DOI:10.5748/9788599693131-14CONTECSI/RF-4493

SCRUM: THE CONTRIBUTION OF AGILE METHODOLOGY TO SOFTWARE PROJECT MANAGEMENT

Ramom Scheidegger Santos (Instituto de Pós-graduação e Graduação, Goiás, Brasil) – ramonscheidegger@gmail.com
Roberto Marinho Fernandes (Universidade Federal de Goiás, Goiás, Brasil) – robertomarinhofernandes@gmail.com
Adriano César Santana (Universidade Federal de Goiás, Goiás, Brasil) – adrianosantana@gmail.com

In order to develop quality products and services, resources must be allocated to assist inproject management and activities. Software engineering provides methodologies and practices that show the most appropriate paths tofollow, among them there is Scrum. This methodology, which allows improvements onthe team working management, approaches planning techniques and responsibilities assignment among the professionals involved. The purpose of this article, besides disseminating the Scrum methodology and serving as a basis for future research, is to demonstratepractically the characteristics and aspects of Scrum. This work was developed over studies and exploratory research in printed collections, books, scientific articles, theses and monographs.

Keywords: Scrum. Methodology. Project. Quality. Software Engineering.

1. Introdução

Através da engenharia de software são aplicadas tecnologias inovadoras, que contribuem para o desenvolvimento de sistemas computacionais e constituem produtos com alto nível de qualidade.

A engenharia de software colabora com metodologias, práticas e técnicas que apresentam caminhos adequados para a construção e manutenção de sistemas de softwares. Essas ferramentas são aplicadas em diversas áreas da tecnologia, como em linguagens de programação, em banco de dados, em testes de software, em práticas e processos, dentre outros.

Nos meados da década de 70, ocorreu a crise do software, na qual a demanda por software aumentou consideravelmente, e os profissionais envolvidos com a tecnologia na época tiveram que lidar com grandes e complexos problemas. À época, praticamente não existiam metodologias que monitorassem os processos de desenvolvimento dos sistemas em construção. Havia atrasos nos prazos estabelecidos, os orçamentos não eram cumpridos, os softwares não correspondiam às necessidades dos clientes e não era possível manter a qualidade mediante a gestão de projetos.

Com a crise do software nos anos 70, foram realizados estudos para analisar quais as medidas que poderiam ser adotadas. Percebeu-se a necessidade da criação de novas práticas, técnicas e ferramentas, então surgiu o que atualmente é chamado de engenharia de software. A mesma seria composta por novos métodos e metodologias capazes de manter a garantia da qualidade. Esse novo ramo do conhecimento buscou a quebra do paradigma, pois visava apresentar novas metodologias que levariam ao desenvolvimento adequado de sistemas.

Existem diversos frameworks específicos para o desenvolvimento de software. Os métodos ágeis reduzem os riscos ao fracionar o desenvolvimento em pequenos períodos denominados iteração.

Através do uso adequado das metodologias ágeis, torna-se possível garantir a satisfação do cliente, inserir mudanças tardias no projeto, melhorar o relacionamento entre os stakeholders, e acelerar adaptações referentes às mudanças.

Muitos desenvolvedores sofrem, atualmente, com problemas de comunicação, de ausência de gerenciamento de riscos e de falta de otimização da produtividade, pois não fazem uso de metodologias para auxílio dos projetos. Sendo assim, acabam tendo impactos negativos nos cronogramas e custos previstos no projeto.

Esse trabalho apresentará abordagens explicativas da engenharia de software, com ênfase no Scrum, por esta ser uma metodologia ágil, que busca a qualidade e envolvimento de equipes, e por apresentar resultados eficazes.

Dentre as metodologias ágeis, o Scrum mostra-se eficaz quanto aos resultados no desenvolvimento de software, porém com determinados comprometimentos quanto ao planejamento futuro.

O Scrum é uma metodologia ágil e flexível, que visa especificar um processo de desenvolvimento interativo e incremental em projetos de pequena e grande escala. Essa metodologia apresenta conjuntos de regras e procedimentos direcionados à gestão. Sua

aplicação contribui para a minimização dos riscos, para o acompanhamento da execução do projeto, para o aumento da produtividade e para a real transparência no planejamento e execução.

De acordo com Bissi (2007, p.3) "o Scrum assume-se como uma metodologia extremamente ágil e flexível, que tem por objetivo definir um processo de desenvolvimento interativo e incremental podendo ser aplicado a qualquer produto ou no gerenciamento de qualquer atividade complexa".

2.1. Engenharia de Software

O desenvolvimento de software tornou-se, nos últimos tempos, uma atividade de extrema relevância para os negócios, pois as organizações têm gerado uma grande demanda de sistemas informáticos e de softwares, principalmente para o gerenciamento empresarial e para tomadas de decisões.

Após a criação do computador, surgiram alguns desafios para os profissionais da tecnologia. No início, foram encontradas grandes dificuldades para a evolução do hardware, afinal era necessário diminuir significativamente o custo de processamento e, principalmente, ampliar a capacidade de armazenar dados. Atualmente, as preocupações no meio tecnológico são outras, de modo que os profissionais devem encontrar estratégias de baixo custo que contribuam para a qualificação de seus produtos.

A engenharia de software surgiu em meio à crise de software dos 70. Pode-se afirmar que a engenharia de software engloba todos os aspectos do desenvolvimento, desde a concepção do projeto à manutenção do sistema.

