Um Ambiente para Geração de Cenários de Testes para Linhas de Produto de Software Sensíveis ao Contexto

Ismayle de Sousa Santos[1], Rossana Maria de Castro Andrade[1], Pedro de Alcântara dos Santos Neto[2]

[1]Mestrado e Doutorado em Ciência da Computação (MDCC) – Universidade Federal do Ceará (UFC) – Fortaleza, Ceará, Brasil

[2]Departamento de Computação – Universidade Federal do Piauí (UFPI) – Teresina, Piauí, Brasil

{ismaylesantos,rossana}@great.ufc.br; pasn@ufpi.edu.br

Abstract. *This paper presents an environment for the generation of test scenarios for a Context Aware Software Product Line (SPL), which uses textual specification of use cases with context information. In this environment, we have: a use case template, a method for test scenarios generation and a support tool. In order to verify the benefits of this environment, we conducted a preliminary assessment. The results of this assessment showed evidences that the template favors the understanding of the use cases and that the method increases the tests coverage.*

Resumo. *Este artigo apresenta um ambiente de geração de cenários de testes para uma Linha de Produto de Software Sensível ao Contexto (LPSSC) que utiliza especificações textuais de casos de uso com informações de contexto. Fazem parte deste ambiente: um template de especificação de casos de uso de uma LPSSC, um método de geração de cenários de testes e uma ferramenta de apoio. Para propor o template para uma LPSSC, foi executado um experimento controlado avaliando o entendimento proporcionado pelos templates existentes para descrição de caso de uso para uma LPS. Além disso, conduziu-se uma avaliação preliminar do ambiente proposto e os resultados indicaram evidências de que o template favorece o entendimento do caso de uso de uma LPSSC e que o método aumenta a cobertura dos testes.*

1. Introdução

Segundo Northrop e Clements (2007), uma Linha de Produto de Software (LPS) pode ser definida como um conjunto de sistemas de software que compartilham características comuns e que satisfazem às necessidades de um segmento de mercado, sendo desenvolvidas a partir de um conjunto de artefatos comuns de forma sistemática. Logo, uma LPS pode ser usada para maximizar os benefícios do reuso sistemático e planejado, tais como a melhoria da qualidade e a redução dos custos de desenvolvimento [Almeida et al., 2007], durante o desenvolvimento de várias aplicações do mesmo domínio.

Neste cenário, a atividade de testes é mais complexa do que no caso de uma aplicação porque os testes de uma LPS devem levar em consideração os pontos de

variação da linha e os dois processos de desenvolvimento, Engenharia de Domínio e Engenharia de Aplicação [Edwin, 2007; Ali e Moawad, 2010].

Quando uma LPS é concebida para o desenvolvimento de aplicações sensíveis ao contexto, que são aplicações que adaptam o seu comportamento com base em informações de contexto [Wang *et al*., 2007], ela pode ser chamada de Linha de Produto de Software Sensível ao Contexto (LPSSC) [Fernandes et al. 2011]. O contexto neste caso é qualquer informação que pode 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 [Dey, 2001].

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 et al., 2007]. Logo, testar uma LPSSC envolve lidar ao mesmo tempo com os desafios inerentes aos testes de uma LPS, que precisam, por exemplo, lidar com a variabilidade e os dois processos de desenvolvimento, e com testes de aplicações sensíveis ao contexto, que devem levar em consideração as informações de contexto.

Dado a complexidade envolvida, é importante que existam métodos ou ferramentas para auxiliar a atividade de testes de uma LPSSC. Neste cenário, a geração de testes de uma LPS a partir de requisitos é importante porque reduz o tempo gasto na atividade de testes e possibilita a identificação de defeitos ainda na Engenharia de Domínio, antes que esses sejam propagados para os produtos [Ali e Moawad 2010].

Segundo a revisão sistemática de Alves et al. (2010), os formatos mais utilizados para documentar os requisitos de uma LPS são: textual, *features* e casos de uso. Os casos de uso são interessantes porque não dependem de linguagem ou ferramenta e podem ser usados para guiar a geração de testes. Na literatura, existem vários trabalhos que apoiam a geração de teste a partir dos casos de uso para uma LPS [Bertolino e Gnesi, 2003; Kamties et al., 2004; Ali e Moawad, 2010; Lima Neto et al., 2012]. Em geral, essas abordagens adotam uma análise dos cenários de caso de uso e produzem algum modelo ou máquina de estados para guiar a geração dos testes. Contudo, no caso de uma LPSSC 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, como os de Eriksson et al. (2004) e Gomaa (2004), para especificação textual de casos de uso de uma LPS, não foi encontrada nenhuma proposta de *template* para casos de uso de uma LPSSC.

