UNIVERSIDADE FEDERAL DO PARANÁ

JOSEPH BAUER DE OLIVEIRA

LISTA DE VERIFICAÇÃO SCRUM PARA DESENVOLVIMENTO DE SOFTWARE

CURITIBA

2012

JOSEPH BAUER DE OLIVEIRA

LISTA DE VERIFICAÇÃO SCRUM PARA DESENVOLVIMENTO DE SOFTWARE

Monografia apresentada ao Curso de Pós-Graduação em Informática, Departamento de Informática, Setor de Ciências Exatas, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Especialização em Informática, ênfase em Tecnologia da Informação.

Orientador: Professor Msc. Nelson Suga.

CURITIBA

2012

PARECER DE APROVAÇÃO

MONOGRAFIA DE ESPECIALIZAÇÃO EM INFORMÁTICA

ÊNFASE EM TECNOLOGIA DA INFORMAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA/UFPR

Declaramos que o aluno Joseph Bauer de Oliveira entregou a versão final da sua Monografia de Especialização em Informática da Universidade Federal do Paraná, com Ênfase em Tecnologia da Informação, intitulada Lista de Verificação Scrum para Desenvolvimento de Software.

Curitiba, 03 de maio de 2012.

Prof. Msc. NELSON SUGA
Professor Adjunto
Universidade Federal do Paraná
Setor de Ciências Exatas
Departamento de Informática
Caixa Postal 19081
CEP 81531-990 - Curitiba-PR

Prof. Msc. SETEMBRINO SOARES FERREIRA JUNIOR
Professor Assistente
Universidade Federal do Paraná
Setor de Ciências Exatas
Departamento de Informática
Caixa Postal 19081
CEP 81531-990 - Curitiba-PR

AGRADECIMENTOS

Agradeço a meus pais, pelos sólidos princípios a mim transmitidos, e por sua dedicação infindável.

Ao professor e orientador Nelson Suga, pela incrível disposição em ajudar pessoas a construir e disseminar novos conhecimentos.

RESUMO

A rápida adequação às mudanças de requisitos de *software* garante às empresas diferenciais competitivos e tecnológicos. Estas mudanças vem ao encontro das reais necessidades dos clientes destas empresas. Scrum é uma metodologia para desenvolvimento ágil de *software* criada para lidar facilmente com estas características citadas. Baseada no Manifesto Ágil para Desenvolvimento de Software e nas constantes reuniões presentes nas suas práticas gerenciais, como *Daily Meetings, Sprint Review* e *Retrospective*, Scrum permite avaliar o real estágio de andamento de um projeto, garantindo assim informações confiáveis ao time e às partes interessadas da companhia. Esta monografia tem como objetivo apresentar uma lista de verificação para a condução de um projeto de desenvolvimento de *software*, ou parte dele, que utilize a metodologia ágil Scrum. Esta lista pode ser utilizada por gerentes de projetos ou times de desenvolvimento para avaliar o estado do Scrum nas equipes. Deve ser empregada em todos os momentos do projeto, pois acompanha as fases previstas no Scrum. Seus itens tratam aspectos como a composição do time, locais definidos para as reuniões e intervalos entre as fases de desenvolvimento. Neste trabalho analisamos os pontos presentes na metodologia ágil Scrum e formas para verificar se são atendidos. Como resultado desta análise é moldada a lista de verificação e seus meios de utilização.

Palavras-chaves: Scrum. Metodologia ágil. Manifesto ágil. Lista de Verificação.

ABSTRACT

The quick adjustment to software requirements changes ensures to enterprises technological and competitive advantages. These changes come to attend the real needs of the company's customers. Scrum is an agile software development methodology created to easily deal with these mentioned characteristics. Based on Agile Manifesto for Software Development and in the constant meetings available on their management practices, like Daily Meetings, Sprint Review and Retrospective, Scrum allows to evaluate the real state of a project development, assuring reliable information to the team and the company's stakeholders. This monograph aims show a checklist to conduce a software development project, or part of it, that uses the Scrum agile methodology. This list can be used by project managers or development teams, to evaluate the state of Scrum in the teams. Must be applied at all project moments, because it follows the stages set out in Scrum. Their items deal with aspects such as the team composition, defined locations for meetings and intervals between stages of development. In this paper we have analysed the points available in the Scrum agile methodology and ways to check if they are achieved. As a result of this analisis it's shaped the checklist and their ways of use.

Key words: Scrum, Agile methodology. Agile Manifesto. Checklist.

LISTA DE QUADROS

Sumário

1 INTRODUÇÃO

Prazos cada vez menores e demanda crescente por *software* fazem com que as empresas continuamente clamem por metodologias que contemplem suas variáveis de tempo, custo e qualidade. O excesso de formalidade imposto por metodologias tradicionais para desenvolvimento de *software* por muitas vezes acarreta no descumprimento destas variáveis.

O desenvolvimento iterativo é a grande chave das metodologias ágeis, assim como o Scrum, pois determina *software* funcionando o mais cedo possível e com frequência.

A contínua interação e sincronia entre clientes e desenvolvedores faz com que as mudanças de requisitos de *software* sejam atendidas sempre nos prazos acordados. Assim são trabalhadas as metodologias ágeis, onde o processo é voltado para todas as pessoas envolvidas.

1.1 PROBLEMÁTICA DO TEMA

Um dos desafios em projetos de desenvolvimento de *software* é prover ao cliente *feedback* constante, explanando o atual andamento e as previsões de conclusão.

Outra questão importante, que por muitas vezes leva projetos ao fracasso, é a constância da mudança de requisitos. Se estes não estão muito bem definidos, o que é verdade em grande parte dos casos, as metodologias tradicionais para desenvolvimento de *software* geralmente não conseguem plena adaptação dos custos, tempo e pessoal necessário para conduzir um cenário imprevisto. E nem sempre os clientes são precisos em fornecer informações a respeito dos requisitos: falta clareza no que se precisa, ou, por algumas vezes, o cliente não conhece a sua real necessidade.