É necessário que as atividades mais complexas sejam devidamente monitoradas para que a construção do software tenha sucesso. Entretanto, essas atividades devem ser realizadas rapidamente, dentro do prazo estabelecido, com garantia de qualidade e a custos reduzidos.

Sob o ponto de vista de Engholm (2010, p.20) "com base na importância cada vez maior do software no dia-a-dia das empresas, devemos nos preocupar com a maneira com que ele agrega valor aos negócios das mesmas, aumentando a produtividade e diminuindo custos".

Para superar as dificuldades encontradas nos projetos de softwares, a engenharia de software fornece boas práticas, metodologias e tecnologias, as quais facilitam o andamento do projeto. Nisso, os profissionais envolvidos poderão planejar, projetar, implementar e avaliar os procedimentos, garantindo que os requisitos impostos pelo cliente sejam devidamente validados.

> De modo geral, podemos dizer que Engenharia de Software poderia ser resumida à utilização de princípios de engenharia para o desenvolvimento de software, ou seja, levantar os requisitos associados, construir modelos para representar a solução a ser desenvolvida, implementar as diversas unidades que irão compor o produto, verificando se tais unidades atendem aos requisitos

> identificados, realizar a integração entre as unidades, também verificando seu funcionamento, até que tenhamos o produto por completo, que deve passar por uma série de verificações (teste funcionais, desempenho e estresse, usabilidade, etc.) para que possamos concluir o desenvolvimento. (Neto, 2010, p.16)

De forma objetiva, a engenharia de software procura disponibilizar, com alto índice de qualidade e baixo custo, diversas tecnologias para apoiar a produção de software.

A engenharia de software também deve focar o negócio da organização para qualificar os softwares produzidos, de modo a manter excelentes resultados e a auxiliar os gestores nas tomadas de decisões estratégicas e operacionais.

2.2. Qualidade de Software

A qualidade é um conceito relativo, pois vários quesitos devem ser levados em consideração. É possível citar conforto, estética, segurança, flexibilidade, custo e vários outros determinantes na satisfação do cliente.

Conforme Falbo (2005, p.2) "Visando melhorar a qualidade dos produtos de software e aumentar a produtividade no processo de desenvolvimento, surgiu a Engenharia de Software".

A qualidade de softwares está relacionada aos requisitos explicitamente declarados pelos clientes, se prioriza sua satisfação para suprir as necessidades dos negócios.

As métricas de qualidade de software surgiram nos anos 70 para colaborar com atividades e com processos de desenvolvimento.

> Com a finalidade de atender a esses objetivos, a área de engenharia de software destina parte de sua atenção ao quesito qualidade na construção de software, utilizando a definição de modelos e processos para melhoria da qualidade e diminuição de custos no desenvolvimento e na manutenção de sistemas. Existem hoje em dia várias propostas de modelo buscando melhorar o processo de desenvolvimento de software e a qualidade envolvida. (Engholm, 2010, p. 20)

São vários os aspectos que devem ser analisados mediante a qualidade do software. Sob o ponto de vista do usuário, o software produzido deve envolver características vinculadas à usabilidade, ao custo, à confiabilidade e à portabilidade. Esses vínculos agregam valor ao negócio quando trazem "feedbacks" positivos da parte de seus clientes. Entretanto, da parte da equipe de desenvolvimento, observa-se a qualidade do produto sob ângulo mais técnico e operacional. Frisam-se aspectos favoráveis à programação e às perspectivas internas do projeto em si.

> O que há de comum nas várias perspectivas discutidas acima é que

> todas elas estão focadas no produto de software. Ou seja, estamos falando de qualidade do produto. Entretanto, como garantir que o produto final de software apresenta essas características? Apenas avaliar se o produto final as apresenta é uma abordagem indesejável para o pessoal de desenvolvimento de software, tendo em vista que a constatação a posteriori de que o software não apresenta a qualidade desejada pode implicar na necessidade de refazer grande parte do trabalho. É necessário, pois, que a qualidade seja incorporada ao produto ao longo de seu processo de desenvolvimento. De fato, a qualidade dos produtos de software depende fortemente da qualidade dos processos usados para desenvolvê-los e mantê-los. (Falbo, 2005, p.3)

Para que todo o processo de desenvolvimento de software seja realizado com sucesso, a equipe deve dispor de mecanismos que auxiliem no planejamento, no monitoramento, no controle e na avaliação do produto final. Deve-se analisar quais são as reais necessidades dos clientes e então executar, com métodos e técnicas que garantam o atendimento dos requisitos levantados, os processos que possam supri-las.

É importante que sejam definidos e priorizados os aspectos que qualificam positivamente o projeto em questão. De acordo com Magalhães (2013, p. 3) "Pela dificuldade de se atender diversas características de qualidade simultaneamente, torna-se necessário priorizar aquelas que irão determinar a qualidade do produto, o que faz parte do planejamento da qualidade".

2.3. Utilização dos Modelos de Processos Tradicionais de Software

Sob a linha de pensamento de Pressman (2011, p.40) "Processo é um conjunto de atividades, ações e tarefas realizadas na criação de algum produto de trabalho".

Os modelos de processos de softwares fornecem aos desenvolvedores diretrizes no projeto e através de técnicas e métodos, apresentam formas de organização no decorrer da produção do software. Basicamente, os modelos de processos de software descrevem as fases do produto desde sua concepção até a manutenção.

