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

THIAGO FERRAZ VIEIRA DA CUNHA

SLESS 2.0: UMA EVOLUÇÃO DA ABORDAGEM DE INTEGRAÇÃO DO SCRUM E LEAN SIX SIGMA PARA APLICAÇÕES MÓVEIS

2014

THIAGO FERRAZ VIEIRA DA CUNHA

SLESS 2.0: UMA EVOLUÇÃO DA ABORDAGEM DE INTEGRAÇÃO DO SCRUM E LEAN SIX SIGMA PARA APLICAÇÕES MÓVEIS

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

Área de Concentração: Ciência da Computação

Orientação: Profa. Dra. Rossana Maria de Castro Andrade.

2014

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

C98s Cunha, Thiago Ferraz Vieira da.
SLeSS 2.0: uma evolução da abordagem de integração do Scrum e Lean Six Sigma para aplicações móveis / Thiago Ferraz Vieira da Cunha. - 2014.
132 p.: il., enc.; 30 cm.

Dissertação (mestrado) - Universidade Federal do Ceará, Centro de Ciências, Departamento de Computação, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2014.
Área de Concentração: Ciência da Computação.
Orientação: Profa. Dra. Rossana Maria de Castro Andrade.

1. *Software* - Desenvolvimento. 2. Gestão da qualidade. 3. *Scrum* (*Computer software development*). 4. Dispositivos móveis. I. Título.

CDD 005

THIAGO FERRAZ VIEIRA DA CUNHA

SLESS 2.0: UMA EVOLUÇÃO DA ABORDAGEM DE INTEGRAÇÃO DO SCRUM E LEAN SIX SIGMA PARA APLICAÇÕES MÓVEIS

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

Aprovada em: __/__/____

BANCA EXAMINADORA

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

Prof. Dr. Francisco Rodrigo Porto Cavalcanti
Universidade Federal do Ceará - UFC

Prof. Dr. Danielo Gonçalves Gomes
Universidade Federal do Ceará - UFC

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

Aos meus pais João Augusto e Sávia, à minha esposa Beatriz e ao meu filho Samuel.

AGRADECIMENTOS

Agradeço a Deus por ter sempre me abençoado e cuidado da minha caminhada. À Jesus Cristo, pois creio que por intermédio Dele e para Ele todas as coisas são criadas.

Agradeço ao estímulo e paciência da minha amada esposa Beatriz.

Aos exemplos de busca do saber dos meus pais João Augusto e Sávia.

Aos exemplos de luta dos meus sogros Tomaz e Izidora.

Ao apoio e estímulo das minhas irmãs Patrícia, Carolina e Victória.

À toda a minha família pois não há palavras para expressar o quanto amo, admiro e me orgulho de vocês.

À minha professora e orientadora, Dra. Rossana, pela partilha do seu saber, orientação precisa e valiosa que direcionaram esse trabalho. Agradeço acima de tudo, por ela sempre estimular o meu interesse pelo conhecimento e pela vida acadêmica.

À Dataprev pelo grande estímulo e o apoio.

Ao Atlântico pelo investimento, companheirismo e crença na minha capacidade profissional.

Ao GREat pelo incentivo desde os primeiros projetos, pelo apoio, investimento e colaboração que me foram dedicados nos projetos em que participei.

A todos os amigos que contribuíram para a concretude deste trabalho, estimulando-me intelectual e emocionalmente.

E ao final, ao meu filho amado, Samuel, que com a sua chegada encheu de novas perspectivas a minha vida, impulsionando-me a lutar com uma energia renovada.

“Fico satisfeito em ter feito uma pequena parte para ajudar a tornar o potencial da criatividade humana em realidade.”

(Jack Kilby)

RESUMO

O desenvolvimento de *software* para dispositivos móveis como *smartphones*, celulares e *tablets* requer conhecimento dos processos e das tecnologias relacionadas às plataformas de *software* e de *hardware* desses dispositivos. A melhoria contínua no desempenho dessas plataformas e a demanda crescente por uma variedade de aplicações impõem uma alta competitividade e, por conseguinte, exigem níveis elevados de produtividade e de qualidade dos projetos de desenvolvimento. Nesse cenário, as metodologias ágeis são utilizadas por se adequarem bem às frequentes mudanças de requisitos e às demandas de prazo desse nicho de mercado. Dentre essas metodologias, o *Scrum* é uma das mais aceitas e utilizadas, contribuindo na melhoria da gestão de projetos, na produtividade do time, na qualidade dos produtos e no aumento da probabilidade de sucesso dos projetos. Por outro lado, há uma crescente adoção de metodologias de gestão da qualidade como o *Lean Six Sigma* por organizações da área de Tecnologia da Informação e Comunicação (TIC) devido aos seus resultados comprovados na melhoria dos processos de desenvolvimento e da qualidade dos serviços e dos produtos desenvolvidos. Metodologias como o *Scrum* e o *Lean Six Sigma* possuem objetivos distintos, entretanto, elas podem contribuir juntas no desenvolvimento de *software* para dispositivos móveis. Na literatura existem trabalhos que propõem a integração de metodologias ágeis e de qualidade de *software*, entretanto, esses trabalhos possuem lacunas no que tange a possibilidade de reuso sistemático das mesmas bem como a avaliação do uso dessas metodologias integradas. Este trabalho propõe então uma evolução de uma abordagem já existente, denominada *SLeSS*, que integra o *Scrum* ao *Lean Six Sigma* e que foi utilizada na customização de *software* para celulares. Essa evolução visa ampliar a abordagem tanto para o desenvolvimento de *software* além da customização quanto para melhorar os mecanismos de integração inicialmente propostos com um foco na avaliação das práticas e princípios do *Scrum* a partir do uso de técnicas do *Lean Six Sigma*. Para avaliar a nova versão, ela é aplicada em sete projetos reais de desenvolvimento e customização de *software* para dispositivos móveis e seus resultados são também discutidos nesta dissertação.

Palavras-chave: Metodologias Ágeis. Metodologias de Gestão da Qualidade. Desenvolvimento e Customização de *Software*. *Scrum*. *Lean Six Sigma*. Dispositivos Móveis.

ABSTRACT

The software development for mobile devices such as mobile phones, smartphones and tablets requires knowledge not only of processes but also related to software and hardware technology platforms of these devices. Continuous improvement in the performance of these platforms and the increasing demand for a variety of applications require a high competitiveness and, therefore, high levels of productivity and quality. In this scenario, agile methodologies are used and fit well to frequently changing requirements and to time to market. Among these methodologies, Scrum is one of the most accepted and used, contributing to the improvement of project management, team productivity, product quality and increasing the probability of project success. On the other hand, there is an increasing adoption of quality management methodologies such as Lean Six Sigma in Information Technology and Communication (ICT) organizations due to their proven results in improving processes development and quality of services and products developed. Methodologies such as Scrum and Lean Six Sigma have different goals, however, they can contribute together in the development of software for mobile devices. In literature, there are studies that propose the integration of agile methodologies and software quality, however, they have gaps regarding the possibility of systematic reuse of this integration, as well as in evaluating the use of these integrated methodologies. Thus, this paper proposes an evolution of an existing approach, called SLeSS, that integrates Scrum to Lean Six Sigma and has been used in the software customization for mobile phones. The evolution of this approach is to add software development in addition to customization and improve integration mechanisms initially proposed with a focus on evaluation of the practices and principles of Scrum from the use of techniques of Lean Six Sigma. This new version is applied in seven actual projects that are related to software development and customization for mobile devices and their results are also discussed in this dissertation.

Keywords: Agile Methodologies. Quality Management Methodologies. Software Development and Customization. Scrum. Lean Six Sigma. Mobile Devices.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

4-DAT	*4-Dimensional Analytical Tool*
AAF	*Agile Adoption Framework*
ASD	*Adaptive Software Development*
CAS	*Checklist* de Avaliação do *Scrum*
CSS	*Cascading Style Sheets*
DC	Departamento de Computação
DM	Dispositivos Móvel
DMADV	*Define, Measure, Analyze, Design, Verify*
DMAIC	*Define, Measure, Analyze, Improve, Control*
EAP	Estrutura Analítica de Projetos
FDD	*Feature Driven Development*
FMEA	*Failure Mode Effects Analysis*
FTA	*Fault Tree Analysis*
GREat	Grupo de Redes de Computadores, Engenharia de *Software* e Sistemas
HME	*Hybrid Method Engineering*
HTML	*HyperText Markup Language*
IMS	Índice de Medição do Nível do *Scrum*
Java ME	*Java Micro Edition*
LPS	Linha de Produto de *Software*
LSS	*Lean Six Sigma*
ME	*Methodology Engineering*
MI	Mecanismo de Integração
MMS	*Multimedia Messaging Service*
MP3	*MPEG-1 or MPEG-2 Audio Layer III*
MPEG	*Moving Picture Experts Group*
P&D	Pesquisa e Desenvolvimento
P&D&I	Pesquisa, Desenvolvimento e Inovação
PMBoK	*Project Management Body of Knowledge*
PO	*Product Owner*
SBES	Simpósio Brasileiro de Engenharia de Software

SIM	*Subscriber Identity Module*
SIPOC	*Suppliers, Inputs, Process, Outputs, Customers*
SLeSS	*Scrum Lean Six Sigma*
SMS	*Short Message Service*
SW	*Software*
RIM	*Research In Motion*
RUP	*Rational Unified Process*
TDD	*Test Driven Development*
TIC	Tecnologia da Informação e Comunicação
UFC	Universidade Federal do Ceará
VM	*Virtual Machine*
WAP	*Wireless Application Protocol*
WEB	*World Wide Web*
XP	*eXtreme Programming*

1 INTRODUÇÃO

Este trabalho propõe uma evolução do *SLeSS*, uma abordagem de integração entre o *Scrum* e o *Lean Six Sigma* para a gestão de projetos e a melhoria de processos de customização de *software* para dispositivos móveis (DMs). A nova versão dessa abordagem, chamada de *SLeSS* 2.0, é proposta para o desenvolvimento de aplicações, além da customização, de *software* para DMs.

O presente capítulo descreve as principais razões que levaram ao desenvolvimento desta dissertação, assim como o seu objetivo. A organização das seções deste capítulo é apresentada na Figura 1.1.

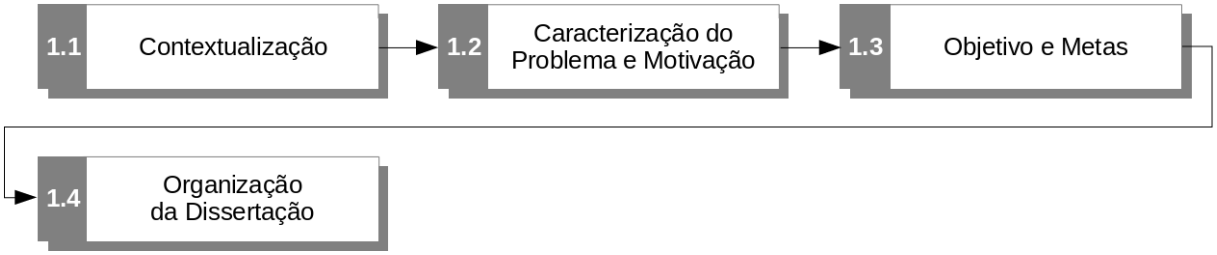


Figura 1.1: Organização das Seções do Capítulo 1

Inicialmente, a Seção 1.1 introduz o tema desta dissertação. A Seção 1.2 trata do uso de metodologias ágeis no desenvolvimento de *software* para DMs e as necessidades da gestão sistemática da qualidade nesse desenvolvimento, e também expõe a motivação para o desenvolvimento deste trabalho. O objetivo e as metas da pesquisa são então descritos na Seção 1.3 e, por fim, a Seção 1.4 apresenta a estrutura organizacional desta dissertação.

1.1 Contextualização

O desenvolvimento de *software* para DMs (i.e., dispositivos portáteis capazes de se comunicar através de alguma tecnologia de comunicação sem fio) tornou-se uma área de grande interesse para a computação devido à crescente demanda por uma variedade de aplicações, consequência da popularização de DMs como celulares, *smartphones* e *tablets*, da grande disponibilidade desses dispositivos no mercado e da praticidade dos mesmos no acesso a conteúdo multimídia e à internet de banda larga (FLING, 2009).

Esses dispositivos permitem, cada vez mais, o acesso à informação de qualquer lugar e a qualquer instante e isso tem influenciado a forma como as pessoas se comunicam, com elas trabalham, se divertem e até mesmo como se relacionam (FONTELES et al., 2013).

Como resultado dessa expansão e da crescente demanda social pela mobilidade no acesso à informação, as organizações na área de Tecnologia da Informação e Comunicação (TIC) vem investindo em novas metodologias e na melhoria de seus processos de desenvolvimento (HOLZER; ONDRUS, 2009).

O desenvolvimento de *software* para DMs precisa lidar com as limitações de recursos dos dispositivos, com a dependência do *hardware* e atender aos requisitos de desempenho, mobilidade, segurança e interfaces (SHEN et al., 2012). Não obstante, as frequentes mudanças de requisitos e ambiente, os prazos agressivos, as exigências por níveis elevados de qualidade e as altas pressões de competitividade são também alguns desafios desse desenvolvimento (GRINE et al., 2005).

Além do desenvolvimento de aplicações para DMs, surge também a necessidade de customizar o *software* embarcado para dispositivos como celulares e *tablets* para atender aos requisitos de operadoras de telefonia, onde nesse caso não são gerados novos produtos e sim várias versões de um mesmo produto. Essa customização de *software* é necessária para adaptar o código base do dispositivo para atender às especificações de interface com o usuário, funcionalidades e configurações solicitadas pelas operadoras e que dependem, por exemplo, da região onde o dispositivo será comercializado (CUNHA et al., 2011).

Nesse cenário, as metodologias ágeis são aceitas e utilizadas por serem apropriadas às demandas de prazos desse nicho de mercado e adequadas às frequentes mudanças inerentes a esse desenvolvimento (CORRAL et al., 2013)(HOLLER, 2006). Além disso, essas metodologias contribuem na melhoria da produtividade das equipes de desenvolvimento e da qualidade dos produtos desenvolvidos (DYBÅ; DINGSØYR, 2008).

Entretanto, a exigência por níveis mais elevados de qualidade tem demandado uma gestão sistemática dessa qualidade, visando a melhoria contínua de processos, serviços e produtos. Sendo assim, a adoção de metodologias de qualidade também tem se tornado comum nessas organizações de TIC, onde são empregadas, principalmente, na melhoria de processos de desenvolvimento, colaborando na realização de objetivos de negócio dessas organizações (PAN et al., 2007).

1.2 Caracterização do Problema e Motivação

Diante dos desafios do desenvolvimento de *software* para DMs mencionados anteriormente, foram propostas metodologias ágeis voltadas especificamente para esse desenvolvimento (ABRAHAMSSON et al., 2004)(RAHIMIAN; RAMSIN, 2008)(JEONG et al., 2008). Entretanto, essas metodologias possuem uma documentação superficial e, em sua maioria, não apresentam resultados em projetos reais, dificultando a análise das mesmas, bem como o seu reúso sistemático.

Por outro lado, há metodologias ágeis como o *Scrum* (SCHWABER, 2007a) e o *eXtreme Programming* (*XP*) (WELLS, 2009) que, embora não tenham sido propostas especificamente para o desenvolvimento de *software* para DMs, são aceitas e utilizadas para tal, apresentando resultados de melhoria de produtividade e qualidade (FLORA et al., 2014).

De acordo com Salo e Abrahamsson (2008), a utilização do *Scrum* e do *XP* tem apresentado resultados de melhoria de produtividade e qualidade ainda que em alguns projetos apenas parte das práticas dessas metodologias sejam realmente utilizadas. Além disso, de

acordo com Qumer (2006), essas metodologias possuem um nível de agilidade relativamente semelhante. O *Scrum* é voltado para a gestão de projetos, foco do trabalho desta dissertação, enquanto o *XP* tem um enfoque maior no desenvolvimento de *software* propriamente dito.

O *Scrum* é uma das metodologias ágeis mais utilizadas no desenvolvimento de *software* para DMs e tem contribuído na melhoria da agilidade da gestão de projetos e no aumento da probabilidade de sucesso desses projetos (SALO; ABRAHAMSSON, 2008). Entretanto, uma avaliação de seu nível de agilidade, realizada através de um método proposto por Qumer (2008), possibilitou identificar que o *Scrum* possui uma limitação quanto ao suporte à produção enxuta no desenvolvimento de *software*. Essa produção enxuta consiste no uso de técnicas e ferramentas de gestão da qualidade para a melhoria sistemática da qualidade dos processos, serviços e produtos.

Embora o *Scrum* possua práticas que incentivem a inspeção e a reflexão sobre o processo utilizado, contribuindo assim na melhoria da qualidade, ele não fornece as técnicas para uma gestão sistemática dessa qualidade, que é necessária para o atendimento de níveis cada vez mais elevados exigidos nesse desenvolvimento. Dessa forma, as metodologias de gestão da qualidade podem ser utilizadas para suprir essa limitação do *Scrum*.

O *Six Sigma* e o *Lean Six Sigma* (*LSS*) são metodologias de gestão da qualidade utilizadas na indústria por seus resultados comprovados. A adoção dessas metodologias vem crescendo em organizações da área de TIC, onde são empregadas, principalmente, na melhoria dos processos de desenvolvimento (PAN et al., 2007).

Existem trabalhos, então, que propõem a combinação de metodologias ágeis e de qualidade. Por exemplo, Hashmi e Baik (2008) propõem um mapeamento de ferramentas do *Six Sigma* às práticas do *XP*, Roriz (2010) sugere uma combinação do *Scrum* ao *Six Sigma*, apresentando um mapeamento entre os papéis dessas metodologias e Cunha *et. al* (2011) propõem uma abordagem de integração entre o *Scrum* e o *LSS*, chamada de *SLeSS*, estabelecendo mecanismos para a integração dessas metodologias. Os trabalhos de Hashmi e Baik (2008) e Roriz (2010) não propõem de fato abordagens, eles apenas sugerem o mapeamento de práticas e papéis para a combinação das metodologias.

Já o *SLeSS* é uma abordagem que foi aplicada em projetos reais de customização de *software* para DMs e obteve resultados de melhoria de produtividade, redução da densidade de defeitos e diminuição de horas extras mensais nos projetos (CUNHA et al., 2011). Essa abordagem objetiva aumentar a agilidade da gestão de projetos, a partir do uso de princípios e práticas do *Scrum*, e melhorar os processos de customização, a partir de técnicas e ferramentas do *LSS*.

No entanto, o *SLeSS* é restrito à customização de *software* para DMs e, embora reutilize princípios e práticas do *Scrum* e do *LSS*, também possui uma documentação incompleta para que seja reutilizado de forma sistemática. Além disso, o *SLeSS* demanda uma capacitação nas técnicas e ferramentas do *LSS* e essa necessidade é uma das principais vulnerabilidades à sua implantação, devido ao custo de treinamentos técnicos e ao tempo necessário para o aprendizado dessa metodologia.

Sendo assim, o *SLeSS* necessita ser adequado ao desenvolvimento de *software* para DMs e, levando em consideração que o reúso sistemático consiste na prática de reúso a partir de um processo bem definido e repetível (ALMEIDA et al., 2007), ele necessita também ser aprimorado para ser reutilizado de forma sistemática nesse desenvolvimento. Além disso, esse aprimoramento deve avaliar como o *Scrum* está sendo efetivamente adotado, uma vez que o *SLeSS* requer que os projetos primeiramente adotem essa metodologia e porque, nessa abordagem, o *Scrum* é utilizado tanto na gestão da customização do *software* quanto na gestão das melhorias de processos de customização.

Para a avaliação da adoção e do uso de metodologias ágeis, há métodos e *checklists* que possibilitam medir a forma de utilização de práticas ágeis e servem como guia de como melhorar a agilidade nos projetos (QUMER et al., 2007)(SIDKY et al., 2007)(VODE; SUTHERLAND, 2008)(KNIBERG, 2009). No entanto, por serem usualmente muito abrangentes, focando em práticas de várias metodologias ao mesmo tempo, esses métodos são extensos.

Sendo assim, para a avaliação do uso do *Scrum*, há a necessidade de métodos e *checklists* mais específicos, que forneçam uma visão dos pontos fortes e fracos no uso dos seus princípios e práticas. Também é necessário que esses métodos auxiliem às equipes na identificação e no tratamento das causas desses problemas. Portanto, o aprimoramento do *SLeSS* pode ser uma alternativa para a avaliação e melhoria do uso do *Scrum* nos projetos, tendo em vista que, ao reutilizar as técnicas e ferramentas do *LSS*, ele possui os meios necessários para possibilitar a evolução do uso dessa metodologia.

1.3 Objetivo e Metas

O presente trabalho tem como objetivo propor uma evolução para a abordagem de integração entre o *Scrum* e o *Lean Six Sigma* (*SLeSS*) (CUNHA et al., 2011), com foco na avaliação e melhoria do uso de princípios e práticas do *Scrum* a partir de técnicas do *Lean Six Sigma* para o desenvolvimento, além da customização, de *software* para dispositivos móveis. Além disso, as melhorias propostas para essa evolução visam também possibilitar o reúso sistemático da abordagem.

Para atingir esse objetivo da pesquisa, foram estabelecidas três principais metas, as quais são descritas a seguir:

(i) *Aprimorar o SLeSS quanto à avaliação e melhoria do uso do Scrum* - o *SLeSS* define mecanismos para a integração do *Scrum* e *LSS* e um deles é relacionado à aplicação de técnicas do *LSS* para a melhoria do uso do *Scrum* nos projetos. Entretanto, o *SLeSS* apresenta uma documentação incompleta desse mecanismo e ainda utiliza o *DMAIC* sem simplificações, o que o torna extenso e de difícil reúso. Sendo assim, é necessário definir um meio mais eficiente para a avaliação dos projetos quanto ao uso de princípios e práticas do *Scrum* e estabelecer uma forma de auxiliar as equipes na identificação de problemas em sua atual implementação e na análise, priorização e solução de causas. Como resultado esperado, o aprimoramento proposto deve possibilitar que as equipes identifiquem e solucionem os principais problemas no uso dessa metodologia;

(ii) *Aprimorar o SLeSS para possibilitar o seu reúso sistemático* - como resultado esperado, o aprimoramento proposto deve definir processos, regras e orientações que forneçam o suporte adequado à reutilização do *SLeSS*; e

(iii) *Adequar o SLeSS ao desenvolvimento, além da customização, de software para dispositivos móveis* - o *SLeSS* foi inicialmente proposto e avaliado para a customização de *software* para DMs. Dessa forma, é necessário adequá-lo ao cenário do desenvolvimento de aplicações para DMs.

Para avaliar a proposta de evolução do *SLeSS*, ela é aplicada em 7 projetos reais de desenvolvimento e customização de *software* para DMs. Esse estudo de caso objetiva avaliar as melhorias propostas (metas citadas anteriormente). Com essa avaliação, é possível detectar limitações a partir da análise dos resultados e do *feedback* das equipes envolvidas, diminuindo os erros na implantação da abordagem.

1.4 Organização da Dissertação

A organização desta dissertação está dividida em 6 capítulos, conforme apresentado na Figura 1.2. O presente capítulo faz uma breve introdução ao tema, contextualizando o assunto abordado neste trabalho e apresentando a motivação, o objetivo, as metas e a própria organização da dissertação.

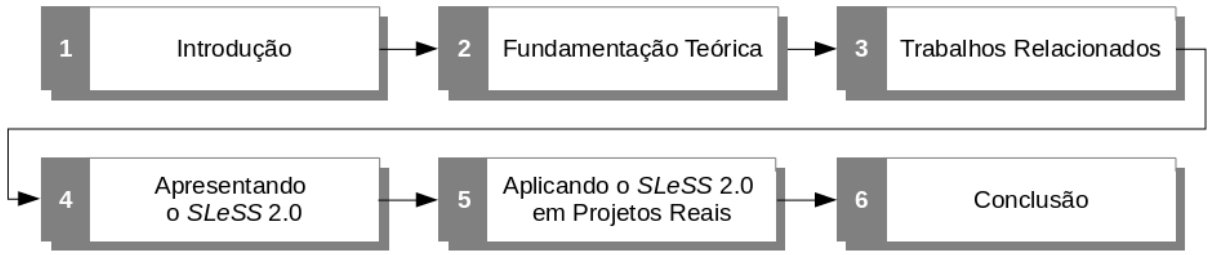


Figura 1.2: Organização dos Capítulos da Dissertação

O Capítulo 2 apresenta os principais conceitos relacionados à temática abordada por esta dissertação. Inicialmente, o desenvolvimento e a customização de *software* para DMs são introduzidos. Em seguida, são descritos os conceitos, valores e princípios das metodologias ágeis e, então, o *Scrum* é apresentado em detalhes. As metodologias de gestão da qualidade como o *Lean*, o *Six Sigma* e o *LSS* são apresentadas na sequência. Esse capítulo finaliza com a descrição de boas práticas na elaboração de *checklists*, pois este trabalho procurou utilizá-las na avaliação do uso de princípios e práticas do *Scrum* nos projetos.

O Capítulo 3 descreve os trabalhos relacionados ao uso de metodologias ágeis e de gestão da qualidade no desenvolvimento e na customização de *software* para DMs. Também são apresentados os trabalhos relacionados à avaliação e melhoria do uso de práticas ágeis em projetos de *software*.

O Capítulo 4 apresenta o *SLeSS* 2.0, iniciando com uma breve descrição da história da concepção dessa abordagem e, em seguida, detalhando a documentação dessa nova versão,

com a sua definição, os papéis, o processo de execução, os mecanismos de integração e um guia de implantação. Ao final, as principais mudanças entre as versões da abordagem são discutidas.

O Capítulo 5 descreve a avaliação deste trabalho de dissertação. O estudo de caso para a avaliação do *SLeSS* 2.0 é apresentado, descrevendo o seu objetivo, contextualizando os projetos selecionados e detalhando a aplicação da nova versão da abordagem em cada um desses projetos. Ao final, os resultados consolidados do estudo são também discutidos.

O Capítulo 6 descreve as conclusões e contribuições deste trabalho, apresentando as possíveis linhas de pesquisa a serem consideradas em trabalhos futuros.

Por fim, o Apêndice A apresenta um resumo da avaliação dos trabalhos relacionados elencados no Capítulo 3.

2 FUNDAMENTAÇÃO TEÓRICA

A adoção de princípios e práticas ágeis no desenvolvimento de *software* para DMs tem se tornado uma resposta à necessidade de se produzir mais rápido, um *software* com melhor qualidade e que satisfaça as necessidades dos usuários (GRINE et al., 2005). A comunidade ágil reconhece as vantagens do uso das metodologias ágeis nesse desenvolvimento e o *Scrum* representa uma alternativa frente às abordagens tradicionais (DYBÅ; DINGSØYR, 2008).

Já a adoção do *Six Sigma* e do *Lean Six Sigma*, metodologias de gestão da qualidade amplamente utilizadas na indústria e com resultados comprovados, vem crescendo em organizações da área de TIC, onde são empregadas, principalmente, na melhoria contínua dos processos de desenvolvimento (PAN et al., 2007).

Neste capítulo são abordados os conceitos fundamentais que serviram de alicerce para o desenvolvimento deste trabalho. A organização das seções deste capítulo é apresentada na Figura 2.1.

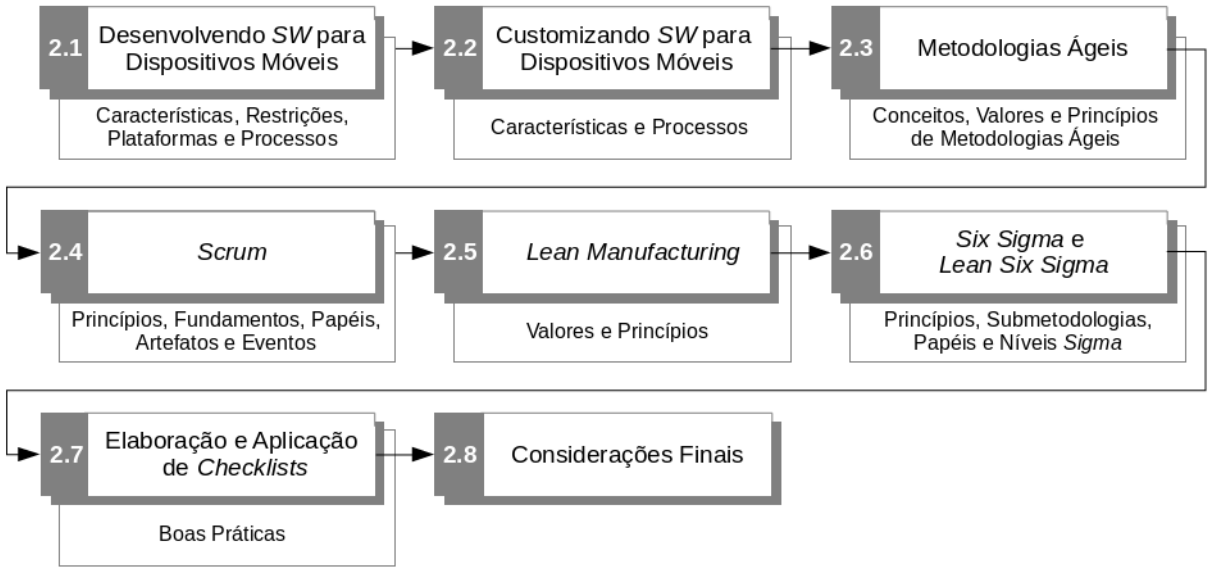


Figura 2.1: Organização das Seções do Capítulo 2

As Seções 2.1 e 2.2 tratam das características e particularidades do desenvolvimento e customização de *software* para DMs. A Seção 2.3 descreve os fundamentos, os princípios e os valores das metodologias ágeis. A Seção 2.4 apresenta os fundamentos, os papéis, os artefatos e os eventos do *Scrum*. O *Lean Manufacturing* é apresentado na Seção 2.5. O *Six Sigma* e *Lean Six Sigma* são apresentados na Seção 2.6. A Seção 2.7 apresenta as principais práticas utilizadas neste trabalho para a elaboração e aplicação de *checklists* e, então, a Seção 2.8 finaliza com as considerações finais sobre este capítulo.

2.1 Desenvolvendo *Software* para Dispositivos Móveis

Com a facilidade do acesso à banda larga, a popularização de DMs como celulares, *smartphones* e *tablets* e a melhoria contínua no desempenho das plataformas de *software* e *hardware* desses dispositivos, a demanda por uma ampla variedade de aplicações móveis mantém o mercado desse seguimento em expansão, impulsionando a indústria de desenvolvimento de *software* a investir na pesquisa por novas metodologias e tecnologias.

O desenvolvimento de *software* para DMs possui particularidades como a alta competitividade, prazos agressivos, difícil identificação dos envolvidos e suas necessidades, frequentes mudanças de escopo e restrições de largura de banda, cobertura, segurança, memória, processamento e armazenamento. Essas particularidades e restrições tornam o desenvolvimento para DMs ainda mais desafiador.

As principais características desse desenvolvimento podem ser descritas a seguir (SHEN et al., 2012):

- *Perfil da equipe* - o desenvolvimento para DMs requer profissionais com um bom nível técnico, com experiência tanto no desenvolvimento de *software* quanto no domínio das aplicações móveis;
- *Dependências do hardware* - tipicamente, o desenvolvimento do *hardware* é realizado paralelamente ao do *software* e alterações nesse *hardware* geralmente afetam o *software*. O progresso do desenvolvimento do *software*, portanto, é quase sempre impactado e limitado por mudanças no *hardware*;
- *Pressões advindas da alta competitividade* - prazos agressivos e restrições de orçamento são impostos pela alta competitividade desse nicho de mercado;
- *Limitações de recursos* - o desenvolvimento precisa lidar com as restrições relacionadas às características dos dispositivos e à necessidade de mobilidade como por exemplo as limitações de capacidade de memória, poder de processamento, interfaces de entrada e saída de dados;
- *Mudanças* - o desenvolvimento para DMs precisa lidar com frequentes mudanças de requisitos e ambiente, como por exemplo, as advindas de alterações nas regras de *marketing*, necessidades de clientes e inovações de concorrentes; e
- *Requisitos de desempenho* - funcionalidades do *software* precisam atender a requisitos de alto desempenho como a necessidade de execução em tempo real e requisitos de segurança.

Os aplicativos para DMs geralmente são desenvolvidos para realizar uma tarefa específica, fazem uso extensivo dos recursos do dispositivo como sensores, reprodução de mídia, gerenciamento de arquivos e acesso à rede e focam em um conjunto de modelos desses dispositivos.

Dentre os vários tipos de aplicativos, há os que disponibilizam serviços como previsões do tempo e navegação de mapas, os que fornecem acesso a conteúdos informativos como guias de compras, promoções, consulta de produtos, os que são utilizados para a comunicação entre pessoas, por exemplo, aplicativos de mensagens instantâneas e chamadas de vídeo e também os focados no entretenimento como os mais variados tipos de jogos.

Atualmente, existe uma ampla variedade de marcas e modelos e as aplicações para esses dispositivos podem ser desenvolvidos a partir de diversas plataformas e linguagens de programação. As principais plataformas do mercado são apresentadas na Tabela 2.1.

Tabela 2.1: Principais Plataformas de Desenvolvimento para Dispositivos Móveis

Plataforma	**Linguagem de Programação**	**Compatibilidade**
Adobe AIR	*Action Script*, *HTML*, *CSS* e *JavaScript*	*iOS* (*iPhone*, *iPad*, *iPod touch*), *Android* e *BlackBerry*
Android	Java, porém, porções de código em C, C++ podem ser inserido	*Android*
BlackBerry	Java	*BlackBerry* apenas, devido a arquitetura *RIM API*
Java ME	Java	Há *VMs* que necessitam de implementações específicas por dispositivo
Ubuntu Touch	*QML*, C, C++, *JavaScript*, *HTML5*, *CSS*	*Ubuntu desktop/Apps* baseados em *WEB* disponível para *browsers* e outras plataformas
Windows Phone	C#, Visual Basic, C, C++	*Windows Phone*

Por exemplo, o *Android* é uma plataforma cuja principal linguagem de programação é o Java e é compatível apenas com dispositivos com o sistema operacional *Android*. Já as aplicações para dispositivos com o *iOS*, como o *iPhone*, podem ser desenvolvidos com a plataforma *Adobe AIR* a partir de linguagens de programação como *Action Script* e *JavaScript*.

Devido a grande variedade de modelos, plataformas, características e recursos disponíveis dos dispositivos, bem como a diversidade de *softwares* distintos em funcionamento no mercado, as empresas de desenvolvimento têm investindo em ferramentas para automação de testes e portabilidade de aplicativos.

Um dos desafios do desenvolvimento de aplicativos para DMs, por exemplo, é a fragmentação que decorre da diversidade de contextos em que esses aplicativos precisam funcionar. Essa diversidade de contextos pode ser entendida a partir dos seguintes (RAJAPAKSE, 2008): (a) *Diversidade de hardware* - diferenças nos parâmetros de tela, quantidade de memória, poder de processamento, modos de entrada de dados, presença de *hardware* adicional como a câmera e o gravador de voz, bem como opções de conectividade; (b) *Diversidade de software* - diferentes plataformas e sistemas operacionais, plataformas de *middlewares*, versões de *software*, bem como diferentes versões de uma mesma aplicação; e (c) *Diversidade ambiental* - tais como a diversidade na infraestrutura de implantação como as restrições de acesso à rede e serviços de segurança.

2.2 Customizando *Software* para Dispositivos Móveis

A customização de *software* é necessária para adaptar o código fonte embarcado (i.e., *software* com o sistema operacional, plataformas de *middlewares*, aplicativos e conteúdo de mídia) no DM para atender às especificações de *interface* com o usuário, funcionalidades e configurações solicitadas pelas operadoras de telefonia. As adaptações necessárias e suas implementações variam de acordo com as plataformas de *software* e *hardware* do DM, com os requisitos da operadora de telefonia e, inclusive, com a região na qual o dispositivo será comercializado.

As principais adaptações são desenvolvidas manualmente ou automaticamente, geralmente, através de ferramentas proprietárias. A lista a seguir resume as principais adaptações realizadas no código fonte embarcado para atender aos requisitos das operadoras: (a) animação gráfica de *Power On/Off*; (b) a função das teclas do dispositivo (*keymap*); (c) idioma, menus, mensagens, textos de títulos de aplicativos, botões, listas e *grid menus*; (d) *ringtones* e papéis de parede; (e) configurações do *player* de *MP3*, agenda de contatos, vídeo e camera; (f) configurações de *MMS*, *SMS*, *Email*, *WAP* e *Bluetooth*; (g) configurações de ferramentas, jogos e aplicativos; (h) configurações de versão do *software*, *SIM Lock*, *SIM Toolkit* e suporte à criptografia.

Um exemplo de um processo de customização de *software* para celulares é apresentado na Figura 2.2 (CUNHA et al., 2011). Os principais envolvidos nesse processo são a organização, responsável pelas customizações de *software*, o fabricante de DMs e a operadora de telefonia. Existem casos em que esse fabricante também atua como a organização desenvolvedora, pois também desenvolve as customizações.

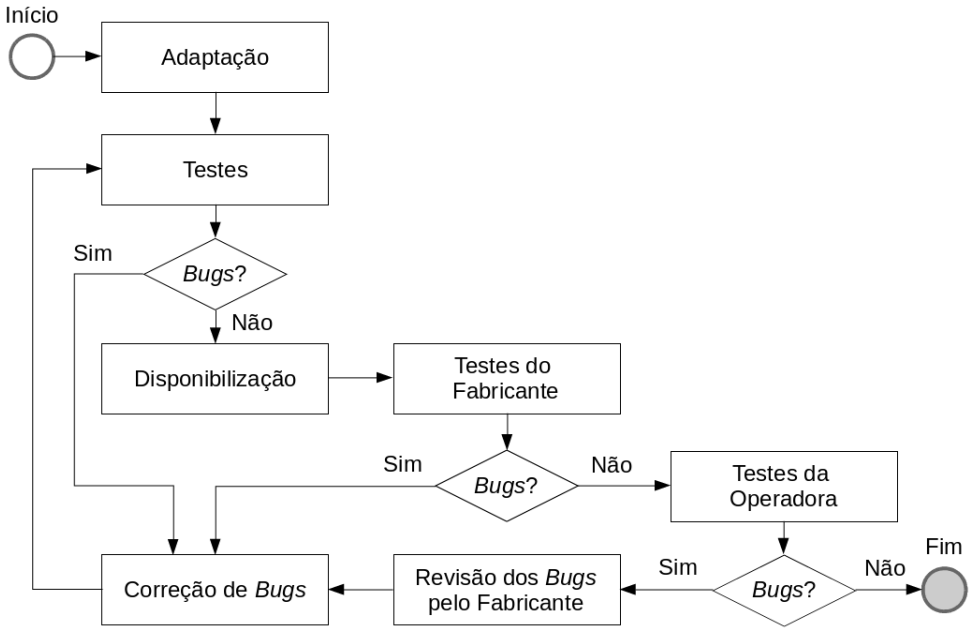
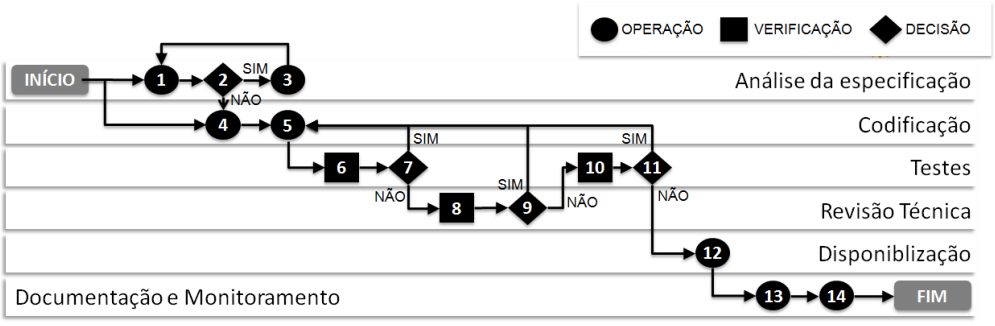


Figura 2.2: Processo de Customização de *Software* para Celulares, fonte: (CUNHA et al., 2011)

As etapas da customização do *software* como a adaptação do código base do DM, realização de testes, correções de *bugs* e disponibilização da versão customizada são geralmente

realizadas pela organização desenvolvedora. A versão de software resultante dessas etapas é validada pelo fabricante antes dos testes na operadora, que nesse processo é a responsável pela aprovação da customização. Essa aprovação é o marco final do processo de desenvolvimento da customização. Em seguida, o DM poderá ser fabricado com a customização desenvolvida e distribuído para comercialização.

Um exemplo de um mapa de processo da adaptação do *software*, uma das primeiras etapas do processo de customização, é apresentado na Figura 2.3. Um mapa de processo contém informações como as entradas e saídas do processo, os passos, os envolvidos e os responsáveis.



PASSOS DO PROCESSO

1) Analisar especificações da operadora	(4h)	8) Realizar revisão técnica	(2h)
2) Há informação pendente?	(1h)	9) Foram encontrados erros?	(0,5h)
3) Requisitar e receber a informação	(0,5h)	10) Realizar testes de aceitação	(1d)
4) Criar esqueleto de código	(4h)	11) Foram encontrados erros?	(0,5h)
5) Implementar adaptação	(2d)	12) Disponibilizar versão	(1d)
6) Realizar testes	(1d)	13) Atualizar documentos da versão	(2h)
7) Foram encontrados erros?	(0,5h)	14) Monitorar progresso dos testes	(2,5d)

Figura 2.3: Mapa do Processo de Adaptação do *Software*, fonte: (CUNHA et al., 2011)

A adaptação do *software* se inicia como a análise das especificações dos requisitos da customização (passo 1). Em caso de dúvidas ou mesmo de informações pendentes nessas especificações, essas dúvidas ou a solicitação de informações adicionais são encaminhadas à operadora (passos 2 e 3).

De posse das informações necessárias sobre os requisitos, a estrutura base do código (esqueleto) da customização é implementada (passo 4). Em seguida, as adaptações necessárias nessa estrutura são realizadas conforme as especificações da operadora (passo 4).

Após essa etapa de codificação, são realizados testes (passos 6 e 7) e a revisão técnica dos códigos e demais entregáveis é realizada em seguida (passos 8 e 9).

Em seguida, são realizados mais testes (passo 10 e 11) e se não forem encontrados erros, uma versão dessa customização é disponibilizada para os testes do fabricante (passo 12). Os documentos dessa versão são então atualizados e os testes realizados pelo fabricante são monitorados (passos 13 e 14).

2.3 Metodologias Ágeis

Uma metodologia de desenvolvimento é considerada ágil porque é incremental, com entregas frequentes de *software* e relativamente pequenas, cooperativa, clientes e desenvolvedores trabalhando juntos, simples, ou seja, fácil de aprender e modificar e adaptativa, capaz de sofrer alterações (ABRAHAMSSON et al., 2002).

As metodologias ágeis objetivam acelerar o desenvolvimento de *software* visando a melhoria contínua dos processos, gerando benefícios na comunicação e interação da equipe, organização, diminuição de falhas, respostas mais rápidas às mudanças e aumento da produtividade (SCHWABER KEN E BEEDLE, 2001).

Fundamentadas no desenvolvimento iterativo, flexibilidade e na entrega frequente e contínua de *software* ao cliente, essas metodologias têm contribuído na execução dos projetos e sua crescente difusão está relacionada à simplicidade como são descritas e à aderência às necessidades atuais do desenvolvimento de *software* (MARTIN, 2003b). As metodologias ágeis são uma alternativa às abordagens tradicionais, que atualmente ainda se aplicam às situações em que os requisitos do sistema são estáveis e previsíveis (SOARES, 2004).

A história das metodologias ágeis obteve um marco significativo em 2001, quando membros da comunidade mundial de desenvolvimento de *software* reuniram-se para discutir boas práticas, abordando e caracterizando os fatores para o sucesso dos projetos de *software*. As conclusões foram publicadas no Manifesto Ágil (BECK, 2001), que contém os valores e princípios que fundamentam o Desenvolvimento Ágil de *Software*, os quais são apresentados na Tabela 2.2.

Os valores e princípios ágeis apresentados na Tabela 2.2 tem foco nas pessoas (equipe, cliente, usuários e demais envolvidos) e suas interações, na entrega de *software* funcionando e com valor real para o negócio do cliente, na colaboração mútua entre essa equipe e o cliente facilitando o entendimento das necessidades de ambas as partes e também na postura proativa frente às mudanças, na expectativa que as mesmas possam trazer vantagens ao cliente (MARTIN, 2003a).

O uso desses valores e princípios objetiva alcançar três tipos de sucessos: o pessoal, que está relacionado à motivação individual dos profissionais, o técnico, relacionado à qualidade das entregas do projeto, principalmente do código-fonte, quanto ao seu correto funcionamento e atendimento aos requisitos e o sucesso organizacional, relacionado aos resultados do projeto e ao retorno sobre o investimento para a organização (SHORE; WARDEN, 2007).

Muitas metodologias ágeis são utilizadas no desenvolvimento de *software* para dispo- sitivos móveis como por exemplo o *Scrum*, *eXtreme Programming* (*XP*) (WELLS, 2009), *Feature Driven Development* (*FDD*), *Adaptive Software Development* (*ASD*) e *Crystal* (COCKBURN, 2004). Essas metodologias compartilham os valores e princípios do Manifesto Ágil e possuem, dessa forma, algumas práticas semelhantes, porém cada uma dessas metodologias utiliza uma terminologia própria.

Tabela 2.2: Valores e Princípios do Manifesto Ágil, fonte: (BECK, 2001)

Valores
Indivíduos e interações mais que processos e ferramentas
Software em funcionamento mais que documentação abrangente
Colaboração com o cliente mais que negociação de contratos
Responder a mudanças mais que seguir um plano
Princípios
A prioridade é satisfazer o cliente através da entrega frequente e contínua de *software* com valor agregado
Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente
Entregar frequentemente *software* funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo
Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto
Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho
O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face
Software funcionando é a medida primária de progresso
Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente
Contínua atenção à excelência técnica e bom design aumentam a agilidade
Simplicidade, a arte de maximizar a quantidade de trabalho não realizado, é essencial
As melhores arquiteturas, requisitos e *designs* emergem de equipes auto-organizáveis
Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo

O objetivo dessas metodologias é tornar o desenvolvimento de *software* enxuto e rápido, entregando *software* de qualidade, capaz de atender às necessidades do usuário e contribuir para a melhoria do negócio do cliente.

2.4 *Scrum*

Criado em 1996 por *Ken Schwaber* e *Jeff Sutherland*, o *Scrum* é uma metodologia ágil para gestão de projetos, desenvolvimento e manutenção de produtos de *software* (SCHWABER, 2007a). Essa metodologia adota uma abordagem empírica e pode ser utilizada em projetos complexos com, por exemplo, grande quantidade de produtos diferentes, uso simultâneo de várias tecnologias, elevado número de envolvidos, necessidade de conhecimentos técnicos especializados, muitas restrições tecnológicas e incertezas associadas (SCHWABER, 2007b).

O *Scrum* adota os princípios ágeis definidos no Manifesto Ágil (veja a Tabela 2.2) e seus principais fundamentos estão relacionados a esses princípios e são descritos a seguir: (i) *Transparência* - todos os artefatos do projeto precisam estar visíveis aos envolvidos e as decisões e definições do projeto devem ser compartilhadas de forma efetiva. Um artefato é um dos vários tipos de subprodutos utilizados ou produzidos durante o desenvolvimento

com o *Scrum*. Por exemplo, o *Backlog* do Produto, conjunto dos requisitos conhecidos do produto, é um artefato utilizado no gerenciamento do escopo do produto e planejamento dos lançamentos do projeto. Esse *Backlog* do Produto precisa está sempre atualizado e visível a todos os envolvidos. (ii) *Inspeção* - os artefatos são continuamente inspecionados e os problemas encontrados devem ser resolvidos o mais rápido possível; e (iii) *Adaptação* - a melhoria contínua deve ser executada em todas as frentes do projeto, com foco na melhoria da produtividade do time e da qualidade do produto.

