

Utilização de práticas de Melhoria Contínua no desenvolvimento ágil de software: Estudo de caso em uma empresa do ramo metal mecânico

Celso Luiz Gonçalves (DEP/UFSCar) celso.goncalves@ufscar.br

Fabiane Letícia Lizarelli (DEP/UFSCar) fabiane@dep.ufscar.br

José Carlos de Toledo (DEP/UFSCar) toledo@dep.ufscar.br

Resumo

Melhorar continuamente os processos é objetivo constante nas organizações e se torna um fator diferencial na busca de vantagem competitiva. Várias abordagens têm sido utilizadas com o objetivo de melhorar continuamente os processos, tais como a Gestão da Qualidade Total (GQT), *Lean* e Seis Sigma. O intuito dessas abordagens é aumentar a eficiência e a qualidade dos processos evitando desperdícios e retrabalhos, e melhorando a satisfação do cliente. Para tanto, é necessário tratar a Melhoria Contínua (MC) como um processo de resolução de problemas. Isso se aplica também no processo de desenvolvimento de softwares, da área de Tecnologia da Informação (TI), que é o foco desse trabalho, utilizando-se de um conjunto de ferramentas e técnicas apropriadas. O presente trabalho tem por objetivo evidenciar a aplicação de práticas de MC na resolução de problemas no processo de desenvolvimento ágil de software. Apresenta-se, discute-se e analisa-se os resultados da aplicação das práticas de MC no processo de desenvolvimento ágil de um software de controle numa empresa do ramo metalúrgico. Para investigação do referido processo utilizar-se-á a revisão bibliográfica, como suporte teórico para descrição e análise de um estudo de caso.

Palavras chave: Melhoria Contínua, Ferramentas da Qualidade, Desenvolvimento Ágil de Software.

Use of Continuous Improvement practices in agile software development: a case research in Mechanical Industry

Abstract

Continuous Improvement is a constant aim in organizations and becomes a differential factor to get competitive advantage. Several approaches have been used in order to continuously improve the processes, such as Total Quality Management (TQM), Lean and Six Sigma. The aim of these approaches is to increase the efficiency and quality and avoiding the waste and rework and increase customer satisfaction. For this, it is necessary to treat the continuous improvement (CI) as a problem-solving process, as such in the Agile Software Development process, of the Information Technology (IT) using a set of appropriate of tools and techniques. The aim of this article is to show the practical application of CI in solving problems in the process of Agile Software Development. For the research process will be used the Literature Review to get theoretical support for the analysis of a case research.

Key-words: Continuous Improvement, Quality Tools, Agile Software Development, Scrum

1. Introdução

A dinâmica do ambiente dos negócios coloca as organizações em uma grande pressão para que elas possuam respostas rápidas às mudanças. Isso força as organizações a encontrar meios

para se tornar mais flexíveis e ágeis para realizar seus trabalhos. Essas mudanças também foram percebidas na área de desenvolvimento de software e com interesse nos métodos que abordem tais agilidades (SWAMINATHAN; JAIN, 2012). Segundo Beck et al., (2001) esses métodos seguem um conjunto comum de valores como enunciado no Manifesto Ágil. Outro assunto que vem ganhando destaque na área de desenvolvimento de software é o método ágil SCRUM elaborado por Schwaber (2004).

Os métodos ágeis de desenvolvimento de software possuem o foco na flexibilidade em termos de resultado e agilidade na entrega do produto para o cliente (VLAANDEREN; JANSEN; BRINKKEMPER; JASPERS, 2010). Entretanto o foco no desenvolvimento de práticas de MC está no fato da criação de valor aos processos, diminuição da variabilidade, eliminação de desperdícios e melhorias de padrões existentes (IMAI, 2011). Para superar os pontos fracos de uma abordagem e maximizar os pontos fortes de outra, algumas empresas recentemente tem mesclado diferentes iniciativas de MC e desenvolvimento ágil de software e que juntas resultam em um abordagem combinada que alcança melhores resultados que qualquer um individualmente. Observa-se que há um consenso em relação a sobreposição nos princípios e práticas de desenvolvimento ágil de software e MC e o fato é que ambos complementam um ao outro, e parece haver uma necessidade de mais estudos empíricos nessa direção (SWAMINATHAN; JAIN, 2012).

Este artigo pretende contribuir descrevendo um estudo de caso no setor industrial na implementação de práticas de MC como um processo que auxilia a resolução de problemas em um projeto de desenvolvimento ágil de software utilizando o método SCRUM.

O artigo está estruturado da seguinte forma. A Seção 2 apresenta o referencial teórico relativo a conceituação de MC e a metodologia ágil de desenvolvimento de software. A seção 3 apresenta o método de pesquisa, apresentação da empresa do estudo de caso, as etapas desenvolvimento SCRUM com a aplicação das práticas de MC, os resultados alcançados, bem como, as lições aprendidas e finalmente na seção 4 apresenta as análises e conclusões.