Segundo Falbo (2005) um processo de software pode ser considerado um conjunto de atividades, métodos, práticas e transformações que guiam pessoas na produção de software de qualidade.

2.3.1 Modelo Cascata

O modelo-cascata, também denominado clássico, linear ou tradicional, é composto pelas etapas levantamento de requisitos, análise de requisitos, projetos, implementação, teste, implantação e, posteriormente, manutenção de software.

> A principal desvantagem do modelo-cascata é que boa parte do

> sistema não estará disponível até um ponto adiantado no cronograma do projeto e, geralmente, é difícil convencer o usuário de que é preciso paciência. Além disso, existe a dificuldade de acomodação das mudanças depois que o processo está em andamento. Portanto, esse modelo é mais apropriado quando os requisitos são bem entendidos. No entanto, há também algumas vantagens associadas ao modelo, pois ele oferece uma maneira de tornar o processo mais visível, fixando pontos específicos para a escrita de relatórios e, didaticamente, é uma maneira mais fácil de introduzir os principais conceitos de Engenharia de Software. (Vasconcelos,Rouiller, Machado, &Medeiros, 2006, p.28)

As fases do modelo seguem uma sequência rígida e sistemática, pois requer-se o final da fase anterior para o início de uma nova etapa. O uso adequado desse processo demanda requisitos minuciosamente definidos.

2.3.2 O Modelo de Desenvolvimento Baseado em Reuso

O modelo de desenvolvimento baseado em reuso fundamenta-se na existência de um número significativo de componentes de software reutilizáveis. O reuso viabiliza a redução na quantidade de software a ser desenvolvido, nos custos e nos riscos. Ajuda também na entrega dentro do prazo estabelecido.

Geralmente, o uso desse modelo é informal, mas intenso. A orientação a reuso facilita a codificação no projeto, pois os programadores copiam os códigos e recursos de outros sistemas em seus próprios projetos.

2.3.3 O Modelo Espiral

Cada fase deste modelo é concluída em uma volta no espiral que representa o ciclo. A espiral é dividida em quatro setores: definição de objetivos, avaliação e redução de riscos, desenvolvimento e validação, planejamento.

No modelo espiral não existem etapas fixas. Os riscos são analisados detalhadamente e os problemas encontrados são solucionados no decorrer do processo.

2.3.4 O Modelo Incremental

O modelo incremental é utilizado em projetos de software que busquem um rápido fornecimento de funcionalidades aos seus clientes. Incrementam melhorias e correções em paralelo à adição de funcionalidades.

Inicia na criação de uma versão básica do sistema seguida pela implementação de funcionalidades adicionais, que envolve o reprojeto e listas de gerenciamento do projeto.

O desenvolvimento incrementado é um clássico que surgiu para superar as dificuldades do modelo-cascata.

2.3.5 Prototipação

A prototipação é um modelo que auxilia na melhor visualização dos softwares e suas funcionalidades por apresentar de forma prática a solução aos requisitos. O protótipo permite compreender o propósito do software, maximiza a probabilidade de lucros e reduz significativamente os riscos negativos.

A prototipação contribui intensamente para o desenvolvimento do software, pois o alinhamento entre a equipe e o cliente é garantido em diferentes níveis de fidelidade prática.

2.4. Desenvolvimento Ágil de Software

As metodologias ágeis de software podem ser definidas como o uso de conceitos das metodologias tradicionais voltados à velocidade na entrega do produto. Tentam garantir a satisfação do cliente e também que mudanças tardias não prejudiquem o projeto. São marcadas pelo grande envolvimento da equipe, pela simplicidade e pela facilidade de adaptação às mudanças.

Segundo Pressman (2011, p.82) "atualmente, agilidade tornou-se a palavra da moda quando se descreve um moderno processo de software".

> A engenharia de software ágil combina filosofia com um conjunto de princípios de desenvolvimento. A filosofia defende a satisfação do cliente e a entrega de incremental prévio; equipes de projeto pequenas e altamente motivadas; métodos informais, artefato de engenharia de software mínimos e, acima de tudo, simplicidade do desenvolvimento geral. Os princípios de desenvolvimento priorizam a entrega mais que a análise e projeto (embora essas atividades não sejam desencorajadas); também priorizam a comunicação ativa e contínua entre desenvolvedores e clientes. (Pressman, 2011, p.81)

A engenharia de software ágil preza a comunicação, o planejamento, a modelagem, a construção e a auto-organização. Requer dos profissionais espírito de equipe, competência multifacetada e capacidade de tomada de decisões.

Existem variadas metodologias de desenvolvimento ágil de software, entre elas:

- Extreme Programming (XP) possui cinco valores que estabelecem as bases para todos os trabalhos realizados. São eles a comunicação, a simplicidade, o feedback, a coragem e o respeito. Realiza uma abordagem orientada a objetos e exige quatro atividades metodológicas: planejamento; projeto; codificação; testes.

- Desenvolvimento de Software Adaptativo (ASD) é utilizado para construção de softwares e sistemas complexos. Possui fases de especulação, de colaboração e de aprendizagem.

- Método de Desenvolvimento de Sistemas Dinâmicos (DSDM) é um processo de software iterativo no qual somente o trabalho suficiente é requisitado no incremento.

