Memória **Informação de Contexto:** Informações do celular	**CT 03** **Contexto:** **Cenário:** Principal **Origem:** Memória **Informação de Contexto:** Informações do celular
CT 04 **Contexto:** SENSOR AND SENSORES_AMBIENTE **Cenário:** Alternativo 01 **Origem:** Sensor **Informação de Contexto:** Localização do visitante	**CT 04** **Contexto:** NOT_SENSOR OR NOT_SENSORES_AMBIENTE **Cenário:** Alternativo 01 **Origem:** Qr_code **Informação de Contexto:** Localização do visitante
CT 05 **Contexto:** **Cenário:** Alternativo 01 **Origem:** Memória **Informação de Contexto:** Localização do visitante	**CT 05** **Contexto:** **Cenário:** Alternativo 01 **Origem:** Memória **Informação de Contexto:** Localização do visitante
CT 06 **Contexto:** **Cenário:** Alternativo 01 **Origem:** Memória **Informação de Contexto:** Informações do celular	**CT 06** **Contexto:** **Cenário:** Alternativo 01 **Origem:** Memória **Informação de Contexto:** Informações do celular

Figura 4.17: Cenários de teste para cada estado do Produto A.

Esses conjuntos de cenários de testes podem ser usados para verificar se as *features* realmente estão ativas como deveriam e se funcionam como deveriam. Por fim, esse processo

de escolher um estado e selecionar os cenário é feito até se atingir a cobertura de cenários ou de mudança de contexto desejada.

4.4 Ferramenta

A terceira contribuição deste trabalho diz respeito a uma ferramenta de apoio ao uso do template e do método que foram apresentados nas Seções 4.1 e 4.2, respectivamente. A ferramenta denominada de ToChAPTER, como referência ao método ChAPTER apresentado na Seção 4.2, foi implementada como uma aplicação web.

A ToChAPTER foi implementada utilizando a IDE Eclipse[2] e com o apoio de alguns *frameworks*: a) Hibernate[3], para a persistência dos dados; b) VRaptor[4], para a implementação da aplicação web com a linguagem de programação Java[5]; e c) SNACKS[6], para geração da máquina de estados correspondente a terceira etapa do método ChAPTER. Já com respeito a persistência dos dados, essa é feita utilizando um banco de dados MySQL[7]. Nas subseções a seguir são apresentados detalhes sobre a arquitetura, o funcionamento e as limitações da ferramenta desenvolvida.

4.4.1 Visão Geral

A Figura 4.18 apresenta um diagrama de pacotes UML com uma visão estática da arquitetura da ferramenta ToChAPTER. Essa visão permite destacar os principais módulos funcionais da ferramenta, que no caso são o módulo de controladores, o módulo de geração de testes e o módulo de persistência dos dados.

O módulo de controladores contém as classes responsáveis pelo controle da lógica da ferramenta. Nesse módulo estão presentes as classes responsáveis por receber os dados fornecidos pelo usuário bem como prover conteúdo ao usuário. Além disso, são os controladores que invocam, mediante as ações do usuário, as funcionalidades dos módulos de geração de testes e persistência dos dados.

O módulo de persistência de dados, por sua vez, tem por objetivo gravar os dados da ferramenta no banco de dados. Para isso as entidades da ferramenta são modeladas com anotações do *framework* hibernate, que permite persistir objetos Java em um banco de dados relacional. As entidades implementadas na ferramenta, bem como o relacionamento entre elas é apresentado no diagrama de classes da Figura 4.19.

Conforme observado na Figura 4.19, a ferramenta tem oito entidades: *SPL, Product, Slice, Piece, Step, AlternativeFlow, Variant e VariantPoint*. Na ferramenta, uma linha de produto

[2]http://www.eclipse.org/

[3]http://www.hibernate.org/

[4]http://vraptor.caelum.com.br/pt/

[5]http://www.java.com

[6]A Tool for Specification of Normal and Abnormal Context-Aware Behavior as Kripke Structures. Resultado parcial do trabalho de doutorado em andamento de Lincoln Rocha, do MDCC-UFC

[7]http://dev.mysql.com/downloads/

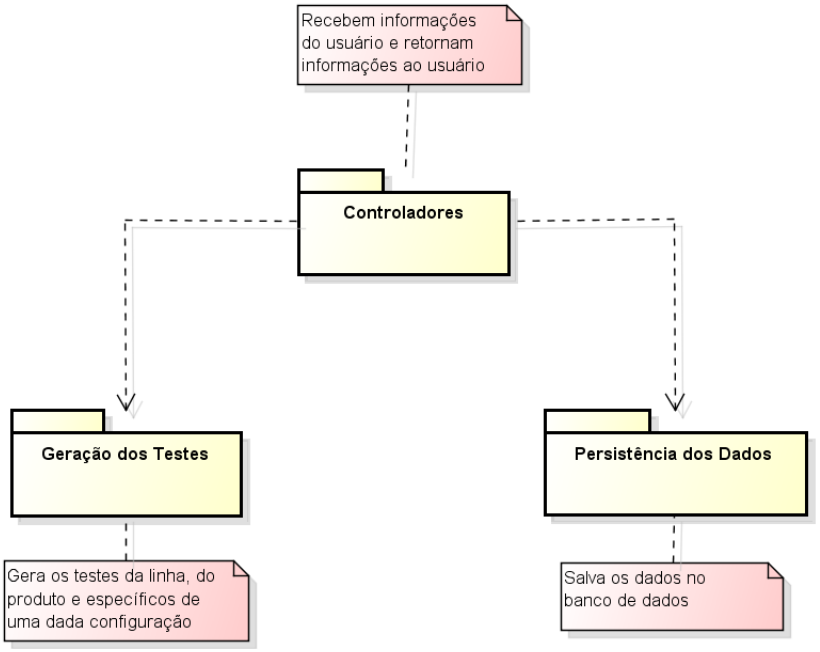


Figura 4.18: Visão estática da ferramenta ToChAPTER.

de software (SPL) está relacionado a vários SLICEs, que por sua vez possui vários passos (Steps). Cada Step não tem ou tem vários passos alternativos (AlternativeFlow) e pode estar ou não relacionado a uma variante (Variant). Já essas variantes estão associadas a um ponto de variação (VariantPoint).

Ressalta-se que os pontos de variação estão relacionados a uma linha de produto para que seja possível reutilizar os pontos de variação em vários casos de uso. Por fim, uma linha tem vários produtos (Product) que tem vários PiECEs. Os PiECEs representam casos de uso do produto e, portanto, também tem vários passos (Step) e estão ligados a um caso de uso da linha (SLICE). Isso porque, segundo o ChAPTER, os PiECES são gerados a partir dos SLICES definindo, por exemplo, quais variantes estão presentes no produto.

O terceiro módulo funcional da ferramenta é o de geração dos testes, o qual implementa a geração dos cenários de testes nas três etapas previstas no método ChAPTER. Com relação aos cenários de testes (ScenarioTest), estes são implementados na ferramenta como um conjunto de opções (Option), onde cada opção está associado a uma categoria (Category). O uso de opções e categorias está relacionada ao método de geração de testes utilizado que no caso é o de Partição de Categorias. Essa relação entre cenários de testes, opções e categorias é apresentado na Figura 4.20.

Uma visão dinâmica da ToChAPTER, por meio de um diagrama de sequência, é apresentado na Figura 4.21. Essa visão dinâmica exibe a interação dinâmica do módulos funcionais da ferramenta. Conforme sintetizado nessa figura, os Controladores gerenciam a lógica da aplicação invocando funcionalidades do Gerador de Cenários de Testes, que retorna cenários de testes, e do Gerenciador de Dados, que é responsável por persistir os casos de uso no banco de dados.

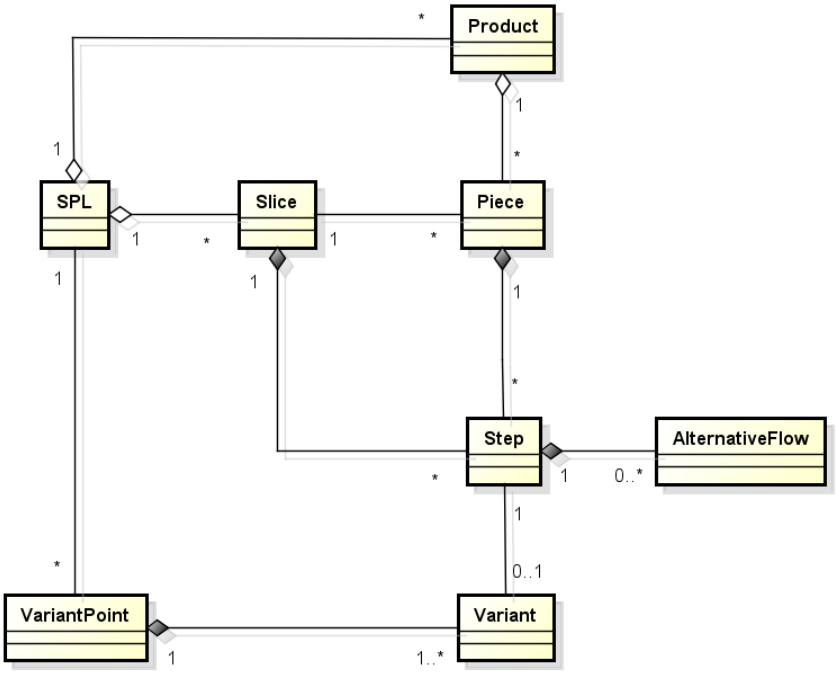


Figura 4.19: Relacionamento entre as entidades da ToChAPTER.

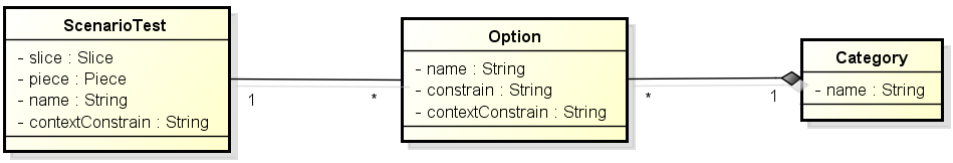


Figura 4.20: Relacionamento entre Cenários de testes, opções e categorias na ToChAPTER.

4.4.2 Funcionamento da ferramenta

Nesta subseção são apresentados detalhes do funcionamento da ToChAPTER, bem como as principais interfaces de usuário. Como a ToChAPTER implementa o método de geração de testes ChAPTER, a ferramenta apresenta suporte para as três etapas desse método: i) modelagem dos SLICES e geração de testes da linha; ii) criação dos PiECES e geração dos testes dos produtos; e iii) atualização dos PiECES e geração dos testes para as configurações do produto final.

A Figura 4.22 apresenta a tela inicial da ferramenta desenvolvida. Como pode ser observado nessa figura, o *menu* da ferramenta (1) só tem três opções: i) *Início*, que leva a tela inicial; ii) *SLICE*, onde os SLICEs podem ser modelados e os testes para a linha podem ser gerados; e iii) *PiECE*, o qual redireciona a tela de modelagem dos PiECES e a geração dos testes do produto.

Para a modelagem dos SLICES, é preciso clicar na opção *SLICE* no menu da ferramenta (elemento 1 da Figura 4.22). Após essa ação será exibido ao usuário uma lista das linhas de produto cadastradas nessa ferramenta com um *link* para a lista de SLICES de cada linha. Ao acessar a lista de SLICES de uma linha de produto, como na Figura 4.23, é possível, então, criar

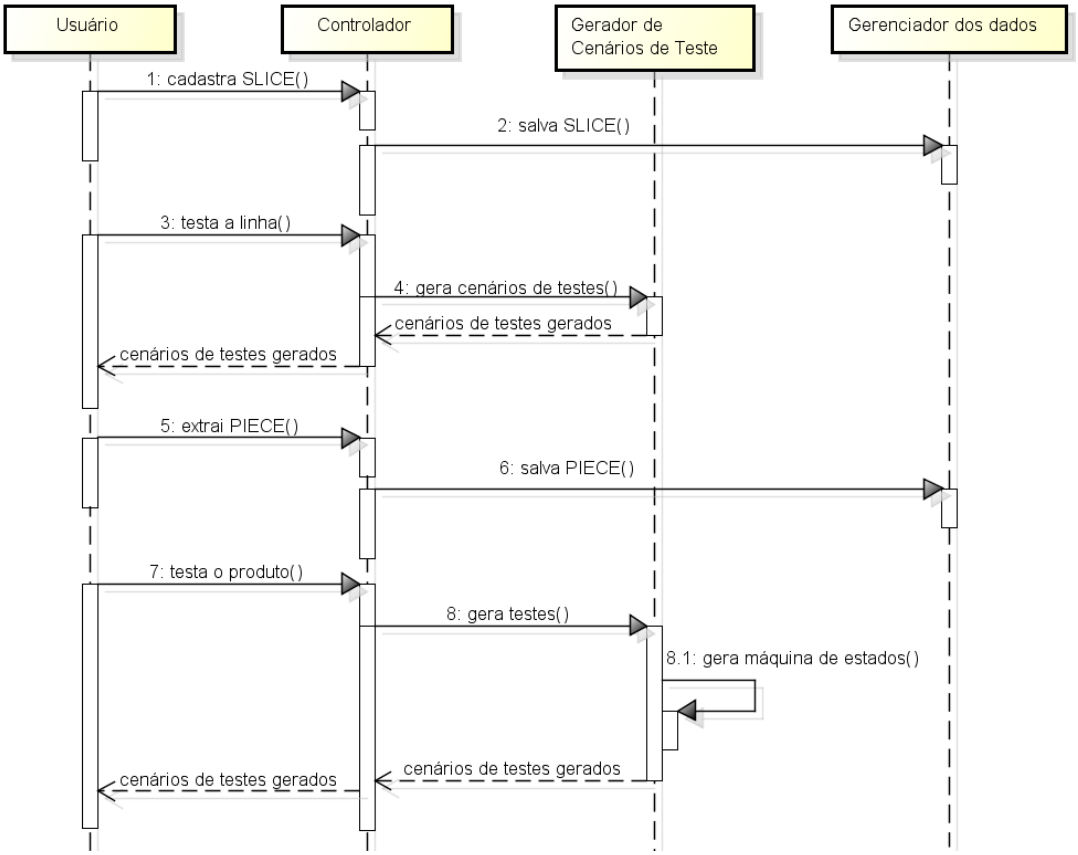


Figura 4.21: Visão dinâmica da arquitetura do ToChAPTER.

novos SLICES (2) ou testar todos os SLICES (3). Além disso, para cada SLICE, há a opção para editar, remover, ver ou testar (1).

