UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

UM MÉTODO DE DESENVOLVIMENTO DE SOFTWARE COMBINANDO LINHA DE PRODUTO DE SOFTWARE COM SCRUM: ESTUDO DE CASO SISTEMA DE NAVEGAÇÃO DE TV

ANTÔNIO FERREIRA DOS SANTOS JÚNIOR

MANAUS-AM
2010

UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

ANTÔNIO FERREIRA DOS SANTOS JÚNIOR

UM MÉTODO DE DESENVOLVIMENTO DE SOFTWARE COMBINANDO LINHA DE PRODUTO DE SOFTWARE COM SCRUM: ESTUDO DE CASO SISTEMA DE NAVEGAÇÃO DE TV

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Amazonas, como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica, área de concentração: Automação de Sistemas.

Orientador:

Prof. Dr. –Ing. Vicente Ferreira de Lucena Júnior

MANAUS
2010

ANTÔNIO FERREIRA DOS SANTOS JÚNIOR

UM MÉTODO DE DESENVOLVIMENTO DE SOFTWARE COMBINANDO LINHA DE PRODUTO DE SOFTWARE COM SCRUM: ESTUDO DE CASO SISTEMA DE NAVEGAÇÃO DE TV

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Amazonas, como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica, área de concentração Automação de Sistemas.

Aprovado em 23 de agosto de 2010.

BANCA EXAMINADORA

Prof. Dr. –Ing. Vicente Ferreira de Lucena Júnior
Universidade Federal do Amazonas – UFAM

Profa. Dra. Tayana Uchôa Conte
Universidade Federal do Amazonas – UFAM

Prof. Dr. Carlos Maurício Serôdio Figueiredo
Fundação Centro de Análise, Pesquisa e Inovação Tecnológica - FUCAPI

Ficha Catalográfica

(Catalogação realizada pela Biblioteca Central da UFAM)

Santos Júnior, Antônio Ferreira dos

S237m Um método de desenvolvimento de software combinando linha de produto de software com *Scrum*: estudo de caso sistema de navegação de TV. - Manaus: UFAM, 2010.

100 f.: il. ; 30 cm

Dissertação (Mestrado em Engenharia Elétrica, área de concentração em Automação de Sistemas) — Universidade Federal do Amazonas, 2010.

Orientador: Prof. Dr. Ing. Vicente Ferreira de Lucena Júnior

1. Televisão digital 2. Software – Desenvolvimento 3. Sistemas de computação 4. I. Lucena Júnior, Vicente Ferreira de (Orient.) II.Universidade Federal do Amazonas III. Título

CDU 004.4’2:621.397(043.2)

Dedico este trabalho à minha família, a meus amigos de infância, a meus amigos do grupo de TV digital da UFAM, ao Prof. Vicente e a todos aqueles que, como eu, são fascinados pela Engenharia de Software.

Agradecimentos

Este trabalho é de grande importância para minha carreira, pois representa o fim de um ciclo de muito aprendizado, chegando a determinar a próxima direção da minha vida. Portanto quero agradecer de coração a todos aqueles que compartilharam comigo diversos momentos aqui na UFAM e nos vários lugares que visitei neste período. Em especial:

A meus amigos do grupo de TV digital Alexandre Pereira, Marcelo Prado, Lady Daiana, Luciano Pinto, Nairon Viana, Orlewilson Bentes, Ricardo Rosa, Vandermi Silva e Wanderlan Carvalho.

A meus amigos do Mestrado Charles Melo, Gisele Lira, Luiz Neres e Vanderson Reis. Aos colegas dos grupos de reuso e de processamento de sinais da COPPE/UFRJ e que conviveram comigo na minha estada no Rio de Janeiro, em especial Celso Carvalho e Mariana Tonin, Diego Fekilis, Eliamara Souza, Gabriel Matos e Rodrigo Santos.

Aos colegas que compartilharam bons momentos em congressos, em especial Denise Alves, Fernanda Lopes, Idriss Tyler, Juan Portillo, Martín Espiña e Vladmir Gonzáles.

A todos os professores das disciplinas que cursei.

A meus amigos de infância, que sempre me deram apoio em várias ocasiões, em especial a Enzo e Lígia Santos, Márcio Torres e Remerson e Lana Mogi, por estarem sempre próximos e por todo apoio dado a mim à minha família durante o mestrado, e, principalmente, enquanto estive no Rio de Janeiro.

À CAPES e à FAPEAM por terem me fornecido bolsas de estudo.

Ao professor Vicente por todo apoio e incentivo.

À toda minha família: minha Mãe e meu Pai, Diogo, Elissa, Eveline, Keilla, Kyssya e Mysea Santos.

E por fim, e o mais importante, a Deus, que sempre esteve comigo nesta caminhada, me abençoando e orientado.

Nosso maior medo não é o de sermos inadequados. Nosso maior medo é o de sermos poderosos além das medidas. É a nossa luz, não nossa escuridão o que mais nos apavora. Perguntamos a nós mesmos: Quem sou eu para ser brilhante, esplêndido, talentoso e fabuloso? Na verdade, eu pergunto: Por que não? Você é um filho de Deus! Bancar o pequeno não serve ao mundo. Nada nos esclarece no sentido de nos diminuirmos para que outras pessoas não se sintam inseguras em torno de nós. Todos temos luz. Ela não está apenas em alguns de nós; e quando deixamos nossa própria luz brilhar, inconscientemente damos a outras pessoas permissão para fazer o mesmo. Quando nos libertamos de nosso próprio medo, nossa presença automaticamente liberta outros.

Nelson Mandela (Prêmio Nobel da Paz).

Resumo

A indústria, para alcançar um número maior de consumidores, tem adotado o conceito de customização em massa, ou seja, lançar produtos similares, mas que atendam a requisitos específicos de vários nichos de clientes. A metodologia que trata de desenvolvimento de software voltado à customização em massa é a engenharia de linha de produto de software (SPLE), a qual busca reutilizar componentes de software de forma sistemática e eficiente resultando nas linhas de produto de software (SPL). *Scrum* é um método ágil usado para gerenciar e controlar o desenvolvimento de produtos e softwares complexos através de um processo iterativo e incremental, que foi criado para introduzir novos produtos no mercado de forma rápida. Esse método tem apresentado ganhos de produtividade de 5 a 10 vezes maior que a média da indústria convencional de desenvolvimento de software. Ambas as metodologias são projetadas para desenvolver produtos de software de forma produtiva, onde a SPLE adota o reuso sistemático de componentes como seu principal princípio, ao passo que o método *Scrum* lança mão de times auto-gerenciados e do manifesto ágil como os seus. Este trabalho apresenta e discute um processo, denominado ScrumPL, que combina ambas as metodologias para o desenvolvimento de SPLs, baseando-se nas entradas e saídas de cada etapa do *Scrum* e da SPLE. Por fim é discutida a aplicação deste novo processo no desenvolvimento de uma linha de produto de software de um sistema de navegação de TV digital interativa voltada para três seguimentos de mercado, três padrões de TV digital (ISDB, DVB e ATSC) e vários idiomas.

Palavras-chave: Linha de Produto de Software, Métodos Ágeis, Scrum, Guia Eletrônico de Programação (EPG), Sistema de Navegação de TV.

Abstract

The industry, to reach more consumers, has adopted the mass customization concept to develop several and different products sharing the same components, in other words, launch similar products attending specific requirements from several market segments. The methodology used to develop software for mass customization is the software product line engineering (SPLE) that uses component reuse concepts in an efficient and systematic way, resulting on several software products sharing a common platform, which are part of a software product line (SPL). Scrum is an agile process that can be used to manage and control complex product and software development by using iterative and incremental practices. It was invented to rapidly drive new products to market, and was designed for hyper productive teams where productivity increases by 5-10 times over industry averages and many collocated teams have achieved this effect. Both, SPLE and Scrum, are designed to develop software products in a productive way, but SPLE has adopted the systematic reuse as its main principle, and Scrum has adopted the self-management teams and agile manifesto. This M.Sc. thesis shows the ScrumPL, a method combining both SPLE and Scrum methodologies based on their input and output needs and the Scrum lifecycle. This process was applied to develop a TV navigation system SPL, with one of the digital TV standards (ISDB-TB, DVB or ATSC), one language, for low-end, mid-end or high-end set-top-box market segment.

Keywords: Software Product Line, Agile Methods, Scrum, Electronic Program Guide (EPG), TV navigation system.

Índice de Figuras

Índice de Quadros

Lista de Siglas

ASPLM	*Agile Software Product Line Methods*
ATSC	*Advanced Television System Comitee*
DSDM	*Dynamic Systems Development Method*
DVB-T	*Digital Vídeo Broadcasting-Terrestrial*
EA	*Processo de Engenharia de Aplicação*
ED	*Processo de Engenharia de Domínio*
EPG	*Electronic Program Guide*
EVO	*Abreviação de Evolutionary Project Management*
FDD	*Feature Driven Development*
HDMI	*High-Definition Multimedia Interface*
HDTV	*High-Definition TV*
ISDB-T	*Integrated Services Digital Broadcasting – Terrestrial*
ISDB-TB	*Integrated Services Digital Broadcasting – Terrestrial Brazilian*
PiP	*Picture in Picture*
PVR	*Personal Video Recorder*
SBTVD	*Sistema Brasileiro de TV Digital*
SPL	*Software product line*
SPLE	*Software product line Engineering*
STB	*Set-Top-Box*
TDD	*Test Driven Development*
UML	*Unified Modeling Language*
USB	*Universal Serial Bus*

Sumário

Capítulo 1- Introdução

O desenvolvimento de software com a melhor qualidade, no menor tempo e custo possíveis, ou seja, buscando produtividade e lucratividade, é uma busca constante de qualquer empresa de tecnologia da informação e/ou desenvolvimento de produtos. Considerando o lançamento de novos produtos em um mercado de eletrônica de consumo cada vez mais competitivo, a pressão pelo menor tempo e custo, e melhor qualidade, é ainda maior.

As empresas fabricantes de produtos de eletrônica de consumo costumam lançar, dentro de um mesmo período, produtos similares, mas com características e preços específicos, sendo cada produto voltado para um determinado público-alvo, realizando, desta forma, uma personalização em massa do produto. Como exemplo tem-se os televisores que possuem o mesmo design, mas tamanhos e preços variados, ou os set-top boxes (STB) de um mesmo fabricante, que possuem o mesmo guia de programação, mas podem vir ou não com entrada HDMI e/ou porta USB, comunicação *Bluetooth*, entre outras características. Esses produtos são desenvolvidos contendo uma parte de hardware e outra de software, sendo que esta última parte está tomando uma fatia cada vez maior de tempo e custo de desenvolvimento.

Uma metodologia que provou fornecer ganhos significativos de qualidade, tempo e custo no desenvolvimento de produtos de software é a engenharia de linha de produto de software – *Software Product Line Engineering*, SPLE (Pohl et al., 2005). Através dessa metodologia se desenvolve um conjunto de aplicações distintas, que compartilham determinadas características, formando uma de linha de produto de software – *Software Produt Line* – SPL.

Outras metodologias que buscam a produtividade no desenvolvimento de software são os métodos ágeis, entre eles *Scrum* (Schwaber e Beedle, 2002), *Extreme Programming* (Beck, 1999), DSDM (2010), EVO (Larman, 2004) e FDD (Palmer, 2003), que compartilham os mesmos princípios, nos quais se dá mais valor a: (a) indivíduos e interações sobre processos e ferramentas, (b) software funcionando sobre documentação compreensiva, (c) colaboração com o cliente sobre negociação de contratos, e (d) responder a mudanças sobre seguir um plano (Beck et. al, 2001). Os métodos ágeis

aplicam desenvolvimento iterativo e incremental, com tempo fixo e com plano adaptável; promovem entrega incremental, e incluem outros valores e práticas que encorajam a agilidade – resposta à mudança de forma rápida e flexível (Larman, 2004).

Tian e Cooper (2006) fazem uma comparação entre os métodos ágeis e a SPLE encontrando, respectivamente, as seguintes diferenças: “colaboração direta com o cliente x colaboração com representantes do cliente bem treinados” e “desenvolvedores usam conhecimento implícito x desenvolvedores seguem planos e processos”, entre outras. Os citados autores também encontraram semelhanças entre os métodos, tais como acomodar mudanças e entregar software com qualidade de forma rápida, entre outras. Mohan, Hamesh e Sugumaran (2010) citam outra diferença importante entre métodos ágeis e SPLE, respectivamente, “objetivos táticos de curto prazo x objetivos estratégicos de longo prazo”.

Sendo assim e apesar dessas diferenças, tem-se buscado criar novos métodos que combinem métodos ágeis e SPLE: um método de linha de produto de software ágil (agile *software product line method* – ASPLM). As vantagens dos ASPLM citadas por Noor et. al (2008) dizem respeito a tornar as SPLs mais suscetíveis a quaisquer mudanças nas necessidades dos clientes, bem como ao desenvolvimento dos mercados. Para Tian e Cooper (2006), um ASPLM pode ser aplicado a sistemas críticos e de segurança, sistemas de larga escala ou de equipes distribuídas, com a vantagem de ser mais simples, flexível e adaptável.

Scrum é um método ágil que pode ser usado para gerenciar e controlar o desenvolvimento de softwares e produtos complexos utilizando práticas iterativas e incrementais. Este processo tem sido usado por projetos simples para mudar a forma como as corporações fazem negócios e aumentar a produtividade de forma significativa, enquanto facilita o desenvolvimento de sistemas empíricos e adaptativos (Schwaber, 2004).

Neste trabalho buscou-se a integração entre o método ágil Scrum e a SPLE. Como resultado foi desenvolvido um processo, denominado ScrumPL, que é baseado nos artefatos de entrada e saída internos e externos, bem como nos princípios, do *Scrum* e da SPLE para o desenvolvimento de SPLs.

Para verificar o funcionamento da metodologia proposta, foi definida uma SPL de um sistema de navegação de TV, possuindo diversas funções que tornam mais fácil ao telespectador selecionar o que deseja assistir ou gravar, tomando como base suas preferências ao assistir TV.

Esta SPL é formada por um conjunto de aplicações que compartilham características comuns. Mas cada uma é destinada a uma determinada região do planeta, sendo personalizada de acordo com as características dessas regiões no que se refere ao padrão de TV digital, ao idioma e aos segmentos de mercado.

Neste trabalho é apresentado o processo ScrumPL e sua aplicação no desenvolvimento desta SPL de um sistema de navegação de TV, bem como a análise crítica dos resultados. No restante deste capítulo são apresentados: a motivação, objetivos gerais e específicos, a metodologia usada e a organização do restante do trabalho.

1.1 Motivação

O mercado atual de eletrônica de consumo tem forçado os fabricantes a lançarem produtos novos para um público grande, exigente e com necessidades distintas, em períodos cada vez mais curtos. Software vem se tornando um componente cada vez mais presente nesses produtos. Desta forma, para lançar um produto em um prazo curto deve-se levar em conta que o software embarcado deve ser concluído o mais rápido possível, com qualidade e atendendo a requisitos variados.

A SPLE atende à premissa de entregar produtos que atendam às necessidades distintas de um público grande através do reuso de componentes (Pohl et al., 2004) e o método *Scrum* atende à premissa de entrega rápida de software (Schwaber, 2004). Entretanto, como o tempo de entrega dos primeiros produtos usando a SPLE é maior comparado ao desenvolvimento desses produtos de forma única (Pohl et al., 2004), surge uma oportunidade de investigar uma forma de reduzir esse tempo através da integração da SPLE com o *Scrum*. Sendo assim, o propósito deste trabalho foi investigar como realizar esta integração.

1.2 Objetivo Geral

Propor uma metodologia que combine o método ágil *Scrum* com os processos de engenharia de domínio e de engenharia de aplicação da SPLE, de modo a reduzir o tempo de desenvolvimento dos primeiros produtos da SPL e, em conseqüência, de todos os produtos subseqüentes. E em seguida, aplicar essa metodologia no desenvolvimento de uma SPL para um sistema de navegação de TV.

1.3 Objetivos Específicos

- Investigar os *frameworks* de SPLE existentes, identificando princípios, regras, etapas, atividades, ferramentas e artefatos, tanto do processo de engenharia de domínio quanto o de engenharia de aplicação, que possam convergir com os definidos pelo *Scrum*, identificando também os pontos que possam divergir.
- Definir um processo que use ambas as metodologias para o desenvolvimento de software, a partir das etapas, regras, atividades, artefatos e princípios convergentes identificados, bem como das ferramentas.
- Desenvolver uma linha de produto de software de um sistema de navegação de TV, cujos requisitos deverão ser obtidos a partir da literatura disponível, usando o processo definido.
- Fazer análises críticas do uso do processo, identificando e relatando pontos fortes e fracos, bem como sugestões de melhoria, indicando situações onde a metodologia é adequada e onde não é, e propondo trabalhos futuros.

Observa-se que não foram definidas atividades para estudo do método ágil *Scrum*. Isto ocorreu em razão deste método ser bastante simples, tendo como principal referência Schwaber e Beedle (2002), sendo complementado em relatos de experiências descritos por Schwaber (2004). Ambas as referências são bem conhecidas pelo autor desta dissertação, que as aplicou em casos práticos descritos em Santos Jr, Silva e Lucena (2008) e Santos Jr e Santos (2009).

1.4 Metodologia

A metodologia usada para o desenvolvimento deste trabalho de pesquisa consistiu em cumprir etapas visando atingir cada um dos objetivos específicos descritos na seção 1.3, através de pesquisa de material bibliográfico em diversas fontes de literatura científica, descrição do processo, especificação das aplicações usando os processos descritos e avaliação dos resultados do uso do processo.

Antes do início desta pesquisa, o autor já havia feito um estudo do método ágil Scrum, aplicando-o em 13 projetos, tendo os resultados sido publicados em Santos Jr., Silva e Lucena (2008) e Santos Jr. e Santos (2010). Sendo assim, esta pesquisa se iniciou com um estudo aprofundado dos fundamentos de SPL definidos por Northrop (2002), do

framework de SPLE de Pohl et al (2005), dos conceitos de análise de domínio definidos por Pietro-Díaz e Arango (1991) e da visão geral do desenvolvimento de linhas de produto tratadas por Nyholm (2002). Este estudo visou o entendimento do desenvolvimento de SPLs, buscando pontos de convergência e divergência com o método *Scrum.*

Na etapa seguinte, foi definido o processo ScrumPL a partir da avaliação dos artefatos de entrada e saída das duas metodologias, tomando cuidado para que não fossem inseridos elementos que interferissem muito na agilidade e no reuso de artefatos, que são os pontos fortes do método *Scrum* e da SPLE, respectivamente.

Definido o processo ScrumPL, o mesmo foi aplicado ao desenvolvimento de uma SPL de um sistema de navegação de TV, com as especificações advindas da literatura sobre guia eletrônico de programação (EPG). O ScrumPL é composto das fases pré-jogo (contendo as etapas planejamento e projeto), desenvolvimento e entrega. Tendo sido cumprida a fase pré-jogo, que resultou na especificação e arquitetura da referida SPL. Ao término da execução destas fases do ScrumPL, foi feita uma avaliação, indicando seus pontos fortes e fracos, sugerindo situações em que poderia ou não ser aplicado.

1.5 Justificativa

A relação entre o desenvolvimento com SPL x desenvolvimento sem SPL, considerando o tempo de entrega e custo de desenvolvimento é demonstrada nas Figuras 1.1 e 1.2, respectivamente.

No desenvolvimento sem SPL, cada software é desenvolvido desde o princípio isoladamente, ou seja, sem haver o compartilhamento de artefatos entre eles, ao passo que no desenvolvimento com SPL é desenvolvida a plataforma contendo os artefatos comuns a todas as aplicações, e, a partir destes artefatos, são formadas as aplicações específicas.

Na Figura 1.1 observa-se que o tempo de entrega de uma SPL é bem maior nos dois primeiros produtos de software, já que os artefatos comuns devem ser desenvolvidos primeiro, mas, a partir do terceiro, o tempo de entrega é significativamente menor, pois muitos artefatos podem ser reusados para cada novo produto. A linha representando o tempo de desenvolvimento sem SPL é constante apenas porque a nível de comparação é razoável considerá-lo assim (Pohl et al., 2005), mas em situações reais há variações.

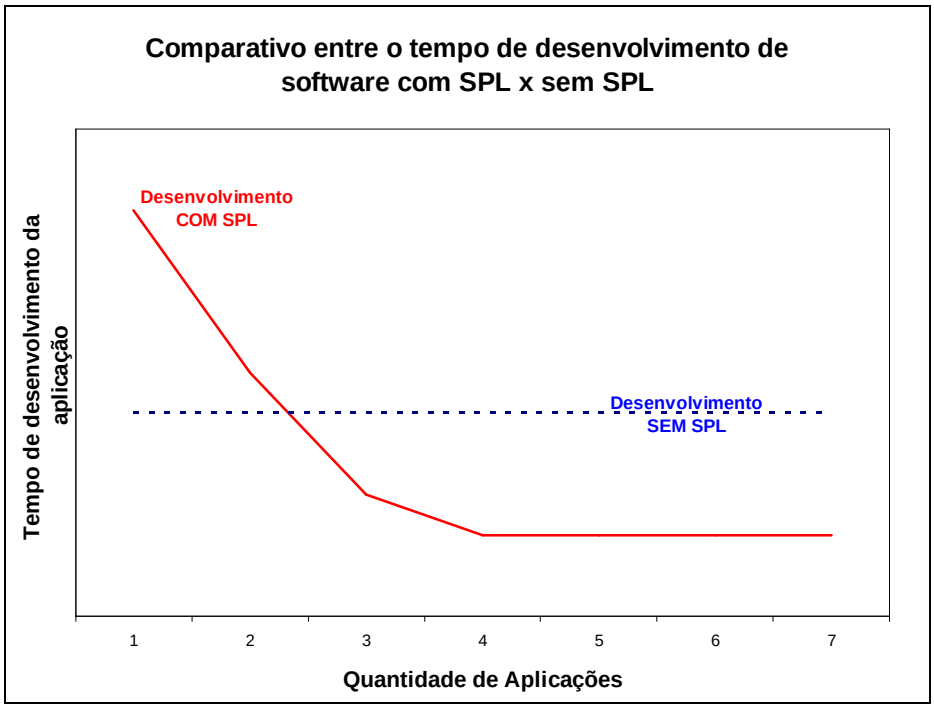

Figura 1.1. Comparativo do tempo de desenvolvimento de software com SPL e sem SPL.
Fonte: adaptado de Pohl et al (2005)

A relação entre o custo de desenvolvimento de uma SPL e o custo acumulado do desenvolvimento de software sem SPL é apresentada na Figura 1.2, onde se observa que a SPL passa a ter vantagens a partir do terceiro produto. Sendo assim, embora a SPLE traga redução de custos e tempo no desenvolvimento dos produtos de software pertencentes a uma SPL, tanto o tempo quanto o custo dos primeiros produtos é maior com a SPL, o que pode representar a perda do mercado para um concorrente que lançou um produto da mesma categoria antes.

Sendo assim, surgem questões tais como: seria possível reduzir o tempo de entrega e o custo dos primeiros produtos de uma SPL de forma a tanto aproveitar seus benefícios quanto aos benefícios oferecidos pela entrega de produtos antes dos concorrentes? De que forma isso poderia ser feito mantendo os benefícios oferecidos pela SPLE? Isso poderia ser feito através da integração entre a SPLE e o método *Scrum*? Como fazer essa integração? O primeiro passo para reduzir esse tempo de entrega, é definir um processo que integre ambas as metodologias, sem prejuízo à agilidade e ao reuso, que é o objetivo deste trabalho.

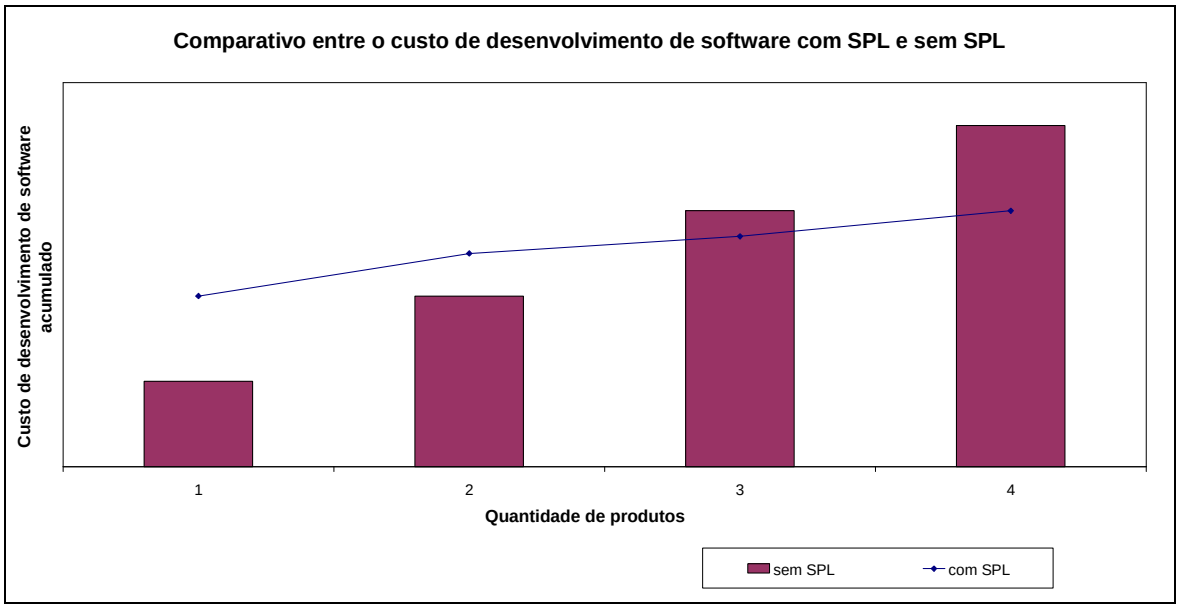

Figura 1.2. Comparativo do custo de desenvolvimento do software com SPL e sem SPL
Fonte: adaptado de Northrop e Clements (2007)

1.6 Organização do Trabalho

Este trabalho foi organizado de forma a cobrir os seguintes tópicos: descrever os fundamentos sobre SPLE e *Scrum*, definir o processo ScrumPL, compará-lo a outros processos, aplicar o processo ao desenvolvimento do sistema de navegação de TV e discutir a respeito de seu uso.