- Crystal foi criado para priorizar a adaptabilidade. Seu objetivo principal é a seleção, no conjunto de exemplos fornecidos pela metodologia, do membro da família Crystal mais apropriado ao projeto e seu ambiente.

- Desenvolvimento Dirigido a Funcionalidade (FDD) é um modelo de processo prático para a engenharia de software orientada a objetos. Tal modelo descreve um processo ágil adaptativo.

- Desenvolvimento de Software Enxuto (LSD) é resumido em: eliminar desperdícios; incorporar qualidade; criar conhecimento; adiar compromissos; entregar rápido; respeitar as pessoas; otimizar o todo.

- Modelagem Ágil adota todos os valores consistentes com manifesto ágil. São muitos os princípios da AM, porém os que os tornam única são: modele com um objetivo; use modelos múltiplos; viajar leve; conteúdo é mais importante que a representação; tenha conhecimento; domínio dos modelos e das ferramentas que for utilizar; adapte localmente.

- Processo Unificado Ágil adota fases clássicas PU chamadas início, elaboração, construção e transição. Cada iteração PUA é composta pelas atividades de modelagem, de implementação, de aplicação, de configuração e de gerenciamento do projeto.

2.5. Scrum

Com o nome inspirado em uma jogada do Rúgbi, o Scrum é uma metodologia ágil de software. Nas palavras de Schwaber& Sutherland (2013, p.3) “ Um framework onde pessoas podem tratar e resolver problemas complexos e adaptativos, enquanto produtiva e criativamente entregam produtos com o mais alto valor possível”.

> Scrum aplica-se a projetos tanto pequenos como grandes. Esforçando-se para liberar o processo de quaisquer barreiras, o seu principal objetivo é conseguir uma avaliação correta do ambiente em

> evolução, adaptando-se constantemente ao "caos" de interesses e necessidades, indicado e utilizado para o desenvolvimento de softwares em ambientes complexos, onde os requisitos mudam com certa frequência, sendo o caminho utilizado para aumentar produtividade nesses tipos de sistemas. (Bissi, 2007, p.3)

O conceito Scrum foi formalizado em 1994 por Jeff Sutherland e Ken Schwaber que decidiram implantar o framework na própria empresa. Segundo Lacerda &Sganderla (2010, p.5) "não é um processo prescritivo e não descreve o que deve ser feito em cada uma das situações. Scrum é usado para trabalhos complexos, onde é impossível predizer tudo que irá acontecer".

> Scrum é uma metodologia baseada nas melhores práticas da indústria, usadas e comprovadas por décadas e em crescimento constante nos ambientes de desenvolvimento de software. Devido a sua característica empírica, adaptativa e inovadora é usada no desenvolvimento de sistemas de modo incremental, onde os requisitos sofrem constantes mudanças durante o ciclo de vida do produto. (Leitão, 2010, p.24)

2.5.1 Funções no Scrum

A metodologia Scrum define três funções (papéis) para membros do projeto:

- Dono do Produto é o lider do projeto. Especifica e prioriza, junto ao cliente, requisitos de negócio do produto. É o responsável por gerenciar o Backlog do Produto.
- Mestre Scrum é o responsável pela aplicação do Scrum. Deve buscar meios de facilitar e acelerar o trabalho da equipe, além de garantir que a metodologia seja compreendida por todos envolvidos no desenvolvimento. É também o coordenador do Scrum Diário e demais eventos.
- Equipe de Desenvolvimento é composta pelos envolvidos na execução. Deve variar entre 5 e 9 pessoas. Seus membros devem se autoorganizar e buscar a melhor maneira de atingir as metas definidas em cada Sprint. Definem, por votação, o tamanho relativo de cada elemento do Backlog. O conjunto de habilidades dos membros deve ser suficiente para atender os requisitos do produto.

> O Time Scrum é composto pelo ProductOwner, o Time de Desenvolvimento e o Scrum Master. Times Scrum são auto-organizáveis e multifuncionais. Times auto-organizáveis escolhem qual a melhor forma para completarem seu trabalho, em vez de serem dirigidos por outros de fora do Time. Times multifuncionais possuem todas as competências necessárias para completar o

> trabalho sem depender de outros que não fazem parte da equipe. O modelo de time no Scrum é projetado para aperfeiçoar a flexibilidade, criatividade e produtividade. Times Scrum entregam produtos de forma iterativa e incremental, maximizando as oportunidades de realimentação. Entregas incrementais de produto "Pronto" garantem que uma versão potencialmente funcional do produto do trabalho esteja sempre disponível. (Schwaber& Sutherland, 2013, p.5)

2.5.2 Backlog do Produto, Backlog do Sprint e Incremento

O Backlog do Produto é uma lista que reúne todos as funcionalidades demandas no produto. Sua criação e gestão cabem ao Dono do Produto. Cada elemento, ou requisito, do Backlog precisa, dentre outras características, de uma Definição de Feito, segundo a qual o produto será testado.

Cada elemento do Backlog possui uma estória associada. Essa estória deve descrever o ganho de valor para o usuário. Cabe ao Dono do Produto os elementos com as estórias de maior valor ao topo do Backlog.

A dimensão do esforço necessário para executar cada item é expressa em Pontos de Estória (Story Points). É uma maneira de relativizar o tamanho das atividades entre si sem usar estimativas de hora. O Pôquer do Planejamento (Planning Poker) é utilizado pela Equipe de Desenvolvimento para dar um valor em pontos para cada elemento. Isso pode ser feito durante a reunião de planejamento do Sprint ou, o que é mais recomendado, em uma reunião exclusiva para adequação do Backlog.

