Aplicação de uma Estratégia Incremental para o Teste de Linha de Produto de Software

Carlos Alberto Zorzo[1], Rafaella Aline Lopes da Silva[2], Thelma Elita Colanzi[2,3], Silvia Regina Vergilio[2]

[1]Departamento de Informática – Universidade do Alto Vale do Rio do Peixe (UNIARP)
89500-000 – Caçador – SC – Brasil

[2]Departamento de Informática – Universidade Federal do Paraná (UFPR)
Caixa Postal 19.081 – 81.531-980 – Curitiba – PR – Brasil

[3]Departamento de Informática – Universidade Estadual de Maringá (UEM)
87.020-900 – Maringá – PR – Brasil

zorzo@uniarp.edu.br, rafaella.a.lopes@gmail.com, thelma@din.uem.br, silvia@inf.ufpr.br

Resumo. *No contexto de teste de Linha de Produto de Software (LPS), estratégias incrementais permitem a reutilização de casos de teste entre os produtos. Entretanto, teste de LPS é ainda um tema emergente de pesquisa e na literatura há uma carência de trabalhos relacionados à avaliação de estratégias de teste de LPS. Considerando este fato, no presente trabalho é descrito um estudo de caso utilizando uma estratégia de teste incremental de uma LPS para o domínio de jogos para aparelhos móveis. Para permitir a aplicação da estratégia é proposta uma metodologia de integração com um método de geração de casos de teste a partir de casos de uso, por ser este um modelo bastante utilizado no desenvolvimento de LPS. No estudo, a estratégia se mostrou eficaz quanto ao reúso de casos de teste, e contribuiu para reduzir tempo e esforço da atividade de teste, quando comparada a uma estratégia tradicional.*

Abstract. *In the test of Software Product Line (SPL), incremental strategies enable the systematic reuse of test cases. However, SPL testing is an emergent research area, and in the literature, works that report results from the use of test strategies for SPL are scarce. Considering this fact, this work describes a case study using an incremental test strategy to test a SPL in the domain of games for mobile devices. A methodology is proposed to allow the application of the strategy and its integration with a test case generation method based on use cases, since this is a model largely used in the development of SPL. In the study, it was observed that the strategy allows test cases reuse and reduction of time and effort of the test activity, compared with a traditional strategy.*

1. Introdução

A indústria de desenvolvimento de software procura meios de desenvolver software de forma mais eficiente, principalmente por meio de reúso. Parte-se do princípio de que reusando partes bem especificadas, desenvolvidas e testadas pode-se construir software

em menor tempo e com maior confiabilidade. Neste contexto, Linha de Produto de Software (LPS) constitui uma abordagem de grande importância para viabilizar o reúso e que tem sido adotada na indústria recentemente.

Uma LPS pode ser entendida como um conjunto de sistemas de software que compartilham um conjunto comum e gerenciado de características (*features*) que satisfazem às necessidades específicas de um segmento particular de mercado e que foram desenvolvidas a partir de uma base comum (*core assets*) (van der Linden et al, 2007). O objetivo da LPS é a criação eficiente e sistemática de produtos por meio do reúso de artefatos, sendo que cada produto é definido por uma combinação única de características (Uzuncaova et al, 2010). O desenvolvimento de LPS pode trazer benefícios como o aumento da produtividade, redução de custos e de prazo de entrega. Entretanto, a atividade de teste de LPS torna-se mais crítica e complexa do que no processo tradicional devido ao desenvolvimento de um produto de LPS ocorrer em duas etapas: o desenvolvimento dos componentes para reúso, na engenharia do domínio, e o desenvolvimento do produto a partir dos componentes reusáveis, na engenharia da aplicação.

Lamancha (2009) menciona que não existe um processo definido para o teste de LPS e ressalta que apenas algumas diretrizes para se conduzir esta atividade estão de fato disponíveis. Para McGregor (2001), o teste de LPS deve envolver os ativos centrais, o produto de software específico e suas interações. Pohl e Metzger (2006) relacionam seis princípios essenciais a serem considerados para o teste de LPS. Um destes princípios é o reúso dos artefatos de teste entre os diferentes produtos da LPS.

Edwin (2007) identificou os principais desafios de pesquisa sobre teste de LPS e dentre eles estão (i) estratégias para redução de tempo e custo de teste; (ii) métodos de projeto e geração de casos de teste para LPS; e (iii) o estabelecimento de passos bem definidos e de procedimentos para realizar o teste em LPS. Além disso, uma das questões em aberto é como garantir a qualidade dos produtos da LPS sem a necessidade de realizar todas as atividades de teste para cada produto (Trew, 2005).

Na literatura são encontrados alguns trabalhos relacionados a estes desafios. Para planejamento e geração de dados de teste (desafio ii) destacam-se os trabalhos de McGregor (2001) e de Bertolino e Gnesi (2004). Eles derivam casos de teste genéricos a partir de cenários extraídos dos casos de uso na engenharia de domínio e, na engenharia de aplicação, casos de teste específicos são derivados para o produto a ser testado. Além disto, no trabalho de Tevanlinna et al (2004) são descritas algumas estratégias de teste de LPS (desafio iii), dentre estas, a estratégia incremental parece ser bastante promissora (Uzuncaova et al, 2010). Esta estratégia propõe o reúso dos artefatos de teste entre os produtos da LPS, possibilitando a redução do tempo e custo associado à realização dos testes.