O capítulo 2 descreve o *framework* de SPLE definido por Pohl et. al. (2005), apresentando e descrevendo suas etapas e sub-processos, bem como os artefatos solicitados e gerados por cada sub-processo. Também são descritos o método *Scrum* definido por Schwaber (2004), com suas etapas, papéis envolvidos e artefatos e, por fim, o ciclo de vida *Scrum* definido por Larman (2004), apresentando uma visão geral e suas fases. Ao fim deste capítulo é feito um comparativo entre a SPLE e o método *Scrum*.

O capítulo 3 apresenta o processo ScrumPL, mostrando uma visão geral do processo, seus papéis e artefatos, a base lógica de sua construção através da análise das entradas e saídas da SPLE e do *Scrum*, e um comparativo entre o ScrumPL e outros ASPLMs.

O capítulo 4 descreve a SPL do sistema de navegação de TV, bem como o uso do ScrumPL em sua construção. São apresentados o *roadmap* de produtos, os requisitos da SPL, a arquitetura de referência, o SPL *backlog* desta SPL, a descrição dos componentes e

de suas interfaces, bem como aplicações específicas para os mercados americano e europeu.

No capítulo 5 é feita uma análise crítica do uso do processo ScrumPL no desenvolvimento da SPL do sistema de navegação de TV. Os pontos fortes e fracos do processo, sugestões de melhoria e de ferramentas.

E, por fim, no capítulo 6 são apresentadas as conclusões, recomendações, trabalhos futuros e dificuldades encontradas.

Capítulo 2- Engenharia de Linha de Produto de Software e Scrum

A personalização em massa tem sido adotada pela indústria para desenvolver vários e diferentes produtos compartilhando os mesmos componentes, lançando produtos similares, mas que atendam requisitos específicos de vários seguimentos de mercado. Exemplos disso estão na indústria de automóveis, onde um mesmo *design*, e peças são compartilhados por diversos modelos de carros que podem ser *hatch*, sedan, ou *pick-up*, assim como podem ter motores de diferentes potências. A personalização em massa também pode ser observada na indústria de software, onde um mesmo software, por exemplo de controle de produção, é usado em diversos clientes, mas é personalizado, de forma a atender necessidades específicas.

A metodologia usada para desenvolver software para personalização em massa é a engenharia de linha de produto de software – *Software product line* Engineering (SPLE). Esta metodologia utiliza os conceitos de reuso de componentes de uma forma sistemática e eficiente. Através da SPLE são desenvolvidas várias aplicações que compartilham uma plataforma comum, formando uma linha de produto de software (*Software product line* – SPL). A partir de três aplicações, o desenvolvimento de SPLs é feito em menor tempo e custo, e melhor qualidade comparando-se ao desenvolvimento das mesmas aplicações de forma isolada, ou seja, sem que compartilhem artefatos.

O método Scrum também é voltado para o desenvolvimento de aplicações no menor tempo e custo, e melhor qualidade, comparado a métodos tradicionais de desenvolvimento de software (Schwaber, 2004). Este método consiste num *framework* simples e fácil de aprender, com poucas práticas, artefatos e regras, que mantém todo o desenvolvimento visível (Schwaber e Beedle, 2002).

Tanto a SPLE quanto o método *Scrum* são projetados para desenvolver aplicações de forma produtiva, mas a SPLE adota o reuso sistemático como seu maior princípio, ao passo que o método *Scrum* adota o princípio de equipes auto-gerenciadas e o manifesto ágil (Beck et. Al, 2001). A seguir será descrito o funcionamento de ambas as metodologias, seguido de um comparativo entre as mesmas em diversos aspectos.

2.1 Engenharia de Linha de Produto de Software

Linhas de produto de software são projetadas para fornecer produtos personalizados a custos razoáveis, com melhor qualidade e menor tempo de introdução de produtos no mercado comparado ao desenvolvimento de cada um desses produtos isoladamente (Pohl et al., 2005). O custo e o tempo são reduzidos quando os artefatos da plataforma – que contém artefatos e tecnologias comuns a todos os produtos – são reutilizados em vários tipos de sistema. A melhoria da qualidade vem da revisão e teste dos artefatos da plataforma nos vários produtos da SPL.

O tempo de introdução dos três primeiros produtos no mercado é inicialmente alto, já que os artefatos comuns são os primeiros a serem construídos. No entanto, passada essa etapa, esse tempo é reduzido de forma considerável, já que muitos artefatos serão reutilizados nos vários novos produtos (Pohl et. Al, 2005). De acordo com Northrop e Clements (2007), a partir do terceiro produto o custo de desenvolvimento e o tempo de introdução do software no mercado são menores que o de desenvolvimento de cada produto individualmente.

A SPLE fornece dois processos para desenvolvimento de SPLs: engenharia de domínio e engenharia de aplicação. O primeiro trata da definição das partes comuns, das variabilidades e do escopo da SPL, bem como do conjunto de aplicações planejadas para fazer parte da SPL, e o segundo constrói as aplicações da SPL reutilizando o máximo possível dos artefatos advindos da engenharia de domínio e explorando a variabilidade (Pohl et al., 2005). A Figura 2.1 apresenta ambos os processos, bem como seus sub-processos, artefatos e ligações entre eles. As subseções a seguir descrevem os dois processos.

2.1.1 Processo de Engenharia de Domínio

O desenvolvimento da SPL inicia-se com o processo de engenharia de domínio, onde é definida e realizada a plataforma que contém os artefatos reutilizáveis. A engenharia de domínio possui os sub-processos: gerenciamento de produtos, engenharia de requisitos de domínio, projeto de domínio, realização de domínio e teste de domínio. A seguir são descritos o propósito e os artefatos de entrada e saída de cada um desses sub-processos, de acordo com Pohl et. al. (2005).

No sub-processo gerenciamento de produto são definidos os possíveis produtos que farão parte da SPL considerando aspectos econômicos e de *marketing*. A entrada para o processo são os objetivos da empresa, definida pela alta gerência. E seu resultado é um *roadmap* de produto, indicando o lançamento e as características das aplicações, bem como as características que estão dentro e fora do escopo da SPL.

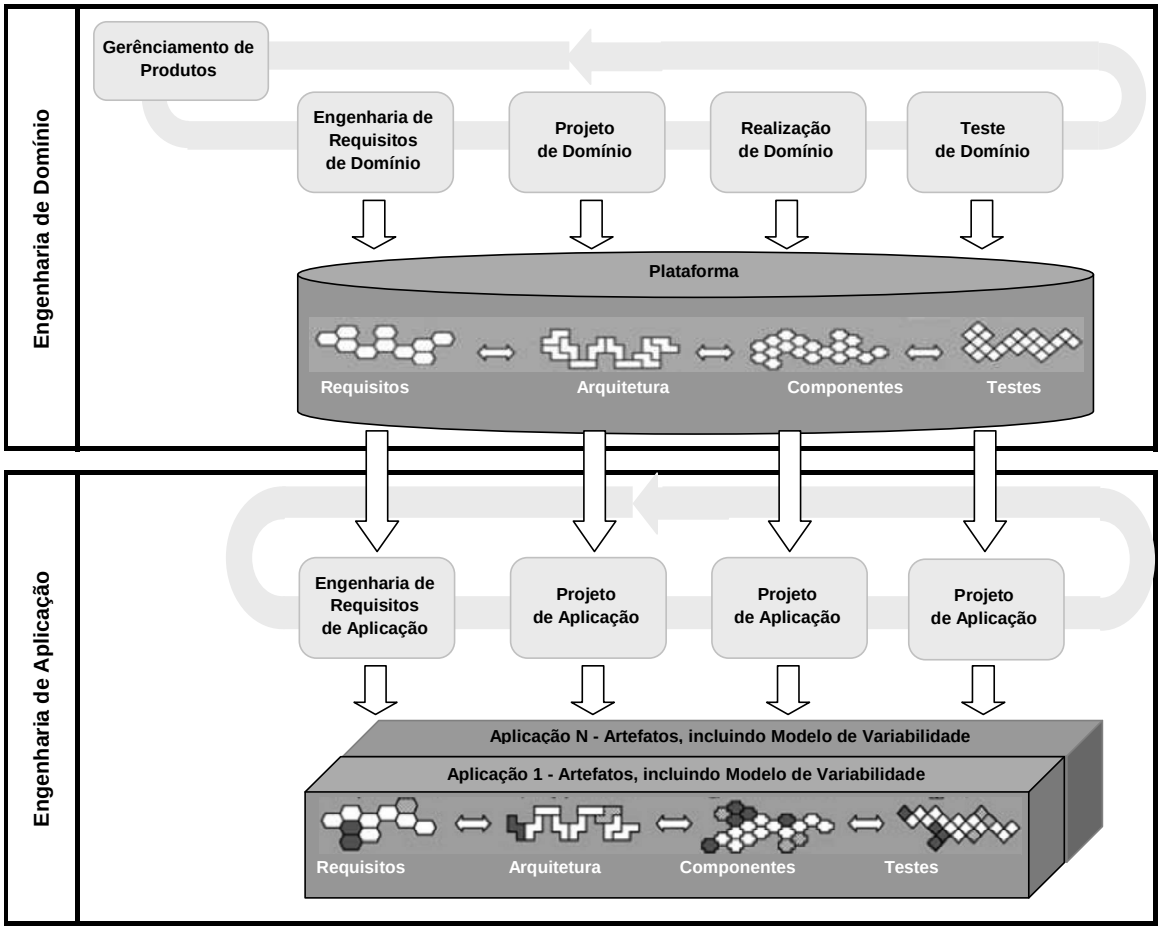

Figura 2.1. Visão geral do processo de engenharia de linha de produto de software
Fonte: adaptado de Pohl et al (2005)

O *roadmap* de produtos descreve as características de todas as aplicações da SPL, categorizando-as em comuns, que são parte de várias aplicações, e variáveis, que são parte de apenas algumas aplicações. Além disso, o *roadmap* de produtos define um calendário para introdução de produtos no mercado. Um exemplo é apresentado na seção 4.2.1.

Através do sub-processo de engenharia de requisitos de domínio, são elicitados os requisitos comuns a todos os produtos (requisitos de domínio) e os pontos de variação da SPL. Este sub-processo tem como entrada o *roadmap* de produtos e fornece requisitos textuais comuns a todos os produtos e suas variações para todas as aplicações previstas para a SPL e, em particular, o modelo de variabilidade da SPL.

O modelo de variabilidade define a variabilidade da SPL, especificando as partes em que ocorrerá variação, os chamados pontos de variação, e o tipo de variação que irá ocorrer, as chamadas variantes. O modelo de variabilidade é representado através de um diagrama de variabilidade, tal como o exemplo mostrado na Figura 2.2, que representa as variantes e pontos de variação de um sistema de segurança para portas. Neste exemplo, o ponto de variação “Pacote de Segurança” tem duas variantes: “Básico” e “Avançado”, o que determina que, para uma aplicação de segurança de portas desta SPL, uma destas variantes deverá ser escolhida . Ao escolher a opção “Básico”, por exemplo, obrigatoriamente serão escolhidas as variantes “Sensor de Movimento”, do ponto de variação “Detector de intrusos”, e “Teclado”, do ponto de variação “Fechadura de porta”. Esta obrigatoriedade é definida através das setas com o estereótipo “<requires>”. Desta forma, este diagrama é usado para auxiliar na escolha das variantes que farão parte de cada aplicação.

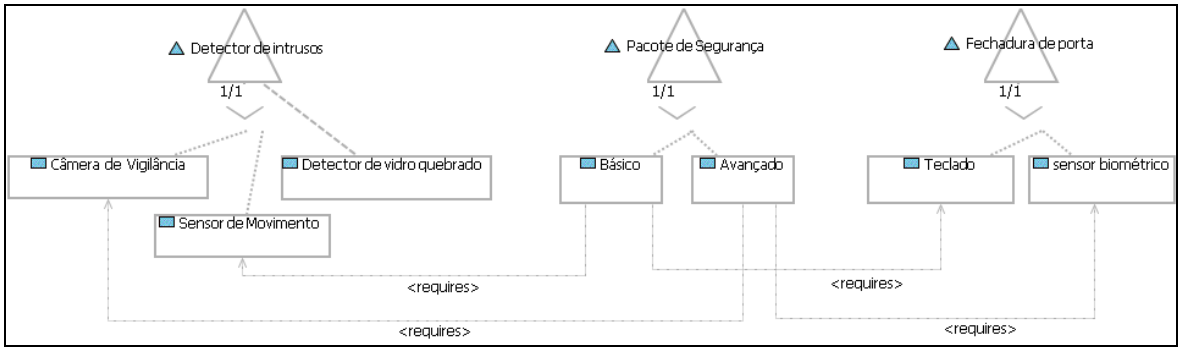

Figura 2.2. Exemplo de um diagrama de variabilidade
Fonte: adaptado de Pohl et al (2005)

O sub-processo de projeto de domínio inclui todas as atividades para a definição da arquitetura de referência da SPL. A arquitetura de referência fornece uma estrutura comum e de alto nível para todas as aplicações da SPL. Esse processo recebe os requisitos de domínio e o modelo de variabilidade. Seu resultado consiste na arquitetura de referência e no modelo de variabilidade refinado, contendo variabilidades internas.

A arquitetura de referência determina a estrutura e a textura das aplicações na SPL. A estrutura determina a decomposição estática e dinâmica que é válida para todas as aplicações da SPL. A textura é a coleção de regras comuns que guiam o projeto e a realização das partes e como elas são combinadas para formar as aplicações.

O sub-processo de realização de domínio se encarrega de detalhar o projeto e implementar os componentes reutilizáveis da SPL, ele recebe a arquitetura de referência,

incluindo uma lista de artefatos de software reusáveis para serem desenvolvidos, fornecendo códigos fonte, arquivos de configuração, que realizam variantes da SPL.

E, por fim, o sub-processo de teste de domínio é responsável pela verificação e validação dos componentes reutilizáveis implementados. Através desse sub-processo os componentes são testados com base em suas especificações, requisitos e artefatos de arquitetura e projeto. Além disso, são desenvolvidos artefatos de testes reutilizáveis para reduzir o esforço dos testes de aplicação. Este sub-processo recebe os requisitos de domínio, a arquitetura de referência, o projeto de componentes e interfaces, e os componentes reutilizáveis implementados e fornece os resultados dos testes executados, assim como os artefatos de teste. Os artefatos de teste incluem planos de teste, casos de teste e cenários de testes, referentes às partes comuns das aplicações.

2.1.2 Processo de Engenharia de Aplicação

Definida a plataforma, serão desenvolvidos os artefatos de cada produto através do processo de engenharia de aplicação, no qual são definidas as aplicações reutilizando o máximo possível dos artefatos de domínio contidos na plataforma e explorando as partes comuns e variantes da SPL. A engenharia de aplicação possui 4 sub-processos: engenharia de requisitos de aplicação, projeto de aplicação, realização de aplicação e teste de aplicação, que serão descritos a seguir juntamente com seus artefatos.

No processo de engenharia de aplicação, o desenvolvimento dos requisitos da aplicação, a partir dos requisitos de domínio e requisitos novos, é feita usando-se o sub-processo de engenharia de requisitos de aplicação. Este sub-processo recebe os requisitos de domínio e o *roadmap* de produto com as principais características do mesmo, além de requisitos específicos da aplicação em particular não capturados durante a engenharia de requisitos de domínio. O resultado deste sub-processo consiste nos requisitos específicos da aplicação em particular.

O sub-processo de projeto de aplicação realiza as atividades referentes à produção da arquitetura da aplicação. A arquitetura de referência é usada para instanciar a arquitetura de aplicação. Esse sub-processo seleciona e configura partes da arquitetura de referência e incorpora adaptações específicas da aplicação. Esse sub-processo recebe a arquitetura de referência e os requisitos da aplicação e fornece a arquitetura de uma aplicação específica.

O sub-processo de realização da aplicação cria a aplicação em si. Durante este sub-processo são selecionados e configurados os componentes assim como os artefatos específicos da aplicação. Os componentes reutilizáveis e específicos da aplicação são construídos e integrados para formar a aplicação completa. Esse sub-processo recebe a arquitetura da aplicação e os artefatos de aplicação reusáveis advindos da plataforma e fornece uma aplicação funcionando, juntamente com os artefatos de projeto detalhado.

E, por fim, a verificação e validação de todos os artefatos é feita através do sub-processo de teste da aplicação. Esse sub-processo recebe as especificações de requisitos de aplicação, a arquitetura da aplicação, o projeto de componentes e interfaces, juntamente com os artefatos de teste reutilizáveis, e os utiliza para testar a aplicação implementada. Esse sub-processo fornece relatórios a respeito dos testes executados, assim como relatórios de problemas a respeito de defeitos detectados.

2.2 O Método Ágil Scrum

O método *Scrum* (Schwaber e Beedle, 2002; Schwaber, 2004) é uma metodologia ágil para gestão e planejamento de projetos de desenvolvimento de software. Ele foi desenvolvido por Ken Schwaber e Jeff Sutherland na década de 1990, baseando-se em experiências no desenvolvimento de sistemas e processos, a partir do reconhecimento de que o desenvolvimento de software é muito complexo para ser planejado corretamente desde o início.

O método *Scrum* contempla uma visão empírica baseada em aspectos teóricos de controle de processos, ou seja, parte do pressuposto de que nem todas as características do produto são conhecidas na análise e que os requisitos mudarão com o passar do tempo. Contempla ainda duas atividades principais: inspeção e adaptação. Como o processo não é definido, o gerente de projeto, denominado *Scrum master* tem que inspecionar a execução diariamente, o que requer transparência, e fazer as adaptações necessárias com o passar do tempo (Santos Jr. e Santos, 2009). A Figura 2.3 apresenta uma visão geral do *Scrum*, seus elementos serão descritos a seguir.

O desenvolvimento é dividido em iterações (*sprints*), de aproximadamente trinta dias, e um *Scrum team* – equipe de até 10 pessoas, na qual podem fazer parte gerentes de projeto, projetistas, programadores, engenheiros, testadores, entre outros com as habilidades necessárias para atingir os objetivos a serem alcançados em cada *sprint*– se

compromete a selecionar e realizar *product backlog items* (conjunto de requisitos priorizados, na forma de uma lista de itens, que devem ser desenvolvidos para o produto) dentro de um *sprint*.

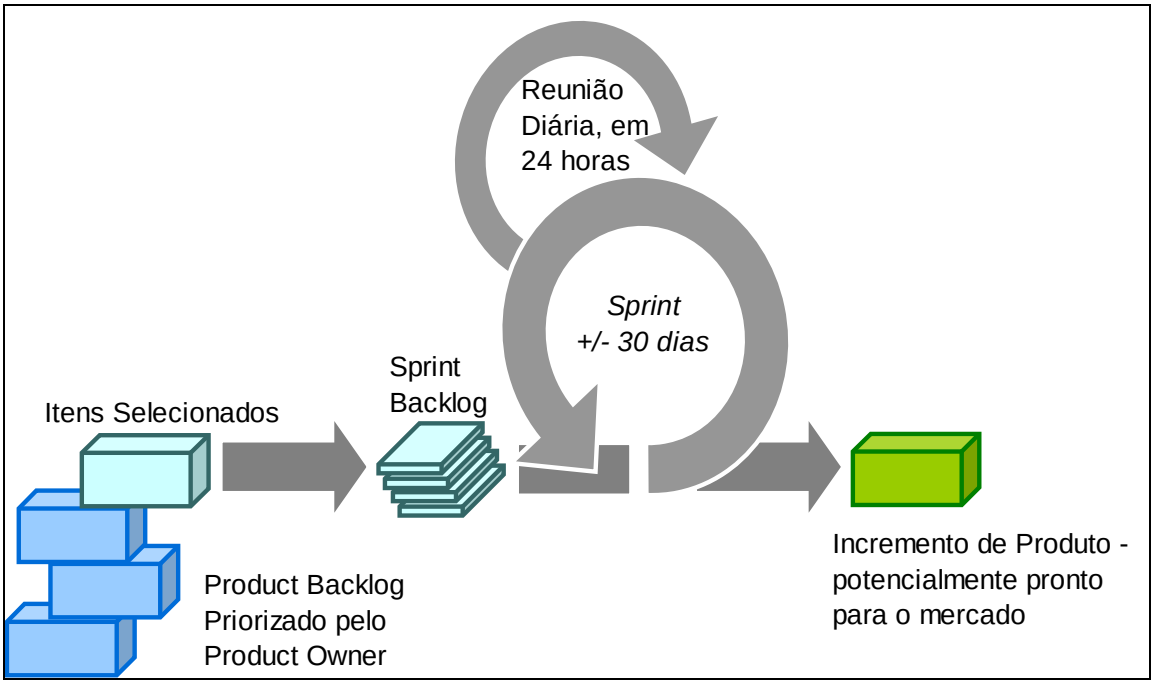

Figura 2.3. Visão geral do método Scrum
Fonte: adaptado de Schwaber e Beedle (2002)

O método *Scrum* possui três papéis bem definidos: o *product* owner, o *Scrum master* e o *Scrum team* (Schwaber e Beedle, 2002). O primeiro é responsável por determinar a visão do produto, financiar o projeto, criar e priorizar os *product backlog items*, definir os objetivos de cada *sprint* e revisar o produto ao fim de cada *sprint*. O segundo é responsável por garantir que os valores e práticas do método *Scrum* sejam seguidos, ser o mediador entre a gerência e o time, conduzir as reuniões e remover impedimentos. O terceiro é responsável por escolher realizar os *product backlog items* durante o *sprint*, sendo um conjunto de profissionais com habilidades suficientes para atingir os objetivos determinados pelo *product owner*.

Os artefatos gerados pelo método *Scrum* são todos aqueles que o time acha necessários para atingir os objetivos determinados pelo *product owner*. Mas há 3 artefatos que são determinados pela própria metodologia: o *product backlog*, o *sprint backlog* e o *burndown chart* (Schwaber e Beedle, 2002).

O *product backlog* é uma lista de requisitos priorizados, com tempos estimados para torná-los uma funcionalidade completa. Essa lista de requisitos, denominados *product*

backlog items, é determinada pelo *product owner*, podendo ser alterada a qualquer momento. O *Scrum team* é responsável por estimar o esforço necessário para desenvolver cada *product backlog item* e por escolher aqueles que irão realizar durante o *sprint*. Apesar de ser permitida a mudança de *product backlog items* a qualquer momento, aqueles escolhidos pelo *Scrum team* para realização num *sprint* não poderão ser modificados até o fim desse *sprint*.

O *sprint backlog* consiste numa lista de tarefas que define o trabalho a ser realizado pelo *Scrum team* durante um *sprint*. As tarefas e o tempo estimado para cumpri-las são definidos no início do *sprint*, durante a *sprint planning meeting*, uma reunião na qual participam o *product owner, o Scrum master e o Scrum team*. Durante o *sprint*, o *Scrum team* seleciona tarefas do *product backlog* para realizar, atualizando o esforço restante das tarefas selecionadas ao fim de cada dia até cumpri-las. Tarefas podem ser adicionadas ou canceladas no *sprint backlog*, caso o *Scrum team* sinta necessidade. Um exemplo do uso do *sprint backlog é apresentado na seção 3.2.3*

O *burndown chart* consiste num gráfico de linha – cuja tendência é de descida – representando o esforço restante do *sprint* para: (a) tornar funcionais os *sprint backlog items* escolhidos durante um *sprint*, (b) entregar uma parte funcional do produto ao mercado, (c) completar o produto. Este esforço restante do *sprint* consiste na soma do esforço de todas as tarefas, calculado diariamente. Um exemplo do *burndown chart* é apresentado na seção 3.2.4.

Visando definir os objetivos *sprint* e fazer planejamento, sincronizar tarefas e solucionar impedimentos, apresentar os resultados aos *stakeholders*, bem como melhorar o desempenho do *scrum team*, cinco reuniões são realizadas: *sprint planning meeting, daily meeting, review meeting e retrospective meeting* (Schwaber e Beedle, 2002).

A reunião denominada *sprint planning meeting* tem duração de oito horas e dá início a um *sprint*. Nas primeiras quatro horas, o *product owner* define os objetivos do *sprint* e o *Scrum team* estima o esforço para realizar os *product backlog items* com maior prioridade, e seleciona os que serão realizados durante o *sprint*. Nas quatro horas seguintes, o *Scrum team* define tarefas para realizar os *product backlog items* selecionados e estima o esforço necessário para realizá-las. Essas tarefas irão compor o *sprint planning*.