O *Scrum* define apenas três papéis: o *Product Owner*, que representa o cliente e é responsável pela gestão do escopo; o *Scrum Master*, que é o líder-servidor do time e responsável por garantir que o *Scrum* seja compreendido e seguido, e o Time, que realiza o desenvolvimento propriamente dito.

Uma equipe do *Scrum* é composta por esses papéis e seu tamanho ideal é de 5 a 11 integrantes. Essa equipe deve possuir todas as competências necessárias para tornar o conjunto dos requisitos do produto potencialmente funcional ao ponto de compor o incremento do *software*, não sendo necessário auxilio de equipes externas.

Os principais artefatos e eventos do *Scrum* são descritos a seguir:

- *Backlog do Produto* - conjunto completo dos requisitos funcionais e não funcionais conhecidos do produto, com os itens priorizados pelo *Product Owner*. O *Backlog* do Produto evolui de acordo com o produto e o ambiente ao qual está inserido e o *Product Owner* é responsável por mantê-lo atualizado;

- *Plano de Lançamentos* - reflete as expectativas sobre quais itens do *Backlog* do Produto serão implementados e quando eles estarão concluídos. Esse plano também serve como base para monitorar o progresso do projeto. Os lançamentos podem ser entregas intermediárias realizadas durante o projeto ou mesmo a entrega final.

- *Sprint* - janela de tempo que corresponde a uma iteração, cujo tamanho ideal é de 1 a 4 semanas;

- *Backlog do Sprint* - contém as atividades planejadas pelo Time para atender aos itens priorizados do *Backlog* do Produto. Apenas o Time pode modificar o *Backlog* do *Sprint* e é responsável por mantê-lo atualizado;

- *Planejamento do Sprint*- reunião de planejamento realizada pelo Time para a execução do *Sprint*;

- *Reunião Diária* - reunião envolvendo o time e o *Scrum Master* onde cada participante informa o que fez, o que fará e quais são os impedimentos que estão atrapalhando a execução das atividades;

- *Gráfico de Burndown* - utilizado pelas equipes para representar diariamente o progresso do trabalho em desenvolvimento. Ou seja, após cada dia de trabalho esse gráfico apresenta a porção de trabalho finalizada em comparação com o trabalho total planejado;

- *Incremento do Produto* - A cada *Sprint*, o Time entrega incrementos de funcionalidade, que consistem em códigos revisados, testados e executáveis, bem como acompanhados da documentação necessária para a sua utilização;
- *Revisão do Sprint* - reunião em que o *Product Owner* revisa os *softwares* e demais produtos entregues e aprova ou não o *Sprint*; e
- *Retrospectiva do Sprint* - reunião em que o time reporta os pontos positivos e negativos do *Sprint* anterior, identificando as ações necessárias para que os problemas encontrados sejam evitados nos *Sprints* futuros.

O ciclo de desenvolvimento do *Scrum* é baseado em três fases principais, conforme apresentado na Figura 2.4. Essas fases e o fluxo de execução do *Scrum* são explicados a seguir.

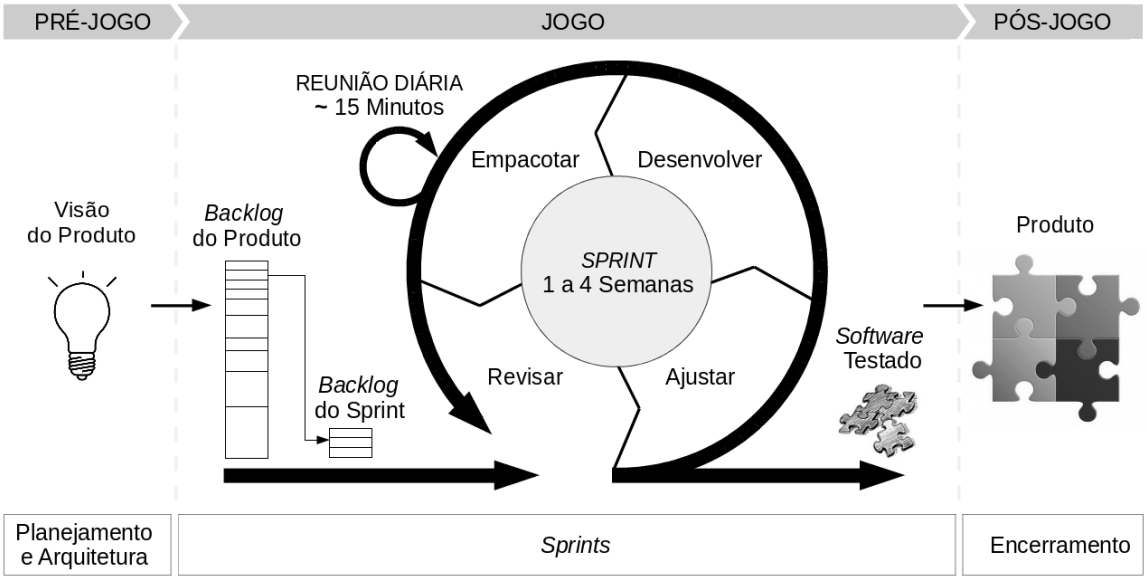


Figura 2.4: Fases do Ciclo de Desenvolvimento do *Scrum*

- *Pré-jogo* - dividido em duas subfases, Planejamento e Preparação. Na primeira, a visão do projeto e as expectativas dos envolvidos são especificados e também são definidos os recursos necessários para a execução desse planejamento. A segunda tem por objetivo identificar itens adicionais do *Backlog* do Produto (além dos itens do escopo do produto) como, por exemplo, os relacionados ao tipo do sistema, Time, ambiente de desenvolvimento, testes e homologação. A especificação de uma arquitetura inicial do sistema pode ser realizada na Preparação;
- *Jogo* - consiste no planejamento e execução de múltiplas *Sprints* para o desenvolvimento dos incrementos do produto. O Time analisa a situação atual do produto, avalia quais mudanças devem ser implementadas, procedendo com o desenvolvimento através de análise, projeto, implementação, testes e documentação;
- *Pós-jogo* - fase de encerramento, que tem por objetivo preparar o produto para o seu lançamento. As atividades de encerramento incluem, por exemplo, a integração do

produto, documentação do usuário, realização de treinamento, implantação do sistema no ambiente do cliente.

O diagrama de *SIPOC* (*Suppliers*, *Inputs*, *Process*, *Outputs* e *Customers*) do *Scrum* (EXPEDITH, 2011) é apresentado na Figura 2.5. O *SIPOC* é uma ferramenta do *LSS* (detalhado na Seção 2.6) utilizada para identificar e mapear elementos relevantes de um processo como as entradas, saídas, etapas e papéis envolvidos.

Nesse diagrama da Figura 2.5, por exemplo, o fluxo do *Scrum* e seus principais eventos são detalhados. A sequência de números de 1 a 10 indica a ordem dos acontecimentos das etapas em cada um dos eventos. As colunas do diagrama apresentam os principais eventos do *Scrum*, enquanto as linhas relacionam as etapas de cada evento aos papéis. Esse fluxo é explicado em detalhes a seguir.

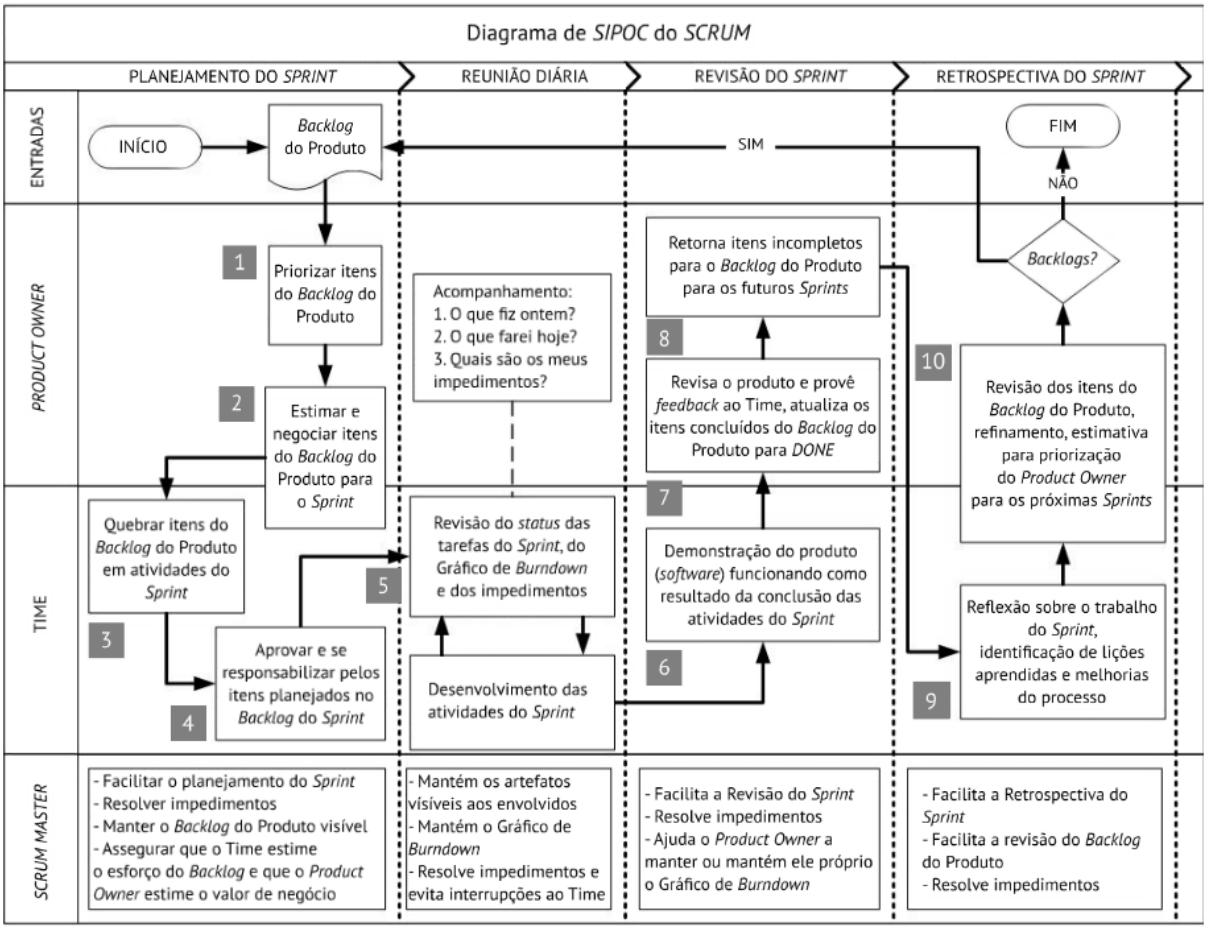


Figura 2.5: Diagrama de *SIPOC* do *Scrum*, fonte: (EXPEDITH, 2011)

Um projeto se inicia com a visão do produto e uma lista inicial de suas características, informações comumente identificadas em conjunto com o cliente, além de premissas e restrições.

O *Backlog* do Produto é, então, elaborado pelo *Product Owner* a partir dessas informações. Nesse *Backlog*, o *Product Owner* lista e prioriza os requisitos funcionais e

não funcionais do sistema (etapa 1). Com base nessa priorização, é elaborado o Plano de Lançamentos, que contém os *Sprints* para o desenvolvimento iterativo e incremental do produto.

Cada *Sprint* inicia-se com uma reunião de planejamento (Planejamento do *Sprint*), na qual o *Product Owner* e o Time decidem em conjunto o que deverá ser implementado, ou seja, quais itens do *Backlog* do Produto serão incluídos no *Sprint*.

Essa reunião de planejamento é dividida em duas partes. Na primeira parte, o *Product Owner* apresenta os requisitos prioritários e que devem ser implementados. O Time então define colaborativamente o que poderá entrar no desenvolvimento do próximo *Sprint*, considerando sua velocidade (capacidade de produção) (etapa 2). Na segunda parte, o Time planeja seu trabalho, definindo o *Backlog* do *Sprint*, que são as tarefas necessárias para implementar as funcionalidades selecionadas do *Backlog* do Produto (etapas 3 e 4). Essa tarefas podem ser alteradas ao longo do *Sprint* pelo Time. Na execução dos *Sprints*, o Time se reúne diariamente (Reunião Diária, por volta de 15 minutos) para acompanhar o progresso do trabalho (etapa 5).

A reunião de revisão (Revisão do *Sprint*) é realizada ao final do *Sprint* para que o Time apresente ao *Product Owner* o resultado alcançado na iteração (etapa 6). Nesse momento, as funcionalidades são inspecionadas e adaptações do projeto podem ser realizadas (etapas 7 e 8). Em seguida o *Scrum Master* conduz a reunião de retrospectiva (Retrospectiva do *Sprint*), com o objetivo de identificar ações para a melhoria do processo, atuação do Time e qualidade do produto (etapas 9 e 10).

Embora existam muitas informações e treinamentos sobre o *Scrum*, a adoção dessa metodologia é particular ao projeto e à organização e, dessa forma, cada implantação torna-se um desafio único e que não deve ser subestimado.

2.5 *Lean Manufacturing*

O termo "*Lean Manufacturing*" foi utilizado pela primeira vez em 1988 no artigo "*Triumph of the Lean Production System*" escrito pelo engenheiro de qualidade da *Toyota*, *John Krafcik*. A pesquisa iniciada por *Krafcik* foi continuada por *James Womack*, autor do *Best-Seller* "A Máquina que Mudou o Mundo", de 1990. Esse livro descreve a história da indústria automobilística, apresentando um estudo comparativo dos métodos utilizados por japoneses, americanos e europeus em suas linhas de produção.

O *Lean Manufacturing* (*Lean*) é uma filosofia de gestão, derivada do Sistema *Toyota* de Produção (STP), e é focada na redução de desperdícios relacionados à super-produção, tempo de espera, transporte, excesso de processamento, inventário, movimento e defeitos(AULAKH; GILL, 2008).

Os princípios do *Lean* também têm sido adotados no desenvolvimento e manutenção de *software* em organizações da área de TIC (EBERT et al., 2012). O uso desses princípios nessas organizações é focado na redução de desperdícios dos processos de desenvolvimento e

serviços, a partir da remoção ou não priorização de atividades que não geram valor ao negócio do cliente, considerando suas necessidades e expectativas (ABBAS; MEHTA, 2011).

O *Lean* é baseado nos seguintes princípios (POPPENDIECK; CUSUMANO, 2012):

1. *Eliminar desperdícios* - tudo o que não agrega valor para o cliente nos processos, serviços ou mesmo no produto em desenvolvimento é considerado um desperdício. Dessa forma, em uma aplicação de *software*, funcionalidades que não resultem em benefícios ao negócio do cliente devem ser removidas ou não priorizadas;

2. *Gerar qualidade* - o objetivo é controlar as condições dos processos e serviços de forma a evitar que os defeitos ocorram. Uma forma de diminuir os defeitos é identificar as suas causas raízes, definir e executar ações para eliminá-las;

3. *Criar conhecimento* - deve-se focar na compreensão das principais expectativas e necessidades do cliente. Esse conhecimento objetiva priorizar o desenvolvimento das funcionalidades importantes para o negócio desse cliente. Além disso, através de entregas frequentes de *software*, dos testes dos usuários finais e da obtenção de *feedbacks*, os processos e serviços podem ser ajustados para que melhores resultados sejam alcançados nas entregas seguintes.

4. *Adiar compromisso* - decisões irreversíveis devem ser tomadas o mais tarde possível, para manter as opções em aberto por mais tempo;

5. *Entregar rápido* - a equipe deve continuamente melhorar os processos de desenvolvimento e serviços para dar velocidade ao desenvolvimento possibilitando entregas mais frequentes e rápidas. Essa estratégia colabora na identificação e correção de defeitos, contribuindo na qualidade global do projeto;

6. *Respeitar as pessoas* - deve-se demonstrar dedicação em ajudar a equipe a obter o sucesso. Para tanto, é necessário respeitar a integridade e as decisões da equipe, permitindo que ela própria organize e planeje seu trabalho, com responsabilidade e recompensa; e

7. *Otimizar o todo* - deve-se focar na melhoria dos processos e serviços utilizados no desenvolvimento do produto e que geram valor ao cliente e não somente em uma pequena parte do processo de desenvolvimento.

Segundo (WANG, 2011), as metodologias ágeis podem ser combinadas ao *Lean* de diferentes maneiras e para diferentes fins no desenvolvimento de *software*, conforme é apresentado na Tabela 2.3. Os princípios do *Lean* são por vezes utilizados como guias no uso de práticas ágeis e muitas vezes como forma de facilitar a adoção dessas práticas e metodologias ágeis no desenvolvimento de *software*.

De acordo com Wang (2011), há três tipos de percepção sobre a utilização dos princípios e métodos ágeis e do *Lean* (veja a Tabela 2.3): a primeira é que essas metodologias tem o mesmo significado e portanto não há diferenciação entre as mesmas e essa percepção ocorre geralmente quando a combinação delas não é realizada de forma proposital; a segunda

Tabela 2.3: Diferentes Combinações de *Agile* e *Lean* no Desenvolvimento de *Software*, fonte: (WANG, 2011)

Percepção das Diferenças entre *Agile* e *Lean*	**Tipos de Combinações**
Sem diferenciação entre *Agile* e *Lean*	A combinação dessas metodologias não é proposital
Agile e *Lean* estão em níveis diferentes. *Lean* é um conjunto de princípios e *Agile*, conjunto de práticas	Uso dos princípios *Lean* para orientar o desenvolvimento e a adequação aos métodos ágeis
Agile e *Lean* estão no mesmo nível, mas possuem escopo e foco diferentes	Implementação top-down do *Lean* para criar o ambiente para evoluir o bottom-up ágil
	Uso do *Lean* para escalar o ágil
	Uso do *Lean* para melhorar os processos ágeis de desenvolvimento
	Uso de práticas ágeis para dar suporte aos processos de desenvolvimento *Lean*

percepção diferencia essas metodologias, considerando que o *Lean* é um conjunto de princípios, enquanto os métodos ágeis são um conjunto de práticas; e a terceira percepção é que essas metodologias tem escopo e foco diferentes e que podem ser combinadas de forma a trazer benefícios para ambas as partes.

2.6 *Six Sigma* e *Lean Six Sigma*

O *Six Sigma* é uma metodologia quantitativa que objetiva aumentar o desempenho e a lucratividade das empresas. Criado em meados de 1980, o *Six Sigma* é amplamente utilizado por seus resultados comprovados e é considerado como "a metodologia de qualidade do século XXI" (WERKEMA, 2012). Essa metodologia melhora os resultados financeiros e de qualidade das empresas a partir da identificação e remoção das causas de defeitos e da diminuição da variabilidade de produtos, processos e serviços.

O *Six Sigma* utiliza um conjunto de métodos e ferramentas de gestão de qualidade, incluindo métodos estatísticos, para realizar a análise detalhada dos processos, a fim de descobrir as causas dos defeitos (CARVALHO et al., 2013). Isso é conseguido através do uso de duas submetodologias, o *DMAIC* (*Define, Measure, Analyze, Improve, Control*) e *DMADV* (*Define, Measure, Analyze, Design, Verify*). O *DMAIC* é utilizado para a melhoria de processos existentes, enquanto o *DMADV* é utilizado para o desenvolvimento de novos processos ou produtos em níveis de qualidade *Six Sigma*.

As fases dessas submetodologias são sumarizadas na Tabela 2.4 (GEORGE, 2003). Essas fases são executadas tradicionalmente na sequência apresentada e, antes mesmo dessas fases, pode ser realizada uma etapa inicial de planejamento.

O *Six Sigma* cria uma infraestrutura de pessoas que são especialistas nesses métodos. Cada projeto *Six Sigma* segue uma sequência definida de passos e tem metas quantificadas de redução de custos financeiros (e aumento do lucro).

Tabela 2.4: Submetodologias DMAIC e DMADV, fonte: (GEORGE, 2003)

	Fase	Descrição
DMAIC	Definir (Define)	Definir os objetivos do projeto e de clientes (interno e externo)
	Medir (*Measure*)	Medir o processo para determinar o desempenho atual
	Analisar (*Analyze*)	Analisar e determinar as causas de defeitos
	Melhorar (*Improve*)	Melhorar o processo a partir da eliminação de defeitos
	Controlar (*Control*)	Controlar o desempenho do processo após implantação de melhorias
DMADV	Definir (Define)	Definir os objetivos do projeto e de clientes (interno e externo)
	Medir (*Measure*)	Determinar as especificações do cliente
	Analisar (*Analyze*)	Analisar o processo quanto ao atendimento das necessidades do cliente
	Projetar (*Design*)	Realizar o projeto do processo com foco nas necessidades do cliente
	Verificar (*Verify*)	Verificar se o desempenho e a capacidade do novo processo atendem às necessidades do cliente

O *Lean Six Sigma* (*LSS*) é uma metodologia para definição e melhoria de produtos, processos e serviços com foco na redução de defeitos ou falhas, na eliminação da variação e dos desperdícios, priorizando, de forma planejada e objetiva, a obtenção de resultados de qualidade e financeiros (GEORGE, 2003). Essa metodologia consiste na integração dos princípios e técnicas do *Lean* (POPPENDIECK, 2005) e do *Six Sigma* e potencializa os benefícios dessas duas metodologias, aliando os ganhos em velocidade do *Lean* aos ganhos em qualidade do *Six Sigma*.

O *LSS* identifica os seguintes papéis: Liderança Executiva, que é a alta gerência responsável pela visão estratégica da organização, a qual será utilizada para a implantação e execução do *LSS*; *Champions*, que assumem a responsabilidade pela implantação da metodologia na organização; *Master Black Belts*, que atuam como orientadores internos; *Black Belts*, que aplicam a metodologia em projetos específicos e *Green Belts*, que se ocupam da execução dos projetos *LSS*, juntamente com suas outras responsabilidades, sob a orientação de *Black Belts*.

Um dos conceitos importantes do *LSS* é a capacidade do processo atender aos limites de especificação exigidos pelo cliente. Quanto maior essa capacidade, menor a quantidade de defeitos em relação aos limites especificados e maior o nível de *sigma* associado à qualidade. O *LSS* considera como defeitos quaisquer características do produto, processo ou serviço que estão fora dos limites de especificação ou que não atendem aos requisitos do cliente (HARRY, 1998). Alguns níveis de *sigma* e seus respectivos defeitos por milhão de oportunidades (*DPMO*) e percentuais de defeitos são apresentados na Tabela 2.5.

Quanto menor o nível *sigma*, maior a quantidade de defeitos a serem encontrados no processo. O nível 6 *sigma*, por exemplo, representa um processo com 3,4 *DPMO* (GEORGE, 2003). Isso significa que a cada milhão de vezes que esse processo for executado, estatisticamente, ocorrerão apenas 3,4 defeitos. Já um processo com nível 3 *sigma* poderá apresentar cerca de 66.807 defeitos. Dessa forma, quanto maior o sigma do processo melhor será a sua capacidade e a qualidade dos produtos gerados a partir do mesmo.

Tabela 2.5: DPMO e Percentuais de Defeitos de Níveis Sigma, fonte: (GYGI et al., 2005)(EL-HAIK; SUH, 2005)

Nível de Qualidade	DPMO	Defeitos (%)
1 sigma	691.462	69
2 sigma	308.538	31
3 sigma	66.807	6,7
4 sigma	6.210	0,62
5 sigma	233	0,023
6 sigma	3,4	0,00034

2.7 Boas Práticas na Elaboração e Aplicação de *Checklists*

Os *checklists* são úteis para determinar a situação corrente de um determinado processo, possibilitando comparações com situações anteriores e posteriores do mesmo (COOK, 1995). Frequentemente, quando se pretende controlar e melhorar um processo é necessário inicialmente definir como medi-lo. Entretanto, essa medição não melhora o processo, mas possibilita identificar o estado em que se encontra e estabelecer a necessidade de ações para a melhoria de seus resultados.

A inspeção através da aplicação de *checklists* é por muitas vezes um meio efetivo de coleta de dados. Esses *checklists* são bastante comuns e aparentam ser simples, entretanto demandam um certo esforço em sua adequada elaboração e aplicação. Existem boas práticas que ajudam na elaboração e aplicação desses *checklists* e as utilizadas nesse trabalho são apresentadas a seguir (DIEM, 2002):

- Determine o propósito da pesquisa - determine o que é necessário investigar e porquê, também como o resultado da pesquisa será utilizado;
- Decida o que está sendo medido - as questões devem estar alinhadas ao que está sendo medido e, portanto, desde o início é importante definir claramente o objeto da pesquisa;
- Defina quem responde à pesquisa - o grupo de pessoas que responde ao *checklist*, como esse grupo é amostrado e o tamanho dessa amostra afetam como os resultados dessa pesquisa podem ser generalizados;
- Considere o público alvo - o *checklist* deve ser apropriado ao público ao qual será aplicado e, portanto, fatores como a familiaridade com *checklists*, aspectos culturais, conhecimentos técnicos são importantes e devem ser considerados;
- Escolha o método de coleta de dados - email, telefone, entrevista presencial ou pesquisa via *Web*;
- Selecione o procedimento de coleta: anônimo ou confidencial;
- Escolha a escala de medição e pontuação: as questões podem ser por exemplo do tipo escala de classificação, múltipla escolha, de respostas sim ou não e abertas. Entretanto,

o importante é que o formato dessas questões seja adequado à informação medida e seja compreensível aos envolvidos;

- Inclua instruções de preenchimento - por exemplo, informações como as questões devem ser respondidas e o prazo para responder o *checklist*;
- Use uma linguagem simples e direta - não utilize jargões e acrônimos e inclua definições quando necessário;
- Seja breve - o *checklist* deve ser enxuto, com foco no que é essencial saber para a pesquisa;
- Ponha as questões mais importantes no início do *checklist*;
- Assegure que as questões estão alinhadas com o formato e categorias das opções de resposta;
- Realize apenas uma pergunta por questão;
- Organize as questões numa ordem lógica - numere-as e agrupe-as por similaridades; e
- Forneça um espaço para comentários e sugestões.

2.8 Considerações Finais

Este capítulo apresentou os principais conceitos relacionados ao tema e que serviram de alicerce para o desenvolvimento deste trabalho. Ele introduziu os principais desafios do desenvolvimento e customização de *software* para DMs e focou na apresentação detalhada do *Scrum* e do *LSS*, por serem as metodologias utilizadas na abordagem proposta nesta dissertação. Também foram apresentadas as boas práticas utilizadas na elaboração e aplicação de *checklists*, úteis na avaliação do uso de princípios e práticas *Scrum* nos projetos.

O capítulo seguinte descreve os trabalhos relacionados ao uso de metodologias ágeis e de gestão da qualidade no desenvolvimento de *software* para DMs, bem como os trabalhos relacionados à avaliação e melhoria do uso de práticas ágeis em projetos de *software*.

3 TRABALHOS RELACIONADOS

Neste capítulo é conduzida uma revisão de literatura sobre o desenvolvimento de *software* para dispositivos móveis (DMs), com foco na utilização de metodologias ágeis e gestão da qualidade frente aos desafios desse desenvolvimento.

Também são analisados os trabalhos relacionados à adoção, avaliação e melhoria do uso de métodos ágeis como a utilização de modelos de maturidade ágil e *checklists* para medir a forma de adoção de princípios e práticas ágeis em projetos de *software*.

Parte das metodologias ágeis analisadas neste capítulo são voltadas especificamente ao desenvolvimento para DMs e, além dessas, são também analisadas metodologias que, embora não sejam específicas a esse desenvolvimento, são aceitas e utilizadas para tal, contribuindo na melhoria de agilidade desse desenvolvimento.

Já sobre as metodologias de gestão da qualidade, são analisados trabalhos que combinam as metodologias ágeis com algum método de gestão da qualidade no desenvolvimento de *software*, não especificamente para DMs. Também os trabalhos sobre modelos de maturidade ágil e os *checklists* aqui analisados são voltados ao desenvolvimento de *software* em geral.

Para encontrar os trabalhos relacionados elencados neste capítulo foi realizada uma busca na literatura por publicações sobre pelo menos um dos seguintes tópicos: (a) uso de metodologias ágeis no desenvolvimento para DMs, procurando identificar as metodologias mais utilizadas e como é a adoção do *Scrum* nesse desenvolvimento; (b) uso de metodologias de gestão da qualidade no desenvolvimento para DMs, com foco em trabalhos relacionados ao *Lean*, *Six Sigma* e *LSS*; e (c) combinação de metodologias ágeis e de gestão da qualidade no desenvolvimento de *software*, para analisar estratégias de combinação do *Scrum* e *LSS*. Ainda em relação às buscas, embora não tenha sido executada uma revisão sistemática (KITCHENHAM, 2004), para cada um dos tópicos supracitados foram definidas *strings* de busca. Os locais de buscas utilizados foram as bases *ACM DL Library*[1] e o *IEEE Explorer*[2], bem como os anais do Simpósio Brasileiro de Engenharia de *Software* - *SBES* e do Simpósio Brasileiro de Qualidade de *Software* - *SBQS*.

A organização das seções deste capítulo é apresentada na Figura 3.1. Inicialmente, a Seção 3.1 apresenta os trabalhos que propõem métodos ágeis no desenvolvimento de *software* para DMs. Em seguida, a Seção 3.2 expõe os trabalhos que propõem combinações do uso de metodologias ágeis com métodos de gestão da qualidade no desenvolvimento de *software*, embora essas combinações não tenham sido utilizadas no desenvolvimento para DMs. A Seção 3.3 analisa os trabalhos relacionados à adoção, avaliação e melhoria do uso de métodos ágeis em projetos de *software* em geral. A Seção 3.4 apresenta um resumo comparativo entre as abordagens dos trabalhos elencados e a Seção 3.5 finaliza o capítulo com as considerações finais sobre esses trabalhos.

[1] http://dl.acm.org
[2] http://ieeexplore.ieee.org

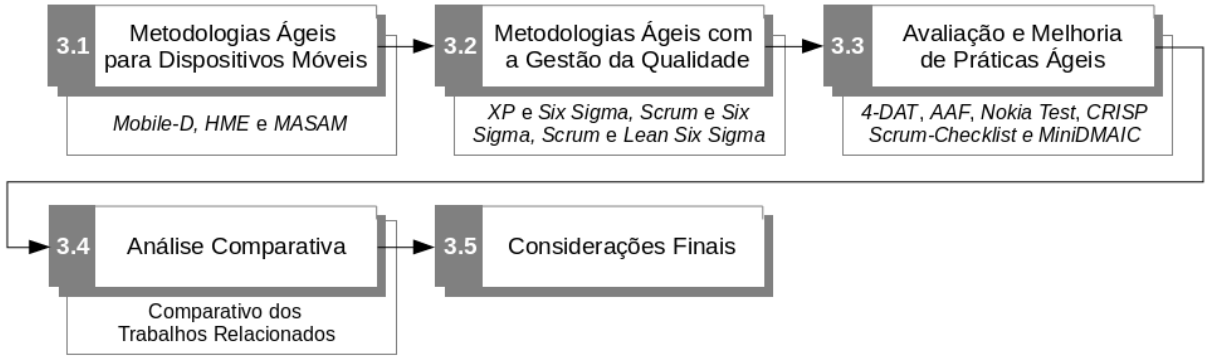


Figura 3.1: Organização das Seções do Capítulo 3

3.1 Metodologias Ágeis para Dispositivos Móveis

Devido à grande demanda por uma variedade de aplicativos para DMs como celulares, *smartphones* e *tablets* e os benefícios econômicos associados à essa demanda, um número crescente de empresas, instituições e pesquisadores demonstram cada vez mais interesse em como organizar o desenvolvimento de forma a entregar soluções mais rápidas, melhores e com menor custo e uma das respostas é a adoção de metodologias ágeis.

Conforme mencionado no Capítulo 2, as metodologias ágeis são amplamente aceitas e utilizadas no desenvolvimento de *software* para DMs por se ajustarem às rápidas mudanças de requisitos e ambiente, bem como por atenderem bem às demandas de prazo desse nicho de mercado (IHME; ABRAHAMSSON, 2004). Além disso, o uso dessas metodologias contribui na melhoria da produtividade da equipe e da qualidade do produto (LAYMAN et al., 2004)(ILIEVA et al., 2004).

Shen *et al.* (SHEN et al., 2012) apresentam uma revisão sistemática sobre o uso de métodos ágeis no desenvolvimento de *software* para DMs e sinalizam que metodologias como o *XP* (WELLS, 2009), *Scrum* (SCHWABER, 2007a), *Feature Driven Development* (*FDD*), *Adaptive Software Development* (*ASD*), *Rational Unified Process* (*RUP*) (JACOBSON et al., 1999) e *Crystal* (COCKBURN, 2004) são utilizadas sozinhas ou de forma híbrida (combinadas) nesse desenvolvimento e estão, nessa ordem, entre as que possuem mais trabalhos relacionados ao tema na literatura.

Entre essas metodologias ágeis, as mais aceitas e utilizadas são o *Scrum* e *XP*, embora não tenham sido especificamente propostas com foco nesse desenvolvimento. O *Scrum* proporciona uma abordagem ágil voltada à gestão de projetos aumentando a probabilidade de sucesso desses projetos, enquanto o *XP* é utilizado mais no desenvolvimento de *software* propriamente dito com práticas como o *Test Driven Development* (TDD) (BECK, 2002) e a Programação em Pares (BECK; ANDRES, 2004), que estão diretamente relacionadas à forma como o *software* é desenvolvido (SALO; ABRAHAMSSON, 2008).

Nas próximas seções são apresentadas as abordagens propostas especificamente para o desenvolvimento de *software* para DMs.

3.1.1 *Mobile-D*

Abrahamsson *et al.* (ABRAHAMSSON et al., 2004) propõem o *Mobile-D* como uma abordagem ágil para o desenvolvimento de *software* para DMs. O *Mobile-D* é baseado nas práticas de desenvolvimento do *XP*, no método de escalabilidade das metodologias *Crystal* e no ciclo de vida do *RUP*. O *XP* é uma metodologia de desenvolvimento ágil de *software*, com entregas frequentes em pequenos ciclos de desenvolvimento, cujo objetivo é melhorar a qualidade do *software* e a capacidade de resposta à evolução das necessidades do cliente. As metodologias *Crystal* são técnicas e métodos para a gestão de pessoas, que podem ser selecionados e adaptados de acordo com a criticidade do projeto e tamanho da equipe de desenvolvimento. Já o *RUP* é um processo de desenvolvimento de *software* que define práticas a serem seguidas pela equipe, customizadas de acordo com o cenário do projeto, e que objetivam aumentar a produtividade dessa equipe no processo de desenvolvimento.

O *Mobile-D* divide o desenvolvimento de um projeto em cinco iterações (veja a Figura 3.2): *Setup*, *Core*, *Core2*, Estabilizar e Empacotar. Em cada uma dessas iterações são executados diferentes dias de desenvolvimento (tipos de eventos): Dia de Planejamento, Desenvolvimento, Lançamento ou Disponibilização e Dia de Integração, esse último só se faz necessário em caso de múltiplos times trabalhando no mesmo projeto. Nessas iterações, podem ser utilizadas as seguintes práticas: *Phasing* e *Pacing*, *Architecture Line*, *TDD*, Integração Contínua, Programação em Pares, Métricas, Melhoria do Processo Ágil, *Off-Site Customer* e Foco Centrado no Usuário.

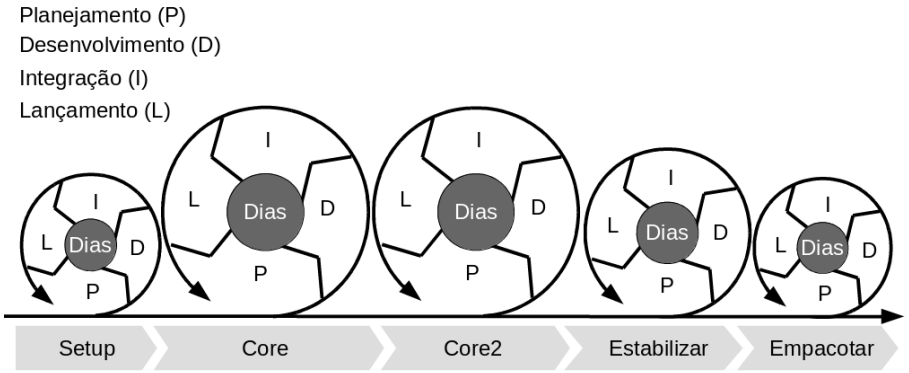


Figura 3.2: Iterações do Mobile-D

O *Mobile-D* foi empiricamente testado no desenvolvimento de extensões de sistemas de banco de dados para DMs, utilizado com times de até dez desenvolvedores, trabalhando juntos em um mesmo local geográfico, com o objetivo de desenvolver aplicações completas em menos de dez semanas. De acordo com os autores, foram observados os seguintes pontos positivos: melhoria da visibilidade do progresso do projeto, identificação precoce e resolução de problemas técnicos, responsabilidade compartilhada, partilha eficiente de informações, alta coerência entre a prática e o processo, baixa densidade de defeitos nos produtos lançados e ritmo constante de desenvolvimento.

Parte das práticas utilizadas no *Mobile-D* são advindas do *XP*, por exemplo, o *TDD*, Integração Contínua e a Programação em Pares. Por conseguinte, parte dos benefícios dessa

metodologia são resultado direto do uso dessas práticas do *XP*. Além disso, há indícios da utilização de métricas para o monitoramento de como a metodologia está sendo utilizada e de seus resultados durante a execução das iterações e entre as práticas definidas pelo *Mobile-D* há uma para a avaliação e melhoria do processo ágil utilizado.

Entretanto, segundo Rahimian *et al.* (RAHIMIAN; RAMSIN, 2008), embora o *Mobile-D* pareça promissor, a sua documentação está superficial e incompleta. Considerando o cenário desta dissertação, isso compromete o seu reuso sistemático para o desenvolvimento de *software* para DMs.

3.1.2 *Hybrid Method Engineering*

Rahimian e Ramsin (RAHIMIAN; RAMSIN, 2008) propõem a *Hybrid Method Engineering* (*HME*), uma abordagem ágil criada a partir da *Methodology Engineering* (*ME*) (BRINKKEMPER, 1996) e *Hybrid Methodology Design* (*HMD*) (RASMIN, 2006). A *ME* é a disciplina da engenharia responsável por todas as atividades relacionadas ao projeto, construção e adaptação de métodos, técnicas e ferramentas para o desenvolvimento de sistemas de informação e pode ser utilizada na elaboração de novas metodologias para diferentes cenários de desenvolvimento. Já o *HMD* é uma metodologia que utiliza conhecimentos e requisitos chaves de outras metodologias, padrões de processo e metamodelos para a elaboração de novas metodologias.

Além da apresentação da nova metodologia, a *HME*, uma das principais contribuições de Rahimian e Ramsin (2008) está na identificação de características de uma metodologia ideal para o desenvolvimento de *software* para DMs. Essas características são apresentadas a seguir:

- *Agilidade* - os métodos ágeis melhoram a flexibilidade e a produtividade do desenvolvimento para DMs por prover os meios para lidar adequadamente com as mudanças de requisitos e ambiente, além de fomentar o aprendizado a partir do processo de desenvolvimento iterativo e incremental;
- *Consciência do mercado*: o processo de desenvolvimento para DMs deve ser orientado ao desenvolvimento do produto, ao invés de ser orientado ao desenvolvimento do projeto;
- *Suporte à Linha de Produto de Software (LPS)* - as empresas tendem a desenvolver uma família de produtos reutilizáveis na tentativa de reduzir os custos de desenvolvimento;
- *Desenvolvimento de arquitetura base* - o sucesso de uma LPS depende de uma plataforma comum, que necessita de uma arquitetura base bem definida;
- *Suporte ao reuso* - o suporte ao desenvolvimento baseado em componentes e camadas é essencial em uma metodologia de desenvolvimento para DMs; e
- *Especificação da arquitetura física desde as fases iniciais do desenvolvimento* - as restrições de *hardware* dos DMs devem ser consideradas desde os estágios iniciais do projeto de *software*.

As características supracitadas foram utilizadas como requisitos na concepção da metodologia alvo, a *HME*, a qual foi elaborada em iterações e cuja versão final é apresentada na Figura 3.3. Essa concepção é utilizada para a realização de uma análise preliminar, que inclui também a análise de negócio do produto, etapas que são realizadas na iniciação do projeto. Em seguida, é realizada uma análise detalhada e protótipos funcionais podem ser criados para facilitar o entendimento dos requisitos iniciais do produto. Após a conclusão desse entendimento, o projeto arquitetônico pode ser elaborado e o planejamento dos ciclos do projeto que serão executados para o desenvolvimento dos componentes pode então ser iniciado. Nessa fase, começa a execução do Motor do Desenvolvimento, que consiste no projeto, análise e desenvolvimento concorrente de componentes, na revisão da qualidade desse desenvolvimento e demais artefatos gerados, bem como na validação do incremento do produto a partir de testes de *marketing*. Ao final do processo, a biblioteca de componentes é atualizada e o produto entra na fase de comercialização.

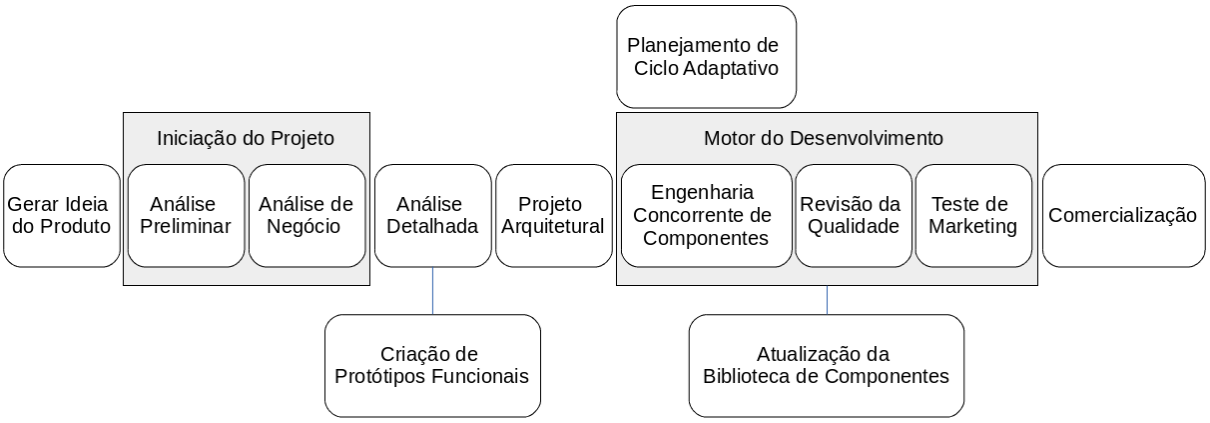


Figura 3.3: *Hybrid Method Engineering*, Fonte: (RAHIMIAN; RAMSIN, 2008)

Embora o *Mobile-D* tenha sido criticado pelos autores, o *HME* não resolveu os problemas apontados naquela abordagem, como a documentação superficial e incompleta. O trabalho de Rahimian e Ramsin (2008) focou mais no processo de concepção da metodologia, apresentando a evolução dessa concepção e o uso de métodos do *ME* e *HMD*, do que propriamente na metodologia *HME* em si. Dessa forma, poucas são as informações apresentadas nesse trabalho sobre o *HME*, bem como a respeito de sua utilização em projetos reais. O cenário das aplicações para DMs em que foi utilizada, o tamanho dos times de desenvolvimento, vantagens e desvantagens, por exemplo, não foram sequer mencionados, o que dificulta o entendimento completo da metodologia e, em consequência, o seu reuso.

3.1.3 *MASAM*

Jeong *et al.* (JEONG et al., 2008) propõem outra abordagem ágil chamada *MASAM*, baseada em arquiteturas e componentes reutilizáveis e com foco no desenvolvimento de aplicativos móveis. O *MASAM* propõe um ciclo de desenvolvimento com quatro diferentes fases: Preparação, Personalização, Desenvolvimento e Comercialização. Essas fases, suas atividades e tarefas são apresentadas na Tabela 3.1.

Tabela 3.1: Fases do Ciclo de Desenvolvimento do *MASAM*, Fonte: (JEONG et al., 2008)

Fase	Atividade	Tarefas
Preparação	Compreensão do produto	Sumário do produto e pré-planejamento
	Compartilhamento do conceito do produto	Definição do usuário e análise inicial do produto
	Setup do projeto	*Setup* do processos de desenvolvimento e dos recursos do projeto, realização de estudo inicial
Personalização	Entendimento das necessidades do usuário	*Story card workshop* e *design* da interface de usuário
	Definição e elaboração da arquitetura	Análise dos requisitos não funcionais, definição da arquitetura e identificação dos padrões de *software* a serem utilizados
Desenvolvimento	Preparação e Implementação	*Setup* do ambiente de desenvolvimento e planejamento do desenvolvimento
	Ciclos de lançamentos	Planejamento dos lançamentos, dos ciclos ou iterações e a realização propriamente dita desses lançamentos
Comercialização	Testes do sistema	Testes de aceitação e testes pelo usuário
	Comercialização do produto	Testes de lançamento e lançamento do produto

O processo definido pelo *MASAM* se inicia com a fase de preparação, a partir da atividade de entendimento do produto (compreensão do produto, veja a Tabela 3.1). Nessa atividade, uma lista dos requisitos do produto é especificada e, com base nesse entendimento inicial, é elaborado um pré-planejamento do projeto. Em seguida, o conceito do produto é compartilhado com os envolvidos para uma análise mais detalhada do mesmo e, em seguida, é realizada a definição dos usuários. Essa fase de preparação termina com o *setup* do projeto, que abrange o *setup* dos processos de desenvolvimento e o planejamento dos recursos necessários para a execução do mesmo. A fase seguinte consiste na personalização e compreende as atividades de entendimento das necessidades do usuário, por exemplo a partir de *workshops* de requisitos e elaboração de protótipos da interface do sistema, e da especificação de uma arquitetura inicial. De posse dos requisitos do produto e dessa arquitetura inicial, a fase de desenvolvimento começa com a preparação do ambiente de desenvolvimento e o planejamento dos ciclos de execução desse desenvolvimento. Esse planejamento guia as iterações e contém os marcos de lançamentos dos incrementos do produto. Após a execução do desenvolvimento, a fase final do *MASAM* consiste na realização dos testes de aceitação e do usuário, dos testes de lançamento e do lançamento propriamente dito do produto para a comercialização do mesmo.

O *MASAM* é recomendado para pequenas empresas com foco no desenvolvimento de aplicativos para DMs. Entretanto, esse trabalho de Jeong *et al.* (2008) não apresenta nenhum estudo de caso onde a implementação dessa metodologia tenha sido utilizada. Além disso, o baixo nível de detalhamento do *MASAM* compromete a análise da sua aplicabilidade e, por conseguinte, o seu reuso.

3.2 Metodologias Ágeis com a Gestão da Qualidade

Os processos de desenvolvimento de *software* se modernizaram em resposta às novas demandas, pressões por prazos menores e à necessidade de flexibilidade às mudanças. Os princípios e práticas ágeis são amplamente adotados por equipes de desenvolvimento nesse cenário e, geralmente, essas equipes compreendem que a qualidade dos processos de desenvolvimento contribui na qualidade final do produto e que todos são responsáveis por esse resultado, não somente os testadores e analistas de qualidade. Entretanto, muitas vezes não existe uma definição clara de papéis e responsabilidades, e de como cada integrante pode contribuir.

Por outro lado, metodologias como o *Six Sigma* e *Lean Six Sigma* (*LSS*) vem sendo utilizadas no desenvolvimento de *software*, contribuindo na melhoria contínua dos processos de desenvolvimento e, por conseguinte, na qualidade dos produtos desenvolvidos. Além disso, há outros benefícios no uso dessas metodologias tais como a capacidade de planejar com maior precisão a partir de dados históricos, a utilização de ferramentas estatísticas para a análise em tempo real do progresso e o auxílio dessas análises nas decisões do projeto.

O desenvolvimento de *software* para DMs possui uma necessidade constante de resolução de problemas com a finalidade de desenvolver produtos com cada vez menos defeitos. Metodologias como o *Six Sigma* e o *LSS*, dessa forma, podem ser uma alternativa para atender aos níveis elevados de qualidade demandados por esse nicho de mercado.

Nas próximas seções são apresentados os trabalhos que propõem combinações de métodos ágeis com metodologias de gestão da qualidade.