No entanto, apesar da existência dessas estratégias e de várias empresas, como Philips, Nokia e Siemens terem grande experiência em engenharia de LPS, Tevanlinna et al (2004) apontam haver uma clara falta de estudos de caso publicados descrevendo experiências obtidas no teste de LPS, e que forneçam evidências da eficácia de tais estratégias. Uma explicação para isto é que para conduzir tais estudos são necessárias propostas de adaptação dos métodos e critérios de teste existentes ao contexto de LPS para serem aplicados em conjunto com as estratégias a serem avaliadas e este ainda é um tema emergente de pesquisa, sendo que os trabalhos que abordam este tema

(Bertolino e Gnesi 2004; Clements et al 2006; Reyus et al 2005) não envolvem a realização de estudos empíricos de avaliação dos métodos em uma estratégia de teste. Portanto, pode-se concluir que a realização de estudos empíricos para avaliar estas estratégias é fundamental, e este é o foco do presente trabalho.

O trabalho tem como objetivo avaliar a aplicação de uma estratégia incremental de teste para LPS que privilegia reúso, em comparação a uma estratégia dita convencional, que testa cada produto individualmente e que tem sido utilizada por muitas empresas (Jaaksi, 2002). Uma metodologia de aplicação de cada estratégia com um método de geração de casos de teste baseado em caso de uso é proposta, semelhantemente aos trabalhos relativos à geração de casos de teste mencionados acima (McGregor (2001) e Bertolino e Gnesi (2004)). A metodologia é avaliada em um estudo de caso com a LPS Arcade Game Maker (AGM) (SEI, 2010). Neste estudo a estratégia incremental é avaliada considerando reúso e casos de teste gerados a partir dos casos de usos, utilizando o método proposto por Heumann (2001).

O texto está estruturado da seguinte maneira: na Seção 2 aborda-se o teste de LPS. Na Seção 3 é ilustrado o método de geração de dados de teste, adotado no estudo de caso, usando como exemplo a própria LPS do estudo. Na Seção 4 é descrita a metodologia de aplicação de cada estratégia em conjunto com o método de geração descrito na Seção 3. Na Seção 5 são discutidos os resultados alcançados e, por fim, na Seção 6 são apresentadas as conclusões.

2. Teste de Linha de Produto de Software

A atividade de teste de LPS é complexa devido à existência de duas etapas que compõem a engenharia de LPS. Na engenharia de domínio, o desenvolvimento é orientado ao reúso e o conjunto de componentes básicos do domínio da aplicação é construído. Depois, na engenharia de aplicação ocorre a construção dos produtos da LPS a partir do reúso dos componentes já desenvolvidos. Neste cenário, devem ser testados tanto os componentes básicos do domínio da aplicação quantos os produtos construídos a fim de descobrir e corrigir erros e, com isso, proporcionar indicações sobre a confiabilidade e a qualidade dos produtos da LPS.

Uma dificuldade encontrada no teste de LPS diz respeito ao teste dos componentes básicos criados na engenharia de domínio, visto que os testes de integração e de sistema, de acordo com o modelo V não são factíveis já que a tentativa de se testar diferentes combinações dos componentes do domínio pode levar a um crescimento exponencial das configurações de teste (Tevanlinna et al 2004). Além disso, a qualidade e corretude dos componentes básicos são de extrema importância visto que seus defeitos serão propagados para os vários produtos da LPS.

A estratégia mais fácil de implementar e que tem sido utilizada por muitas empresas é testar cada produto separadamente após a engenharia de aplicação, utilizando métodos convencionais de teste de produtos individuais. Esta estratégia é conhecida por estratégia de teste de Produto por Produto (Tevanlinna et al 2004). Apesar de essa estratégia não ter sido projetada especificamente para LPS, e de não explorar benefícios associados ao reúso, ela é bastante adotada devido a sua simplicidade, e por garantir a qualidade do teste, visto que cada produto é testado intensivamente. Existem relatos de aplicação desta estratégia pela Nokia no desenvolvimento de uma LPS para navegadores móveis (Jaaksi, 2002).

A fim de considerar reúso, a estratégia incremental parece ser mais promissora por permitir que casos de teste criados para um produto da LPS sejam reutilizados para o teste dos demais produtos. A estratégia de teste incremental propõe o desenvolvimento e aplicação de um conjunto de casos de teste para um primeiro produto da LPS. À medida que outros produtos são desenvolvidos, cada um deles será testado usando técnicas de teste de regressão, e novos casos de teste serão desenvolvidos somente para explorar as características específicas desse produto, reaproveitando os casos de teste comuns aos produtos da LPS. A filosofia desta estratégia está presente no trabalho de Uzuncaova et al (2010). Eles desenvolveram uma abordagem incremental para geração de dados de teste na qual cada produto é associado a uma composição de características especificadas como uma especificação formal. O conjunto de teste cresce incrementalmente a cada grupo de características selecionadas para a geração de casos de teste, isto é a partir de especificações parciais de um produto a ser testado. Este tipo de geração apresentou um ganho e muitas vantagens sobre o uso convencional. Entretanto, o tipo de especificação é mais restrito. Esta limitação está relacionada à outra questão relativa ao uso prático e a avaliação de tais estratégias que diz respeito à adaptação de critérios e abordagens de teste convencionais ao contexto de LPS. Um desafio é a geração de dados de teste (Tevalinna et al 2004).

Propor soluções para este desafio é ainda um tema emergente de pesquisa. Os trabalhos da literatura relacionados, em geral utilizam o modelo de casos de uso para a geração dos casos de teste (Bertolino e Gnesi 2004; Clements et al 2006; Reyus et al 2005; Jaaksi 2002; McGregor 2001), talvez porque este modelo sempre está disponível durante o desenvolvimento de LPS, e já nas fases iniciais. Entretanto, os trabalhos em geração de dados de teste, não procuram avaliar seus métodos aplicando-os com as estratégias mencionadas acima.

Dada esta limitação, nas próximas seções é descrita uma metodologia de integração de um método de geração de casos de teste a partir de casos proposto por Heumann (2001) com ambas as estratégias de teste de LPS: produto por produto e incremental.