Visando sincronizar o trabalho do time, existem reuniões diárias de acompanhamento, as chamadas *daily meetings*. Essas reuniões têm duração de aproximadamente quinze minutos, nas quais cada participante responde a três perguntas: “o que foi feito desde a última *daily meeting*?”, “o que será feito até a próxima?” e “o que

está impedindo o andamento dos trabalhos?". Todos os participantes observam o que os outros falam e podem, dependendo da situação, oferecer ajuda, reunir para discutir um assunto após a *daily meeting*, sincronizar os trabalhos, entre outras ações que levem à conclusão de tarefas. Os impedimentos deverão ter uma solução definida pelo *scrum máster* em até uma hora após a *daily meeting*.

Nessa lista de reuniões, uma das mais importantes é a *review meeting*, na qual o *Scrum team* mostra o que foi alcançado durante o *sprint* ao *product owner* e outros *stakeholders*. Durante esta reunião, são apresentados os *product backlog items* que o *scrum team* realizou. Também são informados aqueles não realizados, que serão acrescentados novamente ao *product backlog* para priorização, seleção e realização num outro *sprint*.

As *retrospective meetings* consistem em reuniões realizadas ao final de cada *sprint* com o objetivo de relatar os pontos fortes e fracos deste *sprint*, definir ações para manter os pontos fortes e eliminar ou reduzir os pontos fracos relatados. Por fim, a *sprint planning meeting* é feita novamente, iniciando um novo *sprint*, com a presença de todos os *stakeholders* envolvidos.

2.3 Ciclo de Vida Scrum

O ciclo de vida *Scrum* (Larman, 2004) é composto de três fases: pré-jogo (composta pelas etapas planejamento, projeto), desenvolvimento e entrega. O Quadro 2.1 fornece um resumo de cada uma dessas fases. O propósito da etapa de planejamento é estabelecer a visão, definir as expectativas e obter financiamento para o projeto. Os resultados desta fase são um *product backlog* inicial com *product backlog items* priorizados, a visão em relação ao produto, um projeto de alto nível do produto e protótipos. O propósito da etapa de projeto é identificar mais requisitos e priorizar a quantidade de itens suficiente para o primeiro *sprint*, seus resultados consistem num plano, projeto de alto nível e protótipos.

Ambas, a etapa de planejamento e a de projeto, formam a fase denominada pré-jogo, também chamada *sprint zero*[1], que precede o primeiro *sprint*. Ao fim desta fase o

[1] De acordo com o treinamento *Certified Scrum Máster* (CSM) fornecido pela Object Mentor (http://www.objectmentor.com)

product backlog deverá ter itens detalhados de tal forma que o *Scrum team* possa escolher aqueles que devem ser realizados no primeiro *sprint*, ou seja, o tempo estimado desses *product backlog items* deve ser menor que o tempo de um *sprint*, caso contrário, deverão ser divididos em dois ou mais.

Pré-Jogo		**Desenvolvimento**	**Entrega**
Planejamento	**Projeto**		
Propósito: - Estabelecer a visão, definir as expectativas e assegurar o financiamento. Atividades: - Escrever a visão, definir o orçamento e o *product backlog* inicial, e estimar os itens. - Fazer protótipos e projetos.	Propósito: - Identificar mais requisitos e priorizar o suficiente para o primeiro *sprint*. Atividades: - Planejamento - Fazer protótipos e projetos.	Propósito: - Implementar um sistema pronto para entrega em uma série de *sprints*. Atividades: - Planejamento do Sprint. - Definição do *sprint backlog* e das estimativas. - Realização das atividades planejadas. - Reunião diária. - Revisão do sprints.	Propósito: - Implantar o sistema. Atividades: - Documentação. - Treinamento. - Marketing e vendas. - ...

Quadro 2.1. Fases, atividades e artefatos do ciclo de vida Scrum
Fonte: adaptado de Larman (2004)

O propósito da fase de desenvolvimento é implementar um produto de software pronto para entrega através de uma série de *sprints*. As atividades de cada *sprint* são feitas de acordo com o descrito na seção 2.2. Larman (2004) descreve que, em geral, os últimos *sprints* dessa fase envolvem tarefas relativas à qualidade do produto.

A fase de entrega tem como propósito executar tarefas de implantação do software, tais como treinamento, documentação, marketing, entre outras. Esta fase também é chamada pós-jogo.

2.4 Comparativo Entre o Método Scrum e a SPLE

O método Scrum, assim como os métodos ágeis em geral, possui tanto pontos convergentes quanto divergentes em relação à SPLE. O quadro 2.2 apresenta diferenças e semelhanças entre o método Scrum e a SPLE. Por ser um método ágil, essas diferenças têm relação com o manifesto ágil, portanto alguns dos itens do manifesto farão parte deste quadro, em especial os critérios “uso de processo” e “documentação”. As diferenças e semelhanças entre os métodos ágeis e a SPLE podem ser vistos com maior detalhe em Tian e Cooper (2006) e Mohan, Hamesh e Sugumaran (2010).

Diferenças		
Critério de comparação	**Scrum / Métodos ágeis**	**SPLE**
Clientes	Colaboração direta	Colaboração com representantes
Desenvolvedores	Conhecimento implícito	Seguir planos e processos
Objetivos	Táticos, de curto prazo	Estratégicos, de longo prazo
Uso de processos	Dá mais valor a indivíduos e interações sobre processos.	Dá mais valor a processos
Documentação	Dá mais valor a software funcionando sobre documentação abrangente	Dá bastante valor a documentação abrangente
Entrega de produtos	Ao fim de um *sprint* entrega-se uma funcionalidade potencialmente pronta do produto para o mercado	Desenvolvem-se os artefatos comuns, utilizando-os, juntamente com os específicos, para compor um produto
Semelhanças		
Acomodam mudanças		
Entregam software de forma rápida e com qualidade		

Quadro 2.2. Comparativo entre os métodos ágeis e a SPLE.
Fonte: do próprio autor

O método *Scrum* possui princípios que podem entrar em conflito com os da SPLE, tais como o princípio *Scrum* de que ao fim de um *sprint* deve ser entregue uma funcionalidade potencialmente pronta para o mercado, ao passo que na SPLE deve-se primeiro desenvolver os artefatos de domínio, que compõem a plataforma, e só depois os artefatos de aplicação. A SPLE descreve fases e atividades bem definidas a serem seguidas pelos *stakeholders*, ao passo que, no método *Scrum*, quem deve definir as atividades para atingir os objetivos de um *sprint* é o *Scrum team*, de acordo com sua experiência.

O método *Scrum* define que ao final de um *sprint* deve ser entregue uma ou mais funcionalidades parcialmente ou totalmente prontas para ir ao mercado, o que implica ter qualidade, mas esta metodologia não define como se deve atingir essa qualidade, deixando esta definição a cargo do *Scrum team*. Neste ponto, a própria SPLE fornece meios para garantir a qualidade das funcionalidades realizadas voltadas à verificação e validação dos artefatos de domínio e de aplicação.

O método *Scrum*, sendo um método ágil, segue os princípios do manifesto ágil (Beck et al., 2001). Por outro lado, a SPLE é dirigida por planos e baseada em procedimentos, além de exigir documentação, principalmente dos artefatos da plataforma, de forma a permitir o reuso (Pohl et al., 2004; Northrop e Clements, 2007). Isso implica em dar valor a processos e a documentação abrangente. Assim, percebe-se que há tanto pontos em comum das duas metodologias quanto pontos de conflito.

Os métodos ágeis têm apresentado diversos casos de sucesso em relação ao tempo de entrega de software para o mercado e o aumento da qualidade, considerando o

desenvolvimento de produtos únicos (Lindvall et al., 2004). Sutherland et al. (2009) descrevem que, com o método *Scrum*, uma equipe pode ter aumento de produtividade no desenvolvimento de software de 5 a 10 vezes em relação à média industrial. Santos Jr. et al. (2008) relatam resultados expressivos em 72% dos projetos de uma empresa fabricante de dispositivos móveis durante dois anos de uso do *Scrum*. Em Santos Jr. e Santos (2009) relata-se um projeto de desenvolvimento de software onde o tempo de entrega foi de 4 meses, tendo sido planejados 6 meses. Schwaber (2004) também relata diversos casos de sucesso em termos de tempo de entrega usando este método ágil.

Larman (2004) definiu o termo cerimônia para representar o quanto uma metodologia ágil aceita a introdução de documentos e outras formalidades em seu uso. Ele comparou o *Scrum* a outros métodos ágeis, indicando que o mesmo é flexível em relação à escala de cerimônia, já em termos de ciclos, cada iteração do *Scrum* tem mais ou menos 30 dias (ver Figura 2.4). Isso permite que o mesmo seja adaptado para uso em processos que possam ser certificados em modelos tais como ISO 9001 (ABNT, 2001) e CMMI (Crissis et al., 2003).

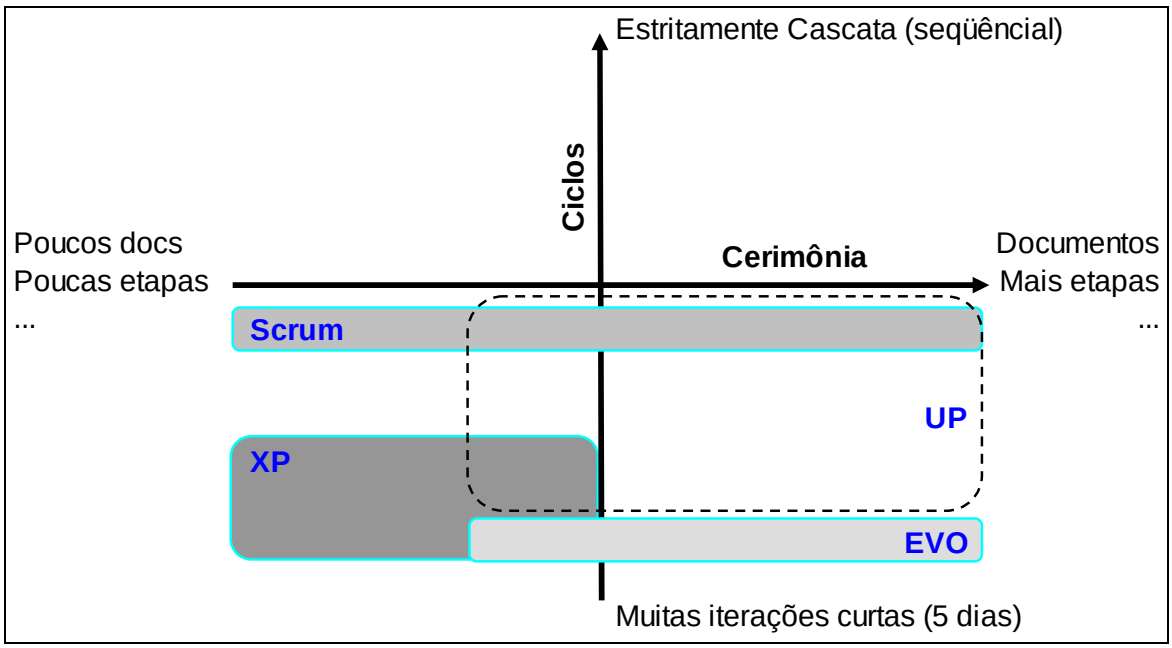

Figura 2.4. Escala de ciclos e cerimônia
Fonte: adaptado de Larman (2004)

McMichael e Lombardi (2007) e Santos et al. (2008) apresentam relatos de experiência de empresas que usavam *Scrum* e que foram certificadas ISO 9001, após iniciativas de melhoria de processos. No website SCRUMMI (2009) é apresentado um processo baseado no *Scrum* e aderente ao CMMI. Já Vriens (2003) apresenta um relato de

experiência descrevendo como obteve as certificações ISO 9001 e CMMI nível 2 com *Scrum* e XP (*eXtreme Programming*).

A alta cerimônia do método *Scrum*, os relatos de experiência indicando sucesso usando-o com modelos consagrados e a alta produtividade alcançada por muitas equipes que o utilizaram, são um bom indicativo de que este processo pode ser integrado à SPLE, através de um novo processo, sem prejuízo à agilidade e ao reuso.

2.5 Trabalhos Relacionados

Em 2006, na conferência SPLC (*Software product line Conference*), foi realizado o workshop denominado APLE – 1st *International Workshop on Agile Product Line Engineering* (Engenharia de Linha de Produto Ágil) (Cooper e Franch, 2006), cujo objetivo foi unir as comunidades de SPLE e métodos ágeis de forma a investigar pontos em comum e divergentes entre estas metodologias, bem como custos e benefícios potenciais no desenvolvimento de uma abordagem de engenharia de linha de produto ágil. Desde então, diversos trabalhos foram publicados com este objetivo. Estas publicações serão apresentadas e discutidas a seguir.

Tian e Cooper (2006) comparam as metodologias ágeis à SPLE pelas perspectivas de engenharia de requisitos, garantia da qualidade de software e gerência de projetos, observando que as metodologias compartilham o objetivo de entregar software de qualidade de forma rápida, mas usando estratégias e práticas diferentes, concluindo que é possível fazer adaptações à SPLE de forma a torná-la ágil.

Tian e Cooper (2006) buscam descrever como adotar os princípios ágeis às atividades da SPLE. Eles não usam um método ágil específico para integrar à SPLE e nem fazem qualquer menção de uso das atividades e artefatos usados pelos métodos ágeis, desta forma não buscaram a integração da SPLE com métodos ágeis, mas buscaram melhorar a SPLE usando os princípios dos métodos ágeis. A comparação contribui ao identificar pontos de convergência e de conflito entre os métodos.

Carbon et al. (2006) tratam da abordagem de reuso estratégico e proativo da SPLE integrando métodos ágeis com o processo de engenharia de aplicação PuLSE-I para realizar uma adaptação ou calibração de um produto voltado a um cliente específico. Eles buscam usar o PuLSE-I para desenvolver a SPL, e métodos ágeis apenas para desenvolver

aplicações que atendam a novas necessidades de clientes, buscando reutilizar os artefatos já existentes na SPL.

Noor et al. (2008) conduziram um estudo de caso incorporando agilidade à engenharia de linha de produtos, usando o conceito de *thinkLets* para desenvolvimento colaborativo de um mapa de produtos, chegando à conclusão de que é viável incorporar os princípios ágeis ao planejamento de uma linha de produtos. Eles focam no aumento da velocidade do planejamento da linha de produto através de um processo de colaboração entre os *stakeholders*, de forma a desenvolver rapidamente o mapa de produtos, apresentando também um estudo de caso. O processo definido por Noor et al. (2008) abrange apenas a etapa de planejamento da SPL, não tratando da etapa de desenvolvimento do software.

Ghanam e Maurer (2008) apresentam uma proposta de modelo orientado a testes (*Test Driven Development* – TDD), usando também aspectos do *Scrum*, em especial os métodos de planejamento e controle. Esta proposta segue uma abordagem *bottom-up*, onde artefatos reutilizáveis são extraídos, sob demanda, de artefatos de produtos existentes, ou seja, eles buscam construir uma SPL a partir da refatoração dos artefatos destes produtos para torná-los reutilizáveis e, conseqüentemente, criar novas aplicações com os mesmos.

Hanssen e Faegri (2007), com o objetivo de investigar como a SPLE e os métodos ágeis podem ser combinados através de uma abordagem qualitativa, descrevem e analisam um estudo de caso, onde uma indústria desenvolveu um processo unindo a SPLE ao método ágil Evo. Esse processo possui três partes distintas: uma parte estratégica, onde são tratados um *roadmap* e a plataforma, uma parte tática, onde são desenvolvidos softwares que atendam a solicitações de clientes, e uma operacional, onde é fornecido o suporte técnico ao cliente. O processo descrito por Hanssen e Faegri foi desenvolvido considerando uma SPL já pronta, mas ainda assim é usado tanto para desenvolver artefatos de domínio quanto os de aplicação.

2.6 Conclusão

Neste capítulo foram apresentados os princípios, fundamentos, processos e artefatos da SPLE e do método *Scrum*, e as etapas do ciclo de vida *Scrum*, bem como trabalhos que buscam integrar métodos ágeis à SPLE. A apresentação dos processos de engenharia de domínio e de aplicação da SPLE foi importante para demonstrar as diferenças dessas

etapas, bem como seus principais artefatos, especialmente o *roadmap* de aplicações, a arquitetura de referência, a arquitetura específica de cada aplicação, os componentes e as interfaces.

Na apresentação do método *Scrum* devem ser destacados o papel do *product owner*, ao criar e manter o *sprint backlog*, e os artefatos de planejamento e controle, bem como as reuniões. Já o ciclo de vida *Scrum* tem como destaque a descrição de suas etapas. Como será visto no próximo capítulo, esses pontos destacados foram de fundamental importância para a elaboração do processo ScrumPL, também fazendo parte deste processo de forma adaptada.

Capítulo 3- ScrumPL – Desenvolvimento de SPLs com Scrum

O processo ScrumPL foi projetado para o desenvolvimento de SPLs através da combinação de atividades advindas do método *Scrum* e da SPLE. Sua concepção foi feita a partir da combinação do *framework* de SPLE definido por Pohl et. al (2005), do método ágil *Scrum* e do ciclo de vida *Scrum* (Larman, 2004).

A Figura 3.1 apresenta uma visão geral do ScrumPL, que é composto pelas fases: pré-jogo (composto pelas etapas planejamento, projeto), desenvolvimento e entrega, advindas do ciclo de vida do *Scrum* e representados pelas colunas. Os processos engenharia de domínio e engenharia de aplicação, da SPLE, são representados pelas linhas. A combinação entre as fases do ciclo de vida do *Scrum* e os processos da SPLE para formar o ScrumPL será descrita a seguir. Na seção 3.1, serão definidas as atividades ocorridas nas fases do ScrumPL. Na seção 3.2 serão definidos dos artefatos para acompanhamento e controle. Na seção 3.3 será feito um comparativo do ScrumPL com outros APLMs, fechando com as conclusões deste capítulo, na seção 3.4.

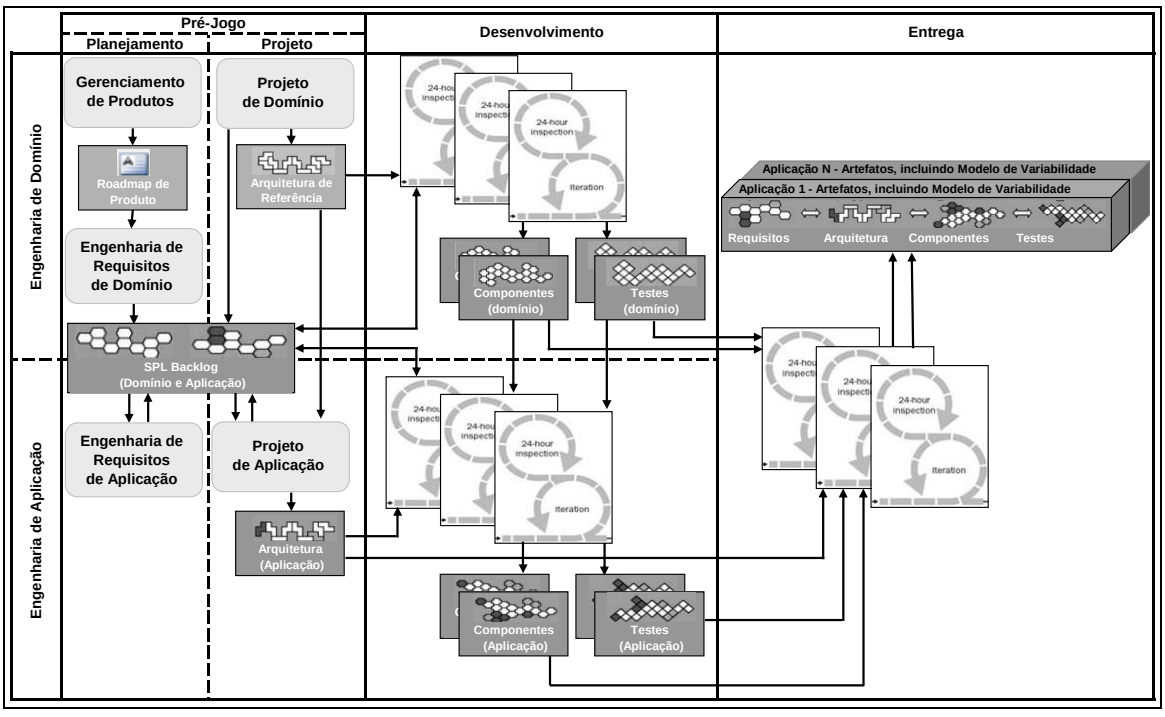

Figura 3.1. Visão geral do processo ScrumPL
Fonte: do próprio autor

O alicerce da integração entre os dois métodos é o ciclo de vida *Scrum*, que provê as fases para: (a) definição e elicitação de requisitos, e elaboração das arquiteturas de referência e específicas de cada aplicação, denominada pré-jogo; (b) desenvolvimento de componentes da plataforma e das aplicações, denominada desenvolvimento; e (c) integração e entrega de aplicações, denominado entrega.

A SPLE fornece os sub-processos: gerenciamento de produtos, engenharia de requisitos de domínio, projeto de domínio, engenharia de requisitos de aplicação e projeto de aplicação. Estes sub-processos provêem: os requisitos comuns, a arquitetura de referência da SPL, os requisitos específicos das aplicações e a arquitetura específica das aplicações, respectivamente.

O método *Scrum* fornece os *sprints*, através dos quais são desenvolvidos: os componentes implementados e testados, a especificação e implementação dos testes – para que possam ser reusados em futuros testes de regressão – e as aplicações completas através da integração dos componentes desenvolvidos. O *Scrum team* poderá realizar, durante os *sprints*, quaisquer artefatos que representem os componentes, dentre eles os solicitados explicitamente pelo *product* owner, através do *product backlog* (doravante denominado SPL *backlog*), ou outros artefatos que julgarem necessários.

3.1 Descrição do Processo ScrumPL

O ScrumPL é composto das fases pré-jogo, desenvolvimento e entrega. A primeira visa a definição das aplicações, dos requisitos da SPL e das arquiteturas de referência e das aplicações. A segunda visa a realização dos componentes e de suas interfaces. Já na terceira são feitas a integração dos componentes em aplicações, bem como a implantação. As atividades desenvolvidas em cada uma dessas fases serão descritas nas subseções a seguir.

3.1.1 Fase Pré-Jogo

Nesta fase, a visão da SPL, seus pontos de variação, aplicações e períodos de entrega são fornecidos pelo sub-processo gerenciamento de produtos, da SPLE, através do *roadmap* de produtos. Essas informações são utilizadas pelo sub-processo engenharia de requisitos de domínio para fornecer as características dos produtos, que são adicionadas, na

fase de planejamento, ao SPL *backlog* – que tem a mesma função do *product backlog*, mas gerencia toda a SPL ao determinar as aplicações e variantes afetadas por cada *SPL backlog item* (ver exemplo na seção 3.2).

Essas características são entradas para o sub-processo engenharia de requisitos de aplicação para extrair requisitos para uma aplicação em particular e reusá-los o máximo possível, durante a fase de planejamento; e também são entradas para o sub-processo projeto de domínio para criar e manter a arquitetura de referência.

A arquitetura de referência contém pontos de variação, variantes, bem como a descrição e interfaces de componentes de software reutilizáveis. O *product owner*, além de suas atribuições já definidas pelo *Scrum*, é responsável por criar e manter a arquitetura de referência, e adicionar cada um dos componentes e interfaces, assim como outros artefatos que julgar necessários, ao SPL *backlog* como SPL *backlog items*. Isto é feito durante a fase de projeto. O conceito de componente de software usado na arquitetura de referência é o definido por Szyperski (2008): são unidades binárias e independentes de produção, aquisição e implantação que interagem para formar um sistema funcional.

Juntamente com a arquitetura de referência, é definida uma estratégia de desenvolvimento indicando tanto as habilidades necessárias ao *Scrum team* para desenvolver a SPL, quanto a abordagem da criação da SPL (Nyholm, 2002), que pode ser: evoluir um conjunto de produtos existentes em uma SPL, trocar um conjunto de produtos existentes por uma SPL, evoluir uma nova SPL ou desenvolver uma nova SPL.

Os componentes reutilizáveis da arquitetura de referência, e adicionados ao SPL *backlog*, são realizados através dos *sprints*, durante a fase de desenvolvimento, e também servem como entrada para o sub-processo projeto de aplicação.

Através do sub-processo projeto de aplicação são produzidas as arquiteturas específicas de cada aplicação, a partir da arquitetura de referência, selecionando variantes, componentes e outros artefatos, para, em seguida adaptar o projeto de acordo com os requisitos específicos da aplicação. Os artefatos das arquiteturas das aplicações, especialmente os componentes e suas interfaces, são adicionados ao SPL *backlog*, como SPL *backlog items*.

3.1.2 Fase de Desenvolvimento

A fase de desenvolvimento se inicia quando o SPL *backlog* construído na fase de pré-jogo, é priorizado e está pronto para a estimativa. A fase pré-jogo é fechada e o *Scrum team* pode trabalhar de acordo com o método *Scrum*, descrito na seção 2.2, começando um *sprint* com a *planning meeting*, realizando diariamente as *daily meetings*, finalizando com a *review meeting*, seguido da *retrospective meeting*.