3.2.1 *eXtreme Programming* e *Six Sigma*

Em (HASHMI; BAIK, 2008), o *Six Sigma* é combinado ao *eXtreme Programming* (*XP*) a partir do mapeamento de ferramentas do *Six Sigma* às atividades envolvidas em cada fase do *XP*. Nessa combinação, o *Six Sigma* é utilizado na análise de dados, identificação e implantação de melhorias no desempenho das práticas do *XP*. O ciclo de vida do *XP* é apresentado na Figura 3.4 e suas fases são explicadas de forma simplificada a seguir.

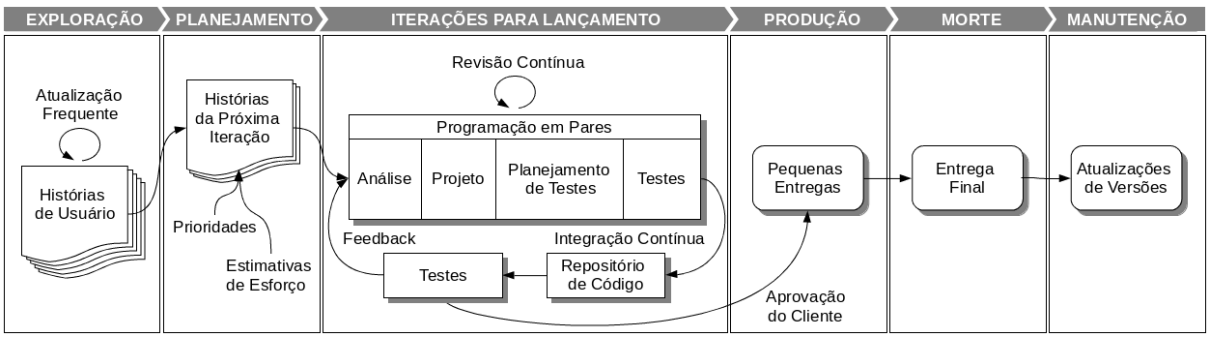


Figura 3.4: Ciclo de Vida do *eXtreme Programming*, Fonte: (WELLS, 2009).

O processo do *XP* se inicia na fase de exploração com a especificação das histórias de usuário (veja a Figura 3.4), que correspondem à documentação dos requisitos do produto. Essas histórias são frequentemente atualizadas durante todo o projeto para incorporar as mudanças e a evolução dos requisitos. Em seguida, na fase de planejamento, as histórias são estimadas e priorizadas e um conjunto das mesmas é selecionado para a próxima iteração. Na fase seguinte, as iterações são executadas a partir de atividades de análise, projeto, codificação e testes e resultam em entregas de *software* que podem ir para produção, mediante a aprovação do cliente. Esse desenvolvimento é geralmente realizado através da programação em pares e o código é continuamente integrado ao repositório de código. Durante essas iterações, o cliente participa da elicitação dos requisitos, dos testes funcionais e de aceitação e seus *feedbacks* são refletidos no desenvolvimento atual ou em iterações seguintes. Ao término dessas iterações, a versão final do produto é entregue e o projeto alcança a fase de encerramento (ou morte). A partir de então, o ciclo do *XP* inicia a fase de manutenção onde as atualizações necessárias do produto são desenvolvidas e entregues ao cliente.

Uma das principais contribuições de Hashmi e Baik (2008) é o mapeamento e a exemplificação do uso das ferramentas do *Six Sigma* combinadas às práticas do *XP*, conforme apresentado a seguir:

- *Análise de defeitos de atividades do XP* - os defeitos coletados a partir do ciclo de vida do *XP* podem ser analisados e priorizados. Por exemplo, após os testes funcionais na fase de execução das iterações para lançamento (veja a Figura 3.4), os defeitos identificados podem estar relacionados à erros na análise e projeto, codificação, no entendimento ou especificação dos requisitos e à problemas na integração do código ao repositório. Essas análises de defeitos podem ser realizadas através de ferramentas do *Six Sigma* como os diagramas de Pareto, Causa e Efeito e análises de variância, regressão e correlações. A utilização dessas ferramentas auxilia na identificação das causas raízes e os padrões de *design* e código podem ser alterados como resultado, minimizando a quantidade de novos defeitos nas iterações seguintes;

- *Planejamento de iterações com uso da produtividade e modelos de estimativas* - a medição da produtividade e elaboração de modelos de estimativas podem ser úteis em análises de prazo, riscos e custos. Durante a fase de planejamento do *XP* (veja a Figura 3.4), por exemplo, as histórias de usuários são selecionadas com base na priorização e estimativa de esforço. A quantidade de histórias de uma iteração depende, portanto, da capacidade de produção da equipe (produtividade) e dessas estimativas, informações utilizadas no planejamento das iterações. O conhecimento e a precisão do histórico de estimativas, portanto, pode ajudar a melhorar a previsão de futuras iterações, as estimativas e o planejamento do projeto;

- *Análise de correlação entre a produtividade e a duração das iterações* - no *XP*, o tamanho das iterações geralmente varia de 1 a 4 semanas e esse tamanho pode ser continuamente avaliado, por exemplo, a partir de uma análise de correlação da produtividade da equipe e a duração das iterações. Essa análise pode ajudar a identificar o tamanho de iteração mais adequado ao projeto;

- *Análise de esforço estimado versus realizado* - a realização de testes de variância para análise do esforço estimado versus o realizado pode contribuir na melhoria do modelo de estimativa e, por conseguinte, na previsão dos prazos do projeto; e
- *Análise de regressão entre os valores estimados e realizados* - essa análise pode, por exemplo, identificar uma função de regressão para avaliação e melhoria das estimativas da equipe.

Hashmi e Baik (2008) apresentam informações a partir de tabelas e gráficos que facilitam o entendimento dos exemplos supracitados no uso das ferramentas do *Six Sigma* com o *XP*. Entretanto, um dos pontos fracos do trabalho de Rahimian e Ramsin (2008) é não contextualizar os projetos em que essa abordagem foi utilizada. Não é possível, por exemplo, identificar como o *Six Sigma* foi implantado, se houve a execução de um projeto de melhoria com o *DMAIC*, qual o tamanho dos times de desenvolvimento envolvidos e se a equipe de desenvolvimento participou das análises realizadas com o uso dessas ferramentas.

3.2.2 *Scrum* e *Six Sigma*

Roriz (RORIZ, 2010) apresenta uma análise de como o *Scrum* pode ser utilizado em conjunto com o *Six Sigma* no desenvolvimento de *software*. Roriz (2010) sinaliza que a combinação do *Scrum* ao *Six Sigma* pode resultar em benefícios para ambas as metodologias, uma vez que o *Scrum* possui um forte alinhamento aos princípios do *Lean* (POPPENDIECK, 2005) e que esses princípios assim como os do *Six Sigma* estão diretamente relacionados à redução de defeitos e melhoria da qualidade.

Uma das principais contribuições de Roriz (2010) é a proposta de mapeamento dos papéis do *Scrum* aos do *Six Sigma*, que é apresentada na Figura 3.5.

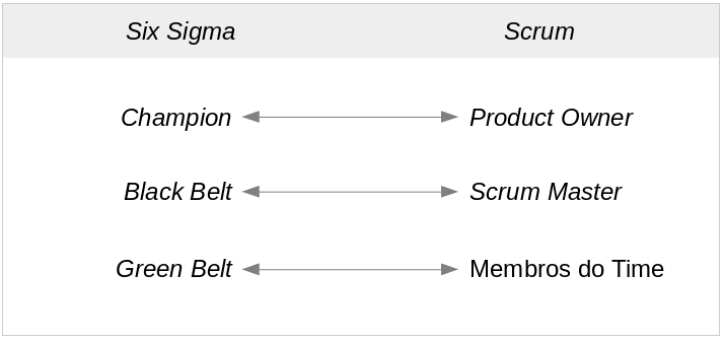


Figura 3.5: Mapeamento entre os Papéis do *Scrum* e *Six Sigma*, Fonte: (RORIZ, 2010).

Esse mapeamento sugere que, na combinação do *Scrum* com o *Six Sigma*, um mesmo profissional pode assumir papéis de ambas as metodologias, por exemplo, o *Scrum Master* pode exercer o papel do *Black Belt* do *Six Sigma*. As principais responsabilidades dos papéis do *Six Sigma* e *Scrum* mencionados no mapeamento proposto por Roriz (2010) são apresentadas na Tabela 3.2 e discutidas a seguir.

Tabela 3.2: Definição e Responsabilidades dos Papéis do *Six Sigma* e *Scrum*

	Responsabilidades
SIX SIGMA	***Champions*** • Implementar o *Six Sigma* através da organização de maneira integrada • Identificar projetos para essa implementação do *Six Sigma* • Atuar como mentor dos *Black Belts*
	Black Belts • Aplicar a metodologia em projetos específicos • Supervisionar a atuação dos *Green Belts* • Dedicar-se exclusivamente ao *Six Sigma*
	Green Belts • Participar da implementação do *Six Sigma* junto às suas outras responsabilidades no trabalho • Aplicar a metodologia em projetos específicos sob supervisão dos *Black Belts*
SCRUM	***Product Owner*** • Especificar e priorizar as funcionalidades do produto • Estabelecer uma visão compartilhada do produto entre os clientes e a equipe • Definir as datas para os lançamentos e entregas • Representar a perspectiva dos clientes
	Scrum Master • Assegurar que o time siga os princípios e práticas do Scrum • Manter o ritmo das reuniões do *Scrum*, controlando horários e garantindo que o ambiente seja favorável para que aconteçam • Monitorar o progresso do projeto • Garantir que impedimentos ao progresso sejam resolvidos • Mediar a comunicação entre o time e o exterior • Manter o time focado em cumprir o que foi prometido, impedindo que fatores externos ao time prejudiquem esse foco
	Time • Se adequar ao ritmo de trabalho necessário para finalizar um incremento previsto • Participar das reuniões do *Scrum* e defender suas ideias de estimativas e possibilidades • Auto-gerenciar suas tarefas para cumprir a previsão • Atribuir seus impedimentos ao *Scrum Master*

O *Champion* é responsável pela implementação do *Six Sigma* na organização, atua como mentor dos *Black Belts* e identifica e prioriza os projetos *Six Sigma* (veja a Tabela 3.2). Já o *Product Owner*, no *Scrum*, representa o cliente nas decisões do projeto e é responsável por

especificar e priorizar os requisitos do produto, compartilhar a visão desse produto com todos os envolvidos, além de planejar e acompanhar os lançamentos e entregas do projeto. Roriz (2010) sugere que esses papéis podem ser combinados e exercidos por um mesmo profissional, entretanto, enquanto o primeiro atua a nível organizacional, por exemplo, em relação à missão de implementar o *Six Sigma* de forma integrada em toda a organização, o segundo atua mais no nível de projetos de desenvolvimento. Além disso, um profissional que exerce o papel do *Champion* geralmente possui bastante experiência no *Six Sigma*, podendo, por exemplo, treinar e acompanhar *Black Belts* e *Green Belts* e, dessa forma, o custo de capacitação de um *Product Owner* para atuar também como *Champion* pode inviabilizar essa combinação. Se por um lado a capacitação de um *Champion* como *Product Owner* seja viável, por outro, o custo da alocação desse profissional no projeto é mais um possível ofensor à combinação desses papéis.

O *Black Belt* é responsável tanto por implementar parte dos projetos *Six Sigma* quanto por acompanhar os demais projetos liderados pelos *Green Belts*. Dessa forma, o profissional que atua como *Black Belt* é geralmente integralmente dedicado ao *Six Sigma*. Já o *Scrum Master* é focado em assegurar que a equipe siga adequadamente o *Scrum* e é responsável pela resolução de impedimentos no progresso do trabalho dessa equipe. Roriz (2010) também sugere que esses papéis podem ser combinados e exercidos por um mesmo profissional, entretanto, além dos custos de capacitação, a necessidade de alocação integral do *Black Belt* no *Six Sigma* e do *Scrum Master* no *Scrum* vão de encontro ao mapeamento proposto.

O *Green Belt* lidera projetos *Six Sigma* mas geralmente não é integralmente alocado ao *Six Sigma* e pode exercer outras atividades, por exemplo, assumir uma liderança de equipe em um projeto de desenvolvimento de *software*. O papel do *Green Belt*, assim como os demais do *Six Sigma*, requer que o profissional tenha experiência com o *DMAIC*, *DMADV* e ferramentas do *Six Sigma*. Parte dessas ferramentas demanda um bom conhecimento de estatística, essencial para que o *Green Belt*, com o auxílio dos envolvidos, controle e melhore os processos a partir da identificação de causas raízes de defeitos desses processos. Já sobre os times de desenvolvimento de *software*, o conhecimento de estatística dos integrantes desses times é geralmente suficiente para que contribuam em parte das análises do *Six Sigma*. Entretanto, esse conhecimento não é suficiente para que, sem os treinamentos necessários, liderem projetos *Six Sigma*, assumindo o papel do *Green Belt*.

No trabalho do Roriz (2010) não há sinalização de utilização da abordagem proposta (*Scrum* e *Six Sigma*) em projetos reais, uma vez que não foi apresentado nenhum estudo de caso realizado. Além disso, esse trabalho não apresenta uma fundamentação teórica nem prática para o mapeamento sugerido entre os papéis do *Scrum* e *Six Sigma*.

3.2.3 *Scrum* e *Lean Six Sigma*

Cunha *et al.* (CUNHA et al., 2011) propõem uma abordagem ágil que estabelece mecanismos para a integração do *Scrum* ao *LSS* (*SLeSS*), definida especificamente para projetos de customização de *software* para DMs como celulares e *smartphones*. O objetivo do *SLeSS* é aumentar a agilidade da gestão desses projetos a partir do uso de princípios e práticas do *Scrum*, bem como melhorar os processos de customização com o uso do *DMAIC* e ferramentas do *LSS*.

Uma visão geral do *SLeSS* é apresentada na Figura 3.6. Nessa abordagem, o *Scrum* é utilizado tanto na customização de *software* para DMs quanto na gestão das melhorias de processos. Já o *LSS* é utilizado tanto na melhoria dos processos de desenvolvimento quanto na melhoria da utilização do *Scrum* nesses projetos.

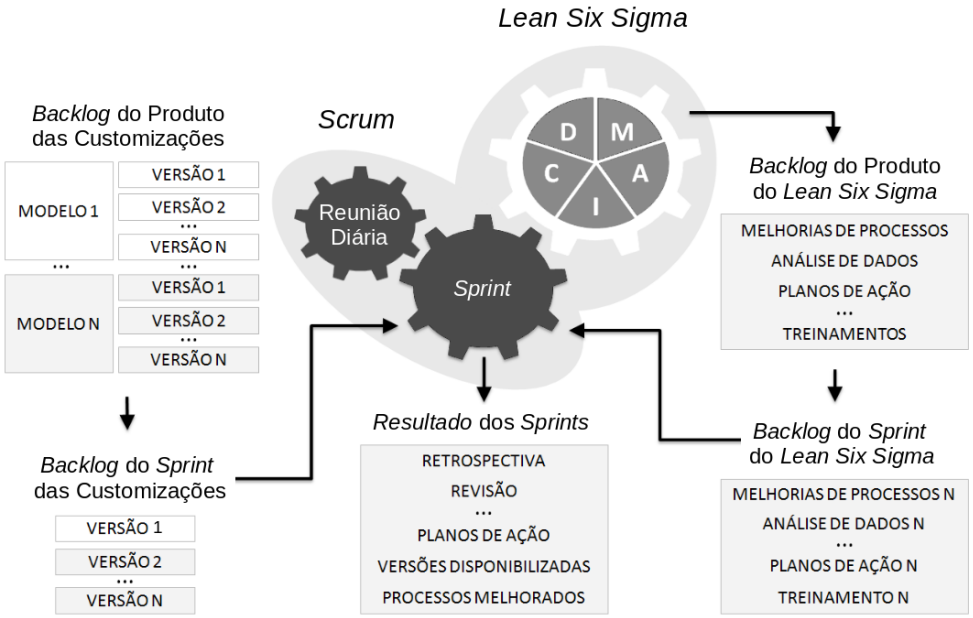


Figura 3.6: *SLeSS* - Uma Abordagem de Integração do *Scrum* e *Lean Six Sigma*, Fonte: (CUNHA et al., 2011)

O *SLeSS* define dois tipos de *backlogs*, o *Backlog* do Produto das Customizações e o do *LSS* (veja a Figura 3.6). Enquanto o primeiro é utilizado para gerenciar os requisitos das customizações de *software* dos modelos do projeto, o segundo é utilizado para gerenciar o escopo de melhorias dos processos de desenvolvimento dessas customizações. Por exemplo, o *Backlog* do Produto das Customizações apresentado na Figura 4.2 contém os requisitos das N versões do *Modelo 1*. Cada uma dessas versões possui requisitos especificados de acordo com as plataformas de *software* e *hardware* desse modelo, com informações fornecidas pelas operadoras de telefonia e que dependem, inclusive, da região onde o modelo será comercializado.

No *SLeSS*, o *Product Owner* e *Scrum Master* exercem também o papel de *Green Belt* e, além de suas responsabilidades no *Scrum*, lideram os *DMAICs* de melhoria de processos. Durante a execução dos *Sprints*, são desenvolvidos tanto os itens do *Backlog* das Customizações quanto itens do *Backlog* de melhorias de processos. O *Product Owner* é, portanto, o responsável pela priorização desses *backlogs* enquanto o *Scrum Master* auxilia o time nas análises para identificação de causas raízes de problemas nos processos de desenvolvimento.

Ao final desses *Sprints*, os resultados da Revisão e Retrospectiva são insumos para os *DMAICs* de melhoria de processos, que priorizam, a partir de análises qualitativas e quantitativas, as soluções dos problemas identificados. De acordo com Cunha *et al.* (2011), essas análises conduzem a definição de planos de ações, permitindo a melhoria contínua e o alinhamento da produtividade e qualidade às necessidades do cliente.

O *SLeSS* apresentou bons resultados na melhoria de produtividade, redução da densidade de defeitos e diminuição das horas extras nos projetos utilizados como estudo de caso. Entretanto, essa abordagem é restrita à customização de *software* para DMs, além disso, a sua documentação está incompleta, dificultando o seu reuso. Por exemplo, um dos mecanismos de integração definidos no *SLeSS* é utilizado para melhorar o uso do *Scrum* nos projetos, no entanto, não são apresentados os artefatos utilizados nem há uma documentação desse método, de forma que o seu reuso se torna inviável.

3.2.4 *MiniDMAIC*

Bezerra *et al.* (BEZERRA et al., 2007) propõem o *MiniDMAIC* como uma simplificação do *DMAIC* para tratar causas e resolução de problemas em projetos de *software*. Essa simplificação tem como objetivo reduzir a duração do *DMAIC* e, por conseguinte, o seu custo, associando-o ao tratamento de riscos dos projetos. Enquanto o *DMAIC* geralmente possui uma duração de 3 a 6 meses, segundo Bezerra *et al.* (2007), o *MiniDMAIC* pode ser executado em 1 a 6 semanas e, além disso, esse novo método requer um conhecimento básico de estatística, possibilitando que seu custo, em relação ao *DMAIC*, seja geralmente menor e, portanto, mais adequado aos projetos de *software*.

O ciclo do *MiniDMAIC* é apresentado na Figura 3.7. Esse ciclo se inicia na fase *Definir* com o entendimento do problema a ser tratado (passo 1) e consiste da definição clara do problema, de sua importância, impacto e consequências para o projeto.

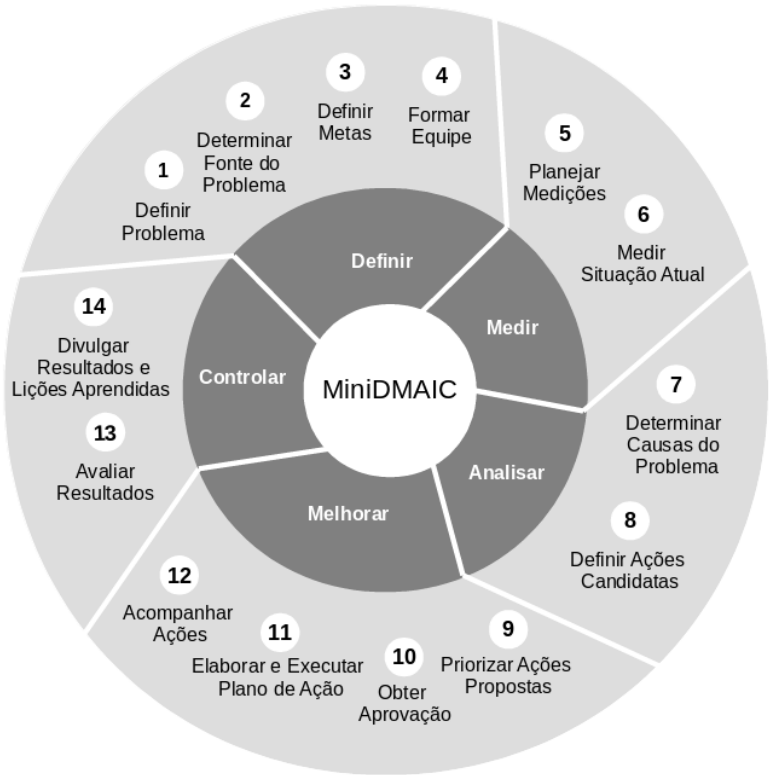


Figura 3.7: Passos do Ciclo do *MiniDMAIC*

Na etapa seguinte, a fonte de ocorrência do problema é identificada (passo 2). Por exemplo, o indicador de produtividade pode ser identificado como fonte de ocorrência do problema de desvio na produtividade. Em seguida, deve-se definir metas quantitativas de melhoria e elaborar um planejamento para a execução do *MiniDMAIC* (passo 3), contendo inclusive uma estimativa de custo-benefício. Essa fase é finalizada com a definição da equipe que será envolvida no *MiniDMAIC* (passo 4). A fase *Medir* abrange dois passos, o planejamento das medições (passo 5) e a medição da situação atual (passo 6). O primeiro consiste na análise e definição de novas medições para o problema enquanto o segundo utiliza essas novas medições para a identificação do estado atual do problema. Na fase seguinte são realizadas análises para identificação das causas do problema (passo 7) e a partir do resultado dessas análises são definidas ações candidatas para a resolução das mesmas (passo 8). A fase *Melhorar* se inicia com a priorização das ações propostas na fase anterior (passo 9). Essa priorização é realizada com base no resultado das análises das causas do problema e considera os efeitos positivos e negativos dessas ações. De posse dessa priorização, as ações serão planejadas, executadas e acompanhadas, mediante aprovação da gerência sênior do projeto (passos 10, 11 e 12). A partir de então se inicia a fase *Controlar* que consiste na avaliação dos resultados alcançados com a execução das ações (passo 13) e da divulgação desses resultados, bem como das lições aprendidas durante a execução do *MiniDMAIC* para a organização (passo 14).

O *MiniDMAIC* é um método voltado ao desenvolvimento de *software* em geral com foco no tratamento de problemas como desvios na produtividade, prazo e densidade de defeitos e sua utilização obteve bons resultados na melhoria de produtividade e na redução da densidade de defeitos em projetos reais. Embora Bezerra *et al.* (2007) não especifique se o *MiniDMAIC* foi ou não aplicado em projetos que utilizavam metodologias ágeis, não há em sua documentação nenhum indicativo de que esse método não possa ser aplicado ao desenvolvimento ágil.

3.3 Avaliação e Melhoria no Uso de Práticas Ágeis

O *Scrum* é um processo iterativo e incremental utilizado na gestão e desenvolvimento de projetos, que define uma estrutura base a partir de fundamentos, princípios, valores e práticas, e inclui orientações para a sua utilização (SCHWABER, 1995)(BEEDLE et al., 2000). A essa estrutura base são adicionadas práticas de engenharia e gestão para a instanciação do *Scrum* conforme o cenário e a necessidade de cada projeto (RUBIN, 2012).

A implementação do *Scrum*, portanto, é considerada única por projeto e sua adequada utilização depende de fatores como a experiência do time e do cliente, as competências desse time, a natureza do projeto, entre outros (COHN, 2009). Esses fatores influenciam a forma na qual o *Scrum* deve ser instanciado para que seus benefícios sejam alcançados. Não é raro, todavia, casos em que o time do projeto sugere não mais utilizar o *Scrum*, devido a perdas de qualidade e produtividade durante a implantação ou nas primeiras iterações (MOE; DINGSØYR, 2008)(BALKANSKI, 2008). Consequentemente, em alguns desses casos, a culpa de tais resultados é associada a nova metodologia adotada e não ao mau uso da mesma (CRESCÊNCIO, 2013). Nesse cenário, o time comumente volta a utilizar a abordagem anterior e uma segunda tentativa de adoção do *Scrum* se torna improvável.

O sucesso da implantação e continuidade com o *Scrum* depende, portanto, de como cada projeto instancia essa metodologia e, além disso, de como essa instanciação é ajustada conforme o projeto evolui. Existem estratégias de adoção do *Scrum* para que os resultados iniciais sejam mais assertivos, por exemplo, selecionar um projeto piloto com uma equipe pequena, que possua um patrocinador engajado e cuja importância seja a necessária para motivar os envolvidos (COHN, 2009). Entretanto, mesmo após uma implantação bem sucedida, os resultados do projeto podem evoluir lentamente ou até mesmo retroceder.

Nesse cenário, surge a necessidade de gerenciar a implementação do *Scrum*. Para a boa utilização de práticas ágeis, existem modelos de maturidade e *checklists* que possibilitam medir a forma de adoção dessas práticas e, se aplicados corretamente, servem como guia de como melhorar a agilidade nos projetos (QUMER et al., 2007)(SIDKY et al., 2007). Entretanto, por serem usualmente generalistas, esses modelos são muito extensos, focando em práticas de várias metodologias ao mesmo tempo, o que dificulta quando o projeto já se utiliza do *Scrum* e deseja inicialmente melhorar o uso dessa metodologia. Além disso, para a avaliação do *Scrum*, há a necessidade de *checklists* mais específicos, que forneçam uma visão de forças e fraquezas no uso dos princípios e práticas ágeis do *Scrum*.

Nas próximas seções são analisados os trabalhos relacionados à adoção, melhoria e avaliação de práticas ágeis, modelos de maturidade e *checklists*, bem como sobre o uso do *DMAIC* em projetos de *software*.

3.3.1 *4-Dimensional Analytical Tool*

Qumer e Henderson-Sellers (QUMER; HENDERSON-SELLERS, 2008) propõem o *4-Dimensional Analytical Tool* (*4-DAT*) como um método de análise e comparação de metodologias quanto ao nível de agilidade e adoção de princípios e práticas ágeis. O *4-DAT* estabelece quatro dimensões (Escopo, Agilidade, Valores Ágeis e Processos) e, para cada uma dessas dimensões, define um conjunto de critérios para direcionar a avaliação e comparação das metodologias.

As dimensões e critérios do *4-DAT* são apresentados na Tabela 3.3. Por exemplo, a segunda dimensão (*2-D*) é utilizada para avaliar e comparar a agilidade das metodologias. Essa dimensão especifica cinco critérios: a Flexibilidade, que verifica se a metodologia se adapta adequadamente às mudanças; a Velocidade, que avalia se a metodologia produz resultados rapidamente; o *Leanness*, critério utilizado para verificar se a metodologia foca na produção enxuta, ou seja, no desenvolvimento no menor tempo e com otimização dos recursos, e a partir do uso sistemático de técnicas para a melhoria contínua da qualidade; o Aprendizado, que avalia se utiliza conhecimento e experiência atualizados para o aprendizado contínuo; e o Tempo de Resposta, que verifica se a metodologia possui um tempo de resposta adequado em relação às alterações nas principais entradas desse método e eventuais mudanças.

Em (QUMER, 2006), o *4-DAT* é utilizado para comparar as fases e práticas do *Scrum* e *XP*. Como exemplo de um dos resultados dessa comparação, o cálculo de agilidade do *Scrum*, a partir da segunda dimensão do *4-DAT* é apresentado na Tabela 3.4. Para realizar esse

Tabela 3.3: Dimensões e Critérios do *4-Dimensional Analytical Tool*, Fonte: (QUMER; HENDERSON-SELLERS, 2008)

Dimensão	**Critério**
1-D Escopo	*Tamanho do Projeto* - fornece suporte para projetos pequenos, médios e grandes?
	Tamanho do Time - fornece suporte para equipes pequenas e grandes (única equipe ou múltiplas)?
	Estilo de Desenvolvimento - quais estilos (iterativo, incremental, rápido) são utilizados?
	Estilo de Código - especifica o estilo de codificação (simples ou complexo)?
	Ambiente Tecnológico - especifica quais tecnologias para o ambiente de desenvolvimento?
	Ambiente Físico - fornece suporte a quais tipos de ambiente (co-localizado ou distribuído)?
	Cultura - que tipo de cultura (colaborativa, cooperativa ou não-cooperativa) o método especifica?
	Mecanismo de Abstração - especifica o uso de orientação a objetos, orientação a aspectos?
2-D Agilidade	*Flexibilidade* - o método acomoda mudanças esperadas ou inesperadas?
	Velocidade - produz resultados rapidamente?
	Leanness - foca na produção enxuta com técnicas para a melhoria da qualidade dessa produção?
	Aprendizado - se utiliza de experiência e conhecimento atualizados para o aprendizado?
	Tempo de Resposta - o método possui tempo de resposta adequado?
3-D Valores Ágeis	*Indivíduos mais que processos e ferramentas*
	Software mais que documentação
	Colaboração com o cliente mais que contratos
	Responder a mudanças mais que seguir um plano
	Manter a agilidade do processo
	Manter o processo com custo efetivo
4-D Processos	*Desenvolvimento* - quais as práticas utilizadas nos ciclos de desenvolvimento e testes?
	Gestão de Projetos - quais as práticas de gestão de projeto utilizadas pelo método?
	Configuração, Controle e Suporte - quais as práticas de configuração, controle e suporte?
	Gestão de Processos - quais são gestão dos processos envolvidos?

cálculo, as fases e práticas do *Scrum* são avaliadas quanto aos critérios de agilidade definidos no *4-DAT*: Flexibilidade (Fx), Velocidade (Vc), *Leanness* (Ls), Aprendizado (Ap) e Tempo de Resposta (Tr). Nessa avaliação, é atribuído o valor 1 (um) se a fase ou prática fornece suporte ao critério de agilidade e o valor 0 (zero), em caso contrário. Como exemplo, de acordo com Qumer (2006), a fase de Encerramento (Pós-Jogo) do *Scrum* fornece suporte apenas à Velocidade (Vc) enquanto a prática de Revisão do *Sprint* atende a todos os critérios com exceção do *Leanness* (Ls). No *4-DAT*, os níveis de agilidade são calculados para as fases e práticas, de forma separada, e esse cálculo é realizado a partir da média aritmética dos totais de cada critério. Nessa avaliação do *Scrum* (veja a Tabela 3.4), os níveis de agilidade calculados são 0,65 (13/20) para as fases e 0,80 (48/60) para as práticas dessa metodologia.

De acordo com a avaliação de Qumer (2006) sobre o *Scrum* a partir da utilização do método *4-DAT*, é possível sinalizar que o nível de agilidade do *Scrum* poderia aumentar se suas fases e práticas atendessem ao critério *Leanness* (veja a coluna *Leanness* (Ls) da Tabela 3.4). Esse critério está relacionado à produção enxuta e ao uso sistemático de técnicas para a melhoria contínua da qualidade. Dessa forma, é possível concluir que há uma lacuna de

Tabela 3.4: Cálculo de Agilidade do *Scrum* com Uso do *4-Dimensional Analytical Tool*, Fonte: (QUMER, 2006)

	Critérios de Agilidade					
Fases	**Fx**	**Vc**	**Ls**	**Ap**	**Tr**	**Total**
Planejamento (Pré-jogo)	1	1	0	1	1	4
Preparação (Pré-jogo)	1	1	0	1	1	4
Desenvolvimento (Jogo)	1	1	0	1	1	4
Encerramento (Pós-jogo)	0	1	0	0	0	1
Nível de Agilidade	3/4	4/4	0/4	3/4	3/4	13/20
Práticas	**Fx**	**Vc**	**Ls**	**Ap**	**Tr**	**Total**
Product Owner	1	1	0	1	1	4
Scrum Master	1	1	0	1	1	4
Times do *Scrum*	1	1	0	1	1	4
Backlog do Produto	1	1	0	1	1	4
Backlog do *Sprint*	1	1	0	1	1	4
Plano de Lançamentos	1	1	0	1	1	4
Gráfico de *Burndown*	1	1	0	1	1	4
Planejamento do *Sprint*	1	1	0	1	1	4
Sprint	1	1	0	1	1	4
Reunião Diária	1	1	0	1	1	4
Revisão do *Sprint*	1	1	0	1	1	4
Retrospectiva do *Sprint*	1	1	0	1	1	4
Nível de Agilidade	12/12	12/12	0/12	12/12	12/12	48/60

agilidade no *Scrum* e essa lacuna poderia ser reduzida, por exemplo, a partir da combinação dessa metodologia com métodos voltados à produção enxuta, como por exemplo, o *LSS*.

O *4-DAT* é um método para avaliação e comparação de metodologias e não se propõe, sozinho, a avaliar a forma ou nível do uso dessas metodologias nos projetos de desenvolvimento. Por exemplo, um projeto que adota o *Scrum* e precisa identificar em que nível se encontra, quanto ao uso de princípios e práticas dessa metodologia, não consegue avaliar esse nível utilizando apenas o *4-DAT*.

3.3.2 *Agile Adoption Framework*

Sidky *et al.* (2007) propõem o *Agile Adoption Framework* (*AAF*) como um processo para adoção de práticas ágeis em projeto de *software*. Uma visão geral desse método e seus componentes é apresentada na Figura 3.8. O *AAF* é composto de dois componentes, o Índice de Medição Ágil, que identifica o potencial ágil alcançável por projeto e organização; e o Processo de 4-Estágios, utilizado para avaliar se o projeto e a organização estão aptos a avançar na adoção de práticas ágeis e a identificar quais práticas devem ser adotadas.

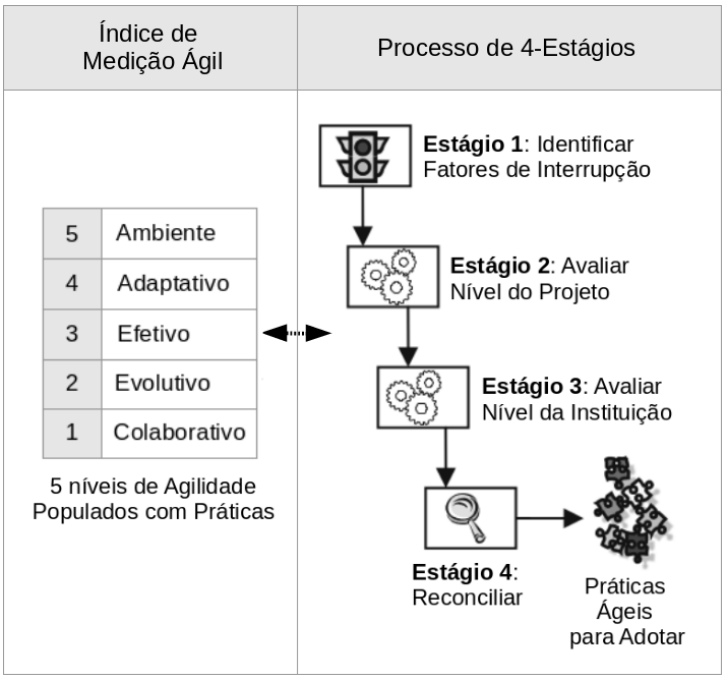


Figura 3.8: Visão Geral do *Agile Adoption Framework*, Fonte: (SIDKY et al., 2007)

O *AAF* estabelece cinco níveis de agilidade, populados com práticas ágeis agrupadas por princípios ágeis. O mapeamento desses níveis às práticas ágeis é utilizado para classificar a agilidade dos projetos a partir da verificação do uso dessas práticas pelos projetos. Esse mapeamento guia a evolução da agilidade dos projetos, à medida que indica quais práticas ágeis podem ser adotadas para o atendimento dos níveis seguintes do *AAF*. Os níveis definidos no *AAF* são, do 1 ao 5, respectivamente (veja Figura 3.8): Colaborativo, promove a comunicação e colaboração entre os envolvidos do projeto; Evolutivo, foca na entrega contínua de *software*; Efetivo, objetiva o desenvolvimento de *software* testado e de alta qualidade, realizado de forma eficiente e efetiva; Adaptativo, estabelece as práticas que atuam para agilizar as respostas às mudanças; e Ambiente, que promove um ambiente altamente produtivo.

O Processo de 4-Estágios utiliza o Índice de Medição Ágil na avaliação de agilidade dos projetos e está subdivido em quatro estágios, os quais são do 1 ao 4, respectivamente (veja Figura 3.8):

- *Identificar fatores de interrupção* - são identificados os fatores que dificultam a adoção das práticas ágeis como, por exemplo, a ausência de apoio por parte da gestão executiva da organização, falta dos recursos necessários para a adoção das práticas ou mesmo a identificação de quais práticas não são adequadas ao projeto.
- *Avaliar nível do projeto* - consiste na avaliação da forma do uso das práticas ágeis no projeto e, a partir dessa avaliação, o Índice de Medição Ágil é utilizado para a classificação do nível desse projeto.
- *Avaliar nível da organização* - uma vez que os projetos foram avaliados, a avaliação da organização consiste na consolidação do resultado desses projetos. O Índice de Medição Ágil também é utilizado para a classificação do nível da organização.

- *Reconciliar* - nesse estágio são identificadas as práticas ágeis que devem ser adotadas para o atendimento dos próximos níveis do *AAF*.

Para a avaliação dos projetos e da organização, o *AAF* utiliza aproximadamente 300 indicadores, que são itens do tipo escala de classificação (uma única classificação é selecionada para a questão, entre uma sequência gradual de classificações possíveis). Esses indicadores são agrupados por público alvo (desenvolvedores, gerentes e avaliadores da qualidade) e, por observação e entrevistas, são classificados para medir o atendimento das características relacionadas aos princípios e práticas ágeis. Se por um lado, essa quantidade de indicadores torna o método abrangente, por outro lado, o torna extenso, demandando esforço significativo em sua utilização.

3.3.3 *Nokia Test*

Para a análise e comparação dos *checklists* de avaliação do *Scrum*, as categorias e subcategorias apresentadas na Tabela 3.5 foram identificadas e são utilizadas na classificação dos itens dos *checklists*. Essas categorias e subcategorias abrangem os fundamentos, valores, princípios, papéis, artefatos e eventos do *Scrum* e possibilitam a comparação de itens com certa relação (categoria ou subcategoria) em um único *checklist* bem como a comparação de itens entre *checklists* distintos.

Vode e Sutherland (VODE; SUTHERLAND, 2008) propõem o *Nokia Test* para a avaliação do uso do *Scrum* em projetos de *software*. Esse *checklist* foi elaborado em 2005 e inicialmente focava no desenvolvimento ágil. Em 2007, ele foi ajustado para avaliar as práticas do *Scrum* e, em 2008, Jeff Sutherland, cocriador do *Scrum*, estendeu o mesmo de forma a fornecer uma pontuação do uso do *Scrum*. O *Nokia Test* possui 50 itens e foca na avaliação do uso de artefatos e papéis do *Scrum*.

A distribuição percentual dos itens do *Nokia Test* em relação às categorias identificadas é apresentada na Figura 3.9 e a análise dessa distribuição é apresentada a seguir:

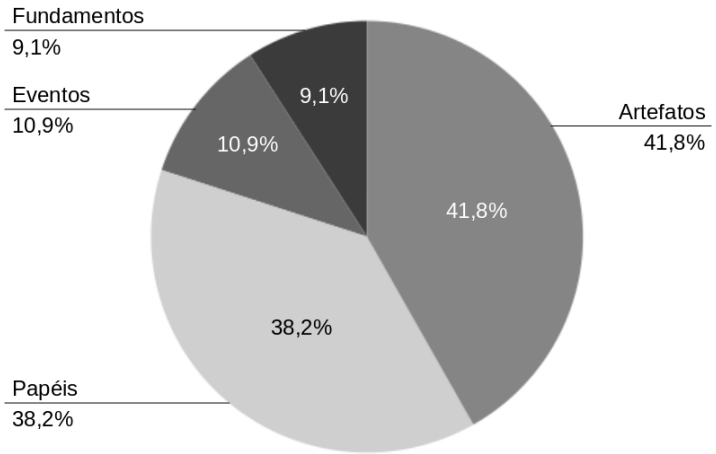


Figura 3.9: Distribuição Percentual de Itens do *Nokia Test* por Categoria

Tabela 3.5: Categorias e Subcategorias Utilizadas na Análise dos *Checklists* do *Scrum*

Categorias	**Subcategorias**
Fundamentos	Transparência
	Inspeção
	Adaptação
Valores e Princípios	Indivíduos mais que processos e ferramentas
	Software mais que documentação
	Colaboração com o cliente mais que contratos
	Responder a mudanças mais que seguir um plano
	Manter a agilidade do processo
Papéis	*Product Owner*
	Scrum Master
	Time
Artefatos	*Backlog* do Produto
	Backlog do Sprint
	Plano de Lançamentos
	Gráfico de *Burndown*
Eventos	Planejamento do *Sprint*
	Sprint
	Reunião Diária
	Revisão do *Sprint*
	Retrospectiva

- *Fundamentos* - 9,1% dos itens focam na avaliação da utilização dos fundamentos do *Scrum* e todos são relacionados à inspeção. Por exemplo, há itens sobre a periodicidade das inspeções, os tipos de testes realizados e inclusive se há a alocação de um analista de qualidade ao projeto.

- *Papéis* - 38,2% dos itens avaliam os papéis do *Scrum*, priorizando o *Product Owner* e o Time. Há itens que avaliam a experiência, competência, como são realizadas as estimativas e o planejamento, a interação entre esses papéis e também se o Time é interrompido por agentes externos à equipe do projeto.

- *Artefatos* - 41,8% dos itens avaliam como os principais artefatos são utilizados, qual a periodicidade de uso, atualização e qualidade dos mesmos. Nesse cenário, são avaliados como são especificados os requisitos do projeto, como o *Backlog* do Produto é estimado e sobre a utilização do Gráfico de *Burndown*.

- *Eventos* - 10,9% dos itens avaliam os eventos do *Scrum* e o único evento diretamente avaliado é o *Sprint*.

Dessa forma, por esse *checklist* não avaliar parte dos princípios e valores do *Scrum* e, sobre os eventos dessa metodologia, avaliar somente o *Sprint*, bem como por não avaliar o papel do *Scrum Master*, o *Nokia Test* pode ser considerado pouco abrangente em relação à cobertura dos princípios e práticas dessa metodologia.

3.3.4 *CRISP Scrum-Checklist*

Kniberg (KNIBERG, 2009) propõe o *CRISP Scrum-Checklist* como guia para os primeiros passos com o *Scrum* e para avaliar a implementação dessa metodologia nos projetos. Esse *checklist* foi elaborado em 2008 e tem como principal grupo-alvo as equipes iniciantes no *Scrum*. O *CRISP Scrum-Checklist* possui 78 itens e foca na avaliação de eventos e papéis. Esses itens são organizados por seções, cada uma associada implicitamente a uma prioridade.

A distribuição percentual de itens do *CRISP Scrum-Checklist* em relação às categorias identificadas é apresentada na Figura 3.10 e a análise dessa distribuição é apresentada a seguir:

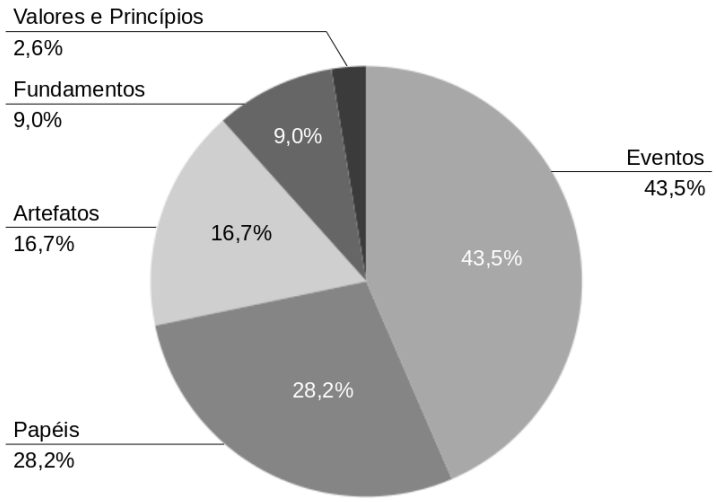


Figura 3.10: Distribuição Percentual de Itens do *CRISP Scrum-Checklist* por Categoria

- *Fundamentos* - 9% dos itens avaliam a utilização dos fundamentos do *Scrum* com foco na transparência e adaptação. Há itens a respeito da melhoria contínua do processo, da visibilidade dos artefatos e da utilização da definição de *software* pronto.
- *Valores e Princípios* - 2,6% dos itens avaliam a utilização dos valores e princípios do *Scrum*. São basicamente itens que verificam o foco na colaboração com o cliente e a entrega de *software* conforme a necessidade do negócio desse cliente.
- *Papéis* - 28,2% dos itens avaliam todos os papéis do *Scrum*. Há itens sobre a experiência, competência, a interação entre os papéis, se há interferências externas ao time, se o *Product Owner* tem a autoridade adequada e até mesmo itens relacionadas ao escalonamento de times do *Scrum*.
- *Artefatos* - 16,7% dos itens avaliam a utilização dos principais artefatos. O *checklist* abrange diretamente os seguintes artefatos: *Backlog* do Produto, *Backlog* do *Sprint*, Gráfico de *Burndown* e Plano de Lançamentos. São avaliados como esses artefatos são utilizados, a qualidade dos mesmos, como são estimados e utilizados no planejamento e acompanhamento.

- *Eventos* - 43,5% dos itens avaliam os eventos do *Scrum*. Há itens sobre praticamente todos os principais eventos: Planejamento do *Sprint*, *Sprint*, Reunião Diária, Revisão do *Sprint* e Retrospectiva. Há também itens que avaliam a utilização desses eventos, duração e participantes.

Ao contrário do *Nokia Test*, o *CRISP Scrum-Checklist* é bastante abrangente em relação aos princípios e práticas do *Scrum*. Entretanto, por incluir a avaliação de práticas recomendadas mas não obrigatórias ele se torna desnecessariamente extenso.

3.4 Análise Comparativa

O comparativo dos principais trabalhos relacionados elencados ao longo deste capítulo é apresentado na Tabela 3.6. Esses trabalhos são comparados quanto à utilização no desenvolvimento e customização de *software* para DMs, com relação às metodologias ágeis e de gestão da qualidade utilizadas, quanto a possibilidade do reuso sistemático das abordagens propostas nesses trabalhos e se há algum estudo de caso apresentado pelos mesmos. Esses critérios de comparação estão diretamente relacionados ao tema desta dissertação.

Tabela 3.6: Visão Geral dos Principais Trabalhos Relacionados

Trabalhos Relacionados	Desenvolvimento de *SW* para DMs	Customização de *SW* para DMs	Metodologia Ágil	Gestão da Qualidade	Reuso Sistemático	Estudo de Caso
Mobile-D, Abrahamson *et al.* (2004)	■	□	■	□	□	■
HME, Rahimian e Ramsin (2008)	■	□	■	□	□	□
MASAM, JEONG *et al.* (2008)	■	□	■	□	□	□
XP e *Six Sigma*, Hashmi e Baik (2008)	□	□	■	■	□	◧
Scrum e *Six Sigma*, Roriz (2010)	□	□	■	■	□	□
SLeSS, Cunha *et al.* (2011)	□	■	■	■	□	■
MiniDMAIC, Bezerra *et al.* (2007)	■	□	□	■	■	■

■ Atende ◧ Atende parcialmente □ Não atende

O *Mobile-D* é uma abordagem ágil proposta para o desenvolvimento de *software* para DMs que, embora não adote uma metodologia de gestão da qualidade (veja a Tabela 3.6), há indícios em sua documentação de que ela utiliza métricas para a avaliação do processo ágil. Um de seus pontos positivos é o reuso de princípios e práticas do *XP* e de metodologias *Crystal*. As metodologias *Crystal* são voltadas à gestão de pessoas, enquanto as práticas do *XP* focam mais no desenvolvimento de *software* propriamente dito. Um dos pontos fracos do *Mobile-D* é a sua documentação, pois está incompleta e superficial, dificultando a análise e o reuso dessa abordagem.