Ao decidir por criar um novo SLICE, o usuário deve primeiro fornecer os dados referentes a alguns elementos desse caso de uso, que foram descritos em detalhes na Seção 4.1, como por exemplo, um identificador, um resumo, os atores, pré e pós-condição. Nesse ponto, uma restrição de contexto pode ser adicionada para todo o SLICE, indicando que esse SLICE só pode ser executável se esta restrição for verdadeira.

A adição dos passos do SLICE é a última etapa a ser feita na modelagem do SLICE com a ferramenta. A Figura 4.24 ilustra a interface de edição dos passos, já com alguns passos cadastrados para o SLICE "Acesso ao Contexto" (1). Cada passo do SLICE é constituído por um índice, uma ação do usuário, uma ação do sistema e pode ou não estar associado a alguma variação (2), conforme o template CAPLUC (Capítulo 04). O primeiro passo de índice 02 da Figura 4.24, por exemplo, está ligada a variante *Síncrona* do ponto de variação alternativo *Acesso*. Vale notar que como *Síncrono* e *Assíncrono* são variantes alternativas, os passos associadas a elas estão com o mesmo índice seguindo as especificação do template CAPLUC. Para adicionar novos passos, basta clicar no botão *Novo* (4).

Ainda na Figura 4.24 é possível ver que cada passo além de ser passível de edição e remoção, também pode ser associado a um fluxo alternativo ou a alguma variação (3). Ao clicar em *Fluxos Alternativos* ou *Variação* o *frame* inferior (7) carrega tela e opções apropriadas.

Figura 4.22: Tela Inicial da ferramenta ToChAPTER.

Figura 4.23: Lista de Slices de uma Linha de Produto.

No caso de um fluxo alternativo, ele apresenta um índice, uma restrição que deve ser verdadeira para o fluxo iniciar, uma ação do usuário e uma ação do sistema. Já com respeito a associação do passo com uma variação, primeiro deve-se selecionar um ponto de variação, que é constituído por um nome, o tipo da variação (obrigatório, opcional ou alternativo) e uma questão descritiva. Em seguida, após selecionado o ponto de variação, variantes podem ser criadas com um nome e uma restrição de contexto (opcional) e com isso é possível relacionar uma variante a um passo por meio da opção *Relacionar* presente para cada variante cadastrada. Uma vez relacionada variações aos passos, a lista de passos pode ser atualizada (5) e após adicionado todos os passos, a especificação do SLICE pode ser concluida (6).

Com os SLICES modelados é possível gerar os testes da linha. Nesse ponto, pode-se escolher gerar testes para apenas um SLICE (ver elemento 1 da Figura 4.23) ou gerar testes para todos os SLICES da linha (ver elemento 3 da Figura 4.23). Ao solicitar a geração de testes de um SLICE é possível fazer pequenas configurações no processo de geração dos testes.

A Figura 4.25 ilustra a tela de configuração do mecanismo de geração de testes. No início da página é exibido as categorias que foram extraídas do SLICE (1). No caso dessa figura, existem duas categorias: a) *Acesso*, que é um ponto de variação com duas variantes, Síncrono e Assíncrono; e b) *Cenarios*, que está relacionado aos cenários do caso de uso e que

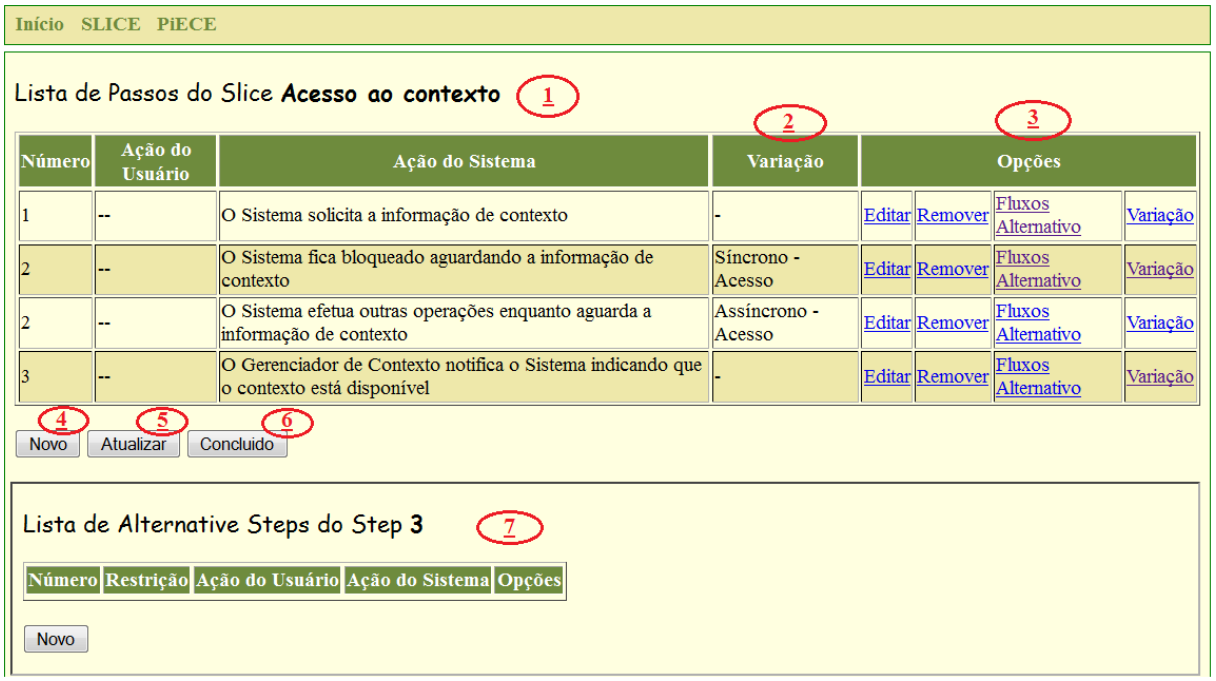


Figura 4.24: Passos de um SLICE na ToChAPTER.

no caso são dois, o principal e o alternativo. Ainda nessa página, o usuário pode optar por incluir novas categorias (2), que ele considerar relevante ao teste. Além disso, é permitido ao usuário especificar restrições entre as categorias (3) com o intuito de evitar cenários de testes com escolhas que não representam uma configuração válida do produto.

Na ferramenta, para a extração dos PiECES o primeiro passo é a escolha do produto ao qual o PiECE pertence. Para isso, após clicar no *link* PiECE do menu principal (ver elemento 1 da Figura 4.22), o usuário seleciona uma das linhas de produto cadastradas na ferramenta, e então, a lista de produtos que já foram cadastradas para essa linha é exibida.

Na tela com a lista dos produtos é possível criar, editar ou remover um produto, bem como testar esse produto ou visualizar os PiECES desse produto. A opção para testar o produto é responsável por gerar a máquina de estados conforme o método ChAPTER. Já a opção *Pieces* ela leva o usuário a lista de PiECEs do produto em questão. Uma vez estando na tela de PiECES, o usuário tem a disposição opções para editar, remover ou testar o PiECE, que seria o equivalente a testar um SLICE no nível do produto.

Com relação a criação de um novo PiECE, o primeiro passo é identificar o SLICE base. Isso é necessário porque os PiECES são SLICES configurados ou editados para os produtos finais. Uma vez selecionado o SLICE base, o usuário pode então definir quais variantes estão presentes no produto, como ilustrado na Figura 4.27. Coletadas essas informações, a ToChAPTER pega as informações do SLICE base e copia para gerar o PiECE, removendo todos os passos que estão associados a variações que não foram selecionadas pelo usuário.

Quanto aos testes gerados, esses são criados em formatos próprios para a visualização na Fitnesse (FITNESSE, 2012). A escolha dessa ferramenta se deu porque ela é uma ferramenta de teste de aceitação gratuita bastante conhecida entre a comunidade de testado-

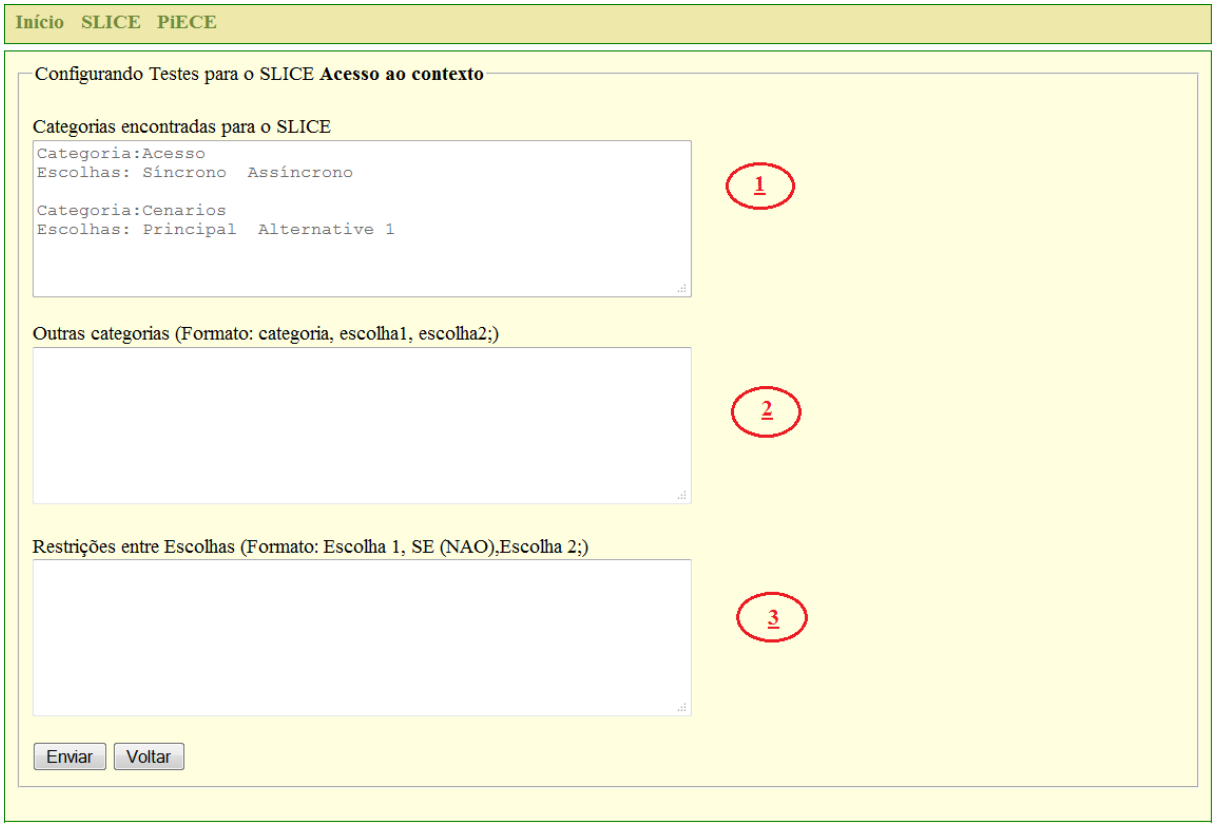


Figura 4.25: Tela de configuração dos testes.

res. Como diferenciais dessa ferramenta está o fato dela permitir a criação dos testes antes do produto ser desenvolvido e gerenciar os testes por meio de tabelas, tornando mais fácil a comunicação entre usuários finais e testadores. Dessa forma, utilizando a Fitnesse é possível focar na lógica de negócios, ao invés do código fonte.

A Figura 4.28 apresenta um exemplo de cenário de teste gerado para o PiECE "Captura Contexto". No exemplo dessa figura, o cenário indica um teste que exercite o fluxo básico do caso de uso (Main), bem como os passos relacionados a variante "Qr_code". Além disso, o cenário especifica que o contexto NOT_SENSOR AND NOT_SENSORES_AMBIENTE deve estar ativo para esse teste.

4.4.3 Limitações

Embora a ToChAPTER atenda a execução de todas as etapas do método ChAPTER, essa ferramenta possui algumas limitações conhecidas:

- A ferramenta ainda não exporta os casos de uso modelados para formatos como .Doc, .RTF ou .PDF. Isso seria interessante para permitir que os usuários possam manipular fora da ferramenta a documentação dos casos de uso da linha;
- A ferramenta não exibe ao usuário a máquina de estados gerada e nem fornece um mecanismo de busca deste estados. Como resultado, o testador não tem apoio para realizar

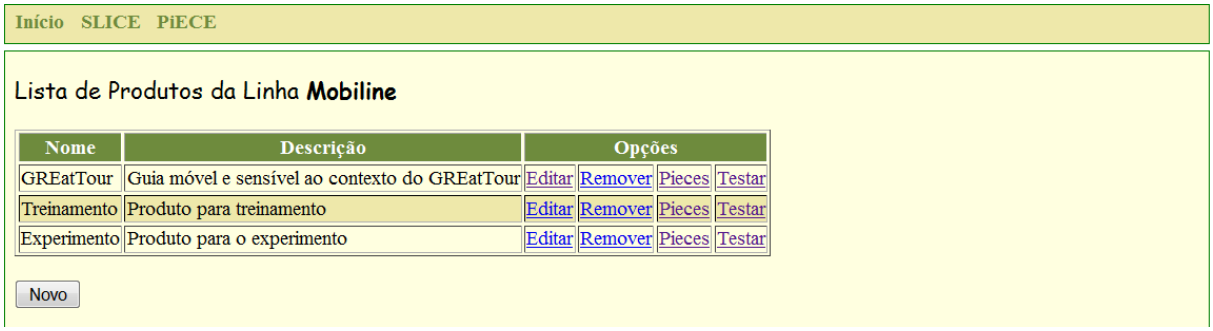


Figura 4.26: Tela de lista dos produtos

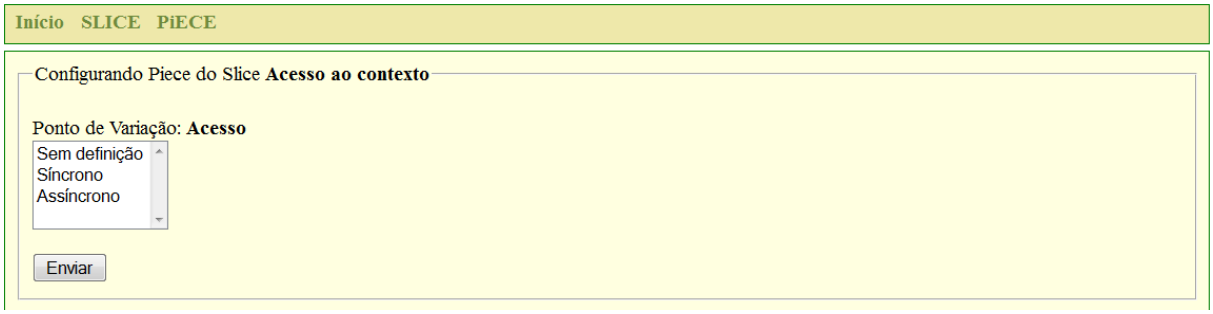


Figura 4.27: Tela de configuração dos Pieces.

uma seleção de quais cenários de testes devem ser executados;

- O relacionamento entre casos de uso, quando um caso de uso chama outro caso de uso, ainda não foi tratado na ferramenta;
- O mecanismo de caso de uso estendido da ferramenta, quando um caso de uso estende outro, ainda precisa ser melhorado tanto para facilitar o trabalho do usuário quando para possibilitar buscas com esses casos de uso que foram estendidos;
- Atualmente, a ferramenta ainda exige um pequeno esforço manual para levar os cenários de testes gerados para a Fitnesse, pois é preciso copiar os arquivos gerados com a To-ChAPTER para o diretório do Fitnesse e editar a página do Fitnesse para incluir esses testes;
- A ferramenta não está preparada para lidar com uma LPSSC de centenas de informações de variabilidade e de contexto. Isso porque ela utiliza um algoritmo de combinação de escolhas sem tratamento para escalabilidade, e com isso, uma LPSSC com centenas de *features* e informações de contexto levaria a uma explosão de cenários de testes; e
- Como a ToChAPTER só gera os cenários de testes, para executar os testes na Fitnesse ainda é preciso gerar os casos de testes e as *fixture*, que são um artefato da Fitnesse para permitir a execução dos testes na aplicação.