Os resultados das atividades do *Scrum team* são componentes potencialmente prontos para entrega (realizados e testados), bem como descrição e implementação de casos de testes unitários e de integração usados para executar os testes de aceitação do *Scrum* no *sprint*, e para ser reutilizados em futuros testes de regressão. Os componentes realizados farão parte de um repositório que representa a plataforma, caso sejam reutilizáveis, ou farão parte de um repositório representando uma das aplicações, se forem específicos.

Durante o *sprint*, o *Scrum team* realiza os objetivos dos sub-processos de realização de domínio e de realização de aplicação, ambos da SPLE. O primeiro consiste em elaborar o projeto detalhado e implementar artefatos reutilizáveis, resultando nos componentes reusáveis. Já o segundo consiste em desenvolver aplicações que podem ser testadas e entregues ao mercado após garantir qualidade suficiente. Também, durante o *sprint*, são especificados e/ou realizados os testes caixa branca e caixa preta. Eventuais bugs podem ser corrigidos no *sprint* corrente ou serão acrescentados ao SPL *backlog* para correção em futuros *sprints*.

O *product* owner, pode fazer quaisquer mudanças nos SPL *backlog items* não selecionados pelos *Scrum team*s durante um *sprint*. Essas mudanças podem ser feitas por motivos de problemas relatados, novos requisitos, mudanças em interfaces, defeitos em descrições de interfaces, problemas na realização de artefatos de domínio ou de aplicação, dentre outros, e são adicionadas ao SPL *backlog* e repriorizadas pelo *product owner*, sendo re-estimadas e selecionadas pelos *Scrum team*s durante a *sprint planning meeting* em futuros *sprints*.

3.1.3 Fase de Entrega

Durante a fase de entrega, os componentes serão integrados de forma a ter a aplicação completa, e assim, realizar os testes de integração e de sistema e corrigir eventuais *bugs*. Os testes de integração são realizados reutilizando os testes desenvolvidos na fase anterior, podendo ser feitas especificações e execuções de novos testes não previstos.

Ao fim da integração e dos testes, são feitos documentos e atividades de implantação, tais como instalação, configuração, treinamento, entre outras.

3.2 Artefatos de Planejamento e Controle do ScrumPL

Para o acompanhamento do progresso de desenvolvimento das aplicações da SPL, são usados dois artefatos: SPL *backlog* (ver Figura 3.2) e SPL *backlog burndown chart* (ver Figura 3.3). Ambos serão apresentados nesta seção, juntamente com os artefatos do método *Scrum*: *sprint backlog* (ver Figura 3.4) e *sprint backlog burndown chart* (ver Figura 3.5). Nesta seção, estes artefatos contêm dados fictícios para demonstrar seu uso e o relacionamento entre eles.

3.2.1 SPL Backlog

Contém os SPL *backlog items*, que representam os componentes e interfaces advindos da arquitetura de referência e arquitetura específica dos produtos, ou bugs, testes, entre outras necessidades da SPL, determinadas pelo *product owner*. Para cada SPL *backlog item*, o *product owner* descreve a estória e a prioridade, e identifica as aplicações das quais fazem parte e quais pontos de variação e variantes estão realizando. A estória consiste numa descrição de como o SPL *backlog* item executará as tarefas necessárias a atingir os objetivos a que se propõe.

O *Scrum team*, por sua vez, estima o esforço necessário para desenvolver o SPL *backlog item*, com base em seu objetivo, estória e outras informações fornecidas pelo *product* owner, e escolhe quais serão realizados em um *sprint*, mudando a situação destes de “aguardando” para “selecionado”. Após o *Scrum team* ter concluído a realização, a situação muda para “pronto”.

Para cada SPL *backlog item* é determinada uma variante, o que permite controlar a realização das variantes em função dos SPL *backlog items* realizados. Pela mesma razão, também são determinadas as aplicações da SPL da qual fazem parte cada SPL *backlog item*, o que permite fazer o controle do progresso de realização de cada aplicação em função dos SPL *backlog items*.

Desta forma, o SPL *backlog* é usado para gerenciamento e controle de todas as aplicações da SPL, permitindo que este gerenciamento seja feito tanto para toda a SPL quanto para cada aplicação específica, bem como por variante, que é importante para determinar o progresso de futuras aplicações, ainda não inseridas no SPL *backlog* por estarem na etapa de planejamento mas que já têm as variantes pré-determinadas.

A Figura 3.2 mostra um exemplo de um SPL *backlog* contendo os SPL *backlog items* PBI1 a PBI12. Cada um possui uma estória, não descritas aqui por questões de espaço da figura. Também é fornecida uma prioridade, onde as que têm número menor são as de maior prioridade. O esforço estimado para realizar cada SPL *backlog item*, em homem-hora, é fornecido pelo *scrum team* antes do início de cada *sprint* ou durante a primeira parte da *sprint planning meeting*.

SPL Backlog

Product Backlog Item	Estória	Prioridade	Esforço * estimado	Sprint	Situação	Ponto deVariação	Variante	Aplicações A	B	C	D	E	F	
PBI1	-	1	480	1	pronto	VP1	VAR1.1	X		X			X	
PBI2	-	1	416	1	pronto	VP1	VAR1.2		X			X	X	
PBI3	-	10	384	2	pronto	VP1	VAR1.3		X	X				
PBI4	-	20	320	2	pronto	VP2	VAR2.1						X	
PBI5	-	30	160	3	selecionado	VP1	VAR1.4		X	X	X		X	
PBI6	-	40	352	3	selecionado	VP2	VAR2.2	X	X		X			
PBI7	-	50	256	3	selecionado	VP3	VAR3.1					X		
PBI8	-	50	128	4	aguardando	VP1	VAR1.5	X	X	X			X	
PBI9	-	50	224	4	aguardando	VP2	VAR2.3		X		X			
PBI10	-	60	1920	6	aguardando	VP3	VAR3.2	X			X	X		
PBI11	-	70	640		aguardando	VP2	VAR2.4	X		X		X		
PBI12	-	80	960		aguardando	VP3	VAR3.3		X		X			

* Em Homem-Hora

Figura 3.2. SPL backlog, o product backlog do processo ScrumPL
Fonte: do próprio autor

Ainda na Figura 3.2, os itens PBI1 e PBI2 foram realizados no *sprint* 1. Já os itens PBI3 e PBI4 foram realizados no *sprint* 2. Esses quatro itens estão com a situação "pronto". O *scrum team* está neste momento no *sprint* 3 realizando os itens PBI5, PBI6 e PBI7, que têm situação "selecionado". Os itens PBI8 a PBI12 já tiveram seu esforço definido pelo *scrum team* e estão aguardando serem selecionados em um dos próximos *sprints*, tendo sido feita a estimativa de que os itens PBI8 e PBI9 serão realizados no *sprint*

4 e de que o PBI10 será concluído no *sprint* 6. Observa-se que o PBI10 tem um esforço estimado muito acima dos outros, portanto será necessário dividi-lo em dois ou mais itens a serem realizados nos *sprints* 5 e 6.

Por fim, na figura 3.2 observa-se que cada SPL *backlog item* está associado a um ponto de variação, a uma variante, e a várias aplicações, indicando que irá realizar estas variantes e pontos de variação, bem como fará parte das aplicações marcadas com X.

3.2.2 SPL Backlog Burndown Chart

O SPL *backlog burndown chart* consiste num gráfico de linha, onde o eixo das abscissas representa os *sprints*, ao passo que o eixo das ordenadas representa o número de horas restantes para concluir a aplicação – que corresponde à soma dos tempos estimados de cada item que faz parte da referida aplicação.

Este gráfico evolui no início de cada *sprint*, indicando o esforço restante para concluir a aplicação nesse instante. As linhas do gráfico têm tendência de descida, indicando que o desenvolvimento da aplicação está progredindo. Podem ocorrer subidas na linha, que indicam novos SPL *backlog items* acrescentados ao produto ou uma re-estimativa dos existentes.

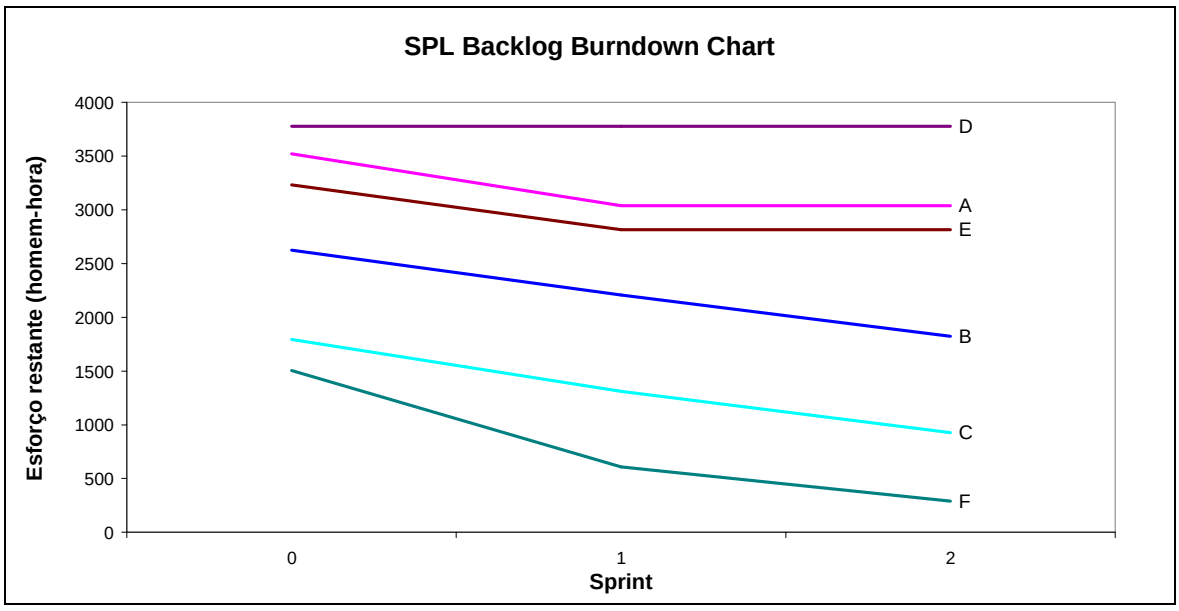

Figura 3.3. SPL backlog burndown chart
Fonte: do próprio autor

A Figura 3.3 mostra o SPL *backlog burndown chart* referente ao SPL *backlog* da Figura 3.2. Diferente de um *burndown chart* comum, este possui as várias aplicações da

SPL, neste caso A, B, C, D, E e F. Também é apresentanda a evolução do esforço restante de cada aplicação correspondente aos *sprints* 1 e 2.

Ainda na Figura 3.3, observa-se que a aplicação F obteve o declive mais acentuado nesses dois *sprints*, isso porque os dois itens realizados durante o *sprint* 1 fazem parte desta aplicação, além de um dos dois itens realizados no *sprint* 2. As aplicações A,B,C e E obtiveram um progresso porque pelo menos um dos itens realizados durante os *sprints* 1 e 2 fazem parte destas aplicações. Já a aplicação D não obteve progresso, porque nenhum dos itens realizados nos *sprints* 1 e 2 fazem parte da mesma. Por fim, observa-se que, ao concluir componentes e outros artefatos comuns a vários produtos, cada produto que usa estes componentes e artefatos terão suas horas restantes reduzidas ao mesmo tempo.

3.2.3 Sprint Backlog

Sprint Backlog (Sprint 3)						Esforço Restante (homem-hora)					
						Dia 1	Dia 2	Dia 3	Dia 4	Dia 5	Dia 6
Product Backlog Item (PBI)	Tarefa	Respon-sável	Status	Estimativa PBI (Horas)	Estimativa Tarefa (Horas)	5/7/2010	6/7/2010	7/7/2010	8/7/2010	9/7/2010	12/7/2010
Esforço restante do *sprint:*				768		752	728	696	664	648	616
PBI5			Selecionado	160		144	120	88	56	40	16
	Tarefa5.1	Mateus	Pronto		16	8	0				
	Tarefa5.2	Marcos	Pronto		8	8	0				
	Tarefa5.3	Lucas	Pronto		12	4	0				
	Tarefa5.4	João	Pronto		8	8	0	0			
	Tarefa5.5	Lucas	Pronto		12	12	8	0			
	Tarefa5.6	Mateus	Pronto		12	12	12	4	0		
	Tarefa5.7	Marcos	Pronto		4	4	4	0			
	Tarefa5.8	João	Pronto		8	8	8	0			
	Tarefa5.9	Marcos	Pronto		12	12	12	8	0		
	Tarefa5.10	Lucas	Pronto		12	12	12	12	4	0	
	Tarefa5.11	João	Pronto		8	8	8	8	0		
	Tarefa5.12	Mateus	Pronto		16	16	16	16	12	4	0
	Tarefa5.13	Marcos	Pronto		8	8	8	8	8	0	
	Tarefa5.14	João	Pronto		12	12	12	12	12	8	0
	Tarefa5.15	Mateus	Selecionado		8	8	8	8	8	16	8
	Tarefa5.16	Lucas	Selecionado		4	4	12	12	12	12	8
PBI6				352		352	352	352	352	352	344
	Tarefa6.1	Marcos	Selecionado		16	16	16	16	16	16	8
	Tarefa6.2	João	Selecionado		8	8	8	8	8	8	8
	Tarefa6.3				12	12	12	12	12	12	12
	Tarefa6.4				8	8	8	8	8	8	8
	Tarefa6.5				4	4	4	4	4	4	4
	Tarefa6.6				12	12	12	12	12	12	12
	Tarefa6.7				16	16	16	16	16	16	16

Figura 3.4. Sprint backlog
Fonte: do próprio autor

Para o acompanhamento do progresso das atividades elaboradas pelo *Scrum team* durante os *sprints* é usado o *sprint backlog*, de acordo com a metodologia *Scrum*. Durante a *sprint planning meeting*, o *Scrum team* preenche o *sprint backlog* com um conjunto de tarefas que julgam ser necessárias para atingir os objetivos do sprint, informando o esforço estimado para concluí-las. Como é regra no método *Scrum*, nenhuma destas tarefas deverá

ter mais de 16 homens-horas estimados. Caso isso ocorra, a tarefa deverá ser dividida em duas ou mais tarefas.

Diariamente, os membros do *Scrum team* selecionam tarefas do sprint *backlog* para realizar, atualizam o esforço restante de cada tarefa que estão realizando, acrescentam novas tarefas ou cancelam tarefas desnecessárias. Cada tarefa está ligada a um SPL *backlog item*, sendo que o esforço restante para concluir este SPL *backlog item* é igual à soma do esforço de cada uma destas tarefas. Este cálculo é feito diariamente até o fim do sprint. A soma dos esforços restantes de cada SPL *backlog* item fornece o esforço restante para o *sprint*, e este pode ser visualizado através de um sprint *backlog burndown chart*.

A Figura 3.4 mostra um exemplo de um sprint *backlog*, que representa as tarefas necessárias para realizar os SPL *backlog*s items "PBI5", "PBI6" e "PBI7". Estes SPL *backlog items* foram selecionados pelo *Scrum team* para o *sprint* 3, durante a *sprint planning meeting*. No SPL *backlog* da Figura 3.2, observa-se que a situação é "selecionado" para "PBI5", "PBI6" e "PBI7". O *Scrum team*, que selecionou e irá realizar estes SPL *backlog items*, é composto por 4 membros: Lucas, Mateus, Marcos e João, que possuem as habilidades necessárias para concluir os objetivos do *sprint*.

No decorrer do *sprint*, cada um destes membros seleciona uma tarefa, atribuindo-a si mesmo como responsável pela mesma e mudando seu status para selecionado. O esforço restante para concluir estas tarefas é atualizando diariamente. Assim que a tarefa é concluída, seu status passa a ser "pronto", e o membro do *Scrum team* seleciona uma nova tarefa disponível para realizar. Os membros do *Scrum team* podem também cancelar ou adicionar tarefas ao *sprint backlog*, se necessário. Para cancelar, basta mudar a situação da referida tarefa para "cancelado". Ao adicionar uma tarefa, o membro do time que a adicionou atualiza diariamente o esforço restante desta, a partir do dia em que a tarefa foi adicionada.

Ainda na Figura 3.4, o PBI5 exige um esforço de 160 homens-hora para sua conclusão, já o PBI6 exige 352, enquanto que o PBI7 256. Estas informações podem ser vistas na coluna estimativa PBI (homem-hora).

Após a *sprint planning meeting*, Lucas, no dia 1 do *sprint*, selecionou a tarefa "Tarefa5.3", já estimada em 12 horas restantes, mudando sua situação para "selecionada". Ao fim do dia 1, atualizou as horas restantes desta tarefa para 4 horas. No dia 2, Lucas concluiu esta tarefa, mudando seu status para "Pronto" e seu esforço para 0 (zero). Em seguida, ele selecionou a tarefa "Tarefa 5.5", que, neste dia, estava disponível (status em branco), mudando seu status para "Selecionado". Ao fim do dia 2, Lucas mudou o esforço

restante desta tarefa para 8 (oito) homens-hora. Todas as tarefas seguem este ciclo, até o fim do *sprint*.

3.2.4 Sprint Burndown Chart

Consiste num gráfico de linha, onde o eixo das abscissas representa os dias de um *sprint*, ao passo que o eixo das ordenadas representa o esforço restante para concluir os objetivos do *sprint*. Este gráfico representa o progresso diário do esforço restante para completar os SPL *backlog items* selecionados, sendo atualizado sempre que os membros do *Scrum team* mudam o esforço restante das tarefas que estão executando, no *sprint backlog*.

Este gráfico é composto de duas linhas, uma chamada de descida ideal e a outra de esforço restante (ver Figura 3.5). A descida ideal é plotada logo no início do *sprint*, e corresponde a um progresso diário uniforme do esforço restante das tarefas. Para plotar esta linha, em cada dia do *sprint* representado no gráfico, o esforço restante estimado do *sprint* é deduzido do decréscimo ideal, uma constante que corresponde ao esforço estimado dividido pelo número de dias do sprint. A linha correspondente ao esforço restante é atualizada uma vez a cada dia, e corresponde à soma do esforço restante de todas as tarefas do *sprint*, calculada após a atualização do esforço restante feito por todos os membros do *Scrum team* ao fim do dia.

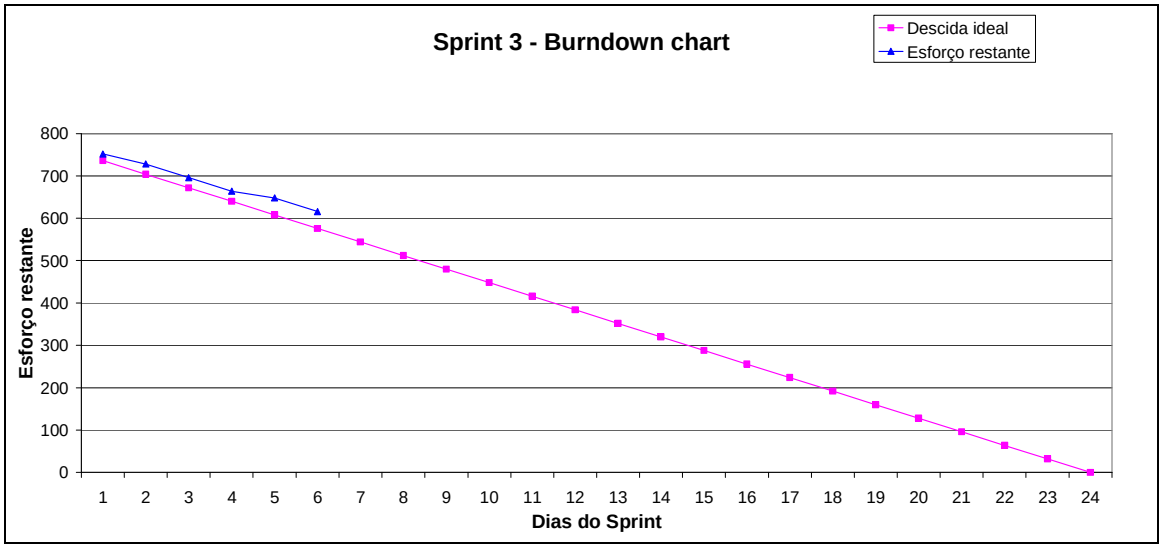

Figura 3.5. Sprint burndown chart
Fonte: do próprio autor

A Figura 3.5 refere-se ao *sprint burndown* chart do sprint 3, que reflete o progresso diário do *Scrum team* de acordo com o esforço restante de cada dia do *sprint backlog* representado na Figura 3.4. O *sprint* 3 apresentado se inicia com um esforço restante estimado de 768 homens-hora, tendo sido planejados 24 dias. O decréscimo ideal é de 32 homens-hora (768 dividido por 24). Assim, o esforço da descida ideal do primeiro dia é de 736, o do segundo dia é de 704, até atingir 0 (zero) no último dia.

Ainda na Figura 3.5, observa-se o esforço restante nos 6 primeiros dias do *sprint* 3. A linha decrescente que representa o esforço restante reflete o esforço restante do *sprint* 3, atualizado diariamente no *sprint backlog* à medida que os membros do *Scrum team* atualizam o esforço restante de suas tarefas. O esforço restante inicial é de 768 homens-hora e reduz ao fim do primeiro dia para 752, terminando o sexto dia em 616.

Fazendo-se um comparativo entre as duas linhas do *sprint burndown chart*, nota-se que o esforço restante está acima descida ideal, o que significa que o *Scrum team* precisa progredir em uma velocidade maior para atingir aos objetivos do *sprint* 3. Se o *Scrum team* manter-se neste ritmo, terá que reduzir os objetivos do *sprint* 3 antes de seu término, o que é previsto pelo método *Scrum*. Essa progressão ainda teve uma pequena redução no dia 5, onde se observa que a inclinação da linha é menor que nos outros dias. Isso porque, conforme pode ser visto no *sprint backlog* da Figura 3.4, a “Tarefa 5.15” teve seu esforço restante aumentado de 8, no dia 4, para 16, no dia 5.

Nesta seção foram apresentadas ferramentas para planejamento e controle do processo *Scrum*PL, que são o SPL *backlog*, o SPL *backlog burndown chart*, o *sprint backlog* e o *sprint backlog* burndown chart. Os dois últimos são os mesmos usados no método *Scrum*, já os dois primeiros são adaptações do *product backlog* e do *product backlog burndown chart* do método *Scrum* para que sejam controladas, ao invés de uma só, várias aplicações – que é uma das principais características das SPLs.

3.3 Comparativo entre o ScrumPL e outros ASPLMs

Nesta seção, o ScrumPL é comparado a outras abordagens que combinam SPLE e métodos ágeis, os ASPLMs, ver quadro 3.1. Esta comparação é feita em relação: à estratégia adotada pelo método para uso da SPLE e do método ágil; ao método ágil adotado; e aos processos e sub-processos da SPLE que utiliza.

No geral, a principal vantagem do ScrumPL em relação aos métodos comparados é que o mesmo possui atividades que permitem a construção de uma SPL desde o início, tanto para artefatos de domínio quanto de aplicação. O ScrumPL também permite mudanças em uma SPL de forma natural, visto que o *product owner* pode fazer modificações sempre que necessário, o que é uma das principais características do método ágil Scrum. Outra vantagem do ScrumPL é o menor esforço de implantação em empresas que já utilizam o método ágil *Scrum* (que é um dos métodos ágeis mais populares), visto que poucas mudanças foram feitas no uso do método *Scrum* para a construção de SPLs.

ASPLMs	**Estratégia**	**Método ágil usado**	**Processos/Sub-Processos da SPLE**
Tian e Cooper (2006)	Compararam os princípios ágeis à SPLE buscando diferenças e semelhanças no que diz respeito a requisitos, qualidade e gerência de projetos.	Princípios ágeis	Processo não definido
Carbol et al (2006)	O framework de SPLE PuLSE-I é usado para desenvolver SPLs, e os métodos ágeis para adaptar a SPL às necessidades dos clientes.	Princípios ágeis	EA
Hanssen e Faegri (2007)	Dada uma SPL existente, atender a novos requisitos de clientes e adaptar a SPL às necessidades dos clientes usando o EVO.	EVO	ED/EA
Noor et al (2008)	Colaboração entre vários *stakeholders* para determinar requisitos e planejar a SPL.	Princípios ágeis	ED/Engenharia de Requisitos
Ghanam e Maurer (2008)	Adaptar produtos existentes para torná-los reutilizáveis e formar uma SPL.	TDD	EA
ScrumPL	**Desenvolver a SPL desde o princípio, identificando requisitos, partes comuns e variáveis através da SPLE; e implementando e testando componentes reutilizáveis e específicos usando o método Scrum.**	**Scrum**	**ED/EA**
Legenda: TDD – Método ágil *Test Driven Development* (Desenvolvimento Orientado a Testes); ED – Processo de Engenharia de Domínio; EA – Processo de Engenharia de Aplicação; EVO – Método ágil EVO.			

Quadro 3.1. Comparativo entre os métodos ágeis de engenharia de linha de produto de software (ASPLMs).
Fonte: do próprio autor

Para o caso da adoção de métodos ágeis para a evolução de SPLs já existentes, ou seja, adaptá-las a novas necessidades de clientes, não há como determinar se é vantajoso ou não usar o ScrumPL, visto que seria necessário fazer uma adaptação no mesmo, considerando processos e artefatos já usados na SPL existente, o que teria como resultado um novo processo.