3. Método de Geração de Dados de Teste Adotado

Uma das primeiras tarefas do teste de software é a criação dos casos de teste. Um caso de teste consiste de um par de elementos formado por um dado de teste (elemento do domínio de entrada de um programa) e o resultado esperado para a execução do programa ou modelo a partir daquele dado de teste. Neste trabalho propõe-se a utilização do método de geração de dados de teste proposta por Heumann (2001) para o teste de LPS. A geração de casos de teste a partir de casos de uso ocorre em 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.

Para a criação dos casos de teste, os cenários para cada um dos casos de uso do sistema são identificados. Um cenário pode ser entendido como um caminho completo pelo caso de uso, combinando seu fluxo básico com os fluxos alternativos. Após a identificação dos cenários dos casos de uso parte-se para a definição dos casos de teste. Para isso, deve-se analisar os cenários e identificar as condições ou elementos requeridos para sua execução. Pode-se usar uma representação matricial na qual as linhas correspondem aos cenários, as colunas correspondem à combinação dos fluxos

básico e alternativo, incluindo-se também uma coluna com a saída esperada. Por fim, a matriz de casos de teste deve ser revisada para identificar testes redundantes ou ausentes e, uma vez validada, as células devem ser preenchidas com valores válidos e inválidos para os elementos requeridos na execução de cada um dos cenários especificados.

Para exemplificar o uso do método, utiliza-se a LPS do estudo de caso descrito na Seção 5, a Arcade Game Maker (AGM) (SEI, 2010). A AGM é uma LPS para o domínio de jogos para aparelhos móveis, criada para experimentos e suporte ao aprendizado de LPS. Ela é composta por três produtos constituídos por jogos distintos: Pong, Bowling e Brickles. O diagrama de casos de uso da AGM é mostrado na Figura 1. Os três produtos da LPS são constituídos pelos casos de uso *Play Game*, *Save Score*, *Save Game*, *Exit Game, Check Previous Best Score, Inicialization e Animation Loop*. Além disso, cada produto especializa o caso de uso *Play Game*, assim, o produto Pong possui o caso de uso *PlayPong*, que no Bowling é substituído pelo *PlayBowling* e no Brickles é substituído pelo *PlayBrickles*. Neste mesmo contexto pode-se observar que os três produtos possuem vários cenários em comum, o que viabiliza a utilização da estratégia incremental.

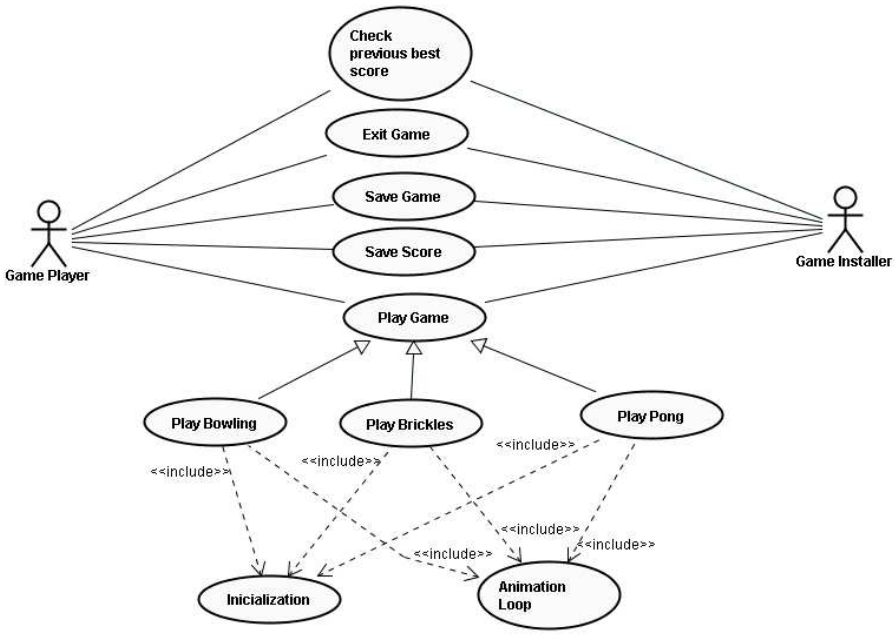

Figura 1: Diagrama de casos de uso da LPS Arcade Game Maker (SEI, 2010)

A aplicação do método de derivação dos casos de teste é exemplificada por meio do caso de uso *Exit Game*, cujo fluxo de execução é mostrado na Tabela 1. Com base na identificação dos fluxos de execução, o segundo passo na aplicação do método consiste em identificar os cenários dos casos de uso, ou seja, os caminhos completos de execução dos casos de uso. A matriz de cenários do caso de uso *Exit Game* é dada na Tabela 2. Após a identificação dos cenários do caso de uso devem-se identificar os elementos requeridos para a execução desses cenários. A Tabela 3 apresenta a matriz com os elementos requeridos para a execução dos cenários do caso de uso *Exit Game*.

Pode-se ver, na Tabela 3, que a execução dos cenários do caso de uso *Exit Game* implica na seleção de uma opção do menu do sistema indicando a intenção do jogador de sair do jogo. Este interesse também pode ser indicado pela opção de fechar a janela

do jogo. Em ambos os casos, uma janela será apresentada ao usuário para que ele confirme ou não a opção de sair do jogo. Estas três opções são representadas na tabela pelas colunas Seleção MENU, Janela e Prompt, respectivamente. A matriz deveria ser preenchida com os valores V, I ou N/A que indicam, respectivamente, que o valor a ser atribuído ao elemento requerido deve ser válido, inválido ou não aplicável para a execução do teste. No entanto, a Tabela 3 não apresenta valores inválidos (letra I), que no caso da AGM seriam relacionados a teclas do teclado, porque esta LPS trava todas as teclas inválidas. Sendo assim, os valores inválidos não foram derivados para este caso de uso visto que o usuário só poderá interagir com o jogo por meio de opções disponíveis no menu ou na janela. Além disso, na Tabela 3 foi especificada uma coluna com a saída esperada de cada cenário.