As abordagens *HME* e *MASAM* também são ágeis e com foco no desenvolvimento de *software* para DMs (veja a Tabela 3.6). Um ponto positivo comum a esses métodos é que seus processos se iniciam na concepção e vão até a etapa de comercialização do produto, sendo, portanto, abrangentes em relação ao ciclo de vida do produto. Além disso, o *HME* inclui uma etapa para a realização da análise de negócio do produto. Essas abordagens, entretanto, não utilizam uma metodologia de gestão da qualidade, embora o *HME* inclua em seu processo uma etapa para a revisão da qualidade do produto.

Sobre a combinação do *Six Sigma* e *XP* (veja a Tabela 3.6), Hashmi e Baik (2008) sinalizam que a elaboração de mapas de processos com o uso do *Six Sigma* tem, aparentemente, menos aplicabilidade ao *XP*, visto não haver processos complexos a serem mapeados no caso dos métodos ágeis. Essa conclusão deve ser compreendida do ponto de vista de que esses autores estão focados no uso do *Six Sigma* para a melhoria da aplicação do *XP* nos projetos. Uma das principais contribuições de Hashmi e Baik (2008) é o mapeamento e a exemplificação do uso de ferramentas do *Six Sigma* às atividades do *XP*. Esse mapeamento é considerado na evolução da abordagem proposta nesta dissertação.

Roriz (2010) apresenta uma análise de como o *Scrum* pode ser utilizado em conjunto com o *Six Sigma*. Essa análise foca no mapeamento dos papéis dessas metodologias. No entanto, esse mapeamento proposto por Roriz (2010) vai de encontro ao nível de atuação organizacional desses papéis e à definição das responsabilidades dos mesmos. Além disso, Roriz (2010) não justifica o mapeamento proposto através de um embasamento teórico e prático.

O *SLeSS* é uma abordagem ágil que estabelece mecanismos para a integração do *Scrum* e do *LSS* e um desses mecanismos é voltado à aplicação de princípios e técnicas do *LSS* para a avaliação e melhoria do uso do *Scrum*. Entretanto, devido à ausência de sua documentação e por utilizar o *DMAIC* sem simplificações, esse mecanismo do *SLeSS* sobre a avaliação e melhoria do *Scrum* se torna extenso e de difícil reuso. Além disso, o *SLeSS* é restrito à customização de *software* para DMs e sua documentação como um todo também está incompleta.

Já o *MiniDMAIC* é um método de gestão da qualidade voltado ao desenvolvimento de *software* em geral e que foi aplicado em projetos de desenvolvimento de *software* para DMs. A utilização do *MiniDMAIC* nesses projetos apresentou resultados de melhoria de produtividade e redução de defeitos. Portanto, frente a esses resultados e benefícios como a redução do tempo do ciclo e custos envolvidos em projetos de melhoria, esse método é considerado na evolução da abordagem proposta nesta dissertação.

Um resumo dos pontos positivos e de melhoria dos outros trabalhos elencados ao longo deste capítulo e relacionados à avaliação e melhoria do processo ágil são apresentados na Tabela 3.7.

O *4-DAT* é um método para análise e comparação de metodologias ágeis que pode ser utilizado, por exemplo, para uma análise da agilidade do *Scrum*. Entretanto, esse método não se propõe a avaliar a forma do uso dessas metodologias ágeis em projetos de *software*. Por exemplo, somente com o *4-DAT* não é possível avaliar um projeto quanto ao uso do *Scrum*.

Tabela 3.7: Trabalhos Relacionados à Avaliação e Melhoria do Processo Ágil

Outros Trabalhos Relacionados	Pontos Positivos	Pontos de Melhoria
4-DAT	• Possibilita a análise e comparação de metodologias ágeis • Fornece o cálculo de agilidade	• Sozinho, esse método não pode ser utilizado para avaliação e melhoria do uso de práticas ágeis em projetos de *software*
AAF	• Avaliação e melhoria de práticas ágeis em projetos de *software* e na organização	• Utiliza cerca de 300 indicadores, o que torna esse método extenso
Nokia Test	• Avaliação da forma de utilização do *Scrum* em projetos de *software* • Fornece o cálculo do nível do *Scrum*	• Pouco abrangente, não avalia parte dos princípios e valores, eventos e papéis do *Scrum*
CRISP Checklist	• Avaliação da forma de utilização do *Scrum* em projetos de *software*	• Não fornece o cálculo do nível do *Scrum* • Inclui a avaliação de práticas recomendadas (não obrigatórias) o que o torna desnecessariamente extenso

Já o *AAF* pode sim ser utilizado para avaliar o *Scrum*, embora não seja específico para ele. O *AAF* é um método ágil que permite a avaliação e melhoria de práticas ágeis em projetos de *software*. Um ponto a ser considerado nesse método é que, por ser bastante genérico e abrangente, ele também é extenso e demanda esforço significativo em sua aplicação.

O *Nokia Test* e o *CRISP Scrum-Checklist* são voltados à avaliação do uso do *Scrum* em projetos de *software*. Enquanto o primeiro é pouco abrangente, o segundo é bastante abrangente e chega inclusive a ser desnecessariamente extenso. Esses *checklists* são estudados neste trabalho e um novo *checklist* é proposto para maximizar os pontos fortes desses checklists e solucionar os pontos fracos dos mesmos.

3.5 Considerações Finais

Este capítulo apresentou os principais trabalhos relacionados ao desenvolvimento e customização de *software* para DMs, ao uso de metodologias ágeis e de gestão da qualidade, bem como os trabalhos sobre a adoção, avaliação e melhoria do uso de princípios e práticas ágeis em projetos de *software*. As abordagens avaliadas possuem, em sua maioria, uma documentação insuficiente para que o reuso sistemático das mesmas seja considerado e parte delas também não apresenta exemplos e resultados de sua utilização em projetos reais, o que impede a análise mais detalhada dessas abordagens.

Diante das informações apresentadas neste capítulo, fica mais claro que existe o espaço para que metodologias de gestão da qualidade, como o *Six Sigma* e o *LSS*, sejam incorporadas ao processo ágil de desenvolvimento para DMs, fazendo parte do dia a dia dos times de desenvolvimento.

Considerando isso e o cenário do desenvolvimento e customização de *software* para DMs, é essencial que a gestão da qualidade seja sistematicamente utilizada com foco na entrega de valor ao cliente e aos usuários finais das soluções de *software* desenvolvidas.

Dessa forma, este trabalho de dissertação visa propor uma evolução da abordagem *SLeSS* que solucione as lacunas deixadas pelos trabalhos elencados neste capítulo, focando nos seguintes pontos:

- ✓ Melhoria da adoção e aplicação dos princípios e práticas ágeis aos processos de desenvolvimento e customização de *software*;
- ✓ Adequação da abordagem para o seu reuso sistemático, detalhando os mecanismos de integração, os papéis, o processo de execução, um guia de implantação, exemplos de uso em projetos reais e a discussão dos resultados da aplicação dessa abordagem nesses projetos; e
- ✓ Adequação da abordagem ao desenvolvimento, além da customização, de *software* para DMs.

O Apêndice A apresenta um resumo da avaliação dos trabalhos relacionados elencados neste capítulo.

O capítulo seguinte descreve a nova versão da abordagem proposta nesta dissertação, apresentando a sua definição, os papéis, o processo de execução e o um guia de sua implantação.

4 APRESENTANDO O *SLESS* 2.0

Este capítulo apresenta o *SLeSS* 2.0, que é a nova versão da abordagem de integração entre o *Scrum* e o *Lean Six Sigma*. Essa nova versão é proposta nesta dissertação para o desenvolvimento e customização de *software* para DMs.

A organização das seções deste capítulo é apresentada na Figura 4.1. Inicialmente, a Seção 4.1 descreve uma breve história sobre o *SLeSS*, apresentando o contexto da concepção dessa abordagem, as instituições envolvidas, as características dos projetos utilizados no primeiro estudo de caso e principais acontecimentos relacionados ao *SLeSS*. A Seção 4.2 descreve a abordagem, denominada *SLeSS* 2.0, os papéis, o seu processo de execução, os mecanismos propostos para a integração do *Scrum* e *LSS*, bem como o fluxo de implantação dessa abordagem. Além disso, essa seção apresenta também o *Agile DMAIC*, que é um método proposto neste trabalho para evolução de um dos mecanismos de integração da versão inicial do *SLeSS*. A Seção 4.3 apresenta um comparativo entre a primeira versão do *SLeSS* e a proposta nesta dissertação, identificando as principais mudanças na nova versão. Por fim, a Seção 4.4 apresenta as considerações finais sobre o capítulo.

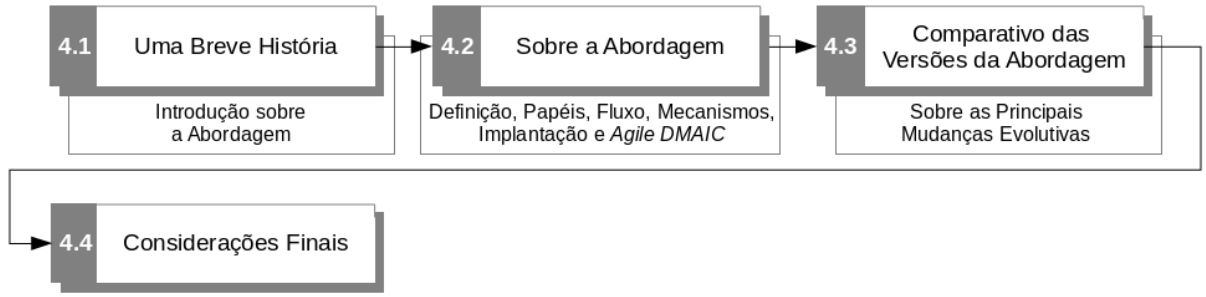


Figura 4.1: Organização das Seções do Capítulo 4

4.1 Uma Breve História

Conforme apresentado no Capítulo 3, o *Scrum Lean Six Sigma* (*SLeSS*) é uma abordagem ágil voltada à gestão de projetos, proposta inicialmente para customização de *software* para dispositivos móveis (DMs) como celulares e *smartphones*. A concepção dessa abordagem foi iniciada em 2009 em resposta à necessidade de melhoria da produtividade e qualidade de projetos desenvolvidos em laboratórios de P&D&I do Grupo de Redes de Computadores, Engenharia de *Software* e Sistemas (*GREat*[1]), que é vinculado ao Departamento de Computação (DC) da Universidade Federal do Ceará (UFC). O *GREat* desenvolve pesquisa e realiza projetos em parceria com empresas e Instituições de Ensino Superior (IES).

Os projetos utilizados como estudo de caso da primeira versão do *SLeSS* faziam parte de um programa de projetos de *outsourcing* desenvolvidos para um dos maiores fabricantes mundiais de DMs. Em 2008, a partir da parceria entre o *GREat*/DC/UFC, Atlântico[2] e o

[1] http://great.ufc.br/
[2] http://www.atlantico.com.br/

fabricante, esse programa foi iniciado com o objetivo de realizar a pesquisa e desenvolvimento para modelos de celulares e *smartphones* desse fabricante.

Cada projeto focava na customização de *software* para um conjunto de modelos de celular ou *smartphone*, possuía uma duração de 4 a 6 meses e uma equipe integralmente alocada de 7 a 12 profissionais, formada por profissionais e pesquisadores da computação e engenharia. Na customização de *software*, os times utilizavam a linguagem de programação *C ANSI* e um ambiente de desenvolvimento apropriado às plataformas de *software* e *hardware* do modelo, com ferramentas proprietárias do fabricante, em sua maioria. Além disso, grande parte dos processos de customização, padrões de código, ferramentas de revisão de código, testes automáticos e gestão de configuração eram definidos pelo próprio fabricante, que restringia a forma de desenvolvimento do *software* para manter uma interface adequada aos seus processos.

Antes da concepção do *SLeSS*, a gestão desses projetos era realizada a partir de processos do *Project Management Body of Knowledge* (*PMBoK*) (INSTITUTE, 2008) e *RUP*. Nesse contexto, os planos dos projetos como, por exemplo, os cronogramas, planos de comunicação e de recursos eram elaborados pelo gestor do programa em conjunto com as lideranças técnicas. Entretanto, as frequentes mudanças nos requisitos das customizações, prazos e também nas próprias equipes demandavam um esforço significativo na manutenção desses planos que, por muitas vezes, ficavam defasados. Além disso, a rotatividade dos integrantes das equipes era alta e não raro os novatos precisavam passar por semanas de treinamentos antes do ingresso efetivo no desenvolvimento. Havia, portanto, uma dificuldade crescente em manter os planos dos projetos e raramente se tinha uma boa visibilidade do progresso desses projetos. Frente à necessidade de gerenciar de forma mais adequada o programa como um todo, a equipe de gestão decidiu avaliar a utilização de metodologias ágeis.

Diante da análise de abordagens ágeis utilizadas nesse desenvolvimento (veja os trabalhos relacionados elencados no Capítulo 3), da boa experiência do Atlântico (diretamente envolvido na equipe de gestão do programa) no uso de métodos ágeis como o *Scrum* e o *XP*, bem como frente às restrições impostas pelo fabricante na forma como o *software* deveria ser desenvolvido, o *Scrum* foi selecionado e implantado em um projeto piloto do programa. A partir desse momento, a adoção do *Scrum* no programa foi assessorada por especialistas do Atlântico e praticamente toda a equipe de gestão recebeu treinamentos nessa metodologia.

Entretanto, mesmo após a adoção do *Scrum*, fatores como a rotatividade da equipe e processos complexos de desenvolvimento, alguns sem documentação e outros bastante trabalhosos (como o processo de disponibilização, definido pelo próprio fabricante), contribuíam para que a produtividade e qualidade dos projetos estivessem abaixo dos níveis exigidos.

Dessa forma, métodos para a documentação e melhoria dos processos foram avaliados, pois com uma documentação adequada dos principais processos, os novatos poderiam ser treinados mais rapidamente e, por outro lado, esses processos poderiam ser melhorados de forma a evitar os erros que estavam impactando nos resultados dos projetos. A partir dessa necessidade foi realizada a avaliação da adequação de metodologias como o *Six Sigma* e *Lean Six Sigma* (*LSS*) ao programa e, dessa forma, o *Scrum* e *LSS* começaram a ser utilizados em

conjunto, dando início a uma experimentação empírica que resultou na proposta da primeira versão do *SLeSS*.

Com a adoção do *SLeSS* no programa, os principais processos de customização foram documentados e controlados, possibilitando identificar as causas de desvios de produtividade e qualidade, a aplicação contínua de melhorias e a remoção dos principais gargalos do desenvolvimento. Com a utilização dessa abordagem foram obtidos bons resultados na melhoria da produtividade, diminuição da densidade de defeitos e redução da quantidade de horas extras mensais dos projetos.

Em 2011, os primeiros resultados alcançados com a utilização do *Scrum* e do *LSS* na customização de *software* para DMs foram publicados no XXV Simpósio Brasileiro de Engenharia de *Software* (SBES). O termo *SLeSS* foi introduzido pela primeira vez a partir dessa publicação, que apresentou a proposta inicial da abordagem e documentou a sua aplicação em projetos reais (CUNHA et al., 2011).

A partir dessa versão, novos estudos foram realizados para propor uma evolução da abordagem, focando na sua generalização para o desenvolvimento de *software* além da customização de software para DMs, na revisão de sua documentação, do fluxo de implantação e dos mecanismos de integração inicialmente propostos, bem como no seu aprimoramento quanto ao uso de princípios e práticas ágeis. Também como resultado desses estudos, o método *Agile DMAIC* foi proposto para melhorar um dos mecanismos de integração da abordagem. Essa evolução da abordagem é denominada *SLeSS* 2.0 e é apresentada nesta dissertação.

4.2 Sobre a Abordagem

O *SLeSS* 2.0 é uma abordagem ágil de integração de princípios e práticas do *Scrum* e do *Lean Six Sigma* (*LSS*), tendo como principal foco a gestão de projetos e melhoria de processos de desenvolvimento e customização de *software* para DMs.

Essa abordagem estabelece um conjunto de mecanismos para a integração dessas metodologias, definidos através da experimentação empírica especificamente para o desenvolvimento e customização de *software* para dispositivos como celulares, *smartphones* e *tablets*. Os principais objetivos dessa nova versão da abordagem podem ser resumidos a seguir:

- ✓ Aumentar a agilidade da gestão de projetos, do desenvolvimento e da customização de *software* a partir de princípios e práticas do *Scrum*;
- ✓ Controlar e melhorar os processos de desenvolvimento e customização com as técnicas do *LSS* para a melhoria sistemática da qualidade dos produtos e serviços desenvolvidos;
- ✓ Capacitar a equipe do projeto na identificação e resolução de causas de problemas dos processos de desenvolvimento e customização; e
- ✓ Avaliar e melhorar o uso de princípios e práticas do *Scrum* nos projetos.

Uma visão geral do *SLeSS* 2.0 é apresentada na Figura 4.2. Essa abordagem reutiliza os fundamentos, valores e princípios do *Scrum*, bem como os princípios do *LSS*. Dessa forma, ao invés de especificar novos fundamentos, valores e princípios, essa abordagem reutiliza os já definidos nessas metodologias, procurando garantir que sejam adequadamente seguidos.

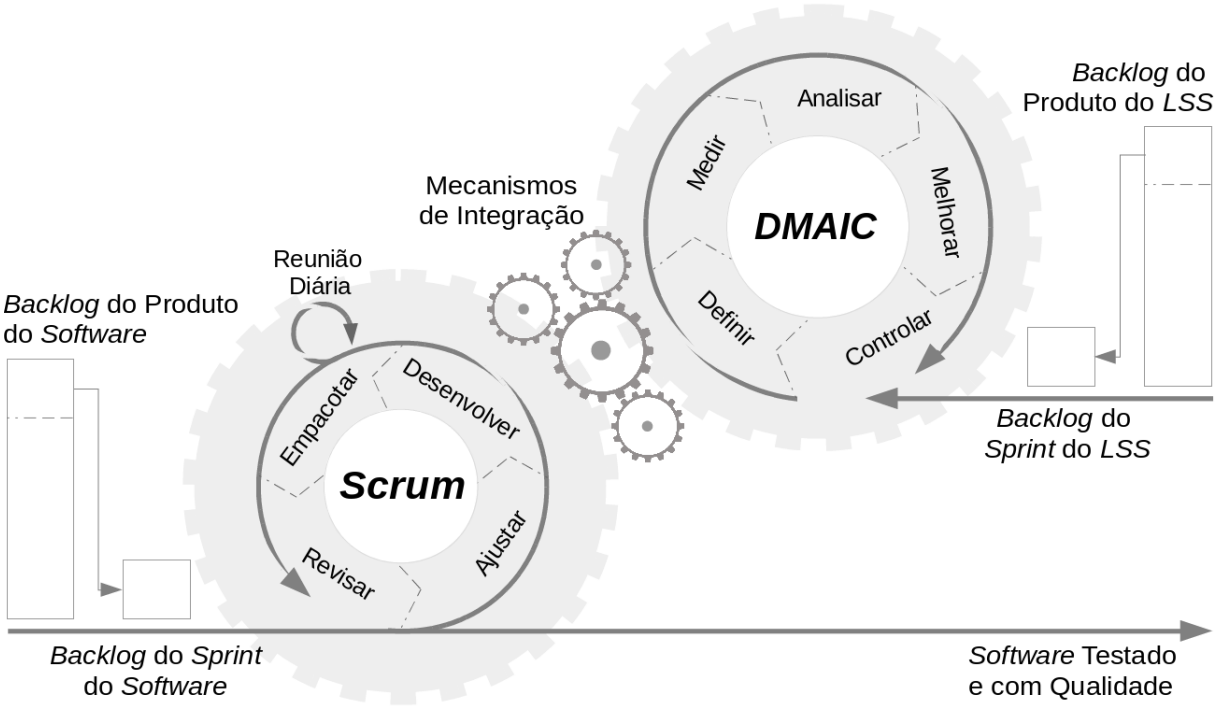


Figura 4.2: *SLeSS* 2.0 - Uma Abordagem de Integração do *Scrum* e do *Lean Six Sigma*

Nessa abordagem, o *Scrum* é utilizado tanto na gestão de projetos e desenvolvimento de *software* quanto na gestão de melhorias de processos. Já o *LSS* é utilizado tanto na melhoria de processos quanto na avaliação e melhoria do uso de princípios e práticas do *Scrum*. Para tanto, o *SLeSS* 2.0 utiliza o *DMAIC*, que é uma submetodologia do *LSS* voltada à melhoria de processos existentes. Entretanto, enquanto o *DMAIC* é tradicionalmente executado a partir do modelo em cascata (o progresso de uma fase para a próxima ocorre de uma forma puramente sequencial), o *SLeSS* 2.0 se propõe a executá-lo de forma iterativa e incremental a partir do *Scrum*.

O *SLeSS* 2.0 utiliza dois tipos de *Backlog* do Produto: um para gerenciar os requisitos do *software* a ser desenvolvido, *Backlog* do Produto do *Software*, e outro para gerenciar as melhorias dos processos de customização e desenvolvimento, *Backlog* do Produto do *LSS* (veja a Figura 4.2).

Enquanto o *Backlog* do Produto do *Software* é elaborado a partir da identificação dos requisitos funcionais e não-funcionais provenientes da visão do produto, o *Backlog* do Produto do *LSS* é elaborado a partir de problemas identificados nos processos durante a execução dos *Sprints*, como desvios de produtividade, prazo e densidade de defeitos.

Uma proposta de *Backlog* do Produto do *LSS* é apresentada na Tabela 4.1. Nessa proposta, há itens de *backlog* para cada fase do *DMAIC*. Por exemplo, na fase *Definir*, há o item

Tabela 4.1: *Backlog* do Produto do *Lean Six Sigma*, Fonte: (CUNHA et al., 2011)

Fase	**Item de *Backlog***	**Descrição**
Definir (*Define*)	Contrato	Elaborar o contrato: definir a equipe, o foco do projeto, a meta; e calcular os ganhos previstos e aprovar o contrato
	Indicadores de desempenho	Definir os indicadores de desempenho
	Coleta de dados inicial	Coletar os resultados dos indicadores de desempenho para análise inicial
	Análise inicial	Analisar os resultados da coleta para entendimento inicial dos problemas
Medir (*Measure*)	*SIPOC*	Elaborar diagrama de *SIPOC* (*Suppliers*, *Inputs*, *Process*, *Outputs*, *Customers*)
	Mapa de processo	Elaborar os mapas dos principais processos (etapas que agregam ou não valor, entradas e requisitos do cliente)
	Diagrama de causa e efeito	Elaborar diagrama de espinha de peixe (visão das possíveis causas e efeitos, através das entradas de processos)
	Matriz de causa e efeito	Elaborar uma matriz priorizando as entradas dos processos a partir do impacto desses nos indicadores
	Matriz de esforço e impacto	Elaborar uma matriz priorizando as entradas dos processos a partir do impacto e do esforço (tempo, dinheiro, pessoas)
	Capacidade inicial	Calcular essa capacidade para os principais processos
	Sistemas de medição e inspeção	Avaliar a confiabilidade e precisão dos sistemas de medição e inspeção
	Coleta de dados	Realizar a coleta de dados após validação e ajustes dos sistemas de medição e inspeção
Analisar (*Analyze*)	*FTA*	Realizar análises de causa e efeito utilizando diagramas de *FTA*
	FMEA	Utilizar a ferramenta *FMEA* para identificação das falhas potenciais e ações de melhorias
	Entradas críticas	Analisar entradas que realmente influenciam nos indicadores utilizando as técnicas do *LSS*
Melhorar (*Improve*)	Plano de ações	Realizar reuniões de *brainstorm* para elaboração do plano de melhoria
	SIPOC	Realizar otimizações no *SIPOC* após a execução do plano de ações
	Mapa de Processo	Realizar otimizações nos mapas de processos após a execução do plano de ações
	Análise dos resultados	Confirmar as melhorias, auditar e verificar os processos avaliando evidências da eficácia do plano de ações
Controlar (*Control*)	Plano de controle	Elaborar, revisar e implantar plano de controle
	Fechamento do projeto	Entregar, formalmente, os resultados do projeto *LSS* aos líderes (responsáveis) dos processos

Contrato relacionado à elaboração do contrato do projeto de melhoria (*DMAIC*), que abrange a especificação da equipe, o foco desse projeto e a meta com ganhos previstos calculados.

Há também sugestões de uso de ferramentas do *LSS* como, na fase *Analisar*, os itens *Fault Tree Analysis* (*FTA*) e *Failure Mode Effects Analysis* (*FMEA*) (veja a Tabela 4.1), que são relacionados à identificação de falhas em processos. Enquanto o *FTA* é um método dedutivo (análise *top-down*), o *FMEA* é indutivo (análise *bottom-up*), entretanto, ambos são utilizados na investigação de causas de falhas com o objetivo de redução do risco da ocorrência dessas falhas.

No *SLeSS* 2.0, o *Product Owner* e o *Scrum Master* exercem também o papel de *Green Belt* e, além de suas responsabilidades no *Scrum*, lideram os projetos de melhoria de processos (*DMAICs*).

Durante a execução dos *Sprints*, são desenvolvidos pelo Time tanto os itens do *Backlog* do *Software* quanto os do *LSS*. O *Product Owner* é o responsável pela priorização desses *backlogs* e o *Scrum Master* auxilia o time nas análises para identificação de causas raízes de problemas nos processos.

Cada *Sprint* inicia com uma reunião de planejamento, na qual o *Product Owner*, *Scrum Master* e o Time decidem em conjunto o que deverá ser implementado, por exemplo, quais itens dos *Backlogs* do *Software* e do *LSS* serão incluídos no *Sprint*.

Os itens selecionados para o *Sprint* compõem os escopos dos *Backlogs* do *Sprint* do *Software* e do *LSS* (veja a Figura 4.2). Embora esses itens possam ser executados em um mesmo *Sprint* por uma mesma equipe, o *SLeSS* 2.0 não restringe a execução dos *Sprints* a essa única opção. Dessa forma, o *Product Owner*, *Scrum Master* e o Time podem decidir em conjunto que em um *Sprint* sejam executados apenas itens do *software* ou apenas itens de melhorias de processos ou mesmo um conjunto de itens de cada um desses.

Na execução dos *Sprints*, o Time se reúne diariamente (Reunião Diária, veja a Figura 4.2) para acompanhar o progresso do trabalho.

A Reunião de Revisão é realizada ao final do *Sprint* para que o Time apresente ao *Product Owner* o resultado alcançado na iteração, por exemplo, as versões de *software* e as melhorias implantadas nos processos de desenvolvimento e customização.

Em seguida o *Scrum Master* conduz a Reunião de Retrospectiva, com o objetivo de identificar ações para a melhoria do processo, atuação do Time e qualidade do produto.

Ao final desses *Sprints*, os resultados da Revisão e Retrospectiva são insumos para os projetos de melhoria de processos (*DMAICs*), que priorizam, a partir de análises qualitativas e quantitativas, as soluções dos problemas identificados. Essas análises conduzem a definição de planos de ações, permitindo a melhoria contínua e o alinhamento do projeto às necessidades do cliente.

4.2.1 Papéis

O mapeamento dos papéis do *Lean Six Sigma* (*LSS*) aos do *Scrum* proposto pelo *SLeSS* 2.0 é apresentado na Figura 4.3. Esse mapeamento sugere que, na combinação dessas metodologias, um mesmo profissional pode assumir papéis de ambas as metodologias, por exemplo, o *Product Owner* e o *Scrum Master* podem execer também o papel do *Green Belt* do *LSS* e apenas esses papéis estão mapeados, conforme indicam as setas dessa Figura.

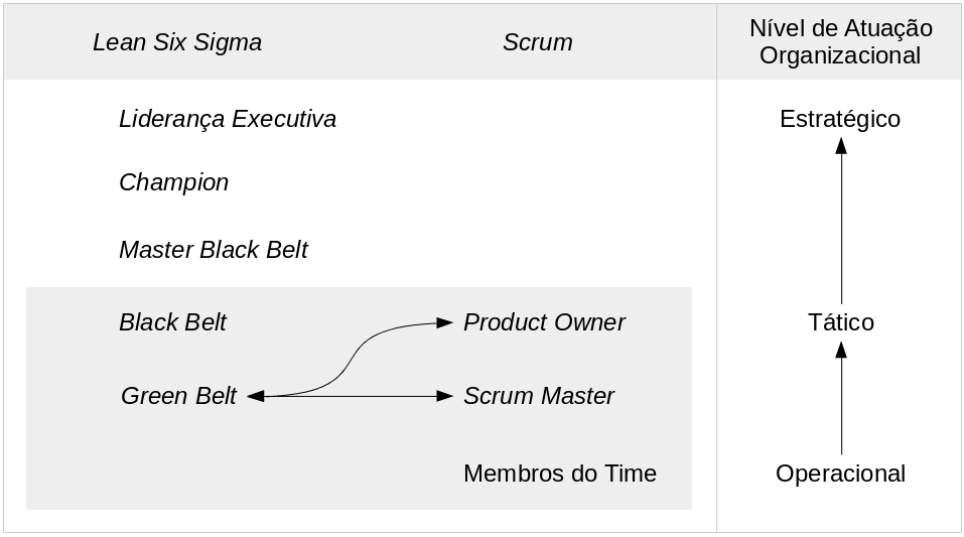


Figura 4.3: Mapeamento entre os Papéis do *Scrum* e *Lean Six Sigma*

Os papéis estão apresentados numa ordem decrescente quanto ao nível de atuação na organização (veja a Figura 4.3). Por exemplo, no *LSS*, os papéis da Liderança Executiva, *Champions* e *Master Black Belts* geralmente atuam de forma mais estratégica e tática, enquanto os *Black Belts* e *Green Belts* atuam de forma mais tática e operacional. Quanto ao *Scrum*, enquanto o *Product Owner* e o *Scrum Master* geralmente atuam de forma tática e operacional, os membros do Time atuam de forma mais operacional.

O *Green Belt* lidera projetos de melhoria *LSS* e, por não ser integralmente alocado ao *LSS*, pode atuar também como *Product Owner* ou *Scrum Master*. O papel do *Green Belt* requer que o profissional tenha experiência com o *DMAIC* e ferramentas do *LSS*. Parte dessas ferramentas demanda um bom conhecimento de estatística, essencial para que, com o auxílio dos envolvidos, o *Green Belt* controle e melhore os processos a partir da identificação de causas raízes de defeitos desses processos.

No *SLeSS* 2.0 é necessário que haja pelo menos um *Green Belt* e este pode ser ou o *Product Owner* ou o *Scrum Master*. Entretanto, o ideal é que tanto o *Product Owner* quanto o *Scrum Master* sejam também *Green Belts*.

O *Product Owner* atuando como *Green Belt* é responsável tanto por garantir que o time desenvolva incrementos de *software* que agregam valor ao cliente, a partir da priorização adequada dos requisitos do produto, quanto por garantir que esse desenvolvimento esteja continuamente alinhado às exigências de produtividade e qualidade estabelecidas por esse

cliente. Além disso, a atuação do *Product Owner* como *Green Belt* possibilita que ele priorize adequadamente os itens do *Backlog* do Produto do *Software* e os do *LSS*.

Já o *Scrum Master*, principal responsável por garantir que o time siga corretamente o *Scrum*, atuando também como *Green Belt* pode liderar tanto os *DMAICs* de avaliação e melhoria do uso de princípios e práticas do *Scrum* quanto os diretamente relacionados à melhoria dos processos de desenvolvimento e customização.

Dessa forma, no *SLeSS* 2.0, o *Product Owner* e o *Scrum Master*, além de suas responsabilidades no *Scrum*, lideram os *DMAICs* de melhoria de processos. Durante a execução dos *Sprints*, são desenvolvidos tanto os itens do *Backlog* do Produto do *Software* quanto itens do *Backlog* de melhorias de processos. O *Product Owner* é, portanto, o responsável pela priorização desses *backlogs* enquanto o *Scrum Master* auxilia o time nas análises para identificação de causas raízes de problemas nos processos de desenvolvimento e customização.

Uma organização iniciante no *LSS*, que pretende implantar o *SLeSS* 2.0, pode iniciar sua adoção pela formação de *Green Belts* e ir formando os demais papéis do *LSS* (*de Black Belts* à Liderança Executiva) a médio ou longo prazo.

4.2.2 Processo de Execução

O processo de execução do *SLeSS* 2.0 é apresentado na Figura 4.4. Esse processo inicia na fase *Pré-jogo* com a elaboração do *Backlog* do Produto do *Software* a partir dos requisitos funcionais e não-funcionais identificados através da visão do produto (passo 1). Com base nesses requisitos, o *Product Owner* e o *Scrum Master* elaboram um planejamento inicial do projeto, enquanto o Time define a primeira versão da arquitetura do sistema (passo 2).

A fase seguinte chama-se *Jogo* e inicia com o Planejamento do *Sprint*. A etapa inicial desse planejamento consiste na priorização dos itens dos *Backlogs* do Produto (passo 3).

Entretanto, no primeiro *Sprint*, o *Backlog* do Produto do *Lean Six Sigma* (*LSS*) geralmente ainda não foi elaborado, e, nesse momento, há apenas o do *Software*. Nessa etapa, o *Product Owner* estima o valor de negócio dos itens desse *backlog* para priorização.

Durante os *Sprints*, a equipe pode identificar problemas relacionados aos processos de desenvolvimento e customização, e os relacionados à utilização do processo ágil. Esses problemas são insumo na elaboração do *Backlog* do Produto do *LSS*. O *Product Owner* é responsável por manter e priorizar tanto o *Backlog* do *Software* quanto o do *LSS*.

Uma vez que os *backlogs* estão priorizados, o Time e o *Product Owner* negociam sobre quais itens serão executados no *Sprint* que inicia (passo 4). Para tanto, o Time realiza uma estimativa de esforço dos itens com maior prioridade para identificar quais podem ser desenvolvidos durante o *Sprint*. A negociação resulta, portanto, na definição dos escopos dos *Backlogs* do *Sprint* do *Software* e do *LSS*.

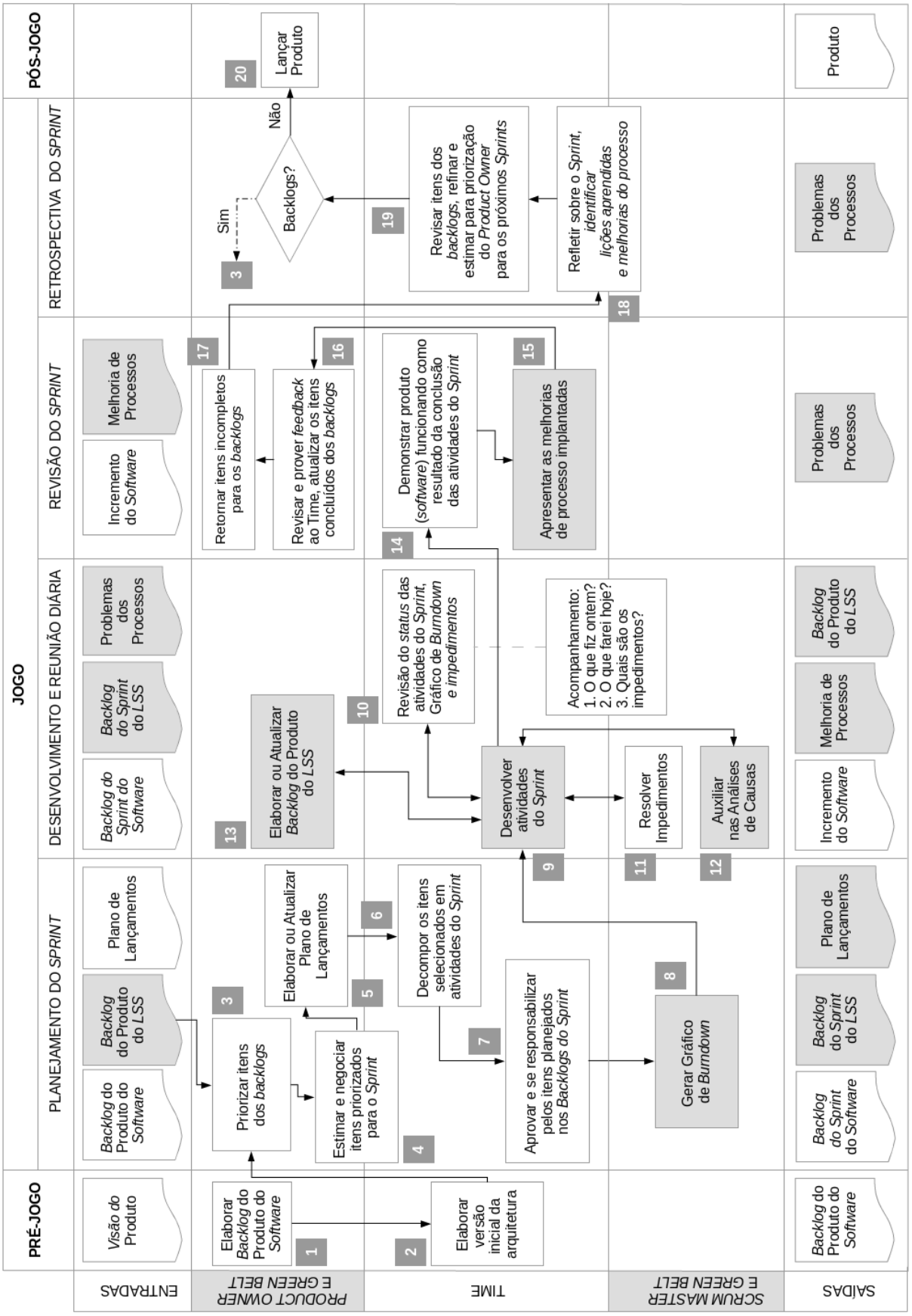


Figura 4.4: Processo de Execução do *SLeSS* 2.0

A partir da priorização desses *backlogs*, é elaborado o Plano de Lançamentos do projeto (passo 5), que contém os *Sprints* nos quais serão desenvolvidos o *software* e as melhorias dos processos de desenvolvimento.

Dessa forma, para um mesmo projeto, há apenas um Plano de Lançamentos que contém tanto o planejamento de quais itens do *Backlog* do *Software* serão implementados e quando estarão concluídos quanto o planejamento dos itens do *Backlog* do *LSS*. Esse Plano de Lançamentos estabelece o desenvolvimento iterativo e incremental do *software* e das melhorias de processos.

De posse dos *Backlogs* do *Sprint*, o Time decompõe os itens de *backlogs* em atividades que serão realizadas no *Sprint* e se responsabiliza por esse planejamento (passo 6 e 7). Em seguida, o *Scrum Master* gera o Gráfico de *Burndown* e fica responsável por mantê-lo atualizado para auxiliar no acompanhamento do progresso do *Sprint* (passo 8).

Na etapa seguinte, o Time desenvolve as atividades planejadas para o *Sprint* (passo 9). Durante esse desenvolvimento são realizadas as Reuniões Diárias para revisão do status das atividades do *Sprint* e impedimentos (passo 10). É responsabilidade do *Scrum Master* solucionar os impedimentos identificados mantendo o Time sempre focado nas atividades do *Sprint* (passo 11).

Além disso, como podem haver atividades de melhorias de processo para o *Sprint*, o *Scrum Master* também se coloca à disposição para auxiliar o Time nessas atividades, principalmente nas análises de causas dos problemas identificados (passo 12). Os problemas dos processos podem ser identificados pelo Time, *Scrum Master* ou mesmo pelo *Product Owner* e são insumos na elaboração do *Backlog* do Produto do *LSS* (passo 13).

Ao final de cada *Sprint* são realizadas as reuniões de Revisão e Retrospectiva do *Sprint*. A Revisão do *Sprint* inicia com a demonstração do incremento de *software* ao *Product Owner*, resultado do desenvolvimento das atividades do *Sprint* (passo 14). Em seguida, também são apresentadas as melhorias implantadas nos processos de desenvolvimento (passo 15). O *Product Owner* revisa, então, os resultados alcançados, provê um feedback para a equipe e, em seguida, atualiza os *backlogs* do projeto (passos 16 e 17).

Já na Retrospectiva do *Sprint*, o time reflete sobre os pontos positivos e de melhoria relativos à execução da iteração. Nesse momento, podem ser identificadas as lições aprendidas e melhorias para os processos (passo 18). Os resultados da Revisão e Retrospectiva do *Sprint* são, portanto, insumos para os projetos de melhoria de processos. Ao final da Retrospectiva, o Time também revisa os itens dos *backlogs*, realiza um refinamento e estimativa desses itens para a priorização dos mesmos pelo *Product Owner* para os próximos *Sprints* (passo 19).

Caso haja mais *Sprints* a serem executados, o ciclo do *Scrum* inicia novamente (passo 3) e, em caso contrário, o projeto entra em sua fase de encerramento, quando são realizados os preparativos para o lançamento do produto final (passo 20).

4.2.3 Mecanismos de Integração

A utilização do termo *Mecanismos de Integração* foi escolhida após a finalização da primeira versão da visão geral dessa abordagem, apresentada na Figura 3.6. Nessa Figura, os ciclos de iterações do *Scrum* e melhoria do *DMAIC* lembram as engrenagens de um relógio e, por analogia, embora possuam tamanhos distintos, a partir dos mecanismos corretos de integração, podem funcionar adequadamente em conjunto, de maneira sincronizada e harmônica.

Os mecanismos de integração do *SLeSS* 2.0 são estratégias de combinação dos princípios e práticas do *Scrum* aos do *Lean Six Sigma* (*LSS*) e vice-versa, propostos com base na experimentação empírica realizada em projetos reais. O *SLeSS* 2.0 estabelece 4 mecanismos e 8 práticas, conforme são apresentados na Figura 4.5 e explicados a seguir.

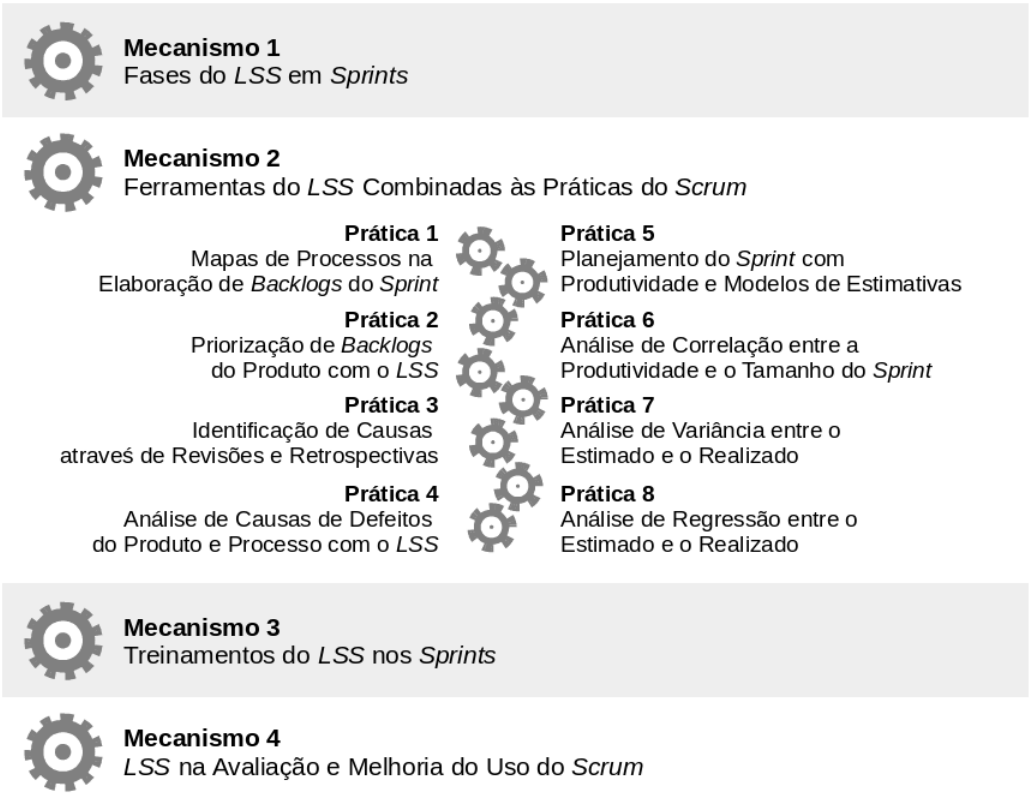


Figura 4.5: Mecanismos de Integração do *SLeSS* 2.0

4.2.3.1 Mecanismo 1 - Fases do *Lean Six Sigma* em *Sprints*

Os projetos de melhoria de processos com o *LSS* são desenvolvidos com o *DMAIC* que, tradicionalmente, é executado no modelo cascata. No entanto, é comum que atividades de fases seguintes sejam executadas antecipadamente para suprir informações para a fase atual. Na fase *Definir*, por exemplo, é necessário determinar as metas de melhoria e calcular o custo-benefício da execução do *DMAIC*. Entretanto, nem sempre os processos de desenvolvimento são medidos e por vezes as medições existentes não são suficientes ou mesmo confiáveis para a análise inicial do problema e determinação das metas. Portanto, há a necessidade de planejar

novas medições e realizá-las antes mesmo do início da fase *Medir*. O mesmo também pode ocorrer, por exemplo, na fase *Medir*, quando na medição da situação atual do processo é necessário identificar, mesmo que superficialmente, quais medições foram afetadas por causas especiais (fontes de variação imprevisíveis, intermitentes e instáveis).

Dessa forma, na execução tradicional do *DMAIC* geralmente há interseção entre as fases e, na prática, essa execução não segue rigorosamente a estratégia de somente iniciar a fase seguinte após o término da atual. Além disso, a busca por níveis elevados de produtividade e qualidade exigem que os processos de desenvolvimento sejam continuamente revisados e melhorados e esse ambiente de mudanças no desenvolvimento pode se beneficiar do uso de princípios e práticas ágeis.

Sendo assim, o *SLeSS* 2.0 propõe a execução das fases do *DMAIC* de forma iterativa e incremental, a partir da gestão dos projetos de melhoria com o *Scrum*. Uma visão da execução do *DMAIC* tradicional versus iterativo e incremental é apresentada na Figura 4.6.

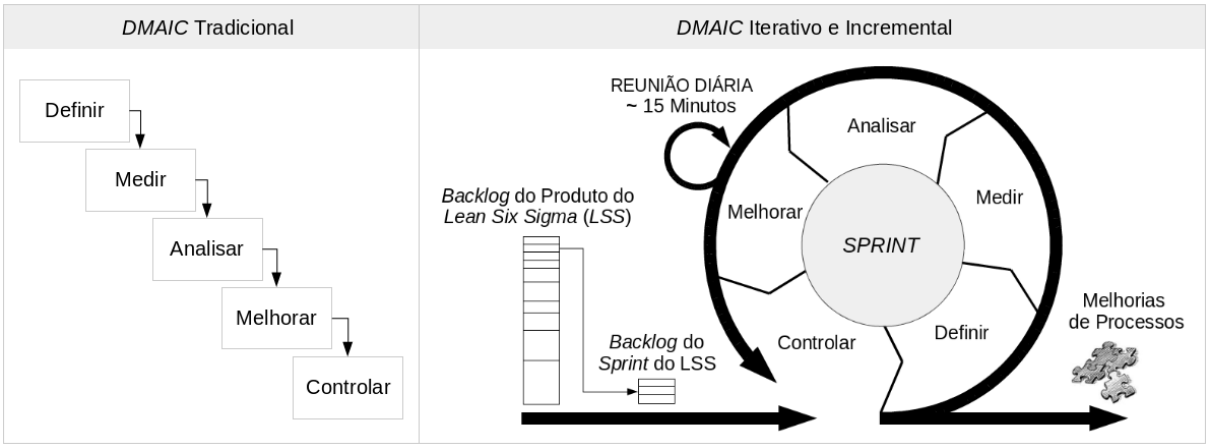


Figura 4.6: *DMAIC* Tradicional Versus *DMAIC* Iterativo e Incremental

Para cada projeto de melhoria, um *Backlog* do Produto do *LSS* é elaborado considerando o escopo das fases do *DMAIC* (veja a Figura 4.6 e também a proposta de *Backlog* na Tabela 4.1). A priorização dos itens desse *backlog* pode considerar, por exemplo, os resultados de análises de causas de defeitos dos processos. A partir dessa priorização, os itens são selecionados para a execução no *Sprint*, compondo o *Backlog* do *Sprint* do *LSS*.