O presente artigo tem então como objetivo apresentar um ambiente de suporte a geração de cenários testes para linhas de produto de software sensíveis ao contexto, que foi proposto em Santos (2013). Fazem parte deste ambiente:

- Um método para geração de cenários testes para uma LPSSC a partir de requisitos descritos em especificações textuais de casos de uso;
- Um *template* de caso de uso para dar suporte ao método, próprio para a descrição das variabilidades e informações de contexto de uma LPSSC; e

- Uma ferramenta de apoio à modelagem de casos de uso segundo o *template* proposto e a geração dos testes de acordo com o método proposto.

O restante deste trabalho está organizado da seguinte forma. Na Seção 2 são descritos os trabalhos relacionados. Na Seção 3 é apresentado o ambiente proposto para a geração de cenários de testes para uma LPSSC a partir de casos de uso. A avaliação preliminar conduzida é descrita na Seção 4. Por fim, as considerações finais e trabalhos futuros são apresentadas na Seção 5.

2. Trabalhos Relacionados

Para encontrar os trabalhos relacionados foi realizada uma busca na literatura por publicações que tivessem pelo menos um dos seguintes tópicos: i) Especificação de casos de uso para uma LPS; ii) Especificação de casos de uso para aplicações sensíveis ao contexto; e iii) Teste baseado em casos de uso.

Para a realização dessa busca, foram aplicados alguns dos princípios de uma Revisão Sistemática [Kitchenham, 2004], tais como a definição de um processo de seleção dos artigos, da *string* de busca e dos locais de pesquisa. Como locais de buscas utilizou-se o Google[1] e Google Scholar[2] para buscar *grey literature* (e.g., relatório técnicos), bem como as bases ACM DL Library[3] e IEEE Explorer[4], e os anais do Simpósio Brasileiro de Engenharia de Software (SBES), do Simpósio Brasileiro de Componentes e Arquitetura de Sistemas (SBCARS), e do Simpósio Brasileiro de Qualidade de Software (SBQS).

Como exemplos de *templates* de caso de uso identificados para uma LPS, pode-se citar os *templates* de John e Muthig (2002), Gomaa (2004), Eriksson et al (2004) e Bonifácio e Borba (2009). John e Muthig (2002) propõem o uso de *tags* <*variant*> e </*variant*> para identificar pontos de variação dentro da descrição textual do caso de uso. Já o *template* de Gomaa (2004) apresenta uma seção específica para documentar a variação da LPS, indicando a linha do caso de uso onde a variação pode ser inserida. O *template* de Eriksson et al (2004), por sua vez, propõe especificar as variabilidades da LPS com o identificador dos passos do caso de uso (e.g. passos com o mesmo identificador indicam passos alternativos), enquanto que o *template* de Bonifácio e Borba (2009) apresenta as variabilidades por meio de *advices use case.*

Com relação aos trabalhos identificados para modelagem de contexto em casos de uso, destaca-se o trabalho de Yang (2010), que descreve as variações de contexto como fluxos alternativos com restrições de contexto associadas.

Em se tratando do teste baseado em casos de uso, diversas abordagens foram encontradas. Bertolino e Gnesi (2003), por exemplo, apresentam 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 (*Product Line Use Case*), enquanto que Ali e Moawad (2010) usam casos de uso e diagramas de atividade para permitir a geração de testes a partir de casos de uso.

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

Dessa forma, é possível observar que embora tenham sido encontrados vários trabalhos relacionados, nenhum deles tratava de uma LPSSC. Mais detalhes desses trabalhos, bem como a descrição completa de todos os trabalhos relacionados, podem ser encontrados na dissertação de Santos (2013).

3. Proposta

Nesta seção é descrito o ambiente de geração de cenários de testes proposto por Santos (2013), o qual é constituído por um *template* para descrição textual de caso de uso, um método de geração de testes e uma ferramenta de apoio.

3.1. *Template* de Caso de Uso para Linhas de Produto de Software Sensíveis ao Contexto

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 a ideia deste método é utilizar casos de uso descritos em linguagem natural, surgiu a seguinte questão: *como descrever as variabilidades e informações de contexto de uma LPSSC em casos de uso?*

Nesse sentido foi conduzida uma busca na literatura e pelos resultados obtidos (sumarizados na Seção 2) não foi encontrado nenhum *template* para descrição textual de casos de uso para uma LPSSC. Sendo assim, optou-se por propor um *template* de caso de uso que apresentasse elementos próprios para a especificação de informações de contexto e das variabilidades de uma LPSSC.

Para guiar a proposta deste novo *template*, foram analisados os *templates* de caso de uso para uma LPS identificados na pesquisa bibliográfica. Decidiu-se então executar um experimento controlado esses *templates* com o objetivo de extrair conhecimento para guiar a proposta do *template* de caso de uso para uma LPSSC.