Tabela 1: Fluxo de execução do caso de uso *Exit Game*

Ação do Ator	Resposta do Sistema
Fluxo Básico	
1. Seleciona a opção EXIT no menu do sistema	**2.** Mostra uma opção para salvar ou sair do jogo
3. Salva o jogo	**4.** Salva o jogo e sai do programa
Fluxo Alternativo	
1.a Cancela a ação de sair	**2.a** Retorna para o jogo
1.b Inicia a opção de sair clicando no canto superior direito da janela de jogo	**2.b** Mostra uma opção para salvar ou sair do jogo
3.b Salva o jogo	**4.b** Salva o jogo e sai do programa

Tabela 2: Matriz de cenários do caso de uso *Exit Game*

Cenário	Fluxo de Execução
Sair do jogo com jogo salvo (EXIT/SAVE)	Fluxo Básico (1+2+3+4)
Sair do jogo sem jogo salvo (EXIT/EXIT)	Fluxo Básico (1+2+3+4)
Cancelar sair do jogo (EXIT/CANCEL)	Fluxo Básico (1+2+3+4) + Fluxo Alternativo 1a (1a+2a)
Sair do jogo com jogo salvo (Fechar janela/SAVE)	Fluxo Alternativo 1b (1b+2b+3b+4b)
Sair do jogo sem jogo salvo (Fechar janela/EXIT)	Fluxo Alternativo 1b (1b+2b)
Cancelar sair do jogo (Fechar janela/CANCEL)	Fluxo Alternativo 1b (1b+2b+3b+4b) + Fluxo Alternativo 1a (1a+2a)

Tabela 3: Matriz de elementos requeridos para o teste do caso de uso *Exit Game*

Cenário	Seleção MENU	Janela	Prompt	Saída Esperada
Sair do jogo com jogo salvo (EXIT/SAVE)	V	N/A	V	Sair do jogo + jogo salvo
Sair do jogo sem jogo salvo (EXIT/EXIT)	V	N/A	V	Sair do jogo + jogo não salvo
Cancelar sair do jogo (EXIT/CANCEL)	V	N/A	V	Continuar jogo
Sair do jogo com jogo salvo (Fechar janela/SAVE)	N/A	V	V	Sair do jogo + jogo salvo
Sair do jogo sem jogo salvo (Fechar janela/EXIT)	N/A	V	V	Sair do jogo + jogo não salvo
Cancelar sair do jogo (Fechar janela/CANCEL)	N/A	V	V	Continuar jogo

Com base nesta matriz de elementos requeridos para a realização do teste, um conjunto de valores válidos, inválidos e não aplicáveis foi associado aos elementos requeridos para a definição dos casos de teste, conforme mostrado na Tabela 4. Os casos de teste verificam a intenção de sair do jogo por meio dos elementos requeridos Seleção MENU (opção EXIT) ou Janela (opção X). O elemento requerido Prompt confirma ou não a saída do jogo (opções EXIT ou CANCEL), dando também a possibilidade do usuário salvar o jogo (opção SAVE).

Para alguns casos de uso, mais especificamente o *PlayPong, PlayBowling e PlayBrickles*, a combinação de cenários com outros casos de uso se faz necessária em

função do *include*. Assim, após a identificação dos cenários para cada um dos casos de uso envolvidos no relacionamento, novos cenários foram criados por meio da combinação dos cenários específicos de cada um dos casos de uso. Isto é exemplificado para o caso de uso *PlayPong* na Tabela 5.

Tabela 4: Matriz de casos de teste do caso de uso Exit Game

Cenário	Seleção MENU	Janela	Prompt	Saída Esperada
Sair do jogo com jogo salvo (EXIT/SAVE)	EXIT	N/A	SAVE	Sair do jogo + jogo salvo
Sair do jogo sem jogo salvo (EXIT/EXIT)	EXIT	N/A	EXIT	Sair do jogo + jogo não salvo
Cancelar sair do jogo (EXIT/CANCEL)	EXIT	N/A	CANCEL	Continuar jogo
Sair do jogo com jogo salvo (Fechar janela/SAVE)	N/A	X	SAVE	Sair do jogo + jogo salvo
Sair do jogo sem jogo salvo (Fechar janela/EXIT)	N/A	X	EXIT	Sair do jogo + jogo não salvo
Cancelar sair do jogo (Fechar janela/CANCEL)	N/A	X	CANCEL	Continuar jogo

Tabela 5: Matriz de casos de teste para o caso de uso PlayPong

Cenário	Play Pong	Initialization				Play Pong	Animation Loop	Saída Esperada
	Seleção MENU	Erro de Memória	Seleção MENU	Caixa Escolha	Memória Disponível	Mouse Left Click	Seleção Mouse	
Carregar o jogo / erro de memória na instanciação das classes básicas da aplicação	PLAY	Sim	N/A	N/A	N/A	N/A	N/A	Jogo não Executando MSG Erro Memória
Carregar o jogo / instanciação das classes básicas da aplicação / erro na instanciação das classes específicas do jogo	PLAY	Não	LOAD GAME	PONG	Não	N/A	N/A	Jogo não Executando MSG Erro Memória
Carregar o jogo / jogo em execução	PLAY	Não	LOAD GAME	PONG	Sim	Sim	N/A	Jogo Executando
	PLAY	Não	LOAD GAME	PONG	Sim	Não	N/A	Jogo não Executando
Carregar o jogo / jogo em execução / pausa na execução do jogo	PLAY	Não	LOAD GAME	PONG	Sim	Sim	HOLDS DOWN LEFT BUTTON	Jogo em Pausa
	PLAY	Não	LOAD GAME	PONG	Sim	Sim	HOLDS DOWN RIGHT BUTTON	Jogo Executando
	PLAY	Não	LOAD GAME	PONG	Sim	Não	HOLDS DOWN LEFT BUTTON	Jogo não Executando
	PLAY	Não	LOAD GAME	PONG	Sim	Não	HOLDS DOWN RIGHT BUTTON	Jogo não Executando