Projetos pequenos ou até mesmo aqueles em que o retorno de investimento deve se dar de forma rápida não encontram nas metodologias tradicionais adequação às suas necessidades.

1.2 JUSTIFICATIVA

O Scrum vem para garantir estes *feedbacks*, devido as frequentes reuniões que abordam vários dos estágios do projeto. Reuniões diárias, de abertura, de fechamento e de melhoria acontecem em cada etapa Scrum, chamada *Sprint*.

As frequentes mudanças de requisitos são vistas de forma muito positiva pela metodologia Scrum. Os escopos para os projetos são flexíveis, o que possibilita que a equipe de desenvolvedores esteja preparada para qualquer mudança. Ao aparecer um novo requisito, basta adicioná-lo na lista de tarefas futuras a fazer.

Scrum também sugere que se divida o projeto em pequenas partes, para assim agilizar o processo de desenvolvimento. Assim objetivos de curto prazo são cumpridos facilmente. As necessidades do cliente podem sofrer variações durante a condução do projeto, o que poderia afetar o resultado da entrega final nas metodologias clássicas.

1.3 OBJETIVOS

Os objetivos deste trabalho estão divididos em geral e específicos.

1.3.1 OBJETIVO GERAL

Este trabalho tem como objetivo geral apresentar uma lista de verificação para a condução de um projeto de desenvolvimento de *software*, ou parte dele, que utilize a metodologia ágil Scrum.

1.3.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos deste trabalho são a realização de um breve estudo sobre lista de verificação e sobre desenvolvimento ágil de *software*.

Após a apresentação sobre o desenvolvimento ágil de *software*, a metodologia ágil Scrum é explicada, juntamente com suas práticas gerenciais e sua definição de papéis.

Assim, é proposta uma lista de verificação que pode ser utilizada por um gerente de desenvolvimento de *software* ou até mesmo por uma equipe com o mesmo propósito, para avaliar o quão próximo da metodologia Scrum está o time. Esta lista deve ser utilizada antes, durante e depois de uma fase de desenvolvimento, pois acompanha as fases de planejamento, execução e verificação do Scrum.

1.4 ESTRUTURA DA MONOGRAFIA

Nos capítulos 2, 3 e 4 deste trabalho é apresentada a revisão bibliográfica. No capítulo 2 é realizado um breve estudo sobre lista de verificação, descrevendo seu conceito, aplicabilidade, principais tipos de listas existentes, e o que deve ser feito para elaborar uma lista de verificação.

No capítulo 3 é realizado um estudo sobre desenvolvimento ágil de *software*, destacando algumas diferenças entre metodologias ágeis e tradicionais para desenvolvimento de *software*. É explicado o Manifesto Ágil para desenvolvimento de *software*, onde são apresentados seus valores fundamentais e princípios.

Dentre várias metodologias ágeis existentes para desenvolvimento de *software*, o capítulo 4 apresenta a metodologia Scrum, a qual é o foco deste trabalho. São descritas as práticas de gerenciamento, os papéis de cada pessoa do time e quais são suas responsabilidades.

No próximo capítulo, o de número 5, é proposta a lista de verificação para desenvolvimento ágil de *software* em Scrum, contendo as atividades a serem cumpridas por uma equipe que utiliza Scrum para desenvolvimento de módulos ou novos projetos de *software*. Cada atividade da lista é descrita brevemente para um rápido entendimento. Logo após é demonstrada a lista de verificação, em forma de tabela, baseada na proposta desenvolvida no capítulo.

No capítulo 6 é apresentada a conclusão deste trabalho e posteriormente adicionamos algumas recomendações para trabalhos futuros.

2 LISTA DE VERIFICAÇÃO

Listas de verificação, ou *checklists*, são ferramentas para verificar se ações ou passos importantes estão sendo cumpridos ou também para evitar que alguns erros aconteçam, pois falhas humanas são comuns e ocorrem com frequência. São de fundamental importância para alcançar a qualidade (ISHIKAWA, 1998).

Por ser uma ferramenta simples e objetiva, a coleta de dados acontece de forma fácil e rápida.

Para elaborar uma lista de verificação, é necessário conhecimento sobre o assunto a ser tratado. Estas listas tornam-se mais completas e eficientes com o passar do tempo, já que são moldadas com a real necessidade de uma organização.

As listas de verificação são utilizadas nas mais variadas áreas do conhecimento, desde simples inspeções diárias em uma fábrica, até complexas cirurgias do crânio (BRUN, 2011). Processos ágeis e desenvolvimento de *software* também são foco destas listas.

Neste trabalho, trataremos lista de verificação como uma lista de tarefas a serem atingidas.

Potter e Sakry (2011) definem algumas diretrizes para a criação e execução de listas de verificação:

- Há dois estilos principais de listas: aquelas onde processos críticos são feitos e depois verificados (faça-confirme), ou aquelas que mostram quais passos devem ser executados para atingir um objetivo (leia-faça).
- Defina pontos no fluxo de trabalho onde será possível verificar os itens constantes na lista.
- Crie a lista com sentenças fáceis de ler e entender.
- Leia a lista em voz alta para a equipe: assim, garante-se que alguém que esteja com problemas possa se manifestar.
- Revise seu conteúdo e sua aplicação quantas vezes forem necessárias para que rapidamente problemas possam ser detectados.

Agrupar os itens por características ou afinidades semelhantes, que possuam situações em comum é recomendado ao definir listas. É necessário ter isso em

mente antes de qualquer elaboração. A este agrupamento é dado o nome de estratificação (ISHIKAWA, 1998).

3 DESENVOLVIMENTO ÁGIL DE SOFTWARE

O processo de desenvolvimento de *software* tradicional geralmente começa com a identificação e a especificação dos requisitos, para em seguida prosseguir com as fases de design de arquitetura, codificação, testes e manutenção (CHAGAS, 2008).