Dessa forma, o propósito deste experimento, que seguiu o guia de Wohlin et al. (2000), 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.

Como foram identificados nove *templates*, o custo para fazer o experimento com todos seria oneroso, uma vez que cada *template* exige um treinamento apropriado e participantes que o utilizem. Logo, foram estabelecidos alguns critérios de seleção (e.g. não indicar o produto final dentro da especificação do caso de uso, pois reduz a manutenibilidade) e após avaliação dos *templates* mediante estes critérios, apenas quatro foram selecionados para o experimento: de John e Muthing (2002), de Eriksson et al. (2004), de Gomma (2004), e o de Bonifácio e Borba (2009).

O experimento contou com a participação de 48 voluntários, 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[5].

[5] Grupo de Redes de Computadores, Engenharia de Software e Sistemas. http://great.ufc.br

Com relação as atividades do experimento, primeiro os voluntários responderam um questionário, cujo objetivo era caracterizar o perfil dos participantes. Em seguida, os voluntários participaram de um treinamento sobre os conceitos básicos envolvidos e sobre os *templates* que seriam utilizados nas atividades do experimento. Depois do treinamento, os participantes executaram as tarefas de compreensão, na qual eles deveriam responder um questionário ao tempo que analisavam uma descrição textual de caso de uso especificada em um dos quatro *templates* selecionados. Por fim, os participantes responderam um questionário pós-experimento que possuía, dentre outras questões, a pergunta sobre qual *template* havia sido considerado mais útil.

A análise dos resultados do experimento indicou que os participantes 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 não tenham sido encontradas diferenças significativamente estatística com os resultados relacionados ao *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.

Com a realização do experimento também foram observadas quais as características de cada *template* geravam mais impacto no entendimento dos casos de uso. Os voluntários que selecionaram o *template* do "Eriksson et al." como o melhor, apresentavam como justificativa o fato dele possuir uma descrição simples e objetiva, facilitando a identificação de quais passos do caso de uso eram obrigatórios, opcionais ou alternativos.

Desse modo, de posse do conhecimento das características de cada template avaliado que impactavam no entendimento dos casos de uso, foi proposto um *template* próprio para LPSSCs. Este *template* foi denominado de CAPLUC (acrônimo de *Context Aware software Product Line Use Case template*) [Santos et al. 2013] e é apresentado na Figura 1.

De acordo com o CAPLUC, o caso de uso de uma 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, Pré-condição, Pós-Condição, Passos, Fluxos Alternativos, Sumário de Variações]. A seguir, são descritos mais detalhes dos elementos específicos para tratar das informações de variabilidade e de contexto de uma LPSSC:

- *Categoria de Reuso*: Identifica se o caso de uso é obrigatório, opcional ou alternativo;
- *Restrição de Contexto*: Especifica em qual contexto o caso de uso é aplicável. Esse elemento foi proposto porque em uma LPSSC, mudanças de contexto afetam a configuração dos produtos finais da linha;
- *Passos*: O CAPLUC segue a proposta de Eriksson et al. (2004) de especificar as variabilidades por meio dos indexes, mas também propõe a associação dos passos com os pontos de variação. Além disso, o símbolo "*" deve ser usado para indicar passos que estão relacionados a restrições de contexto; e

- *Sumário de variações*: Neste elemento são sumarizados os pontos de variação que afetam o caso de uso. Para cada ponto de variação deve-se descrever o tipo de variabilidade e as variantes. Além disso, para cada ponto de variação pode-se ter uma questão descritiva que serve tanto para auxiliar o entendimento do caso de uso quanto para apoiar a instanciação dos produtos finais. Cabe ressaltar ainda, que cada variante pode ter uma restrição de contexto associada, indicando em qual contexto aquela variante pode estar ativa. Por fim, no caso das variações opcionais, o CAPLUC permite especificar um comportamento que ocorre na presença da *feature* opcional (COM) e outro que ocorre na ausência dessa *feature* opcional (SEM).