Esta tabela apresenta novos cenários referentes à combinação dos cenários dos casos de uso *PlayPong*, *Initialization e Animation Loop*, de acordo com o seu fluxo de execução. Por exemplo, o início do jogo, no caso de uso *PlayPong*, invoca o cenário "Cria instância" no caso de uso *Initialization*, que instancia as classes básicas da aplicação e carrega o jogo escolhido, retornando o fluxo para o caso de uso *PlayPong* que, por sua vez, inicia o jogo e invoca o cenário "Animação" no caso de uso *Animation Loop*. Os valores especificados na Tabela 5 definem os casos de teste para o teste dos cenários identificados, incluindo valores válidos, inválidos e não aplicáveis.

4. Metodologia Utilizada

Para aplicar as estratégias foram usados dois diferentes grupos de alunos de pós-graduação. O Grupo 1 aplicou a estratégia produto por produto e o Grupo 2 aplicou a estratégia incremental. Os alunos tinham conhecimentos similares a respeito de LPS e de teste baseado em modelos. Entretanto, para evitar qualquer influência, os grupos foram compostos aleatoriamente.

Depois da divisão, cada grupo estudou sua estratégia e método de geração de dados de teste, sem tomar conhecimento do trabalho do outro grupo. A combinação de casos de uso para a geração dos casos de teste de integração foi padronizada para os dois grupos. Este processo consistiu em gerar casos de teste para cenários incluindo combinações de casos de uso ligados por relacionamentos estereotipados com <<include>> e <<extend>> (Figura 1).

A combinação entre os cenários de diferentes casos de uso tem como objetivo exercitar todas as possibilidades de comportamento que podem ocorrer no software. Por exemplo, considerando um caso de uso A que contém dois cenários: AC1 e AC2 e é relacionado com o caso de uso B que contém três cenários: BC1, BC2 e BC3. Os casos de teste são gerados pela combinação de todas as possibilidades: AC1 com BC1, BC2 e BC3, e AC2 com BC1, BC2 e BC3.

A seguir é proposta a metodologia de aplicação de cada uma das estratégias juntamente com o método de geração descrito na seção anterior.

4.1 Estratégia Convencional

Esta estratégia, também conhecida como teste de produto por produto, foi aplicada segundo os procedimentos ilustrados na Figura 2. É uma estratégia de fácil gerenciamento porque foca em um produto de cada vez. O primeiro produto Pr_1 é testado individualmente com o conjunto de dados de teste T_1. Em T_1 há alguns dados de teste que são comuns aos outros produtos da LPS, Pr_2 e Pr_3, no entanto, eles não são usados no teste destes dois produtos. Os produtos Pr_2 e Pr_3 são testados individualmente com os conjuntos de teste T_2 e T_3.

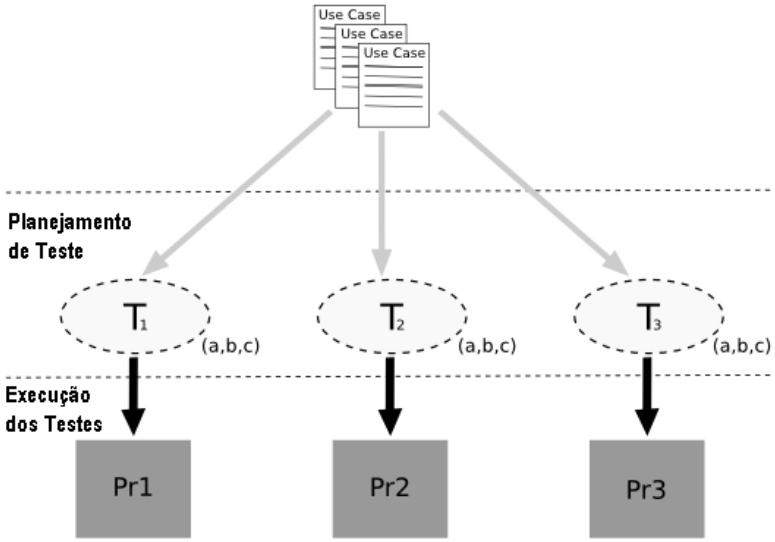

Figura 2: Estratégia Produto por Produto Adotada

Na estratégia produto por produto o processo de geração dos dados de teste foi dividido em três passos: (a) geração dos cenários que contém todas as possíveis combinações entre o fluxo normal e os fluxos alternativos da aplicação descrita nos casos de uso (Tabela 2); (b) identificação dos dados de teste a partir da lista de cenários e os casos de uso. A matriz é gerada depois da identificação dos dados de teste. Esta matriz contém a identificação dos cenários, todas as possíveis entradas e saídas esperadas. Esta matriz não tem valores reais, mas a identificação de valores válidos, inválidos e não aplicáveis (Tabela 3); e (c) identificação dos valores de teste que consiste em substituir os valores válidos e inválidos por valores reais obtendo assim os casos de teste. A cardinalidade dos conjuntos de teste obtidos para cada produto é apresentada na Tabela 6.