Metodologias clássicas para o desenvolvimento de *software*, como os modelos espiral e cascata, ajudaram a reduzir o estado caótico dos desenvolvimentos, focando-se em:

- Extenso planejamento para delinear o processo a ser seguido, identificando tarefas a serem concluídas durante o desenvolvimento do produto, e determinando pontos de monitoramento.
- Documentação completa na forma de especificação de requisitos de *software* com alto e baixo nível de detalhamento de documentos de projeto e planos de teste.
- Antecipação de necessidades futuras e projeto de arquitetura do sistema para acomodar as necessidades atuais e que virão.
- Monitoramento e controle de processos, gestão de riscos, controle de qualidade de *software* e garantia.

Porém desenvolvimentos que têm de tratar mudanças de requisitos de forma rápida não encontraram adequação às suas necessidades nas abordagens tradicionais. Estas muitas vezes além de inadequadas, são também caras. O custo de um requisito não identificado a ser tratado é muito alto nestes casos.

Isto trouxe a necessidade de criar métodos onde a relação custo-benefício acomodasse a mudança rápida de requisitos de *software*. Assim a abordagem chamada ágil passou a ser utilizada, e algum tempo depois foi divulgado o manifesto para o desenvolvimento ágil de *software*, que conta com quatro valores fundamentais (MANIFESTO, 2001):

1. Indivíduos e interação entre eles mais que processos e ferramentas.

2. Software em funcionamento mais que documentação abrangente.
3. Colaboração com o cliente mais que negociação de contratos.
4. Responder a mudanças mais que seguir um plano.

Estes valores são brevemente explanados abaixo:

1. Aqui os aspectos humanos no processo de desenvolvimento de *software* são considerados mais importantes do que até mesmo o processo em si. O espírito de equipe entre o time é motivado.

2. Adotar abordagens iterativas e incrementais são defendidas pelos métodos ágeis. O time de desenvolvimento de *software* deve produzir *software* em intervalos regulares. Uma documentação mínima é produzida.

3. As relações entre o time de desenvolvimento e os clientes são priorizadas, e contratos rigorosos deixados para um segundo plano. Métodos ágeis entregam *software* regularmente que proporcionam o máximo valor de negócio para os clientes, reduzindo o risco de contratos não serem cumpridos. E com os clientes também envolvidos no processo, os relacionamentos são fortalecidos.

4. Todos os envolvidos nos processos ágeis estão preparados para mudanças repentinas de planos para adaptação de qualquer mudança, independentemente da etapa do processo em que se encontram.

O manifesto ágil é guiado por doze princípios e valores que são seguidos por aqueles que o praticam (SANTOS; LUZ, 2003):

1. A maior prioridade é satisfazer o consumidor com a entrega contínua e adiantada de *software* com valor agregado.
2. Mudanças de requisitos são bem aceitas, até mesmo no fim do desenvolvimento. Processos ágeis se adequam a mudanças e obtêm vantagens através delas, visando vantagens competitivas para os clientes.
3. Softwares que funcionam são entregues frequentemente, na escala de poucas semanas e no máximo poucos meses, com preferência aos períodos mais curtos.

4. Durante todo o curso de um projeto, pessoas relacionadas a negócios e desenvolvedores devem trabalhar em conjunto diariamente.
5. Conduza o projeto em torno de pessoas motivadas. Proporcione todo o suporte necessário e confie nelas para fazer o trabalho.
6. O método mais eficiente e eficaz de transmissão de informações para e entre um time de desenvolvimento é através de uma conversa face a face.
7. Software funcionando é a medida primária de progresso.
8. Processos ágeis promovem um ambiente sustentável. Desenvolvedores, patrocinadores e usuários devem ser capazes de manter passos constantes.
9. A atenção contínua à excelência técnica e ao design aumenta a agilidade.
10. Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito.
11. As melhores arquiteturas, designs e requisitos vêm de equipes auto-gerenciáveis.
12. Em intervalos regulares, a equipe reflete em como se tornar mais eficaz e ajusta o seu comportamento para ficar de acordo.

Assim, nota-se que o objetivo do manifesto e dos princípios demonstrados não é desconsiderar processos, ferramentas e documentação, mas sim demonstrar o valor secundário que estes têm perante a *software* funcionando, pessoas e interações, colaboração entre todos e rápidas respostas às mudanças.

A mudança de paradigma em como se desenvolver *software* se caracteriza como a maior diferença entre as metodologias ágeis e as tradicionais. Ou seja, muda-se a forma de pensar e de adotar novos princípios para o desenvolvimento. Algumas outras diferenças foram relacionadas por Cavamura (2008):

- Metodologias ágeis são baseadas em dados estatísticos e levantados a partir do histórico de implantação do código, já as metodologias tradicionais baseiam-se em normas e padrões.

- Metodologias ágeis estão preparadas para aceitar mudanças repentinas durante qualquer fase de um projeto, ao passo que metodologias tradicionais oferecem resistência às mudanças, pois adotam a estratégia de previsibilidade, valendo-se de várias técnicas para levantamento de requisitos e para dominar todo o problema antes de iniciar o desenvolvimento.
- Metodologias ágeis têm sua forma de trabalhar imposta internamente. A própria equipe molda o seu jeito de trabalhar. Metodologias tradicionais seguem imposições externas, baseadas em modelos preexistentes, seguindo processos já definidos.
- Geralmente não há contrato com o cliente. Mas se houver, é totalmente flexível e é enfatizado que mudanças são bem vindas. As metodologias tradicionais são fundadas em contratos rigorosos.
- O cliente está presente junto à equipe de desenvolvimento, onde auxilia e propõe alterações na implementação. Nas metodologias tradicionais o cliente interage somente através de extensas reuniões, com poucos poderes para sugerir mudanças no desenvolvimento.
- O número de integrantes em uma equipe ágil é pequeno, em contrapartida, equipes tradicionais contam com vários integrantes.