As análises dos processos possibilitam que as ações de melhoria sejam identificadas e priorizadas a partir de uma avaliação do custo-benefício e considerando também as necessidades do cliente. Essas ações de melhoria podem ser desenvolvidas no *Sprint* e benefícios dessas melhorias já podem ser observados em *Sprints* futuros, possibilitando ganhos de produtividade do Time e qualidade do produto.

4.2.3.2 Mecanismo 2 - Ferramentas do *Lean Six Sigma* Combinadas às Práticas do *Scrum*

As ferramentas do *LSS* podem ser utilizadas, por exemplo, nas estimativas de esforço do Time, planejamento e controle dos *Sprints* e na identificação e análise de causas

de defeitos. A seguir, são apresentadas as práticas propostas pelo *SLeSS* 2.0 para o uso das ferramentas do *LSS* no *Scrum*:

- *Prática 1 - Mapas de Processos na Elaboração dos Backlogs do Sprint* - um mapa de processo contém informações como as entradas e saídas do processo, os passos, envolvidos e responsáveis.

  Por exemplo, no processo de adaptação das customizações, há atividades como a análise de especificações das operadoras, implementação de adaptações, realização de testes, revisão técnica e disponibilização. Essas atividades são necessárias na customização e precisam ser planejadas para execução durante os *Sprints* do projeto.

  Esses mapas dos processos podem ser utilizados, portanto, durante o planejamento dos *Sprints Backlogs*, no momento da decomposição dos itens de *backlog* nas atividades do Time. Esses mapas servem, dessa forma, como uma Estrutura Analítica de Projetos (EAP) (INSTITUTE, 2008), fornecendo a visão das atividades que precisam ser adicionadas aos *Sprints*.

  A utilização desses mapas possibilita a revisão dos processos de desenvolvimento, uma vez que a equipe está continuamente os utilizando nos planejamentos, recicla e capacita essa equipe nesses processos e auxilia na identificação de problemas.

- *Prática 2 - Priorização dos Backlogs do Produto com o LSS* - a partir das análises realizadas com as técnicas do *LSS*, as ações corretivas de processos e melhorias são priorizadas no *Product Backlog* pelo *Product Owner*. A correta priorização entre os itens de *backlog* de melhorias e itens de *backlog* do *software* dependerá das decisões do *Product Owner*, que precisará conhecer bem as necessidades do cliente, os processos de desenvolvimento e as técnicas do *LSS*.

- *Prática 3 - Identificação de Causas através de Revisões e Retrospectivas* - durante as Revisões dos *Sprints*, os resultados da iteração, versões de *software* e outros entregáveis, são apresentados ao *Product Owner* que os inspeciona, identificando defeitos e melhorias.

  Parte desses defeitos pode está relacionada a erros nos processos ou mesmo na execução desses processos. Esses defeitos podem, portanto, ser posteriormente analisados pelo time em reuniões técnicas, a partir da utilização de diagramas de Causa e Efeito, Pareto, *FTA* e *FMEA*, por exemplo. O *Scrum Master* que também exerce o papel do *Green Belt* pode auxiliar o time no uso dessas ferramentas.

  Durante essas análises, a utilização dos mapas de processos pode contribuir no mapeamento das causas às etapas e entradas dos processos, bem como na identificação de entradas críticas, que são as reais fontes de defeitos.

  Já as Retrospectivas dos *Sprints* possibilitam que o time reflita sobre a execução da iteração e identifique ações para a melhoria dos processos, inclusive melhorias do próprio processo ágil. As medições e análises dos projetos de melhoria em andamento podem ser utilizados para auxiliar o time na identificação de ações de melhoria mais eficazes.

- *Prática 4 - Análise de Causas de Defeitos do Produto e Processo com o LSS* - os defeitos coletados a partir do ciclo de vida do *Scrum* podem ser analisados e priorizados. Os defeitos identificados durante os testes podem estar relacionados à erros na análise e projeto, codificação, no entendimento ou especificação dos requisitos, integração do código ou mesmo problemas relacionados aos processos de desenvolvimento. Essas análises de defeitos podem ser realizadas através de ferramentas do *LSS* como os diagramas de Pareto, Causa e Efeito e análises de variância, regressão e correlações. O uso dessas ferramentas auxilia na identificação das causas raízes e os padrões de análise e projeto, codificação podem, por exemplo, ser alterados como resultado, minimizando a quantidade de novos defeitos nos *Sprints* seguintes;

- *Prática 5 - Planejamento do Sprint com Produtividade e Modelos de Estimativas* - a medição da produtividade e elaboração de modelos de estimativas podem ser úteis em análises de prazo, riscos e custos. Durante a fase de planejamento, por exemplo, os itens dos backlogs são selecionadas com base na priorização do *Product Owner* e estimativa de esforço do Time. A quantidade de itens de um *Sprint* depende, portanto, da capacidade de produção da equipe e estimativas, informações utilizadas no planejamento das iterações. O conhecimento e a precisão do histórico de estimativas, portanto, pode ajudar a melhorar a previsão de futuras iterações, as estimativas e o planejamento do projeto;

- *Prática 6 - Análise de Correlação entre a Produtividade e o Tamanho do Sprint* - o tamanho dos *Sprints* geralmente varia de 1 a 4 semanas e esse tamanho pode ser continuamente avaliado, por exemplo, a partir de uma análise de correlação da produtividade da equipe e a duração desses *Sprints*. Essa análise pode ajudar a identificar o tamanho de *Sprint* mais adequado ao projeto;

- *Prática 7 - Análise de Variância entre o Estimado e o Realizado* - a realização de testes de variância para análise do esforço estimado versus o realizado pode contribuir na melhoria do modelo de estimativa e, por conseguinte, na previsão dos prazos do projeto; e

- *Prática 8 - Análise de Regressão entre o Estimado e o Realizado* - uma análise de regressão estuda o relacionamento entre variáveis e esse relacionamento é geralmente representado por um modelo matemático, por exemplo, por uma equação que associa essas variáveis. Dessa forma, a análise de regressão entre o estimado e o realizado pode ser utilizado, por exemplo, para identificar uma equação ou função de regressão para avaliação e melhoria das estimativas da equipe.

4.2.3.3 Mecanismo 3 - Treinamentos do *Lean Six Sigma* nos *Sprints*

Um dos objetivos da abordagem é a capacitação da equipe do projeto na identificação e resolução de causas de problemas dos processos de desenvolvimento. Para atingir esse objetivo, esse mecanismo estabelece a preparação e a realização de treinamentos no *LSS* durante os *Sprints*, fornecendo a capacitação para que a equipe atue adequadamente no desenvolvimento de melhorias dos processos.

Entretanto, uma das dificuldades dessa capacitação é exatamente o custo dos treinamentos nessa metodologia. Para amenizar esse custo, o *SLeSS* 2.0 propõe o planejamento e a realização de treinamentos sob demanda, conforme a necessidade da utilização de técnicas e ferramentas do *LSS* no projeto. Esses treinamentos podem ser teóricos e práticos, realizados a partir de palestras, seminários, fóruns e auto-estudo. Os próprios integrantes da equipe, por exemplo, o *Product Owner* e o *Scrum Master*, são responsáveis pelo planejamento e pela execução dos treinamentos.

4.2.3.4 Mecanismo 4 - *Lean Six Sigma* na Avaliação e Melhoria do Uso do *Scrum*

O *SLeSS* 2.0 propõe o método *Agile DMAIC* para avaliação e melhoria do uso do *Scrum* nos projetos. Esse método se baseia em coletar periodicamente a percepção da equipe do projeto (*Product Owner*, *Scrum Master* e Time) em relação ao uso do *Scrum* e, a partir dessa percepção, identificar possíveis problemas em sua implementação. O *Agile DMAIC* é apresentado na Subseção 4.2.5.

4.2.4 Fluxo de Implantação

O processo para a implantação do *SLeSS* 2.0 é apresentado na Figura 4.7 e um resumo desse processo é apresentado na Tabela 4.2. O processo inicia com a elaboração de um plano de implantação (passo 1), que abrange a identificação e priorização dos projetos, a elaboração de um cronograma macro inicial e a identificação, preparação (ou contratação) de treinamentos *Scrum* e *LSS*.

Em seguida, um dos projetos é selecionado como piloto (passo 2) e os treinamentos *Scrum* são então planejados e realizados para a equipe desse projeto (passo 3). Esses treinamentos são teóricos e práticos e apresentam ao time os fundamentos, princípios, valores, papéis, artefatos e eventos do *Scrum*.

Na etapa seguinte, a implantação do *Scrum* é realizada para o projeto piloto selecionado (passo 4). Essa implantação abrange a definição da equipe, papéis de cada integrante, tamanho do *Sprint*, velocidade inicial (capacidade de produção), elaboração do *Product Backlog* inicial, Plano de Lançamentos, realização dos planejamentos e execução dos *Sprints*, Reuniões Diárias, Revisões e Retrospectivas. Ao final dessa etapa, o planejamento de implantação do *SLeSS* 2.0 é revisado e ajustado para incorporar as lições aprendidas. O *Agile DMAIC* pode ser utilizado durante a implantação do *Scrum* para guiar o Time na adoção dessa metodologia a partir da avaliação contínua do uso de princípios e práticas do *Scrum* no projeto.

Os treinamentos *LSS* são então planejados e realizados para o *Product Owner* e o *Scrum Master* e seus possíveis substitutos (passo 5). Esses treinamentos são teóricos e práticos e tem foco no *DMAIC*, apresentando os fundamentos, princípios, papéis e ferramentas do *LSS*. Após a finalização desses treinamentos e já executados os primeiros *Sprints*, os problemas identificados no projeto e na adoção do *Scrum* são analisados (passo 6).

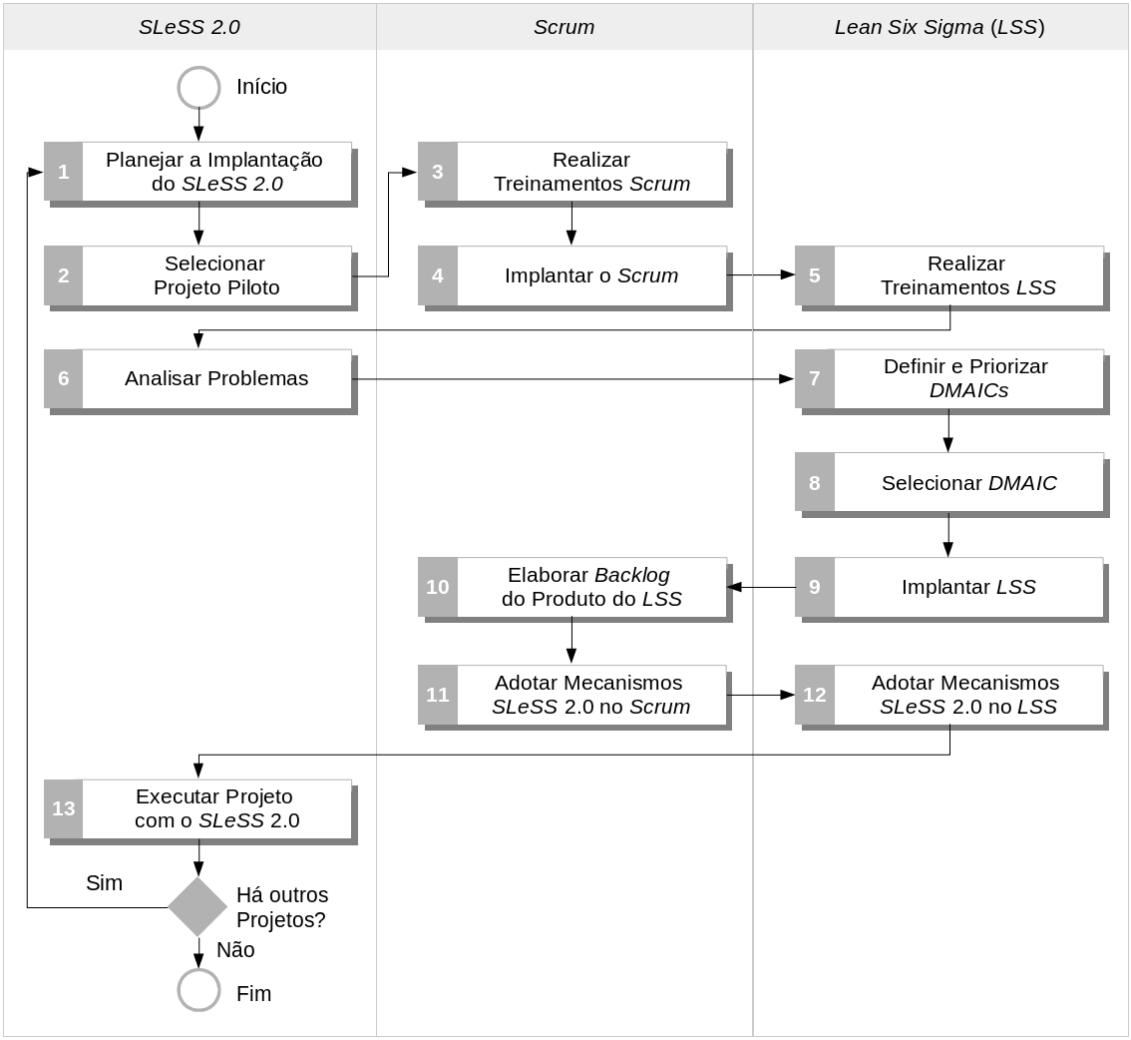


Figura 4.7: Fluxo de Implantação do *SLeSS* 2.0

A partir desses problemas, os projetos de melhoria (*DMAICs*) são definidos e priorizados (passo 7). Essa etapa consiste na definição do escopo desses *DMAICs*, seus objetivos e metas. A priorização dos *DMAICs* pode ser realizada com base em uma análise do custo-benefício dos mesmos. Um desses *DMAICs* é então selecionado para execução (passo 8), o que representa o início do processo de implantação do *LSS* (passo 9).

Em seguida, o *Product Owner* e o *Scrum Master* elaboram o *Product Backlog* do *LSS* e iniciam o *DMAIC* para solucionar os principais problemas identificados, atuando nas causas raízes desses problemas (passo 10).

Na etapa seguinte, os mecanismos do *SLeSS* 2.0 são adotados no projeto (passos 11 e 12) e, após isso, o projeto é então executado com o *SLeSS* 2.0 (passo 13). Ao final dessa etapa, o plano de implantação pode ser ajustado com as lições aprendidas e utilizado para implantação dessa abordagem nos demais projetos priorizados.

O fluxo de implantação do *SLeSS* 2.0 inclui as etapas para implantação do *Scrum* e *LSS* nos projetos. Entretanto, no caso de projetos que já utilizam essas metodologias, as etapas de implantação das mesmas podem ser simplesmente ignoradas.

Tabela 4.2: Passo a Passo para a Implantação do *SLeSS* 2.0

Etapa	**Passo**	**Descrição**
Planejar a Implantação do *SLeSS* 2.0	1	- Identificar e priorizar projetos - Elaborar cronograma macro - Preparar treinamentos *Scrum* e *LSS*
Selecionar Projeto Piloto	2	Definir o projeto piloto para implantação do *SLeSS* 2.0
Realizar Treinamentos *Scrum*	3	Treinamentos teóricos e práticos, apresentando os fundamentos, princípios, valores, papéis, artefatos e eventos do *Scrum*
Implantar o *Scrum*	4	- Definir a equipe e papéis de cada integrante - Determinar o tamanho do *Sprint* e a velocidade do time (capacidade de produção) - Elaborar o *Backlog* do Produto, estimando o valor de negócio e esforço dos itens desse *backlog* para priorização - Realizar o planejamento dos *Sprints* e executá-los - Realizar Reuniões Diárias, Revisões e Retrospectivas dos *Sprints*
Realizar Treinamentos *LSS*	5	Treinamentos teóricos e práticos com foco no *DMAIC* apresentando os fundamentos, princípios, papéis e ferramentas do *LSS*
Analisar Problemas	6	Analisar os problemas do projeto, em relação ao desenvolvimento do produto e implantação do *Scrum*. Essa análise inicial servirá de insumo para a definição e priorização de *DMAICs*
Definir e Priorizar *DMAICs*	7	Definir o escopo dos projeto e seus objetivos e metas. A priorização deve pode ser realizada com base em uma análise macro do custo-benefício dos *DMAICs*
Selecionar *DMAIC*	8	Um dos projetos de melhoria é selecionado
Implantar o *LSS*	9	A implantação inicia com a execução do *DMAIC* selecionado. As fases desse *DMAIC* são então decompostas em itens de *backlog* para que seja executado como o *Scrum*
Elaborar *Backlog* do Produto do *LSS*	10	O *Backlog* do Produto do *LSS* pode ser elaborado a partir do *template* apresentado na Tabela 4.1
Adotar Mecanismos *SLeSS* 2.0 no *Scrum*	11	- Utilizar as ferramentas do *LSS* combinadas às práticas do *Scrum* - Realizar com a equipe as análises de causa e efeitos dos principais problemas encontrados - Avaliar e melhorar o nível de uso do *Scrum*, utilizando o *Agile DMAIC*
Adotar Mecanismos *SLeSS* 2.0 no *LSS*	12	- Executar o *DMAIC* de forma iterativa e incremental com o *Scrum*, ou seja, com *Backlog* do Produto do *LSS* priorizado e o planejamento dos *Sprints* - Incluir treinamentos *LSS* no *Backlog* do *LSS* e priorizá-los de acordo com a necessidade do projeto - Utilizar as análises realizadas com as ferramentas do LSS para priorização dos *backlogs* (desenvolvimento de *software* e melhoria de processos)
Executar o Projeto com o *SLeSS* 2.0	13	Realizar a execução do projeto após a implantação dos mecanismos de integração

4.2.5 *Agile DMAIC*

Nesta seção é apresentado o *Agile DMAIC*, que é um método sistemático proposto neste trabalho para avaliar e melhorar o uso do *Scrum* em projetos de *software*. Esse método pode ser utilizado tanto na adoção do *Scrum* quanto na melhoria de sua implementação. Os objetivos do *Agile DMAIC* podem ser resumidos a seguir:

- ✓ Possibilitar a gestão quantitativa da implementação do *Scrum*;
- ✓ Avaliar o uso de princípios e práticas do *Scrum* por projeto; e
- ✓ Identificar e tratar as causas de problemas para a evolução da implementação do *Scrum*, considerando o contexto e a necessidade dos projetos e clientes.

O *Agile DMAIC* utiliza princípios e técnicas do *LSS* e foca na execução, de forma iterativa e incremental, de uma versão simplificada do *DMAIC* (TAYNTOR, 2007). A execução do *Agile DMAIC* é realizada em paralelo à execução de *Sprints* do *Scrum* e à medida que problemas no uso do *Scrum* são identificados e tratados, as melhorias são implantadas a cada *Sprint* do *Agile DMAIC*.

Uma visão geral do *Agile DMAIC* é apresentada na Figura 4.8 e os detalhes são apresentados no decorrer desta seção. O *Agile DMAIC* é composto de três componentes:

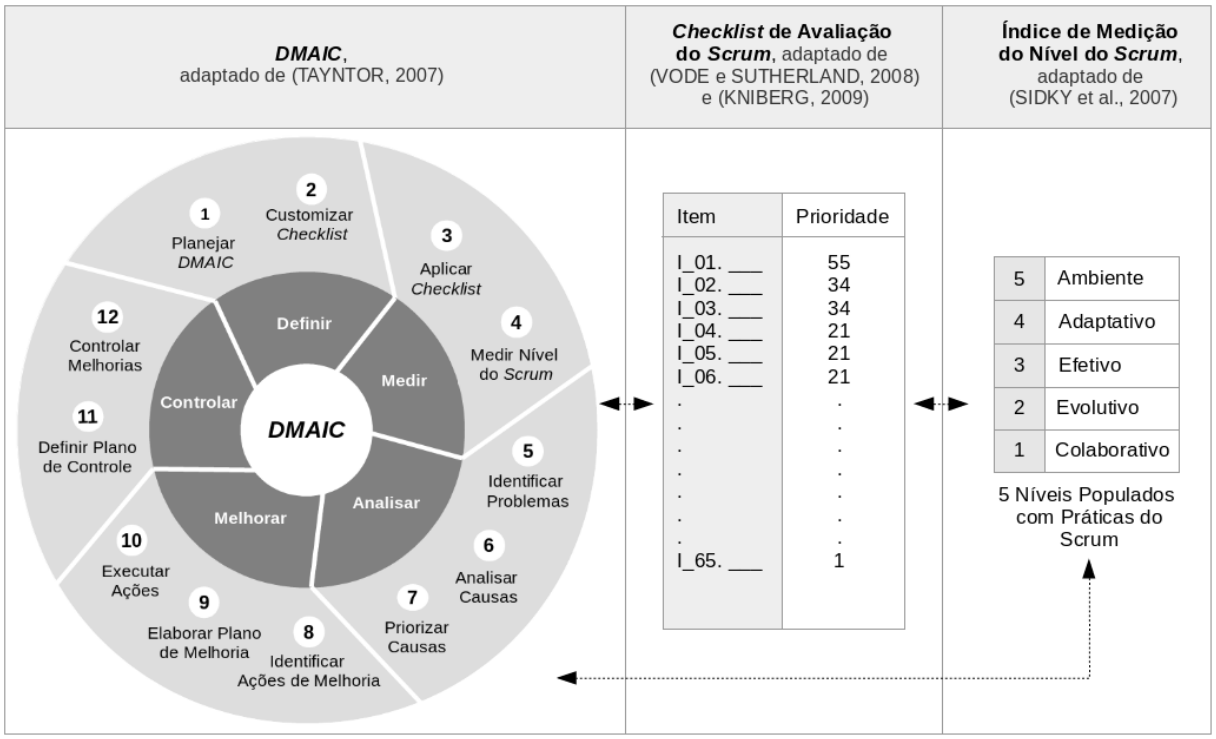


Figura 4.8: Visão Geral do *Agile DMAIC*

- *DMAIC* (TAYNTOR, 2007), uma submetodologia do *LSS* (GEORGE, 2003) voltada à melhoria de processos. No método proposto neste trabalho, o *DMAIC* é utilizado para

a definição de metas, identificação e análise de causas raízes, implantação e controle de melhorias para possibilitar a gestão quantitativa da implementação do *Scrum*.

- *Checklist* de Avaliação do *Scrum* (*CAS*), utilizado para coletar a percepção das equipes quanto ao uso de princípios e práticas do *Scrum*; e
- Índice de Medição do Nível do *Scrum* (*IMS*), que avalia a adoção das práticas do *Scrum* por projeto;

O *Agile DMAIC* se baseia em coletar periodicamente a percepção da equipe do projeto (*Product Owner*, *Scrum Master* e Time) em relação ao uso do *Scrum* e, a partir dessa percepção, identificar possíveis problemas em sua implementação.

O resultado da aplicação do *CAS* é utilizado como insumo na identificação de problemas e, além desse *checklist*, as Reuniões de Revisão e Retrospectivas também podem ser fontes de identificação de problemas.

A partir da aplicação do *CAS* e da análise da coleta é possível classificar o nível do *Scrum*, utilizando o *IMS*, e identificar onde a equipe pode atuar para melhorar seus resultados com essa metodologia.

Uma vez identificados os problemas, é necessário analisar as causas, definir e implantar ações de melhoria, antecipando os benefícios do *Scrum*, como a melhoria na produtividade do time e na qualidade do produto (DYBÅ; DINGSØYR, 2008).

Sendo assim, o *Agile DMAIC* pode ser utilizado como um guia durante a adoção do *Scrum* nos projetos, facilitando a implantação do *SLeSS* 2.0, uma vez que a adoção do *Scrum* é uma das etapas iniciais do fluxo de implantação dessa abordagem.

Além disso, mesmo após a adoção do *Scrum*, o *Agile DMAIC* pode ser utilizado na avaliação contínua do uso dessa metodologia nos projetos, auxiliando os times na evolução da agilidade da gestão de projetos e desenvolvimento de *software*. Como no *SLeSS* 2.0 a gestão de melhorias de processos é realizada a partir do *Scrum*, essa melhoria de agilidade tende a impactar positivamente nessa gestão também.

Se por um lado a necessidade de capacitação nas técnicas e ferramentas do *Lean Six Sigma* (*LSS*) para o *Product Owner* e *Scrum Master* é uma das principais vulnerabilidades à implantação do *SLeSS* 2.0, devido ao custo de treinamentos técnicos e ao tempo necessário para o aprendizado adequado de princípios e técnicas do *LSS*, por outro lado, o *Agile DMAIC* utiliza uma versão simplificada do *DMAIC* e por essa simplificação não demandar um nível alto de experiência no *LSS*, esse método consegue introduzir os principais conceitos do *SLeSS* 2.0, que são úteis tanto na melhoria do processo ágil quanto na melhoria de processos de desenvolvimento, sem a necessidade de treinamentos avançados no *LSS*.

Uma vez que o *Agile DMAIC* propõe o uso do *DMAIC* de forma iterativa e incremental e também é utilizado na avaliação e melhoria do uso do *Scrum*, com a adoção desse método, dois mecanismos de integração propostos no *SLeSS* 2.0 são consequentemente introduzidos nos projetos (Mecanismos 1 e 4, veja a Seção 4.2.3).

O *Agile DMAIC* é resultado da continuação da pesquisa apresentada em (CUNHA et al., 2011) e tem como objetivo aprimorar a abordagem *SLeSS* com foco na avaliação das práticas e princípios do *Scrum*.

Em relação aos outros trabalhos apresentados no Capítulo 3, o *Agile DMAIC* pretende ser menos extenso, como o *MiniDMAIC*, mas sendo específico para avaliação e melhoria do *Scrum*. Vale ressaltar que ele pode ser utilizado em conjunto com o *MiniDMAIC*, mas possuem objetivos distintos.

Dessa forma, o *Agile DMAIC* é utilizado como ferramenta para a definição de uma meta de melhoria no uso do *Scrum*, na análise de seu estado atual, na identificação e implantação de melhorias e no controle das mesmas para que o uso dessa metodologia continue em constante evolução.

4.2.5.1 *DMAIC*

O *Agile DMAIC* utiliza uma simplificação do *DMAIC* proposta nesta dissertação para a avaliação e melhoria do uso do *Scrum* em projetos de *software*. Nessa simplificação, o *CAS* é utilizado para a coleta da percepção da equipe e o nível do *Scrum* é calculado a partir dessa percepção e da priorização de itens do *CAS*, customizado pela equipe. As fases e os passos dessa simplificação do *DMAIC* utilizada pelo *Agile DMAIC* são apresentados na Tabela 4.3.

A customização dos itens e da priorização do *CAS* é realizada de acordo com o contexto do projeto e do cliente e visa identificar o nível máximo possível do *Scrum* para o projeto, isso porque podem existir restrições impostas pelo negócio, pelo cliente, ou pela organização que dificultem ou inviabilizem o uso de algumas práticas do *Scrum*. Essas alterações no *DMAIC* objetivam definir um indicador de desempenho, o nível do *Scrum*, para gerir a implementação do *Scrum* a partir do *DMAIC*.

Enquanto o *DMAIC* é tradicionalmente implementado a partir do modelo em cascata (o progresso de uma fase para a próxima acontece de uma forma puramente sequencial), o *Agile DMAIC* se propõe a utilizá-lo de forma iterativa e incremental, possibilitando que a duração de seu ciclo seja reduzida, potencializando a antecipação da implantação de melhorias em relação ao uso tradicional do *DMAIC*.

Dessa forma, com essa simplificação do *DMAIC*, o método objetiva melhorar a agilidade nos projetos, focando na avaliação e melhoria do *Scrum*, e sua utilização é justificada pelo fato de que os princípios e práticas do *Scrum* contribuem na melhoria da produtividade da equipe e qualidade do produto (LAYMAN et al., 2004)(DYBÅ; DINGSØYR, 2008).

Tabela 4.3: Fases e Passos do DMAIC Utilizado no *Agile DMAIC*

	Passo		Descrição
Definir	1	Planejar *DMAIC*	Elaborar plano de execução do *DMAIC*, especificando a meta de melhoria do nível do *Scrum* por projeto, o escopo de atuação (projetos alvo), o responsável (*Green Belt*) pelo *DMAIC* e demais envolvidos
	2	Customizar *Checklist*	Definir a frequência de aplicação do *CAS* e customizá-lo (revisão de itens e priorização) de acordo com o projeto e cliente. Por exemplo, por necessidade do negócio, o cliente define que o *software* só seja implantado em seu ambiente ao final do projeto e, dessa forma, o item I_13 do *CAS* deixa de ser prioridade
Medir	3	Aplicar *Checklist*	O *CAS* é aplicado à equipe do projeto (*Product Owner*, *Scrum Master* e o Time). Seções do *CAS* completamente atendidas podem ser excluídas em aplicações seguintes até que haja indicativo de regressão
	4	Medir Nível do *Scrum*	A coleta é validada e itens com menos de 25% e mais de 75% de pontuação, em relação ao número de participantes da coleta (idealmente, toda a equipe), são definidos como não atendidos e atendidos, respectivamente. Itens fora desses intervalos precisam ser revisados em conjunto com a equipe. A classificação do nível do *Scrum* é acessada pelos mapeamentos das Tabelas 4.4 e 4.5. Já o indicador do nível do *Scrum* é a razão da soma de prioridades de itens atendidos pela soma de todas as prioridades, multiplicada por 10, $$10 * \frac{\sum Prioridades\ dos\ Itens\ Atendidos}{\sum Todas\ as\ Prioridades}$$
Analisar	5	Identificar Problemas	Os itens não atendidos pelo projeto são considerados problemas e são priorizados conforme customizados pela equipe no *CAS*
	6	Analisar Causas	De posse da coleta e de problemas já identificados, é possível discernir entre causas relacionadas ao entendimento da equipe, desconhecimento de princípios e práticas, falha na comunicação ou demais causas, as quais podem ser identificadas com uso de técnicas do *LSS*
	7	Priorizar Causas	As causas são priorizadas a partir das análises realizadas, da priorização do *CAS* e dos mapeamentos de níveis e práticas das Tabelas 4.4 e 4.5
Melhorar	8	Identificar Ações de Melhoria	Uma vez identificadas as causas raízes dos problemas, a definição de ações de melhoria é realizada com a equipe
	9	Elaborar Plano de Melhoria	O plano de melhoria estabelece a priorização das ações, os prazos e os responsáveis pela execução e acompanhamento dessas ações
	10	Executar Ações	As ações são executadas conforme estabelecido no plano de melhoria
Controlar	11	Definir Plano de Controle	Avaliar riscos, definir gatilhos, ações, responsáveis e prazos
	12	Controlar Melhorias	Controlar os processos do projeto conforme definido no plano de controle, para assegurar que as melhorias já implantadas sejam duradouras

4.2.5.2 *Checklist* de Avaliação do *Scrum*

O *Checklist* de Avaliação do *Scrum* (*CAS*[3]) foi elaborado a partir do *Nokia Test* (VODE; SUTHERLAND, 2008) e do CRISP *Scrum-Checklist* (KNIBERG, 2009) com o intuito de consolidar o melhor desses *checklists* e solucionar os pontos de melhoria dos mesmos, conforme já mencionado na Seção 3.4. Esse novo *checklist* é aplicado à equipe do projeto (*Product Owner*, *Scrum Master* e o Time) para obter, a partir da percepção dessa equipe, os pontos fortes e fracos da atual implementação do *Scrum*.

O *CAS* é formado de 65 itens (I_01 a I_65) distribuídos em treze seções, ordenadas de forma decrescente quanto à prioridade das mesmas em relação aos benefícios do *Scrum*. Cada item possui uma priorização (1 a 55, utilizando a sequência de *Fibonacci*, sendo 1 a menor e 55 a maior prioridade) e essa priorização foi predefinida com base na experimentação do *Agile DMAIC* nos projetos reais utilizados como estudo de caso neste trabalho (veja o Capítulo 5) e pode ser customizada pela equipe.

As seções e itens do *CAS* são apresentados na Figura 4.9. Por exemplo, a seção *Planejamento do Sprint* possui 8 itens (I_04 a I_11), onde o primeiro item dessa seção verifica se há planejamento a cada iteração e possui prioridade 34 (P_34).

O *CAS* foca em avaliar como são utilizados os eventos e artefatos do *Scrum*. Entretanto, ele também abrange bem os papéis, fundamentos, valores e princípios dessa metodologia. A distribuição percentual dos itens desse checklist em categorias (veja a Seção 3.3.3) é apresentada na Figura 4.10 e a análise detalhada dessa distribuição é apresentada a seguir:

- *Fundamentos* - 11,1% dos itens avaliam a utilização dos fundamentos do *Scrum*, com itens sobre a transparência, inspeção e adaptação. Há itens a respeito da melhoria contínua do processo, da visibilidade dos artefatos e da utilização da definição de *software* pronto, bem como da realização de testes durante a iteração;
- *Valores e Princípios* - 4,8% dos itens avaliam a utilização dos valores e princípios do *Scrum*. São basicamente itens que verificam o foco na colaboração com o cliente e a entrega de *software* conforme a necessidade do negócio desse cliente;
- *Papéis* - 17,5% dos itens avaliam todos os papéis do *Scrum*. Há itens sobre a experiência e competência, a interação entre os papéis, o tamanho e alocação do time, se há interferências externas ao time e se o *Product Owner* tem a autoridade adequada;
- *Artefatos* - 19% dos itens avaliam a utilização dos principais artefatos. Os itens abrangem diretamente os seguintes artefatos: *Backlog* do Produto, *Backlog* da *Sprint*, Gráfico de *Burndown* e Plano de Lançamentos. São avaliados como esses artefatos são utilizados, a qualidade dos mesmos, como são estimados e utilizados no planejamento e acompanhamento; e
- *Eventos* - 47,6% dos itens avaliam os eventos do *Scrum*. Há itens sobre todos os principais eventos: Planejamento da *Sprint*, *Sprint*, Reunião Diária, Revisão da *Sprint*

[3]http://www.great.ufc.br/CAS/Checklist_de_Avaliacao_do_Scrum(CAS).pdf

Figura 4.9: *Checklist* de Avaliação do *Scrum*

e Retrospectiva. Há itens que avaliam a utilização desses eventos, a duração e os participantes.

Para uma comparação entre os *checklists* analisados neste trabalho, os gráficos de abrangência de práticas do *Scrum* são apresentados na Figura 4.11 para o *Nokia Test*, *CRISP Scrum-Checklist* e o *CAS*, respectivamente. O *Nokia Test* possui itens a respeito do *Backlog* do Produto, *Sprint*, Plano de Lançamentos, Gráfico de *Burndown*, *Product Owner*, Time, Fundamentos, Valores e Princípios. No entanto, esse *checklist* não abrange algumas práticas

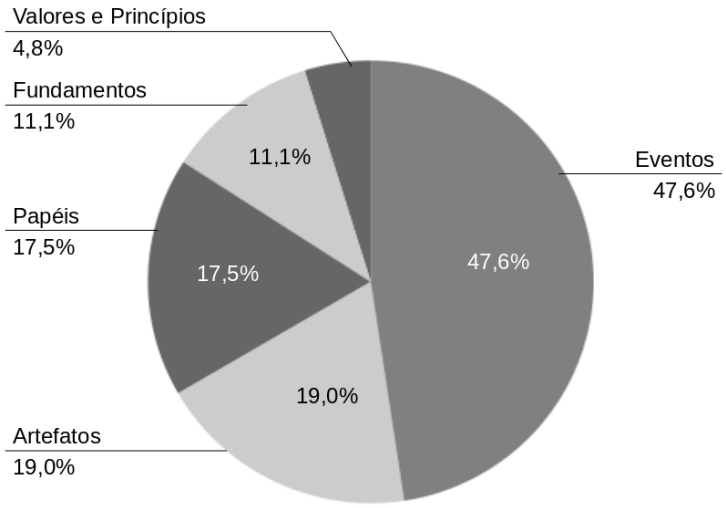


Figura 4.10: Distribuição Percentual de Questões por Categoria - Checklist de Avaliação do *Scrum*

do *Scrum*, como exemplo, não há itens sobre a Retrospectiva, Reunião de Revisão, *Scrum Master*. Já o *CRISP Scrum-Checklist* não abrange diretamente apenas um dos fundamentos (i.e., inspeção) e o *CAS* abrange todas as categorias identificadas.

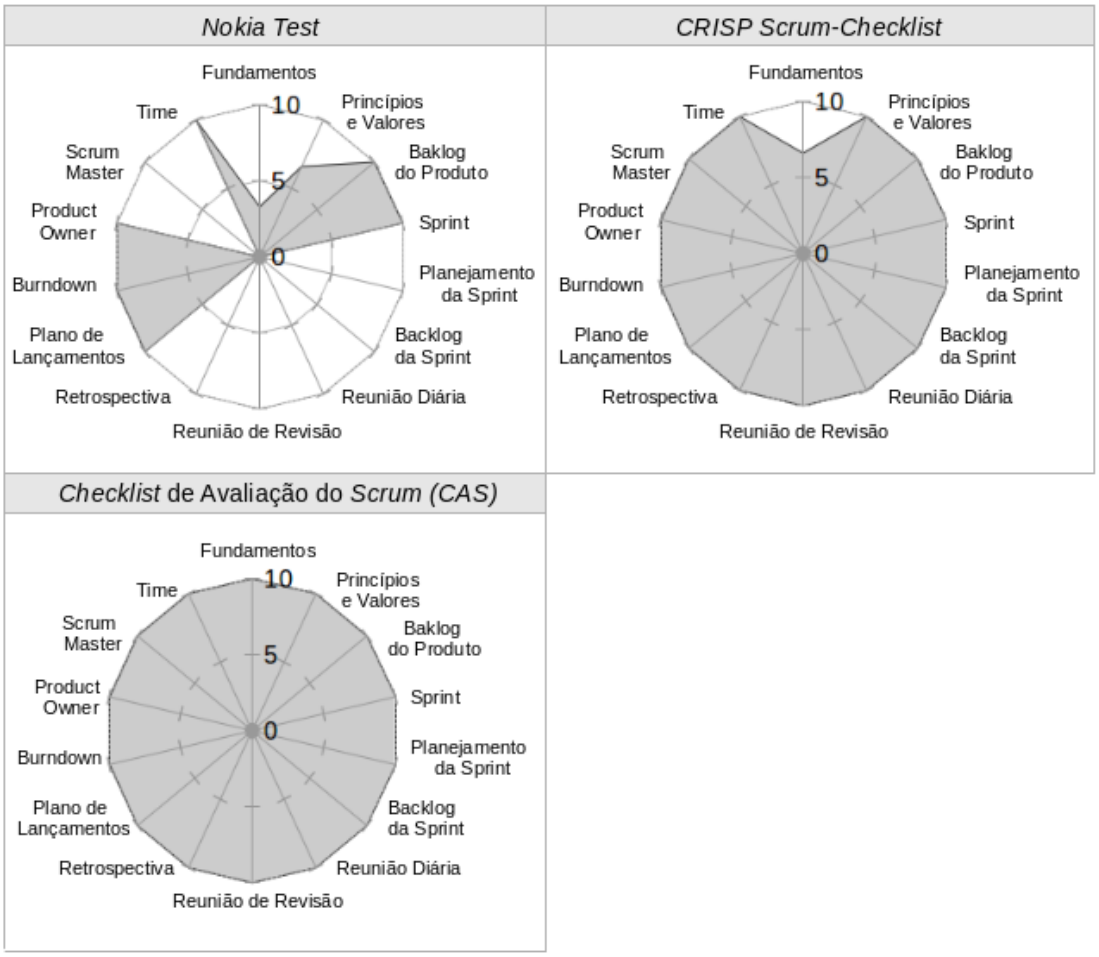


Figura 4.11: Abrangência dos *Checklists* Quanto às Práticas Ágeis do *Scrum*

4.2.5.3 Índice de Medição do Nível do *Scrum*

O Índice de Medição do Nível do *Scrum* (*IMS*) é utilizado para classificar os projetos quanto ao nível de adoção de princípios e práticas do *Scrum*. Esse índice é uma especialização do Índice de Medição Ágil especificado no *Agile Adoption Framework* (*AAF*) (SIDKY et al., 2007), a partir de sua simplificação ao remover as práticas e conceitos não relacionados ao *Scrum*.

Como exemplo dessa simplificação, o *Test Driven Development* (TDD) e a Programação em Pares foram removidos do mapeamento proposto em (SIDKY et al., 2007), pois embora não haja restrições em usá-los com o *Scrum* não são especificados como práticas dessa metodologia. Em resumo, o *Agile DMAIC* proposto neste trabalho adaptou o Índice de Medição Ágil definido por Sidky *et al.* (2007) e não utilizou, por exemplo, os indicadores e demais componentes do *AAF*.

Os níveis definidos no *IMS* são: (Nível 1) *Colaborativo* - promove a comunicação e colaboração entre os envolvidos do projeto; (Nível 2) *Evolutivo* - foca na entrega contínua de *software*; (Nível 3) *Efetivo* - objetiva o desenvolvimento de *software* testado e de alta qualidade, realizado de forma eficiente e efetiva; (Nível 4) *Adaptativo* - estabelece as práticas que atuam para agilizar as respostas às mudanças; e (Nível 5) *Ambiente* - promove um ambiente altamente produtivo.

Os níveis do *IMS* são acessados a partir de práticas agrupadas por princípios ágeis do *Scrum*, conforme mapeamento apresentado na Tabela 4.4. Já o atendimento dessas práticas é verificado a partir da coleta da percepção da equipe com o *CAS*. O mapeamento de práticas do *IMS* aos itens do *CAS* é apresentado na Tabela 4.5. A partir desses mapeamentos é possível classificar o nível do *Scrum* do projeto.

Para verificar o atendimento de uma prática, o indicador do *Scrum* para essa prática é primeiramente calculado (veja o passo 4 da Tabela 4.3). Esse cálculo é realizado a partir da seguinte fórmula:

$$10 * \frac{\sum Prioridades\ dos\ Itens\ Atendidos\ da\ Prática}{\sum Prioridades\ dos\ Itens\ da\ Prática}.$$

O *Agile DMAIC* define que se esse cálculo é superior a 7,5 (>=7,5), então a prática é atendida pelo projeto. Já se esse cálculo é inferior a 7,5 (<7,5), então o *Agile DMAIC* define que a prática não é atendida pelo projeto.

Para verificar o atendimento de um um nível, o indicador do *Scrum* para esse nível é primeiramente calculado. Esse cálculo é realizado a partir da seguinte fórmula:

$$10 * \frac{\sum Prioridades\ dos\ Itens\ Atendidos\ das\ Práticas\ do\ Nível}{\sum Prioridades\ dos\ Itens\ das\ Práticas\ do\ Nível}.$$

O *Agile DMAIC* define que se esse cálculo é superior a 7,5 (>=7,5), então o nível é atendido pelo projeto. Caso o cálculo seja inferior a 7,5 (<7,5), então o *Agile DMAIC* define que o nível não

Tabela 4.4: Os 5 Níveis do Índice de Medição do Nível do *Scrum* Populados com Práticas do Scrum, Adaptado de (SIDKY et al., 2007)

	Princípios Ágeis do *Scrum*				
	Abraçar mudanças para entregar valor ao cliente	*Planejar e entregar software com frequência*	*Focar nas pessoas*	*Excelência técnica*	*Colaboração com o cliente*
Nível 5 **Ambiente**	**[P16]** Iterações orientadas ao cliente, **[P17]** Feedbacks contínuos de sua satisfação				
Nível 4 **Adaptativo**		**[P12]** Entregas menores e frequentes, **[P13]** Planejamento adaptativo		**[P14]** Monitorar o progresso diário	**[P15]** Cliente prontamente acessível
Nível 3 **Efetivo**			**[P10]** Times auto-organizáveis, **[P11]** Comunicação face a face		
Nível 2 **Evolutivo**		**[P7]** Entregas contínuas, **[P8]** Planejamentos em níveis distintos		**[P9]** Monitorar o status da iteração	
Nível 1 **Colaborativo**	**[P1]** Reflexão e ajuste do processo	**[P2]** Planejamento colaborativo	**[P3]** Equipes colaborativas, **[P4]** responsáveis e motivadas	**[P5]** Tarefas executadas de forma voluntária	**[P6]** Compromisso do cliente no trabalho com a equipe

é atendido pelo projeto. Além disso, para o atendimento de um determinado nível, é necessário considerar também todas as práticas dos níveis anteriores.

Como exemplo, o atendimento da prática *Feedbacks contínuos da satisfação do cliente* (P17) é verificada do cálculo do indicador considerando os itens do *CAS* do I_39 a I_43 (P17 <-> I_39 a I_43). Já para acessar o nível 2 é necessário que o projeto atenda as práticas (P1 a P9) e, então, o atendimento desse nível considera todos os itens das práticas P1 a P9.

Tabela 4.5: Mapeamento de Práticas do Índice de Medição do Nível do *Scrum* aos Itens do *Checklist* de Avaliação do *Scrum*

	Prática <–> Itens
Nível 5 **Ambiente**	P17 **<–>** I_39 a I_43
	P16 **<–>** I_02, I_05, I_06, I_32, I_33 e I_49
Nível 4 **Adaptativo**	P15 **<–>** I_06, I_09 e I_44
	P14 **<–>** I_16, I_30, I_57, I_59 a I_65
	P13 **<–>** I_04, I_18, I_36 e I_55
	P12 **<–>** I_01, I_12, I_13, I_19, I_20, I_22, I_23
Nível 3 **Efetivo**	P11 **<–>** I_40, I_43, I_59, I_60 e I_62
	P10 **<–>** I_25 a I_29, I_34, I_35 e I_56
Nível 2 **Evolutivo**	P9 **<–>** I_24, I_37, I_63 e I_65
	P8 **<–>** I_04, I_11, I_49 e I_55
	P7 **<–>** I_19 e I_20
Nível 1 **Colaborativo**	P6 **<–>** I_09, I_38, I_39 e I_40
	P5 **<–>** I_10 e I_56
	P4 **<–>** I_07, I_48 e I_54
	P3 **<–>** I_31, I_52 e I_54
	P2 **<–>** I_8, I_17, I_35, I_55 e I_57
	P1 **<–>** I_03, I_41, I_45, I_46 e I_47

4.3 Comparativo das Versões da Abordagem

A nova versão do *SLeSS* proposta nesta dissertação foi elaborada com um foco na melhoria do uso de princípios e práticas do *Scrum* a partir de técnicas do *Lean Six Sigma* (*LSS*) no desenvolvimento e customização de *software* para DMs. As principais alterações propostas nesta dissertação para o aprimoramento da abordagem são descritas a seguir:

- *Mecanismos de Integração* - as mudanças nos mecanismos de integração entre as versões do *SLeSS* são apresentadas na Figura 4.12. A versão inicial possui 6 mecanismos (Mecanismo 1 ao Mecanismo 6), que foram condensados em 4 mecanismos e 3 práticas na nova versão da abordagem, onde também foram adicionadas mais 5 novas práticas, conforme explicado a seguir.

  Os mecanismos 1, 5 e 6 da versão inicial permaneceram no *SLeSS 2.0*. Já os mecanismos 2, 3 e 4, por estarem relacionados ao uso de técnicas e ferramentas do *LSS* no *Scrum*, foram agrupados como práticas (P1 a P3) do novo mecanismo 2 do *SLeSS 2.0*, juntamente com outras 5 práticas identificadas para esse mecanismo (P4 a P8). Além dessas alterações e da melhoria da documentação desses mecanismos, o *Agile DMAIC* proposto está contido no novo mecanismo 4.