Tabela 6: Cardinalidade dos Conjuntos de Teste - Estratégia Produto por Produto

Produto	T_i	% Reúso
Pong	40	0
Bowling	85	0
Brickles	132	0

4.2 Estratégia Incremental

No presente trabalho a metodologia de integração para a estratégia incremental é ilustrada na Figura 3. A estratégia propõe a exploração das semelhanças entre os produtos de uma LPS. Nesta estratégia, o primeiro produto (Pr_1) é testado individualmente com um conjunto de dados de teste T_1 e os demais produtos são testados usando técnicas de teste de regressão. Para se testar um novo produto Pr_2, seleciona-se um subconjunto $Tr_1 \subset T_1$ explorando as similaridades entre os produtos Pr_1 e Pr_2, que é adicionado a um conjunto Ts_2 que explora características específicas de Pr_2, resultando no conjunto T_2. Para testar um novo produto Pr_3, utiliza-se o conjunto T_3, formado pelos conjuntos Tr_1 e Tr_2, derivados a partir de T_1 e Ts_2 respectivamente e que contém similaridades entre os três produtos, e um conjunto específico, Ts_3.

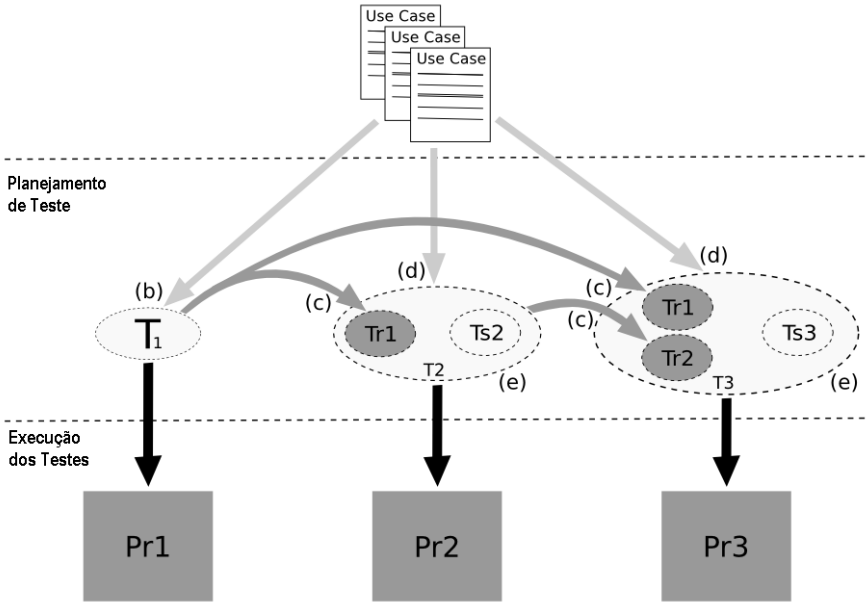

Figura 3: Estratégia Incremental Adotada

A metodologia adotada para a realização do estudo de caso utilizando a estratégia de teste incremental foi conduzida pelos seguintes passos: (a) selecionar a ordem na qual os produtos da LPS seriam testados, (b) geração e execução dos casos de teste do primeiro produto da LPS, (c) reutilização do conjunto de casos de teste existentes que fossem comuns ao produto sob teste (neste caso o segundo produto da LPS), (d) geração de casos de teste específicos para as características do produto sob teste (segundo produto), (e) composição do novo conjunto de casos de teste para o produto em questão (casos de teste comuns + casos de teste específicos), e (f) repetição dos passos (c), (d) e (e) para os demais produtos da LPS. Nesse estudo de caso a ordem de execução dos testes para os produtos da AGM foi: Pong, Bowling e Brickles. A cardinalidade dos conjuntos obtidos em cada etapa é apresentada na Tabela 7.

Tabela 7: Cardinalidade dos Conjuntos de Teste – Estratégia Incremental

Produto	Tr_1	Tr_2	Ts_i	T_i	% Reúso
Pong	-	-	43	43	-
Bowling	35	-	32	67	52,24
Brickles	35	0	56	91	38,47

O conjunto de casos de teste definidos para o produto Pong (T_1) a partir dos casos de uso do produto é constituído de 50 casos de teste. Deste conjunto foram eliminados os casos de teste não executáveis e redundantes, o que resultou em 43 casos de teste executáveis.

Quando da realização dos testes para o produto Bowling, uma análise do modelo de casos de uso deste produto em relação ao modelo de casos de uso do produto Pong permitiu identificar que os casos de uso *Save Score, Save Game, Exit Game, Check Previous Best Score, Inicialization* e *Animation Loop* são comuns entre os dois produtos. Sendo assim, os casos de teste referentes a estes casos de uso constituem o conjunto T_{r1}. A partir dessa análise, os casos de teste de T_{r1} foram reusados para o Bowling. Este reúso implica num conjunto de casos de teste comuns aos dois produtos composto por 35 casos de teste. Os casos de teste referentes ao caso de uso *PlayPong* que faziam parte do conjunto de teste do Pong não foram incluídos no teste do Bowling por serem específicos daquele produto. Por outro lado, o único caso de uso específico para o Bowling é o *PlayBowling* e os casos de teste para este caso de uso foram escritos e incluídos no conjunto de teste deste produto. Novos cenários foram identificados em função do caso de uso *PlayBowling* e sua integração com os casos de uso *Inicialization* e *Animation Loop* e obteve-se um conjunto de 32 casos de teste para verificar as funcionalidades específicas do novo produto, eliminados os casos de teste não executáveis e redundantes. Desta forma, o conjunto de todos os casos de teste T_2 é formado por 67 casos de teste.

A Tabela 8 apresenta um extrato dos novos casos de teste criados para o produto Bowling. Eles são relativos ao cenário “Carrega jogo / jogo em execução”, identificados a partir do caso de uso *PlayBowling* e suas interações com os casos de uso *Inicialization* e *Animation Loop*. Os elementos requeridos para o caso de uso *Inicialization* foram omitidos por limitação de espaço, mas são iguais aos apresentados na Tabela 5 para o Pong. Em comparação com os casos de teste do Pong (Tabela 5) é possível perceber uma quantidade maior de casos de teste para esse cenário em virtude da maior número de elementos necessários à sua execução. Estes elementos estão relacionados às jogadas

1, 2 e extra presentes no caso de uso *PlayBowling* e que constituem características específicas do produto em questão, caracterizando variações existentes entre os produtos da AGM.