Abrahamsson *et al.*(2002) analisam os métodos ágeis de acordo com as áreas:

Área	Características
Desenvolvedores	Ágeis, conhecedores, alinhados com o processo, colaborativos
Clientes	Dedicados, conhecedores, alinhados com o processo, colaborativos, representativos, qualificados
Requisitos	Mudam rapidamente
Arquitetura	Projetada para os requisitos atuais
Refatoração	Sem custos adicionais, bem vistas durante o desenvolvimento
Tamanho	Equipes pequenas
Objetivo principal	Rápido retorno no que foi investido

QUADRO 1 – ÁREAS E CARACTERÍSTICAS NOS MÉTODOS ÁGEIS

FONTE: Abrahamsson *et al.*(2002), tradução nossa.

4 SCRUM

Metodologias tradicionais para desenvolvimento de *software* focam na geração de extensa documentação e processos rígidos para o andamento de projetos. A proposta ágil Scrum concentra as atenções no desenvolvimento e na relação entre as pessoas (CARVALHO; MELLO, 2009).

Scrum é uma das abordagens existentes para desenvolvimento ágil de *software*. Não fornece regras complexas e extensas de como deve ser feito o projeto, pois grande parte disto é deixado para a equipe de desenvolvimento.

O processo Scrum foi criado por Jeff Sutherland em 1993 juntamente com pesquisadores ligados à área de desenvolvimento de *software*, e é nesta área que encontrou maior adesão. Porém, não é restrito somente a projetos de *software*.

É um processo ágil com alta aplicabilidade em qualquer processo iterativo e incremental (CARVALHO; MELLO, 2009).

O desenvolvimento é feito em ciclos curtos com a duração de algumas semanas, chamados de iterações, de modo que ao final de cada ciclo o cliente receba um entregável que agregue valor ao seu negócio. Isto garante adequação total às mudanças de requisitos e faz com que o time receba constantes avaliações do andamento do projeto pelos clientes, reduzindo os riscos.

Apesar de não possuir práticas específicas para desenvolver *software*, algumas práticas de gerenciamento são exigidas pelo Scrum (SCHWABER, 1995):

- *Product backlog*: uma lista com todas as atividades estimadas e priorizadas a serem desenvolvidas que resultarão no produto final. Esta lista pode ser elaborada pelos clientes, áreas de vendas, suporte, desenvolvimento. É atualizada constantemente, adicionando novos itens e melhor descrevendo os já existentes.
- *Sprint*: intervalos de tempo, ciclos, onde o trabalho é realizado e um entregável é apresentado no final.
- *Sprint planning meeting*: reunião organizada pelo *Scrum Master,* em duas fases. Na primeira participam usuários, clientes, gerência, *Product Owner* e a equipe Scrum, para decidir quais os objetivos devem ser atingidos e atividades que devem ser executadas no próximo *Sprint*. Na segunda fase, o

Scrum *Master* e o time se reúnem para definir como as atividades serão desenvolvidas.

- *Sprint backlog*: Cada *Sprint* começa com os itens que serão trabalhados durante uma iteração. O *Sprint Backlog* é a lista resultante da reunião *Sprint Planning*. Quando todos os itens desta são concluídos, uma nova entrega do sistema pode ser feita.
- *Daily meeting*: são reuniões diárias, de no máximo 15 minutos, organizadas para avaliar o progresso do time no *Sprint*. Conta também com um pouco de planejamento, pois nela os participantes devem falar o que cada um fez desde a última reunião, e o que farão até a próxima. Os problemas e impedimentos também devem ser citados. Na presença destes, o Scrum *Master* deve atuar para removê-los e aprimorar o processo.
- *Sprint Review meeting*: no último dia de *Sprint*, o *Scrum Master* juntamente com o time apresenta os resultados do *Sprint* à gerência, usuários e *Product Owner*. Aqui são definidos os passos para as próximas atividades. Novos itens podem ser adicionados ao *backlog* a partir desta reunião.

Além destas práticas, Abrahamsson (2002) destaca seis papéis definidos para a execução do Scrum e suas responsabilidades:

- *Scrum Master*

  O Scrum *Master* é a pessoa responsável por garantir que o projeto é conduzido de acordo com as práticas, valores e regras do Scrum, juntamente com o progresso esperado definido em etapa anterior. É peça chave, pois interage de forma harmoniosa com a equipe de desenvolvimento, os clientes e a gerência. E como papel principal deve garantir que todos os impedimentos e obstáculos sejam removidos do processo para que o time continue trabalhando o mais produtivo possível.

- *Product Owner*

  O *Product Owner* é o responsável por projetar, manter, gerenciar, controlar e tornar visível o *Product Backlog.*

É selecionado em conjunto pelo *Scrum Master*, o cliente e a gerência. Todas as decisões finais relacionadas ao *Backlog* é ele quem define. Participa também da estimativa das tarefas e torna os itens presentes na lista em funcionalidades a serem desenvolvidas.

- Time Scrum

  O time em Scrum é uma equipe autogerenciável e tem o poder de tomar ações necessárias para sua organização no intuito de atender aos objetivos de cada *Sprint*.

  Participa também estimando o esforço de desenvolvimento e na criação do *Sprint Backlog*.

  Multidisciplinar, trabalha em conjunto como uma única unidade. Não há uma clara definição de papéis, porém cada um tem maiores habilidades em determinadas áreas.

- Cliente

  Figura que atua nas atividades referentes aos itens do *Product Backlog* do sistema que está sob desenvolvimento ou sendo melhorado.

  Também pode participar do desenvolvimento, sendo comum colocar um cliente dentro da equipe.

- Gestão

  É responsável pela tomada de decisão final, de acordo com as metas a serem atingidas no projeto. Pode ser formada pela gerência ao qual o projeto que será conduzido ou pelo gerente de projetos.