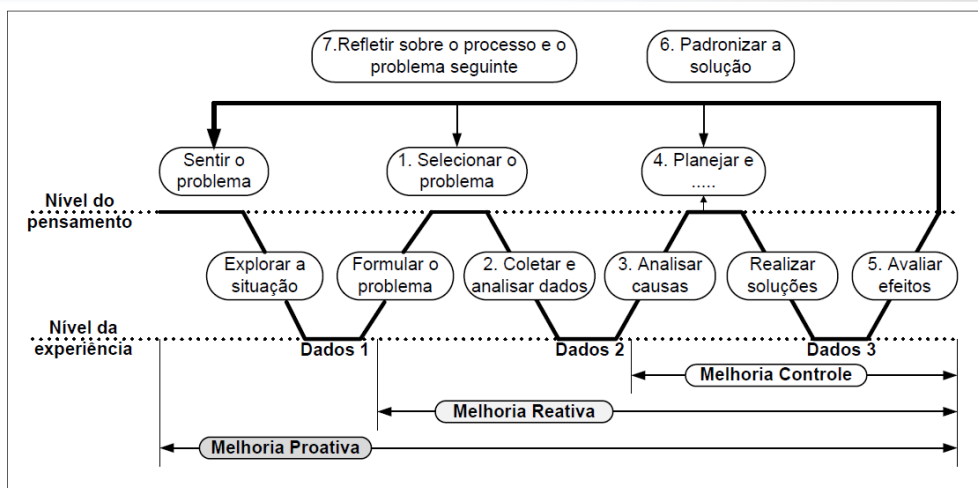
2. Pesquisa Bibliográfica

2.1 Conceituação de Melhoria Contínua

O desempenho dos negócios está relacionado à capacidade de a empresa gerir seus processos e suas operações de maneira eficiente, o que pode ser alcançado por meio da aplicação efetiva das práticas de MC (BESSANT et al., 1994; BESSANT; CAFFYN; GALAGHER, 2001). Jha et al., (1996) definem MC como um conjunto de atividades que constituem um processo de raciocínio e intervenção que busca alcançar a melhoria de desempenho. Trata-se de todos trabalhando juntos para melhorar, com foco nas metas organizacionais e sem necessariamente grandes investimentos de capital (BESSANT; FRANCIS, 1999; BHUIYAN; BAGEL, 2005).

2.2 Tipos de Melhoria Contínua: Modelo WV

De acordo com Shiba et al., (1997) existem três tipos de melhoria que podem ser observados por meio do modelo WV na figura 1. O modelo WV é um auxílio para compreender os três estágios evolutivos da melhoria da qualidade. O modelo WV descreve a forma geral da resolução do problema como alternância entre o nível de pensamento (reflexão, planejamento, análise) e o nível de experiência (obtenção de informação do mundo real, através de entrevistas, experimentos ou mensurações numéricas). O caminho entre estes dois níveis ao longo do tempo tem a forma de um W, seguido de um V, por isso a denominação WV.



Fonte: Shiba et al., (1997)

Figura1: Modelo WV com os três tipos de resolução de problemas

2.2.1 Melhoria pelo controle de processos

Se o processo produz resultados fora de seus limites de controle, deve-se executar ações corretivas padrões normalmente descritas no manual de manutenção para corrigir o problema do processo. De acordo com Shiba et al., (1997), este ciclo muito conhecido no TQM (*Total Quality Management*) e é designado pelo acrônimo SDCA (*Standart = padronizar, Do = executar, Check = verificar, Action = atuar*). O ciclo de controle de processo é uma proposta de solução padronizada até que possam ser identificadas as causas para que o processo possa operar de acordo com o padrão anterior. Nota-se que não existe atividade para a busca da causa do problema (TOLEDO et al., 2013).

2.2.2 Melhoria pelo método reativo

A continuidade das ações decorrentes de melhoria por controle de processos é denominado melhoria reativa quando se busca identificar as causas do problema e sua solução. Em muitas ocasiões é necessário eliminar o problema que ocasiona o desvio do padrão no resultado do processo já existente de modo a prevenir a recorrência, e em outras, é necessário melhorar o processo para um resultado de maior capacidade. Em geral deve-se utilizar um modelo estruturado eficaz e eficiente de solução de problemas (SHIBA; WALDEN, 2002). O objetivo da abordagem reativa requer uma sistemática de trabalho e análise mais estruturada utilizando se, por exemplo, os 07 Passos da Melhoria Reativa e as 07 Ferramentas do Controle da Qualidade conforme evidenciado no quadro1:

PDCA	07 Passos da Melhoria Reativa	07 Ferramentas do Controle de Qualidade
Planejar	1- Definir o problema	Lista de verificação, Gráficos, Diagrama de Pareto, histograma, diagrama de correlação, diagrama de causa e efeito.
	2- Coletar dados	
	3- Analisar as causas	
Executar	4- Planejar e realizar a solução	Lista de verificação, gráficos, diagrama de Pareto, histograma, diagrama de correlação, diagrama de causa e efeito, cartas de controle.
Verificar	5- Avaliar os efeitos	
	6- Padronizar a solução	
Atuar	7- Refletir sobre o processo	

Fonte: Shiba et al., (1997)

Quadro 1 - Relação do ciclo PDCA, 07 Passos da Melhoria Reativa, 07 Ferramentas do Controle da Qualidade

2.2.3 Melhoria pelo método proativo

Nas situações nas quais se faz necessário trocar um processo existente por outro mais eficiente, ou mesmo criar um novo processo para aproveitar uma nova oportunidade, a abordagem pela melhoria proativa é indicada. Entretanto o escopo de uma situação proativa é frequentemente mais amplo e menos estruturado que nos processos de melhoria por controle de processos e nos processos de melhoria reativa. Requer mais ciclos entre os níveis de conhecimento e o nível da prática (SHIBA; WALDEN, 2002). No modelo (WV) métodos proativos são usados para explorar amplamente a área problema e concentrarem-se em um problema específico ao qual as 07 etapas do Controle de Qualidade serão aplicadas. Para situações mais complexas é proposto a utilização de 07 novas ferramentas do Planejamento da Qualidade apresentadas no Quadro 2.

PDCA	07 Etapas do Controle da Qualidade	07 Ferramentas para o Controle da Qualidade	07 novas Ferramentas do Planejamento da Qualidade
Planejar	1. Selecionar o tema 2. Coletar e analisar dados 3. Analisar a causa	Lista de verificação, gráficos, diagrama de Pareto, histograma, diagrama de correlação, diagrama de causa-e efeito.	Diagrama de Afinidades, Diagrama de relações, Diagrama matricial.
Executar	4. Planejar e implementar a solução		Diagrama de árvore, Diagrama matricial, Diagrama de setas, Diagrama PDPC(diagrama de processo decisório)
Verificar	5. Avaliar os efeitos	Lista de verificação, gráficos, diagrama de Pareto, histograma, diagrama de correlação, diagrama de causa-e efeito, cartas de controle.	
Atuar	6. Padronizar a solução 7. Refletir sobre o processo (e o problema seguinte)		Diagrama de setas, Diagrama PDPC(diagrama de processo decisório), Diagrama de afinidades.
(Provê repetições)	(Provê etapas)	(Provê ferramentas)	(Provê ferramentas)

Fonte: Shiba et al., (1997)

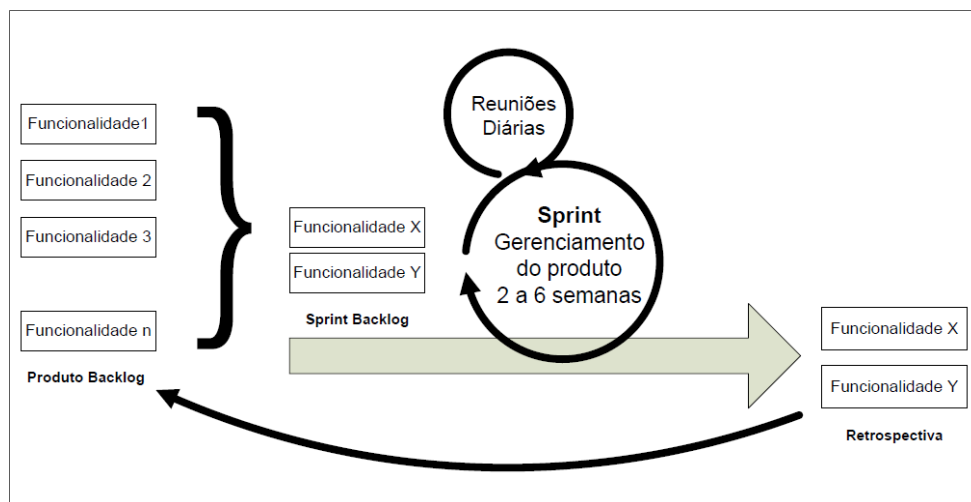
Quadro 2 - As 07 novas Ferramentas do Planejamento da Qualidade em Relação ao PDCA, as 07 Etapas do Controle de Qualidade e as 07 Ferramentas do Controle de Qualidade

2.3 Métodos ágeis de desenvolvimento de softwares

Os métodos tradicionais de desenvolvimento de software têm o foco na geração de documentação sobre o projeto e no cumprimento rígido de processos e é fortemente influenciado pelas práticas da produção enxuta (CARVALHO et al., 2012). Já os métodos ágeis concentram esforços na entrega do produto e nas interações entre os indivíduos. Nesses métodos ágeis, é minimizada a fase de planejamento inicial, de modo que os desenvolvedores se concentram em entregar um produto ao fim de cada iteração, ao invés de traçar diretrizes e planejamentos para o projeto como um todo (CARVALHO et al., 2012). Diversas abordagens são utilizadas como métodos ágeis para o gerenciamento de projetos (ABRAHASSON et al., 2003). Exemplos de tais métodos ágeis são: DSDM (*Dynamic System Development Method*) (STAPLETON, 1999), Extreme Programing (BECK, 1999), Feature Driven Development (PALMER; FELSING, 2002) e o SCRUM (SCHWABER, 1995). Um fator importante nos métodos ágeis é a facilidade de adaptação perante as mudanças de escopo e resolução de problemas que acontecem durante o projeto pelo fato do método não focar somente em registro da documentação e conseguir receber, avaliar e responder as mudanças com o objetivo de aumento considerável na produtividade (BECK, 2001).

2.3.1 Método Ágil Scrum

Conforme Schwaber (2004), o Scrum baseia-se em características de flexibilização na entrega de resultados, agilidade nos prazos de entregas, utilização de equipes pequenas, revisões mais frequentes do escopo, maior colaboração entre os membros da equipe e orientação. A figura 2 ilustra a visão dinâmica do modelo e o fluxo de informações no método ágil Scrum. No início do projeto, cliente e desenvolvedores definem o *Product Backlog*, ou seja, a lista de requisitos e funcionalidades para o software. Também são definidas as datas para entrega de resultados parciais e totais. A partir da priorização mais favorável ao cliente, são estimados os custos do projeto e uma análise inicial de riscos é realizada.



Fonte: Adaptado de Vlaanderen et al., (2011)

Figura 2 - Fluxo de informação no método ágil Scrum

As principais práticas gerenciais associados ao Scrum, segundo Schwaber (2004), são:

- **Product Backlog:** conjunto de requisitos, formando uma lista de atividades a serem desenvolvidas durante o empreendimento;
- **Sprint Backlog:** subconjunto do *Product Backlog* é uma lista de atividades prioritárias que serão desenvolvidas na *Sprint*.
- **Sprint:** atividades rápidas e focadas que implementam os itens do *Sprint Backlog*, sua duração são cerca de duas a seis semanas;
- **Sprint Closure:** acontece após termino da *Sprint* na fase final e a equipe prepara a entrega do produto.

Segundo Schwaber (2004), uma característica importante é a criação de infraestrutura com recursos necessários para assegurar desempenho nas atividades de desenvolvimento ágil. É necessário formar equipes treinadas e com alto nível de maturidade em gerenciamento de projetos e em MC composto pelos seguintes membros:

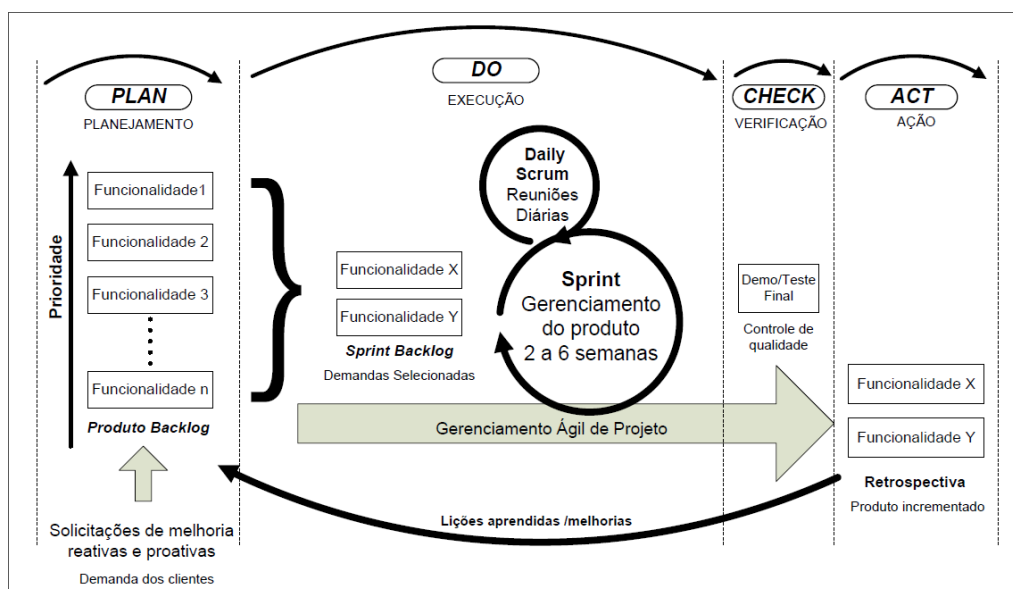
- **Scrum Master:** É responsável pelo sucesso e continuidade e melhoria do método do Scrum, ensinando o processo para todos os envolvidos no projeto, também deve conseguir a conciliação entre a cultura da empresa e o método e entregar os resultados esperados, assegurando que todos sigam as regras e práticas do Scrum (SCHWABER, 2004);
- **Product Owner:** É o representante dos interesses do cliente e serve como suporte no projeto e no aplicativo resultante. Participa também do registro das necessidades do cliente na lista de requisitos denominado *Product Backlog*. Nessa lista, estão as demandas e funcionalidades priorizadas segundo o valor para o produto final, devendo ser focadas as demandas com maiores prioridades a serem realizadas (SCHWABER, 2004);

• **Project Team:** Equipe responsável por desenvolver uma funcionalidade. Os times são auto geridos, auto-organizados, e multifuncionais, sendo responsáveis por transformar os requisitos em funcionalidades, assumindo uma responsabilidade coletiva pelo sucesso do projeto (SCHWABER, 2004).

2.4 Relações entre práticas de melhoria contínua no desenvolvimento ágil de software

Outro método muito importante para a melhoria é o ciclo PDCA, que consiste em uma série de etapas que auxiliam no controle de processos na melhoria reativa e proativa. O Ciclo PDCA implícito no modelo WV de Shiba, e também no método Scrum, tem o conceito de realimentar a melhoria – voltar no ciclo para trabalhar no problema seguinte ou aprofundar a melhoria de um processo já aperfeiçoado. Cada *Sprint* (ou corrida) com a entrega parcial do produto para o cliente é uma iteração que segue o ciclo PDCA (LACERDA. et al., 2011).

O princípio de iteração do PDCA permite realizar melhorias por etapas, executando dentro de ciclos de melhoria relativamente curtos e objetivos. Esse tipo de iteração está alinhado com as linhas do pensamento ágil (COHEN, 2010). A figura 3 ilustra a relação do ciclo de PDCA com o desenvolvimento ágil Scrum.



Fonte: Adaptado de Vlaanderen. et al., (2011)

Figura 3 - Relação do PDCA com o método ágil Scrum

2.5 Descrição das Etapas dos Processos e principais ferramentas

• **Planejamento/Produto Backlog:** Etapa em que a equipe composta de *Scrum Master*, *Product Owner* e *Project Team* recebem, registra e classifica em termos de viabilidade técnica e econômica as novas demandas. O resultado desse processo é o *Sprint Backlog*, ou seja, a relação do escopo a ser desenvolvido na etapa seguinte (SCHWABER, 2004; VLAANDEREN et al., 2011).

• **Execução/Sprint Backlog:** Nessa etapa a equipe de desenvolvimento de software recebe o *Sprint Backlog* e executa as atividades de produção do software de acordo com as funcionalidades desejadas e requisitos pré-estabelecidos (SCHWABER, 2004).

• **Execução/Daily Scrum:** Rápidas reuniões que ocorrem todos os dias entre os membros da equipe, definindo quais tarefas serão realizadas no dia na qual também se discutem os resultados das atividades do dia anterior (SCHWABER, 2004).

• **Verificação/Test:** Cada requisito é verificado e testado pela equipe de desenvolvimento com presença do cliente. Quando todos os requisitos do *Sprint Backlog* são atendidos pelas funcionalidades o software ou parte dele (módulo) é aprovado e os custos são levantados, registrados e comparados com uma referência (VLAANDEREN et al., 2011).

• **Ação /Sprint Closure:** No final de cada *Sprint* é realizada uma reunião de análise crítica em que a equipe discute melhorias e registram as lições aprendidas para os próximos *Product backlog*. Isso permite que a equipe gradualmente melhore o processo não apenas aprendendo com seus próprios erros, mas também com os de outras equipes (VLAANDEREN et al., 2011). Nessa fase há extensa utilização práticas de MC na busca de resolução de problemas tanto nas melhorias reativas listadas no *Product Backlog* como nas proativas. As principais ferramentas utilizadas nessa etapa são listas de verificação com a relação das necessidades dos clientes e as funcionalidades desejadas.

O quadro 3 apresenta um resumo de cada ciclo do PDCA, as etapas correspondentes da metodologia Scrum, a descrição do processo envolvido e as ferramentas para o Controle da Qualidade.

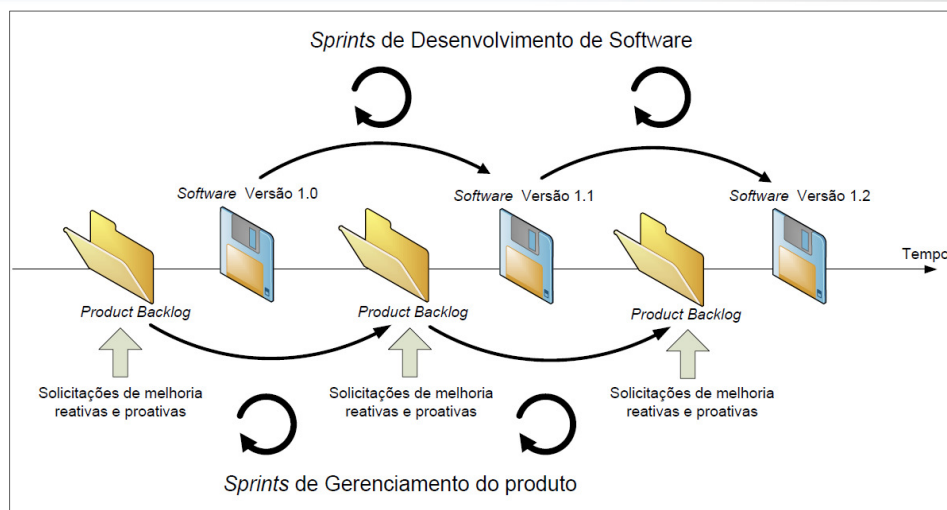
Ciclo PDCA	Etapas Scrum	Descrição do Processo	Ferramentas para o Controle Qualidade
Planejamento	<i>Product Backlog</i>	Recebimento e Classificação das demandas	Lista de verificação, gráficos, diagrama de Pareto, histograma.
Execução	<i>Sprint Backlog</i>	Seleção das funcionalidades a serem executadas	Lista de verificação
	<i>Sprint</i>	Atividades de desenvolvimento	
	<i>Daily Scrum</i>	Reuniões diárias de monitoramento e controle	Lista de verificação, gráficos, histograma, cartas de controle.
Verificação	<i>Test</i>	Atividades de teste com a presença do cliente	Lista de verificação, gráficos, diagrama de Pareto, histograma, diagrama de correlação, diagrama de causa e efeito, cartas de controle.
Ação	<i>Sprint Closure</i>	Validação do produto	Lista de verificação/controle

Fonte: Próprio autor

Quadro 3 - Resumo dos ciclos, etapas, processos e principais ferramentas do Scrum

2.6 Gerenciamento de desenvolvimento de software e os ciclos de melhorias

A figura 4 ilustra o conceito de *Sprints* alternados. A linha horizontal do tempo mostra a sinergia entre o desenvolvimento de software e o gerenciamento do produto alternando num duplo ciclo de realimentação (VLAANDEREN et al., 2011).



Fonte: Adaptado de Vlaanderen et al., 2011

Figura 4 - *Sprints* alternados de desenvolvimento de software e gerenciamento do produto

Nos *Sprints* de desenvolvimento de software há um fluxo de melhoria no conjunto de funcionalidade do aplicativo, ou seja, o foco de melhoria é representado por meio de disponibilidade de versões do software. Nos *Sprints* de gerenciamento de produto é possível observar a entrada das solicitações de melhorias reativas e proativas no sentido de um gerenciamento mais amplo do produto. A alternância de *Sprints* e o uso consistente de um conjunto de ferramentas ajudam a organização identificar e resolver problemas, melhorar os produtos e processos e pode ser entendida como uma cultura de melhoria contínua (BHUIYAN; BAGHEL; WILSON, 2006).

3. Estudo de caso

Segundo Bryman (1989), a pesquisa qualitativa tende a lidar melhor com aspectos processuais da realidade da organização, sendo a compreensão dessa realidade necessária para que se identifiquem novas evidências de aplicações e resultado. Para Yin (1994), o estudo de caso caracteriza-se por uma investigação empírica de um fenômeno dentro de seu contexto da vida real explorando as condições entre o fenômeno e o contexto em que este está inserido. O presente estudo de caso é retrospectivo e único que permite maior aprofundamento porém baixo grau de generalização.

3.1 A empresa do estudo de caso

A empresa é de capital nacional do setor metalúrgico de fabricação de bens e possui um programa interno de melhoria da qualidade na área de TI. Por ser uma empresa voltada a grandes projetos, muitos aplicativos e sistemas de informação são específicos e não há disponibilização no mercado tradicional de software, sendo necessário o desenvolvimento interno tipo *ad-hoc* (sob encomenda). O Departamento de Engenharia do Produto da empresa (cliente interno) apresentava a necessidade de um software de gerenciamento eletrônico de documentos, (GED) para controlar e monitorar a emissão, revisão e *workflow* de seus documentos técnicos. O Departamento de TI analisou as necessidades de todos os *stakeholders* do projeto por meio de um levantamento completo dos mapas de processos, dos principais requisitos e funcionalidades e optou por um projeto piloto com a utilização pela primeira vez na empresa da adoção do método ágil Scrum para o desenvolvimento do aplicativo GED.

3.2 Projeto de desenvolvimento

Esta seção relata as principais etapas, processos, ferramentas e resultados da aplicação do método Scrum conjuntamente com as práticas de MC no desenvolvimento do aplicativo.

3.2.1 Etapa do Planejamento: *Product backlog*

O levantamento inicial de requisitos junto ao cliente é uma etapa do processo de desenvolvimento de software onde são identificadas as demandas das necessidades dos usuários, do sistema a ser desenvolvido, as atividades que estes esperam que sejam automatizadas ou facilitadas e as informações que desejam receber do sistema. Esse levantamento é registrado por meio de uma planilha ou lista de verificação que permite também o controle e monitoramento das várias etapas do método Scrum. A Tabela 1 evidencia a Lista de Verificação utilizada no projeto e serviu como uma ferramenta de controle para todas as demais etapas e atividades do projeto. Além de ser uma lista de registros das demandas dos clientes também foi considerada uma ferramenta útil para análises e tomada de decisões nas reuniões diárias e mensais descritas em etapas posteriores.

Item	Product Backlog (Demandas e Melhorias)	Aplicabilidade	Prioridade	Etapas da Sprint	Horas Previstas	Horas realizadas	Índice de desempenho	Sprint backlog
ENG1	Realizar relatórios de pendências gerais : Relatório de documentos com o cliente, com o fornecedor e com a fabrica	Aplicável	Alta	Análise de Requisitos	5	4	1,25	3
				Design	7	7	1,00	
				Produção	8	9	0,89	
				Testes	8	4	2,00	
ENG14	Emitir Relatórios de Status dos documentos técnicos atrelados a eventos de milestones	Aplicável	Baixa	Análise de Requisitos	8	7	1,14	4
				Design	10	4	2,50	
				Produção	12	10	1,20	
				Testes	8	6	1,33	
ENG18	Ao aprovar um documento na Fase de APROVAÇÃO gerar copia PDF automaticamente para todos os documentos	Aplicável	Baixa	Análise de Requisitos	20	18	1,11	3
				Design	30	22	1,36	
				Produção	40	32	1,25	
				Testes	20	12	1,67	
ENG24	Relatório do HISTORICO DE REVISÕES dos documentos do projeto	Aplicável	Alta	Análise de Requisitos	4	3	1,33	1
				Design	8	5	1,60	
				Produção	10	8	1,25	
				Testes	4	2	2,00	
ENG 29	Relatório de documentos com status de processo "Não liberados para fabricação"	Em Análise	Média	Análise de Requisitos				
				Design				
				Produção				
				Testes				
ENG12	Colocar um pequeno visualizador do documento na barra lateral (idem ao Microsoft Explorer)	Não Aplicável	Baixa	Análise de Requisitos				
				Design				
				Produção				
				Testes				
ENG5	Geração de Código automático para documentos de engenharia	Aplicável	Alta	Análise de Requisitos	15	12	1,25	2
				Design	15	10	1,50	
				Produção	20	18	1,11	
				Testes	10	8	1,25	
ENG7	Ao subir revisão avisar todos e se afeta a produção avisar da subida da revisão e emitir e-mails	Aplicável	Alta	Análise de Requisitos	10	11	0,91	1
				Design	15	16	0,94	
				Produção	20	25	0,80	
				Testes	10	12	0,83	
ENG10	Emitir e-mails de aviso para processos finalizados no workflow	Aplicável	média	Análise de Requisitos	15	10	1,50	5
				Design	20	15	1,33	
				Produção	25	18	1,39	
				Testes	10	8	1,25	

Tabela 1 - Lista de Verificação e os controles de *Product Backlog* e *Sprint Backlog*

3.2.2 Etapa de Execução/*Sprint Backlog*

Na Tabela 1 pode se observar na coluna *Sprint Backlog* o controle da priorização para as atividades de *Sprints* que a equipe de projeto de desenvolvimento tem a realizar. As atividades de classificação de priorização das demandas (*Product Backlog*) foram realizadas em reuniões mensais com os seguintes participantes: *Scrum Master*, *Product Owner* e representante do *Project Team* que normalmente é um profissional de sistemas de TI.

A equipe de planejamento do projeto também informa a equipe de TI as quantidades de horas previstas de cada fase (Análise de requisitos, *Design* de software, Produção e Testes)

conforme observado na coluna “Horas Previstas” na tabela 1. Como dito anteriormente sendo um projeto piloto utilizou-se parâmetros de valores de horas previstas de projetos anteriores e de métodos tradicionais de desenvolvimento de software como base de referência.

Nessas reuniões são analisadas as melhorias solicitadas, a sua aplicabilidade, os recursos disponíveis, análises e resoluções de problemas. Para tanto, utiliza-se de diversas ferramentas da qualidade. O quadro 4 evidencia as principais atividades da etapa de execução do *Sprint Backlog* e as ferramentas de MC utilizadas na resolução de problemas.

Atividades principais <i>Sprint Backlog</i>	Ferramentas utilizadas
Seleção das demandas conforme aplicabilidade	Lista de demandas (<i>Product backlog</i>) histograma,
Análise de prioridades	Cartas de controle, indicadores de desempenho.
Previsão de horas de execução	Lista de recursos disponíveis

Fonte: Próprio autor

Quadro 4 - Atividades das etapas do *Sprint Backlog* e ferramentas utilizadas

3.2.3 Etapa de Execução/*Sprint*

Nessa etapa as equipes de desenvolvedores de software realizam o trabalho de produção de cada item do escopo do *Sprint Backlog* dentro dos requisitos estabelecidos. Após finalização de cada atividade foram registradas as horas realizadas na planilha de controle da tabela 5. Pode-se observar na coluna de Índice de Desempenho, ou seja, a relação entre horas previstas e horas realizadas que houve uma redução no *lead time* dos *Sprints*.

3.2.4 Etapa de Execução/*Daily Scrum*

As reuniões diárias foram realizadas sempre com a liderança do *Scrum Master* e cada participante da equipe pode relatar de forma objetiva sobre o trabalho realizado, o progresso conseguido e as dificuldades em prosseguir. As principais ferramentas da qualidade utilizadas nessa etapa foram as listas de verificação e histogramas.

3.2.5 Etapa de Verificação/*Test*

Nesta etapa a equipe de projeto realizou vários testes de qualidade de software relativo aos testes de aderência, ou seja, verificar se o aplicativo desenvolvido estava em conformidade aos requisitos e funcionalidades necessárias. O software GED foi apresentado para o cliente e para todas as partes envolvidas no processo de desenvolvimento e se caso não apresentasse nenhum problema o cliente apresentava o relatório de aceitação.

3.2.6 Etapa de Ação/*Sprint Closure*

Uma vez aprovado o relatório de aceitação pelo cliente na etapa anterior a equipe do projeto realizou treinamentos detalhados aos usuários chaves (*key-users*), ou seja, usuários com maior entendimento do processo e do aplicativo. Após esse treinamento finalizado a equipe realizou reuniões de análise crítica entre os participantes *Scrum Master*, *Product Owner*, e representante do *Project team* para análise de todo o processo de desenvolvimento com registro formal e divulgação das lições aprendidas que auxiliarão futuros projetos de desenvolvimento ágil de software na empresa.

3.2.7 Lições aprendidas

Durante a implementação do método ágil Scrum e a utilização de práticas de MC, neste estudo de caso, a equipe de projetos absorveu experiências importantes para futuros desafios nessa abordagem. As lições aprendidas mais importantes nesse caso foram:

- Agregação de valor com utilização de Ciclos alternados de *Sprints* com o duplo ciclo de iteração entre o desenvolvimento do software e o gerenciamento do ciclo de vida do produto
- A administração da lista do *Product Backlog* requer disciplina das partes envolvidas;
- Observou-se que a grande maioria das solicitações de melhorias dos usuários focava práticas de Melhoria Reativa
- A utilização dos 07 Passos da Melhoria Reativa e das 07 ferramentas do Controle de Qualidade nos ciclos de melhoria apresentam um ganho em termos de sistematização de resolução de problemas;
- Práticas de Melhoria Proativa recebidas das partes envolvidas no projeto e registrados no *Product Backlog* denotam a preocupação de aproveitar as possíveis oportunidades na busca de maior desempenho dos processos e consequentemente da empresa;
- Reuniões Diárias efetivas (*Scrum Daily*) e utilização de práticas de MC tem um papel importante na busca de solução de problemas;
- Importância da integração com o cliente e colaboração entre as equipes e gerencia de projetos da área de TI;
- Apoio e infraestrutura de TI na manutenção de software por meio de um canal de atendimento aos usuários em solucionar demandas de apoio, tais como, problemas de hardware, problemas de rede interna de computadores.

4. Análise e conclusões

Nesse trabalho foi apresentado um estudo de caso de um projeto com objetivo de evidenciar a aplicação de práticas de MC na resolução de problemas no processo de desenvolvimento ágil de software. No caso estudado a utilização de métodos ágeis no desenvolvimento de software necessitou de um ambiente de infraestrutura adequada de TI e de equipes treinadas e alinhadas em resolução de problemas com utilização de ferramentas adequadas da qualidade.

O mapeamento de processos e levantamento detalhado das necessidades dos clientes internos foram atividades importantes para a definição e priorização das demandas de melhoria reativas e proativas, proporcionando, assim, um escopo de projeto bem definido para a equipe de desenvolvimento de software.

Os resultados observados no presente estudo relativo à redução do *lead time* das entregas do método Scrum necessita de estudos mais detalhados, pois não foi possível analisar com maior propriedade as relações de ganho entre o método e as práticas de melhoria.

Apesar da grande flexibilidade do método Scrum ainda é necessário abordar as outras áreas de conhecimento propostas pelo PMI (*Project Management Institute*) para o gerenciamento de projetos, como por exemplo, no gerenciamento de riscos, recursos humanos e aquisições.

O estudo apresenta limitações em termos de generalização, pois seriam necessários mais estudos do tipo de múltiplos casos em empresas de outros ramos de atividade e análises mais aprofundadas na busca do entendimento para evidenciar os principais fatores que influenciam a relação entre o método ágil e as práticas de MC e suas interações.

Os resultados alcançados nesse estudo mostram que a abordagem possui potencial para estudos futuros na identificação de melhorias em vários aspectos, que extrapola inclusive o objetivo na construção de um aplicativo e aponta para ganhos intangíveis para a organização e para o cliente.

Referencias Bibliográficas

BECK K., *Extreme Programming Explained: Embrace Change*. Addison-Wesley Longman Publishing Co., Inc., Boston, USA, 1999.

- BECK, K., et al.,** *Manifesto for agile software development*, <http://agilemanifesto.org/>, 2001.
- BESSANT, J.; CAFFYN, S.; GILBERT, J.; HARDING, R.; WEBB, S.** *Rediscovering continuous improvement*. Technovation, v.14, p.17-29, 1994.
- BESSANT, J.; CAFFYN, S.; GALLAGHER, M.** *An Evolutionary Model of Continuous Improvement Behavior*. Technovation, v.21, n.2, p. 67-77, 2001.
- BESSANT, J.; FRANCIS, D.** *Developing Strategic Continuous Improvement Capability*. International Journal of Operations & Production Management, v. 19 n. 11, p. 1106-1119, 1999.
- BHUIYAN, N.; BAGHEL, A.** *An overview of continuous improvement: from the past to the present*. Management Decision. v. 43, n. 5, p. 761-771, 2005.
- BHUIYAN, N.; BAGHEL, A. WILSON, J.** *A sustainable continuous improvement methodology at an aerospace company*. International Journal of Productivity and Performance Management.v. 55, n. 8, p. 671-687, 2006.
- BRYMAN A.,** *Research methods and organization studies*. London: Unwin Hyman, 283 p. 1989.
- CARVALHO B.V.; MELLO P.H.C.** *Aplicação do método ágil Scrum no desenvolvimento de produtos de software em uma pequena empresa*. Rev da Gestão da Produção, São Carlos, v. 19, n. 3, p. 557-573, 2012.
- COHEN, G.** *Agile Excellence for Product Managers: A guide to creating winning products with agile development teams*. Super Star Press, California, 2010.
- JHA, S.; NOORI, H.; MICHELA, J. L.** *The dynamics of continuous improvement*. International Journal of Quality Science, v. 1, n. 1, p. 19-47, 1996.
- IMAI, M.** *Kaizen: A Estratégia para o Sucesso Competitivo*. São Paulo: IMAM, 7ª Ed., 2011.
- LACERDA S.G.; BARBOSA B.B; RIBEIRO G.V.** *Adoção do CMMI e das metodologias ágeis nas empresas brasileiras*. Revista Avances en Sistemas e Informática, Medellin, Vol8, Nº3, 2011.
- PALMER S.R, FELSING J.M.** *A Practical Guide to Feature-Driven Development*. Prentice Hall, 2002.
- PROJECT MANAGEMENT INSTITUTE – PMI.** *Guia PMBOK: Um guia do conjunto de conhecimentos do gerenciamento de projetos*. Pennsylvania: Project Management Institute, 5th. ed, 2014.
- SADIQ, W.; ORLOWSKA, M.** *Applying a Generic Conceptual Workflow Modeling Technique to Document Workflow In: AUSTRALIAN DOCUMENT COMPUTING SYMPOSIUM*, 2, Melbourne:[s.n.], 1997.
- SCHWABER, K.,** *Agile Project Management with SCRUM*. Microsoft Press: Washington, 2004.
- SHIBA, S.; GRAHAM, A.; WALDEN, D.** *TQM: Quatro Revoluções na Gestão da Qualidade*. Artes Médicas, Porto Alegre, 1997.
- SHIBA, S. ;WALDEN, D.** *Quality Process Improvement Tools and Techniques*, Massachusetts Institute of Technology and Center for Quality of Management 2002.
- STAPLETON J.** *DSDM: Dynamic Systems Development Method*, in: *TOOLS '99: Proceedings of the Technology of OOS*, IEEE Computer Society, Washington, DC, USA, p. 406, 1999.
- SWAMINATHAN, B.; JAIN, K.** *Implementing the Lean Concepts of Continuous Improvement and Flow on an Agile Software Development Project: An Industrial Case Study*. IEEE AGILE India Conference - Bengaluru, India, 2012.
- TOLEDO J.C. et al.,** *Qualidade: Gestão e Métodos*. RJ, LTC Livros Técnicos e Científicos Editora, 2013.
- VLAANDEREN K., JANSEN S., BRINKKEMPER S. JASPERS E.** *The agile requirements refinery: Applying SCRUM principles to software product management*. Information and Software Technology, v. 53 p.58-70, 2011.
- YIN R. K.,** *Case Study Research: Design and Method*. 2ª Edição. London: Sage, 1994.