Tabela 8: Extrato da matriz de casos de teste para o caso de uso *PlayBowling*

Cenário	Play Bowling	Initialization	Play Bowling	Animation Loop	Play Bowling					Saída Esperada
	Seleção MENU	...	Mouse Left Click	Seleção Mouse	Jogada1		Jogada2		Jogada Extra	
					Mouse	Colide	Mouse	Colide	Colide	
...										
Carrega jogo / jogo em execu-ção	PLAY	...	Sim	N/A	Sim	Sim	Sim	Sim	N/A	Jogo em Execução
	PLAY	...	Sim	N/A	Sim	Sim	Sim	Não	N/A	Jogo em Execução
	PLAY	...	Sim	N/A	Sim	Não	Sim	Sim	N/A	Jogo em Execução
	PLAY	...	Sim	N/A	Sim	Não	Sim	Não	N/A	Jogo em Execução
	PLAY	...	Sim	N/A	N/A	N/A	N/A	N/A	Sim	Jogo em Execução
	PLAY	...	Sim	N/A	N/A	N/A	N/A	N/A	Não	Jogo em Execução
	PLAY	...	Sim	N/A	Sim	Sim	Não	N/A	N/A	Jogo Parado
	PLAY	...	Sim	N/A	Sim	Não	Não	N/A	N/A	Jogo Parado
	PLAY	...	Sim	N/A	Não	N/A	N/A	N/A	N/A	Jogo Parado
	PLAY	...	Não	N/A	N/A	N/A	N/A	N/A	N/A	Jogo Parado

Por fim, para geração dos casos de teste para o produto Brickles a análise do modelo de casos de uso levou ao reúso dos mesmos casos de teste do produto Pong. Uma segunda análise foi realizada verificando a possibilidade de reúso de casos de teste do conjunto de casos de teste específico para as funcionalidades do produto Bowling. No entanto, não foi possível reusar nenhum caso de teste. Na sequência, foram identificados e gerados os novos cenários do caso de uso *PlayBrickles*, específico do produto Brickles, e sua integração com os casos de uso *Inicialization* e *Animation Loop*. Chegou-se ao novo conjunto de casos de teste, Ts_3 para verificar as funcionalidades específicas desse produto. Este conjunto é composto por 56 casos de teste, eliminados os casos de teste não executáveis e redundantes.

No caso de haver a inclusão de um novo produto à LPS, será necessário verificar a possibilidade de reúso de casos de teste dos produtos anteriores, analogamente ao que foi realizado para os produtos Bowling e Brickles.

5. Análise e Discussão dos Resultados

Os dois grupos que aplicaram as duas estratégias utilizaram a mesmo método para a geração dos dados de teste. Ainda assim, os números de casos de teste obtidos nas duas estratégias são diferentes, dada a subjetividade da interpretação dos casos de uso. A Tabela 9 sintetiza a cardinalidade dos conjuntos obtidos para as duas estratégias e seus respectivos percentuais de reúso.

Como era esperado, a aplicação da estratégia de teste produto por produto na AGM não possibilitou o reúso de casos de teste, e com isso, a cada novo produto, foi necessário reescrever os casos de teste pertinentes ao produto em teste.

Tabela 9: Cardinalidade dos Conjuntos de Teste

Produto	Estratégia Produto por Produto	% Reúso	Estratégia Incremental	% Reúso
Pong	40	0	43	-
Bowling	85	0	67	52,24
Brickles	132	0	91	38,47

Ao contrário, um considerável percentual de reúso foi atingido utilizando a estratégia incremental para teste de LPS, uma vez que ela tem como objetivo evitar a necessidade de realizar todas as atividades de teste para cada produto e, com isso, reduzir estrategicamente o tempo e o custo das atividades de teste de LPS. Para analisar os resultados de forma mais detalhada foi calculada a porcentagem de reúso de cada um dos conjuntos de teste. Uma síntese dos resultados é apresentada na Tabela 10.

Tabela 10: Síntese do reúso de casos de teste – Estratégia Incremental

Produto	# Casos de Teste reusados de T_1	% Reúso de T_1	# Casos de Teste reusados de T_2	% Reúso de T_2	% Reúso Total
Bowling	35	81%	-	0%	81%
Brickles	35	81%	0	0%	81%

Nesta tabela pode-se perceber que 35 casos de teste do conjunto T_1 referente ao produto Pong foram reusados nos conjuntos de teste dos outros produtos. Assim, 81% do número de casos de teste deste conjunto foi reusado em outros conjuntos de teste. Esse reúso foi relativo a casos de uso que os três produtos da AGM têm em comum. Nenhum caso de teste que foi criado especificamente para um produto foi reusado para testar outro produto da LPS. Apesar disso, existe grande similaridade entre estes casos de teste específicos de cada um dos três produtos. Assim, embora não tenha ocorrido o reúso completo, a construção dos casos de teste específicos para os produtos Bowling e Brickles foi facilitada em função da sua similaridade com os casos de teste específicos do Pong, o que poderia indicar um reúso parcial.

Considerando o tamanho dos conjuntos de teste dos três produtos, 52% dos casos de teste que fazem parte de T_2 referente ao produto Bowling foram reusados de T_1, como pode ser observado na Tabela 11. Em relação ao conjunto T_3 (Brickles) houve uma taxa de reúso de casos de teste de 38,47%. Desta forma, o esforço de escrita dos casos de teste pode ser reduzido consideravelmente e foi relativo aos casos de teste específicos de cada produto, como mencionado anteriormente.