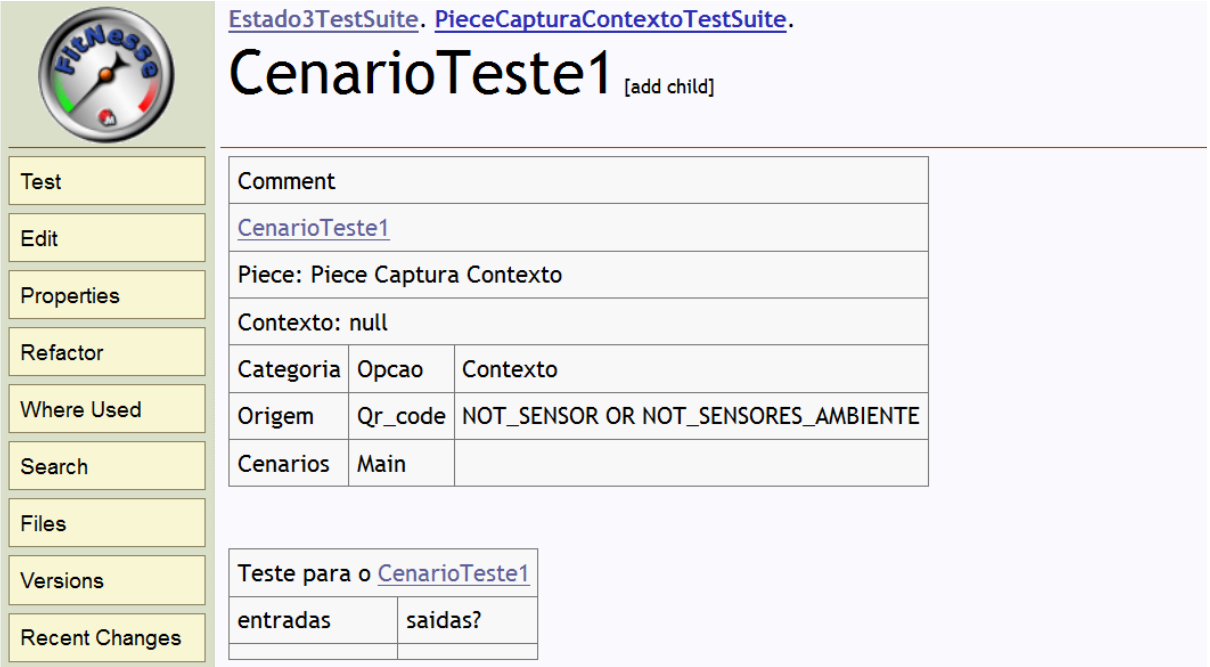


Figura 4.28: Exemplo de cenário de teste gerado pela ferramenta.

4.5 Conclusão

Neste capítulo foram apresentadas as principais contribuições deste trabalho: i) a proposta de um template para descrição textual de caso de uso para LPSSC; ii) a proposta do método de geração de testes para LPSSC; e iii) uma ferramenta de apoio ao uso do template e do método propostos.

O template proposto é denominado de CAPLUC, acrônimo de **C**ontext **A**ware software **P**roduct **L**ine **U**se **C**ase template. O diferencial desse template está no fato dele tratar de LPSSC, pois nenhum outro template para esse tipo de linha foi encontrado na literatura.

Para fundamentar e guiar a proposta do CAPLUC foi executado um experimento controlado com alguns dos templates encontrados na literatura para especificar textualmente casos de uso de LPS tradicionais, não sensíveis ao contexto. Os resultados do experimento revelaram indícios de que os templates que mais favorecem o entendimento de casos de uso de LPS são o do Erikson et al.(2004) e do Gomma (2004).

Além disso, com o experimento foi possível coletar indícios de que especificar as similaridades junto com variabilidades facilita o entendimento do caso de uso enquanto que a especificação separada de similaridades e variabilidades torna rápida a identificação das variações que afetam o caso de uso.

Logo, utilizando os resultados do experimento, o CAPLUC foi proposto incorporando as melhores características identificadas de cada template. Do template do “Eriksson et al.”, o CAPLUC extraiu a forma como representar os passos. A proposta do Gomma (2004), por sua vez, inspirou o elemento Categoria de Reuso e o elemento Sumário de Variações, que resume as variações presentes no caso de uso. Por fim, baseado no template do John e Muthing

(2002), cada variação presente no CAPLUC pode ser associada a uma questão descritiva que auxilia o entendimento da variação dentro do caso de uso e a instanciação dos produtos finais.

Para demonstrar o uso do CAPLUC, também foram apresentados neste capítulo três exemplos de casos de uso da LPSSC Mobiline, que é uma linha destinada para guias de visita móveis e sensíveis ao contexto. Finalmente, dado as características do CAPLUC, acredita-se que seu uso pode trazer os seguintes benefícios: a) Melhor entendimento sobre a influência do contexto nos produtos da linha; b) Suporte a rastreabilidade entre requisitos e contexto; e c) Suporte às abordagens de apoio ao desenvolvimento de LPSSC existentes.

Com relação ao método, ele foi denominado de ChAPTER, acrônimo de ***C****ontext* ***A****ware software* ***P****roduct line* ***TE****sting gene****R****ation method*.

O ChAPTER foi proposto com base no PLUTO (*Product Line Use Case Test Optimisation*) (BERTOLINO; GNESI, 2003) que é um método de geração de testes para linhas de produto de software. Para gerar os testes, o PLUTO utiliza casos de usos descritos textualmente no formato PLUC(FANTECHI et al., 2003) e uma adaptação do método de Partição de Categorias(OSTRAND; BALCER, 1988).

No caso do ChAPTER, esse também utiliza como entrada casos de uso descritos textualmente, mas estes além de especificar funcionalidade e informações de variabilidade também precisam descrever informações de contexto porque o ChAPTER é voltado para LPSSC, linha de aplicações sensíveis ao contexto. Um caso de uso com essas características é denominado no ChAPTER de SLICE (***S****oftware product* ***LI****ne* ***C****ontextual use cas****E***). Quando esses SLICES são adaptados para tratar dos produtos finais, tem-se os PiECES (***P****roduct contextual us****E*** ***C****as****E****s*). Dessa forma, no ChAPTER uma LPSSC é constituída por vários SLICES durante a Engenharia de Domínio, e na Engenharia da Aplicação cada SLICE pode dar origem a vários PiECES.

Baseado nos SLICES e PiECES, o ChAPTER apresenta um método de geração de testes em três etapas. Na primeira etapa, um conjunto de cenários de testes globais da linha é gerado a partir dos SLICES da LPSSC. Esses cenários são relativos a todos os produtos da linha.

Na segunda etapa, PieCES são criados a partir dos SLICES, e com eles, cenários de testes globais do produto são gerados. Esses cenários representam os testes para todas as configurações possíveis do produto a qual os PiECES pertencem.

Como os produtos sensíveis ao contexto se reconfiguram conforme informações de contexto, a terceira etapa do método consiste em gerar uma máquina de estados do produto, onde cada estado representa uma configuração possível do produto. Nesse caso, as transições são feitas por meio de mudanças de contexto. Com isso, para cada estado é associado um conjunto de cenários de testes que pode ser utilizado para testar as funcionalidades do produto ou para verificar se o produto se reconfigura como esperado.

É importante mencionar que foge do escopo do ChAPTER a definição de quais testes devem ser executados na Engenharia de Domínio ou na Engenharia de Aplicação, ou mesmo de quais testes devem ser executados em cada configuração do produto. A proposta

do ChAPTER é apoiar o gerenciamento da atividade de testes, bem como o reuso destes, pois como os cenários de testes são gerados tão logo os casos de uso estejam disponíveis, é possível implementar os testes antes mesmo dos produtos finais da linha estarem prontos.

Como principais contribuições esperadas com o ChAPTER estão: a) Suporte ao teste de LPSSC; b) Apoio ao planejamento dos testes desde os requisitos; c) Redução dos custos dos testes, devido ao reuso dos artefatos de teste; e d) Meios para rastreabilidade entre requisitos, contexto, código, *features* e testes.

Por fim, a última contribuição apresentada neste trabalho foi a implementação da ToChAPTER que é uma ferramenta web que permite a modelagem de SLICEs e PIECES seguindo o CAPLUC, além de possibilitar a geração dos testes com o ChAPTER. A arquitetura dessa ferramenta é composta basicamente por três módulos funcionais principais: dos controladores, de geração de testes e de persistência dos dados.

O módulo dos controladores é responsável por receber as informações do usuário e retornar conteúdo ao usuário por meio das páginas web. Além disso, é esse módulo que invoca funcionalidades dos outros dois módulos com base nas ações do usuário. O módulo de geração de testes cria os cenários de testes seguindo a especificação do método ChAPTER. Já o módulo de persistência dos dados é responsável por armazenar as entidades implementadas na ferramenta no banco de dados.

Além de detalhes da arquitetura, também foram apresentados neste capítulo o funcionamento e as limitações atuais da ToChAPTER. Apesar das limitações identificadas, acredita-se que essa ferramenta representa um suporte ferramental inicial para os engenheiros de uma linha de produto sensível ao contexto modelarem os casos de uso e para os testadores gerarem os cenários de testes levando em conta as mudanças de contexto.

5 AVALIAÇÃO PRELIMINAR

Neste capítulo é apresentada a avaliação preliminar conduzida para verificar se os resultados concretizados neste trabalho atingem os objetivos esperados e para averiguar a viabilidade da execução de um experimento controlado para avaliar tais resultados. Como são três os resultados deste trabalho (i. e., um template para especificação de casos de uso para LPSSC, um método de geração de testes para LPSSC e uma ferramenta de apoio) essa avaliação preliminar é particionada em três etapas: avaliação do template, avaliação do método e avaliação da ferramenta.

O restante deste capítulo está organizado da seguinte forma: a Seção 5.1 apresenta a LPSSC utilizada durante a avaliação; na Seção 5.2 é apresentado o *design* da avaliação conduzida; nas Seções 5.3, 5.4 e 5.5 são apresentados os resultados obtidos quanto a avaliação do template, do método e da ferramenta, respectivamente; na Seção 5.6 é descrita a lista de problemas identificada durante a avaliação preliminar; e por fim, na Seção 5.7 são feitas as considerações finais sobre as avaliações realizadas.

5.1 Objeto da avaliação

Para avaliar os resultados dessa pesquisa, conforme descrito em mais detalhes nas próximas subseções, utilizou-se a LPSSC Mobiline (MOBILINE, 2012). Essa linha foi desenvolvida pelo GREat[1] e é caracterizada pelo seu foco em aplicações móveis e sensíveis ao contexto.

Para construir a LPSSC Mobiline, a Engenharia de Domínio foi decomposta em dois níveis de análise, sendo que o primeiro trata das características de uma LPS móvel e sensível ao contexto, e o segundo foca no domínio de guias móveis (MARINHO et al., 2010). Para mais detalhes sobre essa LPSSC é possível consultar o site do projeto de pesquisa Mobiline[2], que deu origem a LPSSC com o mesmo nome.

Uma aplicação resultante da LPSSC Mobiline é o GreatTour, que é uma aplicação móvel e sensível ao contexto desenvolvida para guiar os visitantes do laboratório do GREat.

A Figura 5.1 ilustra o funcionamento do GREatTour (MARINHO et al., 2010). O GREatTour inicialmente solicita a autenticação do usuário. Após identificação do usuário, a aplicação começa a coletar dados de contexto e apresenta ao visitante um mapa contento sua localização atual (I). À medida que o usuário visita uma sala, o GREatTour atualiza a exibição do mapa indicando a nova posição do usuário (II). Ao chegar em uma sala, o visitante tem a opção de obter informações sobre a mesma. No caso da figura, o sistema exibe as imagens do laboratório de prototipagem: a máquina de prototipagem (III) e o protótipo de um celular (IV).