> O Product Owner é o responsável pelo Backlog do Produto, por seu conteúdo, por sua disponibilidade e por sua priorização. O Backlog do Produto nunca está completo. A seleção inicial para o seu desenvolvimento somente mostra os requisitos inicialmente conhecidos e melhor entendidos. O Backlog do Produto evolui à medida que o produto e o ambiente em que ele será usado evoluem. O Backlog é dinâmico, no sentido de que ele está constantemente mudando para identificar o que o produto necessita para ser apropriado, competitivo e útil. Enquanto existir um produto, o Backlog de Produto também existirá. (Schwaber&Shuterland; 2013; p.16)

No Planejamento do Sprint são escolhidos quais elementos do Backlog serão trabalhados no Sprint. A esse conjunto se dá o nome de Sprint Backlog. Uma das metas da equipe é desenvolver todos os requisitos selecionados no prazo do Sprint. Incremento é a soma de todas as funcionalidades entregues pelos vários Sprints já ocorridos.

Segundo Kniberg (2007, p. 16) é preciso ser prático, utilizando cartões por exemplo, na formulação do Backlog, pois ele precisa ser acessível e, principalmente, bem

compreendido por todos.

2.5.3 Sprint e Scrum Diário

O Sprint é um evento do Scrum do tipo TimeBox (Caixa de Tempo), cuja característica é sempre ter a mesma duração. Tal consistência permite gerenciar a execução das tarefas a fim de evitar o desperdício de tempo.

> "O tempo é o limite insuperável do empreendimento humano, afetando tudo, desde o modo como trabalhamos, passando pelo tempo que as coisas levam, até definir se somos bem-sucedidos ou não. O fluxo impiedoso do tempo define fundamentalmente o modo como enxergamos o mundo e a nós mesmos. Conforme o famoso verso do poeta inglês do século 17, Andrew Marvell, "Se nós tivéssemos mais tempo e mundo", qualquer coisa poderia ser feita. É claro, porém, que um senso de mortalidade paira sobre todos os nossos esforços. Sabemos que nosso tempo é limitado. " (Sutherland, 2014, p.61)

Sprint é antecedido pela reunião de planejamento na qual serão escolhidos os elementos do Sprint Backlog. É pela negociação entre o Dono do Produto e a equipe que se determina quantos itens serão colocados no Sprint, se seu escopo será reduzido, se as definições de feito são apropriadas e se a funcionalidade buscada é bem entendida por todos.

Utiliza-se o Quadro Scrum (ScrumBoard) para organizar os cartões. É um quadro com 3 colunas: "A fazer", "Fazendo" e "Pronto". No início, cada elemento do Sprint Backlog é alocado em um cartão individual na coluna "A fazer". Os membros deslocam para "Fazendo" conforme cada um assume um cartão por vez.

Conforme Sutherland (2014, p.64) "um ponto importante: nenhuma tarefa vai para a coluna "Pronto", a não ser que possa ser cumprida pelo cliente. Em outras palavras: você pode dirigir o carro. E se alguém dirige o carro e diz "ei, a seta está agarrando", então, esse problema será resolvido no próximo Sprint". Conforme o Sprint é executado e itens entram no "Pronto", é possível traçar o Gráfico de Burndown. Esse gráfico represento Incremento e projeta quando o mesmo será zero.

O Scrum Diário é uma reunião, facilitada pelo Mestre Scrum, cujo objetivo é manter o Sprint em bom andamento. Sugerem-se três perguntas a cada membro relacionadas ao andamento dos trabalhos. A intenção do Mestre Scrum é promover a interação e o trabalho em equipe como solução das dificuldades. O Mestre Scrum também procura perceber entraves que estejam diminuindo a velocidade da equipe. Um detalhe curioso da reunião é que se deve fazê-la de pé, sem formalidades, com duração máxima de quinze minutos.

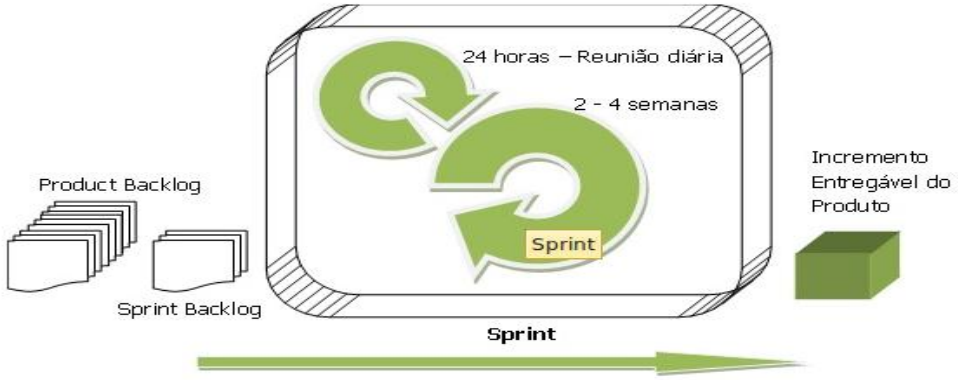

Figura 1: Diagrama do Sprint

A Figura 1 representa o andamento do Sprint do Backlog à entrega do incremento. Há algumas restrições após o início do Sprint:

- As mudanças propostas não podem ser realizadas caso coloquem em perigo o objetivo da Sprint;
- Os objetivos da qualidade não podem diminuir;
- O escopo deve ser renegociado entre o Dono do Produto e o Time de Desenvolvimento;

> Sprints são limitadas a um mês corrido. Quando o horizonte da Sprint é muito longo, a definição do que será construído pode mudar, a complexidade pode aumentar e o risco pode crescer. Sprints permitem previsibilidade que garante a inspeção e adaptação do progresso em direção à meta pelo menos a cada mês corrido. Sprints também limitam o risco ao custo de um mês corrido. (Schwaber&Sutherland, 2013, p.8)

O cancelamento da Sprint apenas poderá ser autorizado pelo Dono do Produto, mesmo que seja influenciado pela Equipe de Desenvolvimento ou pelo Mestre Scrum.

2.5.4 Revisão e Retrospectiva do Sprint

A Revisão do Sprint é uma reunião com ares de apresentação, dimensionada para 4h em Sprints de 30 dias, cujo objetivo é rever e adequar o Backlog do Produto, isso segundo a avaliação do produto pelos Stakeholders presentes.

A perspectiva mais importante é a direção a tomar no desenvolvimento do produto. Discute-se desde possibilidades de mercado à modificação, adição e viabilidade financeira de novas funcionalidades. A Revisão do Sprint é o subsídio principal para o planejamento do próximo Sprint.

A Retrospectiva do Sprint é uma reunião cujo cerne é o processo de desenvolvimento. Os membros do Time Scrum focam em como tornar o processo mais

eficiente e agradável, em como potencializar os acertos e em como mitigar as dificuldades. São abordadas questões de relacionamento, processos e tecnologias. São buscados meios de melhorar a qualidade do produto e suas definições de "Pronto"

2.6 O Scrum Aplicado no Gerenciamento de Projetos de Software

Scrum é a metodologia ágil de maior adoção entre as empresas de software. Projetos preferem as metodologias ágeis por sua flexibilidade e orientação à entrega. As metodologias ágeis são adaptativas, logo as mudanças tardias têm menor impacto no projeto. Além disso, de acordo com Paula (2012, p.10), "a abordagem tradicional é mais adequada para projetos que necessitam de um forte planejamento e muita disciplina no processo, mas ela não promove tão ampla comunicação entre os membros de equipes e seus gerentes".

O framework traz vantagens e desvantagens ao projeto:

- Sua implementação requer mudanças políticas e culturais na empresa;
- Abre mão de muitas das ferramentas de engenharia de software já adotadas na empresa;
- A documentação do software pode ser colocada em segundo plano;
- Melhora a comunicação entre a equipe;
- Antecipa a entrega de funcionalidades ao cliente;

As práticas do Scrum podem ser aplicadas em projetos de diversas áreas de atividades, nos quais as pessoas possuem necessidade de trabalhar em união para buscar atingir um objetivo comum. O Scrum tem maior índice de aproveitamento em projetos voltados para software, projetos automotivos, telecomunicações e projetos baseados em pesquisa e inovação.

3 Resultados

3.1 Comparação prática entre Gerenciamento de Software com Scrum e Gerenciamento de Software com Modelo Cascata

Há diversas diferenças práticas quando se utiliza Scrum ao invés do modelo clássico. Tantas são as diferenças que adotamos os conceitos básicos de engenharia de software para compará-las. É importante definir como cada metodologia lida com as cinco atividades do Processo de Software descritas por Pressman (2011, p. 40):

- Comunicação
- Planejamento
- Modelagem
- Construção
- Emprego

Requer-se analisar primeiro o funcionamento do Scrum e, em seguida, a abordagem clássica compatível em cada tópico.

3.2 Comparativo em Comunicação

No Scrum a comunicação com interessados fica a cargo do Dono do Projeto (Project Owner). É seu papel transpor as necessidades dos stakeholders em itens do Backlog.

Como toda metodologia ágil, a documentação típica da Engenharia de Requisitos é deixada em último plano. Fica a cargo do Dono do Projeto transpor toda a visão em elementos que caracterizem as necessidades do cliente. Trata-se de necessidades majoritariamente de negócios (não técnicas).

Além de número e um título, cada tópico no Backlog terá uma estimativa relativizada de tamanho (em relação aos outros elementos do Backlog). O Dono do Produto deve ordenar a lista de elementos sob prioridade.

Todo elemento do Backlog tem uma estória associada ao mesmo. Deve-se definir o ator, o que deseja e como conferir se obteve o que deseja. Pode-se traçar uma analogia entre as estórias (stories) do Backlog e os Casos de Uso da Modelagem de Requisitos. Ressalta-se que as estórias tendem a ser mínimas quando comparadas aos casos de uso.

Ao passo que simplifica e traduz as necessidades do cliente no Backlog do Produto, o Dono do Produto definirá junto ao cliente eventuais mudanças de requisitos, de orçamento e de cronograma.

Comparando com o Modelo em Cascata, percebe-se que a definição de requisitos em Scrum é menos detalhada. Essa flexibilização é viável devido, primeiramente, às apresentações ao cliente ao fim de cada Sprint e, em segundo, à adoção do paradigma incremental. Uma vez que o cliente irá avaliar e validar o software ao fim de cada Sprint, não há necessidade de firmar os requisitos minuciosamente, mesmo por que o cliente tende a adicionar ou modificar requisitos constantemente.