<table>
<tr><th>Elemento</th><th colspan="3">Descrição</th></tr>
<tr><td>Nome</td><td colspan="3">Nome do caso de uso. O nome deve refletir o objetivo do caso de uso</td></tr>
<tr><td>Caso de Uso Estendido</td><td colspan="3">Nome do caso de uso cujo comportamento é estendido por este caso de uso</td></tr>
<tr><td>Ponto de Extensão</td><td colspan="3">Local do caso de uso estendido onde este caso de uso atua</td></tr>
<tr><td>Categoria de Reuso</td><td colspan="3">Especificar se o caso de uso é Obrigatório, Opcional ou Alternativo {colocar o nome dos casos de uso alternativos aqui}</td></tr>
<tr><td>Restrição de Contexto</td><td colspan="3">Descrição da condição de contexto que precisa ser verdadeira para que este caso de uso seja passível de execução</td></tr>
<tr><td>Resumo</td><td colspan="3">Resume o objetivo do caso de uso</td></tr>
<tr><td>Atores</td><td colspan="3">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)</td></tr>
<tr><td>Pré-condição</td><td colspan="3">Especifica uma ou mais condições que devem ser verdadeiras no início do caso de uso</td></tr>
<tr><td>Pós-condição</td><td colspan="3">Especifica a condição que sempre é verdadeira ao final do caso de uso se a sequencia principal foi seguida</td></tr>
<tr><td colspan="2">Passo</td><td>Usuário</td><td>Sistema</td></tr>
<tr><td>[Nome do Ponto de Variação] (Alt) [Nome da Variante]</td><td>Número do Passo</td><td>Ação do Usuário</td><td>Resposta do Sistema</td></tr>
<tr><td colspan="4">Fluxos Alternativos</td></tr>
<tr><td colspan="2">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</td><td colspan="2">Descrição dos passos alternativos</td></tr>
<tr><td colspan="4">Sumário das Variações Alternativas</td></tr>
<tr><td colspan="2" rowspan="3">[Nome do ponto de variação alternativo]: Questão descritiva</td><td>[Nome da Variante Alternativa 1]
(Restrição de Contexto: contexto que deve ser verdadeiro para a variante ser passível de escolha)</td><td>Identificação do passo na forma “Passo X”</td></tr>
<tr><td>[Nome da Variante Alternativa 2]
(Restrição de Contexto: contexto que deve ser verdadeiro para a variante ser passível de escolha)</td><td>Identificação do passo na forma “Passo X”</td></tr>
<tr><td>[Nome da Variante Alternativa N]
(Restrição de Contexto: contexto que deve ser verdadeiro para a variante ser passível de escolha)</td><td>Identificação do passo na forma “Passo X”</td></tr>
<tr><td colspan="4">Sumário das Variações Opcionais</td></tr>
<tr><td colspan="2" rowspan="2">[Nome do ponto de variação opcional]: Questão descritiva</td><td>[COM Nome da Variante Opcional]
(Restrição de Contexto: contexto que deve ser verdadeiro para a variante ser passível de escolha)</td><td>Identificação do passo na forma “Passo X”</td></tr>
<tr><td>[SEM Nome da Variante Opcional]
(Restrição de Contexto: contexto que deve ser verdadeiro para a variante ser passível de escolha)</td><td>Identificação do passo na forma “Passo X”</td></tr>
</table>

Figura 1. Proposta do CAPLUC

3.2. Método de Geração de Cenários de Testes para LPSSCs

A principal contribuição deste trabalho é a proposta do ChAPTER (*Context Aware software Product line TEsting geneRation method*), que é um método para geração de cenários de testes para uma LPSSC a partir de descrições textuais de casos de uso.

Com relação a proposta do método, ela foi concebida após análise das abordagens existentes, onde percebeu-se que uma adaptação do método PLUTO (*Product Line Use Case Test Optimisation*) [Bertolino e Gnesi, 2003], para tratar de uma LPSSC, poderia ser interessante porque o PLUTO gera cenários de testes diretamente dos casos de uso sem a necessidade de um modelo intermediário.

Para a geração dos testes, Bertolino e Gnesi (2003) propõem uma extensão do método de Partição de Categorias [Ostrand e Balcer, 1988], que é uma abordagem criada originalmente para derivar testes funcionais de especificações escritas em linguagem natural. No método de Partição de Categorias, das unidades funcionais (requisitos) são extraídas as categorias relevantes para teste, bem como as escolhas, que são os valores significantes para cada categoria. A partir daí, o método gera cenários de testes combinando as escolhas possíveis das categorias respeitando as restrições entre escolhas, se houver.

Como o ChAPTER adapta o PLUTO, ele também usa uma adaptação do método de Partição de Categorias. O diferencial do ChAPTER está nos seguintes pontos:

- *Template do caso de uso*: o PLUTO utiliza PLUCs, que permitem a descrições das variações de uma LPS por meio de *tags*. No caso do ChAPTER, como uma LPSSC tem também informações de contexto afetando a configuração e comportamento das aplicações, ele utiliza como entrada casos de uso que descrevem as variabilidades e informações de contexto da linha;
- *Adaptação da técnica de Partição de Categorias*: no PLUTO, as unidades funcionais para teste são os PLUCs e os cenários de casos de uso são sugeridos como uma das categorias. No ChAPTER, por sua vez, os pontos de variação também são considerados como uma categoria e restrições de contexto são associadas as *escolhas* das categorias. Além disso, no ChAPTER as unidades funcionais são casos de uso que descrevem variabilidades e informações de contexto. Tais casos de uso são denominados no ChAPTER de SLICEs *(Software product LIne Contextual use casE*), se forem da linha, e PiECES (*Product contextual usE CasEs*), se forem dos produtos; e
- *Etapas*: O PLUTO é dividido em duas etapas, sendo gerado, na primeira etapa, os cenários de testes da linha e, na segunda, os cenários de testes dos produtos. O ChAPTER, por sua vez, apresenta três etapas, conforme descrito a seguir.