Schwaber (1995) descreve os seguintes grupos de fases para o Scrum:

- Pré-jogo

Planejamento: define-se uma nova versão a ser entregue, baseada no *backlog* conhecido, juntamente com uma estimativa de custo e cronograma. Se um novo sistema deve ser desenvolvido, esta fase é dedicada para conceituação e

análise. Se há um sistema que deva ser aprimorado, limita-se o escopo da análise nesta fase.

Passos a serem executados nesta fase:

- Elaboração de um *backlog*;
- Definição da data de entrega e das funcionalidades contidas no *backlog*;
- Elencar as atividades mais apropriadas para início imediato;
- Definição da equipe que dará andamento do *Sprint*;
- Levantar os riscos e as maneiras de como controlá-los;
- Revisão e ajuste dos itens do *backlog*;
- Validação das ferramentas de desenvolvimento e de infraestrutura;
- Aprovação da área gestora e financeira.

Arquitetura: consiste em definir como os itens do *backlog* serão implementados.

Passos a serem executados nesta fase:

- Identificar mudanças necessárias para implementação dos itens presentes no *backlog*;
- Redefinir a arquitetura do sistema de acordo com novos contextos e requisitos;
- Identificar problemas que possam ocorrer na implementação de mudanças;
- Moldar a reunião de *Review* (revisão).

- Jogo

*Sprint*s: aqui é iniciado o desenvolvimento das funcionalidades alinhadas no planejamento, respeitando as variáveis de tempo, requisitos, qualidade e custos. Vários *Sprint*s são conduzidos sucessivamente, em forma de ciclos, para chegar no resultado esperado da construção ou melhoria do sistema, até que o produto seja considerado pronto para distribuição.

Os *Sprints* são uma série de atividades de desenvolvimento executadas num período pré-determinado de tempo, geralmente de uma a quatro semanas. Os intervalos entre eles são definidos de acordo com a complexidade do produto e seus riscos.

Cada *Sprint* é conduzido por uma ou mais equipes executando as seguintes etapas:

- Desenvolvimento

  Executar as mudanças para os itens de *backlog*, realizar análise, design, desenvolvimento, testes e documentação para as mesmas.

- Empacotamento

  Agrupar os itens da etapa anterior, criar versão executável ou apresentar de outra maneira definida no planejamento.

- Revisão

  Os times se reúnem para apresentar os trabalhos feitos e demonstrar o progresso do trabalho. Problemas são analisados e resolvidos. Novos itens podem ser adicionados ao *backlog*. Riscos são reavaliados, responsabilidades definidas.

- Ajustes

  Consolidar as informações reunidas nas revisões e executar os ajustes necessários.

Após cada *Sprint* a reunião de *Review* deve ser conduzida. Nela toda a equipe, o *Product Owner*, o *Scrum Master* e o responsável pela gestão devem estar presentes e participar. Outras áreas também podem interagir com a reunião, como financeiro, marketing, vendas.

Nela são verificados os quesitos funcionais e executáveis do programa ou módulo desenvolvido, onde avalia-se os objetos atribuídos à equipe e seu cumprimento.

O modo como os itens do *backlog* serão abordados no próximo *Sprint* podem ser mudados com a sugestão dos participantes. Novos itens podem ser adicionados à lista e já atribuídos a algum dos membros dos times envolvidos.

Por fim, a duração do próximo ciclo é definida, baseada no ciclo passado e complexidade das novas atividades. Não é recomendado definir um *Sprint* com mais de quatro semanas.

- Pós-jogo

Encerramento: preparar para liberar a versão, elaboração da documentação final, testes finais e liberar a versão.

Material de treinamento e marketing são preparados também na fase de fechamento.

5 LISTA DE VERIFICAÇÃO PARA SCRUM

Apresentamos a seguir a lista de verificação de acordo com as seguintes fases para desenvolvimento de *software* da metodologia Scrum: planejamento, organização, execução e monitoração.

Na primeira parte, são tratados os tópicos relativos às fases e uma explicação para cada um dos itens. Na segunda parte, estes itens são organizados em forma de tabela, para que se possa indicar o cumprimento ou não de determinada prática. Práticas não aplicáveis em determinado momento também possuem espaço para marcação.

5.1 PLANEJAMENTO

O *Product Owner* está definido

Por escolha dos *stakeholders* é definida uma pessoa que assumirá a função de *Product Owner*.

O *Scrum Master* está definido

Assim como o *Product Owner*, o *Scrum Master* é escolhido pelos *stakeholders*.

Time multidisciplinar

O time deve ser composto por pessoas multidisciplinares, que possam assumir várias tarefas. Assim é possível identificar a pessoa correta para a função correta, sem correr o risco de que, em uma emergência, grandes atrasos aconteçam.

O time possui um *Product Backlog*

O time possui um *Product Backlog* com as atividades definidas e priorizadas pelo *Product Owner*.

O *Product Owner* tem o poder de priorizar itens no *Product Backlog*

Em contato direto com os *stakeholders*, o *Product Owner* define quais os itens prioritários para execução durante os *Sprints*.

O *Product Owner* tem o conhecimento para priorizar itens no *Product Backlog*

Não basta o *Product Owner* ter o poder de priorização, deve possuir também pleno conhecimento de cada história presente no *Product Backlog* e a autoridade para a tomada de decisão. Somente assim elas estarão priorizadas com a real necessidade da companhia. Caso contrário, o conhecimento deve ser alinhado com os *stakeholders*, e então a lista priorizada.

O *Product Owner* está em contato direto com o time e com os *stakeholders*

O processo do Scrum acontece de maneira mais eficiente quando o *Product Owner* possui livre comunicação tanto com o time quanto com os *stakeholders*. Assim todos ficam cientes das necessidades específicas de ambas as partes, minimizando a possibilidade de erros ou mal entendimento de funcionalidades.