[1] Grupo de Redes, Engenharia de Software e Sistemas
[2] http://mobiline.great.ufc.br

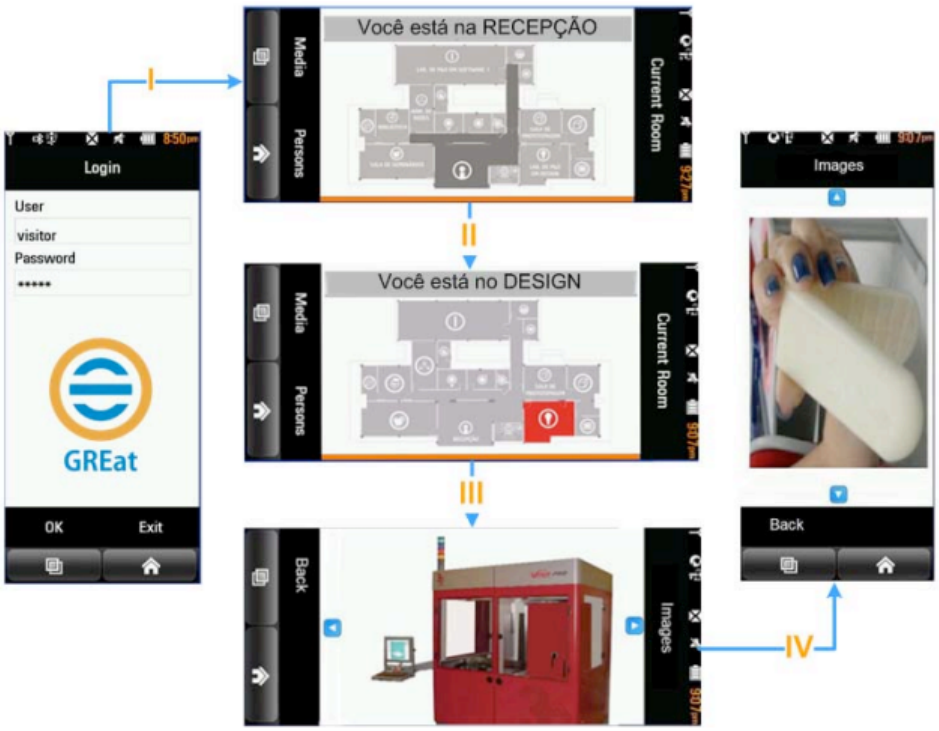


Figura 5.1: Esquema de funcionamento do GreatTour, adaptado de (MARINHO et al., 2010).

5.2 *Design* da avaliação preliminar

Uma vez concretizados os resultados deste trabalho, decidiu-se por executar uma avaliação preliminar nos moldes de um experimento controlado (WOHLIN et al., 2000) para verificar se os objetivos propostos foram atendidos. Optou-se por usar o termo "avaliação preliminar" porque também era objetivo dessa avaliação analisar a viabilidade de um experimento controlado, bem como identificar possíveis problemas que poderiam impactar os resultados desse experimento. Alguns autores como Crescencio (2011) preferem chamar essa avaliação pré-experimento de estudo piloto.

Logo, além de servir de base para um futuro experimento, o propósito da avaliação preliminar conduzida foi avaliar, com respeito ao esforço e aplicabilidade, do ponto de vista do pesquisador, no contexto de estudantes de Ciência da Computação, o impacto do uso do CAPLUC no entendimento de casos de uso de uma LPSSC, do ChAPTER na geração de testes para uma LPSSC, e da ToChAPTER nas atividades de modelagem de caso de uso e geração de testes para LPSSC.

Como a avaliação preliminar foi conduzida quando os estudantes de graduação e pós-graduação em Ciência da Computação da Universidade Federal do Ceará (UFC) se encontravam em final de período letivo, poucos alunos se voluntariaram. No total participaram somente seis voluntários, sendo dois alunos de graduação e quatro alunos de pós-graduação.

É importante mencionar que todos os voluntários dessa avaliação preliminar, com exceção de um dos alunos de graduação, já haviam participado do experimento realizado para avaliar as propostas existentes de templates de caso de uso para LPS, conforme apresentado no

Capítulo 4. Contudo, acredita-se que isso não impactou nos resultados dessa avaliação preliminar porque essa trata de linhas de produto sensíveis ao contexto, diferentemente do experimento do Capítulo 4 que foca somente em linhas de produto não sensíveis ao contexto.

A Figura 5.2 apresenta as atividades executadas na avaliação conduzida. A primeira atividade correspondeu a execução de um questionário pré-experimento (Apêndice D), aplicado para caracterizar o perfil dos participantes de maneira semelhante ao que foi feito no experimento dos templates apresentado no Capítulo 4.

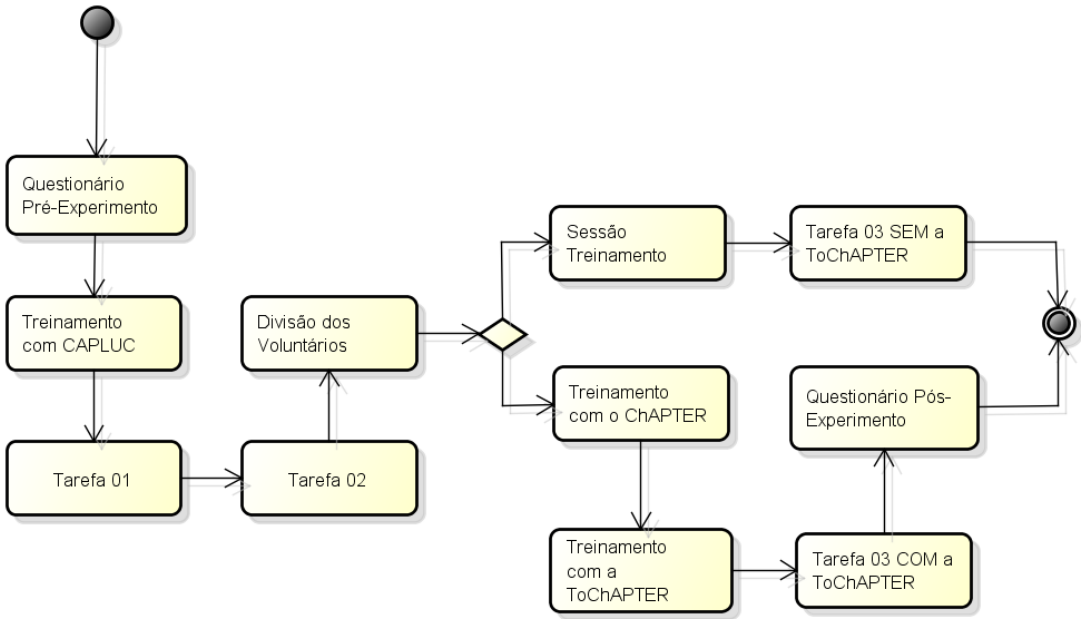


Figura 5.2: Atividades realizadas na avaliação preliminar.

Os resultados do questionário pré-experimento revelaram que, dentre outras coisas, dos quatro alunos de pós graduação que participaram da avaliação todos conheciam as definições básicas de LPS, LPSSC, descrições textuais de casos de uso e teste de software, mas nenhum deles tinha estudado algo relacionado ao teste de aplicações sensíveis ao contexto. Quanto aos alunos de graduação, ambos conheciam o básico de teste de software e afirmaram não ter estudado nada sobre descrições textuais de casos de uso ou teste de aplicações sensíveis ao contexto, mas um deles afirmou saber o básico de LPS. Além disso, todos os voluntários afirmaram não conhecer o método de Partição de Categorias e nem o método PLUTO.

Dessa forma, com o objetivo de minimizar os efeitos da falta de conhecimento sobre os fatores do estudo experimental, na Sessão de Treinamento com o CAPLUC foram explorados os conceitos importantes sobre LPS, LPSSC e descrições textuais de caso de uso. Ainda nesta sessão os voluntários aprenderam sobre a estrutura do CAPLUC e fizeram um exercício para averiguar se todos entenderam como funciona a estrutura deste template.

Após o treinamento com o CAPLUC, os participantes realizaram as duas tarefas de compreensão (Tarefa 01 e Tarefa 02) para avaliar o template. Em cada tarefa, os participantes responderam um questionário (Apêndice E) de quatro questões analisando um caso de uso modelado no CAPLUC. Na Tarefa 01, o caso de uso utilizado foi o Captura Contexto (apresentado na Figura 4.8 da Seção 4.1) enquanto que na Tarefa 02 foi o Mostra Documentos (apresentado

na Figura 4.9 da Seção 4.1). Ressalta-se que não foram utilizados os mesmos casos de uso do experimento descrito no Capítulo 04, porque esses eram de uma LPS não sensível ao contexto.

Com relação ao questionário da tarefa, as três primeiras questões estavam relacionadas a identificação das *features* opcionais, alternativas e as que tinham restrições de contexto associadas. Já a quarta questão era relacionada ao entendimento do caso de uso. Nesse ponto, o tempo gasto para responder o questionário bem como a quantidade de respostas corretas foram utilizadas para avaliar o desempenho com o CAPLUC. Por fim, após essas duas tarefas, os participantes foram indagados sobre a opinião deles quanto aos benefícios do formato do CAPLUC para o entendimento do caso de uso.

A etapa seguinte da avaliação executada consistiu em avaliar o método e por isso os participantes foram divididos em dois grupos, A e B, cada um com um aluno de graduação e dois alunos de pós-graduação. A seleção de quem ficaria em qual grupo foi feita mediante sorteio. O Grupo A tinha por objetivo gerar testes para dois casos de uso de uma LPSSC (os mesmos da avaliação do template) sem usar a ferramenta ToChAPTER ou método ChAPTER. Para isso, os voluntários desse grupo receberam um treinamento sobre como eles deveriam especificar os testes e sobre o objetivo dos testes que deveriam ser planejados. Não foi ensinada nenhuma técnica de geração de testes porque os participantes deveriam planejar os testes com base na experiência particular de cada um.

Já os participantes do Grupo B foram convidados a usar o método e a ferramenta para gerar testes. Logo, eles receberam treinamento para o uso do ChAPTER e da ToChAPTER e depois realizaram a Tarefa 03 usando a ferramenta ToChAPTER, que implementa o método ChAPTER. Como nessa etapa foi utilizada a ferramenta, ao mesmo tempo que se coletou dados para avaliar o método, dados foram coletadas para avaliar a ferramenta. Vale ressaltar que o tempo do treinamento sem o ChAPTER e o tempo do treinamento para o uso do ChAPTER e da ToChAPTER foram similares.

5.3 Avaliação do Template CAPLUC

Para avaliar o template proposto neste trabalho, o CAPLUC, foram utilizados os dados de tempo e acurácia obtidos pelos participantes nas Tarefas 01 e 02 (Tabela 5.2). No caso do template, o propósito da avaliação preliminar era verificar se o CAPLUC tornava fácil o entendimento do caso de uso.

A Tabela 5.1 apresenta os resultados coletados com relação as tarefas de compreensão 01 e 02. Para a Tarefa 01, dois dos seis voluntários erraram a questão final, que era a questão relacionada ao entendimento do caso de uso. Ressalta-se que um erro na questão final sinalizava que o voluntário podia estar fazendo um uso incorreto do template.

Logo, desconsiderando as amostras relacionadas a esses dois voluntários, dos outros quatro apenas um (Voluntário 4) não atingiu a acurácia máxima, pois errou uma questão. Como resultado, para a Tarefa 01, a acurácia média dos quatro participantes que acertaram a questão final foi de 91.5%. Além disso, o tempo médio desses quatro voluntários foi de seis minutos para realizar essa tarefa.

Tabela 5.1: Tabela com os resultados obtidos nas Tarefas 01 e 02

	Tarefa 01: Caso de Uso "Captura Contexto"					Tarefa 02: Caso de Uso "Mostra Documentos"				
Voluntário	Quais features opcionais?	Quais features alternativas?	Quais features tem restrição de contexto?	Questão Final	Tempo (min)	Quais features opcionais?	Quais features alternativas?	Quais features tem restrição de contexto?	Questão Final	Tempo (min)
1	1	1	1	0	10	1	1	1	0	7
2	1	1	1	1	6	1	1	1	1	6
3	1	1	1	1	5	1	0	1	0	4
4	1	1	0	1	8	1	1	1	1	7
5	1	1	1	1	5	1	1	1	0	4
6	1	0	0	0	15	0	0	1	0	10

Legenda

1	acertou a questão
0	errou a questão

Com relação a Tarefa 02, apenas dois voluntários acertaram a questão final, o que sinalizou que os demais ou sentiram alguma dificuldade quanto ao uso do template ou a questão final da Tarefa 02 tinha problemas. Após analisar o ocorrido e por meio de diálogo com alguns dos voluntários, percebeu-se que a pequena diferença existente entre as alternativas da questão final pode ter levado ao grande número de erros nessa questão. Considerando apenas os voluntários que acertaram a questão final, a acurácia obtida por ambos foi de 100% e o tempo médio de execução dessa tarefa foi de seis minutos e 30 segundos.

Dessa forma, os dados de tempo e acurácia coletados indicaram que os voluntários usando o CAPLUC gastaram em média 6,25 minutos e tiverem boa acurácia, com quase todos que acertaram a questão final atingindo 100%.

Por fim, após as Tarefas 01 e 02, os voluntários foram questionados se eles acreditavam que o template auxiliou a entender o caso de uso. Todos os voluntários afirmaram que o CAPLUC auxilou o entendimento do caso de uso. De uma forma geral, a justificativa dada pelos participantes foi que as características do template quanto a representação das *features* opcionais, alternativas e restrições de contexto tornavam a tarefa de localizá-las fácil.

É interessante destacar que um dos voluntários também destacou que leu as questões descritivas presentes associadas aos pontos de variação no CAPLUC (mais detalhes na Seção 4.1) para se orientar melhor. Além disso, um outro voluntário afirmou que o uso dos índices para indicar os pontos com variabilidade e com contexto eram interessantes, mas que era preciso um tempo para se acostumar e, enquanto isso, o sumário de variação auxiliava bastante.

Ainda com relação aos resultados obtidos quanto ao CAPLUC, é interessante estabelecer de alguma forma uma comparação entre os resultados apresentados nesta seção e os resultados obtidos no Capítulo 4, durante a avaliação de templates de casos de uso para LPS. Considerando a acurácia obtida com o CAPLUC para as Tarefas 01 (100%) e 02 (91.5%), a acurácia média com o CAPLUC foi de 95.75%, o que é um pouco superior a acurácia do melhor template de caso de uso identificado para LPS, o do ERIKSSON et al. (2004), que obteve uma acurácia média de 94.87%.

Além disso, o tempo médio com o CAPLUC foi de 6.25 minutos, o que é um pouco superior ao tempo médio do template do ERIKSSON et al., com 4.74 minutos, mas ainda assim permanece abaixo do tempo gasto por outros dois templates do experimento descrito no

Capítulo 4 (8.21 minutos com o template do BONIFACIO e BORBA; 6.64 minutos com o template do JOHN e MUTHING). Vale lembrar que os casos de uso utilizados na avaliação do CAPLUC por serem de uma LPSSC são mais complexos do que os casos de uso do experimento do Capítulo 4 que eram de uma LPS não sensível ao contexto. Contudo, essa comparação feita serve apenas para dar uma noção dos resultados obtidos já que o experimento do CAPLUC foi conduzido em um cenário diferente do experimento dos templates do Capítulo 4.