O ChAPTER atua tanto na Engenharia de Domínio quanto na Engenharia de Aplicação de uma LPSSC e propõe a geração de cenários de testes em três etapas, como apresentado na Figura 2.

A primeira etapa inicia com a modelagem dos SLICES. Esses SLICES são casos de uso que descrevem além de funcionalidades, informações de variabilidade e de contexto de uma LPSSC. No âmbito deste trabalho, os SLICES são especificados utilizando o CAPLUC como *template* base.

A partir dos SLICES da LPSSC são gerados os cenários de teste da linha (Nível 1), conforme uma adaptação do método de Partição de Categorias. Tais cenários de testes englobam todas as configurações de todos os produtos da LPSSC em questão.

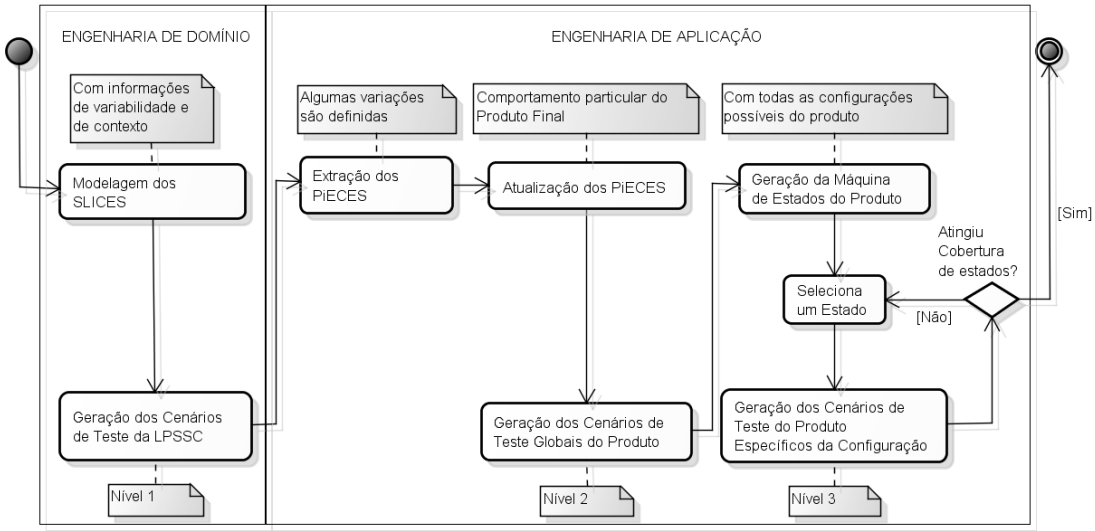

Figura 2. Proposta do ChAPTER

Visto que o número de cenários de teste 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 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.

Em seguida, uma vez que o contexto pode implicar em comportamentos bem específicos para os produtos finais, é previsto no método uma etapa de atualização dos PiECES, na qual tais informações são especificadas.

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. Para os PiECES que foram atualizados, 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.

Por fim, a última etapa corresponde à 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 específico da configuração também podem ser utilizados para garantir que as funcionalidades que executam 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, como 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 podem ser refinados na Engenharia de Aplicação.

3.3. Ferramenta

A terceira contribuição deste trabalho diz respeito a uma ferramenta *web* de apoio ao uso do *template* e do método que foram apresentados nas subseções 3.2 e 3.3, respectivamente. A ferramenta é denominada de ToChAPTER, como referência ao método ChAPTER e 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 cenários de testes dos produtos; e iii) atualização dos PiECES e geração dos cenários de testes para as configurações do produto final.

A arquitetura dessa ferramenta é composta basicamente por três módulos 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.

Quanto aos testes gerados, esses são criados em formatos próprios para a visualização na FitNesse [FITNESSE, 2014]. A escolha dessa ferramenta se deu porque ela é uma ferramenta de teste de aceitação gratuita bastante conhecida entre a comunidade de testadores. 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.

Na Figura 3 é apresentado um exemplo de cenário de teste gerado para o PiECE “Captura Contexto”. No exemplo dessa figura, o cenário indica um teste que exercita 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.