Se esta SPL existente possuir uma arquitetura de referência e arquiteturas específicas definidas e a serem realizadas, seus componentes e interfaces podem ser adicionados ao SPL *backlog*, na fase de pré-jogo, para desenvolvê-las usando o ScrumPL. Deve-se também levar em consideração o esforço para que o time existente aprenda o ScrumPL, ou um *Scrum team* aprenda a usar a SPL existente.

Em relação ao uso do ScrumPL para a refatoração de uma aplicação para torná-la uma SPL, pode-se usar os artefatos existentes para criar uma arquitetura de referência, durante a etapa de pré-jogo, fazendo mudanças na mesma de forma a atender às novas necessidades de aplicações e identificando artefatos que possam fazer parte da plataforma. A estratégia adotada para o desenvolvimento da plataforma é refatorar os artefatos identificados, de forma a torná-los reutilizáveis. Assim, é necessária a participação da equipe existente e o treinamento da mesma no uso do ScrumPL.

3.4 Conclusão

Neste capítulo foi apresentado o processo ScrumPL, suas etapas, atividades e artefatos, bem como foram feitas comparações entre este processo e outros classificados como APLMs. Uma grande contribuição do ScrumPL em relação ao desenvolvimento de SPLs é que o mesmo usa o método *Scrum* sem muitas adaptações, incorporando as vantagens advindas deste método para o desenvolvimento das aplicações, tais como: (a) equipes auto-gerenciadas; (b) acompanhamento diário das atividades do *Scrum team*, através das *daily meetings*; (c) flexibilidade na execução das tarefas, pois são os próprios membros do *Scrum team* que determinam suas tarefas; (d) eliminação de gargalos, já que os *Scrum team*s são multifuncionais e podem exercer quaisquer funções dentro de um *sprint*. Em comparação a abordagens de SPLE tradicionais há uma vantagem extra que consiste na resposta rápida a mudanças, vantagem esta também citada por Tian e Cooper (2006) para os métodos ágeis em geral.

São utilizados no ScrumPL os seguintes sub-processos da SPLE: gerenciamento de produtos, engenharia de requisitos de domínio, engenharia de requisitos de aplicação, projeto de domínio e de aplicação, especialmente na fase de pré-jogo. Sendo acrescentada uma premissa advinda dos princípios ágeis: a fase de desenvolvimento se inicia assim que há SPL *backlog items* suficientes para serem realizados em um *sprint*. Isso abrevia a fase de pré-jogo, permitindo que o *Scrum team* comece seus trabalhos mais cedo, realizando

suas tarefas para cumprir os objetivos do *sprint*, enquanto o *product* owner continua determinando novos requisitos e modificando os existentes quando necessário, usando os citados sub-processos.

A seguir, o ScrumPL será utilizado para construir uma linha de produto de software para um sistema de navegação de TV para uso em diversas partes do mundo. Este domínio foi escolhido como estudo de caso porque, no início dos estudos, definiu-se uma hipótese de que seriam encontradas variabilidades no mesmo, o que foi confirmado com o andamento da pesquisa.

Como será visto, apesar desse sistema compartilhar características comuns, é necessária uma personalização em massa, visto que atualmente há quatro principais padrões de TV digital no mundo e cada país adotou um. Além do mais, cada país tem um idioma próprio, e os consumidores, dentro destes países, têm poder aquisitivo diferente, onde alguns poderão apenas adquirir produtos básicos, ao passo que outros desejarão obter o que há de mais sofisticado em termos de tecnologia e entretenimento.

Sendo assim, um fabricante de receptores de sinal de TV digital (*Set-Top boxes* – STB) pode levar em conta esses três fatores, padrão de TV digital, idioma e poder aquisitivo dos consumidores, para atender aos seus mercados específicos.

Capítulo 4- Desenvolvimento de um Sistema de Navegação de TV Usando ScrumPL

Um sistema de navegação de TV combina recomendação, classificação e recuperação de programas/canais, tornando mais fácil ao telespectador selecionar o que deseja assistir ou gravar, tomando como base suas preferências ao assistir TV (Isobe et. al, 2003; Peng et. al, 2002). Além dessas características, estes sistemas permitem ao telespectador bloquear programas/canais, reservar programas para assistir ou gravar, bem como visualizar informações, horário de início, duração, categoria, classificação indicativa e canal de um ou mais programas de TV. Neste capítulo será descrita a especificação completa da SPL de um sistema de navegação de TV, desde a definição do sistema e especificação até a arquitetura de referência da SPL e arquiteturas específicas das aplicações. Os artefatos apresentados neste capítulo foram feitos pelo próprio autor, tendo sido validados pelo orientador.

4.1 Introdução ao Sistema de Navegação de TV

Um fabricante que queira desenvolver sistemas de navegação de TV para embarcar em *set-top boxes* (STBs) a serem vendidos em diversas partes do mundo – em especial nas Américas e na Europa – deve levar em consideração características específicas dos vários países em que as aplicações serão introduzidas. Para isso, dois aspectos importantes devem ser considerados: o idioma e o padrão de TV digital utilizado. Além do mais, deve-se levar em consideração o poder de compra dos consumidores em cada um desses países, sendo necessário um terceiro aspecto importante: a segmentação de mercado. A seguir serão discutidos esses três aspectos, que são fundamentais para a elaboração da especificação dos sistemas de navegação de TV a serem introduzidos nas Américas e na Europa.

4.1.1 Idioma

Cada sistema de navegação de TV desta SPL deve conter o(s) idioma(s) oficial(is) do país ao qual se destina. Como serão considerados apenas os países das Américas e da

Europa, entre os principais idiomas estão o inglês, o português, o espanhol, o francês e o alemão. No caso das Américas, serão introduzidos os três primeiros idiomas citados, ao passo que na Europa todos os idiomas serão introduzidos.

4.1.2 Padrão de TV Digital

De acordo com Morris e Smith-Chaigneau (2005), os padrões não são algo que o consumidor de um produto perceba, mas são o que garante que o equipamento que compraram irá executar sua função. Os padrões são usados porque garantem que sistemas que os seguem poderão trabalhar em conjunto, não importa o fabricante.

No caso da TV digital, as emissoras enviam o sinal de TV usando os padrões determinados pela região onde atuam, e os fabricantes constroem produtos aderentes a esses padrões para que possam receber e decodificar esse sinal, transformando-o no conteúdo de vídeo, áudio e dados que o telespectador vê e ouve quando está assistindo TV.

Os padrões de TV digital usados nas citadas regiões são o ATSC (1997), DVB-T (1997), ISDB-TB (2007). Esses padrões de sistema de TV digital adotam diferentes normas para: modulação do sinal de difusão; transporte de fluxo elementar de áudio, vídeo, dados e aplicações; codificação e qualidade de áudio e vídeo; e serviços de *middleware* (Fernandes et al, 2004).

A diferença entre esses padrões não serão abordadas neste trabalho, com exceção das que tratam da programação fornecida pelas emissoras de TV, que são obtidas através do fluxo elementar de dados, usando os padrões que tratam de informações de serviço – *Service Information*, SI (ATSC, 1997; DVB-T, 1997; ISDB-TB, 2007).

Em relação às regiões onde são adotados os padrões de TV digital, o padrão DVB-T é adotado nos países da União Européia, e em outros países tais como Austrália, Nova Zelândia, Malásia, Hong Kong, Cingapura, Índia e África do Sul, entre outros países (Fernandes et al, 2004). O ATSC é conhecido como padrão americano de TV digital e foi adotado nos EUA, Canadá, México e Coréia do Sul, entre outros. Já o ISDB-T é conhecido como padrão japonês. O Brasil realizou adaptações ao ISDB-T, criando o padrão brasileiro de TV digital, o chamando ISDB-TB. Além do Brasil e do Japão mais oito países aderiram ao sistema de TV digital ISDB-TB: Argentina, Chile, Costa Rica, Equador, Filipinas, Paraguai, Peru e Venezuela (ver notícia do Anexo I). Uma versão completa e atualizada

dos padrões de TV digital adotados em vários países pode ser vista em *DTV Status* (2010), onde é apresentado também um quarto padrão de TV digital em uso na China: o DMB.

4.1.3 Segmentação de Mercado

No desenvolvimento destes sistemas de navegação de TV também se deve levar em consideração que não é possível para uma empresa satisfazer todos os consumidores de um dado mercado, sendo necessário fazer uma segmentação do mercado (Batista et al, 2009).

A segmentação do mercado também é um ponto importante para determinar características específicas das aplicações desta SPL. Neste caso, a segmentação foi feita de acordo com o poder de compra do consumidor, separando-o em classes econômicas. No Brasil essas classes econômicas são determinadas pela Associação Brasileira de Empresas de Pesquisa (ABEP), a partir de um critério denominado critério Brasil (ABEP, 2010).

As classes econômicas determinadas pelo critério Brasil podem ser resumidas nas classes A, B e C. Fazem parte da classe A famílias com renda média acima de R$ 4.558, da classe B famílias com renda média acima de R$ 1391,00, e abaixo deste valor fazem parte as famílias da classe C. No critério Brasil também há as classes D e E, que serão representadas, neste trabalho, pela classe C. Desta forma, cada sistema de navegação de TV deverá atender a um dos seguintes seguimentos de mercado: classe A, classe B ou classe C. Nos países que usam o idioma inglês, estes seguimentos de mercado podem ser denominados, respectivamente, *high-end*, *mid-range* e *low-end*.

4.1.4 SPL de um sistema de navegação de TV x desenvolvimento de um único produto

Caso o fabricante queira desenvolver apenas um produto para atender a todos esses mercados, deve levar em consideração que terá que embarcar o hardware de cada padrão, o que é caro, já que não há, até então, software com desempenho satisfatório para processar os sinais destes padrões (Silva e Lovisolo, 2008). Além disso, um só produto com todas as características descritas não teria um preço adequado para a classe C, pois, a nível de comparação, um STB Sky HDTV custa 1.499,00 (ver anúncio do Anexo II) e não possui, por exemplo, recursos de recomendação personalizada, o que é mais do que a renda média familiar da classe C.

Considerando que o fabricante esteja convencido de que um produto só para todas essas regiões seja inviável, e que queira desenvolver produtos específicos para cada região e seguimento de mercado, seria necessário um STB para cada região. Sendo um para a América do Norte com o padrão ATSC, um para a América do Sul, com o ISDB-TB, e um terceiro para a Europa, com o DVT-T, cada um com mais de um idioma para atender cada país. Mas teriam que ser considerados produtos que atendam aos três seguimentos de mercado, considerando características e preço. Além disso, e apesar destas características poderem ser as mesmas em cada uma das regiões e considerando o mesmo seguimento de mercado, o fabricante teria que desenvolver pelo menos nove produtos distintos.

Como são nove produtos distintos que compartilham padrão de TV, características e idiomas, o fabricante pode considerar o desenvolvimento de uma SPL do sistema de navegação de TV de forma a obter os ganhos de qualidade, custo e prazo, já citados.

4.1.5 O ScrumPL como processo de desenvolvimento desta SPL

Visto que o fabricante não possui esse produto, esta SPL seria desenvolvida desde o princípio. Além do mais a SPL deverá ser flexível de forma a incorporar novos requisitos dentro de um mercado dinâmico como o de STBs. Para reduzir riscos e obter lucros mais cedo, o fabricante poderia priorizar o lançamento de um dos produtos o mais cedo possível, mas ainda assim deixando suas partes preparadas para compor os outros produtos, desenvolvidos de forma iterativa e incremental.

Dadas as necessidades de flexibilidade nos requisitos e o desenvolvimento iterativo e incremental, o fabricante pode lançar mãos de APLMs para desenvolver esta SPL. Como a SPL será construída desde o início, o fabricante pode desenvolver estas aplicações através do processo ScrumPL.

No restante deste capítulo será mostrado o desenvolvimento de uma SPL usando o processo ScrumPL. Assim, são descritos o *roadmap* de produtos, os requisitos e a arquitetura de referência de uma SPL de um sistema de navegação de TV que atenda a diversos países, considerando idioma, padrão de TV digital e segmento de mercado. Em seguida são apresentadas três arquiteturas específicas de aplicações desta SPL, a primeira para atender ao mercado latino americano, a segunda para atender ao mercado norte americano e a terceira para atender à união européia. E, por fim é apresentado o SPL *backlog* deste sistema de navegação de TV para que o *Scrum team* possa iniciar a fase de

desenvolvimento. Estes artefatos foram desenvolvidos usando as atividades da fase pré-jogo do processo ScrumPL. Ao fim de cada subseção serão feitos comentários do uso do ScrumPL no desenvolvimento desta SPL.

4.2 Roadmap de Produtos e Requisitos da SPL do Sistema de Navegação de TV

Esta SPL tem como propósito facilitar a seleção de canais pelo telespectador. A partir da plataforma desta SPL serão desenvolvidas aplicações, a serem embarcadas em STBs voltados para os vários países das Américas e da União Européia. Nas subseções a seguir serão descritos o *roadmap* de produtos e os requisitos desta SPL.

Sistema de Navegação de TV – *Roadmap* de produtos					
Visão:	Introduzir STBs, com sistemas de navegação de TV embutidos, nas Américas e na União Européia, no curto prazo, com baixo custo, considerando as classificações econômicas, idiomas e padrões de TV digital específicas de cada região ou grupos de países. Priorizando os 8 países da América Latina que adotaram o ISDB-TB, seguidos das classes B e A, norte americanas e européias, concluindo com as classes C americana e européia.				
ID / **Prioridade**	**Descrição**	**Pontos de Variação e Variantes**			**Período de Introdução da Aplicação**
		Padrão DTV	**Idioma**	**Segmento de Mercado**	
AL_C / 10	Sistema de Navegação de TV para atender à classe C dos seguintes países: Argentina, Brasil Chile, Costa Rica, Equador, Filipinas, Paraguai, Peru e Venezuela	ISDB-TB	Português Espanhol	Classe C	Nov/2010
AL_B / 20	Sistema de Navegação de TV para atender à classe B dos seguintes países: Argentina, Brasil Chile, Costa Rica, Equador, Filipinas, Paraguai, Peru e Venezuela	ISDB-TB	Português Espanhol	Classe B	Fev/2011
AL_A / 30	Sistema de Navegação de TV para atender à classe A dos seguintes países: Argentina, Brasil Chile, Costa Rica, Equador, Filipinas, Paraguai, Peru e Venezuela	ISDB-TB	Português Espanhol	Classe A	Mai/2011
AN_B / 40	Sistema de Navegação de TV para atender à classe B da América do Norte	ATSC	Inglês Espanhol	Classe B	Jul/2011
UE_A / 50	Sistema de Navegação de TV para atender à classe A da União Européia	DVB-T	Inglês Espanhol Francês Alemão	Classe A	Set/2011
AN_A / 60	Sistema de Navegação de TV para atender à classe A da América do Norte	ATSC	Inglês Espanhol	Classe A	Set/2011
UE_B / 70	Sistema de Navegação de TV para atender à classe B da União Européia	DVB-T	Inglês Espanhol Francês Alemão	Classe B	Set/2011
AN_C / 80	Sistema de Navegação de TV para atender à classe C da América do Norte	ATSC	Inglês Espanhol	Classe C	Out/2011
UE_C / 90	Sistema de Navegação de TV para atender à classe C da União Européia	DVB-T	Inglês Espanhol Francês Alemão	Classe C	Out/2011

Quadro 4.1. Roadmap de produtos do sistema de navegação de TV.
Fonte: do próprio autor

4.2.1 Roadmap de produtos da SPL

O Quadro 4.1 representa o *roadmap* de produtos da SPL de um sistema de navegação de TV, nela são apresentados: a visão da SPL, a identificação, prioridade e descrição de cada aplicação da SPL, os pontos de variação e o período desejado para introdução da aplicação no mercado. Observa-se que os IDs (identificadores) das aplicações têm o formato região_classe, onde AL é a região da América Latina, AN é a América do Norte e UE é a União Européia.

Ainda no Quadro 4.1, pode ser notado que a prioridade dada ao desenvolvimento das aplicações segue as determinações fornecidas na Visão, ou seja, a priorização maior foi fornecida aos produtos AL_C, AL_B e AL_A, respectivamente, que correspondem às aplicações destinadas à América Latina para as classes C, B e A. Em relação ao número correspondente à prioridade, quanto menor o número maior a prioridade.

O Quadro 4.1 também identifica os pontos de variação desta SPL, que são: padrão de TV digital, idioma e segmento de mercado. O primeiro tem como variantes os padrões ISDB-TB, DVB-T e ATSC. O segundo tem como variantes Português, Inglês, Espanhol, Francês e Alemão. Já as variantes correspondentes aos seguimentos de mercado são Classe A, Classe B e Classe C.

4.2.2 Requisitos da SPL

Definido o *roadmap* de produtos, a seguir serão descritos os requisitos de domínio desta SPL, que foram definidos usando o sub-processo de Engenharia de Requisitos de Domínio do ScrumPL. Este sub-processo visa extrair os requisitos comuns às várias aplicações definidas. Como no *roadmap* de produtos já foram determinadas as variantes que irão compor cada aplicação, os requisitos serão fornecidos em função dos pontos de variação seguimento de mercado, padrão DTV e idioma, bem como das suas variantes.

Ponto de variação segmento de mercado

No que diz respeito ao ponto de variação segmento de mercado temos as variantes: Classe A, Classe B e Classe C, que determinam características do sistema de navegação de TV destinadas às pessoas que se enquadram nestas classes econômicas descritas no critério Brasil. As características dos produtos voltados para as classes B e C estão descritas em maior detalhe no Quadro 4.2.

A variante Classe C possui as seguintes características: informações do canal atual e bloqueio de programas. Para implementar essas características é necessário apenas obter os dados dos programas de TV diretamente das informações de serviço do canal sintonizado, no momento em que são decodificadas pelo STB, sem a necessidade de armazená-las.

Segmento de Mercado	**Descrição das Características deste segmento**
Classe A Classe B Classe C	**Informações do canal atual**: O telespectador pode rapidamente determinar fatos importantes sobre a programação deste canal, tais como programa atual, próximo programa, programação do dia seguinte, de 10 dias (Basic e Monicic, 2002). Os dados são fornecidos por cada emissora de TV a respeito de sua própria programação. Esta característica provê (Peng et. al., 2002): * **Barra de informações do canal**, que mostra informações a respeito da programação do canal corrente tais como: nome da emissora, logotipo do canal, nome do programa, informações detalhadas do programa atual, duração, data e hora do STB, ao mesmo tempo em que é exibido o programa; * **Guia do canal**, que mostra as informações detalhadas do programa atual e do programa seguinte do canal selecionado, enquanto o programa é exibido numa tela pequena; e * **Guia de programação**, que mostra uma agenda contendo até 7 dias de programação do canal selecionado, enquanto mostra informações detalhadas do programa selecionado na agenda.
	Bloqueio de Programas: Bloqueia conteúdos não apropriados a alguns telespectadores (Basic e Monicic, 2002). Este bloqueio é feito de acordo com uma lista criada e mantida por um telespectador, seguindo também as especificações dos padrões a respeito de bloqueio de canais. Isso permite que o STB seja programado para não mostrar programas ou canais específicos contendo violência ou sexo, já que o telespectador pode ser uma criança no momento.
Classe A Classe B	**Busca de Programas**: Apresenta os programas disponíveis de acordo com um filtro personalizado (Isobe et. al., 2003). A função de busca de programas de acordo com palavras-chave (Isobe et. al., 2003; Basic e Monicic, 2002; Zhang et. al., 2005; Weib et. al., 2008) fornece uma lista de programas que estão de acordo com uma palavra fornecida pelo telespectador, através de um teclado (real ou virtual), de um controle remoto ou selecionada a partir de uma lista fornecida pelo STB. Neste último caso, a lista será criada a partir das informações dos programas disponíveis, armazenados em um banco de dados no STB, e serão organizadas de acordo com o perfil do telespectador.
	Navegação em um guia eletrônico de programação (EPG): O telespectador navega no guia, contendo linhas representando os canais e colunas representando os programas com seus nomes, hora de início e duração. A partir do EPG, o telespectador poderá selecionar ou reservar um programa para gravar, ou assistir agora ou no futuro (Isobe et. al., 2003; Basic e Monicic, 2002).
	Lembrete sobre programas reservados: informa ao telespectador quando um de seus programas, que tenha sido reservado via EPG, está disponível para assistir (Basic e Monicic, 2002).

Quadro 4.2. Características dos produtos voltados para as classes A, B e C.
Fonte: do próprio autor

A variante Classe B engloba as características da variante Classe C e as seguintes: busca de programa, navegação em um EPG e lembrete. Estas características exigem que os dados dos programas de TV sejam armazenados, visto que dizem respeito a vários canais

de TV, não sendo possível obtê-las diretamente do sintonizador, todas ao mesmo tempo, visto que o sintonizador só opera um canal de TV por vez.

Já a variante Classe A possui as características da variante classe B e as seguintes descritas no Quadro 4.3: recomendação personalizada, lembrete personalizado, gravador (*Personal Vídeo Recorder* – PVR), PiP (*Picture in Picture*) e mosaico EPG. Essas características exigem que o produto tenha um dispositivo de armazenamento de alta capacidade – tal como um disco rígido –, um sintonizador secundário e uma função de detecção e armazenamento do perfil do telespectador.

Segmento de Mercado	**Descrição das Características deste segmento**
Classe A	Recomendação personalizada: O sistema de recomendação personalizada (Isobe et. al., 2003; Zhang et. al., 2005; Ehrmantraut et. al., 1996, Weib et. al., 2008) recomenda programas de acordo com as preferências e interesses do telespectador, fornecida implicitamente, onde o sistema rastreia as preferências do telespectador automaticamente, criando seu perfil, ou explicitamente, onde o telespectador define seus canais e tipos de programa favoritos através de uma interface. A partir desse perfil, o sistema poderá selecionar um canal automaticamente ou prover uma lista de canais favoritos para que o telespectador possa escolher qual assistir. A recomendação poderá variar de acordo com o horário (manhã, tarde, noite e madrugada). Pela manhã, por exemplo, o telespectador esteja mais propenso a assistir jornais, informações sobre o tempo ou programas infantis, enquanto que pela noite, ele esteja mais propenso a assistir filmes ou esportes ao vivo (Isobe et al, 2003).
	Gravador (PVR): Possui funções tais como: avançar, retornar, pausar, pular comerciais, gravação automática, replay em programas ao vivo, ângulo de câmera (Han et. al., 2008; Zhang et. al., 2005). A partir da função integrada ao EPG, grava o programa corrente ou futuro.
	PiP (*Picture in Picture*): Esta função apresenta o áudio, vídeo e dados do programa advindo do sintonizador primário, usando toda a área de vídeo da TV. Ao mesmo tempo, é mostrado um quadro pequeno com o vídeo de um programa advindo do sintonizador secundário, que corresponde ao de um outro canal.
	Mosaico EPG: será apresentado um EPG personalizado em forma de mosaico (Han et. al., 2008), mostrando os 9 programas favoritos numa tela onde será exibido o audio e o video do canal sendo selecionado e o vídeo de um dos outros canais a intervalos de 10 segundos; melhoria na atualização dos dados do Banco de Dados dos Serviços de Informação, usando o sintonizador secundário para receber a informação de EPG das emissoras, enquanto o telespectador assiste um programa recebido pelo sintonizador principal.
	Lembrete personalizado: Os programas favoritos serão compilados a partir do perfil do telespectador, armazenado em um banco de dados, e sempre que estiver próximo de iniciar um desses programas favoritos, é mostrada uma mensagem no rodapé da tela informando a respeito do programa, dando a opção do telespectador pressionar uma tecla para assistir a este programa.

Quadro 4.3. Características dos produtos classe A.
Fonte: do próprio autor

Ponto de variação padrão TVD

Em relação ao ponto de variação padrão de TV digital, suas variantes são ATSC, DVB e ISDB. Cada aplicação desta SPL será embarcada em um STB que contenha um

conjunto de hardware e software que forneça as informações de serviço advindas do *transport stream* (TS) de acordo com o padrão (ATSC, DVB-T ou ISDB-TB) correspondente à região em que o STB será vendido. Neste caso, ISDB-TB para os STBs destinados à América Latina, ATSC aos destinados à América do Norte, DVB-T aos destinados à Europa.

Em cada aplicação desta SPL haverá somente um desses três padrões, que são considerados requisitos internos desta SPL, de acordo com o Quadro 4.4. Caberá ao *Scrum team* garantir que tanto as informações de programação de TV comuns a todos os padrões quanto específicas sejam tratadas pelos sistemas de navegação de TV.