Logo, os resultados da avaliação, por meio dos dados coletados e das respostas dos voluntários, representam indícios de que o CAPLUC, proposto neste trabalho, auxilia o entendimento de um caso de uso pequeno de uma LPSSC. Entretanto, para generalizar essa afirmação para todo caso de uso de uma LPSSC ainda se faz necessário uma avaliação com uma quantidade maior voluntários e uma diversidade maior de casos de uso.

5.4 Avaliação do Método ChAPTER

Para avaliação preliminar do método, foram utilizados os resultados da Tarefa 03. Conforme apresentado no começo deste capítulo, para avaliar o método e a ferramenta, os voluntários foram divididos em 2 grupos (A e B), ambos com dois alunos de pós-graduação e um de graduação. Cada membro do grupo A planejou, na Tarefa 03, um conjunto de casos de testes com base na experiência individual de cada voluntário, para dois casos de uso descritos textualmente para a LPSSC do GreatTour (MOBILINE, 2012). Por outro lado, os participantes do grupo B realizaram a mesma atividade utilizando a ferramenta ToChAPTER e o método ChAPTER.

Vale ressaltar que dos três membros do Grupo B (que usaram o ChAPTER) apenas os dois alunos de pós-graduação planejaram os testes na Tarefa 03. Isso porque o aluno de graduação desse grupo, o Voluntário 6, desistiu da avaliação preliminar logo após as atividades com o CAPLUC.

Quanto aos dois alunos de pós-graduação do Grupo B, um deles (Voluntário 3) gastou 10 minutos para modelar os dois casos de uso na ferramenta e 19 minutos para criar os casos de testes tomando por base os cenários de testes gerados pela ferramenta, enquanto que o outro (Voluntário 4) levou 14 minutos e 25 minutos para essas respectivas atividades. Contudo, essa medida de tempo não foi utilizada porque ela variava conforme o estilo de escrita (i.e., se usando abreviações ou descrevendo detalhadamente) de cada voluntário. Logo, a métrica utilizada para avaliar o método foi a de cobertura dos testes.

A Tabela 5.2 apresenta os diferentes cenários de testes gerados com o apoio do ChAPTER para os dois casos de uso “Captura de Contexto” e “Mostra Documentos”. É importante mencionar que ao final da Tarefa 03, os voluntários do Grupo B haviam descrito para cada um dos 12 diferentes cenários de testes gerados pela ferramenta, um respectivo caso de teste. Além disso, esses voluntários relacionaram para cada um dos oito estados de contexto, gerados pela ToChAPTER, um conjunto de casos de testes.

Na Tabela 5.2, para cada cenário de teste é descrito o contexto que deve ser verdadeiro e as escolhas para cada categoria. Vale ressaltar a diferença entre as escolhas “Sem

Tabela 5.2: Cenários de teste gerados pelo ChAPTER para a Tarefa 03

Caso de Uso: Mostra Documentos

Número	Contexto Atuante	Escolhas
1	"Nenhuma rede"	Fluxo alternativo
2	WIFI, NOT_LOW_BATTERY	Com Imagem/Com Video
3	WIFI, NOT_LOW_BATTERY	Sem Imagem/Com Video
4	NOT_WIFI, NOT_LOW_BATTERY	Sem Video
5	WIFI, LOW_BATTERY	Com Imagem/Sem Video
6	WIFI, LOW_BATTERY	Sem Imagem/Sem Video
7	NOT_WIFI, LOW_BATTERY	Sem Video

Caso de Uso: Captura de Contexto

Número	Contexto Atuante	Escolhas
8	SENSOR, SENSORES_AMBIENTE	Sensor
9	NOT_SENSOR, SENSORES_AMBIENTE	Qr_Code
10	SENSOR, NOT_SENSORES_AMBIENTE	Qr_Code
11	NOT_SENSOR, NOT_SENSORES_AMBIENTE	Qr_Code
12	"Nenhum meio disponível"	Fluxo alternativo

Imagem/Sem Vídeo", do Cenário 6, e "Sem Vídeo", dos Cenários 4 e 7. No Cenário 6, o contexto WIFI está ativo e com isso a *feature* "Imagem" pode ser selecionada, mas nesse cenário, ela não é, e por isso a escolha fica como "Sem Imagem". No caso dos Cenários 4 e 7, o contexto NOT_WIFI contradiz a restrição de contexto da *feature* "Imagem", pois esta *feature* exige o contexto WIFI. Logo, os Cenários 4 e 7 representam estados em que não é possível selecionar a *feature* "Imagem" e por isso ela não aparece na coluna Escolhas da Tabela 5.2.

A Tabela 5.3 apresenta os testes planejados pelos voluntários que não utilizaram o método. Os Voluntários 1 e 2 são alunos de pós-graduação e o Voluntário 5 é o aluno de graduação. Nessa tabela estão descritos quais dos cenários de testes gerados pela ferramenta eles planejaram casos de testes, bem como a cobertura atingida por esses casos de testes considerando os 12 diferentes casos de testes sugeridos pela ferramenta (Figura 5.2) como sendo uma cobertura 100%.

Tabela 5.3: Resultados dos participantes sem o método ChAPTER para a Tarefa 03

Grupo A	tempo (min)	testes	Cobertura (%)
Voluntario 1	38	1,2,3,7,8,9,12	58,33
Voluntario 2	31	1,2,5,8,11	41,67
Voluntario 5	24	1,2,5,8,9,10	50

Conforme observado na Tabela 5.3, o voluntário com melhor participação conseguiu uma cobertura de apenas 58% se comparado aos testes gerados pela ferramenta. Além disso, vale ressaltar que o voluntário que usou a ferramenta gastou 29 minutos para a Tarefa 03 enquanto que o voluntário com melhor resultado do grupo A (voluntário 1) planejou 7 casos de testes em 38 minutos.

Dessa forma, os resultados revelam indícios de que o método possibilita uma cobertura maior dos casos de testes possíveis, enquanto que o uso do método e da ferramenta de apoio pode auxiliar a reduzir o tempo gasto para o planejamento dos testes. Contudo, vale res-

saltar que é necessário um experimento com um número maior de voluntários e uma diversidade maior de casos de uso para confirmar os resultados obtidos.

5.5 Avaliação da Ferramenta ToChAPTER

A análise da usabilidade da ferramenta ToChAPTER foi feita por meio da observação de dois voluntários do grupo B utilizando a ferramenta. Além dos dados coletados com a observação, esses voluntários foram estimulados a fornecer opiniões sobre a interface da ferramenta. A Tabela 5.4 apresenta os resultados obtidos diretamente pelos voluntários e pela observação.

Tabela 5.4: Resultados obtidos para a ferramenta ToChAPTER

Voltuntário	Comentários dos Voluntários	Problemas observados
3	a) Colocar a numeração dos passos automática; b) Melhorar as mensagens de erro; c) Ao gerar os testes, rolar a barra ao topo da página; d) Colocar link direto para os passos da lista de SLICES da linha para poder editar os passos sem passar pela editar das outras informações do SLICE; e) Fazer atualização automática da variação; f) Aumentar o frame do fluxo alternativo; g) Aumentar o tamanho do campo restrição do fuxo alternativo; h) Permitir que o campo de ações do Usuário possa ser vazio; i) Permitir o cadastro de uma variação alternativa opcional;	1) Problema com o frame dos passos (causa confusão); 2) Falta link da lista de PiECES para o produto;
4	a) Trocar "enviar" por "salvar"; b) Aumentar o campo de restrição dos passos alternativos; c) Colocar para atualizar automaticamente a página dos passos ao adicionar uma variante a um passo; d) Diminuir o números de "cliques" e de passos necessários para modelar na ferramenta; e) Mostrar mensagem de erro no topo da página; f) Aceitar o campo vazio nas acões do usuário;	1) Fazer com que as páginas ao abrirem fiquem no topo (página dos passos); 2) Muitos cliques para relacionar uma variante a um passo;

Conforme pode ser observado na Tabela 5.4, ainda existem alguns problemas de interface que precisam ser resolvidos na ferramenta. Destaca-se que esse problemas podem atrapalhar o usuário e inclusive fazer com que o método de geração de testes usando a ferramenta não seja tão vantajoso quanto deveria. Logo, a avaliação da interface da ferramenta por meio do uso prático com dois voluntários para identificar esses problemas foi essencial para se conhecer o que precisa ser melhorado na ferramenta.

Além dos dados coletados, os voluntários foram questionados sobre a facilidade de uso e usabilidade das interfaces com uma escala de um a cinco: 1 - Péssima; 2 - Ruim; 3 - Média; 4 - Boa; 5 - Ótima. Quanto a facilidade de uso, um voluntário afirmou que era Boa enquanto o outro afirmou que era Média. Já com relação a usabilidade das interfaces de usuários ambos julgaram como Média.

Dessa forma, pelos resultados apresentados nesta seção, percebe-se que embora a ferramenta tenha auxiliado os participantes na aplicação do método ChAPTER, ela ainda carece de melhorias em termos de interface antes de poder ser submetida a um experimento controlado mais rígido ou de ser utilizada em caráter comercial.

5.6 Problemas identificados

Além de possibilitar uma avaliação inicial do template, do método e da ferramenta, era objetivo da avaliação preliminar, descrita neste capítulo, avaliar a viabilidade de um futuro experimento controlado para uma avaliação mais rigorosa. Dessa forma, à medida que as tarefas foram sendo executadas, os problemas identificados foram sendo registrados.

No total foram identificados cinco problemas que precisam ser levados em conta no planejamento e execução de um experimento controlado para avaliar os resultados deste trabalho:

- Erros ortográficos em alguns artefatos. Alguns dos casos de uso entregues aos voluntários durante o treinamento tinham erros de ortografia.
- Questões mal formuladas. Uma das questões finais das tarefas de avaliação do template apresentava alternativas bem semelhantes. Pelas respostas obtidas com os voluntários é possível que essa semelhança tenha contribuído para que os participantes errassem a questão.
- Avaliação longa. Durante a avaliação preliminar percebeu-se que avaliar o template, o método e a ferramenta no mesmo dia torna a atividade cansativa e, com isso, o voluntário pode perder o interesse.
- Exercícios insuficientes. Em cada treinamento foram feitas atividades de exercícios com os participantes no intuito de fixar os conceitos transmitidos. Contudo, observou-se que talvez fosse necessário mais exercícios, principalmente na parte relacionada a geração de testes.
- Tempo dos participantes. Os participantes tinham pouco tempo disponível então a avaliação preliminar foi conduzida em vários dias, respeitando a disponibilidade dos participantes. Além disso, uma vez que a avaliação preliminar foi conduzida quando os alunos estavam em final de período letivo, poucos foram os que se voluntariaram.

5.7 Conclusão

Neste capítulo foi apresentada a avaliação preliminar realizada para a verificar os benefícios do template CAPLUC, do método ChAPTER e da ferramenta ToChAPTER. Além disso, era objetivo dessa avaliação verificar a viabilidade de um experimento controlado e identificar possíveis problemas que poderiam impactar os resultados desse experimento.

Com relação ao template, seis voluntários o utilizaram durante a análise de dois casos de usos de uma linha de produto de software sensível ao contexto. Os resultados indicaram que o CAPLUC favorece o entendimento, porque permite a identificação rápida dos passos relacionados a *features* opcionais e alternativas e também dos passos associados a *features* que apresentavam restrições de contexto.

Quanto ao método, ele foi avaliado em conjunto com o uso da ferramenta ToCHAPTER. Como resultado da avaliação conduzida obteve-se que dos três voluntários que não utilizaram o ChAPTER, o que teve melhor resultado só atingiu 58% da cobertura conseguida pelos testes gerados pelo método. Logo, a avaliação revelou indícios de que o método permite uma cobertura de testes maior se comparado a criação de testes com base na experiência do testador.

Já com respeito a ferramenta ToChAPTER, ela foi analisada quanto a interface por dois voluntários que a utilizaram para modelar casos de uso e para gerar testes. Com o uso da ferramenta, foi possível averiguar que ela representa um suporte necessário a aplicação do método ChAPTER, uma vez que a ToChAPTER automatiza algumas atividades desse método tornando sua aplicação mais rápida.

É importante mencionar que os resultados obtidos quanto ao template e ao método representam indícios iniciais e não podem ser generalizados. Isso porque somente um experimento controlado mais rigoroso, com mais voluntários e maior diversidade de casos de uso permitirá uma análise mais apurada sobre a adequabilidade das contribuições propostas neste trabalho. Além disso, em se tratando da ferramenta, constatou-se com a avaliação preliminar que melhorias ainda precisam ser feitas antes do seu uso em um experimento controlado com mais participantes.

Por fim, com a avaliação preliminar foi possível identificar cinco problemas, relatados na Seção 5.6, que poderiam impactar os resultados de um experimento controlado para avaliar o template, o método e a ferramenta. Esses problemas vão então ser levados em consideração durante o planejamento e a execução desse experimento, que é um dos trabalhos futuros.

6 CONCLUSÃO

Este capítulo sumariza os resultados alcançados e as projeções de trabalhos futuros. Conforme descrito nos capítulos anteriores, esta dissertação apresenta três contribuições. A principal contribuição diz respeito a um método de geração de testes para linhas de produto de software sensíveis ao contexto utilizando como base especificações textuais de casos de uso. Como contribuição secundária foi proposto um template para casos de uso de LPSSC, uma vez que não foi encontrado na literatura nenhum template para especificação textual de casos de uso para LPSSC. Por fim, para prover uma automatização do método e um apoio a modelagem dos casos de uso de uma LPSSC, outra contribuição apresentada neste trabalho foi a ferramenta de apoio desenvolvida.

Na Seção 6.1 são então detalhados os principais resultados alcançados com a realização deste trabalho, enquanto que na Seção 6.2 são descritas as perspectivas de trabalhos futuros.

6.1 Resultados Alcançados

Com o reuso sistemático em larga escala surgiram as Linhas de Produto de Software (LPS), que constituíam famílias de sistemas de um mesmo domínio. Logo, por meio de uma linha de produto de software é possível, dentre outras coisas, elevar o grau de reuso, melhorar a qualidade dos produtos e reduzir os custos de desenvolvimento. Quando a linha de produto é dedicada ao desenvolvimento de aplicações sensíveis ao contexto tem-se uma Linha de Produto de Software Sensível ao Contexto (LPSSC).

Por englobar o domínio das aplicações sensíveis ao contexto e a abordagem de desenvolvimento em forma de linha de produto, testar uma LPSSC envolve lidar, ao mesmo tempo, com os desafios inerentes aos testes de uma LPS, que precisam, por exemplo, maximizar a reutilização, e aos testes de aplicações sensíveis ao contexto, que devem, entre outras coisas, levar em consideração o contexto da aplicação.