Como a ToChAPTER só gera os cenários de testes, para executar os testes na FitNesse ainda é preciso implementar os casos de testes e a *Fixture*, que é um código Java que faz a ligação entre os testes especificados na FitNesse e o sistema sob teste.

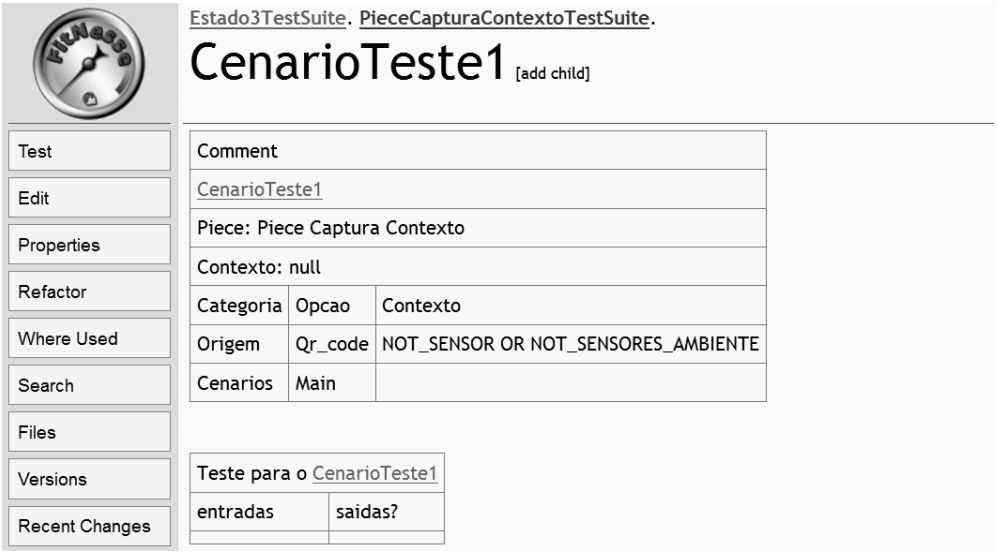

Figura 3. Exemplo de cenário de teste gerado pela ToChAPTER

4. Avaliação Preliminar

Para avaliar os resultados dessa pesquisa, conforme descrito em mais detalhes nas próximas subseções, utilizou-se a LPSSC Mobiline [MOBILINE, 2014]. Essa linha foi desenvolvida pelo GREat e é caracterizada pelo seu foco em aplicações móveis e sensíveis ao contexto. Uma aplicação resultante da LPSSC Mobiline é o GreatTour [Lima et al. 2013], que é uma aplicação móvel e sensível ao contexto desenvolvida para guiar os visitantes do laboratório do GREat.

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.

No total participaram dessa avaliação 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 uma LPS.

Com relação as atividades desta avaliação, primeiramente os voluntários responderam um questionário para caracterizar o perfil dos participantes de maneira semelhante ao que foi feito no experimento dos *templates*. Em seguida, foi ministrado um treinamento sobre o ambiente proposto, bem como sobre alguns conceitos básicos de LPS, LPSSC e casos de uso.

Após o treinamento, os participantes realizaram duas tarefas de compreensão (Tarefa 01 e Tarefa 02) para avaliar o CAPLUC. Em cada uma dessas tarefas, os participantes responderam um questionário de quatro questões analisando um caso de uso modelado no CAPLUC. Para avaliar o desempenho do CAPLUC, foram coletados o tempo gasto para responder o questionário, bem como a quantidade de respostas corretas. Após essas duas tarefas os voluntários foram indagados sobre os benefícios do CAPLUC para o entendimento do caso de uso.

Por fim, para avaliar o método e a ferramenta, a Tarefa 03 da avaliação preliminar era especificar testes a partir de casos de uso. Nesta tarefa, 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 especificar testes para dois casos de uso de uma LPSSC (os mesmos da avaliação do *template*) com base na experiência particular e sem usar a ferramenta ToChAPTER ou método ChAPTER. Já os participantes do Grupo B especificaram casos de testes a partir de cenários de testes gerados pela ToChAPTER. Para avaliar a ferramenta foi coletado a opinião dos voluntários quanto a facilidade de uso e interface enquanto que para avaliar o método foi registrado o número de testes especificados e o tempo gasto para especifica-los.

A seguir são apresentados os principais resultados desta avaliação. Mais detalhes desta avaliação preliminar estão disponíveis na dissertação [Santos 2013].

4.1. Resultados da avaliação do CAPLUC

Na Tabela 1 são apresentados os resultados coletados com relação as tarefas de compreensão 01 e 02. Nesta tabela as respostas corretas são sinalizadas com “1”, e as incorretas com “0”.