Segmento de Mercado	**Descrição dos requisitos internos da SPL**
Classe A	Coletor de perfil do telespectador: Coleta e armazena informações a respeito dos programas assistidos pelo telespectador, tais como gênero, nome, horários, canais, entre outras, de forma implícita ou explícita. Na forma implícita, as informações são coletadas e armazenadas enquanto o telespectador assiste TV. Na forma explícita, o telespectador informa os canais e os gêneros de programas de TV de sua preferência através de uma interface com o usuário.
	Banco de dados de perfis: Armazena informações sobre as preferências do telespectador fornecidas implícita ou explicitamente. Esta informação será usada para recomendação, lembrete sobre programas favoritos e EPG personalizado em forma de mosaico.
Todos	Recepção dos Serviços de Informação: Recebe os serviços de informação a partir do *transport stream* e outras fontes, armazenando-as no banco de dados de serviços de informação.
	Banco de dados de serviços de informação: Armazena as informações da programação das emissoras de TV broadcast, para busca, filtro, classificação e seleção de programas (Isobe et. al., 2003; Zhang et. al., 2005). Esse banco de dados deverá englobar todas as informações relativas a programação transmitidas pelos padrões de TV digital, tanto as que são comuns a todos os padrões quanto informações específicas.
	Suporte a diversos idiomas: Já que esta SPL é projetada para funcionar em diversas partes do mundo, sua interface deve suportar as línguas faladas em cada região a que o sistema de navegação de TV se destina.
	Interface com o usuário: Deve ser fácil de usar e navegar usando o controle remoto. Todos os produtos terão interface similar.
	Padrões de TV Digital: Cada produto irá receber e processar o sinal das emissoras através do receptor dos serviços de informação usando um dos seguintes padrões: ATSC, DVB ou ISDB. Para o sistema de navegação de TV, estes padrões irão fornecer: informações sobre as emissoras de TV, tais como nome e número do canal; região para a qual o sinal está sendo transmitido; informações sobre os programas das emissoras de TV tais como nome do programa, hora de início, duração, classificação indicativa, gênero e sinopse.

Quadro 4.4. Descrição dos requisitos internos da SPL.
Fonte: do próprio autor

Ponto de variação idioma

Em relação aos idiomas, esta SPL fornece suporte a português, inglês, espanhol, francês e alemão, podendo ser acrescentados outros idiomas usados nas regiões em que o

fabricante queira introduzir seus STBs. Mais de um idioma poderá ser usado em um produto, desde que a região de destino suporte.

Requisitos internos da SPL

O Quadro 4.4 fornece informações sobre outros requisitos necessários ao desenvolvimento da SPL, que não dependem dos seguimentos de mercado, mas que também farão parte dos sistemas de navegação de TV advindos desta SPL. Esses requisitos dizem respeito à recepção do sinal da emissora de TV digital e armazenamento das informações de serviços, às especificações de interface e à definição dos padrões de TV digital.

4.2.3 Atividades de elaboração do roadmap de produtos e dos requisitos da SPL

Para a elaboração do *roadmap* de produtos e dos requisitos do sistema de navegação de TV, durante a etapa de planejamento, foram seguidos os seguintes sub-processos do ScrumPL: gerenciamento de produtos, engenharia de requisitos de domínio e engenharia de requisitos de aplicação.

As atividades do processo de gerenciamento de produtos trataram da definição do escopo da SPL (Pohl et al, 2005), que dizem respeito a capturar o contexto e os requisitos mais importantes de forma a definir os critérios de aceitação das aplicações. Sendo assim, para capturar o contexto e os requisitos mais importantes desta SPL usou-se uma abordagem descrita por Pietro-Diaz e Arango (1991), denominada engenharia de conhecimento (*knowledge-engineering*), que assume que o conhecimento que precisamos existe em algum lugar, e assim determina os seguintes passos para obtê-los:

1. Identificar as fontes de conhecimento corretas;
2. Adquirir o conhecimento existente;
3. Representar este conhecimento explicitamente.

Para identificar as fontes de conhecimento corretas, feita uma pesquisa bibliográfica sobre os temas TV digital e Guia Eletrônico de Programação (EPG), a partir de artigos publicados em congressos e revistas bem qualificados. Como resultados foram encontrados 48 artigos (ver Anexo III).

Para a seleção desses artigos foi utilizado o método descrito por Keshav (2007), que consiste num método de 3 passos para leitura de artigos científicos de forma prática e eficiente. Assim, foi feita uma classificação dos artigos, considerando relevância ao tema e inovação, e posterior escolha, resultando em 12 artigos relevantes. Considerando o quesito inovação, foi mudado o tema de EPG para Sistema de Navegação de TV, por este segundo ser mais atual e também englobar o EPG. Este trabalho cobriu os passos 1 e 2 descritos.

A representação explícita do conhecimento (passo 3), considerando a captura do contexto e dos requisitos do sub-processo de gerenciamento de produtos, foi feita através de descrição textual das definições do Sistema de navegação de TV e do *roadmap* de produtos. O passo 3 também se estendeu para o sub-processo de engenharia de requisitos de domínio, como será visto a seguir.

As atividades da engenharia de requisitos de domínio trataram de definir os requisitos das aplicações e suas variabilidades, representando-as através do modelo de variabilidade ortogonal e de descrições textuais desses requisitos. Para isso, foram revistos os artigos selecionados de forma a identificar a totalidade dos requisitos dos sistemas de navegação de TV.

Avaliando detalhadamente a totalidade desses requisitos, observou-se que todos poderiam fazer parte de um produto só, mas que este produto deveria ter componentes de hardware mais sofisticados, tais como um dispositivo de armazenamento de grande capacidade, sintonizador secundário, bom poder de processamento e bastante memória. Assim, percebeu-se que seria necessário fazer uma segmentação por classes econômicas, visto que este produto em sua totalidade não poderia ser vendido à Classe C. Um exemplo disso é o preço do já citado receptor *Sky* HDTV, que custa 1.499,00 (ver Anexo II) que é bastante caro para a Classe C brasileira.

Também se observou que esse conjunto de requisitos poderia atender aos consumidores de várias partes do mundo, visto que os artigos selecionados diziam respeito a pesquisas feitas principalmente no Japão, na Europa, na Austrália e nos EUA, que usam os padrões de TV digital ISDB-TB, DVB-T e ATSC, respectivamente. A terceira observação diz respeito ao idioma, que são diferentes nestes vários países, especialmente os da Europa.

Essas observações determinaram os pontos de variação e as variantes da SPL. Para representá-los definiu-se o *roadmap* de produtos, como já visto, e um modelo de variabilidade ortogonal (representado na próxima seção). Já os requisitos textuais foram representados nos quadros 4.2, 4.3 e 4.4.

Em relação à atividade de engenharia de requisitos de aplicação, foi feita apenas uma análise de forma a tentar identificar requisitos mais específicos em cada aplicação, mas observou-se que os requisitos já encontrados durante a engenharia de requisitos de domínio foram suficientes. Além disso, o próprio *roadmap* de produtos determinou quais variantes iriam fazer parte de cada aplicação. Assim, os requisitos definidos para essas variantes determinam os requisitos específicos de cada aplicação.

4.3 Arquitetura de Referência do Sistema de Navegação de TV

A arquitetura de referência é uma arquitetura central que captura o projeto de alto nível para as aplicações de uma SPL. Ela contém pontos de variação e variantes documentados em um modelo de variabilidade e realizados por componentes (Pohl et al, 2005). A seguir serão descritos os diagramas que fazem parte da arquitetura de referência da SPL do sistema de navegação de TV: modelo de variabilidade, diagrama de componentes de domínio e os diagramas de componentes de 3 aplicações.

Considerando a estratégia adotada para o desenvolvimento (Nyholm, 2002), optou-se por evoluir uma nova SPL, visto que todos os componentes da SPL são novos e que farão parte da plataforma.

4.3.1 Modelo de Variabilidade da SPL

Este modelo define a variabilidade de uma SPL, e relaciona essa variabilidade a outros modelos tais como *feature model*, modelos de caso de uso, de projeto, de componentes e de teste (Pohl et. al., 2005). O modelo de variabilidade é representado por um diagrama de variabilidade contendo os pontos de variação e suas variantes, que são ligados entre si e entre outros diagramas.

A Figura 4.1 representa o diagrama de variabilidade desta SPL, formada por 3 pontos de variação (representados por triângulos): Idioma, Segmento de Mercado e Padrão TVD; e suas variantes (representadas por retângulos). Os pontos de variação e variantes são ligados, através de setas tracejadas, aos pacotes e componentes que os realizam.

O ponto de variação Idioma possui variantes representando os idiomas das regiões onde os produtos serão vendidos. Uma seta tracejada aponta para o componente Idioma, que irá realizar o referido ponto de variação.

O ponto de variação Seguimento de Mercado possui as variantes: Classe A, Classe B e Classe C. As setas tracejadas apontam para os pacotes, de mesmo nome, que contêm os componentes que irão realizar essas variantes. Os componentes desses pacotes são vistos na Figura 4.2, a qual será descrita mais adiante.

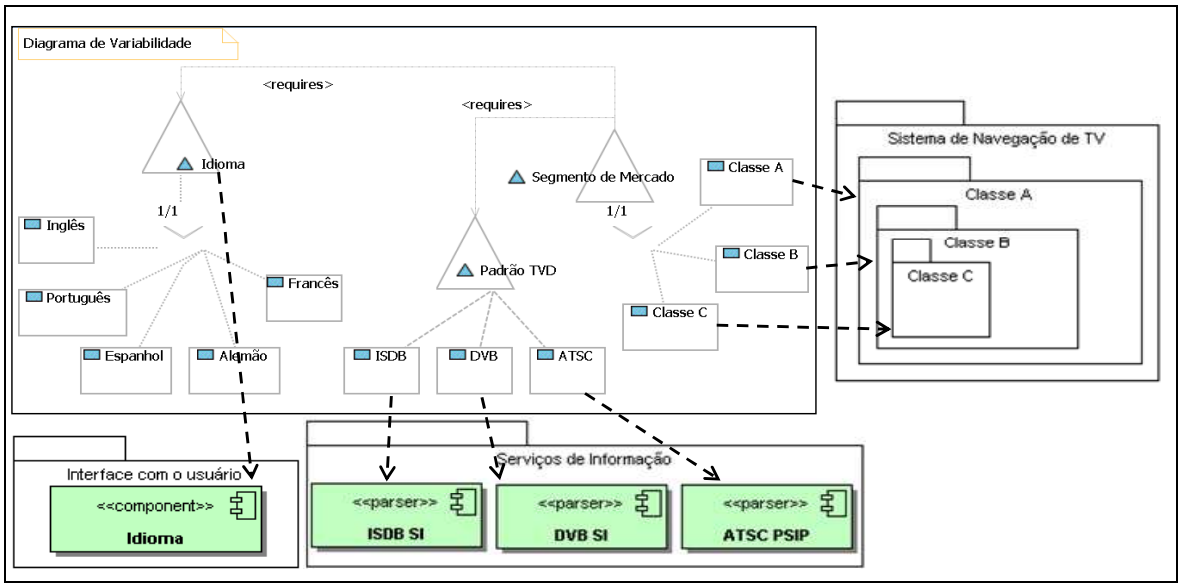

Figura 4.1. Diagrama de variabilidade e de componentes do sistema de navegação de TV
Fonte: do próprio autor

O ponto de variação Padrão TVD tem variante representando os padrões de TV Digital ISDB-TB, DVB-T e ATSC, ou seja, suas especificações correspondentes aos Serviços de Informação. Essas variantes são realizadas pelos componentes “ISDB SI”, “DVB SI” e “ATSC SI”, contidos no pacote Serviços de Informação.

Esses componentes irão implementar as seguintes especificações de serviços de informação: “*Program and System Information Protocol for Terrestrial Broadcast and Cable* (PSIP)”, para o ATSC, “*Specification for Service Information*”, para o DVB-T e “Televisão digital terrestre – Multiplexação e serviços de informação (SI). Parte 2: Estrutura de dados e definições da informação básica de SI”, para o ISDB-TB.

4.3.2 Diagrama de Componentes

O diagrama de componentes é composto de componentes e suas interações, cada componente realiza uma ou mais tarefas específicas dentro do software e se comunica com outros através de suas interfaces. Os requisitos destes componentes estão descritos na seção 4.1, nos quadros 4.2, 4.3 e 4.4, havendo rastreabilidade pelo nome dos mesmos.

A Figura 4.2 mostra o diagrama de componentes desta SPL, formando um segundo elemento importante dentro da arquitetura de referência. Ele representa o sistema através de três pacotes: serviços de informação, sistema de navegação de TV e interface com o usuário, que também foram representados na Figura 4.1. Cada pacote será descrito nas subseções a seguir.

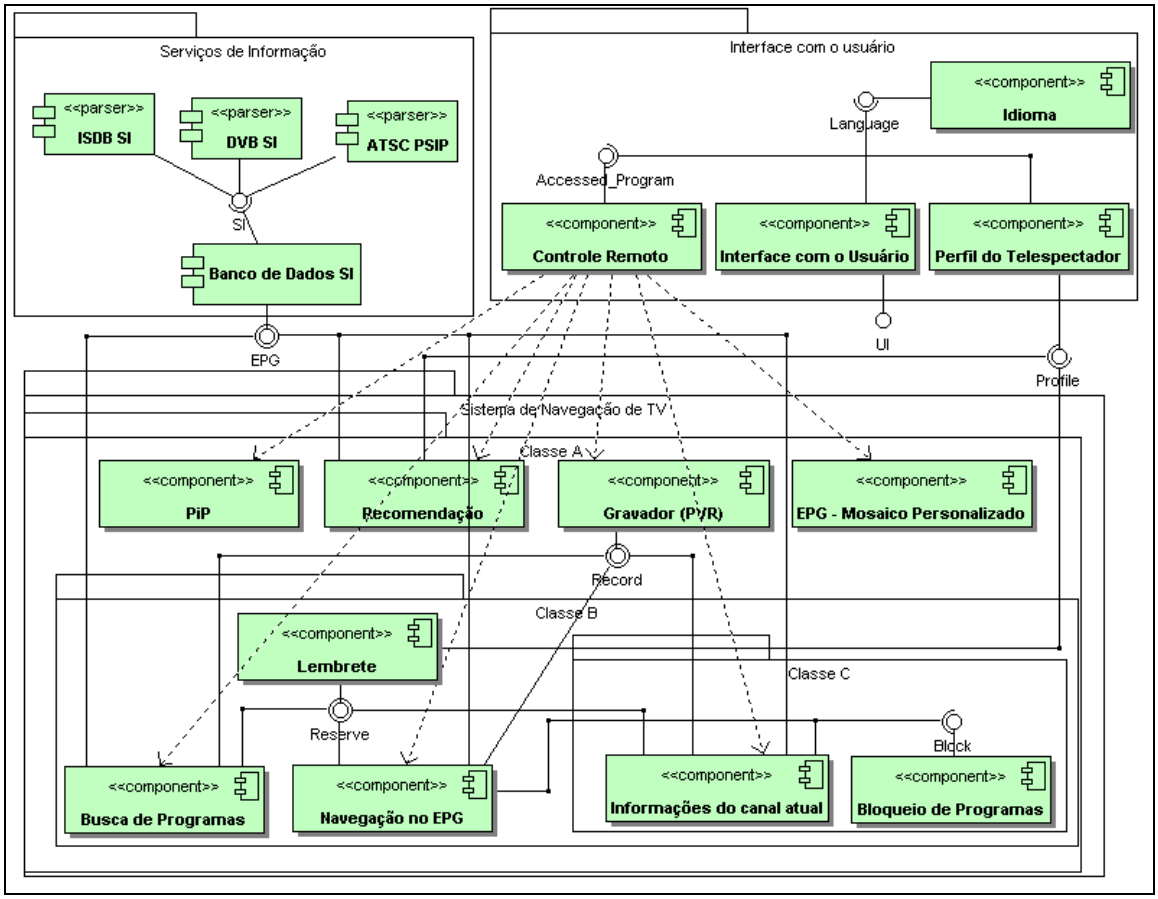

Figura 4.2. Diagrama de componentes de domínio do sistema de navegação de TV
Fonte: do próprio autor

4.3.3 Pacote Serviços de Informação

O pacote Serviços de Informação contém os componentes que irão receber o sinal da emissora de TV, de acordo com os padrões de TV digital, assim como o componente

Banco de Dados de SI, que é responsável por armazenar as informações dos serviços advindas das emissoras de TV e fornecer: nome e número do canal das emissoras, bem como nome, hora de início, duração, atores/participantes e detalhes dos programas de TV aos componentes do pacote sistema de navegação de TV.

As informações sobre os programas de TV são recebidas através da interface SI, que deverá ser implementada pelos componentes correspondentes aos padrões de TV digital “ATSC PSIP”, “DVB SI” e “ISDB SI”. As informações advindas das emissoras de TV, depois de armazenadas, são fornecidas a outros componentes através da interface EPG. O armazenamento é feito visto que cada emissora envia somente informações a respeito de sua programação, desta forma, ser for necessário ver a programação de duas ou mais emissoras ao mesmo tempo, seriam necessárias várias trocas de canais para obter as informações. Sendo assim, cada vez que um canal é selecionado pelo telespectador, sua programação é obtida e armazenada, e enquanto o STB está em *standby*, também é feita a atualização do banco de dados de SI.

A interface EPG fornece uma lista de canais, uma lista de programas de cada canal, bem como informações específicas de cada programa de um canal, tais como hora de início, duração, nome, sinopse, classificação indicativa, gênero, entre outras. Esta interface será acessada principalmente pelos componentes do pacote sistema de navegação de TV.

4.3.4 Pacote Sistema de Navegação de TV

Este pacote contém os componentes que implementam as características atribuídas a cada segmento de mercado, descritos nos Quadros 4.2, 4.3 e 4.4. Esses componentes estão contidos nos pacotes “Classe A”, “Classe B” e “Casse C”. Sendo assim, o pacote “Classe C” contém os componentes: “Bloqueio de Programas” e “Informações do canal atual”. O pacote “Classe B” contém o pacote “Classe C” e os componentes: Lembrete, “Busca de Programas” e “Navegação no EPG”. O pacote “Classe A” contém o pacote “Classe B” e os componentes Recomendação, “Mosaico EPG” e Gravador (PVR).

Os componentes “Busca de Programas”, “Navegação no EPG”, “Informações do canal atual” e Recomendação acessam a interface EPG. O primeiro identifica, dentre os programas armazenados, os que atendem a palavra chave fornecida pelo telespectador. O segundo recebe a lista de canais e todos os programas para formar a tabela contendo linhas representando os canais e colunas representando os programas, dentro do horário e

duração. O terceiro componente recebe a lista de programas e suas informações detalhadas do canal atual. O último recomenda canais para o telespectador assistir, de acordo com o seu comportamento ao assistir TV.

O componente “Bloqueio de Programas” fornece a interface *Block*, que permite ao telespectador definir os programas e os canais bloqueados, ou seja, que não poderão ser acessados sem autorização. Sempre que houver mudança de canal, esse componente será acessado de forma a bloquear ou não o acesso ao canal, autorizando o acesso através de uma senha. O bloqueio será feito aos programas determinados pelo usuário ou de acordo com as normas dos padrões, que no caso do sistema brasileiro corresponde à norma sobre receptores, que pode ser por idade (18, 16, 14, 12 anos ou Livre) ou classificação do conteúdo (violência, sexo, etc.) (ABNT, 2008).

O componente Lembrete fornece a interface Reserve, que será acessada pelos componentes “Busca de Programas”, “Navegação no EPG”, “Informações do canal atual” e “EPG – Mosaico personalizado” para reservar um programa definido pelo telespectador. Este componente também acessará o perfil do telespectador para identificar seus programas favoritos. Sempre que um programa reservado iniciar, este componente apresentará uma mensagem no centro da tela de TV perguntando se o telespectador deseja assitir o programa que reservou. Para o caso de programas favoritos, será mostrada uma mensagem discreta na parte de baixo do vídeo informando o nome de um programa que o telespectador potencialmente deseja assistir, no momento em que este programa de TV esteja começando.

O componente Gravador (PVR) fornece a interface Record, que provê todas as funcionalidades descritas no Quadro 4.3, correspondentes ao requisito Gravador (PVR). Esta interface será acessada pelos componentes: Lembrete, Busca de Programas, Navegação no EPG e Informações do canal atual, bem como diretamente pelo controle remoto, para realizar as funções gravar, pausar, voltar, entre outras.

4.3.5 Pacote Interface com o Usuário

O pacote interface com o usuário contém os componentes que irão receber informações do usuário através do controle remoto ou outro(s) dispositivos e fornecer informações através de telas reproduzidas na TV. Essas informações serão fornecidas no(s) idiomas(s) definidos para cada aplicação.

O componente “Controle Remoto” faz chamadas aos componentes do pacote “Sistema de Navegação de TV” de acordo com a solicitação do telespectador. No diagrama de componentes da Figura 4.2, a chamada a esses componentes é definida pelas setas tracejadas direcionadas a eles. Nas aplicações classe B e A, este componente fornece a interface *Accessed_Programs* para indicar os programas acessados pelo telespectador e, assim formar o seu perfil através do componente perfil do telespectador. Há outras funções deste componentes que podem não fazer parte do escopo desta SPL, tais como ajustes de contraste, tamanho de tela, entre outros, que podem fazer parte de outras aplicações embarcadas no STB, mas não dos sistemas de navegação de TV desta SPL.

Os componentes “Interface com o Usuário” e Idiomas são responsáveis por apresentar as telas ao telespectador através da TV num determinado idioma. Essas telas são acionadas pelos componentes do pacote “Sistema de Navegação de TV” através da interface UI, em seguida as telas serão formatadas e seus textos serão apresentados após serem obtidos a partir da interface *Language*, do componente Idioma, de acordo com o idioma pré estabelecido. Para facilitar o entendimento do diagrama de componentes da Figura 4.2, as conexões à interface UI foram omitidas.

Em cada aplicação desta SPL, o componente Idioma será formado por sub-componentes representando apenas os idiomas destinados à aplicação. Esses sub-componentes implementam as variantes definidas no diagrama de variabilidade, na Figura 4.1, para o ponto de variação Idioma.

O componente “Perfil do Telespectador” será embarcado apenas em aplicações referentes às variantes “Classe A” e “Classe B”. Este componente obtém e armazena informações a respeito do perfil do telespectador baseado em seu comportamento ao assistir programas de TV, indicando os canais e programas que mais assiste, bem como os gêneros de programas. Essas informações são fornecidas através da interface *Profile*, que é acessada pelos componentes Lembrete e Recomendação.

O componente Lembrete utiliza as informações de perfil para lembrar ao usuário de que seus programas de TV favoritos estão prestes a iniciar. Já o componente Recomendação utiliza as informações de perfil para recomendar um ou mais canais preferidos do telespectador de acordo com o horário.

Nesta subseção (4.2), foram apresentados os pontos de variação desta SPL, suas variantes e os componentes e interfaces que os realizam. Estes pontos de variação e variantes, juntamente com o *roadmap* de produto mostrado na subseção 4.1, foram usados para selecionar os componentes e interfaces que fazem parte de cada aplicação desta SPL.

Na subseção 4.3 serão apresentadas três dessas aplicações. A seguir será mostrado como o ScrumPL foi usado para determinar a arquitetura de referência.

A arquitetura de referência da SPL do sistema de navegação de TV foi definida usando-se o sub-processo projeto de domínio, onde, a partir dos requisitos de domínio e de aplicação e do modelo de variabilidade definiu-se um diagrama (ver Figura 4.2) para determinar a estrutura do software e suas partes a serem realizadas, que são ligadas aos pontos de variação e às variantes do modelo de variabilidade da Figura 4.1.

Na primeira versão desta arquitetura de referência, foram definidos três diagramas de componentes, cada um representando um ponto de variação desta SPL. Mas percebeu-se que seria melhor criar pacotes para representar os pontos de variação e agrupá-los em um só diagrama de componentes. Após esse agrupamento, foram feitas análises para facilitar o reuso dos componentes, assim as interfaces e outros componentes – tais como “Banco de Dados SI” e “Interface com o Usuário”, por exemplo – foram definidos de forma a reduzir as mudanças causadas pelas variações da SPL (ver Figura 4.2). Por fim houve uma revisão deste diagrama de componentes, compondo a versão atual mostrada na Figura 4.1.

4.4 Arquiteturas das Aplicações

A seguir serão mostradas as arquiteturas das seguintes aplicações: AL_C, EU_B e AN_A, que representam, respectivamente, aplicações voltadas para: o segmento de mercado classe C da América Latina, classe B da Europa e classe A da América do Norte. Desta forma, para criar as arquiteturas específicas destas aplicações, foram usadas as regras definidas na arquitetura de referência em relação aos componentes e pacotes que realizam cada variante.

4.4.1 Arquitetura da aplicação para a América Latina classe C

Para o mercado da América Latina, classe C, será embarcada no STB a aplicação AL_C com as variantes Português e Espanhol, “ISDB SI”, e “Classe C”. A Figura 4.3 representa a arquitetura desta aplicação, onde foi selecionado o componente “ISDB SI” para receber as informações de serviço. Considerando as características, foram escolhidos os componentes contidos no pacote “Classe C,” que implementam a variante de mesmo nome. Considerando o idioma, foram escolhidos os componentes Português e Espanhol.

Todos os componentes desta arquitetura pertencem à arquitetura de referência, portanto serão reutilizados da mesma e integrados, para formar a aplicação AL_C.