Vale destacar que o teste dos artefatos da LPS nos estágios iniciais é importante para garantir a correção dos defeitos a um custo mínimo. Nesse cenário de geração precoce dos testes, é possível encontrar na literatura alguns trabalhos que apoiam a geração de teste para uma LPS a partir dos requisitos da linha, conforme é descrito no Capítulo 3. Contudo, nenhum trabalho foi encontrado relacionado a LPSSC.

Dessa forma, este trabalho se enquadra nesta lacuna, pois ele apoia os testes de uma LPSSC com a proposta de um ambiente para geração de cenários de testes para LPSSC. Desse ambiente, fazem-se parte: um método de geração de testes a partir de descrições textuais de casos de uso; um template para especificação textual de caso de uso; e uma ferramenta de apoio.

O método proposto, o ChAPTER (acrônimo de *Context Aware software Product line TEsting geneRation method*), possibilita a geração de testes para uma LPSSC em três níveis. No

primeiro nível, um conjunto de cenários de testes globais da linha, relativos a todos os produtos, é gerado a partir dos SLICES da LPSSC. Um SLICE (*Software product LIne Contextual use casE*) é um termo usado no método para se referir a um caso de uso de uma LPSSC que descreve variabilidades, similaridades e informações de contexto.

No segundo nível, PiECES são criados a partir dos SLICES, e com eles cenários de testes globais do produto são gerados. Esses cenários representam os testes para todas as configurações possíveis do produto, ao qual os PiECES pertencem. Nesse ponto, um PiECE (*Product contextual usE CasE*) é um termo usado no ChAPTER para se referir a um caso de uso de um produto final de uma LPSSC. Logo, SLICES criados na Engenharia de Domínio são instanciados e adaptados na Engenharia de Aplicação para dar origem aos PiECEs.

A terceira etapa do método, por sua vez, consiste em gerar uma máquina de estados do produto, onde cada estado representa uma configuração possível do produto. Nesse caso, as transições são feitas por meio de mudanças de contexto. Com isso, para cada estado é associado um conjunto de cenários de testes que pode ser utilizado para testar as funcionalidades do produto ou para verificar se o produto se reconfigura como esperado.

Já com relação ao template, que também faz parte do ambiente proposto, ele é denominado de CAPLUC, acrônimo de *Context Aware software Product Line Use Case template*. O objetivo desse template é especificar como pode ser feita a descrição textual de um caso de uso de uma LPSSC atendendo, dessa forma, a demanda do método ChAPTER, que se baseia em casos de uso descritos textualmente.

É importante mencionar que na literatura não foram encontrados templates para especificação textual de casos de uso próprios para uma LPSSC, e por isso, a proposta do CAPLUC foi fundamentada na análise de alguns dos templates encontrados para LPS não sensíveis ao contexto. Essa análise foi conduzido por meio de um experimento controlado com 46 voluntários. Os resultados do experimento indicaram, dentre outras coisas, que o sistema de índices para representar as variações do template, do ERIKSSON et al. (2004), e a descrição textual das variações ao final do caso de uso, como no template do GOMMA (2004), favoreciam o entendimento de casos de uso de uma LPS. Logo, com base nos resultados obtidos, o CAPLUC foi proposto incorporando as melhores características identificadas nos templates e especificando elementos próprios para a descrição de restrições de contexto da LPSSC.

Além disso, outra parte integrante do ambiente de geração de cenários de testes para LPSSC proposto neste trabalho é a ferramenta web ToChAPTER. Essa ferramenta permite a modelagem de SLICEs e PIECES seguindo o CAPLUC, além de possibilitar a geração dos testes com o ChAPTER. Com relação a arquitetura da ToChAPTER, essa é composta por três módulos funcionais principais: dos controladores, de geração de testes e de persistência dos dados.

O módulo dos controladores é responsável por receber as informações do usuário e retornar conteúdo por meio das páginas web. Além disso, é esse módulo que invoca funcionalidades dos outros dois módulos com base nas ações do usuário. O módulo de geração de testes cria os cenários de testes seguindo a especificação do método ChAPTER. Já o módulo de persistência dos dados é responsável por armazenar as entidades implementadas na ferramenta

no banco de dados.

Nesse momento é interessante salientar o diferencial deste trabalho quanto as abordagens encontradas na literatura. Com relação ao CAPLUC, embora tenha sido encontrado na literatura nove diferentes abordagens para especificação textual de caso de uso para uma LPS, nenhuma delas tratava da descrição de informações de contexto, característica relevante no caso de uma LPSSC. Logo, o diferencial do CAPLUC está no fato dele focar em LPSSC, pois permite que a especificação do caso de uso descreva além das funcionalidades, as variabilidades e restrições de contexto.

Quanto ao método de geração de testes, na literatura há várias abordagens que geram testes a partir de descrições textuais de casos de uso (BERTOLINI; MOTA, 2010; CHATTERJEE; JOHARI, 2010; SOMé; ANTHONYSAMY, 2008; ROUBTSOV; HECK, 2006). Contudo, apenas cinco trabalhos relacionados a uma LPS foram identificados (NETO et al., 2011; BERTOLINO; GNESI, 2003; ZORZO et al., 2011; ALI; MOAWAD, 2010; KAMSTIES et al., 2004) e nenhum deles é voltado a uma LPSSC.

Dessa forma, o diferencial do ChAPTER também é o foco em LPSSC, ainda não abordada por nenhum dos trabalhos relacionados identificados na literatura. Além disso, o ChAPTER usa uma adaptação do método de Partição de Categorias (OSTRAND; BALCER, 1988), o que o torna simples, pois ele não depende de nenhuma ferramenta, modelo ou linguagem para permitir a geração dos testes, ao contrário das abordagens identificadas na literatura para LPS, com exceção da abordagem de Bertolino et al. (2003), que também usa uma adaptação do método de Partição de Categorias.

Por fim, uma avaliação preliminar na forma de experimento controlado foi conduzida para avaliar o ambiente de geração de cenários de testes e verificar a viabilidade de um experimento controlado para avaliá-lo. Participaram dessa avaliação seis alunos, dos quais dois eram de graduação e quatro de pós-graduação em Ciência da Computação da Universidade Federal do Ceará.

Os resultados dessa avaliação revelaram indícios iniciais de que o CAPLUC favorece o entendimento, porque permite a identificação rápida dos passos relacionados a *features* opcionais e alternativas e também dos passos associados a *features* que apresentavam restrições de contexto. Além disso, com relação ao ChAPTER, constatou-se que ele permitiu uma maior cobertura com os cenários de testes gerados se comparado aos testes criados pelos voluntários que não utilizaram o método. Já com respeito a ferramenta ToChAPTER, os resultados da avaliação preliminar indicaram melhorias que ainda precisam ser feitas na interface da ferramenta antes do seu uso em um experimento controlado com mais participantes. Também foi possível por meio dessa avaliação preliminar identificar cinco problemas que poderiam impactar os resultados de um futuro experimento controlado com mais participantes.

Em se tratando de publicação durante o mestrado, quatro artigos foram produzidos, incluindo artigos publicados e artigos aceitos que ainda se encontram em revisão. A Tabela 6.1 apresenta detalhes destes quatro artigos.

Os resultados descritos nos dois primeiros artigos (n. 1 e n. 2) não fazem parte

Tabela 6.1: Lista dos artigos científicos produzidos.

Número	Referência	Tipo	Situação
1	COSTA, P. A. S. ; **SANTOS, I. S.** ; ANDRADE, R. M. C. . PictureFrame: Um padrão para criação de imagens reutilizáveis. In: Miniconferência Latino-Americana de Linguagens de Padrões para Programação (MiniPLoP Brasil), 2011, São Paulo. Writers` Workshop, 2011	Conferência	Publicado
2	**SANTOS, I. S.** ; DANTAS, V. L. L. ; SANTOS, R. M. ; ANDRADE, R. M. C. . Testes de Aplicações Móveis: Uma Análise das Pesquisas Científicas via Revisão Sistemática. In: XI Simpósio Brasileiro de Qualidade de Software, 2012, Fortaleza, CE. Anais do XI Simpósio Brasileiro de Qualidade de Software, 2012	Conferência	Publicado
3	**SANTOS, I. S.** ; SANTOS NETO, P. ; ANDRADE, R. M. C. . Um Método para Geração de Testes para Linhas de Produto de Software Sensíveis ao Contexto. In: II Workshop de Teses e Dissertações do CBSoft. III Congresso Brasileiro de Software: Teoria e Prática, 2012, Natal, RN. Anais do II Workshop de Teses e Dissertações do CBSoft, 2012	Workshop de Teses e Dissertações	Publicado
4	**SANTOS, I. S.**; SANTOS NETO, P. ; ANDRADE, R. M. C. A Use Case Textual Description for Context Aware SPL based on a Controlled Experiment. In CAISE`13 Fórum. 2013	Conferência	Aceito

das contribuições específicas deste trabalho de mestrado, mas são trabalhos que foram desenvolvidos durante esta pesquisa e que ajudaram a consolidar o conhecimento na área de testes e padrões de software. O primeiro artigo (n. 1) apresenta um padrão para produção de figuras reutilizáveis e foi produzido como um resultado parcial da disciplina Reutilização de Software, que foi uma das seis disciplinas cursadas durante esse trabalho de mestrado. O segundo artigo (n. 2) apresenta uma revisão sistemática sobre testes de aplicações móveis que forneceu conhecimento para a realização de uma busca mais sistemática por trabalhos relacionados a esta pesquisa.

O terceiro artigo (n. 3) apresenta as ideias iniciais desta dissertação de mestrado e foi apresentado em um Workshop de Testes e Dissertações, e por fim, o quarto artigo (n. 4) descreve a proposta do CAPLUC, template para descrição textual de casos de uso para LPSSC, que foi apresentado no Capítulo 4.

6.2 Trabalhos Futuros

Conforme mencionado na Seção 6.1, os resultados desse trabalho constituem-se do ChAPTER, um método de geração de testes para LPSSC, do CAPLUC, um template para casos de uso de LPSSC, e uma ferramenta de apoio ao método ChAPTER e a modelagem de casos de uso com o CAPLUC, denominada de ToChAPTER. Tais resultados apresentam limitações que surgem como potenciais trabalhos futuros:

- Geração dos casos de testes: o método permite a geração de cenários de testes, mas para execução dos testes tais cenários ainda precisam ser convertidos em casos de testes. Dessa forma, um próximo passo seria propor um método ou ferramenta que por meio dos cenários de testes gere casos de testes executáveis. Para os casos de testes serem executados também é preciso propor um método de geração de dados de teste que leve em conta as informações de contexto e de variabilidade de uma LPSSC;
- Aplicação do CAPLUC em outras LPSSC: durante o decorrer deste trabalho, o CAPLUC foi utilizado para auxiliar a modelagem dos casos do Mobiline (MOBILINE, 2012), que

se trata de uma linha de produtos móveis e sensíveis ao contexto. Contudo, é interessante que o CAPLUC seja aplicado a outras linhas sensíveis ao contexto para avaliar se ele atende a todas as necessidades de especificação;

- Extensão do ChAPTER: atualmente, o ChAPTER trabalha com casos de uso descritos textualmente, mas como casos de uso podem ser representados por meio de diagramas de casos de uso, de sequência e atividade é interessante que o método seja estendido para dar suporte a outras fontes, além da textual;

- Melhorias na ToChAPTER: conforme os resultados da avaliação preliminar conduzida, embora a ToChAPTER tenha auxiliado na execução das tarefas de modelagem de caso de uso e geração dos testes, os voluntários relataram problemas de interface que precisam ser superados para facilitar o uso da ferramenta;

- Realização de um experimento controlado: os resultados obtidos foram avaliados somente com uma avaliação simplificada. Logo, é necessário que o método, o template e a própria ferramenta sejam submetidas a uma avaliação mais rigorosa com mais voluntários e uma diversidade maior de casos de uso.

REFERÊNCIAS BIBLIOGRÁFICAS

ALI, M. M.; MOAWAD, R. An approach for requirements based software product line testing. In: *Informatics and Systems (INFOS), 2010 The 7th International Conference on*. [S.l.: s.n.], 2010. p. 1 –10.

ALMEIDA, E. S.; ALVARO, A.; GARCIA, V. C.; MASCENA, J. C. C. P.; BURéGIO, V. A. A.; NASCIMENTO, L. M.; LUCRéDIO, D.; MEIRA, S. L. *C.R.U.I.S.E: Component Reuse in Software Engineering*. 1. ed. [S.l.]: C.E.S.A.R e-book, 2007.

ANJOS, A. G. R. dos. *Aplicações móveis cientes de contexto*. 2006. Trabalho de Graduação. Universidade Federal de Pernambuco.

ANTHONYSAMY, P.; SOMÉ, S. S. Aspect-oriented use case modeling for software product lines. In: *Proceedings of the 2008 AOSD workshop on Early aspects*. New York, NY, USA: ACM, 2008. (EA '08), p. 1–8. ISBN 978-1-60558-143-9. Disponível em: <http://doi.acm.org/10.1145/1404946.1404951>.

ARAúJO, D. O. *Elaboração de especificações de casos de uso para linhas de produto de software baseada em fragmentos*. Dissertação (Mestrado) — Instituto de Matemática, Universidade Federal do Rio de Janeiro, 2010.

ASTELS, D. *Test Driven Development: A Practical Guide*. [S.l.]: Prentice Hall, 2003.

BALDAUF, M.; DUSTDAR, S.; ROSENBERG, F. *A Survey on Context-Aware Systems*. [S.l.], 2007.

BARNETT, M.; GRIESKAMP, W.; SCHULTE, W.; TILLMANN, N.; VEANES, M. Validating use-cases with the asml test tool. In: *Proceedings of the Third International Conference on Quality Software*. Washington, DC, USA: IEEE Computer Society, 2003. (QSIC '03), p. 238–. ISBN 0-7695-2015-4. Disponível em: <http://dl.acm.org/citation.cfm?id=950789.951243>.

BERTOLINI, C.; MOTA, A. A framework for gui testing based on use case design. In: *Proceedings of the 2010 Third International Conference on Software Testing, Verification, and Validation Workshops*. Washington, DC, USA: IEEE Computer Society, 2010. (ICSTW '10), p. 252–259. ISBN 978-0-7695-4050-4. Disponível em: <http://dx.doi.org/10.1109/ICSTW.2010.37>.