Tabela 1. Resultados das Tarefas 01 e 02

Voluntário	Tarefa 01					Tarefa 02				
	Q1	Q2	Q3	Q4	Tempo (min)	Q1	Q2	Q3	Q4	Tempo (min)
V1	1	1	1	0	*10*	1	1	1	0	*7*
V2	1	1	1	1	*6*	1	1	1	1	*6*
V3	1	1	1	1	*5*	1	0	1	0	*4*
V4	1	1	0	1	*8*	1	1	1	1	*7*
V5	1	1	1	1	*5*	1	1	1	0	*4*
V6	1	0	0	0	*15*	0	0	1	0	*10*

Para a Tarefa 01, dois (V1 e V6) 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 (V4) 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.

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 uma análise do ocorrido, percebeu-se que a pequena diferença existente entre as alternativas da questão final desta tarefa 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, 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.

Logo, os resultados da avaliação, por meio dos dados coletados e das respostas dos voluntários, representam indícios de que o CAPLUC 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.

4.2. Resultados da avaliação do ChAPTER

A avaliação do método foi feita considerando os resultados da Tarefa 03. Nesta tarefa os dois grupos de voluntários especificaram casos de testes para dois casos de uso da linha Mobiline [MOBILINE 2014]. Os resultados são apresentados na Tabela 2.

Um aluno de graduação do Grupo B desistiu do experimento nesta atividade. Os outros dois voluntários do Grupo B (V3 e V4), que usaram o método e a ferramenta, descreveram 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 do produto, gerados pela ToChAPTER, um conjunto de casos de testes.

Tabela 2. Resultados dos voluntários para a Tarefa 03

Voluntário	Tempo	Testes	Cobertura (%)
V1	38	1,2,3,7,8,9,12	58,33
V2	31	1,2,5,8,11	41,67
V3	17	1,2,3,4,5,6,7,8,9,10,11,12	100
V4	25	1,2,3,4,5,6,7,8,9,10,11,12	100
V5	24	1,2,5,8,9,10	50

Os voluntários do Grupo A, por sua vez, não utilizaram o ChAPTER. 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 como sendo uma cobertura 100%.

Aplicando o teste de *Kolmogorov-Smirnov* [Hollander e Wolfe, 1999] foi possível constatar que os dados relacionados a tempo e a cobertura seguiam a distribuição normal. Além disso, aplicando o teste de *t de student* para comparação dos resultados relativos ao grupo usando a ToChAPTER e o grupo sem usa-la, foi possível constatar que o tempo gasto foi estatisticamente igual e que as coberturas alcançadas foram consideradas estatisticamente diferentes.

Dessa forma, os resultados revelam indícios de que o método possibilita uma cobertura maior dos casos de testes possíveis. Contudo, vale ressaltar que é necessário um experimento com um número maior de voluntários e uma diversidade maior de casos de uso para confirmar os resultados obtidos.

4.3. Resultados da avaliação da ToChAPTER

A análise da usabilidade da ferramenta ToChAPTER foi feita por meio da observação dos 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.

Com o uso da ferramenta na avaliação preliminar, 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. Contudo, também foram identificados pontos de melhoria para a ferramenta, como numeração automática dos passos e melhores mensagens de erro.

5. Considerações Finais

Este artigo apresentou a proposta de um ambiente para geração de cenários de testes criado com o intuito de auxiliar o teste de uma Linha de Produto de Software Sensível ao Contexto (LPSSC). Esse ambiente é composto por um método de geração de testes, um *template* para descrição 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 a partir de descrições textuais de casos de uso. Para dar suporte ao método foi proposto o CAPLUC (acrônimo de *Context Aware software Product Line Use Case template*), que é um *template* de caso de uso que apresenta elementos para a descrição das variabilidades e informações de contexto de uma LPSSC. A ferramenta chamada de ToChAPTER, por sua vez, permite a modelagem de casos de uso segundo o CAPLUC e a geração de testes de acordo com o ChAPTER.

Uma avaliação preliminar na forma de experimento controlado foi conduzida para avaliar o ambiente de geração de cenários de testes. Os resultados dessa avaliação revelaram indícios iniciais de que o CAPLUC favorece o entendimento, que o ChAPTER permite uma maior cobertura dos testes e que a ferramenta ToChAPTER reduz o tempo gasto para especificação de testes.

Como principais contribuições deste trabalho, acredita que o ambiente proposto: a) Fornece um suporte a atividade de teste de uma LPSSC, reduzindo os custos envolvidos nesta atividade; b) Apoia o planejamento dos testes desde os requisitos; c) Viabiliza meios para rastreabilidade entre requisitos, contexto, código, *features* e testes. Com relação à última contribuição, isso acontece porque o contexto será modelado nos casos de uso e com a geração dos testes a partir desses 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 é possível identificar qual código executa qual caso de uso.