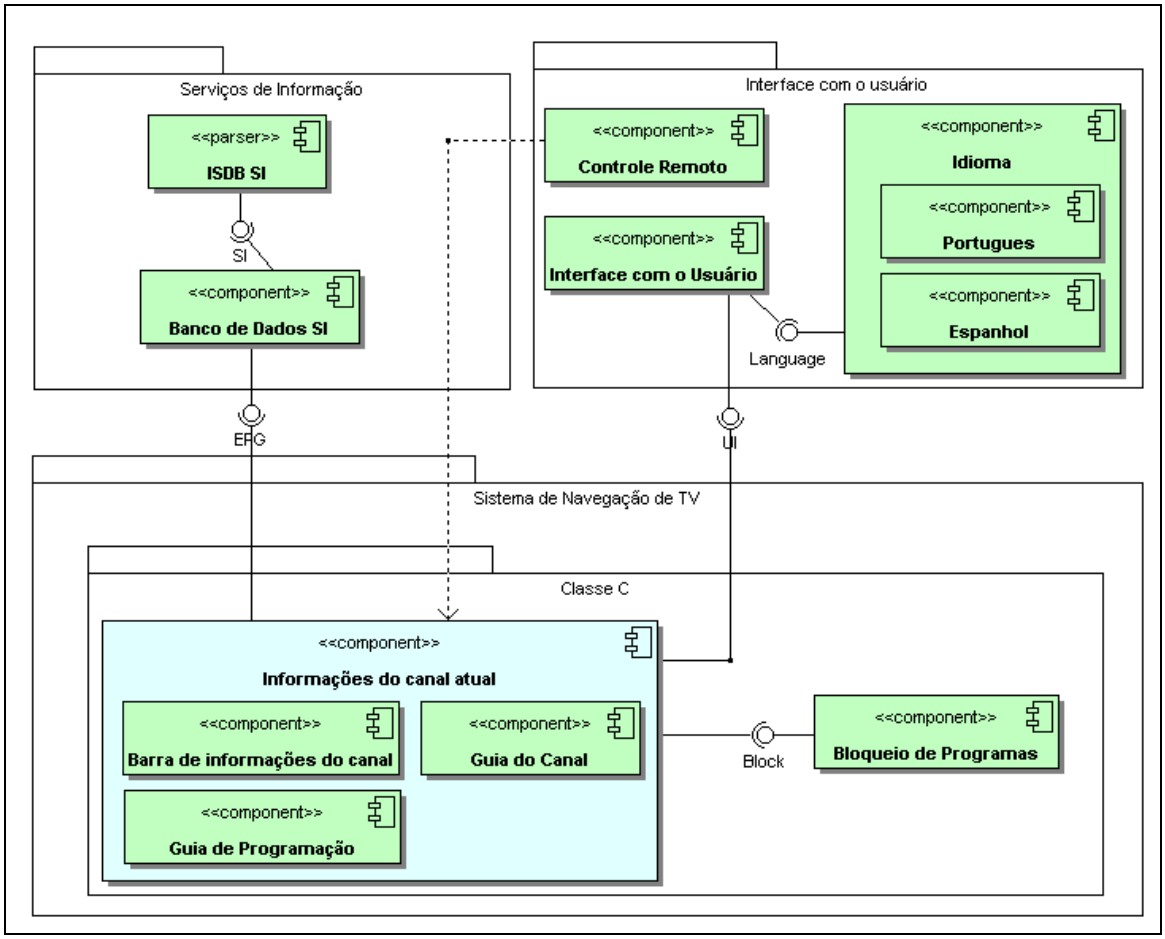

Figura 4.3. Arquitetura da aplicação para o segmento de mercado classe C da América Latina
Fonte: do próprio autor

4.4.2 Arquitetura da aplicação para a Europa classe B

Para o mercado da Europa, classe B, será embarcada no STB a aplicação EU_B com as variantes Português, Inglês, Espanhol, Francês e Alemão, DVB, e "Classe B". A Figura 4.4 representa a arquitetura desta aplicação, onde foi selecionado o componente "DVB SI" para receber as informações de serviço. Considerando as características, foram escolhidos os componentes contidos nos pacotes "Classe B" e "Classe C", que implementam as variantes de mesmo nome. Considerando o idioma, foram escolhidos os componentes Português, Inglês, Espanhol, Francês e Alemão. Todos os componentes desta arquitetura pertencem à arquitetura de referência, portanto serão reutilizados da mesma e integrados, para formar a aplicação EU_B.

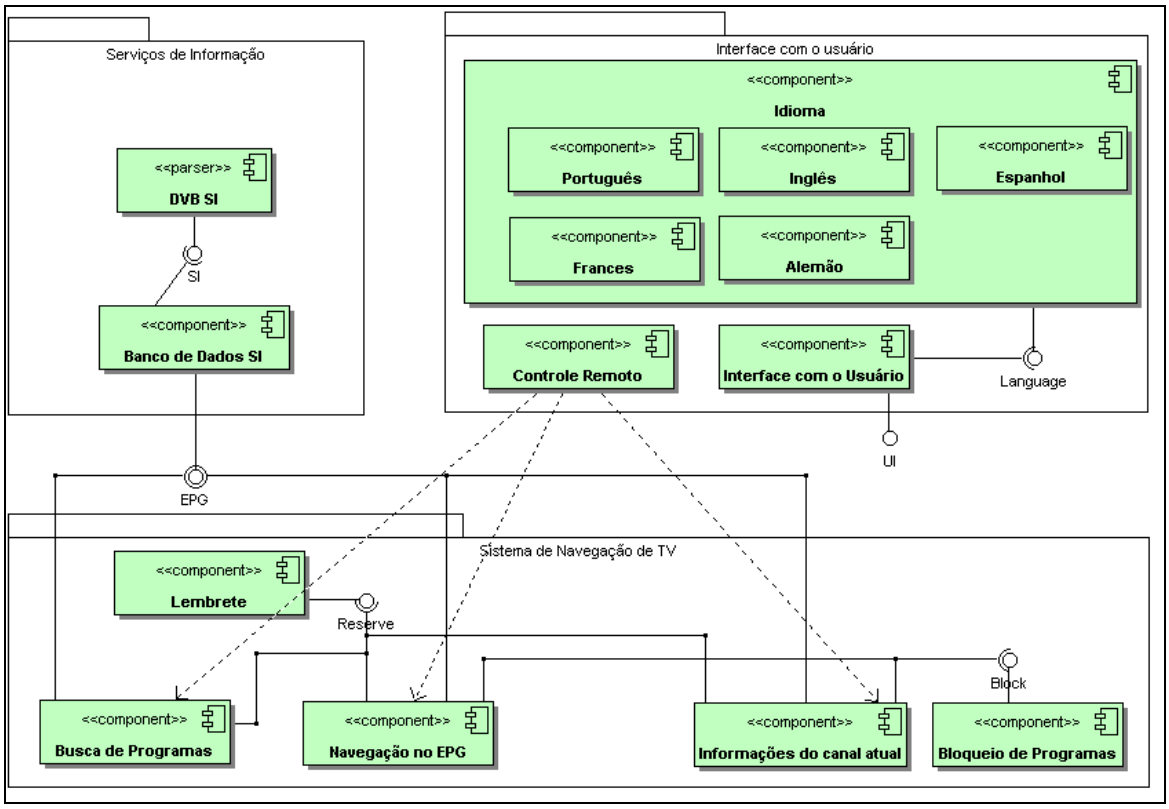

Figura 4.4. Arquitetura da aplicação para o segmento de mercado classe B da Europa
Fonte: do próprio autor

4.4.3 Arquitetura da aplicação para a América do Norte classe A

Para o mercado da América do Norte, classe A, será embarcada no STB a aplicação AN_A com as variantes Inglês, Espanhol, ATSC e Classe A. A Figura 4.5 representa a arquitetura desta aplicação, onde foi selecionado o componente “ATSC PSIP” para receber as informações de serviço. Considerando as características, foram escolhidos todos os componentes contidos no pacote “Sistema de Navegação de TV”, que implementam todas as variantes do ponto de variação “Segmento de Mercado”. Considerando o idioma, foram escolhidos os componentes Inglês e Espanhol. Todos os componentes desta arquitetura pertencem à arquitetura de referência, portanto serão reutilizados da mesma e integrados, para formar a aplicação AN_A.

Definidas as arquiteturas de referência e as específicas, o próximo passo é definir a versão final do SPL *Backlog* para que seja iniciada a fase de desenvolvimento. O SPL *Backlog* será visto na subseção 4.4. A seguir será apresentado como o foram feitas as arquiteturas específicas a partir do processo ScrumPL.

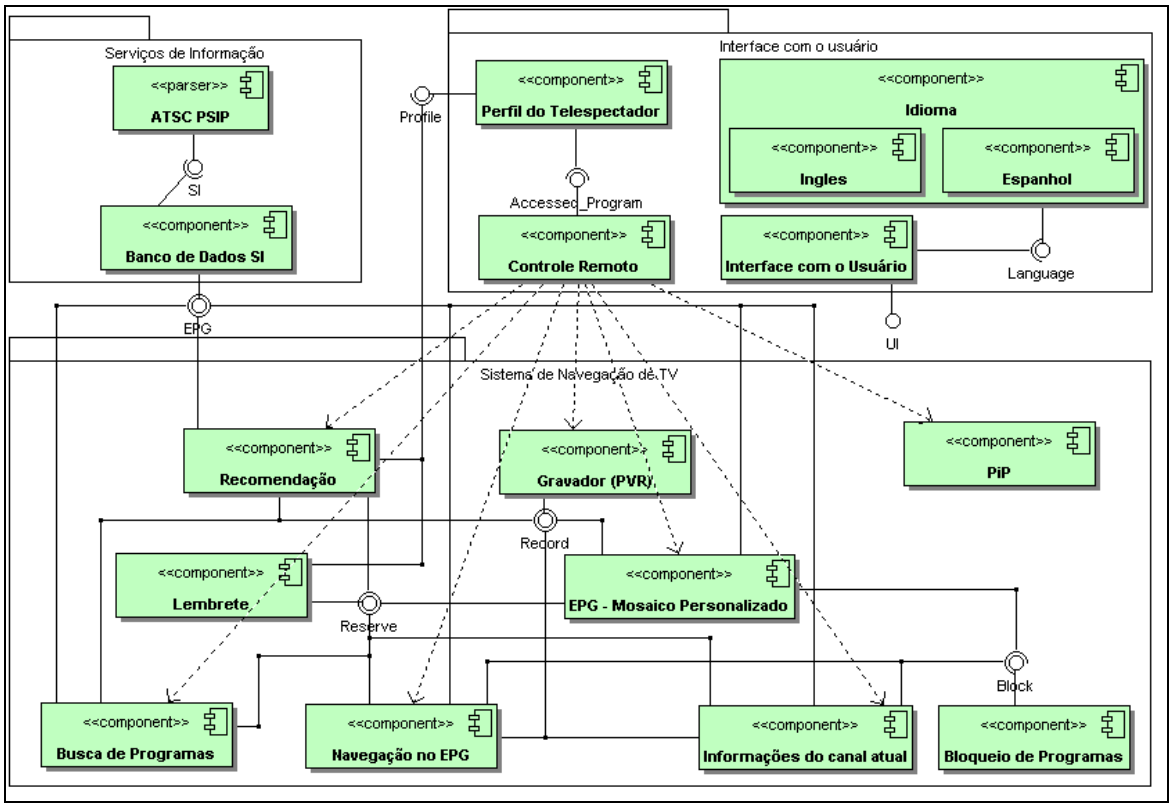

Figura 4.5. Arquitetura da aplicação para o segmento de mercado classe A da América do Norte
Fonte: do próprio autor

O sub-processo do processo ScrumPL usado para determinar as arquiteturas específicas destes produtos é o Projeto de Aplicação. Suas principais atividades são: identificar, na arquitetura de referência, as variantes que fazem parte de cada aplicação para derivar a arquitetura da aplicação, e introduzir nesta arquitetura novos artefatos para atender a requisitos específicos da aplicação. Essas atividades são feitas durante a fase de Projeto.

Como não existiram requisitos específicos para nenhuma aplicação desta SPL, para determinar as arquiteturas das aplicações, foram identificadas, na arquitetura de referência, as variantes que fazem parte de cada aplicação e foram selecionados os pacotes e componentes que as realizam.

4.5 SPL Backlog do Sistema de Navegação de TV

Este SPL *backlog* contém uma lista priorizada de requisitos, componentes e interfaces representados como SPL *backlog items* para que o *Scrum team* possa determinar o esforço estimado em homem-hora para realizar cada SPL *backlog item* e escolher os que irão realizar em cada sprint.

A Figura 4.6 mostra o SPL *backlog* do sistema de navegação de TV. As aplicações definidas são as mesmas que estão no *roadmap* de produtos, os locais marcados com ‘x’ indicam os SPL *backlog items* que fazem parte dessas aplicações.

A priorização foi feita de acordo com o *roadmap* de produtos, definindo os mesmos valores para os SPL *backlog items* correspondentes às aplicações priorizadas no *product roadmap*. A coluna variante tem como propósito facilitar o controle das variantes que são realizadas por cada SPL *backlog* item. Este SPL *backlog* está pronto para ser estimado pelo *Scrum team* para dar início à fase de desenvolvimento.

Este SPL *backlog* foi desenvolvido através dos sub-processos do ScrumPL: engenharia de requisitos de domínio, projeto de domínio, engenharia de requisitos de aplicação e projeto de aplicação. Durante a execução de cada um destes sub-processos foram feitas atualizações no SPL *backlog*, até a conclusão da fase de projeto.

Observando o SPL backlog da Figura 4.6, vê-se que a aplicação AL_A, voltado para a América Latina classe C, será o primeiro a ser entregue, pois foi dada prioridade maior para todos os SPL *backlog items* desta aplicação. Assim que esta aplicação estiver concluída, outras farão uso de seus componentes. Por exemplo, para concluir a aplicação AN_C, basta concluir os SPL *backlog items* “ISDB SI <<parser>>” e Inglês. Mas como a prioridade é a entrega da aplicação AL_B, serão realizados, em seguida, os SPL *backlog items* correspondentes à variante “Classe B”.

Como o *scrum team* ainda não estimou o esforço dos itens prioritários, ainda não foi definido em quais *sprints* cada SPL *backlog item* será realizado. Mas a estimativa deverá respeitar o período de introdução das aplicações definidas no *roadmap* de produtos. Observa-se também na Figura 4.6 que foram inseridos SPL *backlog items* correspondentes à integração e aos testes das aplicações. Essas atividades deverão ser realizadas após a realização de todos os componentes de cada aplicação, durante a fase de entrega, conforme o processo ScrumPL.

A integração diz respeito à compilação e construção de todos os componentes de uma aplicação, que estará pronta para ser embarcada no STB, para que os testes de integração possam ser executados. Em relação aos SPL *backlog items* relativos a esses testes, os mesmos terão atividades de: especificação de testes, revisão de artefatos e código, e execução de testes de integração e de testes de regressão. Essas atividades serão controladas através do *sprint backlog*.

A elaboração do SPL *backlog* da Figura 4.6 foi feita através dos sub-processos engenharia de requisitos de domínio e projeto de domínio do ScrumPL. Através do

primeiro sub-processo foram definidas as aplicações e as prioridades das mesmas, bem como as variantes. Tendo sido acrescentadas ao SPL *backlog* somente as aplicações, visto que as variantes devem ser acrescentadas juntamente com os componentes e interfaces que irão realizá-las. Através do sub-processo projeto de domínio foram acrescentados ao SPL *backlog* os componentes, interfaces, as variantes que realizam e os pontos de variação. Por fim, a priorização dos SPL *backlog items* foi feita com base priorização das aplicações advindas do *roadmap* de produtos.

SPL Backlog								Aplicações								
SPL Backlog Item	Estória	Priori-dade	Esforço * estimado	Sprint	Situação	Ponto de Variação	Variante	AL_C	AL_B	AL_A	UE_C	UE_B	UE_A	AN_C	AN_B	AN_A
Pacote Serviços de Informação																
ISDB SI <<parser>>		10				Padrão de TVD	ISDB	x	x	x						
DVB SI <<parser>>		50				Padrão de TVD	DVB				x	x	x			
ATSC PSIP <<parser>>		40				Padrão de TVD	ATSC							x	x	x
SI <<interface>>		10						x	x	x	x	x	x	x	x	x
Banco de Dados SI		10						x	x	x	x	x	x	x	x	x
EPG <<interface>>		10						x	x	x	x	x	x	x	x	x
Pacote Interface com o Usuário																
Idioma						Idioma		x	x	x	x	x	x	x	x	x
Inglês		40				Idioma	Inglês				x	x	x	x	x	x
Português		10				Idioma	Português	x	x	x	x	x	x			
Espanhol		10				Idioma	Espanhol	x	x	x	x	x	x	x	x	x
Alemão		50				Idioma	Alemão				x	x	x			
Francês		50				Idioma	Francês				x	x	x			
Language <<interface>>		10						x	x	x	x	x	x	x	x	x
Controle Remoto		10						x	x	x	x	x	x	x	x	x
Accessed_Program <<interface>>		30						x	x	x	x	x	x	x	x	x
Interface com o Usuário		10						x	x	x	x	x	x	x	x	x
UI <<Interface>>		10						x	x	x	x	x	x	x	x	x
Perfil do Telespectador		30						x	x	x	x	x	x	x	x	x
Profile <<interface>>		30						x	x	x	x	x	x	x	x	x
Pac. Sistema de Navegação de TV						Segmento		x	x	x	x	x	x	x	x	x
Pacote Classe C		10				Segmento	Classe C	x	x	x	x	x	x	x	x	x
Informações do canal atual		10				Segmento	Classe C	x	x	x	x	x	x	x	x	x
Bloqueio de Programas		10				Segmento	Classe C	x	x	x	x	x	x	x	x	x
Block <<interface>>		10				Segmento	Classe C	x	x	x	x	x	x	x	x	x
Pacote Classe B		20				Segmento	Classe B		x	x		x	x		x	x
Lembrete		20				Segmento	Classe B		x	x		x	x		x	x
Reserve <<interface>>		20				Segmento	Classe B		x	x		x	x		x	x
Busca de programas		20				Segmento	Classe B		x	x		x	x		x	x
Navegação no EPG		20				Segmento	Classe B		x	x		x	x		x	x
Pacote Classe A		30				Segmento	Classe A			x			x			x
Recomendação		30				Segmento	Classe A			x			x			x
Gravador (PVR)		30				Segmento	Classe A			x			x			x
Record <<interface>>		30				Segmento	Classe A			x			x			x
Integração																
Integração AL_C		10						x								
Integração AL_B		20							x							
Integração AL_A		30								x						
Integração AN_B		40													x	
Integração UE_A		50											x			
Integração AN_A		60														x
Integração UE_B		70										x				
Integração AN_C		80												x		
Integração UE_C		90									x					
Teste																
Integração AL_C		10						x								
Integração AL_B		20							x							
Integração AL_A		30								x						
Integração AN_B		40													x	
Integração UE_A		50											x			
Integração AN_A		60														x
Integração UE_B		70										x				
Integração AN_C		80												x		
Integração UE_C		90									x					

*** Homem-hora**

Figura 4.6. SPL backlog do sistema de navegação de TV
Fonte: do próprio autor

4.6 Conclusão

Neste capítulo foi apresentada a aplicação dos sub-processos da fase pré-jogo, do processo ScrumPL, ao desenvolvimento de uma SPL de um sistema de navegação de TV. A partir das atividades destes processos, foram definidos os artefatos: *roadmap* de produtos, requisitos da SPL e das aplicações, arquitetura de referência, arquiteturas específicas de três aplicações e o SPL *backlog* do sistema de navegação de TV.

Com esses artefatos, o *scrum team* pode dar início à fase de desenvolvimento, fazendo a estimativa dos SPL *backlog items* com maior prioridade, neste caso os que correspondem à aplicação AL_C – voltada para o seguimento de mercado classe C dos oito países da América Latina que adotaram o padrão de TV digital ISDB-T. Se necessário esses SPL *backlog items* serão divididos em partes suficientes para que se possa iniciar o primeiro *sprint*, ou seja até que tenham um esforço estimado que caiba dentro de um *sprint*.

Observa-se que os requisitos estão definidos em alto nível, e que as interfaces não foram descritas em termos das operações que fornecem. Isso ocorre em razão dos princípios ágeis de que se deve dar mais valor a indivíduos e interações sobre processos e ferramentas, e a software funcionando sobre documentação abrangente. Isso que quer dizer que o *Scrum team* entrará em contato com o *product owner* para obter mais detalhes a respeito dos requisitos de cada componente e de suas interfaces. Esse contato já se dará na reunião *sprint planning*, na qual o *product owner* deve participar, de acordo com o método *Scrum*, e, conseqüentemente, de acordo com o processo ScrumPL.

Capítulo 5- Análise Crítica do Uso do Processo ScrumPL

Neste capítulo serão discutidos cada um dos sub-processos do ScrumPL, que foram usados para o desenvolvimento do sistema de navegação de TV descrito no capítulo 4. As discussões dizem respeito à execução das atividades determinadas pelo processo ScrumPL. Todos os sub-processos da SPLE tratados aqui são descritos em Pohl et al (2005).

O primeiro sub-processo utilizado para o desenvolvimento da SPL é o de gerenciamento de produtos, cujo principal resultado é o *roadmap* de produtos. O ScrumPL determina que esse sub-processo advém da SPLE. Esta, por sua vez, determina que o sub-processo gerenciamento de produtos é responsável por controlar o desenvolvimento, produção e divulgação da SPL e de suas aplicações. De acordo com a SPLE, o *roadmap* de produtos define as principais características comuns e variáveis de todas as aplicações da SPL, ou seja, seu escopo, bem como o cronograma de entrega de cada produto aos clientes. A SPLE fornece como técnicas para gerenciamento de produtos: portfólio de produtos, métodos de definição de escopo, método de Kano, entre outros.

Com o mesmo objetivo de definir as principais características comuns e variáveis da SPL do sistema de navegação de TV, e, conseqüentemente o *roadmap* de produtos, a técnica utilizada na execução deste processo foi a abordagem descrita por Pietro-Diaz e Arango (1991), em conjunto com o método descrito por Keshav (2007).

Estas técnicas foram usadas em razão dos requisitos desta SPL advirem da literatura científica, ao invés de advirem das necessidades do mercado identificadas por um departamento de marketing. Assim, as atividades realizadas tiveram como resultado: um conjunto completo de requisitos para o sistema de navegação de TV, a definição do que é esse sistema, potenciais mercados para introduzir as aplicações da SPL, pontos de variação e variantes. A partir deste conjunto completo de requisitos, foi definido o *roadmap* de produtos, os requisitos da SPL e das aplicações, e o diagrama de variabilidade. Em suma, estas técnicas cobriram os objetivos tanto do sub-processo de gerência de produtos quanto os de engenharia de requisitos de domínio e engenharia de requisitos de aplicação.

Com base nessa discussão, conclui-se que diversas técnicas podem ser usadas para estabelecer o *roadmap* de produtos, os requisitos, os pontos comuns e variáveis da SPL. O que determina qual técnica será usada é o contexto onde está inserida a organização responsável pela SPL. De qualquer forma, deve-se definir o modelo de variabilidade, associando suas variantes aos requisitos da SPL, através do sub-processo de engenharia de requisitos de domínio. Esse modelo é muito importante porque é através dele que se determinam os requisitos específicos de cada aplicação. As aplicações definidas no *roadmap de produtos* foram inseridas no SPL *backlog.*

O ScrumPL determina que a estratégia da criação da SPL – definida por Nyholm (2002) – é definida durante o projeto de domínio, juntamente com a arquitetura de referência. No caso da SPL do sistema de navegação de TV, essa estratégia já pôde ser definida na etapa de planejamento do processo ScrumPL, em razão desta ser uma nova SPL. Assim, a estratégia adotada já durante a etapa de planejamento de projeto foi: “evoluir uma nova SPL”. Desta forma, todos os requisitos comuns e variáveis da SPL acabaram por fazer parte da plataforma, não tendo sido definidos requisitos específicos das aplicações.

O *product owner*, por ser o responsável pelos produtos, já participa do processo desde a etapa de planejamento, durante a execução do sub-processo de gerenciamento de produtos. Sendo assim, nome deste papel pode ser mais bem representado por SPL *owner*, indicando o mesmo como o responsável pela SPL inteira. Dependendo do tamanho da SPL pode haver *SPL owners* e *product owners*, mantendo a responsabilidade do primeiro e determinando o segundo como responsável por uma aplicação.

Os sub-processos de gerenciamento de produtos, engenharia de requisitos de domínio e engenharia de requisitos de aplicação fazem parte da etapa de planejamento da fase pré-jogo do ScrumPL. Pohl et. al (2005) relatam que no sub-processo de engenharia de requisitos de domínio são definidos requisitos para todas as aplicações possíveis dentro do domínio da SPL, mas, no ScrumPL não se busca extrair todos os requisitos possíveis da SPL ainda nesta etapa, de modo a postergar a etapa de projeto, pois, visto que o ScrumPL, assim como os métodos ágeis, abraça as mudanças, quaisquer requisitos podem ser acrescentados à SPL, em qualquer fase. A fase pré-jogo pode ser feita através de um ou mais *sprints*, tendo como resultado o SPL *backlog* com detalhes suficientes para iniciar o primeiro *sprint* da fase de desenvolvimento.

Considerando a etapa de projeto do ScrumPL, o sub-processo de projeto de domínio determinou, a partir dos requisitos de domínio e do diagrama de variabilidade, a arquitetura de referência do sistema de navegação de TV. Esta foi composta pelos diagramas de variabilidade e de componentes, tendo sido definidos os pacotes, componentes e interfaces que realizam cada ponto de variação ou variante, através de uma associação entre esses artefatos. Uma melhoria que pode ser feita diz respeito à determinação de um identificador para cada requisito, devendo cada componente estar ligado a pelo menos um requisito através de uma associação ao indicador, fornecendo, assim, rastreabilidade entre componentes requisitos, e, mais adiante, aos artefatos de implementação. No sistema de navegação de TV esta associação foi feita através do nome do componente e do requisito. Mas numa SPL com um número maior de requisitos, a abordagem usada não seria suficiente.