  Essa reestruturação visa estabelecer mecanismos de integração fortemente relacionados à combinação do *Scrum* ao *LSS*, ressaltando as estratégias da abordagem como a execução iterativa e incremental do *DMAIC*, a combinação de técnicas e práticas do *Scrum* e *LSS*, a

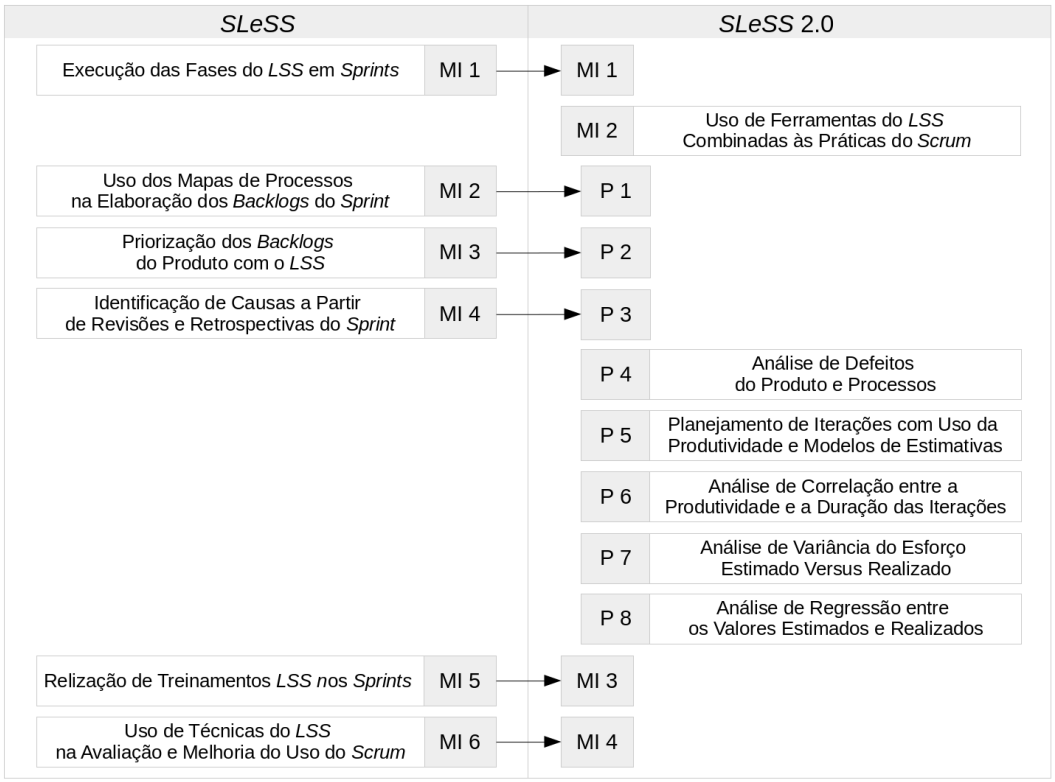


Figura 4.12: Mudanças nos Mecanismos de Integração das Versões do *SLeSS*

capacitação da equipe do projeto na identificação de causas de problemas e a necessidade de avaliação e melhoria do uso de princípios e práticas ágeis para o aumento de agilidade da gestão de projetos e desenvolvimento de *software*. Além disso, com a redução do número de mecanismos e o maior alinhamento às estratégias da abordagem, há um ganho de simplicidade na comunicação e entendimento dos mesmos nessa nova versão.

- *Agile DMAIC* - Novo método para o mecanismo de integração relativo à avaliação e melhoria do uso de princípios e práticas do *Scrum* nos projetos;

- *Fluxo de Implantação* - no fluxo de implantação da abordagem, em sua versão inicial, a implantação do *Scrum* é realizada em todos os projetos priorizados e, só então, o *LSS* é implantado nesses projetos. Em seguida, os mecanismos de integração são adotados para a implantação propriamente dita do *SLeSS*.

  Já no *SLeSS* 2.0, a implantação da abordagem é realizada por projeto. Logo, uma vez identificados os projetos priorizados para a implantação, um deles é selecionado como projeto piloto, que passa então pela implantação do *Scrum*, *LSS* e finalmente pela adoção dos mecanismos de integração do *SLeSS* 2.0.

  Ao final, as lições aprendidas são identificadas e podem ser utilizadas no ajuste do planejamento de implantação. Um novo projeto pode ser selecionado para um novo ciclo de implantação do *SLeSS* 2.0, se beneficiando, dessa forma, tanto do aprendizado da implantação do *Scrum* e *LSS*, quanto da implantação do próprio *SLeSS* 2.0. Além disso, na nova versão, o fluxo de implantação é apresentado de forma gráfica, facilitando o entendimento do mesmo.

- *Documentação* - A documentação da abordagem foi revisada e ajustada para generalizá-la ao desenvolvimento e customização de *software* para DMs, enquanto a versão inicial da abordagem está restrita apenas à customização de *software*.

  Além disso, foram também melhor documentados os seguintes: i) *Processo de Execução* - a partir desse fluxo e da visão geral da abordagem, é possível identificar as fases, atividades, papéis, entradas e saídas do processo de execução do *SLeSS* 2.0. ii) *Papéis* - a versão inicial da abordagem não deixa explícito quais são todos os papéis da abordagem e, para suprir essa deficiência, a nova versão detalha os papéis utilizados e o mapeamento realizado entre os papéis do *Scrum* e *LSS*.

4.4 Considerações Finais

Este capítulo apresentou o *Scrum Lean Six Sigma* (*SLeSS*) 2.0, descrevendo o contexto da concepção dessa nova versão da abordagem, a sua definição, fundamentos, princípios e valores, processo de execução, mecanismos de integração, papéis e o processo para sua implantação. Além disso, o *Agile DMAIC*, método proposto neste trabalho para a evolução da abordagem quanto à avaliação e melhoria do uso dos princípios e práticas do *Scrum* nos projetos foi também apresentado. O comparativo entre as versões do *SLeSS* identificou as principais mudanças propostas nessa nova versão.

Os trabalhos relacionados elencados no Capítulo 3 tiveram uma contribuição direta no desenvolvimento da nova versão da abordagem, *SLeSS* 2.0, por exemplo:

- *Mecanismos de Integração* - as novas práticas adicionadas ao segundo mecanismo foram identificadas a partir do trabalho de Hashmi e Baik (2008), que apresenta um mapeamento de técnicas e ferramentas do *Six Sigma* ao ciclo de vida do *XP*. A análise do mapeamento proposto por Hashmi e Baik (2008) possibilitou a inclusão de 5 novas práticas (P4 a P8) ao segundo mecanismo do *SLeSS* 2.0.

- *Papéis* - os papéis da abordagem foram revisados a partir da análise do mapeamento de papéis proposto por Roriz (2010) para a combinação do *Scrum* e *Six Sigma*. A combinação proposta no *SLeSS* 2.0, na qual o *Product Owner* e o *Scrum Master* assumem também o papel de *Green Belt*, frente a análise apresentada no Capítulo 3 para o trabalho do Roriz (2010), se mostrou mais assertiva em relação à definição, atribuições e responsabilidades desses papéis, bem como em relação ao nível de atuação organizacional dos mesmos.

- *Agile DMAIC* - Os três componentes do *Agile DMAIC* foram elaborados com base nos trabalhos de Sidky *et al.* (2007), Vode e Sutherland (2008), Kniberg (2009) e Cunha *et al.* (2011).

Por fim, a documentação dessa nova versão da abordagem pode ser complementada a partir de informações apresentadas no capítulo seguinte, que apresenta a avaliação e os resultados do uso dessa abordagem em 7 projetos reais.

5 APLICANDO O *SLESS* 2.0 EM PROJETOS REAIS

Este capítulo apresenta uma avaliação qualitativa e quantitativa da aplicação do *SLeSS* 2.0 em projetos reais, uma vez que avalia as informações e também as traduz em números, utilizando técnicas estatísticas para serem classificadas e analisadas (SEVERINO, 2007).

Essa avaliação é realizada através de metodologia utilizada na definição e execução de um estudo de caso (YIN, 2010). Esse estudo foi realizado através de um processo linear e iterativo, que possui as seguintes etapas: (i) *Planejamento*, que consiste na elaboração do plano de desenvolvimento do estudo de caso; (ii) *Projeto*, relacionado à definição do objetivo e das metas da avaliação; (iii) *Preparação*, consiste na definição do escopo, dos meios e das ferramentas para a realização de avaliações do objeto do estudo; (iv) *Coleta*, necessária para a amostragem de informações para as análises; e (v) *Análise e Compartilhamento*, abrangem as análises qualitativas e quantitativas e o compartilhamento de seus resultados com os envolvidos.

O objetivo da avaliação é analisar as principais melhorias propostas no *SLeSS* 2.0 e possibilitar inclusive que sejam aprimoradas durante a execução desse estudo, a partir da análise de seus resultados e do *feedback* das equipes envolvidas. Essa avaliação foi realizada nos projetos desenvolvidos no Grupo de Redes de Computadores, Engenharia de *Software* e Sistemas (*GREat*).

A organização das seções deste capítulo é apresentada na Figura 5.1. Inicialmente, a Seção 5.1 descreve os objetivos do estudo de caso. A Seção 5.2 contextualiza os projetos reais selecionados para o estudo e, em seguida, detalha a aplicação da abordagem em cada um desses projetos, exemplificando a sua utilização e apresentando os resultados das coletas, as análises de problemas e as ações identificadas e implantadas. Os resultados consolidados da aplicação do *SLeSS* 2.0 nos 7 projetos reais são então discutidos na Seção 5.3 e, por fim, a Seção 5.4 apresenta as considerações finais sobre o capítulo.

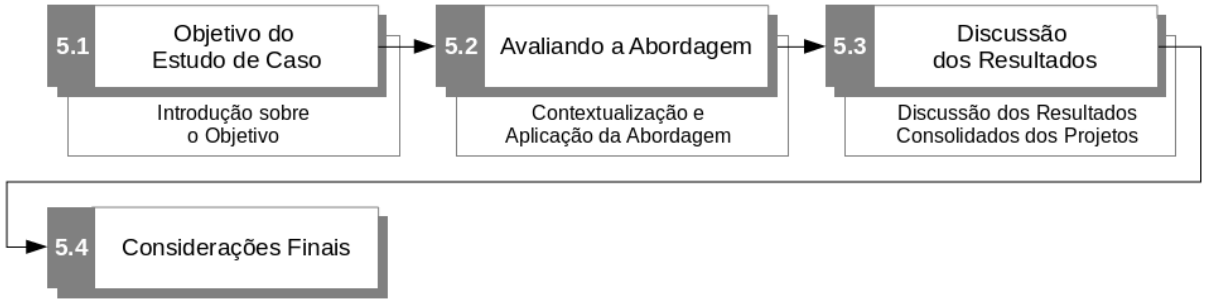


Figura 5.1: Organização das Seções do Capítulo 5

5.1 Objetivo do Estudo de Caso

Uma das principais mudanças na nova versão da abordagem é o *Agile DMAIC*, método proposto nesta dissertação para a evolução do mecanismo de integração do *SLeSS* 2.0 relativo à avaliação e melhoria do uso de princípios e práticas do *Scrum*, conforme mencionado na Seção 4.2.3.

Para avaliar essa nova versão da abordagem, *SLeSS* 2.0, o foco é a aplicação do *Agile DMAIC* em projetos de desenvolvimento e customização de *software* para dispositivos móveis, priorizando a análise dos seguintes critérios:

(i) *Assertividade na identificação de problemas no uso do Scrum* - esse critério avalia o percentual de assertividade da abordagem na identificação de problemas no uso de princípios e práticas do *Scrum* nos projetos.

Nesse estudo, entende-se por assertividade o comportamento da abordagem caracterizado por realizar afirmações sobre a utilização de um princípio ou prática do *Scrum* no projeto, sem a necessidade que essa afirmação esteja correta.

Para facilitar o entendimento desse critério, uma representação da análise da coleta realizada pelo *Agile DMAIC* é apresentada na Figura 5.2. A coleta da percepção da equipe sobre o uso do *Scrum* é realizada a partir do *Checklist* de Avaliação do *Scrum* (*CAS*) (veja a Seção 4.2.5.2).

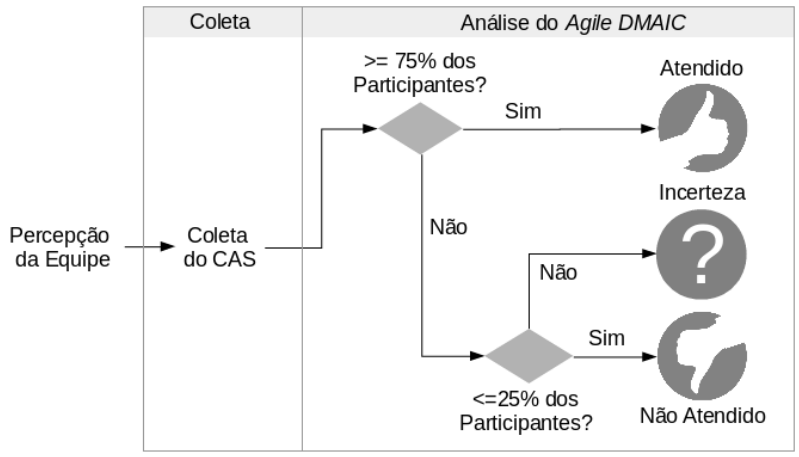


Figura 5.2: Análise da Coleta no *Agile DMAIC*

O *Agile DMAIC* define uma análise da coleta para identificação dos itens que estão ou não atendidos pelo projeto, de acordo com a percepção da equipe (veja o passo 4 da Tabela 4.3). Essa análise considera o número de participantes da coleta, que idealmente é o número de integrantes da equipe.

Para um determinado item, se mais de 75% (>=75%) dos participantes indicam que ele está atendido, o *Agile DMAIC* considera que o item está atendido (veja a Figura 5.2). Por outro lado, o *Agile DMAIC* considera que o item não está atendido quando menos de 25% (<=25%) dos participantes indicam que o item está atendido. Nos demais casos, o

Agile DMAIC sinaliza uma incerteza sobre o atendimento do item no projeto, que deve ser, portanto, solucionada em conjunto com a equipe.

Dessa forma, para a análise desse critério, esse estudo considera o seguinte cálculo:

$$Assertividade = \frac{Total\ de\ Itens\ do\ CAS - Total\ de\ Incertezas}{Total\ de\ Itens\ do\ CAS};$$

(ii) *Exatidão na identificação de problemas no uso do Scrum* - esse critério avalia o percentual de exatidão da abordagem na identificação de problemas no uso de princípios e práticas do *Scrum*.

Nesse estudo, entende-se por exatidão a conformidade das afirmações da abordagem sobre o uso de um princípio ou prática do *Scrum* em relação ao que está realmente em uso no projeto, considerando como referência para esse último uma revisão sobre o uso do princípio ou da prática, realizada através de reunião com o gerente e os líderes do projeto.

Para a análise desse critério, uma etapa de revisão é, portanto, adicionada após a análise do método (veja a Figura 5.3). Essa revisão objetiva confirmar se as afirmações do *Agile DMAIC* estão ou não corretas.

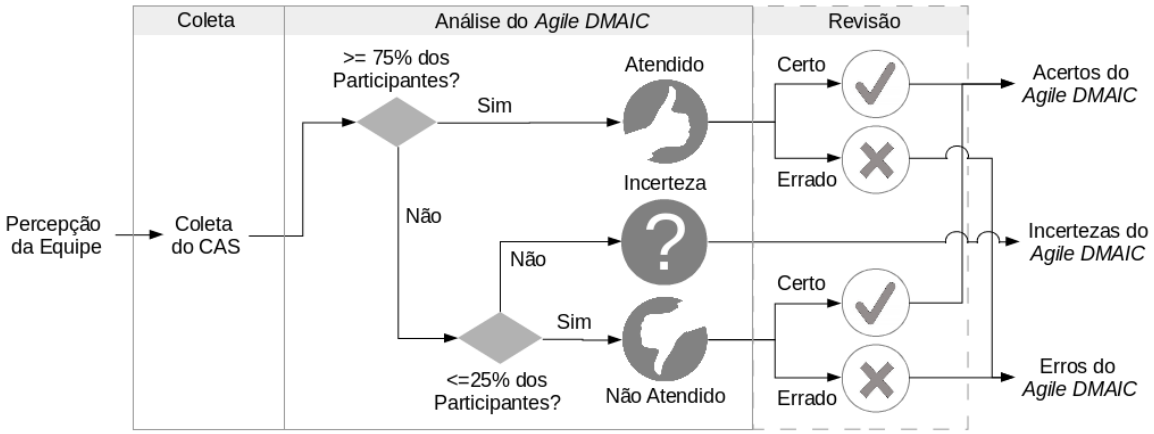


Figura 5.3: Análise e Revisão da Coleta no *Agile DMAIC*

Por exemplo, no caso em que o *Agile DMAIC* sinaliza que o item está atendido e a revisão confirma o mesmo, então esse estudo considera que o *Agile DMAIC* foi exato em sua afirmação. O mesmo ocorre quanto o *Agile DMAIC* indica que o item não está atendido e a revisão também confirma essa informação.

Entretanto, quando a revisão discorda do resultado da análise, então esse estudo considera que o método não foi exato em sua afirmação. Além disso, como não há como analisar a conformidade quando o método sinaliza uma incerteza, essa informação não é considerada no cálculo desse critério.

Dessa forma, para a análise desse critério, esse estudo considera o seguinte cálculo:

$$Exatidão = \frac{Total\ de\ Acertos}{Total\ de\ Acertos + Total\ de\ Erros};$$

(iii) *Capacidade de identificar e analisar causas de problemas no uso do Scrum* - esse critério avalia como a abordagem fornece meios para a identificação e a análise de causas dos problemas no uso do *Scrum*; e

(iv) *Capacidade de priorizar as ações de melhoria para a evolução da agilidade do projeto* - esse critério avalia como a abordagem auxilia as equipes na evolução da agilidade dos projetos a partir da priorização de ações de melhoria.

Diante da restrição de tempo para uma implantação completa da nova abordagem nos projetos do estudo de caso, por exemplo, a adoção de todos os mecanismos de integração, a aplicação do *Agile DMAIC*, que utiliza os principais conceitos da abordagem, fornece os insumos necessários para a avaliação da nova versão da abordagem, uma vez que é a principal mudança proposta nessa versão.

5.2 Sobre os Projetos Reais

Os projetos reais desse estudo de caso foram desenvolvidos em laboratórios de Pesquisa, Desenvolvimento e Inovação (P&D&I) do *GREat* e foram selecionados por serem voltados à Pesquisa e Desenvolvimento (P&D) de aplicações e customizações de *software* para DMs. Um resumo desses projetos e de suas principais dificuldades é apresentado na Tabela 5.1. Essas informações foram obtidas com os gerentes e líderes através de questionário aplicado via *WEB*.

Tabela 5.1: Resumo dos Projetos do Estudo de Caso

Projeto	Tipo	Duração	Equipe	Complexidade	Principais Dificuldades
App1	D	9 meses	8	Alta	Comunicação com o cliente
App2	P&D	9 meses	7	Alta	Comunicação com cliente, complexidade da tecnologia, frequentes mudanças de escopo
App3	P&D	9 meses	8	Alta	Comunicação com o cliente e complexidade da tecnologia
App4	D	2 anos	6	Média	Comunicação com o cliente, dificuldade de concepção do produto, frequentes mudanças de prioridade e escopo
App5	P&D	2 anos	26	Alta	Frequentes mudanças de escopo e prazo, grande dependência de informações do produto, alta rotatividade da equipe
App6	P&D	6 meses	6	Alta	Complexidade técnica da pesquisa e do *design* de interfaces (grande quantidade de dados de entrada)
App7	P&D	9 meses	10	Alta	Comunicação com o cliente, complexidade da tecnologia

As equipes desses projetos são alocadas geograficamente em um mesmo sítio e, dessa forma, não há equipes remotas, entretanto, os clientes são todos remotos. Essas equipes

são formadas por pesquisadores e desenvolvedores de áreas da computação e engenharia e, em sua maioria, são integralmente alocados aos projetos.

A informação da complexidade dos projetos está relacionada ao uso simultâneo de várias tecnologias, ao elevado número de envolvidos e à necessidade de conhecimentos técnicos especializados. A informação de duração consiste na duração planejada dos projetos.

Os projetos desse estudo são voltados à P&D de aplicações para *smartphones*, com exceção do App1 que realiza apenas o desenvolvimento (D) de aplicações para *smartphones* e do App5 que é o único voltado à P&D de customização de *software* para celulares e *smartphones*. Além disso, esses projetos são de complexidade alta, com exceção do App4, que possui uma complexidade média.

5.3 Aplicando a Abordagem

Antes da adoção da abordagem, os projetos *App1*, *App2*, *App3* e App7 já utilizavam o *Scrum*, enquanto os demais passaram a adotá-lo a partir da utilização da abordagem proposta nesta dissertação, com exceção do projeto *App5* que até o momento não se decidiu por utilizá-lo. A inclusão desse projeto no estudo de caso é realizada, então, para analisar preliminarmente se a abordagem consegue diferenciar adequadamente entre os projetos que usam o *Scrum* e aqueles que não o utilizam.

A adoção da abordagem foi antecedida por treinamentos teóricos e práticos aos principais envolvidos. Nesses treinamentos, os fundamentos, princípios, valores, papéis, artefatos e eventos do *Scrum* foram apresentados e discutidos, e algumas práticas foram realizadas para a fixação dos novos conceitos.

Uma visão da linha de tempo com os marcos de início dos projetos, dos treinamentos, da adoção do *Scrum* e do *Agile DMAIC* é apresentada na Figura 5.4.

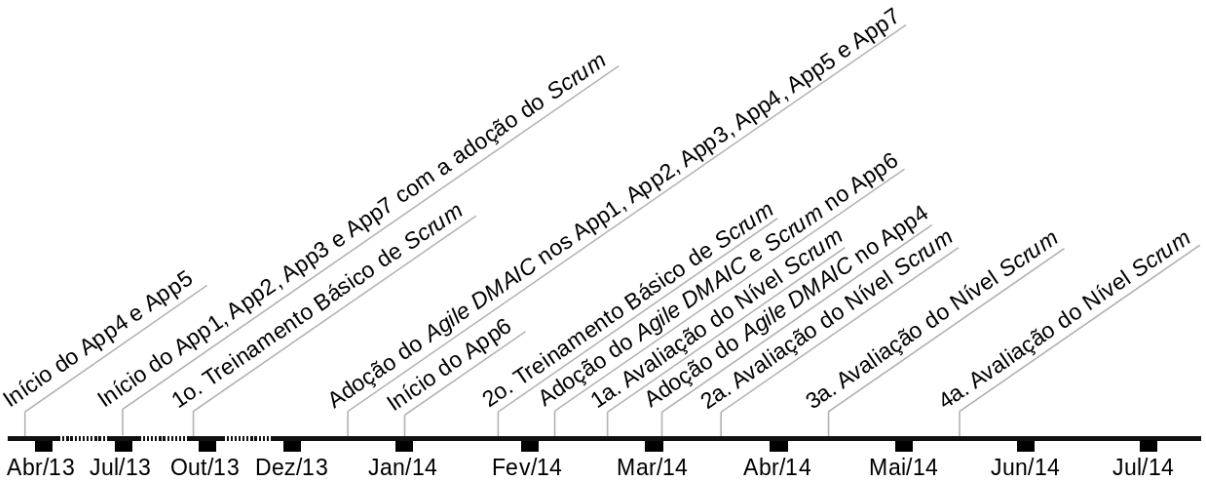


Figura 5.4: Linha de Tempo dos Treinamentos, Adoção do *Scrum* e *Agile DMAIC* nos Projetos

O projeto *App6*, por exemplo, iniciou em janeiro de 2014. A equipe desse projeto participou de treinamento no *Scrum*, adotou essa metodologia e o *Agile DMAIC* em fevereiro e realizou as avaliações do nível do uso do *Scrum* em fevereiro, março, abril e maio de 2014.

Os resultados do uso do *SLeSS* 2.0 nos 7 projetos desse estudo de caso são apresentados nas seções seguintes. Em cada projeto são apresentados exemplos e resultados do uso da abordagem e ao final são discutidos os resultados consolidados desses projetos.

5.3.1 Projeto App1

Desde o seu início em julho de 2013, o projeto App1 foi desenvolvido com o *Scrum*. O nível de conhecimento, o tempo de experiência e o interesse da equipe no uso dessa metodologia foram coletados a partir de um questionário aplicado via *WEB* à equipe.

Um resumo das informações dessa coleta é apresentado na Figura 5.5. Essa equipe possui um conhecimento satisfatório no *Scrum* (veja a escala utilizada na avaliação do conhecimento na Figura 5.5), a maior parte de seus integrantes tem até um 1 ano de experiência com essa metodologia e demonstram um interesse mediano na mesma.

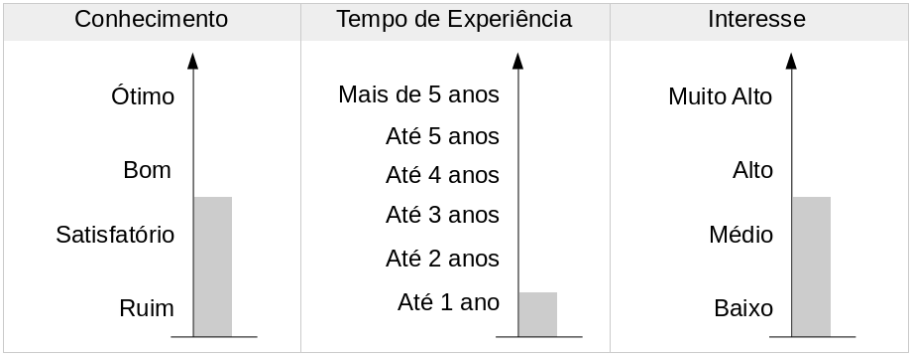


Figura 5.5: Conhecimento, Experiência e Interesse da Equipe App1 no *Scrum*

Como exemplo da utilização da abordagem, os diagramas de Causa e Efeito e de Pareto referentes ao mês de janeiro são apresentados nas Figuras 5.6 e 5.7, respectivamente.

Esses diagramas foram elaborados com o *LibreOffice*[1] na fase *Analisar* do *DMAIC* (veja os passos 5, 6 e 7 da Tabela 4.3) a partir da coleta com o *CAS*, aplicado via *WEB* à equipe.

O diagrama de Causa e Efeito é utilizado para discutir com a equipe os principais problemas e causas do nível atual do uso do *Scrum* no projeto. Já a visão das principais vulnerabilidades desse nível, priorizadas de acordo com a customização do *CAS*, pode ser obtida com o uso do diagrama de Pareto. A customização do *CAS* é realizada para alinhar com a equipe os itens que são utilizados na avaliação do uso do *Scrum* e a priorização desses itens considera o entendimento dessa equipe sobre a importância dos princípios e práticas ágeis avaliados no projeto.

[1] http://www.libreoffice.org/

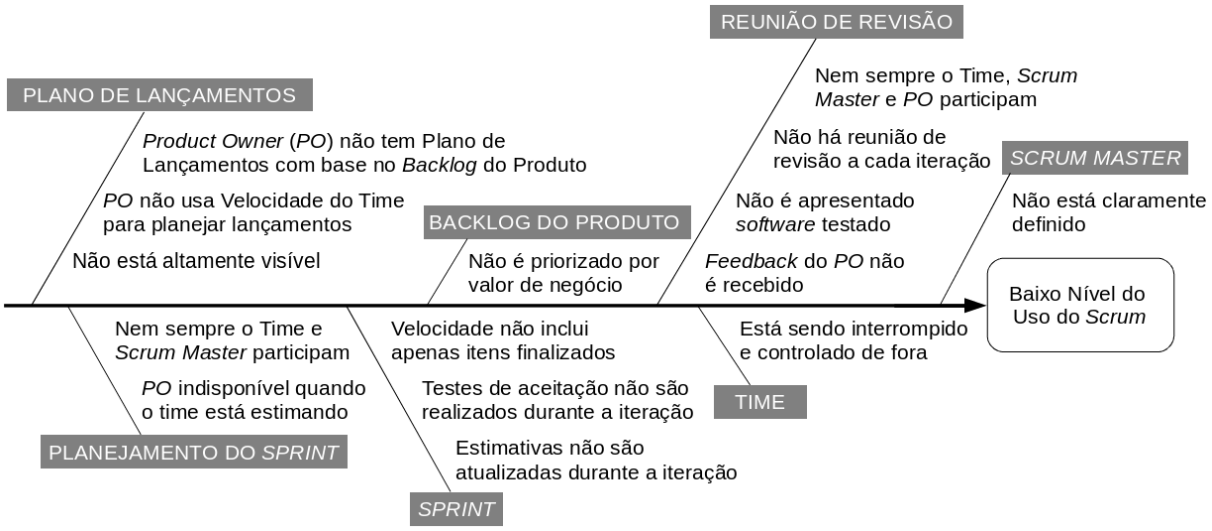


Figura 5.6: Diagrama de Causa e Efeito do Projeto App1 em Janeiro de 2014

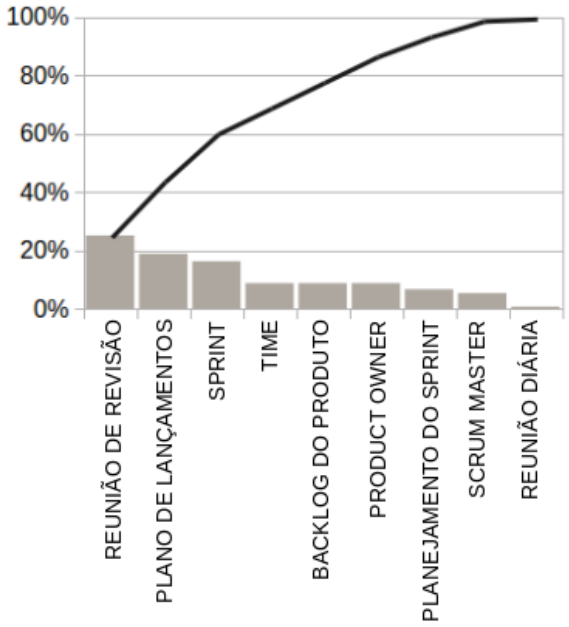


Figura 5.7: Diagrama de Pareto do Projeto App1 em Janeiro de 2014

Com esses diagramas de janeiro, por exemplo, é possível identificar que a equipe do projeto App1 pode priorizar os seguintes itens:

- *Reuniões de Revisão* - em janeiro, o projeto App1 não realizou a Revisão do *Sprint*, evento no qual o *software* testado, resultado direto das atividades do *Sprint*, é apresentado ao *Product Owner* (*PO*). Nessa revisão, o Time pode receber o *feedback* desse *PO* para alinhar melhor os próximos resultados às necessidades e expectativas do cliente, uma vez que o *PO* representa a visão desse cliente;

- *Plano de Lançamentos* - o projeto App1 não possuía em janeiro um Plano de Lançamentos e esse plano pode ser elaborado com base no *Backlog* do Produto utilizando a velocidade (capacidade de produção) do time;
- *Testes de Aceitação* - os testes de aceitação não foram realizados em Janeiro;
- *Time* - foi identificado que o Time está sendo interrompido e controlado de fora e, portanto, esse projeto pode priorizar a identificação e solução de causas desse problema, o qual interfere diretamente na produtividade da equipe; e
- *Backlog* do Produto - em janeiro, o projeto possuía um *Backlog* do Produto, entretanto, o mesmo não estava sendo priorizado por valor de negócio. No *Scrum* a priorização dos requisitos deve considerar as necessidades do cliente e essa priorização deve ser realizada pelo *Product Owner*, que é responsável por garantir o valor do trabalho do Time a cada *Sprint*.

A partir desses problemas, em uma análise realizada em reunião com o gerente e os líderes foi possível identificar que uma das causas está relacionada diretamente à atuação do *Product Owner* (*PO*) no projeto. Por exemplo, o Plano de Lançamentos e a priorização do *Backlog* do Produto são responsabilidades desse *PO*. Além disso, os testes de aceitação são geralmente especificados com a ajuda dele. Dessa forma, pode-se identificar ações para melhorar a atuação desse *PO* no projeto.

Consonante com esses resultados, os gráficos das avaliações do projeto *App1* são apresentados na Figura 5.8. Esses gráficos foram elaborados com o uso do *LibreOffice*. Essas avaliações mostram o resultado do nível do indicador do *Scrum* para cada uma das seções do *CAS* (veja a Figura 4.9). O cálculo desse nível é realizado a partir do passo 4, *Medir Nível do Scrum*, do *DMAIC* (veja a Tabela 4.3).

As informações da coleta de janeiro sobre o Time são apresentadas na Tabela 5.2 para exemplificação do cálculo do indicador do nível do *Scrum* relativo ao Time no projeto App1. Os resultados da coleta, análise, revisão e priorização dos itens I_25 a I_31 do *CAS* utilizados na avaliação do Time são apresentados nessa tabela. Nessa avaliação de janeiro, 7 dos 8 integrantes da equipe responderam ao *checklist*, obtendo, portanto, 87,5% de participação na coleta.

Tabela 5.2: Informações da Coleta sobre o Time do App1 em Janeiro de 2014

Itens do *CAS*	**Coleta**	**Análise**	**Revisão**	**Prioridade**
I_25. Não é interrompido ou controlado de fora	1	0	0	34
I_26. Costuma entregar o que foi prometido	7	1	1	34
I_27. Tem as competências necessárias	7	1	1	21
I_28. Membros não ficam dedicados a papéis específicos	4	-1	1	13
I_29. Máximo de 9 pessoas	2	0	1	13
I_30. Conhece os principais impedimentos	7	1	1	8
I_31. Possui e respeita a definição de “pronto” da iteração	7	1	1	21

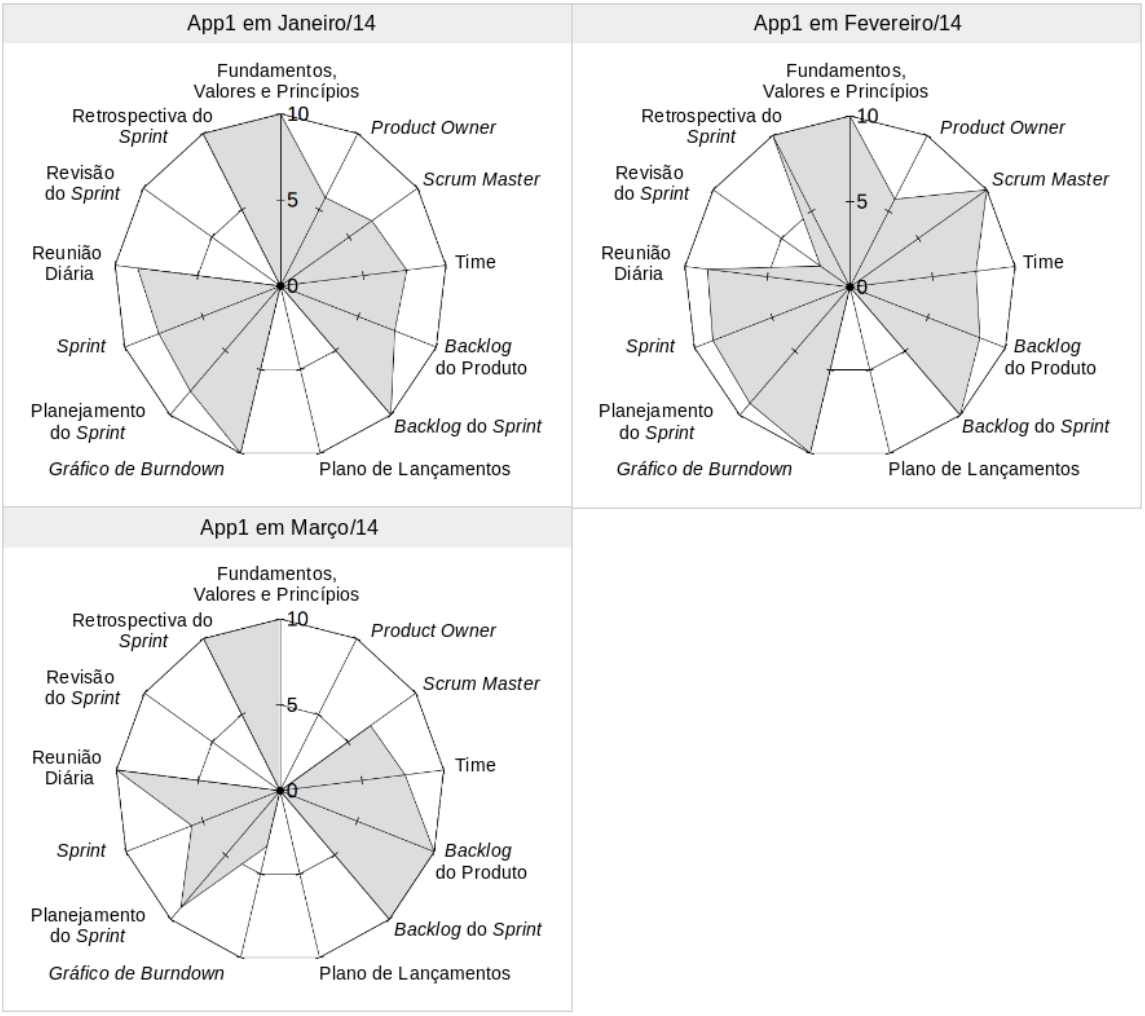


Figura 5.8: Resultado das Avaliações do Projeto App1

Na Tabela 5.2, a coluna *Coleta* contém a quantidade de votos da equipe sobre o atendimento a cada um dos itens. A coluna *Análise* representa o resultado da análise estabelecida pelo *Agile DMAIC* quanto a definição de itens atendidos e não atendidos. O valor 1 nessa coluna representa que o método sugeriu que o item está atendido, o valor 0 representa que o método sugeriu que o item não está atendido e -1 representa que o método não conseguiu definir o estado desse item, e essa incerteza precisa então ser resolvida com a equipe. A coluna *Revisão* representa o resultado da revisão pelo gerente e pelos líderes, confirmando se o item realmente está sendo atendido (valor 1) ou não (valor 0). Essa revisão é realizada nesse estudo de caso para avaliar a exatidão do método. Por fim, a coluna *Prioridade* contém valores de 1 a 55, seguindo a sequência de *Fibonacci*. Quanto maior o valor, maior é a importância que a equipe considera sobre o item avaliado em relação aos seus benefícios para o projeto.

Um exemplo de como é calculado o indicador do nível do *Scrum* para o Time, utilizando as informações da Tabela 5.2, é apresentado a seguir:

$$10 * (\frac{34+21+13+13+8+21}{34+34+21+13+13+8+21}) = 7,6.$$

Esse cálculo é a soma das prioridades dos itens atendidos dividida pela soma de todas as prioridades e ao final multiplicado por 10. Logo, o nível de uso do papel do Time no App1 obteve 7,6 para o mês de janeiro (veja a Figura 5.8).

De acordo com a análise da equipe, esse projeto obteve impacto em Março por falta de prioridade do *Product Owner* (*PO*). Embora tenham sido identificadas e implementadas ações de melhoria como a definição de um *Scrum Master*, a realização de Reuniões de Retrospectiva e a melhor execução de atividades de planejamento como a priorização do *Backlog* do Produto, a alocação e a realização das estimativas pelo time, o indicador foi impactado pela ausência desse *PO* nas atividades do projeto.

5.3.2 Projeto App2

O projeto App2 iniciou em julho de 2013 e, desde o seu início, foi desenvolvido com o *Scrum*. Um resumo do nível de conhecimento, do tempo de experiência e do interesse dessa equipe no uso do *Scrum* é apresentado na Figura 5.9. Embora sinalize um interesse alto no *Scrum*, essa equipe possui um nível de conhecimento satisfatório, pois seus integrantes tem em sua maioria até 1 ano de experiência com essa metodologia.

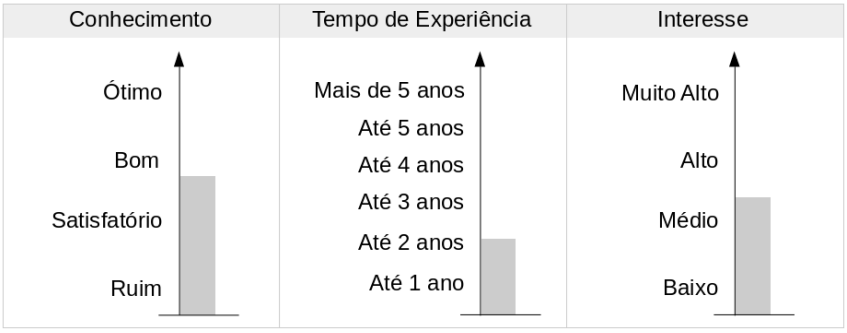


Figura 5.9: Conhecimento, Experiência e Interesse da Equipe App2 no *Scrum*

Como exemplo da utilização da abordagem, as informações da coleta de Março sobre o Planejamento do *Sprint* são apresentadas na Tabela 5.3. Os resultados da coleta, análise, revisão e priorização dos itens I_04 a I_11 do *CAS* utilizados na avaliação do Planejamento do *Sprint* são apresentados nessa tabela.

Nessa avaliação de Março, todos os 7 integrantes da equipe responderam ao *checklist*, obtendo, portanto, 100% de participação na coleta. Por exemplo, todos responderam que há planejamento do *Sprint* a cada iteração (veja o valor 7 na coluna *Coleta* da primeira linha da Tabela 5.3).

Dessa forma, de acordo com o cálculo do indicador do nível do *Scrum*, esse item, I_04, está sendo atendido pela equipe (veja o valor 1 na coluna *Análise*).

Após revisão com o gerente e os líderes do App2, foi ratificado que a análise do *Agile DMAIC* está correta e, portanto, realmente o item I_04 foi atendido em Março (veja o valor 1 na coluna *Revisão*). Além disso, esse item tem prioridade 34 (veja a coluna *Prioridade*),

Tabela 5.3: Informações da Coleta sobre o Planejamento do *Sprint* do App2 em Março de 2014

Itens do *CAS*	Coleta	Análise	Revisão	Prioridade
I_04. Há planejamento do *Sprint* a cada iteração	7	1	1	34
I_05. Utiliza *Backlog* do Produto atualizado	3	-1	0	21
I_06. *Product Owner* satisfeito com as prioridades da iteração	2	0	0	21
I_07. Time acredita no planejamento realizado	5	1	1	21
I_08. Time e *Scrum Master* participam	6	1	1	13
I_09. *Product Owner* disponível quando o time está estimando	0	0	0	13
I_10. Tarefas são estimadas somente pelo time	6	1	1	13
I_11. Itens do *Backlog* do Produto são decompostos em tarefas	4	-1	1	5

representando que a equipe entende a importância da realização do Planejamento do *Sprint* a cada iteração.

Outro exemplo, o item I_11 avalia se os itens do *Backlog* do Produto estão sendo decompostos em tarefas durante o Planejamento do *Sprint* (veja a última linha da Tabela 5.3). Esse item tem baixa prioridade (P_5), obteve 4 votos, indicando que a equipe pode está em dúvida sobre o seu atendimento no mês de Março (veja o valor -1 na coluna *Análise*). Após revisão com o gerente e os líderes, chegou-se ao consenso de que este item está sendo atendido e que há a necessidade de alinhar essa informação com a equipe. Já o item I_09, sobre a disponibilidade do *Product Owner* (*PO*) (veja a sexta linha da Tabela 5.3), ele não recebeu nenhum voto, sugerindo que toda a equipe avalia que o *PO* não está disponível enquanto o time estima os itens de *backlog*.

Um exemplo de como é calculado o indicador do nível do *Scrum* para o Planejamento do *Sprint*, utilizando as informações da Tabela 5.3, é apresentado a seguir:

$$10 * (\frac{34+21+13+13+5}{34+21+21+21+13+13+13+5}) = 6,1.$$

Logo, o nível do uso da prática de Planejamento do *Sprint* no App2 obteve 6,1 para o mês de Março.

Os gráficos das avaliações do projeto *App2* são apresentados na Figura 5.10. Essas avaliações e a análise já apresentada podem ser revisados com a equipe, que pode identificar possíveis causas para os problemas apontados. Por exemplo, uma das causas pode ser a atuação do *Product Owner* (*PO*), pois ele é o responsável por manter o *Backlog* do Produto atualizado e por sinalizar à equipe se está satisfeito com as prioridades da iteração. O *PO* também precisa estar disponível para eventuais dúvidas do Time durante as estimativas de esforço dos itens do *backlog*.

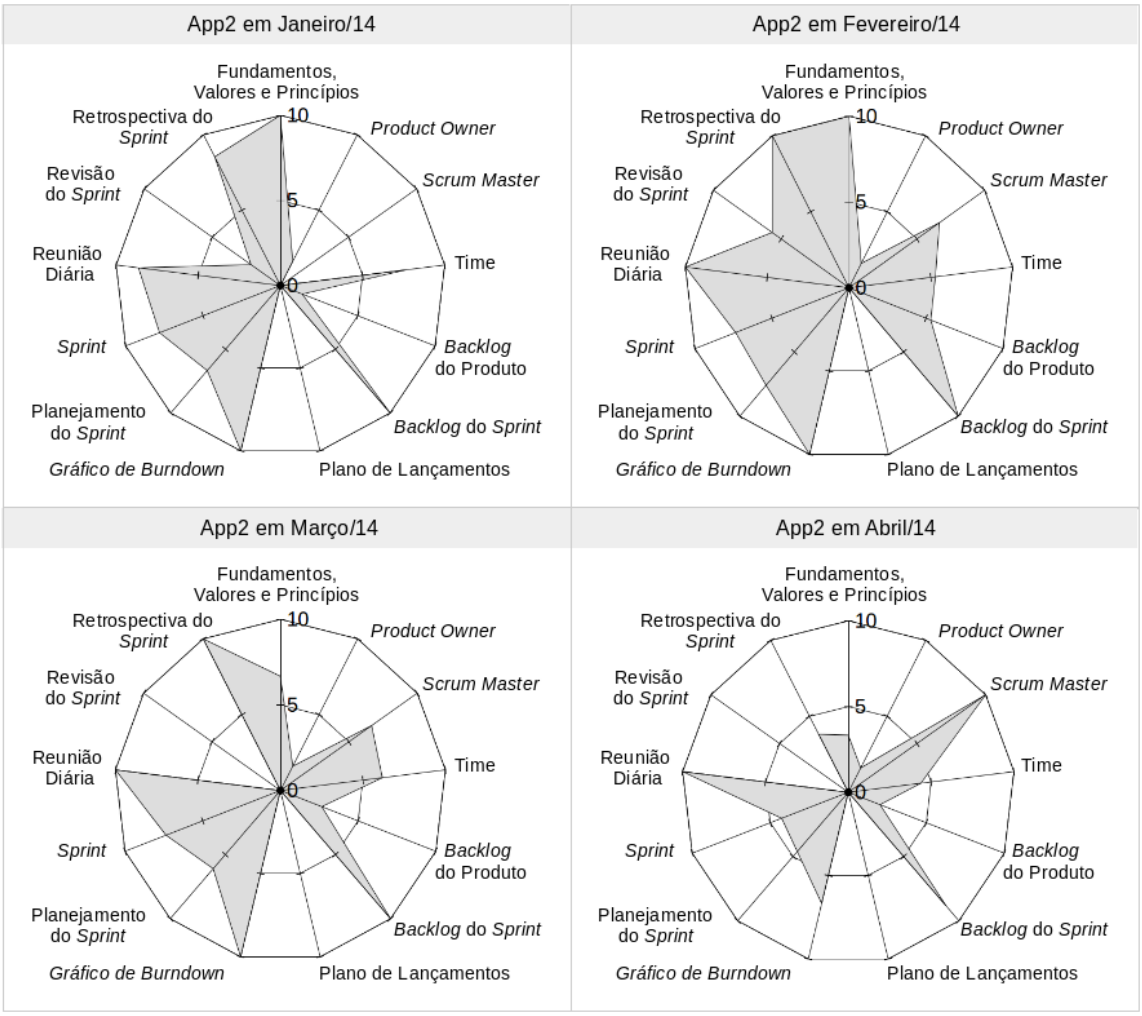
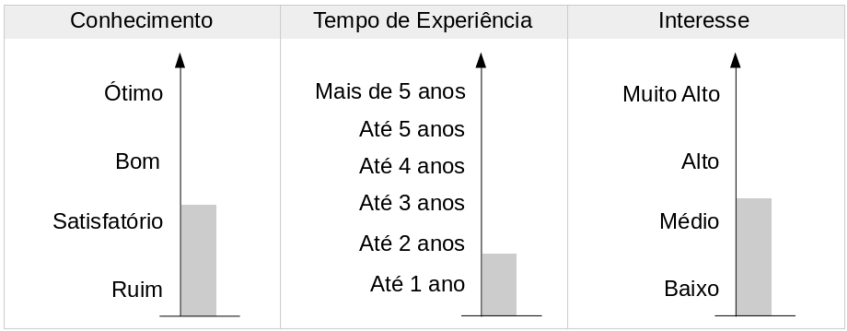


Figura 5.10: Resultado das Avaliações do Projeto App2

5.3.3 Projeto App3

O projeto App3 foi desenvolvido com o *Scrum* desde o seu início em julho de 2013. O nível de conhecimento, o tempo de experiência e o interesse dessa equipe no uso dessa metodologia são apresentados na Figura 5.11.

Figura 5.11: Conhecimento, Experiência e Interesse da Equipe App3 no *Scrum*



A equipe App3 possui um conhecimento satisfatório no *Scrum*, seus integrantes tem em sua maioria até 1 ano de experiência com essa metodologia e sinalizam um interesse mediano na mesma.

Como exemplo do uso da abordagem nesse projeto, o indicador do nível do *Scrum* pode ser acompanhado pela equipe durante as retrospectivas do *Sprint* a partir da tabulação de seus resultados conforme é apresentado na Tabela 5.4.

Tabela 5.4: Acompanhamento do Indicador do Nível do *Scrum* no Projeto App3

	Janeiro	**Fevereiro**	**Março**	**Abril**
Valores, Princípios e Fundamentos	6,67	10	10	10
Product Owner	0	5,8	5,8	1,6
Scrum Master	0	10	10	10
Time	7,64	7,08	7,64	7,64
Backlog do Produto	7,34	1,41	7,34	4,69
Backlog do *Sprint*	10	10	10	10
Plano de Lançamentos	0	0	0	0
Gráfico de *Burndown*	10	10	10	10
Planejamento do *Sprint*	8,16	7,59	7,59	7,59
Sprint	8,37	6,18	6,18	8,09
Reunião Diária	7,27	10	10	10
Reunião de Revisão	0	0	0	5,67
Retrospectiva	10	10	10	10
Indicador do Nível do *Scrum*	**6,18**	**6,3**	**6,92**	**7,23**

O resultado do cálculo desse indicador do nível para cada uma das seções do *CAS* nos meses de janeiro até abril é apresentado na Tabela 5.4. Para cada seção, esse cálculo é a razão entre a soma das prioridades dos itens atendidos dessa seção pela soma das prioridades de todos os itens da seção e ao final multiplicado por 10.

De forma semelhante, para calcular o indicador do nível do *Scrum* num determinado mês, basta considerar nesse cálculo a coleta de todos os itens do *checklist*. Os resultados das avaliações desse indicador para o projeto App3 são apresentados na Figura 5.12.

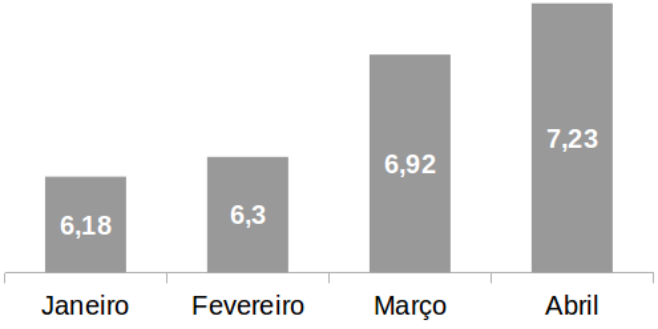


Figura 5.12: Acompanhamento do Indicador do Nível do *Scrum* no Projeto App3

As mesmas informações da Tabela 5.4 são agora apresentadas de forma gráfica na Figura 5.13. A partir dessas informações (nas Figuras 5.12 e 5.13) é possível verificar a evolução do uso do *Scrum* no projeto *App3*.

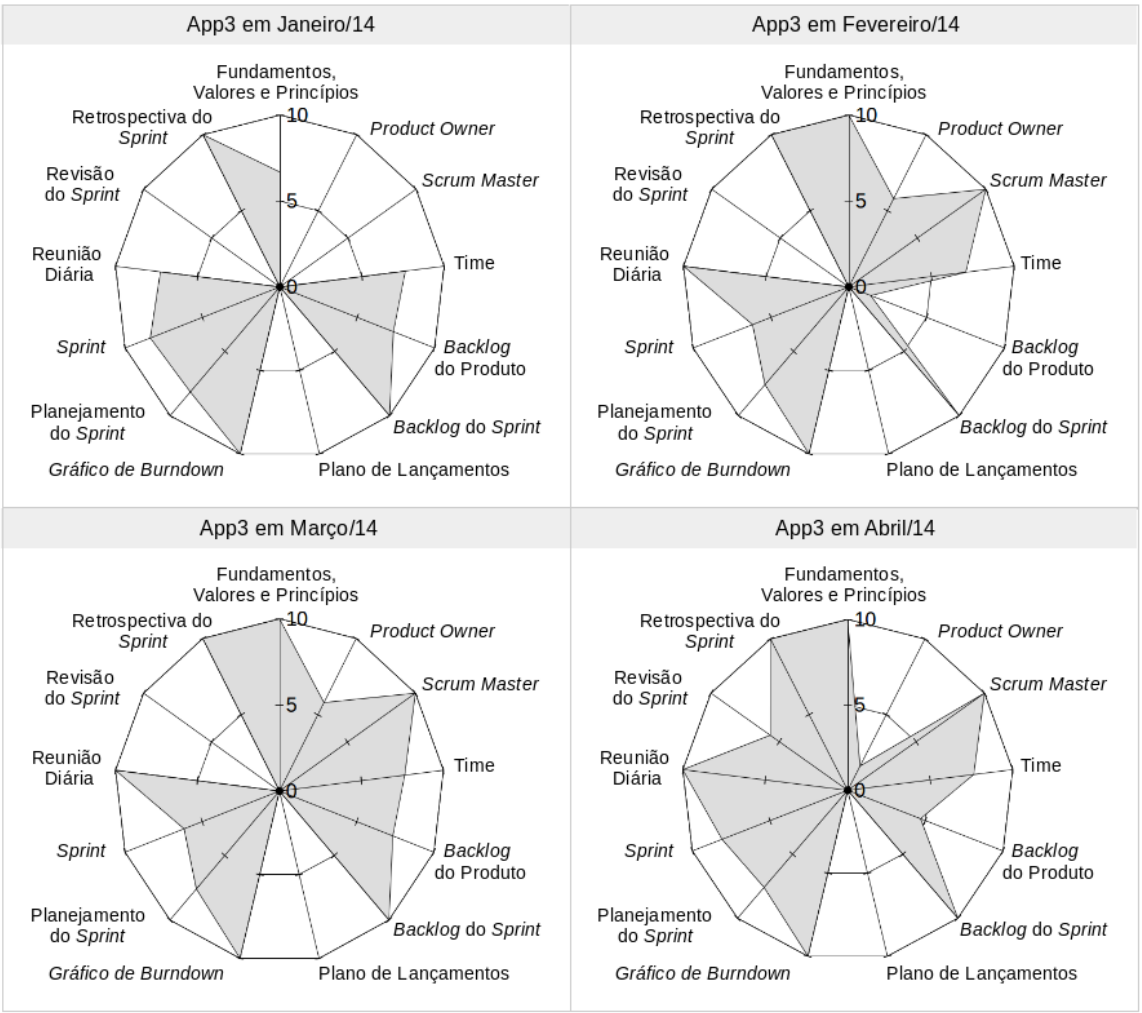


Figura 5.13: Resultado das Avaliações do Projeto App3

Em Janeiro, o nível do indicador do *Scrum* alcançou 6,18, em seguida evoluiu para 6,3 em Fevereiro, depois para 6,92 em Março e, por fim, atingiu o valor de 7,23 em Abril (veja a Figura 5.12). Essas são informações que podem ser apresentadas e discutidas com a equipe durante as Reuniões de Retrospectiva.

Um dos principais interessados no acompanhamento desse indicador é o *Scrum Master*, que é responsável por assegurar que o Time utilize adequadamente os princípios e práticas do *Scrum*. Portanto, essas informações podem estar sempre visíveis para o Time, para que o mesmo conheça o estado atual do uso do processo ágil e entenda as variações dessa utilização.

Esse projeto evoluiu no uso dos Valores, Princípios e Fundamentos, na definição mais clara do *Scrum Master*, na execução de reuniões diárias e na Revisão do *Sprint*. Entretanto, é possível também identificar que houve uma regressão quanto às atividades realizadas pelo *Product Owner*. Uma análise mais detalhada dessas informações possibilita ao Time identificar

tanto as principais vulnerabilidades ao nível atual do uso do *Scrum* quanto as principais oportunidades de melhoria para o mesmo.

5.3.4 Projeto App4

O projeto App4 foi iniciado em abril de 2013 e possui uma equipe com conhecimento satisfatório no *Scrum*, a maior parte de seus integrantes tem até um 1 ano de experiência nessa metodologia e sinalizam um interesse mediano na mesma (veja a Figura 5.14).

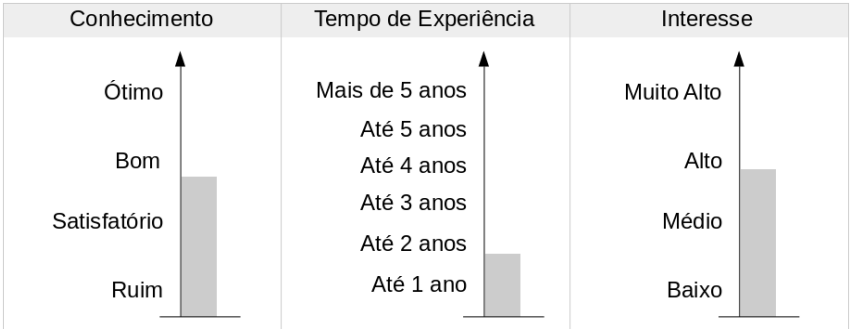


Figura 5.14: Conhecimento, Experiência e Interesse da Equipe App4 no *Scrum*

Como exemplo do uso da abordagem, os resultados da aplicação do *Agile DMAIC* nesse projeto estão resumidos na Tabela 5.5 e na Figura 5.15.

Tabela 5.5: Acompanhamento dos Resultados do *Agile DMAIC* no Projeto App4

	Janeiro	**Fevereiro**	**Março**	**Abril**
Acertos do Agile DMAIC	37	42	38	61
Erros do Agile DMAIC	15	23	4	4
Incertezas sinalizadas no Agile DMAIC	13	0	23	0
Participação da Equipe	100,00%	16,67%	66,67%	50,00%
Exatidão do Agile DMAIC	71,15%	64,72%	90,48%	93,85%
Assertividade do Agile DMAIC	80,00%	100,00%	64,64%	100,00%
Indicador do Nível do *Scrum*	**4,47**	**4,79**	**5,42**	**5,79**

O *Agile DMAIC* é exato na análise da coleta de um item se e somente se a revisão com os líderes ratifica o resultado dessa análise (veja a definição de exatidão na Seção 5.1). Já quando o resultado da análise sinaliza que um item foi atendido ou mesmo que não foi atendido e a revisão com os líderes discorda totalmente desse resultado, então o *Agile DMAIC* erra na previsão da situação desse item.

Por outro lado, quando a equipe está dividida entre o atendimento de um item e isso é refletido na coleta a partir de uma quantidade de votos mediana, uma incerteza pode ser sinalizada pelo *Agile DMAIC* na análise, implicando que os líderes precisam revisar com a equipe esse resultado para chegar a um consenso.

Por exemplo, em Janeiro, o *Agile DMAIC* acertou 37 itens, errou 15 e indicou como incertezas outros 13 (veja a coluna *Janeiro* na Tabela 5.5). Dessa forma, em 37 (de 65) itens do

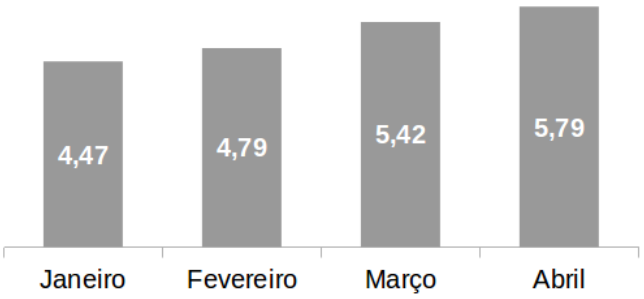


Figura 5.15: Acompanhamento do Indicador do Nível do *Scrum* no Projeto App4

Checklist de Avaliação do *Scrum* (*CAS*) a percepção da equipe foi alinhada à revisão realizada pelo gerente e pelos líderes e refletiu adequadamente a situação do projeto. Já os 15 erros indicam que houve discordância entre análise da percepção da equipe e a revisão pelo gerente e pelos líderes. E, por último, foram apontadas 15 incertezas, as quais demandam um alinhamento com a equipe para a determinação da situação desses itens.

A fórmula a seguir apresenta um exemplo do cálculo da exatidão do *Agile DMAIC* no mês de Março para o projeto App4:

$$Exatidão = \frac{Total\ de\ Acertos}{Total\ de\ Acertos + Total\ de\ Erros} = \frac{38}{38+4} = 90,48\%.$$

Esse valor representa que a percepção da equipe no mês de Março estava alinhada à real situação do uso de princípios e práticas do *Scrum* no projeto.

Já o cálculo da assertividade do método no mês de Abril para o projeto App4 é apresentado a seguir:

$$Assertividade = \frac{Total\ de\ Itens\ do\ CAS - Total\ de\ Incertezas}{Total\ de\ Itens\ do\ CAS} = \frac{65-0}{65} = 100,00\%.$$

Esse valor representa que não houve incertezas sinalizadas pelo método em Abril. Esse nível de assertividade indica que houve um concordância dos participantes nessa avaliação sobre o uso do *Scrum* no projeto, possibilitando identificar de forma assertiva os principais problemas no uso dessa metodologia.

A partir da Figura 5.15 e dos gráficos das avaliações do projeto App4 apresentados na Figura 5.16 é possível identificar uma pequena evolução do indicador do nível do *Scrum* nesse projeto. Além disso, também é possível verificar que algumas práticas do *Scrum* não foram adotadas pelo projeto App4 como, por exemplo, a utilização do Plano de Lançamentos e do *Backlog* do *Sprint*.

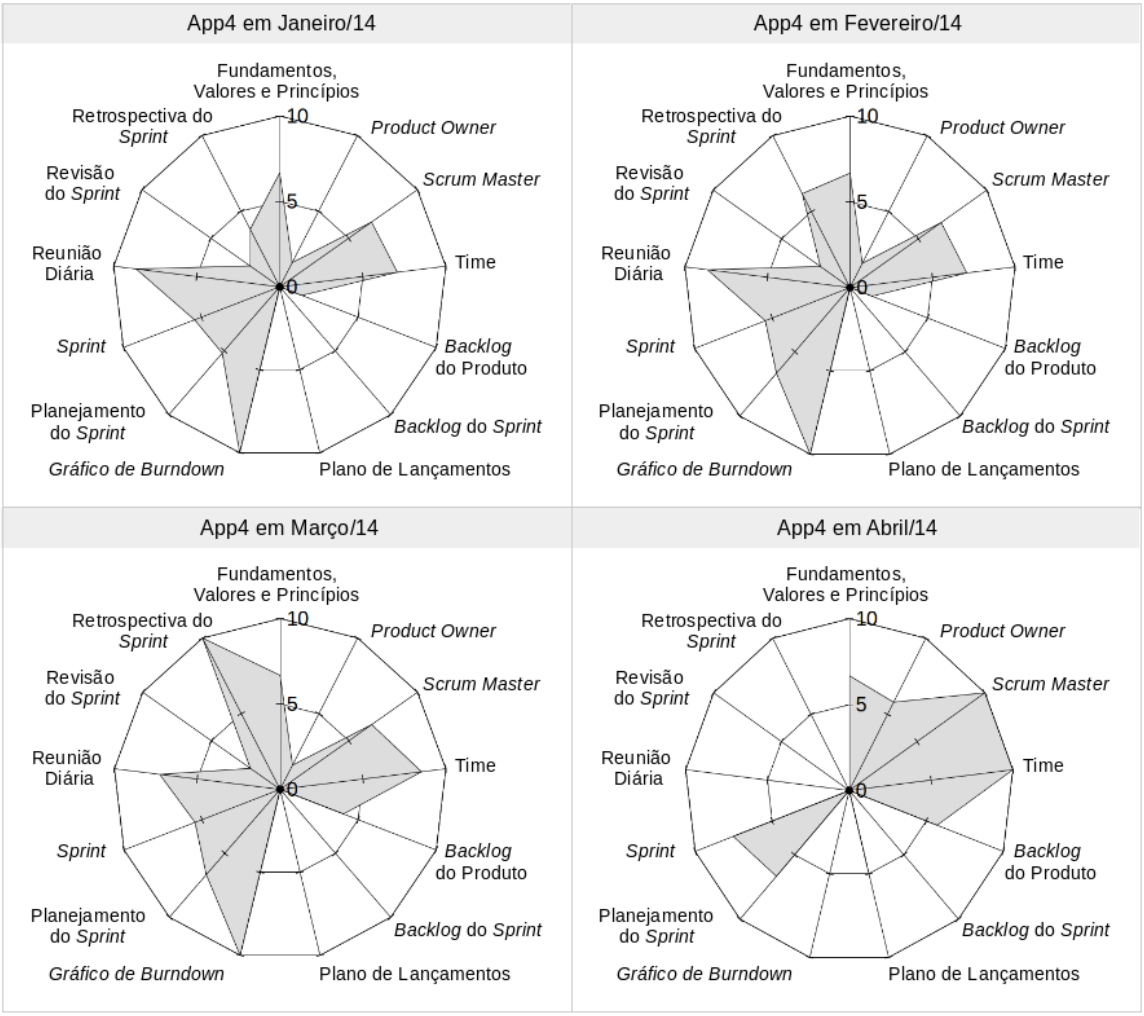


Figura 5.16: Resultado das Avaliações do Projeto App4

5.3.5 Projeto App5

O projeto App5 foi iniciado em abril de 2013, possui uma equipe com bons conhecimentos no *Scrum*, pois há vários integrantes com mais de 1 ano de experiência nessa metodologia, 5 deles inclusive com mais de 3 anos de experiência (veja a Figura 5.17).

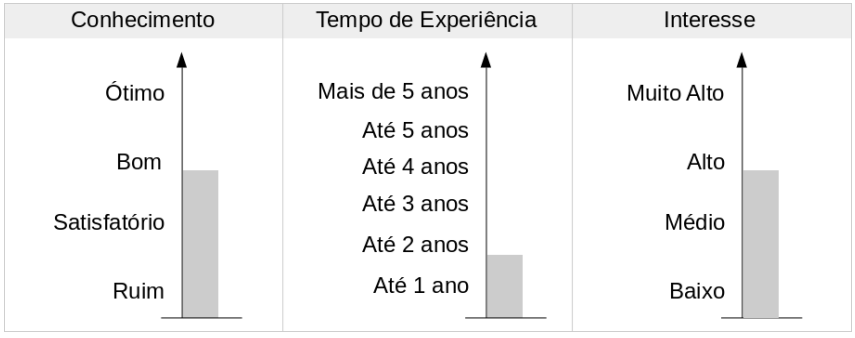


Figura 5.17: Conhecimento, Experiência e Interesse da Equipe App5 no *Scrum*

Embora esse projeto enfrente dificuldades relacionadas às frequentes mudanças de escopo, prazo e alta rotatividade e, além disso, essa equipe sinalize um alto interesse no uso do *Scrum*, esse projeto não utiliza de fato essa metodologia, adotando algumas práticas da mesma como, por exemplo, as Reuniões Diárias e a Retrospectiva, embora não sejam adotadas em todas as iterações.

As avaliações do projeto App5 são apresentadas na Figura 5.18. Essas avaliações refletem o uso de poucas práticas do *Scrum*. Esse projeto foi incluído nesse estudo caso para avaliar se a abordagem diferencia adequadamente entre os projetos que usam o *Scrum* e aqueles que não o utilizam.

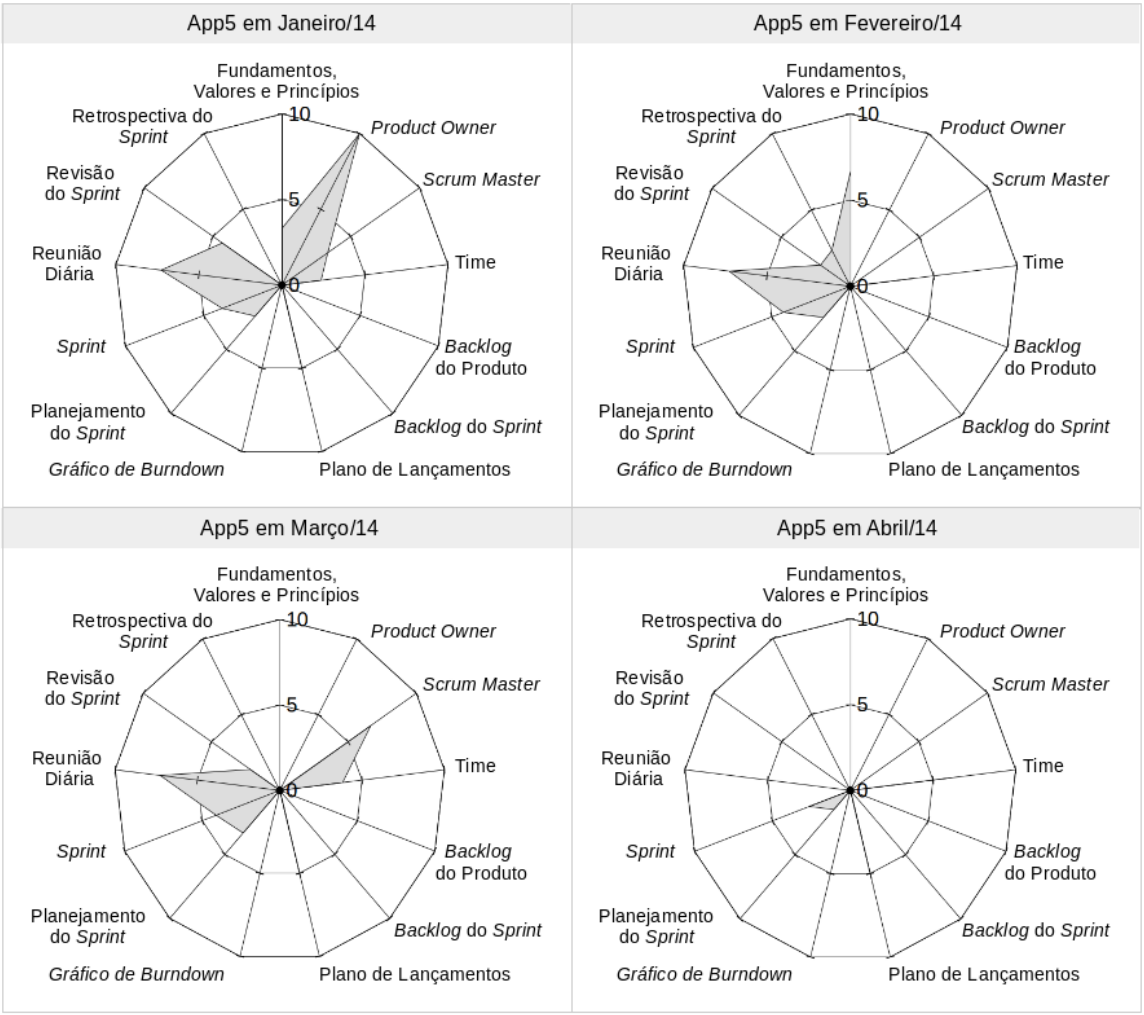


Figura 5.18: Resultado das Avaliações do Projeto App5

5.3.6 Projeto App6

O projeto App6 iniciou em janeiro de 2014 e passou a adotar o *Scrum* e a abordagem proposta nesta dissertação após participar de um dos treinamentos do *Scrum* realizados neste estudo de caso. Um resumo do nível de conhecimento, do tempo de experiência e do interesse da equipe nessa metodologia é apresentado na Figura 5.19.

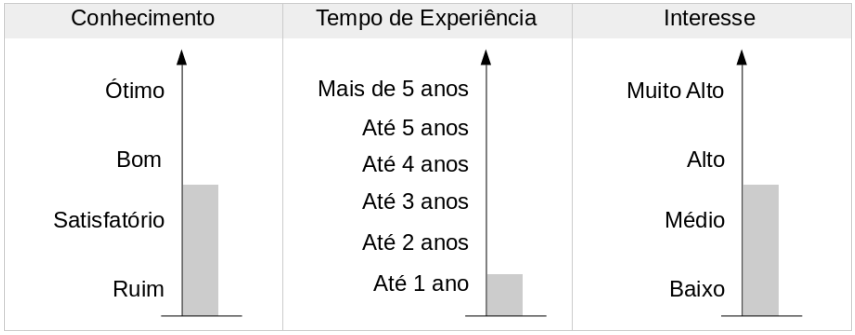


Figura 5.19: Conhecimento, Experiência e Interesse da Equipe App6 no *Scrum*

Além de sinalizar um alto interesse no *Scrum*, essa equipe possui um bom conhecimento nessa metodologia, pois seus integrantes tem em sua maioria um pouco mais de 1 ano de experiência com a mesma.

Como exemplo do uso da abordagem, o indicador do nível do uso do *Scrum* pode ser acompanhado a partir da Tabela 5.6. Com as informações apresentadas nessa tabela é possível verificar a evolução do uso do *Scrum* nesse projeto.

Tabela 5.6: Acompanhamento do Indicador do Nível do *Scrum* no Projeto App6

	Janeiro	**Fevereiro**	**Março**	**Abril**
Valores, Princípios e Fundamentos	10	10	6,67	6,18
Product Owner	0	10	10	10
Scrum Master	7,84	4,33	7,84	7,84
Time	8,64	10	10	9,5
Backlog do Produto	4,27	5	8,09	10
Backlog do *Sprint*	6,67	9,08	10	9,08
Plano de Lançamentos	3,33	0	10	10
Gráfico de *Burndown*	0	0	10	9,32
Planejamento do *Sprint*	2,95	8,86	10	8,86
Sprint	0	3,05	10	7,34
Reunião Diária	8,54	8,54	10	10
Reunião de Revisão	3,33	10	10	10
Retrospectiva	0	8,4	10	0
Indicador do Nível do *Scrum*	**4,73**	**6,83**	**9,03**	**8,38**

A partir das informações da Tabela 5.6 também é possível identificar que uma das principais causas na redução do nível do mês de Março para Abril foi a não realização da Reunião de Retrospectiva. Em Março, o indicador de nível para a Retrospectiva havia alcançado seu valor máximo, 10, entretanto, em Abril esse valor caiu para 0 (veja a penúltima linha dessa tabela).

O acompanhamento do indicador do nível do Scrum é apresentado na Figura 5.20. Em Janeiro, o nível desse indicador alcançou 4,73, em seguida evoluiu para 6,83 em Fevereiro, depois para 9,03 em Março e, por fim, regrediu um pouco alcançando o valor de 8,38 em Abril.

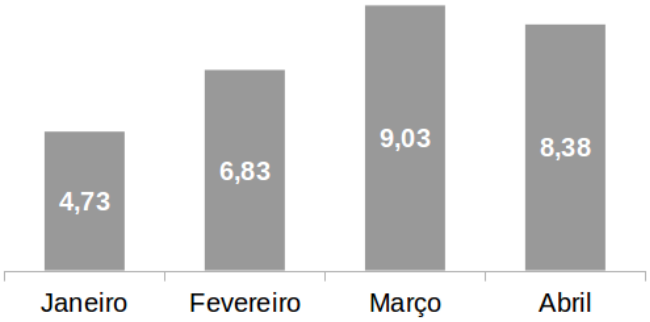


Figura 5.20: Acompanhamento do Indicador do Nível do *Scrum* no Projeto App6

Ainda como exemplo do uso da abordagem, as avaliações do uso de práticas do *Scrum* agrupadas em níveis do Índice de Medição do Nível do *Scrum* (*IMS*) para o projeto App6 são apresentadas na Figura 5.21.

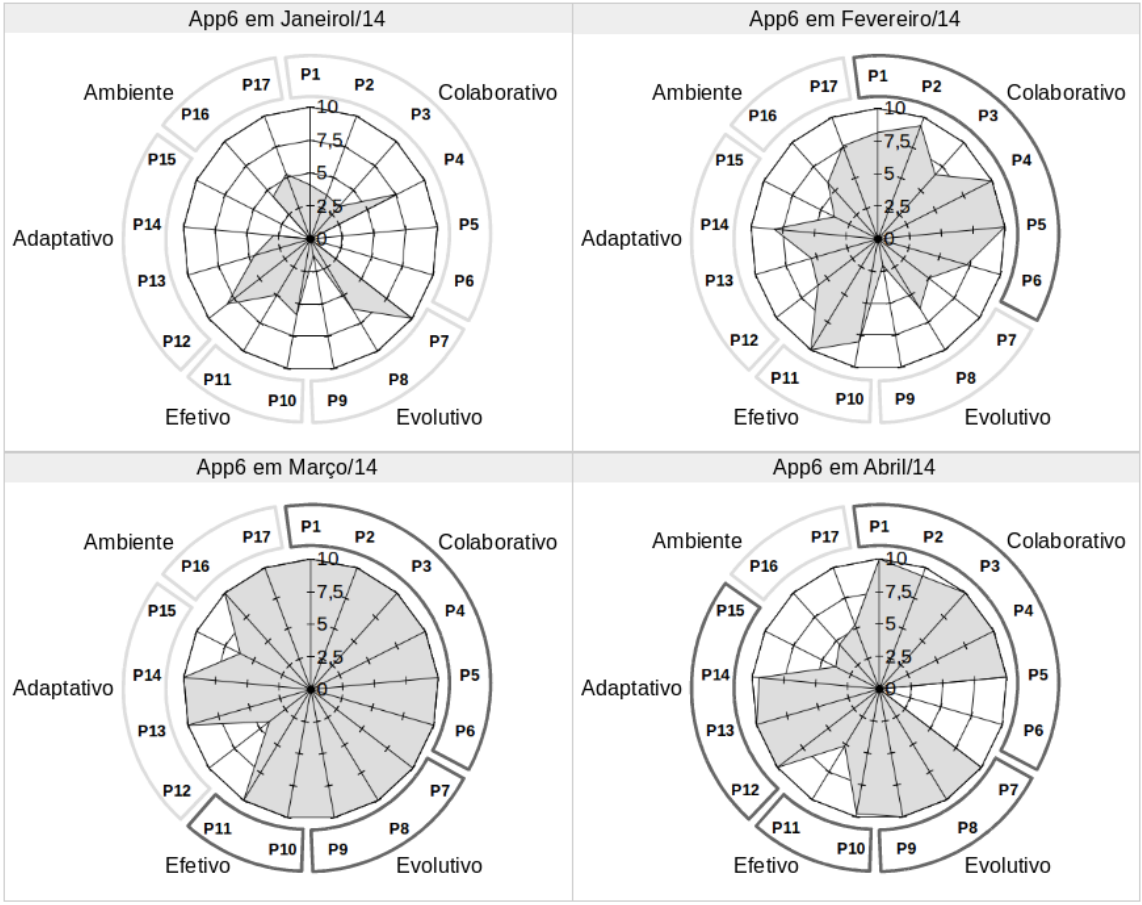


Figura 5.21: Uso do Índice de Medição do Nível do *Scrum* para Acompanhamento das Práticas Ágeis no App6

Em Janeiro, o projeto App6 não atendeu as práticas exigidas para alcançar o primeiro nível, *Colaborativo*, pois nesse período o projeto havia acabado de finalizar o treinamento no *Scrum* e estava ainda em processo de adoção dessa metodologia.

Já em Fevereiro, o projeto App6 continuou melhorando a aplicação das práticas do *Scrum* e alcançou o nível *Colaborativo*. As práticas P1 a P6 (veja a Tabela 4.4), que correspondem ao nível *Colaborativo*, estavam sendo utilizadas no projeto, de acordo com a percepção da equipe, identificando que o projeto promovia a comunicação e a colaboração entre os principais envolvidos.

Em Março, o projeto continuou a evoluir, alcançando o nível *Efetivo*, o qual objetiva o desenvolvimento de *software* testado e com qualidade, e, em seguida, o *Adaptativo* em Abril. Nessa última avaliação, a equipe e os líderes sinalizaram, a partir da coleta e de sua revisão, que o App6 estava utilizando as práticas P1 a P15.

O cálculo do nível do *Scrum* por prática é realizado considerando apenas os itens do *CAS* relacionados à prática específica (veja a Tabela 4.3).

As informações da coleta de Março contendo os itens do *CAS* utilizados na avaliação da Prática P1, relacionada à reflexão e ajuste do processo, são apresentadas na Tabela 5.7 para exemplificação do cálculo do nível para essa prática.

Tabela 5.7: Coleta para o Cálculo do Nível da Prática P1 do App6 em Março de 2014

Itens do *CAS*	**Coleta**	**Análise**	**Revisão**	**Prioridade**
I_03. Processo está continuamente melhorando	4	1	1	55
I_41. Há reunião de revisão a cada iteração	4	1	1	21
I_45. Há retrospectiva ao final da iteração	4	1	1	21
I_46. Resulta em propostas de melhorias concretas	4	1	1	21
I_47. Algumas propostas são implementadas	4	1	1	34

Nessa avaliação de Março, apenas 4 dos 6 integrantes da equipe responderam ao *checklist*, obtendo, portanto, 66,67% de participação na coleta. Por exemplo, a prática P1 abrange os itens I_3, I_41, I_45, I_46 e I_47 do *CAS* e, portanto, o cálculo do nível pode ser obtido a partir da seguintes fórmula:

$$10 * \frac{\sum Prioridades\ dos\ Itens\ Atendidos}{\sum Todas\ as\ Prioridades} = 10 * \frac{55+21+21+21+34}{55+21+21+21+34} = 10$$

Esse cálculo é a soma das prioridades dos itens atendidos dividida pela soma de todas as prioridades e ao final multiplicado por 10. Como todos os itens foram atendidos, a prática P1 obteve nível 10 no App6 para o mês de Março.

As avaliações do projeto *App6* são apresentadas na Figura 5.22. O projeto *App6* obteve a melhor evolução entre os projetos do estudo de caso. Nesse projeto foram identificadas e implantadas ações de melhoria como a definição mais clara dos papéis, *Scrum Master* e *Product Owner*, e a resolução de falhas de comunicação, entendimento e percepção da equipe sobre o uso do processo ágil.

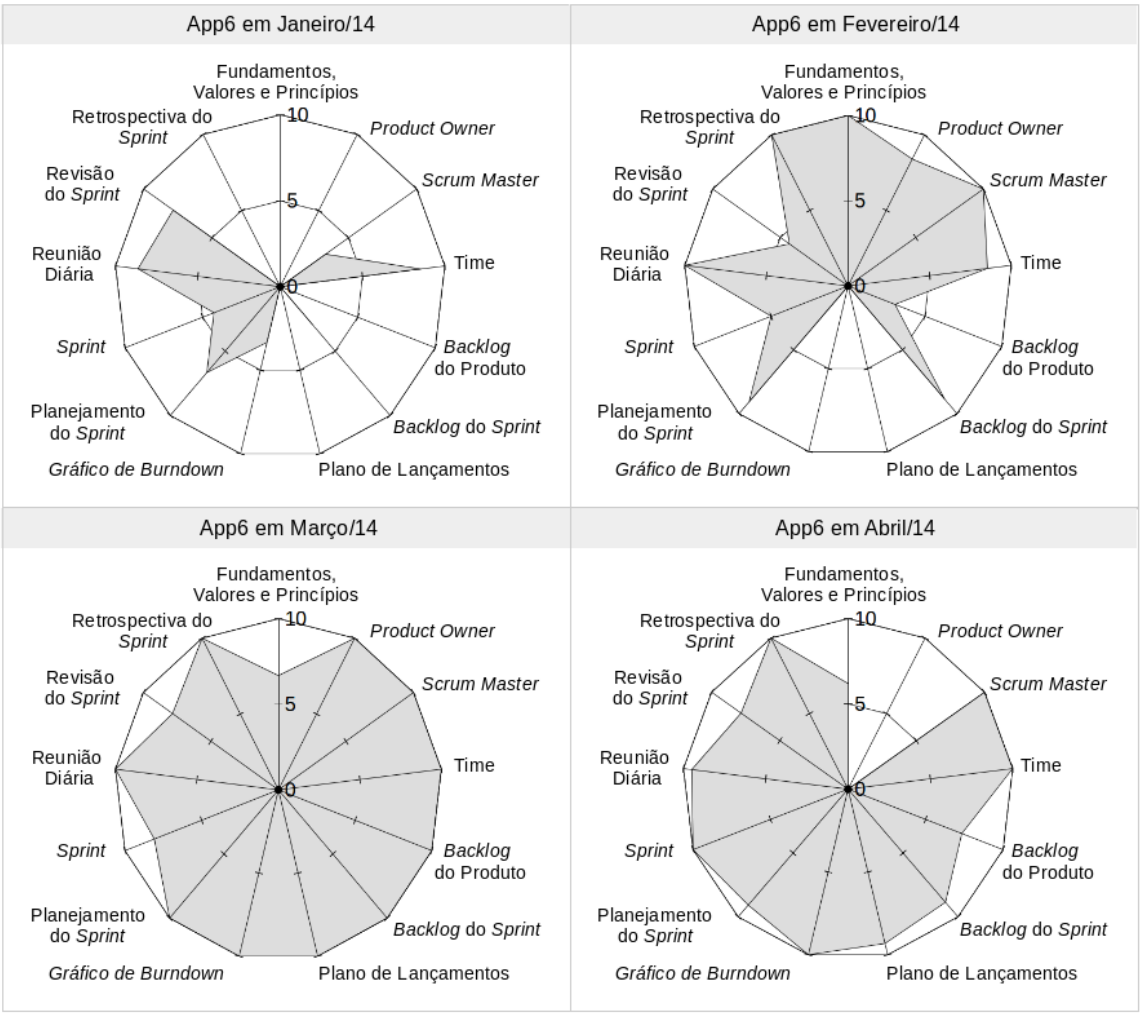


Figura 5.22: Resultado das Avaliações do Projeto App6

5.3.7 Projeto App7

O projeto App7 iniciou em julho de 2013 e passou a adotar o *Scrum* desde o seu início. Um resumo do nível de conhecimento, do tempo de experiência e do interesse da equipe nessa metodologia é apresentado na Figura 5.23.

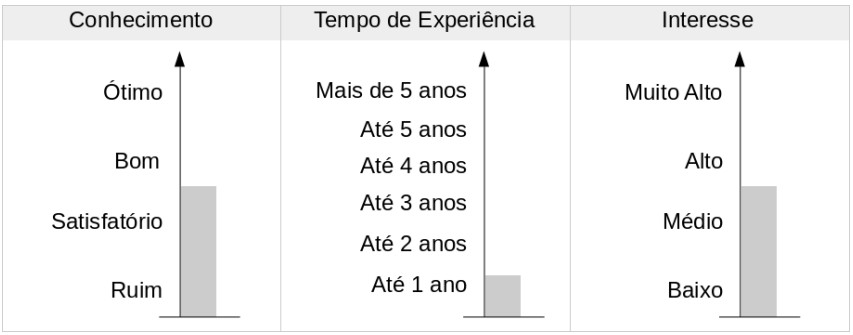


Figura 5.23: Conhecimento, Experiência e Interesse da Equipe App7 no *Scrum*

Essa equipe possui um conhecimento satisfatório no *Scrum*, pois seus integrantes tem em sua maioria até 1 ano de experiência com essa metodologia e sinalizam um interesse mediano na mesma.

Como exemplo da utilização da abordagem, as avaliações do uso de práticas do *Scrum* agrupadas em níveis do *IMS* para o projeto App7 são apresentadas na Figura 5.24.

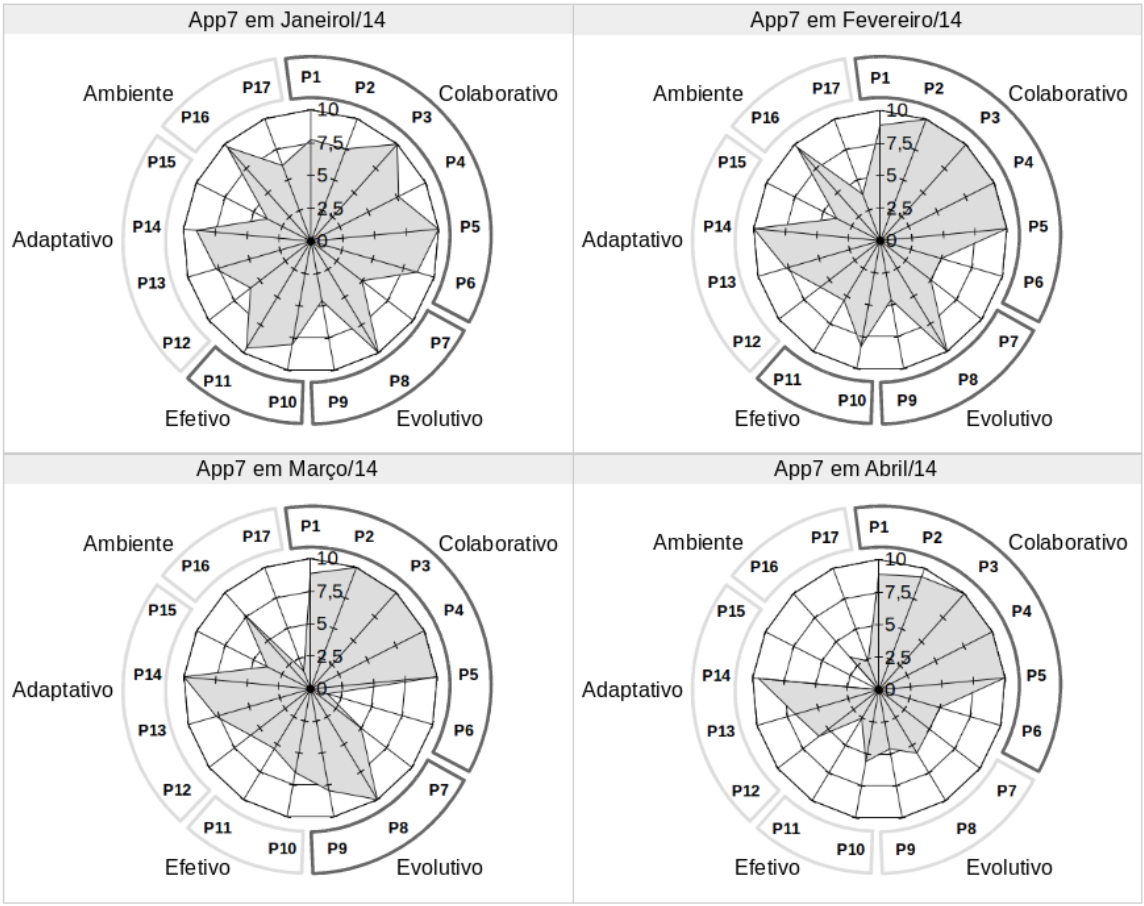


Figura 5.24: Uso do Índice de Medição do Nível do *Scrum* para Acompanhamento das Práticas Ágeis no App7

Esse projeto em Janeiro atendeu as práticas exigidas para alcançar o nível *Efetivo*, embora algumas dessas práticas tenham sido apenas parcialmente atendidas (veja a Tabela 4.4). Em Fevereiro, o projeto continuou nesse nível, entretanto, regrediu para o nível *Evolutivo* em Março e depois para o *Colaborativo* em Abril.

Os gráficos das avaliações do projeto App7 são apresentados na Figura 5.25. De acordo com a análise da equipe, esse projeto também obteve impacto por falta de priorização do *Product Owner* nas atividades do projeto após o mês de Março.

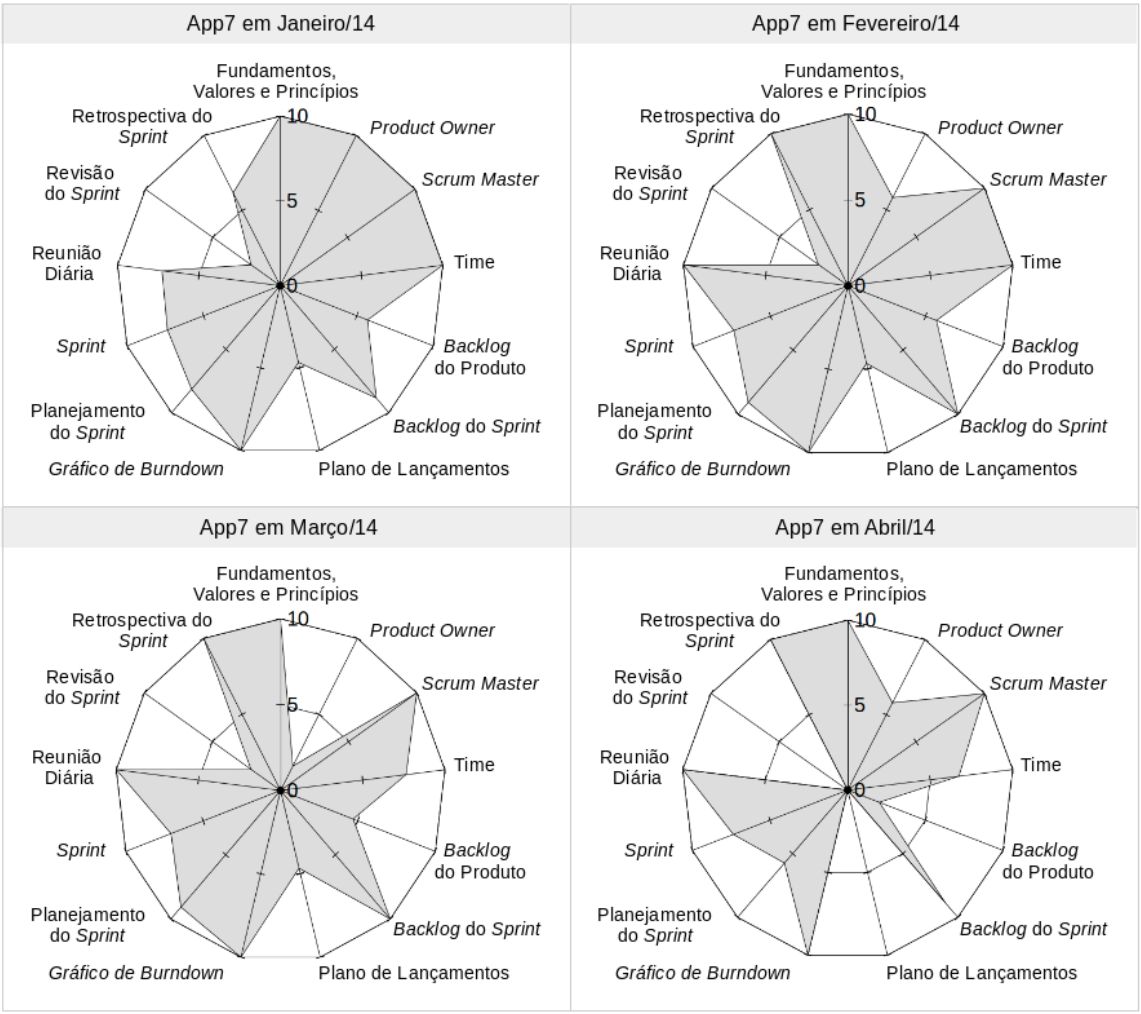


Figura 5.25: Resultado das Avaliações do Projeto App7

5.4 Discussão dos Resultados

Os gráficos apresentados nesta seção foram elaborados com o *Minitab* (INC., 2014), que é um *software* de estatística e gestão de processos.

O gráfico de *Boxplot* dos projetos avaliados nesse estudo de caso é apresentado na Figura 5.26. Esse gráfico é utilizado para a análise da variação do indicador do nível do *Scrum* nesses projetos.

O projeto *App5* obteve a menor variação e o nível mais baixo por não usar o *Scrum*, enquanto o *App6* obteve a maior variação (positiva) a partir de ações de melhoria identificadas com o *Agile DMAIC*, como a necessidade de definição mais clara dos papéis (*Scrum Master* e *Product Owner*) e a resolução de falhas de comunicação e percepção da equipe (entendimento) quanto aos princípios e práticas do *Scrum*, bem como a própria melhoria no uso dessas práticas e de seus artefatos. Quanto aos demais projetos, as análises das equipes também confirmaram que a abordagem refletiu a realidade do uso do *Scrum* nos meses avaliados.

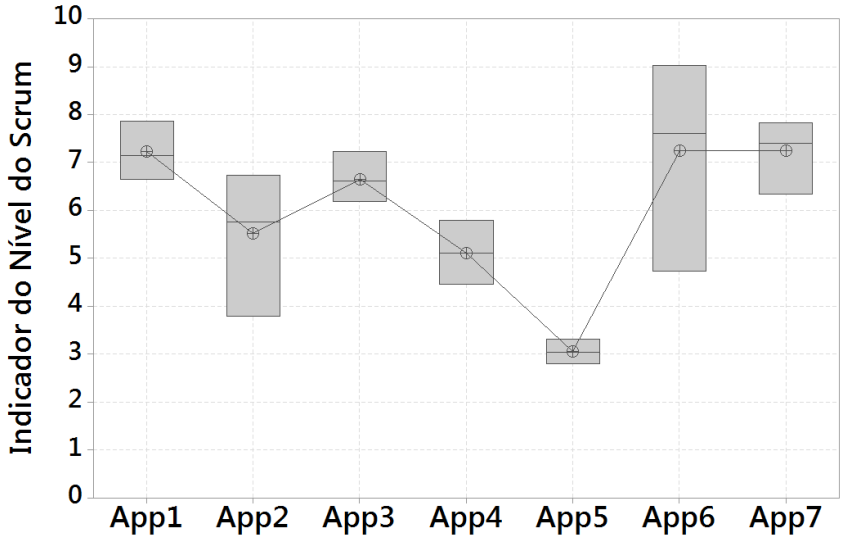


Figura 5.26: Boxplot dos Projetos

A variação desse indicador do nível nem sempre foi positiva em todos os meses. Entretanto, em análises detalhadas, os problemas identificados com o uso da abordagem e que refletiram em uma variação negativa também foram comprovados pelas equipes.

O gráfico de variância entre o resultado do indicador de nível do *Scrum* e as metodologias utilizadas nos projetos é apresentado na Figura 5.27. Como nesse gráfico os intervalos não se sobrepõem, os desvios padrão correspondentes são significativamente diferentes com 95% de confiabilidade ($\alpha = 0,05$) e, dessa forma, a abordagem consegue diferenciar adequadamente entre os projetos que usam o *Scrum* e os que não o utilizam.

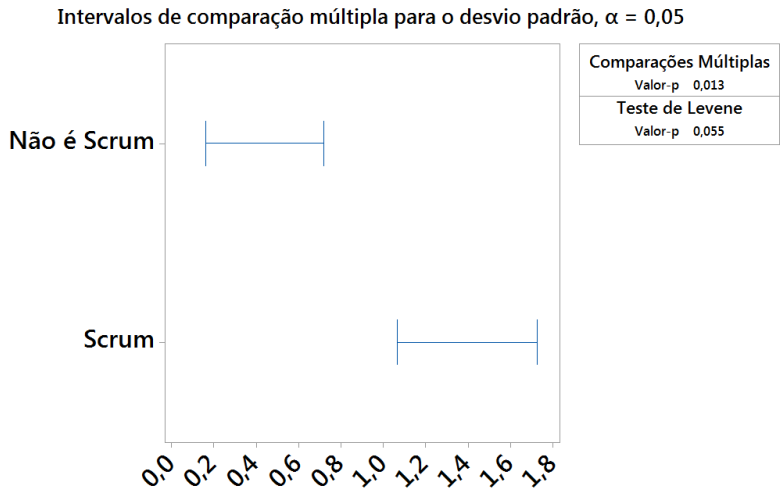


Figura 5.27: Teste de Variância em Função das Metodologias Utilizadas nos Projetos

O gráfico de controle dos projetos que utilizam o *Scrum* é apresentado na Figura 5.28. Nesse gráfico estão apresentados os limites naturais de controle, Limite Inferior de Controle (LIC) e Limite Superior de Controle (LSC), a média ($\bar{X}$) e o Limite de Especificação Inferior (LEI). Os projetos apresentaram uma média de 6,47 para o indicador do nível do *Scrum* e ficaram abaixo da meta estabelecida (alcançar um nível igual ou superior ao LEI, definido

como 8), o que implica em uma baixa capacidade atual do processo ágil. Essa meta foi definida a partir da análise da primeira coleta dos projetos, referente a janeiro de 2014, e da avaliação das principais vulnerabilidades no uso do *Scrum* nesses projetos. Entretanto, mesmo nessa fase inicial de adoção da abordagem, as equipes concordaram que o *Agile DMAIC* foi coerente em suas avaliações e que possibilitou a identificação e resolução dos principais problemas no uso do *Scrum*.

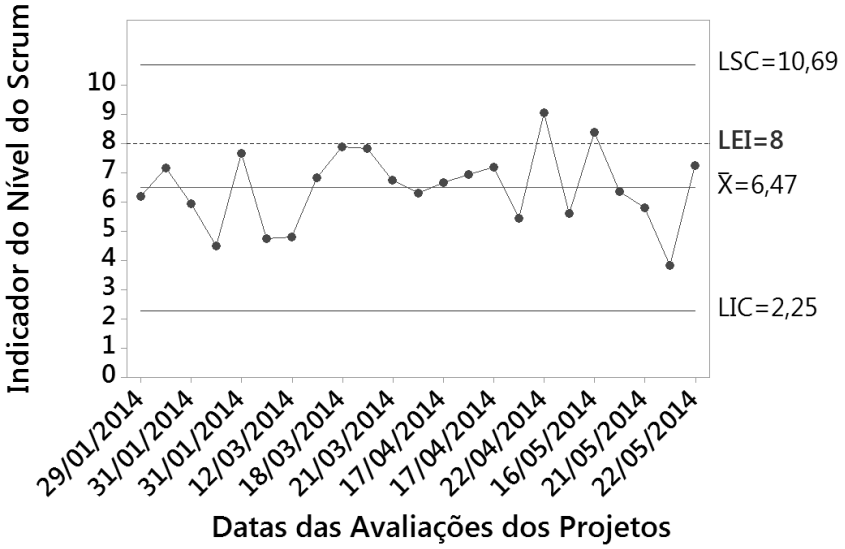


Figura 5.28: Controle dos Projetos que Utilizam o *Scrum*

O gráfico de controle da assertividade do *Agile DMAIC* durante as avaliações dos projetos é apresentado na Figura 5.29. Essas avaliações apresentaram uma média de 84,41% para a assertividade do *Agile DMAIC* indicando que, mesmo que as equipes não possuam um conhecimento avançado no *Scrum* (caso dos projetos aqui apresentados), a percepção dessas equipes pode ser utilizada na identificação e solução dos principais problemas no uso de princípios e práticas dessa metodologia.

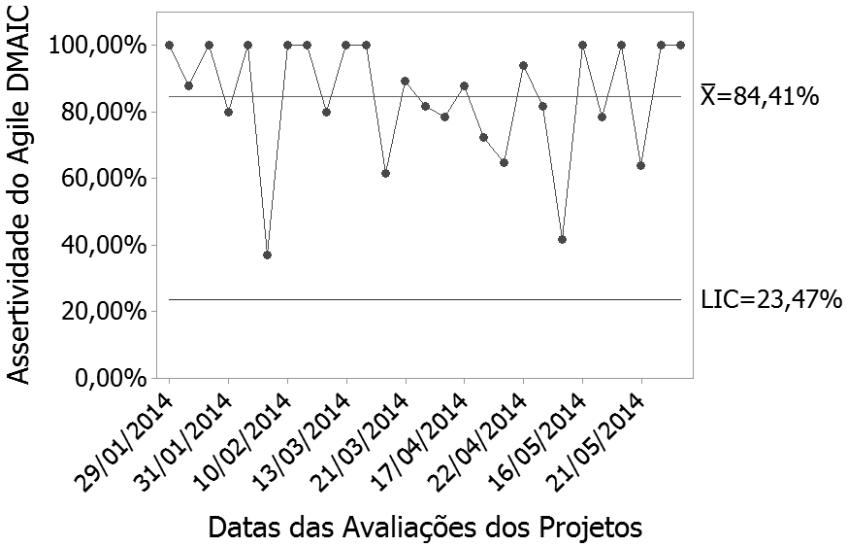


Figura 5.29: Controle da Assertividade do *Agile DMAIC* nas Avaliações dos Projetos

O gráfico de controle da exatidão do *Agile DMAIC* durante as avaliações dos projetos é apresentado na Figura 5.30. Essas avaliações apresentaram uma média de 89,80% para a exatidão do *Agile DMAIC*. Os bons resultado alcançados tanto nos níveis de assertividade e de exatidão do método indicam que a utilização da percepção da equipe na identificação e solução dos principais problemas no uso do *Scrum* é eficaz.

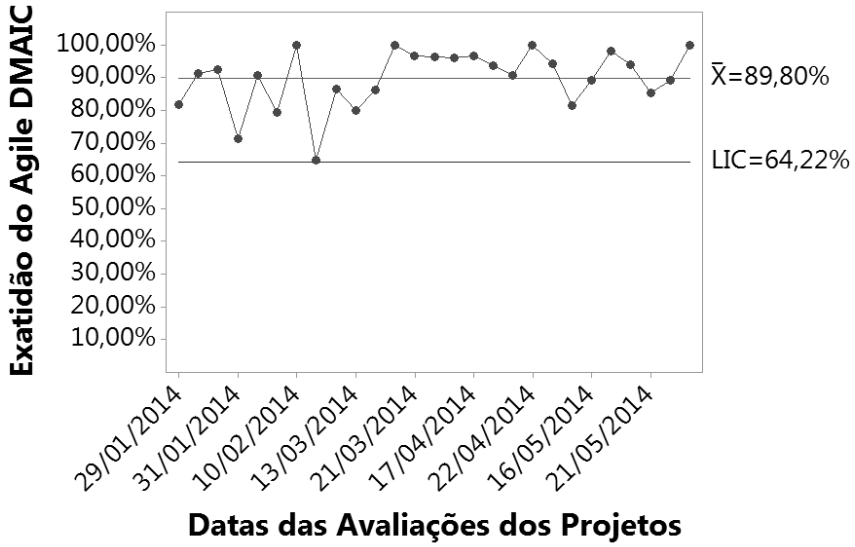


Figura 5.30: Controle da Exatidão do *Agile DMAIC* nas Avaliações dos Projetos

O gráfico de dispersão entre a exatidão do *Agile DMAIC* e a participação dos integrantes do projeto nas avaliações realizadas é apresentado na Figura 5.31. Esse gráfico mostra que a correlação entre a exatidão do método e a participação dos integrantes é muito baixa, o que significa que mesmo com uma baixa participação das equipes, comportamento esperado no que se refere ao preenchimento de *checklists* e questionários por essas equipes, a exatidão do *Agile DMAIC* é alta, indicando que nessas condições o método é eficaz na identificação dos problemas relacionados ao uso do *Scrum*.

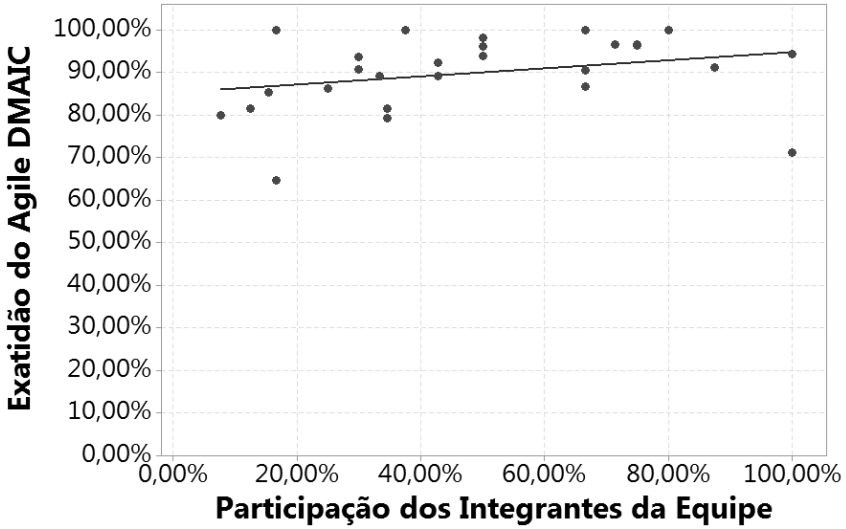


Figura 5.31: Dispersão entre a Exatidão do *Agile DMAIC* e a Participação dos Integrantes

A avaliação das equipes quanto ao tempo de preenchimento do *CAS*, ao nível de cobertura desse *checklist* em relação às práticas do *Scrum* e quanto à sua clareza e objetividade foram coletados a partir de um questionário aplicado via *WEB*. Um resumo das informações dessa coleta é apresentado na Figura 5.32. As equipes informaram que foi necessário em média menos de 15 minutos para o preenchimento do *CAS* e que o mesmo se mostra satisfatório quanto à cobertura das práticas do *Scrum* e à sua clareza e objetividade.

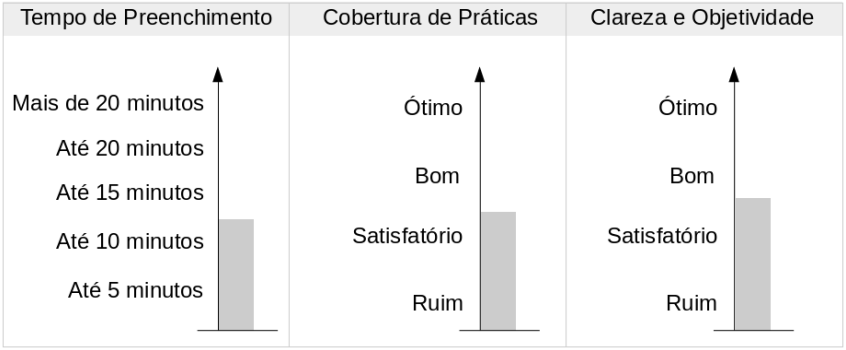


Figura 5.32: Avaliação das Equipes quanto ao Uso do *Checklist* de Avaliação do *Scrum*

Do ponto de vista dos projetos, o *Agile DMAIC* contribui para a melhoria do uso dos princípios e práticas do *Scrum*. A partir dessa melhoria, há a expectativa que as ações já implantadas influenciem positivamente na produtividade da equipe e qualidade dos produtos (LAYMAN et al., 2004)(DYBÅ; DINGSØYR, 2008). Entretanto, ainda há a necessidade de mais avaliações com o *Agile DMAIC* nesses projetos para apresentar dados que correlacionem a evolução do nível do *Scrum* com a melhoria da produtividade, da qualidade e da satisfação do cliente.

5.5 Considerações Finais

Este capítulo apresentou o resultado da utilização do *SLeSS* 2.0 com foco na avaliação e melhoria de princípios e práticas do *Scrum* a partir da aplicação do *Agile DMAIC* em 7 projetos reais, de complexidades média a alta, voltados à P&D de aplicações e customizações de *software* para DMs.

A análise dos resultados apresentados, realizada em conjunto com as equipes, possibilitou identificar que, mesmo com certo tempo e experiência no *Scrum*, essas equipes necessitavam avaliar de que forma estavam utilizando os princípios e práticas dessa metodologia. Essa avaliação periódica dos pontos fortes e fracos da implementação do *Scrum*, a partir da percepção da equipe, possibilitou identificar e tratar os principais problemas no uso dessa metodologia.

Os resultados apresentados mostraram que, mesmo com uma baixa participação da equipe em algumas avaliações realizadas, comportamento esperado das equipes de desenvolvimento, o *Agile DMAIC* conseguiu apresentar uma assertividade média acima de 84% e uma exatidão média acima de 89%, indicando que esse método possui um bom nível de assertividade e exatidão na identificação e solução de problemas no uso do *Scrum*.

A partir dos exemplos de utilização do *SLeSS* 2.0 apresentados neste capítulo, foi possível demonstrar que a abordagem fornece meios adequados para a identificação dos problemas no uso do *Scrum* e, a partir do reuso das técnicas e ferramentas do *LSS*, esse método também possibilita a análise de causas desses problemas e a identificação de ações para a solução dos mesmos. Além disso, a partir dessas técnicas e ferramentas, o método também possibilita a priorização das melhorias identificadas, contribuindo na evolução da agilidade dos projetos. Dessa forma, a utilização do *Agile DMAIC* se mostrou uma alternativa simples, de fácil aplicação e capaz de produzir bons resultados mesmo no curto prazo.

O capítulo seguinte descreve as conclusões e contribuições, apresentando as possíveis linhas de pesquisa a serem consideradas em trabalhos futuros.

6 CONCLUSÃO

Este trabalho apresentou o *SLeSS* 2.0, uma nova versão de uma abordagem de integração do *Scrum* e *Lean Six Sigma*, voltada à gestão de projetos e melhoria de processos de desenvolvimento e customização de *software* para DMs. Essa nova versão foi proposta com foco na avaliação e melhoria do uso de princípios e práticas do *Scrum* a partir da utilização de técnicas do *LSS* e no aprimoramento do *SLeSS* para o desenvolvimento, além da customização, de *software* para DMs.

A organização das seções deste capítulo é apresentada na Figura 6.1. Na Seção 6.1 são apresentadas as contribuições da dissertação e os resultados alcançados durante a realização dessa pesquisa. Na Seção 6.2 são descritos os trabalhos futuros para a continuidade da evolução da pesquisa e da abordagem apresentada nesta dissertação.

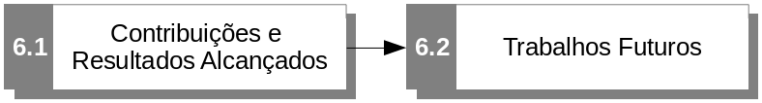


Figura 6.1: Organização das Seções do Capítulo 6

6.1 Contribuições e Resultados Alcançados

No cenário do desenvolvimento e customização de *software* para DMs, o *SLeSS* 2.0, abordagem proposta nesta dissertação, atende ao objetivo de evoluir a versão inicial do *Scrum Lean Six Sigma* (*SLeSS*) (CUNHA et al., 2011) com foco na avaliação e melhoria do uso de princípios e práticas do *Scrum* a partir de técnicas do *LSS*. Além disso, as melhorias propostas possibilitam que essa nova versão seja reutilizada de forma sistemática.

Como uma das evoluções no *SLeSS*, este trabalho propôs o *Agile DMAIC*, um método para avaliação e melhoria do uso do *Scrum* em projetos de *software*. Esse método está contido em um dos mecanismos de integração do *SLeSS 2.0* e pode ser utilizado tanto na adoção do *Scrum* quanto na melhoria de sua implementação, mesmo em projetos que já utilizam essa metodologia há várias iterações.

O *Agile DMAIC* possibilitou avaliar de forma eficaz o uso de princípios e práticas do *Scrum* nos projetos, a partir do *Checklist* de Avaliação do *Scrum* (*CAS*) proposto nesse método. Sendo assim, ao reutilizar as técnicas e ferramentas do *LSS* e fornecer uma simplificação do *DMAIC*, ele também estabeleceu como esses projetos podem identificar os problemas na implementação dessa metodologia e como analisar, priorizar e solucionar as suas causas. Além disso, devido à utilização dessa simplificação do *DMAIC*, que não exige experiência avançada no *LSS*, esse método conseguiu introduzir os principais conceitos do *SLeSS* 2.0, que são úteis tanto na melhoria do processo ágil quanto na melhoria de processos de desenvolvimento, sem a necessidade de treinamentos avançados no *LSS*.

Dessa forma, o *Agile DMAIC* pode ser utilizado para a gestão quantitativa da implementação do *Scrum*, possibilitando a avaliação do uso de princípios e práticas dessa metodologia e a identificação e o tratamento de causas de problemas para a evolução contínua do processo ágil, considerando o cenário e a necessidade dos projetos e dos clientes. Esse método foi aplicado projetos reais e seus resultados, analisados em conjunto com as equipes envolvidas, possibilitaram identificar que, mesmo com tempo e experiência no *Scrum*, essas equipes necessitavam avaliar de que forma estavam utilizando os princípios e práticas dessa metodologia. Essa avaliação periódica da implementação do *Scrum*, a partir da percepção dessas equipes, possibilitou identificar e tratar os problemas dos projetos quanto ao uso do *Scrum*.

Além do *Agile DMAIC*, as alterações propostas nesta dissertação para a evolução da abordagem foram: (i) *a reestruturação dos mecanismos de integração* - os mecanismos foram reestruturados para melhorar o alinhamento dos mesmos às estratégias de integração do *Scrum* e do *LSS* propostas pela abordagem, como a execução iterativa e incremental do *DMAIC*, a combinação de técnicas e práticas do *Scrum* e do *LSS*, a capacitação da equipe na identificação de causas de problemas e a necessidade de avaliação e melhoria do uso de princípios e práticas do *Scrum*; (ii) *o redesenho da visão geral da abordagem* - a nova visão geral procura enfatizar os mecanismos de integração e a proposta de execução iterativa e incremental do *DMAIC*; (iii) *o detalhamento do processo de execução* - nesse processo, são apresentadas as fases, as atividades, os papéis envolvidos, as principais entradas e as saídas da abordagem; e (iv) *os ajustes no fluxo de implantação* - a revisão e alteração do fluxo de implantação com foco na adoção da abordagem por projeto.

Dessa forma, a nova versão da abordagem inclui, portanto, melhorias realizadas em sua definição, nos papéis, nos mecanismos de integração, na apresentação de seu processo de execução e no fluxo de sua implantação. Essas melhorias e os exemplos de utilização dessa nova versão apresentados nesta dissertação são informações que possibilitam o reuso sistemático do *SLeSS* 2.0.

Para avaliar qualitativa e quantitativamente o *SLeSS* 2.0, foi realizado um estudo de caso em 7 projetos reais, de complexidades média e alta, voltados à pesquisa, desenvolvimento e customização de *software* para DMs. Esse estudo envolveu a participação de aproximadamente 70 profissionais de áreas da computação e engenharia, que participaram dos treinamentos técnicos realizados, da aplicação e do aprimoramento da abordagem proposta nesta dissertação.

Os resultados desse estudo incluem desde exemplos da utilização da abordagem, apresentando e discutindo as informações das coletas, das análises estatísticas e das ações de melhorias identificadas a resultados de melhoria da agilidade nos projetos.

A discussão dos resultados desse estudo demonstrou que a abordagem conseguiu diferenciar adequadamente entre os projetos que usam o *Scrum* e aqueles que não o utilizam e que na identificação e solução de problemas no uso do *Scrum*, através do *Agile DMAIC*, a abordagem conseguiu apresentar uma assertividade média acima de 84% e uma exatidão média acima de 89%, indicando que a abordagem possui um bom nível de assertividade e exatidão para a avaliação e melhoria do uso do *Scrum* nos projetos.

Vale ressaltar que durante o desenvolvimento desta pesquisa, que iniciou antes do mestrado, foram publicados artigos relacionados à integração do *Scrum* e do *LSS* no desenvolvimento de *software* para DMs, bem como relacionados à gestão de projetos. A seguir, são apresentadas as publicações já realizadas e os trabalhos em desenvolvimento:

- ✓ *SBES 2011 - SLeSS: a Scrum and LSS Integration Approach for the Development of Software Customization for Mobile Phones*: (a) *Status* - Publicado no XXV Simpósio Brasileiro de Engenharia de *Software* (SBES 2011); e (b) *Autores* - Thiago F. V. da Cunha, Valéria L. L. Dantas e Rossana M. C. Andrade;
- ✓ *SBQS 2014 - Agile DMAIC*: Um Método para Avaliar e Melhorar o Uso do *Scrum* em Projetos de *Software* (CUNHA; ANDRADE, 2014): (a) *Status* - Publicado no XIII Simpósio Brasileiro de Qualidade de *Software* (SBQS 2014); e (b) *Autores* - Thiago F. V. da Cunha e Rossana M. C. Andrade;
- ✓ *MiniPLoP 2013 - Executive Proposal*: Um Padrão para a Apresentação de Propostas de Projetos de *Software*: (a) *Status* - Aceito na Miniconferência Latino-Americana de Linguagens de Padrões para Programação (MiniPLoP Brasil 2013); e (b) *Autores* - Thiago F. V. da Cunha, Corneli G. F. Júnior, Rossana M. C. Andrade, Stenio D. de Lima e Francisco S. de Sousa; e
- ✓ *Journal - Integrating Scrum and Lean Six Sigma: Lessons Learned from Managing Projects for Mobile Applications Development*: (a) *Status* - Em finalização; e (b) *Autores* - Thiago F. V. da Cunha, Valéria L. L. Dantas e Rossana M. C. Andrade.

O *SLeSS* 2.0 é, dessa forma, um resultado da realização de uma pesquisa aplicada ao desenvolvimento de projetos reais, com enfoque na necessidade desses projetos em melhorarem a organização, a produtividade de suas equipes e a qualidade dos produtos desenvolvidos. Essa abordagem procurou ressaltar tanto a importância das pessoas e suas interações, a partir do uso de princípios e práticas ágeis, quanto a importância da melhoria contínua dos processos de desenvolvimento, através do uso de técnicas de gestão da qualidade, para a evolução da agilidade da gestão desses projetos, favorecendo seus resultados frente aos desafios desse tipo de desenvolvimento.

6.2 Trabalhos Futuros

O *SLeSS* 2.0 possibilita o alinhamento do projeto às necessidades dos clientes a partir da priorização tanto do escopo do desenvolvimento quanto das melhorias de processos. Essa priorização atribui valor ao desenvolvimento do time e auxilia na melhoria da qualidade dos produtos. Além disso, uma vez que a abordagem envolve a equipe de desenvolvimento nas melhorias de processos, ela contribui para que essa equipe esteja mais atenta aos processos e consciente da importância dos mesmos na qualidade final do produto.

Entretanto, ainda há uma baixa adesão inicial por parte dos desenvolvedores, principalmente devido ao baixo conhecimento de metodologias como o *Scrum* e o *LSS*. O apoio

contínuo dos patrocinadores, do cliente e da organização é, portanto, essencial para que as equipes se mantenham motivadas no uso dessa abordagem.

Dessa forma, como trabalhos futuros, é necessário analisar os custos-benefícios da utilização do *SLeSS* 2.0, a partir da realização de mais experimentos, avaliando melhores estratégias de adoção e os custos-benefícios de sua utilização a médio e longo prazos.

Um possível linha de pesquisa para a diminuição de custos da utilização do *SLeSS* 2.0 pode envolver uma análise da integração do *MiniDMAIC* proposto por Bezerra *et al.* (BEZERRA et al., 2007) à abordagem, uma vez que o *MiniDMAIC* apresentou resultados de melhoria de produtividade e redução de defeitos, bem como possibilitou a redução do tempo do ciclo *DMAIC* e de custos envolvidos em projetos de melhoria.

Além disso, o *Agile DMAIC* pode ser aprimorado a partir da simplificação de seus componentes. Para a melhoria desse método, foram identificados os seguintes trabalhos futuros: (a) revisar os itens do *Checklist* de Avaliação do *Scrum* (*CAS*) quanto à abrangência dos princípios e práticas do *Scrum* e ao mapeamento desses itens às práticas do Índice de Medição do Nível do *Scrum* (*IMS*); (b) revisar a priorização dos itens do *CAS* a partir de trabalhos como o de Mello *et al.* (MELLO et al., 2014) que avaliam os benefícios da utilização de práticas ágeis; (c) revisar os mapeamentos dos níveis em práticas ágeis e de práticas em itens do *CAS*; (d) melhorar a documentação do *DMAIC*; (e) propor uma automação para o *Agile DMAIC* a partir da implementação do *CAS*, dos mapeamentos propostos no método e do próprio *DMAIC* com o uso de ferramentas de *workflow*; (f) avaliar a relação entre o uso do *Agile DMAIC* e melhorias na produtividade, na qualidade dos produtos e na satisfação do cliente; e (g) aplicar e aprimorar esse método em outros projetos, com diferentes perfis em relação aos já avaliados, bem como em outras organizações.

Por fim, o *SLeSS* 2.0 pode ser ajustado ao desenvolvimento de *software* em geral. Visto que essa abordagem obteve bons resultados em um cenário de desenvolvimento com muitas particularidades e restrições, a sua utilização pode ser analisada em outras áreas de desenvolvimento de *software*, com diferentes características em relação ao desenvolvimento e customização de *software* para DMs.

REFERÊNCIAS BIBLIOGRÁFICAS

ABBAS, S.; MEHTA, J. Lean in software development. In: *Proceedings of the International Conference & Workshop on Emerging Trends in Technology*. New York, NY, USA: ACM, 2011. (ICWET '11), p. 1379–1380. ISBN 978-1-4503-0449-8. Disponível em: <http://doi.acm.org/10.1145/1980022.1980417>.

ABRAHAMSSON, P.; HANHINEVA, A.; HULKKO, H.; IHME, T.; JÃALINOJA, j.; KORKALA, M.; KOSKELA, J.; KYLLONEN, P.; SALO, O. Mobile-d: An agile approach for mobile application development. *Proc of the OOPSLA'04 Conference*, 2004.

ABRAHAMSSON, P.; SALO, O.; RONAKAIMEN, J.; WARSTA, J. Agile software development methods. *Vtt Publications*, Citeseer, v. 478, n. 3, p. 167–168, 2002. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.161.5931&rep=rep1&type=pdf>.

ALMEIDA, E. S.; ALVARO, A.; GARCIA, V. C.; JORGE; BURÉGIO, V. A.; NASCIMENTO, L. M.; LUCRÉDIO, D.; SILVIO. *C.R.U.I.S.E: Component Reuse in Software Engineering*. 1st. ed. Recife: C.E.S.A.R e-book, 2007.

AULAKH, S.; GILL, J. Lean manufacturing- a practitioner #x2019;s perspective. In: *Industrial Engineering and Engineering Management, 2008. IEEM 2008. IEEE International Conference on*. [S.l.: s.n.], 2008. p. 1184 –1188.

BALKANSKI, P. *Team Dysfunctions and Scrum*. 2008. Disponível em: `http://www.scrumalliance.org/community/articles/2008/october/team-dysfunctions-and-scrum`. Acesso em: 01 de Setembro de 2014.

BECK. *Test Driven Development: By Example*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002. ISBN 0321146530.

BECK, K. *Manifesto for Agile Development*. 2001. Disponível em: `http://agilemanifesto.org`. Acesso em: 01 de Setembro de 2014.

BECK, K.; ANDRES, C. *Extreme Programming Explained: Embrace Change (2Nd Edition)*. [S.l.]: Addison-Wesley Professional, 2004. ISBN 0321278658.

BEEDLE, M.; DEVOS, M.; SHARON, Y.; SCHWABER, K.; SUTHERLAND, J. *SCRUM: An extension pattern language for hyperproductive software development*. 2000.

BEZERRA, C. I. M.; COELHO, C. C.; GONÇALVES, F. M. G. S.; PIRES, C. G. S.; TELLES GABRIELA, A. A. B. Minidmaic: Uma abordagem para análise e resolução de causas em projetos de desenvolvimento de software. In: . VIII Simpósio Brasileiro de Qualidade de Software, 2007. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/sbqs/2009/015.pdf>.

BRINKKEMPER, S. Method engineering: engineering of information systems development methods and tools. *Information and Software Technology*, v. 38, n. 4, p. 275–280, 1996.

CARVALHO, S. R. C.; MAIA, P. H.; ANDRADE, R. M. C.; GOMES, D. G. *Uma Proposta para Gerenciamento de Energia em Redes de Sensores utilizando a metodologia Seis Sigma (WGRS)*. 2013.

COCKBURN, A. Crystal clear: A human-powered methodology for small teams. *Addison-Wesley Professional*, 2004.

COHN, M. *Succeeding with Agile: Software Development Using Scrum*. [S.l.]: Addison-Wesley Professional, 2009. ISBN 978-0321579362.

COOK, L. *A Guide to Good Survey Design*. [S.l.]: Statistics New Zealand, 1995. 41–61 p.

CORRAL, L.; SILLITTI, A.; SUCCI, G. Software development processes for mobile systems: Is agile really taking over the business? In: *Engineering of Mobile-Enabled Systems (MOBS), 2013 1st International Workshop on the*. [S.l.: s.n.], 2013. p. 19–24.

CRESCÊNCIO, S. *Disfunções do Scrum*. 2013. Disponível em: `http://oncast.com.br/blog/?p=1515`. Acesso em: 01 de Setembro de 2014.

CUNHA, T. F. V.; DANTAS, V. L. L.; ANDRADE, R. M. C. Sless: A scrum and lean six sigma integration approach for the development of software customization for mobile phones. In: *Software Engineering (SBES), 2011 25th Brazilian Symposium on*. [S.l.: s.n.], 2011. p. 283 –292.

CUNHA, T. F. V. d.; ANDRADE, R. M. C. Agile dmaic: Um método para avaliar e melhorar o uso do scrum em projetos de software. In: *SBQS 2014 - Trabalhos Tecnicos*. [S.l.: s.n.], 2014.

DIEM, K. *A Step-By-Step Guide to Developing Effective Questionnaires and Survey Procedures for Program Evaluation & Research*. Rutgers NJAES Cooperative Extension, 2002. Disponível em: <http://novascotia.ca/psc/pdf/employeeCentre/recognition/toolkit/step7/Guide_to_-Developing_Questionnaires.pdf>.

DYBÅ, T.; DINGSØYR, T. Empirical studies of agile software development: A systematic review. *Inf. Softw. Technol.*, Butterworth-Heinemann, Newton, MA, USA, v. 50, p. 833–859, August 2008. ISSN 0950-5849. Disponível em: <http://portal.acm.org/citation.cfm?id=1379905.1379989>.

EBERT, C.; ABRAHAMSSON, P.; OZA, N. Lean software development. *IEEE Software*, IEEE Computer Society, Los Alamitos, CA, USA, v. 29, n. 5, p. 22–25, 2012. ISSN 0740-7459.

EL-HAIK, B.; SUH, N. P. *Axiomatic Quality*. [S.l.]: ohn Wiley and Sons, 2005.

EXPEDITH, M. V. L. *The Scrum Framework in a SIPOC Nutshell*. 2011. Disponível em: http://www.scrumalliance.org/resource_download/2378. Acesso em: 01 de Setembro de 2014.

FLING, B. *Mobile Design and Development: Practical concepts and techniques for creating mobile sites and web apps*. O'Reilly Media, 2009. (Animal Guide). ISBN 9781449379247. Disponível em: <http://books.google.com.br/books?id=LyMeulBTkH0C>.

FLORA, H. K.; CHANDE, S. V.; WANG, X. Article: Adopting an agile approach for the development of mobile applications. *International Journal of Computer Applications*, v. 94, n. 17, p. 43–50, May 2014. Full text available.

FONTELES, A. S.; NETO, B.; MAIA, M.; VIANA, W.; ANDRADE, R. M. C. An adaptive context acquisition framework to support mobile spatial and context-aware applications. In: LIANG, S.; WANG, X.; CLARAMUNT, C. (Ed.). *Web and Wireless*

Geographical Information Systems. Springer Berlin Heidelberg, 2013, (Lecture Notes in Computer Science, v. 7820). p. 100–116. ISBN 978-3-642-37086-1. Disponível em: <http://dx.doi.org/10.1007/978-3-642-37087-8_8>.

GEORGE, M. L. *Lean Six Sigma For Service: How to Use Lean Speed and Six Sigma Quality to Improve Services and Transactions*. [S.l.]: McGraw-Hill, 2003.

GRINE, H.; DELOT, T.; LECOMTE, S. Adaptative query processing in mobile environment. *Proc. of the 3rd international workshop on Middleware*, 2005.

GYGI, C.; DECARLO, N.; WILLIAMS, B. *Six Sigma for Dummies*. [S.l.]: Wiley Publishing, Inc., 2005.

HARRY, M. Six sigma: A breakthrough strategy for profitability. 1998.

HASHMI, S.; BAIK, J. Quantitative process improvement in xp using six sigma tools. In: *Computer and Information Science, 2008. ICIS 08. Seventh IEEE/ACIS International Conference on*. [S.l.: s.n.], 2008. p. 519–524.

HOLLER, R. *Mobile Application Development: A Natural Fit with Agile Methodologies*. [S.l.], 2006.

HOLZER, A.; ONDRUS, J. Trends in mobile application development. In: HESSELMAN, C.; GIANNELLI, C. (Ed.). *Mobile Wireless Middleware, Operating Systems, and Applications - Workshops*. Springer Berlin Heidelberg, 2009, (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, v. 12). p. 55–64. ISBN 978-3-642-03568-5. Disponível em: <http://dx.doi.org/10.1007/978-3-642-03569-2_6>.

IHME, T.; ABRAHAMSSON, P. *The Use of Architectural Patterns in the Agile Software Development of Mobile Applications*. 2004.

ILIEVA, S.; IVANOV, P.; STEFANOVA, E. Analyses of an agile methodology implementation. In: *Proceedings of the 30th EUROMICRO Conference*. Washington, DC, USA: IEEE Computer Society, 2004. p. 326–333. ISBN 0-7695-2199-1. Disponível em: <http://portal.acm.org/citation.cfm?id=1018420.1019696>.

INC., M. *Minitab - Software for Quality Improvement*. 2014. Disponível em: http://www.minitab.com/pt-br/. Acesso em: 01 de Setembro de 2014.

INSTITUTE, P. M. *A Guide to the Project Management Body of Knowledge (PMBOK Guide) Fourth Edition ed.* [S.l.: s.n.], 2008.

JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. Unified software development process. *Addison-Wesley*, 1999.

JEONG, Y.-J.; LEE, J.-H.; SHIN, G.-S. Development process of mobile application sw based on agile methodology. In: *Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference on*. [S.l.: s.n.], 2008. v. 1, p. 362 –366. ISSN 1738-9445.

KITCHENHAM, B. *Procedures for Performing Systematic Reviews*. 2004.

KNIBERG, H. *Scrum Checklist – version 2.0*. 2009. Disponível em: http://blog.crisp.se/2009/08/14/henrikkniberg/1250265360000. Acesso em: 01 de Setembro de 2014.

LAYMAN, L.; WILLIAMS, L. A.; CUNNINGHAM, L. Exploring extreme programming in context: An industrial case study. In: *Agile Development Conference*. [S.l.: s.n.], 2004. p. 32–41.

MARTIN, R. C. *Agile Software Development: Principles, Patterns, and Practices*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2003. ISBN 0135974445.

MARTIN, R. C. *Desenvolvimento Ágil de Software: Princípios, Padrões e Práticas*. Upper Saddle River, NJ, EUA: Prentice Hall PTR, 2003. ISBN 0135974445.

MELLO, R. M. d.; SILVA, P. C. d.; TRAVASSOS, G. H. Agilidade em processos de software: Evidências sobre características de agilidade e práticas Ágeis. In: . [S.l.]: XIII Simpósio Brasileiro de Qualidade de Software, 2014.

MOE, N.; DINGSØYR, T. Scrum and team effectiveness: Theory and practice. In: ABRAHAMSSON, P.; BASKERVILLE, R.; CONBOY, K.; FITZGERALD, B.; MORGAN, L.; WANG, X. (Ed.). *Agile Processes in Software Engineering and Extreme Programming*. Springer Berlin Heidelberg, 2008, (Lecture Notes in Business Information Processing, v. 9). p. 11–20. ISBN 978-3-540-68254-7. Disponível em: <http://dx.doi.org/10.1007/978-3-540-68255-4_2>.

PAN, Z.; PARK, H.; BAIK, J.; CHOI, H. A six sigma framework for software process improvements and its implementation. In: *Software Engineering Conference, 2007. APSEC 2007. 14th Asia-Pacific*. [S.l.: s.n.], 2007. p. 446 –453. ISSN 1530-1362.

POPPENDIECK, M. Principles of lean thinking. 2005.

POPPENDIECK, M.; CUSUMANO, M. Lean software development: A tutorial. *Software, IEEE*, v. 29, n. 5, p. 26–32, 2012. ISSN 0740-7459.

QUMER, A.; HENDERSON-SELLERS, B. An evaluation of the degree of agility in six agile methods and its applicability for method engineering. *Inf. Softw. Technol.*, Butterworth-Heinemann, Newton, MA, USA, v. 50, n. 4, p. 280–295, mar. 2008. ISSN 0950-5849. Disponível em: <http://dx.doi.org/10.1016/j.infsof.2007.02.002>.

QUMER, A.; HENDERSON-SELLERS, B.; MCBRIDE, T. *Agile adoption and improvement model*. 2007.

QUMER, B. H.-S. A. *COMPARATIVE EVALUATION OF XP AND SCRUM USING THE 4D ANALYTICAL TOOL (4-DAT)*. [S.l.]: European and Mediterranean Conference on Information Systems (EMCIS), 2006.

RAHIMIAN, V.; RAMSIN, R. Designing an agile methodology for mobile software development: A hybrid method engineering approach. In: *Research Challenges in Information Science, 2008. RCIS 2008. Second International Conference on*. [S.l.: s.n.], 2008. p. 337 –342.

RAJAPAKSE, D. C. *Fragmentation of Mobile Applications*. 2008. Disponível em: `http://www.comp.nus.edu.sg/~damithch/df/device-fragmentation.htm`. Acesso em: 01 de Setembro de 2014.

RASMIN, R. *The Engineering of an Object-Oriented Software Engineering Methodology*. Tese (Doutorado) — University of York, York, UK, 2006. Disponível em: <http://www.cs.york.ac.uk/ftpdir/reports/YCST-2006-12.pdf>.

RORIZ, H. *Can Scrum Support Lean Six Sigma?* 2010. Disponível em: http://www.scrumalliance.org/articles/161-can-scrum-support-six-sigma. Acesso em: 01 de Setembro de 2014.

RUBIN, K. S. *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. 1st. ed. [S.l.]: Addison-Wesley Professional, 2012. ISBN 0137043295, 9780137043293.

SALO, O.; ABRAHAMSSON, P. Agile methods in european embedded software development organisations: a survey on the actual use and usefulness of extreme programming and scrum. *Software, IET*, v. 2, n. 1, p. 58 –64, february 2008. ISSN 1751-8806.

SCHWABER, K. Scrum development process. In: *Proceedings of the 10th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications (OOPSLA*. [S.l.: s.n.], 1995. p. 117–134.

SCHWABER, K. *Agile Project Management with Scrum*. [S.l.]: Microsoft Press, 2007.

SCHWABER, K. *The Enterprise and Scrum*. [S.l.]: Microsoft Press, 2007.

SCHWABER KEN E BEEDLE, M. *Desenvolvimento Ágil de Software com Scrum*. Upper Saddle River, NJ, EUA: Prentice Hall PTR, 2001. ISBN 0130676349.

SEVERINO, A. J. *Metodologia do trabalho científico*. São Paulo: Cortez, 2007.

SHEN, M.; YANG, W.; RONG, G.; SHAO, D. Applying agile methods to embedded software development: A systematic review. In: *Software Engineering for Embedded Systems (SEES), 2012 2nd International Workshop on*. [S.l.: s.n.], 2012. p. 30–36.

SHORE, J.; WARDEN, S. *The art of agile development*. First. [S.l.]: O'Reilly, 2007. ISBN 9780596527679.

SIDKY, A.; ARTHUR, J.; BOHNER, S. A disciplined approach to adopting agile practices: the agile adoption framework. *Innovations in Systems and Software Engineering*, Springer-Verlag, v. 3, n. 3, p. 203–216, 2007. ISSN 1614-5046. Disponível em: <http://dx.doi.org/10.1007/s11334-007-0026-z>.

SOARES, M. S. Comparação entre metodologias Ágeis e tradicionais para o desenvolvimento de software. *INFOCOMP Journal of Computer Science*, v. 3, n. 2, p. 8–13, 2004.

TAYNTOR, C. B. *Six Sigma Software Development; 2nd ed.* Hoboken, NJ: Taylor & Francis Ltd, 2007.

VODE, B.; SUTHERLAND, J. *Scrum But Test*. 2008. Disponível em: http://antoine.vernois.net/scrumbut/?page=test&lang=en. Acesso em: 01 de Setembro de 2014.

WANG, X. The combination of agile and lean in software development: An experience report analysis. In: *Agile Conference (AGILE), 2011*. [S.l.: s.n.], 2011. p. 1–9.

WELLS, D. *Extreme Programming: A gentle introduction*. 2009. Disponível em: http://www.extremeprogramming.org/. Acesso em: 01 de Setembro de 2014.

WERKEMA, C. *CRIANDO A CULTURA LEAN SEIS SIGMA*. [S.l.]: Elsevier, 2012. (ENGENHARIA). ISBN 9788535254266.

YIN, R. K. *Estudo de caso: planejamento e métodos*. 4ed. ed. [S.l.]: Bookman, 2010.

APÊNDICE A – RESUMO DA AVALIAÇÃO DOS TRABALHOS RELACIONADOS

Tabela A.1: Pontos Positivos e de Melhoria dos Trabalhos Relacionados

Trabalhos Relacionados	**Pontos Positivos**	**Pontos de Melhoria**
Mobile-D	• Reutiliza práticas do *XP* e *Crystal* • Há indícios de utilização de métricas para avaliação do processo ágil • Foco centrado no usuário	• Documentação superficial e incompleta
HME	• Processo abrange desde a concepção até a comercialização do produto • Inclui a análise de negócio • Fornece suporte à LPS • Há uma etapa de revisão da qualidade	• Documentação superficial e incompleta • Não apresenta estudo de caso
MASAM	• Processo abrange desde a concepção até a comercialização do produto	• Documentação superficial e incompleta • Não apresenta estudo de caso
XP e *Six Sigma*	• Melhoria do desempenho de práticas do *XP* em projetos de *software* • Propõe um mapeamento de ferramentas do *Six Sigma* às atividades do *XP* • Exemplifica o uso desse mapeamento • Controle estatístico de processos, modelos de estimativas e previsão	• Documentação superficial do estudo de caso realizado • Não especifica uma nova abordagem, apenas sugere como as metodologias podem ser combinadas
Scrum e *Six Sigma*	• Propõe um mapeamento entre os papéis do *Six Sigma* e *Scrum* • Controle estatístico dos processos	• Não apresenta uma fundamentação teórica nem prática para o mapeamento • Não apresenta estudo de caso • Não especifica uma nova abordagem, apenas sugere como as metodologias podem ser combinadas
SLeSS	• Propõe um mapeamento de técnicas do *LSS* às atividades do *Scrum* e vice-versa • Melhoria do uso dessas metodologias e dos processos de customização • Controle estatístico de processos, resultados de melhoria de produtividade, redução de defeitos e de horas extras	• Documentação superficial e incompleta • Utiliza o *DMAIC* sem simplificações • Restrito à customização de *software* para DMs
4-DAT	• Possibilita a análise e comparação de metodologias ágeis • Fornece o cálculo de agilidade	• Sozinho, esse método não pode ser utilizado para avaliação e melhoria do uso de práticas ágeis em projetos de *software*
AAF	• Avaliação e melhoria de práticas ágeis em projetos de *software* e na organização	• Utiliza cerca de 300 indicadores, o que torna esse método extenso
Nokia Test	• Avaliação da forma de utilização do *Scrum* em projetos de *software* • Fornece o cálculo do nível do *Scrum*	• Pouco abrangente, não avalia parte dos princípios e valores, eventos e papéis do *Scrum*
CRISP Checklist	• Avaliação da forma de utilização do *Scrum* em projetos de *software*	• Não fornece o cálculo do nível do *Scrum* • Inclui a avaliação de práticas recomendadas (não obrigatórias) o que o torna desnecessariamente extenso
MiniDMAIC	• Ciclo de melhoria de 1 a 6 semanas • Conhecimento básico de estatística • Custo menor que o *DMAIC* tradicional • Melhoria de produtividade e redução de defeitos	• O foco desse método são os problemas gerais dos projetos de *software* e, dessa forma, ele é não focado na avaliação e melhoria do processo ágil