Como trabalhos futuros pretende-se aprimorar a geração dos casos de testes com a definição de critérios de cobertura, bem como conduzir novos experimentos controlados com mais voluntários e casos de uso de diferentes complexidades.

Referências

Ali, M. M.; Moawad, R. (2010) An Approach for Requirements Based Software Product Line Testing. In: *The 7th International Conference on Informatics and Systems*, 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. (2007) *C.R.U.I.S.E: Component Reuse in Software Engineering*. 1. ed. [S.l.]: C.E.S.A.R e-book.

Alves, V.; Niu, N.; Alves, C.; Valença, G. (2010) Requirements Engineering for Software Product Lines: A systematic literature review. *Inf. Softw. Technol*. 52, 8.

Bertolino, A.; Gnesi, S. (2003) Use Case-based Testing of Product Lines. In: *Proceedings of the 9th European Software Engineering Conference* (ESEC). New York, NY, USA: ACM, p. 355–358.

Bonifácio, R.; Borba, P. (2009) Modeling Scenario Variability as Crosscutting Mechanisms. In: *Proceedings of the 8th ACM International Conference on Aspect-oriented Software Development*., NY, USA: ACM, (AOSD '09), p. 125–136.

Dey, A. K. (2001) Understanding and Using Context. *Personal and Ubiquitous Computing,* Springer-Verlag, London, UK, UK, v.5, n. 1, p. 4–7.

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

Eriksson, M.; Börstler, J.; Borg, K. (2004) 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*, p. 73–82.

Fernandes, P., Werner, C., Teixeira, E. (2011) An Approach for Feature Modeling of Context-aware Software Product Line. *Journal of Universal Computer Science*, 17 (5), pp. 807- 829.

FITNESSE (2014) *Ferramenta de teste de aceitação*. Disponível em http://fitnesse.org/. Último acesso em junho de 2014.

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

Hollander, M.; Wolfe, D. A. (1999) *Nonparametric Statistical Methods*, 2nd ed. John Wiley & Sons.

John, I.; Muthig, D. (2002) Product Line Modeling with Generic Use Cases. In *Workshop on Techniques for Exploiting Commonality Through Variability Management*, Second Software Product Line Conference.

Kamsties, E.; Pohl, K.; Reis, S.; Reuys (2004) A. Testing Variabilities in Use Case Models. *Software Product-Family Engineering Lecture Notes in Computer Science*, v. 3014, p. 6–18.

Kitchenham, B. (2004) *Procedures for Performing Systematic Reviews*. Keele University Technical Report TR/SE-0401

Lima, E. R. R.; Araujo, I. L.; Santos, I. S.; Oliveira, T. A.; Monteiro, G. S.; Costa, C. E. B.; Segundo, Z. F. S.; Andrade, R. M. C. (20130. GREat Tour: Um Guia de Visitas Móvel e Sensível ao Contexto. In: *Anais do XII Workshop on Tools and Applications*. 19th Brazilian Symposium on Multimedia and the Web, Salvador, BA.

Lima Neto, C. R.; Almeida, E.S.; Lemos Meira, S.R, (2012) A Software Product Lines System Test Case Tool and Its Initial Evaluation, *IEEE 13th International Conference on Information Reuse and Integration (IRI)*.

MOBILINE (2014) *Uma Linha de Produto de Software Móvel e Sensível ao Contexto*. Disponível em http://mobiline.great.ufc.br/. Último acesso em junho de 2014

Northrop, L. M.; Clements, P. C. (2007) *A framework for software product line practice*, version 5.0. Disponível em https://www.sei.cmu.edu/productlines/tools /framework/. Último acesso em junho de 2014.

Ostrand, T. J. ;Balcer, M. J. (1988) *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

Santos, I. S. (2013) *Um Ambiente para Geração de Cenários de Testes para Linhas de Produto de Software Sensíveis ao Contexto*. Dissertação de Mestrado, Universidade Federal do Ceará.

Santos, I. S.; Santos Neto, P.; Andrade, R. M. C. (2013). A Use Case Textual Description for Context Aware SPL Based on a Controlled Experiment. *In Conference on Advanced Information Systems Engineering (CAiSE) Fórum*, p. 1-8.

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

Wang, Z.; Elbaum, S.; Rosenblum, D. S. (2007) Automated generation of context-aware tests. In: *Proceedings of the 29th International Conference on Software Engineering*. Washington, DC, USA: IEEE Computer Society, p. 406–415.

Wohlin, C.; Runeson, P.; Host, M.; Ohlsson, M.; Regnell, B.; Wesslen, (2000) *A. Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers.