SLeSS 2.0: Uma Evolução da Abordagem de Integração do Scrum e Lean Six Sigma para Aplicações Móveis

Thiago Ferraz V. da Cunha, Rossana M. C. Andrade*

[1]Grupo de Redes de Computadores, Engenharia de Software e Sistemas (GREat)
Mestrado e Doutorado em Ciência da Computação (MDCC)
Universidade Federal do Ceará (UFC), Fortaleza – Ceará – Brasil

{thiagoferraz, rossana}@great.ufc.br

Abstract. *This paper proposes an evolution of an existing approach, called SLeSS, which integrates Scrum to Lean Six Sigma to software customization for mobile phones. The evolution of this approach, called SLeSS 2.0, adds software development in addition to customization and improves the integration mechanisms initially proposed, focusing on the evaluation of the practices and principles of Scrum from the use of Lean Six Sigma techniques. 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 paper.*

Resumo. *Este trabalho propõe 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 agora 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 neste artigo.*

1. Introdução

O desenvolvimento de *software* para dispositivos móveis (DMs), aqui considerados os dispositivos portáteis capazes de se comunicar através de uma tecnologia de comunicação sem fio, ou, de forma simplificada, o desenvolvimento de aplicações móveis, 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 também impõem uma alta competitividade e, por conseguinte, exigem níveis elevados de produtividade e de qualidade dos projetos de desenvolvimento [Fonteles et al. 2013].

Esse desenvolvimento precisa lidar com as limitações de recursos dos dispositivos, com a dependência do *hardware* e atender aos requisitos de desempenho, mobilidade, segurança e interface com o usuário [Shen et al. 2012]. Além disso, as frequentes mudanças de requisitos e ambiente, os prazos agressivos e as altas pressões de competitividade são outros desafios desse desenvolvimento [Dantas et al. 2009].

*Bolsista de produtividade DT 2 (CNPq): 314021/2009-4

Além do desenvolvimento de aplicações móveis, surge a necessidade de customizar o *software* embarcado para dispositivos como celulares e *tablets* para atender os 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 a fim de atender as 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 utilizadas por se adequarem bem às frequentes mudanças de requisitos e às demandas de prazo desse nicho de mercado, bem como às frequentes mudanças inerentes a esse desenvolvimento [Corral et al. 2013]. Além disso, essas metodologias contribuem na melhoria da produtividade das equipes de desenvolvimento e da qualidade dos produtos desenvolvidos [Dybå e Dingsøyr 2008].

Metodologias como o *Scrum* [Schwaber 2007] e o *Lean Six Sigma* (*LSS*) [George 2003] possuem objetivos distintos, entretanto, elas podem contribuir juntas no desenvolvimento de *software* para DMs [Cunha e Andrade 2014a]. 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 reúso 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 *LSS* 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 agora com foco na avaliação das práticas e princípios do *Scrum* a partir do uso de técnicas do *LSS*. Para avaliar a nova versão, ela é aplicada em sete projetos reais de desenvolvimento e customização de *software* para DMs e seus resultados são também discutidos neste artigo.

Para apresentar o *SLeSS 2.0*, este artigo está estruturado em mais cinco seções. Os trabalhos relacionados ao uso de metodologias ágeis e de gestão da qualidade no desenvolvimento e customização de *software* para DMs são discutidos na Seção 2. A Seção 3 apresenta o *SLeSS* 2.0, detalhando a sua definição, os papéis, o processo de execução, os mecanismos de integração e um comparativo das versões da abordagem. Os resultados da aplicação do *SLeSS* 2.0 em sete projetos reais são discutidos na Seção 4 e, finalmente, na Seção 5, o artigo é concluído com as principais contribuições e os trabalhos futuros.

2. Trabalhos Relacionados

Para encontrar os trabalhos relacionados ao desenvolvimento para dispositivos móveis foi realizada uma busca na literatura por publicações nos seguintes tópicos: (i) Metodologias ágeis; e (ii) Metodologias de gestão da qualidade. Além disso, dois outros temas foram pesquisados, a seguir: (iii) Combinação de metodologias ágeis e de gestão da qualidade no desenvolvimento de *software*; e (iv) Adoção, melhoria e avaliação de práticas ágeis e modelos de maturidade.

Para realização dessa pesquisa bibliográfica, embora não tenha sido realizada uma revisão sistemática, foram definidas *strings* e locais de busca conforme descrito em [Kitchenham 2004]. Como locais de busca, foram utilizadas as bases *ACM DL Library*[1]

[1]http://dl.acm.org

e *IEEE Explorer*[2], bem como os anais do Simpósio Brasileiro de Engenharia de *Software* (*SBES*) e do Simpósio Brasileiro de Qualidade de *Software* (*SBQS*).

Em relação às abordagens ágeis para o desenvolvimento de *software* para DMs, pode-se citar o *Mobile-D*, o *HME* [Rahimian e Ramsin 2008] e o *MASAM* [Jeong et al. 2008]. O *Mobile-D* apresenta alguns indícios de que utiliza métricas para a avaliação do processo ágil, mas, de fato, essas abordagens não adotam uma metodologia de gestão da qualidade. O ponto fraco comum a elas é que suas documentações estão incompletas e superficiais, dificultando a análise e o reúso das mesmas.

Em se tratando do uso de metodologias de gestão da qualidade no desenvolvimento para DMs, [Bezerra et al. 2009] *et al.* propõem o *MiniDMAIC* e demonstram o uso desse método em projetos reais, com resultados de melhoria de produtividade e redução de defeitos. Por isso, o *MiniDMAIC* foi considerado na evolução da abordagem proposta neste trabalho.

Há ainda trabalhos que propõem a combinação de metodologias ágeis e de qualidade. Por exemplo, Hashmi e Baik [Hashmi e Baik 2008] propõem um mapeamento de ferramentas do *Six Sigma* às práticas do *XP*. Já Roriz [Roriz 2010] sugere uma combinação do *Scrum* ao *Six Sigma*, apresentando um mapeamento entre os papéis dessas metodologias. Cunha *et. al* [Cunha et al. 2011], citados na Seção 1, 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 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]. Entretanto, devido à ausência de sua documentação, o *SLeSS*, em sua primeira versão, torna-se de difícil reúso e é restrito à customização de *software* para DMs.

Por fim, também há trabalhos relacionados à adoção, melhoria e avaliação de práticas ágeis e modelos de maturidade, os quais também foram considerados na evolução da abordagem proposta neste trabalho, como por exemplo, o *4-DAT* [Qumer e Henderson-Sellers 2008], o *AAF* [Sidky et al. 2007], e os *checklists* de avaliação do *Scrum* propostos em [Vode e Sutherland 2008] e [Kniberg 2009].

Dessa forma, este trabalho objetiva propor uma evolução da abordagem *SLeSS* que solucione as lacunas deixadas na primeira versão e nos outros trabalhos elencados nesta seção, focando nos seguintes pontos: (i) Melhoria da adoção e aplicação dos princípios e práticas ágeis aos processos de desenvolvimento e customização de *software*; (ii) Adequação da abordagem para o seu reúso 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 (iii) Adequação da abordagem ao desenvolvimento, além da customização, de *software* para DMs.

Mais detalhes desses trabalhos, bem como a descrição completa de todos os trabalhos relacionados, podem ser encontrados na dissertação de mestrado em [Cunha e Andrade 2014b].

[2]http://ieeexplore.ieee.org

3. 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*) com foco na gestão de projetos e na melhoria de processos de desenvolvimento e customização de *software* para DMs. As principais contribuições do *SLeSS* 2.0 podem ser resumidas a seguir:

- ✓ Aumentar a agilidade da gestão de projetos, do desenvolvimento e da customização de *software* a partir de princípios e práticas do *Scrum*;
- ✓ Melhorar e controlar 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 1. A abordagem reutiliza os fundamentos, valores e princípios do *Scrum*, bem como os princípios do *LSS*, procurando garantir que sejam adequadamente seguidos. Além disso, o *SLeSS* 2.0 estabelece um conjunto de mecanismos para a integração dessas metodologias, definidos através da experimentação empírica para o desenvolvimento e customização de *software* para dispositivos como celulares, *smartphones* e *tablets*.

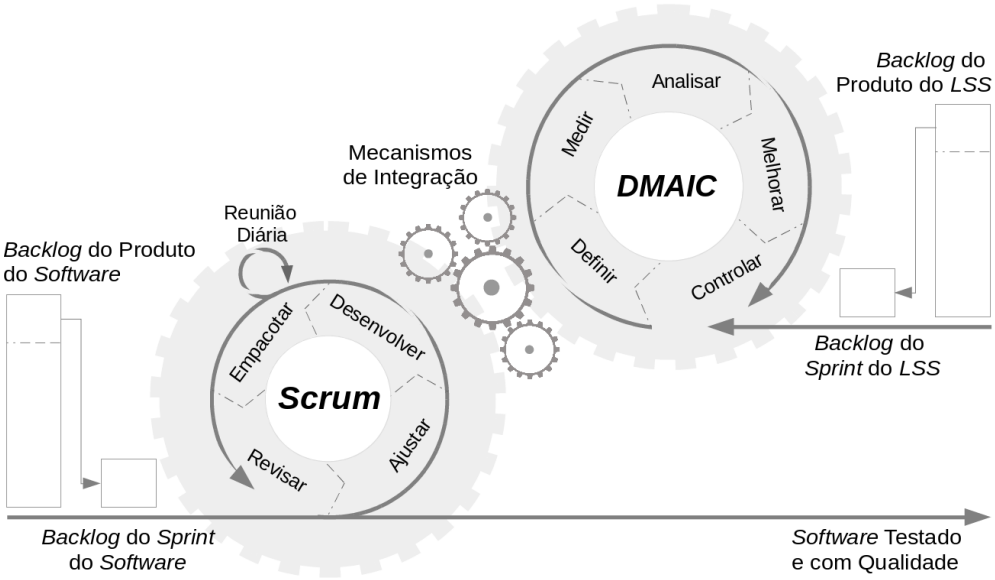

Figura 1. *SLeSS* 2.0 - Uma Abordagem de Integração do *Scrum* e do *LSS*

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 *Software*, e outro para gerenciar as melhorias dos processos de customização e desenvolvimento, *Backlog* do *LSS*. Enquanto o *Backlog* do *Software* é elaborado a partir da identificação dos requisitos funcionais e não-funcionais provenientes da visão do produto, o *Backlog* 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.

3.1. Papéis

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

O *PO* 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 *PO* como *Green Belt* possibilita que ele priorize adequadamente os itens do *Backlog* do Produto do *Software* e os do *LSS*.

Já o *SM*, 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.

3.2. Processo de Execução

O processo de execução do *SLeSS* 2.0, apresentado na Figura 2, é uma inovação da nova versão da abordagem. Dessa forma, pretende-se atigir um reúso sistemático da abordagem. 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 *PO* e o *SM* 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 *LSS* geralmente ainda não foi elaborado, e, nesse momento, há apenas o do *Software*. Nessa etapa, o *PO* 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 *LSS*. O *PO* 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 *PO* 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*.

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

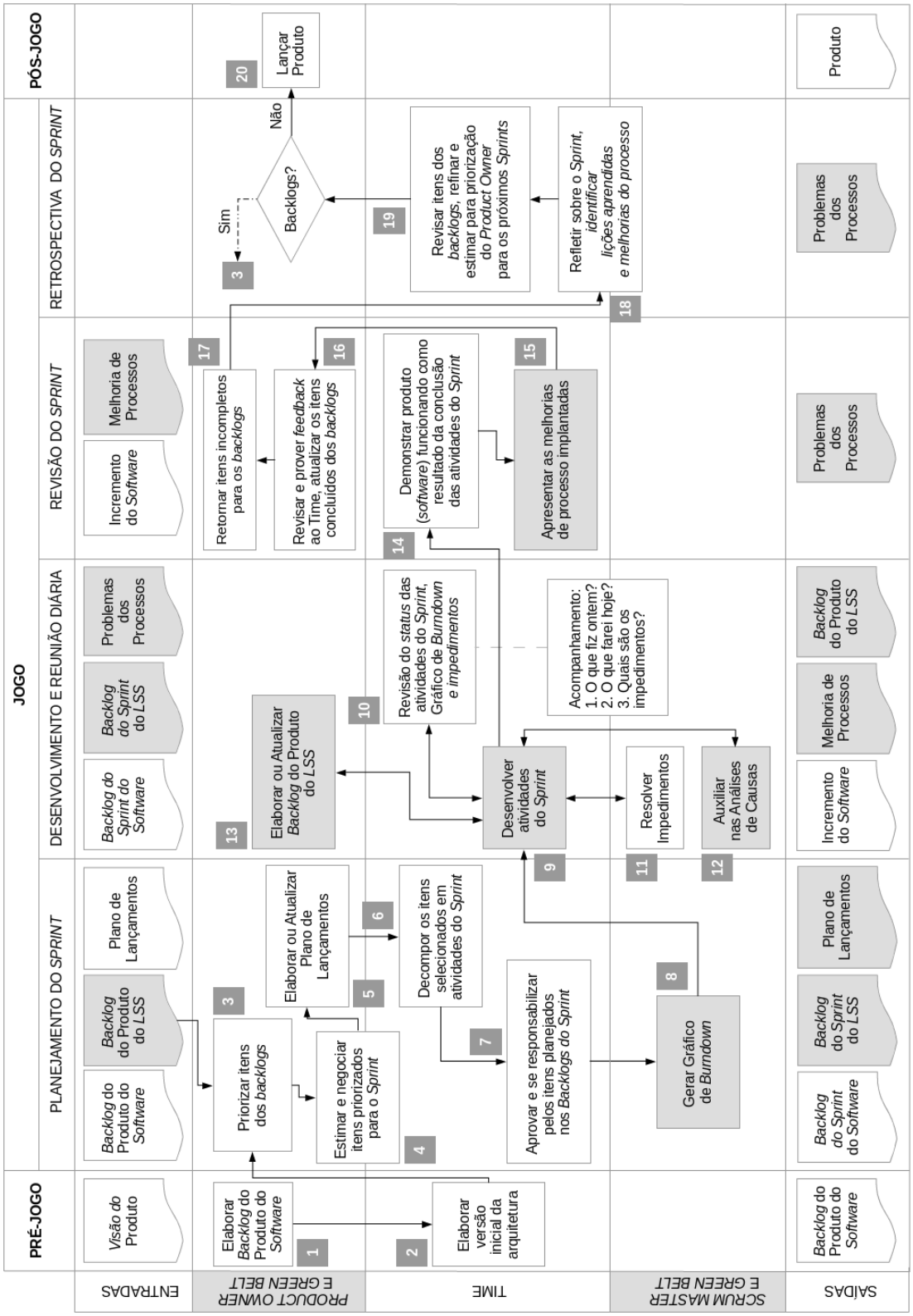

Figura 2. Processo de Execução do *SLeSS* 2.0

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 (passos 6 e 7). Em seguida, o *SM* 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 *SM* 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 *SM* 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, *SM* ou mesmo pelo *PO* e são insumos na elaboração do *Backlog* 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 *PO*, resultado do desenvolvimento das atividades do *Sprint* (passo 14). Em seguida, também são apresentadas as melhorias implantadas nos processos de desenvolvimento (passo 15). O *PO* 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 *PO* 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).

3.3. Mecanismos de Integração

Os seis mecanimos da versão incial foram condensados em quatro no *SLeSS* 2.0, para fortalecer a combinação do *Scrum* ao *LSS*. Os mecanismos de integração do *SLeSS* 2.0 são estratégias de combinação dos princípios e práticas do *Scrum* aos do *LSS* e vice-versa, propostos com base na experimentação empírica realizada em projetos reais. O *SLeSS* 2.0 estabelece quatro mecanismos, conforme são apresentados a seguir:

- *Fases do LSS em Sprints* - 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*. Para cada projeto de melhoria, um *Backlog* do Produto do *LSS* é elaborado considerando o escopo das fases do *DMAIC*.
- *Ferramentas do LSS 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.

- *Treinamentos do LSS nos Sprints* - A preparação e a realização de treinamentos no *LSS* são realizados durante os *Sprints*, fornecendo o conhecimento necessário para que a equipe atue adequadamente nas melhorias dos processos. Esses treinamentos podem ser teóricos e práticos, realizados a partir de palestras, seminários, fóruns e auto-estudo. Os membros da equipe, por exemplo, o *PO* e o *SM*, são responsáveis pelo planejamento e execução dos treinamentos.
- *LSS 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.

3.4. *Agile DMAIC*

O *Agile DMAIC* [Cunha e Andrade 2014a] é um método 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. As principais contribuições do *Agile DMAIC* podem ser resumidas a seguir:

✓ Possibilitar a gestão quantitativa da implementação do *Scrum*;
✓ Avaliar o uso de princípios e práticas do *Scrum*; 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 3 e os detalhes são brevemente apresentados no decorrer desta seção. O *Agile DMAIC* é composto de três componentes:

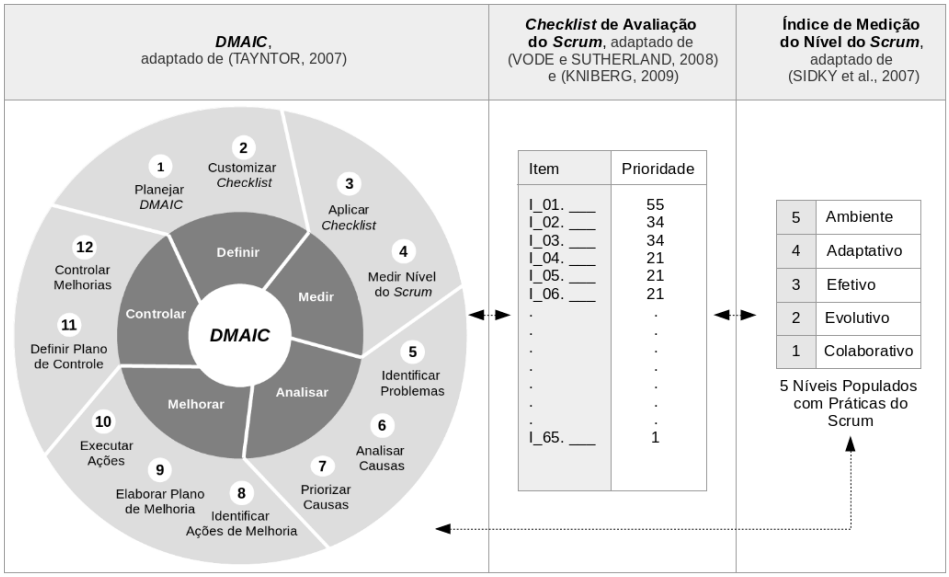

Figura 3. Visão Geral do *Agile DMAIC*

- *DMAIC*, uma submetodologia do *LSS* [George 2003] voltada à melhoria de processos. Neste trabalho, o ciclo do *DMAIC* proposto para o *Agile DMAIC* é uma simplificação do *DMAIC* [Tayntor 2007] e 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*), elaborado a partir do *Nokia Test* [Vode e 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. Esse novo *checklist* é aplicado à equipe do projeto para obter os pontos fortes e fracos da atual implementação do *Scrum*.
- Índice de Medição do Nível do *Scrum* (*IMS*), que avalia a adoção das práticas do *Scrum* por projeto. O *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 *AAF* [Sidky et al. 2007], a partir de sua simplificação ao remover as práticas e conceitos não relacionados ao *Scrum*.

O *Agile DMAIC* se baseia em coletar periodicamente a percepção da equipe do projeto (*PO*, *SM* 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.

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.

3.5. Comparativo das Versões da Abordagem

A nova versão, *SLeSS* 2.0, foi elaborada com foco na melhoria do uso de princípios e práticas do *Scrum* a partir de técnicas do *LSS* no desenvolvimento e customização de *software* para DMs. As principais alterações na versão inicial (SLeSS) são descritas a seguir:

- *Mecanismos de Integração* - A versão inicial possui seis mecanismos, que foram condensados em quatro na nova versão. Essa reestruturação objetivou estabelecer mecanismos fortemente relacionados à combinação do *Scrum* ao *LSS*.
- *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* - 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, um deles é selecionado como piloto, que passa então pela implantação do *Scrum*, *LSS* e, finalmente, pela adoção dos mecanismos de integração do *SLeSS* 2.0; e
- *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, a documentação do processo de execução e do mapeamento dos papéis do *Scrum* ao *LSS* foram adicionados à nova versão.

4. Aplicando o *SLeSS* 2.0 em Projetos Reais

Esta seção apresenta uma avaliação qualitativa e quantitativa da aplicação do *SLeSS* 2.0 em projetos reais. As informações coletadas são discutidas e também quantificadas, utilizando técnicas estatísticas para serem classificadas e avaliadas [Severino 2007].

4.1. Objetivo do Estudo de Caso

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 DMs, priorizando a análise dos seguintes critérios:

(i) *Assertividade* - Avalia o percentual de assertividade da abordagem na identificação de problemas no uso de princípios e práticas do *Scrum* nos projetos;
(ii) *Exatidão* - Avalia o percentual de exatidão da abordagem na identificação de problemas no uso de princípios e práticas do *Scrum*;
(iii) *Capacidade de identificar e analisar causas de problemas* - 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* - Avalia como a abordagem auxilia as equipes na evolução da agilidade dos projetos a partir da priorização de ações de melhoria para a evolução da agilidade do projeto.

4.2. Sobre os Projetos Reais

Os projetos desse estudo de caso foram desenvolvidos no Grupo de Redes de Computadores, Engenharia de *Software* e Sistemas (*GREat*), vinculado ao Departamento de Computação (DC) da Universidade Federal do Ceará (UFC), e selecionados por serem voltados ao desenvolvimento e customização de *software* para DMs. Um resumo desses projetos, com as informações obtidas dos gerentes e líderes através de um questionário aplicado via *Web*, é apresentado na Tabela 1.

Tabela 1. Resumo dos Projetos do Estudo de Caso

Projeto	Tipo	Duração Planejada	Tamanho da Equipe	Complexidade
App1	D&I	9 meses	8	Alta
App2	P&D&I	9 meses	7	Alta
App3	P&D&I	9 meses	8	Alta
App4	D&I	2 anos	6	Média
App5	P&D&I	2 anos	26	Alta
App6	P&D&I	6 meses	6	Alta
App7	P&D&I	9 meses	10	Alta

Os projetos são voltados a P&D&I (Pesquisa, Desenvolvimento e Inovação) de aplicações para *smartphones*, com exceção do App1 que realiza apenas o D&I de aplicações para *smartphones* e do *App5* que não se decidiu por utilizá-lo. A inclusão desse projeto no estudo de caso foi realizada, então, para analisar preliminarmente se a abordagem consegue diferenciar adequadamente entre os projetos que usam o *Scrum* e aqueles que não o utilizam.

As equipes, formadas por pesquisadores e desenvolvedores de áreas da computação e engenharia, estão localizadas geograficamente em um mesmo sítio, entretanto, os clientes são todos remotos. Os membros dessas equipes possuem, em sua maioria, uma alocação integral 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.

4.3. Aplicando a Abordagem

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 4.

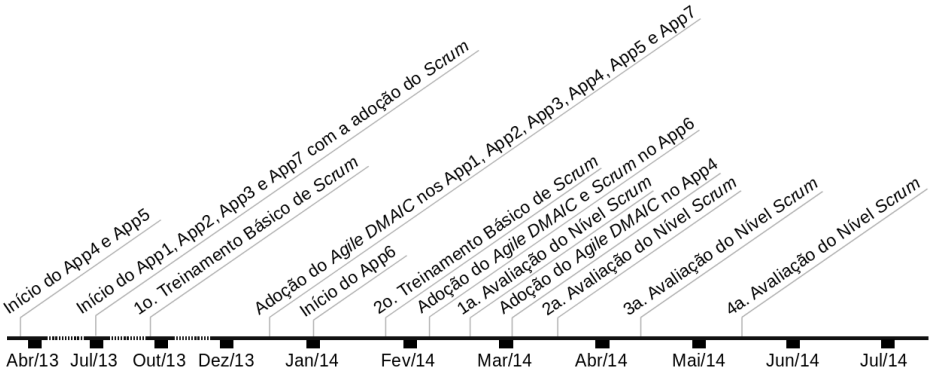

Figura 4. Linha de Tempo dos Treinamentos, Adoção do *Scrum* e *Agile DMAIC* nos Projetos

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 a última avaliação ainda não havia se decidido por utilizá-lo.

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.

4.4. Discussão dos Resultados

O gráfico de *Boxplot* dos projetos do estudo de caso é apresentado na Figura 5. Esse gráfico é utilizado para a análise da variação do indicador do nível do *Scrum* nesses projetos. Essa variação 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.

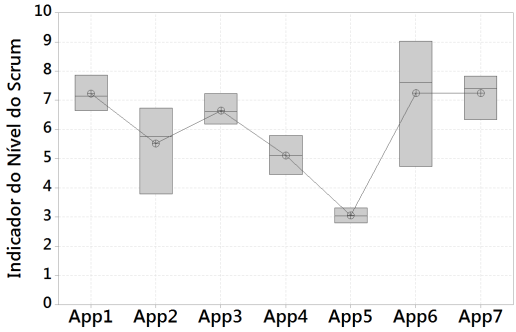

Figura 5. Boxplot dos Projetos

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.

O gráfico de controle dos projetos que utilizam o *Scrum* é apresentado na Figura 6. 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, referente a janeiro de 2014, e da avaliação das principais vulnerabilidades no uso do *Scrum* nesses projetos. Entretanto, mesmo nessa fase inicial, as equipes sinalizaram que o *Agile DMAIC* foi coerente em suas avaliações e que possibilitou a identificação e resolução dos principais problemas no uso do *Scrum*.

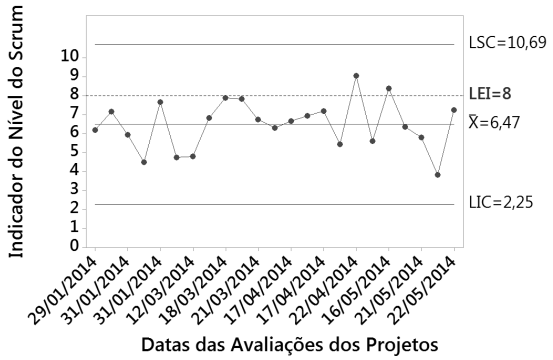

Figura 6. Controle dos Projetos que Utilizam o *Scrum*

O gráfico de controle da assertividade do *Agile DMAIC* durante as avaliações dos projetos é apresentado na Figura 7. 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 do *Scrum*.

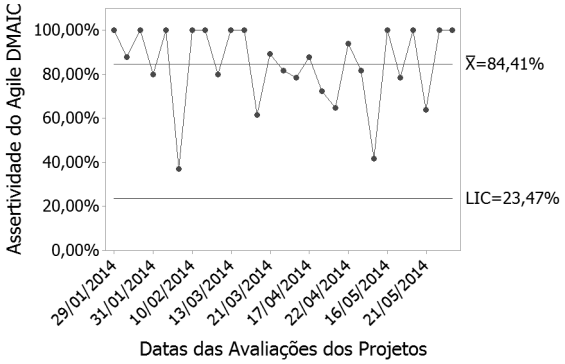

Figura 7. Controle da Assertividade do *Agile DMAIC* nas Avaliações dos Projetos

O gráfico de controle da exatidão do *Agile DMAIC* durante as avaliações dos projetos é apresentado na Figura 8. 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.

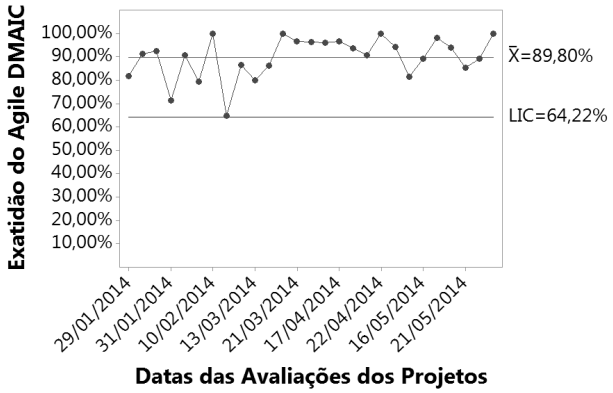

Figura 8. Controle da Exatidão do *Agile DMAIC* nas Avaliações dos Projetos

O gráfico de dispersão entre a exatidão do *Agile DMAIC* e a participação dos integrantes do projeto nas avaliações realizadas é apresentado na Figura 9. 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, a exatidão do *Agile DMAIC* é alta, indicando que nessas condições o método é eficaz na identificação dos problemas relacionados ao uso do *Scrum*.

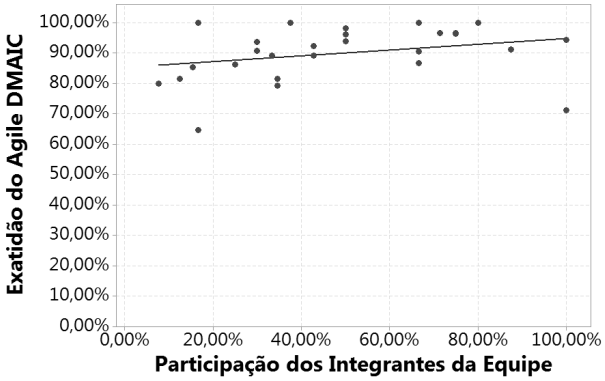

Figura 9. Dispersão entre a Exatidão do *Agile DMAIC* e a Participação dos Integrantes

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. 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. Conclusão e Trabalhos Futuros

No cenário do desenvolvimento e customização de *software* para DMs, o *SLeSS* 2.0, abordagem proposta, atende ao objetivo de evoluir a sua versão inicial, *SLeSS* [Cunha et al. 2011], agora 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 da abordagem, este trabalho propôs o método *Agile DMAIC*, que está contido em um dos mecanismos de integração do *SLeSS* 2.0. Esse método possibilitou avaliar de forma eficaz o uso de princípios e práticas do *Scrum* nos projetos. O *Agile DMAIC* foi aplicado em projetos reais e seus resultados possibilitaram identificar que, mesmo com tempo e experiência no *Scrum*, as equipes necessitavam avaliar de que forma estavam utilizando os princípios e práticas do *Scrum*. Essa avaliação periódica do uso do *Scrum*, a partir da percepção das equipes, possibilitou identificar e tratar os principais problemas no uso dessa metodologia.

Para avaliar 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.

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. Além disso, o *Agile DMAIC* pode ser aprimorado a partir da simplificação de seus componentes. 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

Bezerra, C. I. M., Coelho, C. C., Gonçalves, F. M. G. S., Pires, C. G. S., e Telles, Gabriela, A. A. B. (2009). Minidmaic: Uma abordagem para análise e resolução de causas em projetos de desenvolvimento de software. In *SBQS 2009 - Trabalhos Técnicos*.

Corral, L., Sillitti, A., e Succi, G. (2013). 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*, pages 19–24.

Cunha, T. F. V. d. e Andrade, R. M. C. (2014a). Agile dmaic: Um método para avaliar e melhorar o uso do scrum em projetos de software. In *SBQS 2014 - Trabalhos Técnicos*.

Cunha, T. F. V. d. e Andrade, R. M. C. (2014b). Sless 2.0: uma evolução da abordagem de integração do scrum e lean six sigma para aplicações móveis. Dissertação de Mestrado.

Cunha, T. F. V. d., Dantas, V. L. L., e Andrade, R. M. C. (2011). Sless: A scrum and lean six sigma integration approach for the development of sofware customization for mobile phones. In *SBES*, pages 283–292. IEEE.

Dantas, V. L. L., Marinho, F. G., Costa, A. L. D., e Andrade, R. M. C. (2009). Testing requirements for mobile applications. In *International Symposium on Computer and Information Sciences*, pages 555–560.

Dybå, T. e Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Inf. Softw. Technol.*, 50:833–859.

Fonteles, A. S., Neto, B., Maia, M., Viana, W., e Andrade, R. M. C. (2013). An adaptive context acquisition framework to support mobile spatial and context-aware applications. In Liang, S., Wang, X., e Claramunt, C., editors, *Web and Wireless Geographical Information Systems*, volume 7820 of *Lecture Notes in Computer Science*, pages 100–116. Springer Berlin Heidelberg.

George, M. L. (2003). *Lean Six Sigma For Service: How to Use Lean Speed and Six Sigma Quality to Improve Services and Transactions*. McGraw-Hill.

Hashmi, S. e Baik, J. (2008). Quantitative process improvement in xp using six sigma tools. In *Computer and Information Science, 2008. ICIS 08. Seventh IEEE/ACIS International Conference on*, pages 519–524.

Jeong, Y.-J., Lee, J.-H., e Shin, G.-S. (2008). Development process of mobile application sw based on agile methodology. In *Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference on*, volume 1, pages 362 –366.

Kitchenham, B. (2004). Procedures for performing systematic reviews. Keele university. technical report tr/se-0401, Department of Computer Science, Keele University, UK.

Kniberg, H. (2009). Scrum checklist – version 2.0. Disponível em: `http://blog.crisp.se/2009/08/14/henrikkniberg/1250265360000`. Acesso em: 27 de Junho de 2015.

Qumer, A. e Henderson-Sellers, B. (2008). An evaluation of the degree of agility in six agile methods and its applicability for method engineering. *Inf. Softw. Technol.*, 50(4):280–295.

Rahimian, V. e Ramsin, R. (2008). 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*, pages 337 –342.

Roriz, H. (2010). Can scrum support lean six sigma? Disponível em: `http://www.scrumalliance.org/articles/161-can-scrum-support-six-sigma`. Acesso em: 27 de Junho de 2015.

Schwaber, K. (2007). *Agile Project Management With Scrum*. Microsoft Press, Redmond, WA, USA.

Severino, A. J. (2007). *Metodologia do trabalho científico*. Cortez, São Paulo, 23 edition.

Shen, M., Yang, W., Rong, G., e Shao, D. (2012). Applying agile methods to embedded software development: A systematic review. In *Software Engineering for Embedded Systems (SEES), 2012 2nd International Workshop on*, pages 30–36.

Sidky, A., Arthur, J., e Bohner, S. (2007). A disciplined approach to adopting agile practices: the agile adoption framework. *Innovations in Systems and Software Engineering*, 3(3):203–216.

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

Vode, B. e Sutherland, J. (2008). Scrum but test. Disponível em: `http://antoine.vernois.net/scrumbut/?page=test&lang=en`. Acesso em: 27 de Junho de 2015.