Ainda na atividade de Comunicação, o Scrum define que o contato com os interessados é exclusivo do Dono do Produto. Da mesma forma, a Equipe Scrum é subordinada exclusivamente ao Dono do Produto. Isso implica em uma centralização das informações e decisões estratégicas do projeto, o que provê agilidade e coesão na criação do produto. Já no modelo tradicional, a comunicação tende a seguir aspectos da hierarquia funcional, como acontece em reuniões diretas do stakeholder com a equipe de desenvolvimento, por exemplo.

3.3 Comparativo em Planejamento

Uma das premissas fundamentais do Scrum é que detalhar cronogramas com longos horizontes de tempo é um erro. Ao invés de traçar diagramas de Gantt, a metodologia busca estimar o término do projeto através da medição de velocidade nos

Sprints. Com a velocidade medida, pode-se montar o Gráfico de Burndown do Backlog com o qual projeta-se a entrega final.

Na reunião de planejamento do Sprint, da qual participam todos os envolvidos, define-se quais itens do Backlog entram no Sprint. Cabe ao Dono do Produto negociar com a equipe o total de pontos de tamanho relativo no Sprint. Deve-se definir também um objetivo geral do
Sprint, assim como a data de apresentação do produto (na Revisão do Sprint). Ao fim do Sprint, soma-se o que foi entregue para ter a velocidade em pontos/Sprint.

O Scrum não inicia com a formulação de um cronograma, existe apenas a meta do prazo final de conclusão do projeto. Não são definidas datas para entregas de outras etapas além do Sprint imediato. É evidente também que não se usam horas como controle de ritmo do trabalho, mas pontos de tamanho relativo. A referência temporal é apenas a projeção final do Burndown do Backlog ante o prazo final. Inclusive, só é possível traçar a projeção do Burndown do Backlog após o fim do primeiro Sprint.

Diferentemente da metodologia em cascata, na qual se analisam as atividades individuais, o alvo da metodologia é aumentar a velocidade da equipe a cada Sprint. Cabe ao Mestre Scrum eliminar os obstáculos ao trabalho de cada membro da equipe.

O desapego a cronogramas e gráficos de Gantt é defendido por Shuterland com a apresentação do Cone da Incerteza: quanto mais distante a data, maior o erro da estimativa.

Outra diferença em relação ao Modelo Cascata é que não são controladas as atividades dos membros da equipe. Entende-se que o Scrum Diário, sob a facilitação do Mestre Scrum, alocará os membros da equipe na forma mais eficiente de executar o trabalho.

Sob o aspecto de custos, o Scrum não prescreve nenhum método. Fica a cargo do Dono do Produto utilizar o método que bem entender para gerenciar as finanças junto aos clientes.

3.4 Comparativo em Modelagem

O Scrum surgiu no ambiente do desenvolvimento de software, mas é uma metodologia que não se limita a esse contexto. Não há ferramentas exclusivas de modelagem de software atribuídas à metodologia. O único elemento relacionado à modelagem de software em Scrum é a definição de feito constante na estória de cada elemento do Backlog.

É importante frisar que, devido a seu caráter incremental com entregas em ciclos curtos, modelos não-técnicas acabam prescindidos. Ao invés de modelar diversas opções de interface, por exemplo, é preferido apresentar uma opção plenamente funcional e receber o feedback do cliente ao fim do Sprint. Por esses motivos, a Modelagem em Scrum confunde-se com a etapa de Construção. Os custos advindos de um mal julgamento são reduzidos nessa proposta.

O mesmo não é viável no Método em Cascata, pois somente uma entrega final será feita ao cliente. Os custos de corrigir um modelo são muito inferiores ao custo de corrigir no produto final, tornando-os imprescindíveis na Cascata.

3.5 Comparativo em Construção

A construção ocorre enquanto se executa o Sprint. Uma vez que o mesmo está definido, as tarefas são discutidas e distribuídas no Scrum Diário. Os membros da equipe têm autonomia para buscar a melhor maneira de executar as atividades.

É uma parte importante da filosofia Scrum "fazer certo da primeira vez". Também é defendido que os produtos precisam ser modulares, para facilitar entregas incrementais. Culmina que a construção no Scrum, majoritariamente iterativa, conta com o feedback do cliente como principal controle de qualidade.

Restrições, casos e elementos não previstos na Comunicação, Planejamento ou Modelagem podem ser inseridos como elemento, por qualquer membro da equipe, dentro do próprio Sprint como um novo elemento. Caso não seja possível a execução no Sprint atual, deve-se discutir com o Dono do Produto sua adição no Backlog de forma adequada.

Comparando com o Modelo em Cascata, percebe-se que a principal diferença está na flexibilidade da construção. Ao invés de executar diretamente os minuciosos planos, modelos e requisitos, parte-se do princípio de construir o proposto a cada Sprint. É preciso apresentar algo que funcione ao cliente o mais rápido possível

3.6 Comparativo em Emprego

Enquanto na metodologia clássica o emprego marca o fim do contrato e a expectativa de um produto completamente funcional, na metodologia Scrum o emprego marca o fim do Sprint. Durante desenvolvimento sob o Scrum, espera-se entregar por incrementos diversas vezes durante o projeto. No Cascata a entrega é apenas a final.