BERTOLINO, A.; FANTECHI, A.; GNESI, S.; LAMI, G.; MACCARI, A. Use case description of requirements for product lines. In: *REPL*. [S.l.: s.n.], 2002.

BERTOLINO, A.; GNESI, S. Use case-based testing of product lines. In: *Proceedings of the 9th European software engineering conference held jointly with 11th ACM SIGSOFT international symposium on Foundations of software engineering*. New York, NY, USA: ACM, 2003. (ESEC/FSE-11), p. 355–358. ISBN 1-58113-743-5. Disponível em: <http://doi.acm.org/10.1145/940071.940120>.

BETTINI, C.; BRDICZKA, O.; HENRICKSEN, K.; INDULSKA, J.; NICKLAS, D.; RANGANATHAN, A.; RIBONI, D. A survey of context modelling and reasoning techniques. *Pervasive Mob. Comput.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 6, n. 2, p. 161–180, abr. 2010. ISSN 1574-1192.

BONIFáCIO, R.; BORBA, P. Modeling scenario variability as crosscutting mechanisms. In: *Proceedings of the 8th ACM international conference on Aspect-oriented software development*. New York, NY, USA: ACM, 2009. (AOSD '09), p. 125–136. ISBN 978-1-60558-442-3. Disponível em: <http://doi.acm.org/10.1145/1509239.1509258>.

BUCHHOLZ, S.; HAMANN, T.; HUBSCH, G. Comprehensive structured context profiles (cscp): Design and experiences. In: *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*. [S.l.: s.n.], 2004. p. 43–47.

CHATTERJEE, R.; JOHARI, K. A prolific approach for automated generation of test cases from informal requirements. *SIGSOFT Softw. Eng. Notes*, ACM, New York, NY, USA, v. 35, n. 5, p. 1–11, out. 2010. ISSN 0163-5948.

CHEN, H. *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*. Tese (Doutorado) — University of Maryland, 2004.

CHEVERST, K.; DAVIES, N.; MITCHELL, K.; FRIDAY, A.; EFSTRATIOU, C. Developing a context-aware electronic tourist guide: some issues and experiences. In: *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2000. (CHI '00), p. 17–24. ISBN 1-58113-216-6.

CHOI, W.; KANG, S.; CHOI, H.; BAIK, J. Automated generation of product use case scenarios in product line development. In: *Proceedings of the 8th IEEE International Conference on Computer and Information Technology*. [S.l.: s.n.], 2008.

COCKBURN, A. *Writing Effective Use Cases*. [S.l.]: Addison Wesley, 2001.

DEVMEDIA. *Artigo Engenharia de Software - Introdução a Teste de Software*. 2012. Disponível em http://www.devmedia.com.br/artigo-engenharia-de-software-introducao-a-teste-de-software/8035. Último acesso em dezembro de 2012.

DEY, A. K. Understanding and using context. *Personal Ubiquitous Comput.*, Springer-Verlag, London, UK, UK, v. 5, n. 1, p. 4–7, jan. 2001. ISSN 1617-4909. Disponível em: <http://dx.doi.org/10.1007/s007790170019>.

EDWIN, O. O. *Testing in Software Product Lines*. Dissertação (Mestrado) — School of Engineering, Blekinge Institute of Technology, 2007.

ENGSTRöM, E.; RUNESON, P. Software product line testing - a systematic mapping study. *Inf. Softw. Technol.*, Butterworth-Heinemann, Newton, MA, USA, v. 53, n. 1, p. 2–13, jan. 2011. ISSN 0950-5849. Disponível em: <http://dx.doi.org/10.1016/j.infsof.2010.05.011>.

ERIKSSON, M.; BÖRSTLER, J.; BORG, K. Marrying features and use cases for product line requirements modeling of embedded systems. In: *Proceedings of the Fourth Conference on Software Engineering Research and Practice in Sweden*. [S.l.: s.n.], 2004. p. 73–82.

FANTECHI, A.; GNESI, S.; JOHN, I.; LAMI, G.; DÖRR, J. Elicitation of use cases for product lines. In: *Proceedings of the 5th International Workshop on Product Family Engineering*. [S.l.: s.n.], 2003. p. 152–167.

FERNANDES, P.; WERNER, C.; TEIXEIRA, E. An approach for feature modeling of context-aware software product line. *Journal of Universal Computer Science*, v. 17, n. 5, p. 807–829, mar. 2011.

FERNANDES, P. C. C. *Ubifex: uma abordagem para modelagem de características de linha de produtos de software sensíveis ao contexto*. Dissertação (Mestrado) — COPPE, UFRJ, 2009.

FITNESSE. *Ferramenta de teste de aceitação*. 2012. Disponível em http://fitnesse.org/. Último acesso em dezembro de 2012.

FRAKES, W. B.; KANG, K. Software reuse research: Status and future. *IEEE Trans. Softw. Eng.*, IEEE Press, Piscataway, NJ, USA, v. 31, n. 7, p. 529–536, jul. 2005. ISSN 0098-5589. Disponível em: <http://dx.doi.org/10.1109/TSE.2005.85>.

GALLINA, B.; GUELFI, N.; MONNAT, A.; PERROUIN, G. *A Template for Requirement Elicitation Document of Software Product Lines*. [S.l.], 2006.

GOMAA, H. *Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 2004. ISBN 0201775956.

GUTIERREZ, J.; ESCALONA, M.; MEJIAS, M.; TORRES, J.; CENTENO, A. A case study for generating test cases from use cases. In: *Research Challenges in Information Science, 2008. RCIS 2008. Second International Conference on*. [S.l.: s.n.], 2008. p. 209 –214.

GUTIÉRREZ, J. J.; ESCALONA, M. J.; MEJÍAS, M.; TORRES, J. An approach to generate test cases from use cases. In: *Proceedings of the 6th international conference on Web engineering*. New York, NY, USA: ACM, 2006. (ICWE '06), p. 113–114. ISBN 1-59593-352-2.

HADAR, I.; KUFLIK, T.; PERINI, A.; REINHARTZ-BERGER, I.; RICCA, F.; SUSI, A. An empirical study of requirements model understanding: Use case vs. tropos models. In: *Proceedings of the 2010 ACM Symposium on Applied Computing*. New York, NY, USA: ACM, 2010. (SAC '10), p. 2324–2329. ISBN 978-1-60558-639-7. Disponível em: <http://doi.acm.org/10.1145/1774088.1774569>.

HOLLANDER, M.; WOLFE, D. A. *Nonparametric Statistical Methods*. 2nd edition. ed. [S.l.]: John Wiley & Sons, 1999.

INDULSKA, J.; SUTTON, P. Location management in pervasive systems. In: *Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003 - Volume 21*. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2003. (ACSW Frontiers '03), p. 143–151. ISBN 1-920682-00-7.

JIN-HUA, L.; QIONG, L.; JING, L. The w-model for testing software product lines. In: *Proceedings of the 2008 International Symposium on Computer Science and Computational Technology - Volume 01*. Washington, DC, USA: IEEE Computer Society, 2008. (ISCSCT '08), p. 690–693. ISBN 978-0-7695-3498-5.

JOHN, I.; MUTHIG, D. Product line modeling with generic use cases, splc-2 workshop on techniques for exploiting commonality through variability. In: *Management, Second Software Product Line Conference*. [S.l.: s.n.], 2002.

KAMSTIES, E.; POHL, K.; REIS, S.; REUYS, A. Testing variabilities in use case models. *Software Product-Family Engineering Lecture Notes in Computer Science*, v. 3014, p. 6–18, 2004.

KANG, K. C.; COHEN, S. G.; HESS, J. A.; NOVAK, W. E.; PETERSON, A. S. *Feature-Oriented Domain Analysis (FODA) - Feasibility Study*. [S.l.], 1990.

KITCHENHAM, B. *Procedures for Performing Systematic Reviews*. [S.l.], 2004.

KRAMER, R.; MODSCHING, M.; SCHULZE, J.; HAGEN, K. ten. Context-aware adaptation in a mobile tour guide. In: *Proceedings of the 5th international conference on Modeling and Using Context*. Berlin, Heidelberg: Springer-Verlag, 2005. (CONTEXT'05), p. 210–224. ISBN 3-540-26924-X, 978-3-540-26924-3.

LEE, J.; KANG, S.; LEE, D. A survey on software product line testing. In: *Proceedings of the 16th International Software Product Line Conference - Volume 1*. New York, NY, USA: ACM, 2012. (SPLC '12), p. 31–40. ISBN 978-1-4503-1094-9.

LIMA, A. d. S. *UML 2.0: Do requisito à solução*. 4th. ed. [S.l.]: Editora Érica, 2009.

LINDEN, F. J. v. d.; SCHMID, K.; ROMMES, E. *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007. ISBN 3540714367.

LU, H. *A software testing framework for context-aware applications in pervasive computing*. Tese (Doutorado) — Universidade de Hong Kong, 2009.

MARINHO, F.; COSTA, A.; LIMA, F.; NETO, J.; FILHO, J.; ROCHA, L.; DANTAS, V.; ANDRADE, R.; TEIXEIRA, E.; WERNER, C. An architecture proposal for nested software product lines in the domain of mobile and context-aware applications. In: *Proceedings of the 2010 Fourth Brazilian Symposium on Software Components, Architectures and Reuse*. [S.l.: s.n.], 2010. (SBCARS '10), p. 51–60. ISBN 978-0-7695-4259-1.

MCGREGOR, J. D. *Testing a software product line*. [S.l.], 2001.

MOBILINE. *Uma Linha de Produto de Software Móvel e Sensível ao Contexto*. 2012. Disponível em http://mobiline.great.ufc.br/. Último acesso em dezembro de 2012.

MUSTAFA, B. A. An experimental comparison of use case models understanding by novice and high knowledge users. In: *Proceedings of the 2010 conference on New Trends in Software Methodologies, Tools and Techniques: Proceedings of the 9th SoMeT*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2010. p. 182–199. ISBN 978-1-60750-628-7.

MYERS, G. *The Art of Software Testing*. 2th. ed. [S.l.]: John Wiley & Sons, 2004.

NEBUT, C.; FLEUREY, F.; TRAON, Y. L.; JéZéQUEL, J. M. A requirement-based approach to test product families. In: *Proceedings of the Fifth Workshop on Product Family Engineering (PFE-5)*. Siena, Italy: [s.n.], 2003.

.NET, F. *Plataforma de desenvolvimento para construção de aplicações para o Windowns*. 2012. Disponível em http://www.microsoft.com/net. Último acesso em dezembro de 2012.

NETO, C. R. L. *SPLMT-TE: A Software Product Lines System Test Case Tool*. Dissertação (Mestrado) — Universidade Federal de Pernambuco, 2011.

NETO, P. A. d. M. S.; MACHADO, I. d. C.; MCGREGOR, J. D.; ALMEIDA, E. S.; MEIRA, S. R. L. A systematic mapping study of software product lines testing. *Inf. Softw. Technol.*, Butterworth-Heinemann, Newton, MA, USA, v. 53, n. 5, p. 407–423, maio 2011. ISSN 0950-5849. Disponível em: <http://dx.doi.org/10.1016/j.infsof.2010.12.003>.

NORTHROP, L. M. Sei's software product line tenets. *IEEE Softw.*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 19, n. 4, p. 32–40, jul. 2002. ISSN 0740-7459.

NORTHROP, L. M.; CLEMENTS, P. C. *A framework for software product line practice, version 5.0*. [S.l.], 2007.

OSTRAND, T. J.; BALCER, M. J. The category-partition method for specifying and generating fuctional tests. *Commun. ACM*, ACM, New York, NY, USA, v. 31, n. 6, p. 676–686, jun. 1988. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/62959.62964>.

PADOVITZ, A.; LOKE, S. W.; ZASLAVSKY, A. Multiple-agent perspectives in reasoning about situations for context-aware pervasive computing systems. *Trans. Sys. Man Cyber. Part A*, IEEE Press, Piscataway, NJ, USA, v. 38, n. 4, p. 729–742, jul. 2008. ISSN 1083-4427.

PATRICIO, R. F. A. *CEManTIKA CASE: uma Ferramenta de Apoio ao Desenvolvimento de Sistemas Sensíveis ao Contexto*. Dissertação (Mestrado) — Centro de Informática, Universidade Federal de Pernambuco, 2010.

POHL, K.; G., B.; LINDEN, F. van der. *Software Product Line Engineering: Foundations, Principles and Techniques*. [S.l.]: Springer, 2005.

ROCHA, L.; ANDRADE, R. Towards a formal model to reason about context-aware exception handling. In: *Exception Handling (WEH), 2012 5th International Workshop on*. [S.l.: s.n.], 2012. p. 27 –33.

ROUBTSOV, S.; HECK, P. Use case-based acceptance testing of a large industrial system: Approach and experience report. In: *Proceedings of the Testing: Academic & Industrial Conference on Practice And Research Techniques*. Washington, DC, USA: IEEE Computer Society, 2006. (TAIC-PART '06), p. 211–220.

SILVA, F. S. da. *PersonalTVware: uma infraestrutura de suporte a sistemas de recomendação sensíveis ao contexto para TV Digital Personalizada*. Tese (Doutorado) — Escola Politécnica da Universidade de São Paulo, 2011.

SOMé, S.; ANTHONYSAMY, P. An approach for aspect-oriented use case modeling. In: *Proceedings of the Workshop on Early Aspects at ICSE 2008: Aspect-Oriented Requirements Engineering and Architecture Design*. [S.l.: s.n.], 2008.

SOMMERVILLE, I. *Engenharia de Software*. 6th. ed. [S.l.]: Pearson Addison Wesley, 2003.

STRANG, T.; LINNHOFF-POPIEN, C. A context modeling survey. In: *Proceedings of the First International Workshop on Advanced Context Modelling, Reasoning and Management*. [S.l.: s.n.], 2004. p. 33–40.

TSE, T. H.; YAU, S. S.; CHAN, W. K.; LU, H.; CHEN, T. Y. Testing context-sensitive middleware-based software applications. In: *Proceedings of the 28th Annual International Computer Software and Applications Conference - Volume 01*. Washington, DC, USA: IEEE

Computer Society, 2004. (COMPSAC '04), p. 458–466. ISBN 0-7695-2209-2-1. Disponível em: <http://dl.acm.org/citation.cfm?id=1025117.1025539>.

VIEIRA, V.; TEDESCO, P.; SALGADO, A. C. *Modelos e processos para o desenvolvimento de sistemas sensíveis ao contexto*. [S.l.], 2009.