O local para os *Daily Meetings* está definido

Deve ser em frente ao quadro de acompanhamento, se este existir. É aconselhável evitar a mudança de local durante um *Sprint*, e obrigatório garantir que não existam interrupções durante o momento da reunião.

Definição da duração do *Sprint*

Ao definir a duração de um *Sprint*, deve-se ter em mente que iterações longas prejudicam a agilidade de entregas, um dos pontos principais do Scrum. Não é necessário dispender muito esforço na definição do número de semanas que o *Sprint* vai durar. Apenas defina um tempo e execute. A experiência irá definir este número naturalmente, após algumas entregas.

Executar a reunião de planejamento

O time e o *Product Owner* reúnem-se para a definição do *Sprint Backlog*. São escolhidas as atividades priorizadas anteriormente no *Product Backlog* e é feita uma estimativa de tempo para cada uma. Todas as pessoas presentes na reunião devem entrar em consenso a respeito das horas estimadas para cada tarefa. Esta reunião deve ter horário de início e término definidos.

Todos do time participam da reunião

Aconselha-se que todos os membros do time participem da reunião de planejamento do *Sprint*, onde a opinião de todos deve ser ouvida a analisada. Ao final, todos devem concordar com o planejamento e deve haver comprometimento com ele.

Pontualidade no horário estipulado para início e fim da reunião

A reunião deve começar e terminar no horário proposto. Reuniões extensas são cansativas e improdutivas a partir de certo ponto, além de não pactuar com o processo ágil.

Product Owner participa

A reunião de planejamento do *Sprint* resultará no *Sprint Backlog* definido e estimado. O *Product Owner* deve participar da reunião, pois conhece os itens presentes no *Product Backlog*. Ao final, deve estar satisfeito com as prioridades definidas.

Elaborar o *Sprint Backlog*

As atividades a serem trabalhadas no *Sprint* constam no *Sprint Backlog*, que é resultado da reunião de planejamento *Sprint Planning Meeting*.

Definição dos intervalos entre os *Sprint*s

Para manter o processo ágil, um dia deve ser o espaço máximo entre os *Sprint*s.

5.2 ORGANIZAÇÃO

Os membros localizam-se próximos

Os membros do time devem, preferencialmente, estar localizados próximos uns aos outros. Dificulta a relação interpessoal e a agilidade pessoas em setores, salas, e até mesmo andares diferentes.

O Scrum *Master* encontra-se próximo ao time

O Scrum *Master* deve estar próximo ao time na execução do *Sprint*, para facilitar o contato com esta figura que é o ponto de apoio.

Scrum *Master* conhece seu papel

O foco do Scrum *Master* deve ser a mitigação dos impedimentos e das adversidades levantadas pelo time. Em segundo plano, ele também ajuda na condução das prioridades. Em caso de desconhecimento de suas atividades, uma nova pessoa deve ser escolhida pelos *stakeholders*.

Disponibilização do *Sprint Backlog* em local de fácil visualização e acesso

O *Sprint Backlog* deve estar localizado em um lugar visível por todos, como por exemplo fixado na sala onde o time está localizado ou armazenado em local de fácil acesso, se este for em meios digitais.

Disponibilização do *Product Backlog* em local de fácil acesso

Da mesma forma que o *Sprint Backlog*, o *Product Backlog* deve estar localizado em meio de fácil acesso a todos envolvidos com o Scrum. Dar preferência ao armazenamento digital deste documento.

Product Backlog atualizado antes da ocorrência do próximo planejamento

O *Product Backlog* deve estar atualizado antes da ocorrência da reunião de planejamento de *Sprint*. O *Product Owner* deve conhecer todas as histórias presentes neste *backlog*.

As partes interessadas estão cientes do *Sprint*

Os *stakeholders* e clientes devem estar cientes do acontecimento de um *Sprint*. Devem também ser informados das datas iniciais e finais do *Sprint* definidas previamente.

Outros times e departamentos da empresa estão cientes do *Sprint*

Informar os times e departamentos da empresa sobre o acontecimento de um *Sprint*. É interessante fornecer informações básicas sobre o processo ágil, de fácil entendimento, para que o time, totalmente focado, permaneça bem visto perante as outras áreas.

O time possui um *Burndown Chart*

O *Burndown chart* é uma ferramenta de acompanhamento muito importante do Scrum. Muitas vezes a motivação e a velocidade do time dependem da visualização diária desta ferramenta. Quando o gráfico aponta um atraso, todos se unem para priorizar e executar mais atividades. Quando o gráfico aponta normalidade, todos tendem a não deixá-lo em atraso.

O *Burndown Chart* deve estar visível para todas as partes interessadas

Dê preferência a fixá-lo na sala onde o time trabalha, e também no local onde é realizada a reunião diária de acompanhamento.

5.3 EXECUÇÃO

O time segue as atividades priorizadas pelo *Product Owner*

O *Product Owner* prioriza as atividades do *Product Backlog*, e estas passam a fazer parte de um *Sprint*. Caso todas as atividades acabem antes do previsto, é possível recuperar um item do *Backlog* e executá-lo.

Todos os problemas são repassados ao Scrum *Master*

O time repassa todos os problemas ao Scrum *Master*, e este encontra os meios para saná-los.

Scrum *Master* revisa os itens repassados a ele

O Scrum *Master* deve revisar todos os problemas repassados anteriormente a ele e encontrar formas de saná-los.

Discussão e solução de um problema

Quando um problema ocorre, deve ser discutido no mesmo momento em que aconteceu, e não deixado para depois. Assim, as ideias a respeito do ocorrido ainda estarão recentes na memória e será mais fácil solucioná-lo ou encontrar uma alternativa de contorno.

Atualizar o *Burndown Chart* diariamente

O *Burndown Chart* deve ser atualizado a cada dia, ou em períodos definidos pelo time, mas nunca menores que um dia. Isso facilita a visualização do real andamento do *Sprint*.