O sub-processo de projeto de aplicação foi utilizado para determinar as arquiteturas específicas das aplicações, a partir da seleção das variantes que compõem cada produto. Em conseqüência da seleção das variantes, os componentes associados a elas foram selecionados para compor a arquitetura da aplicação. Desta forma, o diagrama de variabilidade é um elemento chave da SPL, e é através dele que são geradas as arquiteturas específicas das aplicações.

Por fim, os pacotes, componentes, interfaces, da arquitetura de referência, foram acrescentados ao SPL *backlog* como SPL *backlog items*. Os pontos de variação e variantes foram acrescentados para facilitar o controle das variantes sendo realizadas por cada SPL *backlog item*. Apesar deste artefato ter uma coluna para que seja escrita a estória, esta não foi utilizada. Isso ocorreu porque já haviam sido determinados os requisitos, que estão associados aos SPL *backlog items* através dos componentes e interfaces, que por sua vez, estão associados aos requisitos.

A partir deste ponto, se inicia a fase de desenvolvimento e o *scrum team* começa a fazer parte do processo ScrumPL, estimando o esforço para realizar os SPL *backlog items* e iniciando o primeiro *sprint*. Um ponto que deve ser destacado é o fato das interfaces não terem sido especificadas em termos das operações fornecidas. Isso fica a cargo do *Scrum team*, que deverá prover a especificação destas interfaces através do código fonte, usando, por exemplo, estruturas (*structs*), caso a linguagem de programação seja C. Para lembrá-los disso, poderia ter sido acrescentado um SPL *backlog* solicitando descrição das interfaces.

Outro ponto a ser tratado é a presença de mais de um *Scrum team*, através do Scrum escalável (Schwaber e Beedle, 2002), mantendo um só SPL *backlog*. Neste caso, um *Scrum team* escolhe um conjunto de SPL *backlog items* para realizar e outro escolhe um conjunto diferente, de forma a agilizar ainda mais o desenvolvimento dos componentes. Desta forma, torna-se crucial o compartilhamento das descrições das interfaces, devendo as atividades de descrever estas interfaces terem uma prioridade maior.

Também deve ser levada em consideração a ocorrência de variabilidades internas. Um exemplo é o componente “Interface com o Usuário”, que deverá estar preparado para funcionar em telas de TV de diversos tamanhos, mas esse tipo de variabilidade não foi tratada na arquitetura de referência. Assim, o *Scrum team* deverá estar atento para encontrar variabilidades e entrar em acordo com o *product owner* sobre como serão tratadas estas variabilidades.

Pohl et al (2005) descreve uma estratégia adotada por algumas empresas que usam a SPLE, que é definir uma equipe só para os artefatos de domínio e outras equipes para os artefatos de aplicação. Este tipo de prática, apesar de atender bem ao princípio de que ao especializar uma equipe, a qualidade dos produtos com que trabalham aumenta, vai contra o princípio do Scrum de equipes multifuncionais, onde todos podem fazer qualquer trabalho necessário para atingir os objetivos do *sprint*, evitando gargalos, e, conseqüentemente, aumentando a velocidade do time.

Capítulo 6- Considerações Finais

O objetivo deste trabalho foi desenvolver um processo a partir da integração da SPLE com o método ágil *Scrum* e seu ciclo de vida, utilizando o processo resultante em um estudo de caso. Para isso foi feito um estudo sobre os pontos de convergência e divergência destas duas metodologias. Também foi feita uma análise das entradas e saídas exigidas por cada etapa desses dois processos, tendo sido identificados pontos em que os artefatos de saída e de entrada são convergentes.

A partir dessas análises, concluiu-se que o ciclo de vida *Scrum* seria o alicerce para a integração entre os dois métodos, visto que fornece fases bem definidas de desenvolvimento de software, desde a concepção até a entrega. Com base nos objetivos de cada fase, foram introduzidos os sub-processos e artefatos da SPLE, bem como atividades e artefatos do método *Scrum* que melhor se adequaram aos objetivos de cada fase.

Desta forma, da fase pré-jogo fazem parte os sub-processos de engenharia de domínio, projeto de domínio engenharia de aplicação e projeto de aplicação. Sendo que nesta fase são elaborados o *roadmap* de produtos, a arquitetura de referência da SPL, as arquiteturas específicas de cada aplicação e um *product backlog*, aqui denominado SPL *backlog*, contendo componentes, interfaces e outros artefatos tratados como *product backlog items*, aqui denominados, SPL *backlog items*.

Na fase desenvolvimento, foi definido o desenvolvimento dos SPL *backlog items* através dos *sprints*, sejam esses itens pertencentes ao domínio ou à aplicação. Ao final de cada *sprint*, o *Scrum team* entrega componentes potencialmente prontos para uso, bem como especificações de teste para uso em futuros testes de regressão. Na fase de entrega, foi definido que os componentes fossem integrados, através dos *sprints*, para formar as aplicações, fazer os testes de integração e corrigir eventuais *bugs*.

Após a definição do processo, denominado ScrumPL, o mesmo foi utilizado para desenvolver uma linha de produtos de software de um sistema de navegação de TV, para uso em todas as Américas e na Europa. Para o desenvolvimento desta linha de produto de software, foram realizadas todas as etapas da fase pré-jogo, tendo sido elaborados: o *roadmap* de produtos, um conjunto de requisitos, o diagrama de variabilidade, a arquitetura

de referência, as arquiteturas das aplicações e, por fim, o SPL *backlog*, pronto para o início da fase de desenvolvimento pelos *Scrum team*s.

Também foi levado em consideração o uso dos princípios ágeis na durante a fase de pré-jogo, apesar de suas atividades advirem, em sua maioria, da engenharia de linha de produto de software. Os princípios utilizados são os de que deve se dar mais valor a: indivíduos e interações que a processos e ferramentas e software funcionando que a documentação compreensiva. Como resultado desses princípios, os requisitos foram definidos apenas em alto nível e a fase de pré-jogo foi concluída assim que se houveram SPL *backlog items* suficientes para iniciar o primeiro *sprint*.

Desta forma, serão feitas atualizações dos requisitos da SPL, através do SPL *backlog* durante o desenvolvimento, se necessário, o que segue o princípio de que deve se dar valor à resposta a mudanças a seguir um plano. Isso abreviou as tarefas de planejamento e projeto da SPL.

Este trabalho teve como resultados um processo de desenvolvimento de SPLs, denominado ScrumPL, e uma linha de produto de software de um sistema de navegação de TV pronto para ser desenvolvido por um *Scrum team*, através do processo ScrumPL. Esse processo é um método ágil de desenvolvimento de SPL, cujas potenciais vantagens são:

- Flexibilidade na mudança de requisitos e na execução das tarefas;
- Realização tanto artefatos de domínio quanto de aplicação por *Scrum teams*;
- O desenvolvimento de componentes inicia assim que houverem SPL backlog items suficientes para realização dentro de um *sprint*;
- Desenvolvimento iterativo e incremental de aplicações da SPL
- Melhor custo, prazo e qualidade para SPLs com mais de três produtos, comparado ao desenvolvimento sem SPL.

A SPL do sistema de navegação de TV aqui especificada tem como potenciais vantagens menor tempo, e custo de desenvolvimento e melhor qualidade comparada ao desenvolvimento de suas aplicações sem a SPL, além de atingir a um público muito grande (Américas e Europa).

Por fim, nesse momento, não é possível determinar se o ScrumPL reduz ou não o tempo de desenvolvimento dos primeiros produtos da SPL, visto que não foi avaliado em todas as suas etapas. Mas, deve-se lembrar de que o objetivo deste trabalho é a criação de um processo que possa integrar o método ágil Scrum à SPLE. Portanto, e em razão de ter sido especificada uma SPL a partir do uso do processo ScrumPL. Tendo esta SPL pontos

comuns e variáveis, que resultaram num *product backlog*, denominado SPL *backlog*. Sendo que este artefato está pronto para ser usado por um *scrum team* para dar início ao desenvolvimento de componentes. Conclui-se que esse objetivo de integrar a SPLE ao método Scrum foi alcançado através do ScrumPL.

6.1 Sugestões de Trabalhos Futuros

Como trabalhos futuros, sugere-se implementar esta linha de produtos de software para avaliar o uso do ScrumPL nas fases de desenvolvimento e entrega. Para isso será necessário um *Scrum team* que tenha habilidades de desenvolvimento de software embarcado para TV digital. Esse *Scrum team* será treinado no uso do processo ScrumPL e, tendo como primeira atividade fazer a estimativa dos SPL *backlog items* de maior prioridade, seguido da *sprint planning meeting* do primeiro *sprint*. Durante a fase de desenvolvimento sugere-se buscar novos requisitos para incorporar à SPL, especialmente requisitos específicos de cada região, para avaliar o comportamento do ScrumPL em face a uma mudança.

Também se sugere que seja construída uma ferramenta que permita: definir o *roadmap* de produtos; elaborar os requisitos e o modelo de variabilidade; definir a arquitetura de referência com seus componentes e interfaces; derivar a arquitetura das aplicações a partir da arquitetura de referência, permitindo a adição de artefatos específicos da aplicação; gerar o SPL *backlog*, permitindo ao *Scrum team* estimar o esforço para cada SPL *backlog item*; definir o sprint *backlog*; gerar o *sprint burndown chart* e o SPL *backlog burndown chart*. Uma ferramenta com esses requisitos tornará mais fácil aos envolvidos no uso do processo ScrumPL fazer o planejamento e controle do desenvolvimento das SPLs.

6.2 Dificuldades Encontradas

Nesta seção serão apresentadas as cinco principais dificuldades encontradas durante todo este trabalho, bem como será mostrado o primeiro desafio para o desenvolvimento da SPL do sistema de navegação de TV.

No início dos trabalhos, a maior dificuldade foi entender o funcionamento de uma linha de produto de software, visto que o material em uso, na época, foram artigos da área, que proviam opiniões e visões de alto nível, tendências nesta área e resultados alcançados,

ou seja, faltava uma base de conhecimento mais aprofundada. A partir do momento em que foi obtida a referência Pohl et al (2005), o entendimento do funcionamento de uma SPL tornou-se mais fácil, visto que o material é didático – pois é voltado para cursos sobre fundamentos de SPL – e contém não só o *framework* utilizado e descrito pelos autores, mas também discussões a respeito de diversos métodos, ferramentas e processos usados. Com esta literatura, entendeu-se que o ponto principal para o desenvolvimento uma SPL é encontrar, dentro de um domínio, os pontos de variação e suas variantes. Entendidos os fundamentos de SPL, ficou mais fácil entender os artigos já lidos, bastando revisá-los para aplicar suas técnicas.

A segunda dificuldade foi a integração entre a SPLE e o método *Scrum*, a princípio tentou-se juntá-los fazendo-se com que a SPLE assumisse responsabilidade tática, definindo os *product backlog items* que os *scrum teams* deveriam realizar, mas apenas considerando requisitos. Mas esses requisitos não eram suficientes, visto que os principais artefatos a serem reutilizados são os componentes. Não houve sucesso nesta tentativa, mas ela mostrou um caminho a não ser seguido, bem como mostrou que o primeiro passo era voltar à identificação de pontos de variação e variantes dentro de requisitos de domínio.

Assim, partiu-se para um estudo do domínio de guia eletrônico de programação (EPG) para encontrar seus requisitos, pontos de variação e variabilidades, a partir do estudo do estado da arte na área. A dificuldade (a terceira) então foi organizar os artigos na área para encontrar os requisitos adequados, o que foi superado ao utilizar a técnica de avaliação de artigos de Keshav (2007), juntamente com a técnica de análise de domínio de Pietro-Díaz e Arango (1999). Com isso, descobriu-se que o domínio a ser desenvolvido não era o de EPG, mas sim o de um sistema de navegação de TV, tendo sido definidos seus requisitos, pontos de variação e variantes.

Neste momento sabia-se que estes requisitos deveriam ser representados através do *product backlog* do método *Scrum*, mas não como requisitos, mas sim como componentes e interfaces, visto que representam as partes de um produto de software realizadas em um ou mais *sprints*. Daí surgiu a quarta dificuldade: como definir esses componentes a partir de requisitos de uma SPL, considerando, principalmente que devem ser reutilizáveis? A resposta estava novamente nos fundamentos da SPLE, pois o sub-processo projeto de domínio (Pohl et al, 2005) mostra como representar requisitos e variabilidades através de diagramas de componentes e outros da UML, formando a arquitetura de referência, e, conseqüentemente, a arquitetura das aplicações.

A partir deste ponto, sabia-se que antes do primeiro *sprint* começar era necessário que os requisitos e as arquiteturas de referência e das aplicações estivessem definidas e que os componentes destas arquiteturas e suas interfaces deveriam compor o *product backlog* como *product backlog items*. Portanto a elaboração do *product backlog* representava o fim de uma etapa importante. Sendo assim, a quinta última dificuldade foi encontrar uma forma de representar as etapas do processo, que foi resolvida através de Larman (2004), que não só mostra os fundamentos dos métodos ágeis e o funcionamento de XP, Scrum, EVO e UP, mas também definiu muito bem e de forma simples o "ciclo de vida *Scrum*", que foi a última "peça" que faltava para compor o ScrumPL.

Definido o processo ScrumPL, o mesmo foi usado para definir a SPL do sistema de navegação de TV descrito no capítulo 4, faltando agora um *scrum team* preparado para cumprir com a fase de desenvolvimento. Como o pré-requisito deste *scrum team* é saber desenvolver software embarcado para TV digital, o próximo desafio será formar um time que tenha esta habilidade.

Referências Bibliográficas

ABEP (2010). Critério de classificação econômica Brasil.

ABNT (2008) NBR15604:2008. “Televisão digital terrestre – receptores”.

ABNT (2001) “NBR ISO 9001 sistemas de gestão da qualidade – requisitos”. ABNT CE-25:002.18 – Comissão de Estudos de Sistemas da Qualidade, Brasil.

ATSC (1997) Advanced Television Systems Committee (ATSC) A/65. Program and system information protocol for terrestrial broadcast and cable (PSIP).

ABNT (2007) NBR15603-2. Televisão digital terrestre – multiplexação e serviços de informação (SI). Parte 2: estrutura de dados e definições da informação básica de SI”.

BASIC, R., MOCINIC, M. (2002). User's requirements for electronic program guide (EPG) in interactive television (iTV). In Video/Image Processing and Multimedia Communications 4th EURASIP-IEEE Region 8 International Symposium on VIPromCom (16-19 June 2002). DOI=10.1109/VIPROM.2002.1026702, PP 457-462

BATISTA, L., CASTRO, A., BULE, M. (2009). Estratégia de desenvolvimento de produtos para a classe C: um estudo de caso da Multibrás S/A. In Revista6 Cadernos de Administração, ano 2, vol. 1, nº 03. Jan-Jun/2009, 10 páginas.

BECK, K. et al. (2001). Agile manifesto. Disponível em http://agilemanifesto.org/. Acessado em 01/08/2010.

BECK, K. (1999). Extreme programming explained: embrace change. Addison-Wesley.

CHRISSIS, M., KONRAD, M. e SHRUM, S. (2003). CMMI(R): guidelines for process integration and product improvement. 2nd edition. The SEI series in software engineering. Addison Wesley press, United States.

CARBON, R., LINDVALL, M., MUTHIG, D. e Costa, P. (2006). Integrating product line engineering and agile methods: flexible design up-front vs. incremental design. In APLE – 1st International Workshop on Agile Product Line Engineering. SPLC’06.

COOPER, K. e FRANCH, X. (2006). APLE – 1st International Workshop on Agile Product Line Engineering. In 10th International Software product line Conference (SPLC’06)

DSDM (2010). DSDM Consortium. Disponível em http://www.dsdm.org. Acessado em 08/08/2010.

DTV Status (2010). http://en.dtvstatus.net. Acessado em 09/08/2010.

DVB report http://www.dvb.org/documents/white-papers/dvb_ovum_report.pdf. Acessado em 09/08/2010.

EHRMANTRAUT, M., HÄRDER, T., Wittig, H., Steinmetz, R. (1996) The personal electronic program guide—towards the pre-selection of individual TV programs. In Proceedings of the fifth International Conference on Information and Knowledge Management – CIKM ’96. 243-250.

FERNANDES, J., LEMOS, G., SILVEIRA, G. Introdução à televisão digital interativa: arquitetura, protocolos, padrões e práticas. In: Jornada de Atualização em Informática do Congresso da Sociedade Brasileira de Computação, JAI-SBC, Agosto de 2004. pp. 1-56.

GHANAM, Y. e MAURER, F. (2008). An iterative model for agile product line engineering. In 12th International Software Product Line Conference – SPLC 2008. pp 377-384.

HAN, J. HAN, I., PARK, K. (2008) User-Configurable personalized mosaic electronic program guide. In IEEE Transactions on Consumer Electronics. Vol. 54, Issue 1, Feb, 2008.192–196. DOI= 10.1109/TCE.2008.4470043.

HANSSEN, G. e FAEGRI, T. (2008). Process fusion: an industrial case study on agile software product line engineering. In The Journal of Systems and Software 81. pp 843-854.

JENSEN, J. (2005) Interactive television: new genres, new format, new content. In Proc. of the second Australasian Conference on Interactive Entertainment, pp 89-96.

KESHAV, S. (2007). How to read a paper. In ACM SIGCOMM Computer Communication Review, Vol. 33, number 3, July 2007.

LARMAN, C. (2004). Agile and iteractive development: a manager´s guide. Pearson education Inc., United States.

ISOBE, T., FUJIWARA, M., KANETA, H., URATANI, N., Morita, T. 2003. Development and features of a TV navigation system. In IEEE Transactions on Consumer Electronics.Vol. 49, issue 4, Nov, 2003. 1035–1042. DOI = 10.1109/TCE.2003.1261192.

McMICHAEL, B. e LOMBARDI, M. (2007). ISO 9001 and agile development. In AGILE 2007 – Agile 2007 conference.

MOHAN, K., HAMESH, B., SUGUMARAN, V. (2010). Integrating software product line engineering and agile development. In IEEE Software. May/June 2010. pp 48-55.

MORRIS, S. e SMITH-CHAIGNEAU, A. (2005). Interactive TV standards – a guide to MHP, OCAP and JavaTV. Elsevier – Focal press. United Kingdom.

NORTHROP, L., CLEMENTS, P. (2007). A framework for software product line practice, Version 5.0”. Disponível em http://www.sei.cmu.edu/*product*lines/framework.html.

NORTHROP, L. (2002). SEI’s software product line tenets. In IEEE Software, July/August 2002.

NOOR, M. A., RABISER, R., and GRÜNBACHER, P. (2008). Agile product line planning: a collaborative approach and a case study. Journal of Systems and Software 81, 6 (Jun. 2008). DOI= http://dx.doi.org/10.1016/j.jss.2007.10.028, pp 868-882

NYHOLM, C. (2002). Product line development – an overview. In Building Reliable Component-Based Systems – Extended Report. Artech house, pp 44-57.

PENG, C., LUGMAYR, A., VUORIMAA, P. (2002). A digital television navigator. In Multimedia Tools and Applications. Volume 17, number 1. May, 2002. DOI= 10.1023/A:1014687823960, pp 121-141

PIETRO-DÍAZ, R. e ARANGO, G. Domain analysis concepts and research directions. In Domain Analysis and Software Systems Modeling. IEEE Computer Society Press, 1991, pp. 9-26.

PALMER, S. (2003). A practical guide to feature-driven development. Prentice Hall.

POHL, K., BÖCKE, G., LINDER, F., (2005). Software product line Engineering – Foundations, Principles and Techniques. Springer-Verlag Berlin hedelberg. Germany.

SANTOS Jr, A., SILVA, V. e LUCENA Jr, V. (2008). Desenvolvimento de um processo de software aderente à ISO 9001-2000 baseado no processo ágil Scrum. In SBQS 2008 – VII Simpósio Brasileiro de Qualidade de Software.

SANTOS Jr, A. e SANTOS, R. (2009). Aspectos sociotécnicos do desenvolvimento de software utilizando Scrum em um caso prático. In WOSES 2009 – V Workshop um Olhar Sociotécnico sobre a Engenharia de Software.

SILVA, E. e LOVISOLO, L. (2008). TV digital. DEL/Poli/UFRJ 2008.

SCHWABER, K. e BEEDLE, M. (2002). Agile software development with Scrum, Prentice Hall. United States.

SCHWABER, K. (2004). Agile project management with Scrum. Microsoft press. United States.

SCRUMMI (2009). Um processo de gerenciamento ágil projetos baseado no Scrum e aderente ao CMMI. http://www.scrummi.com.br/, Julho, 2009. Acessado em 08/08/2010.

SUTHERLAND, J. (2009). Fully distributed Scrum: Replicating local productivity and quality with offshore teams. In proceedings of the 42nd Hawaii International Conference on System Sciences – 2009.

SZYPERSKI, C. (1998). Component Software – Beyond Object-Oriented Programming. ACM press and Addison-Wesley. England.

TIAN, K., COOPER, K. (2006). Agile and Software product line Methods: Are they so Diferent?. . In APLE – 1st International Workshop on Agile Product Line Engineering. SPLC'06.

VRIENS, C. (2003) Certifying for CMM Level 2 and ISO9001 with XP@Scrum. In Proceedings of the Agile Development Conference (ADC'03).

WEIß, D., SCHEUERER, J., WENLEDER, M., ERK, A., GÜLBAHAR, M., LINNHOFF-POPIEN, C. (2008) A user profile-based personalization system for digital multimedia content. In Proc. of the 3rd Int. Conference on Digital Interactive Media in Entertainment and Arts – DIMEA ’08. 281-288.

ZHANG, H., ZHENG, Z., YUAN, J. (2005). A personalized TV guide system compliant with MHP. In IEEE Transactions on Consumer Electronics. Volume 51, issue 2, May 2005. 731-737. DOI= 10.1109/TCE.2005.1468026

ZHANG, Z., Li, J., PAN, L., NI, H. (2006) Implementation of an advanced TV navigation system. In Proc. of the 8th Int. Conference on Signal Processing. ICSP 2006. Vol. 4. 16-20 Nov. 2006. DOI = 10.1109/ICOSP.2006.346097.

Apêndice A- Publicações

SANTOS JR, A. e LUCENA JR, V. (2010a). A Software product line requirements and reference architecture for TV navigation system. In Adjunct Proceedings of the 8th European Conference on Interactive TV and Video – EuroITV 2010, pp. 178-181.

SANTOS JR, A. e LUCENA JR, V. (2010b). ScrumPL – Software product line with Scrum. In Proceedings of the 5th International Conference on Evaluation of Novel Aproaches to Software Engineering – ENASE 2010, 6 páginas.

Apêndice B- Termos de SPL e de Scrum

Aplicação	Software específico advindo da combinação de artefatos da SPL.
Arquitetura de referência	Arquitetura contendo um diagrama de variabilidade ligado a diversos artefatos.
Arquitetura específica	Arquitetura de uma aplicação da SPL contendo componentes da arquitetura de referência voltados a uma aplicação.
Daily meeting	Reunião de sincronização de tarefas realizada todos os dias, com 15 minutos de duração.
Diagrama de variabilidade	Diagrama contendo pontos de variação e variantes, representando a variabilidade da SPL
Feature-driven development	Método ágil voltado ao desenvolvimento de características de produtos.
Feature model	Modelo da SPL que representa características de produtos e suas variantes.
Ponto de variação	É algo do mundo real que sofre variações.
Product backlog	Ferramenta de gerenciamento dos requisitos do método scrum.
Product backlog burndown chart	Gráfico de linha que indica o progresso do esforço restante para concluir um produto.
Product owner	Responsável pelo produto sendo realizado por um Scrum team.
Retrospective meeting	Reunião de identificação de pontos fortes e fracos no desenvolvimento de software, buscando melhorias.
Review meeting	Reunião de apresentação dos resultados de um sprint.
Roadmap de produtos	Arterfato que define as aplicações de uma SPL, suas variantes e a data de lançamento.
Scrum	Método ágil de desenvolvimento de software com equipes auto-gerenciadas.

Scrum master	Membro do Scrum team responsável por garantir que os princípios e práticas do método Scrum sejam seguidos.
Scrum team	Equipe de 3 a 10 pessoas que desenvolve software de acordo com o método ágil Scrum.
SPL backlog	Product backlog que gerencia todos os produtos de uma SPL.
SPL backlog burndown chart	Gráfico de linha que apresenta o esforço restante de todas as aplicações de uma SPL.
Sprint	Uma iteração com duração em torno de 30 dias.
Sprint backlog	Conjunto de tarefas a serem realizadas por um Scrum team durante um sprint.
Sprint backlog burndown chart	Gráfico de linha que representa o esforço restante para concluir os objetivos de um sprint.
Sprint planning meeting	Reunião de planejamento de um sprint.
Test-driven development	Método ágil onde os requisitos são descritos como casos de teste.
Variante	Variações sofridas por um ponto de variação.

Anexo I- Padrão de TV Digital do Brasil Já Chega a Oito Países

Anexo II- Sky HDTV

Anexo III- Lista de Artigos Consultados