WANG, Z.; ELBAUM, S.; ROSENBLUM, D. S. Automated generation of context-aware tests. In: *Proceedings of the 29th international conference on Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2007. (ICSE '07), p. 406–415. ISBN 0-7695-2828-7. Disponível em: <http://dx.doi.org/10.1109/ICSE.2007.18>.

WANT, R.; HOPPER, A.; aO, V. F.; GIBBONS, J. The active badge location system. *ACM Transactions on Information Systems*, ACM, New York, NY, USA, v. 10, n. 1, p. 91–102, jan. 1992. ISSN 1046-8188.

WOHLIN, C.; RUNESON, P.; HOST, M.; OHLSSON, M.; REGNELL, B.; WESSLEN, A. *Experimentation in Software Engineering: An Introduction*. [S.l.]: Kluwer Academic Publishers, 2000.

YANG, H. *Context-Driven Requirement Analysis and Implementation of Adaptable IS*. Dissertação (Mestrado) — Faculty of Computer Science, University of Magdeburg, 2010.

YE, J.; COYLE, L.; DOBSON, S.; NIXON, P. Using situation lattices to model and reason about context. In: *Proceedings of the Fourth International Workshop on Modeling and Reasoning in Context*. [S.l.: s.n.], 2007.

ZORZO, C. A.; SILVA, R. A. L. da; COLANZI, T.; VERGILIO, S. Aplicação de uma estratégia incremental para o teste de linha de produto de software. In: *Anais do Simpósio Brasileiro de Qualidade de Software*. [S.l.: s.n.], 2011.

APÊNDICE A – TESTE DE COMPREENSÃO PARA O EXPERIMENTO DE TEMPLATES PARA LPS

Nome:

Template:

Caso de Uso:

Q1. Quais as features opcionais (passos opcionais) que você identificou no caso de uso?

Q2. Quais as features alternativas (passos alternativos) você identificou no caso de uso?

Q3. A identificação das features opcionais e alternativas neste template foi:

a) Muito fácil

b) Fácil

c) Neutro

d) Difícil

e) Muito Difícil

Se for o Caso de Uso 01 (extraído de Gomma, 2004)

Q4. O que sempre acontece nos produtos do caso de uso Cozinhar Comida?

a) O Usuário põe a comida no micro-ondas, fecha a porta e já inicia o cozimento.

b) O Usuário põe a comida no miro-ondas e fecha a porta. Senão tiver sensor de peso o cozimento é iniciado. Se tiver, primeiro é verificado se existe ou não um item no micro-ondas.

c) O Usuário põe a comida no micro-ondas e fecha a porta. Em seguida os dois sensores de peso (o booleano e o analógico) são ativados. Os sensores então indicam se há item ou não no micro-ondas.

d) O Usuário põe a comida no micro-ondas e fecha a porta. Em seguida o sensor que ele tiver (booleano ou analógico) é ativado. O sensor então indica se há item ou não no micro-ondas.

e) Todas as anteriores estão incorretas.

Se for o Caso de Uso 02 (extraído de Eriksson et al., 2004)

Q4. Antes de selecionar a quantidade de dinheiro, o Usuário precisa pelo menos:

a) No mínimo ele precisa inserir o PIN.

b) No mínimo ele precisa inserir o cartão com chip na ATM.

c) No mínimo ele precisa inserir a impressão digital OU um exemplo de voz.

d) No mínimo ele precisa inserir o PIN e a impressão digital ou um exemplo de voz.

e) Todas as anteriores estão incorretas.

Se for o Caso de Uso 03 (extraído de Bonifácio e Borba, 2009)

Q4. Segundo o caso de uso descrito:

a) O sistema sempre salva as preferências do Usuário.

b) O sistema sempre envia os produtos via PAC.

c) O sistema permite a compra por meio do carrinho de compras ou pelo produto individual.

d) O sistema disponibiliza frete grátis.

e) Todas as anteriores estão incorretas.

Se for o Caso de Uso 04 (extraído de John e Muthing, 2002)

Q4. Todo produto que tem esse caso de uso ...

a) Tem um regulador de gás.

b) Tem um regulador de distância.

c) Calcula a velocidade e a distância.

d) Comparam aAlvo com aReal.

e) Todas as anteriores estão incorretas.

APÊNDICE B – QUESTIONÁRIO PRÉ EXPERIMENTO DE TEMPLATES PARA LPS

Nome:

Q1. Qual sua experiência em diagramas de casos de uso?

a) Já estudei diagramas de casos de uso, mas nunca usei profissionalmente diagramas de casos de uso.

b) Já estudei diagramas de casos de uso e já usei profissionalmente diagramas de casos de uso por menos de 1 ano.

c) Já estudei diagramas de casos de uso e já usei profissionalmente diagramas de casos de uso entre 1 e 3 anos.

d) Já estudei diagramas de casos de uso e já usei profissionalmente diagramas de casos por mais de 3 anos.

e) Nunca estudei diagramas de casos de uso.

Q2. Qual sua experiência em descrições de casos de uso em linguagem natural?

a) Já estudei descrições textuais de casos de uso, mas nunca usei profissionalmente especificações textuais de casos de uso.

b) Já estudei descrições textuais de casos de uso e já usei profissionalmente especificações textuais de casos de uso por menos de 1 ano.

c) Já estudei descrições textuais de casos de uso e já usei profissionalmente especificações textuais de casos de uso entre 1 e 3 ano.

d) Já estudei descrições textuais de casos de uso e já usei profissionalmente especificações textuais de casos de uso por mais 3 anos.

e) Nunca estudei descrições textuais de casos de uso.

Q3. Qual sua experiência em Linhas de Produto de Software (LPS)?

a) Sei o que são LPS e conheço bem a definição de similaridades, variabilidades e modelo de características.

b) Sei o que são LPS, mas não sei bem o que são similaridades, variabilidades e modelo de características.

c) Não sei o que são LPS.

Q4. Grau de Escolaridade

a) Graduando

b) Graduado

c) Cursando especialização

d) Especialista

e) Mestrando

f) Mestre

g) Doutorando

h) Doutor

Q5. Qual sua profissão atual

APÊNDICE C – QUESTIONÁRIO PÓS EXPERIMENTO DE TEMPLATES PARA LPS

Nome:

Q1. Eu tive tempo suficiente para executar as tarefas

a) Concordo Fortemente

b) Concordo

c) Neutro

d) Discordo

e) Discordo Fortemente

Q2. Os objetivos das tarefas foram perfeitamente claros para mim

a) Concordo Fortemente

b) Concordo

c) Neutro

d) Discordo

e) Discordo Fortemente

Q3. As tarefas que eu executei foram perfeitamente claras para mim

a) Concordo Fortemente

b) Concordo

c) Neutro

d) Discordo

e) Discordo Fortemente

Q4. Qual a Tarefa 01? (ex.: Template 02 - Caso de Uso 01)

Q5. Qual a dificuldade da Tarefa 01 ?

a) Alta

b) Média/Alta

c) Média

d) Média/Baixa

e) Baixa

Q6. Qual a Tarefa 02? (ex.: Template 02 - Caso de Uso 01)

Q7. Qual a dificuldade da Tarefa 02 ?

a) Alta

b) Média/Alta

c) Média

d) Média/Baixa

e) Baixa

Q8. Qual a Tarefa 03? (ex.: Template 02 - Caso de Uso 01)

Q9. Qual a dificuldade da Tarefa 03 ?

a) Alta

b) Média/Alta

c) Média

d) Média/Baixa

e) Baixa

Q10. Qual a Tarefa 04 ? (ex.: Template 02 - Caso de Uso 01)

Q11. Qual a dificuldade da Tarefa 04 ?

a) Alta

b) Média/Alta

c) Média

d) Média/Baixa

e) Baixa

Q12. Quais Templates você utilizou?

a) Template 1 (Bonifacio e Borba)

b) Template 2 (Eriksson et al)

c) Template 3 (Gomma)

d) Template 4 (John e Muthing)

Q13. O template que foi mais útil para compreender as variabilidades e o funcionamento do caso de uso foi

a) Template 1 (Bonifacio e Borba)

b) Template 2 (Eriksson et al)

c) Template 3 (Gomma)

d) Template 4 (John e Muthing)

e) Sem preferência

Q14. Por que você escolheu o template acima?

Q15. Qual estrutura de template foi mais útil para compreender as variabilidades e o funcionamento do caso de uso?

a) Variabilidades descritas juntas com similaridades

b) Variabilidades descritas separadas de similaridades

c) Sem preferência

Q16. O uso de Perguntas no Template 04

a) Auxiliou o entendimento do caso e uso (ajudando a entender onde e porque variava)

b) Prejudicou o entendimento do caso de uso

c) Não ajudou nem atrapalhou

APÊNDICE D – QUESTIONÁRIO PRÉ AVALIAÇÃO PRELIMINAR

Nome:

Q1. Qual sua experiência em descrições de casos de uso em linguagem natural?

a) Já estudei descrições textuais de casos de uso, mas nunca usei profissionalmente especificações textuais de casos de uso.

b) Já estudei descrições textuais de casos de uso e já usei profissionalmente especificações textuais de casos de uso por menos de 1 ano.

c) Já estudei descrições textuais de casos de uso e já usei profissionalmente especificações textuais de casos de uso entre 1 e 3 ano.

d) Já estudei descrições textuais de casos de uso e já usei profissionalmente especificações textuais de casos de uso por mais 3 anos.

e) Nunca estudei descrições textuais de casos de uso.

Q2. Qual sua experiência em Linhas de Produto de Software (LPS)?

a) Sei o que são LPS e conheço bem a definição de similaridades, variabilidades e modelo de características.

b) Sei o que são LPS, mas não sei bem o que são similaridades, variabilidades e modelo de características.

c) Não sei o que são LPS.

Q3. Qual sua experiência em Linhas de Produto de Software Sensíveis ao Contexto (LPSSC)?

a) Sei o que são LPSSC e conheço bem a definição de similaridades, variabilidades, modelo de características e modelo de contexto.

b) Sei o que são LPSSC, mas não sei bem o que são similaridades, variabilidades, modelo de características e modelo de contexto.

c) Não sei o que são LPSSC.

Q4. Você conhece o método de Partição de Categorias?

a) Sim e já usei profissionalmente.

b) Sim, mas nunca usei profissionalmente.

c) Não.

Q5. Qual sua experiência em testes de software?

a) Já estudei sobre testes de software e já usei profissionalmente por mais de 3 ano.

b) Já estudei sobre testes de software e já usei profissionalmente entre 1 e 3 anos.

c) Já estudei sobre testes de software e já usei profissionalmente por menos de 1 ano.

d) Já estudei sobre testes de software mas nunca testei profissionalmente.

e) Nunca estudei testes de software.

Q6. Você já planejou testes a partir de casos de uso?

a) Sim e já fiz isso profissionalmente.

b) Sim, mas nunca fiz isso profissionalmente.

c) Não.

Q7. Qual sua experiência em testes de aplicações sensíveis ao contexto?

a) Já estudei sobre testes de software de aplicações sensíveis ao contexto e já realizei testes profissionalmente em mais de 5 aplicações sensíveis ao contexto.

b) Já estudei sobre testes de software de aplicações sensíveis ao contexto e já realizei testes profissionalmente entre 2 a 5 aplicações sensíveis ao contexto.

c) Já estudei sobre testes de software de aplicações sensíveis ao contexto e já realizei testes profissionalmente em 1 aplicações sensíveis ao contexto.

d) Já estudei sobre testes de software de aplicações sensíveis ao contexto, mas nunca realizei testes profissionalmente.

e) Nunca estudei sobre testes de software de aplicações sensíveis ao contexto.

Q8. Qual sua experiência em testes de LPS?

a) Já estudei sobre testes de software de LPS e já realizei testes profissionalmente em mais de 5 LPS.

b) Já estudei sobre testes de software de LPS e já realizei testes profissionalmente entre 2 a 5 LPS.

c) Já estudei sobre testes de software de LPS e já realizei testes profissionalmente em 1 LPS.

d) Já estudei sobre testes de software de LPS, mas nunca realizei testes profissionalmente.

e) Nunca estudei sobre testes de software de LPS.

Q9. Você conhece o método PLUTO?

a) Sim e já usei profissionalmente.

b) Sim, mas nunca usei profissionalmente.

c) Não.

Q10. Qual sua experiência em testes de LPSSC?

a) Já estudei sobre testes de software de LPSSC e já realizei testes profissionalmente em mais de 5 LPSSC.

b) Já estudei sobre testes de software de LPSSC e já realizei testes profissionalmente entre 2 a 5 LPSSC.

c) Já estudei sobre testes de software de LPSSC e já realizei testes profissionalmente em 1 LPSSC.

d) Já estudei sobre testes de software de LPSSC, mas nunca realizei testes profissionalmente.

e) Nunca estudei sobre testes de software de LPSSC.

Q11. Qual sua escolaridade?

a) Graduando

b) Mestrando

Q12. Qual sua profissão atual?

APÊNDICE E – QUESTIONÁRIO PARA AVALIAÇÃO DO CAPLUC

Nome:

Q1. Quais features ou passos opcionais você identificou no caso de uso?

Caso de Uso 1 - Captura de Contexto	Caso de Uso 2 - Mostra Documentos

Q2. Quais features ou passos alternativos você identificou no caso de uso?

Caso de Uso 1 - Captura de Contexto	Caso de Uso 2 - Mostra Documentos

Q3. Quais features ou passos atuam conforme de mudanças de contexto?

Caso de Uso 1 - Captura de Contexto	Caso de Uso 2 - Mostra Documentos

Q4. Considerando um produto A que tem como features disponíveis o Sensor e o Qr_code e dado o contexto SENSOR AND NOT_SENSORES_AMBIENTES, o segundo passo do caso de uso 01 será:

a) O Sensor retorna a informação de contexto.

b) A Memória retorna a informação de contexto.

c) O leitor QR Code retorna a informação de contexto.

d) O Serviço Externo retorna a informações de contexto.

e) Nenhuma das respostas anteriores.

Q5. No caso de uso 02, considerando o contexto ativo WIFI e LOW_BATTERY, o sistema:

a) Exibi ao Usuário os arquivos de imagem e vídeo.

b) Pode exibir ao Usuário os arquivos de imagem e vídeo.

c) Exibi ao Usuário os arquivos de imagem e informa ao Usuário que os arquivos de vídeos não são exibidos devido a baixa bateria do celular.

d) Pode exibir ao Usuário os arquivos de imagem e pode informar ao Usuário que os arquivos de vídeos não são exibidos devido a baixa bateria do celular.

e) Nenhuma das respostas anteriores.