O time reage quando o *Burndown Chart* indica atraso

A reação do time quando o *Burndown Chart* está distante da linha de acompanhamento prevista inicialmente é auxiliada pelo Scrum *Master*. Após algumas iterações de *Sprint* executadas, esta iniciativa acontece de forma automática.

Evita-se a realização de horas extras

Hora extra deve ser esporádica. O time faz hora extra somente quando realmente necessário.

Administrar atividades não planejadas de maneira sensata

Atividades não planejadas geralmente acontecem, pois não é possível prever todos os acontecimentos. Atividades destas que requererem muitas horas devem ir para o *Product Backlog*, para que posteriormente sejam trabalhadas em um novo *Sprint*.

Demonstração de um entregável

Ao final do *Sprint* deve haver uma entrega, seja documentação ou módulo de um programa, testado e validado.

Daily Meetings acontecem

As reuniões diárias de acompanhamento, *Daily Meetings*, acontecem todos os dias no mesmo horário e lugar.

Daily Meetings são curtas e entendidas por todos

As *Daily Meetings* iniciam e finalizam no tempo estipulado. Todos reconhecem sua importância e colaboram para seu bom andamento.

Todos os membros do *Sprint* devem estar presentes na *Daily Meeting*

Todo o time deve estar presente. O *Product Owner* também é encorajado a participar sempre que possível. Outras áreas também podem assistir à reunião, não existe o conceito de "não convidados".

Todos respondem às três perguntas durante as *Daily Meetings*

Os *Daily Meetings* devem acontecer diariamente, preferencialmente em pé, para que o prazo máximo de 15 minutos não seja estendido.

Os participantes devem responder a três perguntas:

- O que fiz desde a última reunião?
- O que farei até a próxima reunião?
- Estou com algum obstáculo para a execução de alguma atividade?

Aqui o Scrum *Master* anota os comentários para tomar uma ação para os que elencaram problemas.

Qualquer um pode conduzir a reunião

Deve ser incentivada a rotatividade na condução das reuniões. Qualquer membro do time pode tomar a iniciativa, não deixando esta atividade somente para o Scrum *Master*.

As *Daily Meetings* não podem sofrer interrupções ou pausas

Todas as pessoas externas ao *Sprint* são informadas que as reuniões diárias não podem ser interrompidas.

5.4 MONITORAÇÃO

A reunião de retrospectiva deve acontecer.

Após o fechamento do *Sprint*, a reunião de retrospectiva deve acontecer.

Todos participam

Todos os membros do time e o *Product Owner* participam.

Revisão do que foi feito

Neste ponto, deve ser discutido o que foi executado durante o *Sprint* e apontar falhas ou pontos não observados. As atividades não planejadas executadas também podem ser analisadas.

Elencar melhorias

Como produto final, a retrospectiva resulta em propostas de melhoria, a serem implantadas e validadas nos próximos *Sprint*s.

Todos falam

Todos os participantes tem o direito da palavra e devem usá-lo para expor suas opiniões.

LISTA DE VERIFICAÇÃO PROPOSTA PARA O MODELO SCRUM

Planejamento				
Atividade	Descrição	Atende	Não Atende	Não se aplica
Definição do *Product Owner*(PO)	O *Product Owner* está definido			
Scrum *Master*	O Scrum *Master* está definido			
Composição do time	Time multidisciplinar			
Product Backlog	O time possui um *Product Backlog*			
Product Backlog – Product Owner	O *Product Owner* tem o poder de priorizar itens no *Product Backlog*.			
Product Backlog – Product Owner	O *Product Owner* tem o conhecimento para priorizar itens no *Product Backlog*.			
Product Owner	O *Product Owner* está em contato direto com o time e com os *stakeholders*.			
Daily Meetings	O local para os *Daily Meetings* está definido.			
Duração do *Sprint*	Definição da duração do *Sprint*			
Sprint Planning Meeting	Executar a reunião de planejamento			
Sprint Planning Meeting	Todos do time participam da reunião			
Sprint Planning Meeting	Pontualidade no horário estipulado para início e fim da reunião			
Sprint Planning Meeting	*Product Owner* participa			

Sprint Backlog	Elaborar o *Sprint Backlog*			
Sprint – intervalos	Definição dos intervalos entre os *Sprint*s			

Organização				
Atividade	Descrição	Atende	Não Atende	Não se aplica
Localização do Time	Os membros localizam-se próximos			
Localização do Scrum *Master*	O Scrum *Master* encontra-se próximo ao time			
Atividades do Scrum *Master*	Scrum *Master* conhece seu papel			
Sprint Backlog - Localização	Disponibilização do *Sprint Backlog* em local de fácil visualização e acesso			
Product Backlog - localização	Disponibilização do *Product Backlog* em local de fácil acesso			
Product Backlog – Atualização e manutenção	O *Product Backlog* atualizado antes da ocorrência do próximo planejamento			
Stakeholders e clientes	As partes interessadas estão cientes do *Sprint*			
Departamentos da empresa	Outros times e departamentos da empresa estão cientes do *Sprint*			
Burndown Chart	O time possui um *Burndown Chart*.			
Burndown Chart	O *Burndown Chart* deve estar visível para todas as partes interessadas			

Execução				
Atividade	Descrição	Atende	Não Atende	Não se aplica
Sprint – Prioridades	O time segue as atividades priorizadas pelo *Product Owner*			
Sprint - adversidades	Todos os problemas são repassados ao Scrum Master			
Scrum *Master*	Scrum Master revisa os itens repassados a ele			
Sprint - adversidades	Discussão e solução de um problema			
Burndown Chart	Atualizar o *Burndown Chart* diariamente			
Burndown Chart	O time reage quando o *Burndown Chart* indica atraso			
Sprint – Atrasos	Evita-se a realização de horas extras			
Sprint – atividades não planejadas	Administrar atividades não planejadas de maneira sensata			
Sprint – Finalização	Demonstração de um entregável			
Daily Meetings	*Daily Meetings* acontecem			
Daily Meetings	*Daily Meetings* são curtas, geralmente com 15 minutos de duração, e entendidas por todos			
Daily Meetings	Todos os membros do *Sprint* devem estar presentes na *Daily Meeting*			
Daily Meetings	Todos respondem às três perguntas durante as *Daily Meetings*			
Daily Meetings	Qualquer um pode conduzir a reunião			
Daily Meetings	As *Daily Meetings* não podem sofrer interrupções ou pausas			