3.7 Adaptabilidade do Scrum

Com o intuito de aproveitar a agilidade do Scrum, alguns autores associaram-no com outras metodologias de Gerenciamento de Projetos para Engenharia de Software. Leal (2008) analisou a associação das metodologias PMBOK e Scrum sob o framework Agilius. Para tal, estudou o Eclipse Process Framework Composer como ferramenta de trabalho.

Para atender aos padrões do CMMI, Marçal (2009) propôs a metodologia SCRUMMI, uma adaptação do Scrum, e a aplicou ao desenvolvimento de um software sob os parâmetros da certificação.

4. Conclusões e Recomendações

A engenharia de software fornece artefatos e técnicas consolidadas para o desenvolvimento de software, os ditos modelos tradicionais. Há também metodologias ágeis capazes de antecipar resultados aos stakeholders. Ambas vertentes têm ciclos de vida e desenvolvimento bem definidos que maximizam as possibilidades de obter-se resultados positivos.

O Scrum é um framework ágil que se diferencia pela centralização de requisitos e especificações, no papel Dono do Produto, pela busca dos ganhos de velocidade na equipe, pelo Mestre Scrum, e pela diversidade na equipe de desenvolvimento.

As comparações desenvolvidas nesse trabalho fundamentam a percepção de que o Scrum é uma ferramenta versátil. Enquanto permite prescindir de longos planejamentos, essa metodologia é capaz de reagir bem às mudanças de requisitos e de ser adaptada a padrões consagrados.

Adotar Scrum em ambientes de produção de software acrescenta poucos riscos ao projeto, pois seu uso não implica no abandono de padrão. Da mesma forma, por sua estrutura enxuta e sinergia com padrões estabelecidos, Scrum demanda pequenos investimentos de tempo e recursos financeiros.

Através das pesquisas realizadas neste trabalho, percebeu-se que o Scrum, quando comparado ao modelo em cascata, faz uso do feedback do cliente, não do planejamento minucioso, como controle de qualidade.

Por ser agnóstico em termos técnicos, o Scrum pode ser adotado em conjunto com diversas técnicas tradicionais de software, tornando-se uma ferramenta poderosa mesmo para grandes projetos e trabalhos sob sistemas certificados, como CMMI e MPS.br.

Percebe-se que a tendência é o uso de metodologias híbridas adaptadas segundo as necessidades de gestão do projeto. O Scrum se posiciona então como uma plataforma que pode receber incrementos e abarcar desde projetos de sites simplificados a sistemas governamentais certificados.

Diante das pressões por parte dos clientes que demandam entregas cada vez mais rápidas, é de grande valia a adoção do Scrum como uma ferramenta adicional de gestão de projetos e equipe. Tal recomendação é complementada pela manutenção dos padrões e certificações com os quais a equipe já trabalha, assim como a manutenção de seus processos de qualidade e teste.

A customização do Scrum durante a implantação é a opção que apresenta melhor benefício. Deve-se considerar que Scrum não possui ferramentas específicas de qualidade, mas é uma base que deve ser incrementada com outras técnicas para atender as necessidades da equipe.

Referências

BISSI,Wilson. Scrum - Metodologia de Desenvolvimento Ágil. Maringá, 2007.

ENGHOLM , Hélio. **Engenharia de Software na Prática**. Editora: Novatec. São Paulo, 2010.

FALBO, Ricardo de Almeida. **Engenharia de Software**. Espírito Santo. 2005.

KNIBERG, Henrik. **ScrumFromtheTrenches**. Ed. 2. Editora: C4Media. 2007

LEAL, Luciana de Queiroz. **Uma Abordagem Ágil ao Gerenciamento de Projetos de Software Baseada no PMBOK Guide**. Recife, 2008

LEITÃO, Michele de Vasconcelos. **Aplicação de Scrum em Ambiente de Desenvolvimento de Software Educativo.** Recife, 2010.

MARÇAL, A. S. C.**SCRUMMI: Um Processo de Gestão Ágil Baseado no Scrum e Aderente ao CMMI.** Fortaleza, 2009

NETO. Pedro de Alcântara dos Santos. **Introdução à Engenharia de Software**. Piauí, 2008.

PAULA, Renata de Souza Alves. **Scrum na Melhoria do Gerenciamento de Projetos de Software**. DevMedia Artigo Engenharia de Software 23. São Paulo, 2012.

PRESSMAN, Roger. **Engenharia de Software: Uma abordagem profissional**. Ed.7. Editora: bookman. New York, 2011.

SCHWABER, Ken. SUTHERLAND, Jeff. **Guia do Scrum**. Michigan, 2013.

SUTHERLAND, Jeff. **Scrum – A Arte de Fazer o Dobro de Trabalho na Metade do Tempo**. LeYa. São Paulo. 2014

LACERDA, Guilherme Silva, SGANDERLA, Mauricio Andreazza.**Melhorando a Gerência e a Construção de Software com Metodologias Ágeis**. Porto Alegre, 2010.

SOMMERVILLE, Ian. **Engenharia de Software**. Ed.9. São Paulo: Editora Pearson Prentice Hall, 2011.

VASCONCELOS, Alexandre Marcos Lins de. ROUILLER, Ana Cristina. MACHADO, Cristina Ângela Filipak. MEDEIROS, Teresa Maria Maciel de. **Introdução à Engenharia**

de Software e à Qualidade de Software. Lavras. 2006.