Tabela 11: Síntese dos conjuntos de teste – Estratégia Incremental

Conjunto de teste	# Casos de Teste Comuns	# Casos de Teste específicos	# Total de casos de teste	% casos de teste reusados	# Casos de teste excluídos
T_1	43	-	43	-	7
T_2	35	32	67	52,24%	7
T_3	35	56	91	38,47%	7

No caso específico da AGM, verificou-se que grande parte dos casos de uso é compartilhada entre todos os produtos da LPS, à exceção da instanciação de um caso de uso específico para a inicialização de cada produto, explicando assim o considerável percentual de reúso de casos de teste. Tal característica pode não se repetir em outras LPSs. Em LPSs nas quais possa haver um menor compartilhamento de casos de uso por todos os produtos o conjunto de casos de teste comuns seria menor, implicando na

diminuição do percentual de reúso. Por outro lado, diferentemente da AGM, em LPSs nas quais existam características específicas compartilhadas por alguns dos produtos da LPS, haveria também o reúso de casos de teste específicos, aumentando assim o percentual de reúso. No entanto, dadas as peculiaridades da abordagem de LPS pode-se concluir que sempre haverá a possibilidade de reúso de casos de teste entre os produtos da LPS, o que possibilita a diminuição do esforço e tempo necessários à geração dos mesmos. Logo, a estratégia incremental cumpre seu objetivo, exigindo menos esforço de teste do que aquele necessário quando da realização de teste com estratégias convencionais.

A ordem para executar o teste dos produtos de uma LPS pode influenciar a taxa de reúso, uma vez que os casos de teste relativos a características variáveis de um produto podem ser reusadas no teste de outro produto que contenha as mesmas características variáveis. Essa influência não foi observada no nosso estudo de caso porque cada produto está associado a somente um jogo. O mesmo não ocorreria se os produtos incluíssem mais de um jogo da AGM.

Uma limitação encontrada na realização deste trabalho diz respeito à disponibilidade de LPSs para a realização de estudos experimentais. A dificuldade em obter LPSs para realizar estudos experimentais também pode ser o motivo da carência de estudos desse teor publicados, como mencionado anteriormente.

6. Conclusão

O presente trabalho apresentou um estudo de caso de aplicação da estratégia incremental para o teste de Linhas de Produto de Software. Tal abordagem propõe o teste individual do primeiro produto da LPS e, para os demais produtos que são gerados para a LPS, a aplicação de técnicas de teste de regressão, reaproveitando os casos de teste dos produtos anteriormente testados. Os casos de teste do estudo de caso apresentado foram gerados a partir dos casos de uso dos produtos, possibilitando que o processo de teste possa se iniciar o mais cedo possível no processo de desenvolvimento do software. O estudo de caso realizado envolve o domínio de aparelhos móveis, mas a estratégia de teste incremental pode ser aplicada a LPS de qualquer domínio.

Quando comparada à estratégia convencional - produto por produto - a estratégia incremental se mostrou promissora e mais apropriada para o teste de LPS visto que tira proveito do reúso sistemático proposto pela abordagem de LPS, diminuindo o custo e o tempo necessários às atividades de teste.

Pretende-se, em trabalhos futuros, usar ferramentas de teste para avaliar a cobertura do conjunto de testes criado usando a estratégia de teste incremental de LPS, além de realizar estudos empíricos a fim de comparar diferentes estratégias de teste.

Referências

Bertolino, A. and Gnesi, S. (2004) PLUTO: A Test Methodology for Product Families. Lecture Notes in Computer Science. Springer-Verlag Heidelberg.

Clements, P. C. , Jones, L. G., McGregor, J. D. e Northrop, L. M. (2006) "Getting There From Here: A Roadmap for Software Product Line Adoption". Communications of the ACM. Vol. 49, Nº 12. December.

Edwin, O. O. (2007) Testing in Software Product Lines. School of Engineering, Master Thesis, Blekinge Institute of Technology, Sweden.

Heumann, J. (2001). Generating Test Cases from Use Cases. The Rational Edge, June. Disponível:http://www.ibm.com/developerworks/rational/library/content/RationalEdge/jun01/GeneratingTestCasesFromUseCasesJune01.pdf . Acesso em 07 Jun 2010.

Jaaksi, A. (2002) Developing mobile browsers in a product line. IEEE Software, 19(4):73–80, July/August.

Lamancha, B. P., Usaola, M. and Piattini, M. (2009) Software Product Line Testing, A systematic review. International Conference on Software, vol. 49, pp. 78 - 81.

McGregor, J. D. (2001). Testing a software product line. Technical Report CMU/SEI-2001-TR-022, Software Engineering Institute, Carnegie Mellon University.

Pohl, K. and Metzger, A. (2006). Software product line testing: Exploring principles and potential solutions. Communications of the ACM, v. 49, n. 12, December.

Reuys, A., Kamsties, E., Pohl, K. e Reis, S. (2005) Model-Based System Testing of Software Product Families, Conference on Advanced Information Systems Eng. (CAISE) 2005, LNCS, pp. 519 - 534.

SEI. (2010). Arcade Game Maker Pedagogical Product Line. Software Engineering Institute, Carnegie Mellon University. Disponível em http://www.sei.cmu.edu/productlines/ppl/. Acesso em 20 Dec 2010.

Tevanlinna, A.; Taina, J. and Kauppinen,R. (2004). Product Family Testing – a Survey. ACM SIGSOFT Software Engineering Notes 29(2), pp. 12.

Trew, T. (2005). Enabling the smooth integration of core assets: Defining and packaging architectural rules for a family of embedded products. SPLC, pages 137–149.

Uzuncaova, E., Khurshid, S., Batory, D. (2010) "Incremental Test Generation for Software Product Lines", IEEE Transactions on Software Engineering, pp. 309-322, May/June.

van der Linden, V.D., Schimd, F. and Rommes, E. (2007) Software Product Lines in Action – The Best Industrial Practice in Product Line Engineering, Springer.