Monitoração - *Sprint Retrospective*			
Descrição	Atende	Não Atende	Não se aplica
A reunião de retrospectiva deve acontecer.			
Todos participam			
Revisão do que foi feito			
Elencar melhorias			
Todos falam			

6 CONCLUSÕES

Para aplicar Scrum em uma empresa pode-se valer de algum tempo de preparação para a adoção de uma metodologia de desenvolvimento ainda não utilizada. E assim, após aplicada e com um certo nível de maturidade, a empresa pode desejar avaliar o desempenho desta nova escolha.

Neste trabalho foi estudado o conceito do desenvolvimento ágil de *software*, onde foram demonstrados seus valores e princípios. Este conceito está focado nas pessoas, valorizando o trabalho em equipe e a comunicação. Visa também produzir um *software* com qualidade e rapidez, com entregas rápidas e que atenda às reais necessidades de seus clientes.

É possível observar que ambas metodologias para desenvolvimento de software, sejam tradicionais ou ágeis, possuem vantagens e desvantagens. Ao optar por aplicar uma proposta ágil, é necessário estar preparado para mudanças. Qualquer novo método provoca mudança no modo de trabalhar e de conduzir projetos. Os fatores como pessoas, qualidade, custos e prazos podem ter outros parâmetros para serem avaliados.

Listas de verificação são ótimas formas de se avaliar a qualidade de um processo, pois ajudam a relembrar se passos importantes foram executados, e também minimizam a possibilidade de que erros aconteçam.

A lista desenvolvida neste trabalho objetiva ser utilizada como fonte para que as empresas verifiquem se cada uma das práticas gerenciais do Scrum e os papéis definidos nesta metodologia estão sendo implementados ou não em um projeto em andamento.

Para trabalhos futuros, sugerimos mesclar esta lista de verificação em Scrum juntamente com outras metodologias ágeis para desenvolvimento de *software*, como por exemplo *Extreme Programming*. Assim, esta lista ficará mais completa e abordará novos conceitos ágeis, ajudando assim os gerentes ou interessados no assunto na condução de projetos mais eficientes utilizando mais de uma metodologia ágil existente.

Sugerimos também realizar o estudo da aplicação da lista de verificação elaborada em sua forma prática, avaliando seus erros e acertos, modificando-a afim de encontrar pontos de melhoria.

Outra sugestão é elaborar documentos padronizados para as práticas de gerenciamento Scrum, como por exemplo um modelo de *sprint backlog* ou de *burndown chart* que venha a ser utilizado pelas equipes que decidirem implementar a metodologia ágil no desenvolvimento de seus projetos. Uma ferramenta específica pode ser elencada para a elaboração destes documentos, como um *software* de gerenciamento de projetos ou uma planilha eletrônica.

REFERÊNCIAS

ABRAHAMSSON, P.; SALO, O.; RONKAINEN, J.; WARSTA, J. **Agile software development methods**, publicado em 2002. Disponível em: <www.vtt.fi/inf/pdf/publications/2002/P478.pdf>. Acesso em: 17/05/2011.

BRUN, J. **What is a checklist?**, publicado em 2011. Disponível em: <http://www.nimonik.ca/2011/10/what-is-a-checklist/>. Acesso em: 25/11/2011.

CARVALHO, B. V. de; MELLO, C. H. P. **Revisão, análise e classificação da literatura sobre o método de desenvolvimento de produtos ágil Scrum**, publicado em 2009. Disponível em: <http://www.simpoi.fgvsp.br/arquivo/2009/artigos/E2009_T00109_PCN92031.pdf>. Acesso em: 25/12/2011.

CAVAMURA JÚNIOR, L. **Aqua - Atividades de qualidade do contexto ágil**, publicado em 2008. Disponível em: <http://lapes.dc.ufscar.br/teses-e-dissertacoes/pdf/Luiz.PDF>. Acesso em: 10/01/2012.

CHAGAS, M. V. L.; **Processo de desenvolvimento de software: "O estado da arte"**, publicado em 2008. Disponível em: <www2.dc.uel.br/nourau/document/?down=685>. Acesso em: 03/02/2012.

ISHIKAWA, K. **Princípios gerais dos círculos de controle da qualidade**, publicado em 1988. Disponível em: <http://gestaodaqualidade.blog.com/files/2011/03/7-Ferramentas-da-Qualidade-de-Kaoru-Ishikawa-TP.doc>. Acesso em: 01/09/2011.

MANIFESTO for Agile Software Development. Disponível em: <http://agilemanifesto.org/>. Acesso em: 23/09/2011.

POTTER, N; SAKRY, M. **Using checklists to define best practices and improve performance**, publicado em 2011. Disponível em: <http://www.compaid.com/caiinternet/ezine/Potter-quality-article3.pdf>. Acesso em: 25/11/2011.

SANTOS, R. G. dos; LUZ, G. D. **Métodos ágeis**, publicado em 2003. Disponível em: <http://www.ime.usp.br/~gdaltonl/ageis/ageis_6pp.pdf>. Acesso em: 10/01/2012.

SCHWABER, K. **Scrum Development Process**, publicado em 1995. Disponível em: <http://assets.Scrumfoundation.com/downloads/2/Scrumpapers.pdf?1285932052>. Acesso em: 01/08/2011.