ESCOLA POLITÉCNICA
DE PERNAMBUCO

Aplicação de Scrum em Ambiente de Desenvolvimento de Software Educativo

Trabalho de Conclusão de Curso

Engenharia de Computação

Autora: Michele de Vasconcelos Leitão

Orientadora: Profº. Cristine Martins Gomes de Gusmão

ESCOLA
POLITÉCNICA DE
PERNAMBUCO

Michele de Vasconcelos Leitão

Aplicação de Scrum em Ambiente de Desenvolvimento de Software Educativo

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia da Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Recife, junho de 2010.

Agradecimentos

Foram muitos os que contribuíram para o meu crescimento e sucesso ao longo da vida acadêmica, e seria injusto não mencionar a todos, tamanha é minha gratidão. Portanto, espero que seja perdoada se porventura esquecer alguém.

Agradeço, antes de tudo, à minha família, minha fonte de inspiração, meu modelo não de perfeição, mas de perseverança e comprometimento. Pela minha família aprendi o valor de obter as próprias conquistas; e é através desse trabalho que retribuo esse valioso ensinamento. À minha mãe, Eneida Castanha, pela gigante paciência, pelo incondicional amor e pelas críticas (construtivas) que constroem meu caráter dia após dia; à minha prima Alessandra Ramos, pela ajuda e presença sempre, inclusive nos momentos críticos; aos meus avós, Múcia e Ibiapina Leitão, pelo carinho e confiança que sempre terão; ao meu pai, João Damasceno, pelos princípios e virtudes ensinados que serão seguidos para sempre; e à minha tia Patrícia Leitão, pelo cuidado e lembrança eternos.

Agradeço, então, ao meu querido Ricardo Teixeira, que não pode ser rotulado, por representar tanto para mim: amigo, companheiro, amor, cúmplice, entre inúmeros outros. Não somente por estar ao meu lado, mas também por se fazer presente mesmo não estando ao lado. Pelo suporte emocional, intelectual e financeiro (porque não dizer?) imprescindíveis para me ajudar manter minhas metas, para sempre me incentivar a continuar no caminho que eu tracei. Da mesma forma que não posso rotular seu papel em minha vida, não posso rotular meu sentimento de gratidão.

Agradeço, inclusive, aos meus Amigos de jornada, estes que, até o fim, não só proporcionaram força e estímulo para continuar, como mostraram o poder de cativar. Por isso, venho dizer que sei que sou “eternamente responsável por aquilo que cativo”.

Agradeço, não menos, às minhas Amigas da vida toda, que me ensinaram a escutar, a não julgar e a ceder; tarefas tão complicadas e ainda assim tão importantes. A elas, essas 5 pessoas tão diferentes, mas tão queridas, meu muito

obrigado, pelas horas de choros e lamentações, pelos momentos de alegrias e por compreender minhas ausências.

Sem esquecer minha professora e orientadora, Prof.ª Cristine Gusmão, a quem tive a oportunidade de conhecer e admirar, mesmo que há pouco tempo, e que soube como ninguém me guiar pelas etapas deste trabalho; e os mestrandos Júlio Venâncio e Keldjan Alves, que com cuidado, paciência e disponibilidade muito ajudaram na elaboração da minha apresentação: muito obrigado!

Agradeço, também, aos mais envolvidos neste trabalho, as pessoas que tornaram possível, através de muita paciência e dedicação, a realização dessa experiência: meu Scrum *Team*. Aos programadores Lara Dantas e Rafael Castanha, aos designers Guilherme Botelho e Thiago Canuto e ao professor Sérgio Bezerra, meu muito obrigado pela participação ativa nesse projeto e pela determinação que possibilitou o sucesso deste trabalho. Agradeço a Educandus, empresa onde se deu o estudo e, em especial, a Ricardo Lessa, meu *Product Owner*, pelo aval para a implantação dessa proposta, e pelo interesse e apoio na ideia.

Por fim, agradeço a todos que não foram especificados, mas sabem que participaram dessa conquista e merecem, sempre, minha gratidão.

Resumo

É possível perceber que os problemas de atrasos na entrega do projeto, produtos de baixa qualidade e aumento significativo dos custos enfrentados pelas empresas de desenvolvimento de software são causados principalmente pela falta de gerenciamento nos processos, que gera falhas de impactos econômicos e humanos enormes. Nesse sentido, visando encontrar uma maneira de desenvolver software com qualidade e entrega em prazo em ambiente de desenvolvimento de softwares educativos, este trabalho avalia o uso da metodologia ágil Scrum, como uma alternativa. Para iniciar sua utilização, através da elaboração de uma estratégia de implantação, foi realizado um experimento numa empresa real do setor de desenvolvimento de softwares educativos. Nesta empresa foram analisadas mudanças de papéis e responsabilidades nos recursos humanos; inclusão de recursos e aplicativos de suporte ao uso da metodologia e alterações na distribuição de tarefas pelo tempo de desenvolvimento. Por meio da análise dos resultados da implantação, pôde-se observar que a metodologia ágil Scrum se mostrou adequada para o uso em ambientes de desenvolvimento de softwares educativos, e que sua aplicação engloba todas as etapas do desenvolvimento desses ambientes, através de pequenas e médias adaptações. Além disso, essa análise mostrou a boa adaptabilidade e aceitação da equipe à metodologia que, por ter sido rapidamente assimilada, não deixou que a queda de produtividade inicial à implantação do método pudesse ter impacto no cronograma total do projeto.

Abstract

It's possible to notice that the problems of delays in project delivery, low quality products and significant cost increases faced by software development's enterprises are mainly caused by the lack of management processes, which generate flaws with enormous human and economic impacts. So, in order to find a way to develop software with quality and in-time delivery on the educational software development's environments this work evaluates the use of Scrum agile methodology, as an alternative. To initiate the application, by developing a deployment strategy, were executed an experiment on a real enterprise from the sector of educational software development. At this enterprise were analyzed changes on roles and responsibilities in human resources, including resources and applications to support the use of the methodology, and changes in the distribution of tasks by development time. By analyzing the results of the implementation could be observed that the Scrum agile methodology was adequate for the use on educational software development's environments, and its application encompasses all stages of development of these environments, through small and medium-size adjustments; besides, this analysis showed a good adaptability and acceptance of the methodology by the team. By having been quickly assimilated, did not allow the initial drop in productivity – caused by the implementation of the method – to have an impact to the project schedule.

Sumário

Índice de Figuras

Índice de Tabelas

Tabela de Símbolos e Siglas

COBIT – *Control OBjectives for Information and related Technology*

CMMI – *Capabitity Maturity Model Integration*

ITIL – *Information Technology Infrastructure Library*

PO – *Product Owner*

ROI – Retorno Sobre Investimento (*Return on Investment*)

TI – Tecnologia da Informação

XP – *EXtreme Programming*

Capítulo 1

Introdução

Nos últimos anos, o software se tornou um componente vital nos negócios. O sucesso de uma organização está cada vez mais dependente da utilização do software como um diferencial competitivo [PRIKLADNICKI, 2003]. Entretanto, a globalização de mercado – tornando possível o acesso a novos tipos de mercados e o acesso de "jogadores" exteriores aos mercados domésticos de cada empresa [TAPSCOTT & CASTON, 1993], iniciou um processo de aumento da concorrência, tornando-a muito acirrada. E essa concorrência, exigindo melhores produtos, acarretou na menor, a tolerância às falhas, atrasos e cancelamentos de projetos. [SAVOINE et al., 2009].

Analisando o cenário de produção de software é possível perceber que esses são os maiores problemas enfrentados pelas empresas de desenvolvimento de software (atrasos na entrega do projeto, produtos de baixa qualidade e aumento significativo dos custos) e são causados, entre outros fatores, pela falta de gerenciamento nos processos. De acordo com Beck [2004], o desenvolvimento de software tem falhas na entrega e nos valores entregues e essas falhas têm impactos econômicos e humanos enormes. É necessário achar uma maneira de desenvolver software com qualidade e entregas frequentes.

Neste sentido, organizações que procuram melhoria em seus processos de produção através de populares frameworks de gestão como *Capabitity Maturity Model Integration* (CMMI), COBIT, ITIL, entre outros, agora também acreditam que introduzir conceitos ágeis em seus processos buscando um equilíbrio entre agilidade e maturidade é uma alternativa capaz de promover a melhoria da qualidade dos seus produtos e, consequentemente, aumento da competitividade no mercado [SZIMANSKI, 2009, apud PLAYFAIR, 2008].

O ambiente de desenvolvimento de software não é “tímido” quando se trata em introduzir novas metodologias. De fato, nos últimos 25 anos, um grande número de diferentes abordagens para o desenvolvimento de software tem sido introduzido,

dos quais apenas alguns sobreviveram para serem usados hoje [ABRAHAMSSON et al., 2002].

Dentre essas abordagens até hoje utilizadas, os métodos ágeis têm recebido destaque, principalmente no contexto de mudanças, oferecendo, em vários casos, respostas rápidas às novas exigências de desenvolvimento: requisitos mutáveis e não totalmente esclarecidos e entrega do produto com valor aceitável.

Entretanto, as organizações têm grandes dificuldades na implantação de métodos de desenvolvimento. A maior dificuldade encontrada num processo de implantação é de característica comportamental: a resistência à mudança. Outra dificuldade é a inevitável queda na produtividade, em um primeiro momento. Isto ocorre por representar uma mudança na maneira pela qual os profissionais estão habituados a realizar suas atividades. Enquanto a metodologia não estiver assimilada, os desenvolvedores encontrarão mais dificuldades em exercer suas tarefas do que do modo com o qual estavam acostumados.

1.1 Motivação

Este trabalho buscou uma motivação dentro de um cenário real, numa empresa do setor de desenvolvimento de softwares educativos, focada na área específica de construção de objetos de aprendizagem, caracterizando assim um estudo de caso. O processo de desenvolvimento dessa empresa conta com as equipes de produção compostas por programadores, designers, professores e coordenador de equipe; entretanto, não há uso de qualquer ferramenta ou metodologia para apoiar o gerenciamento da construção dos objetos de aprendizagem.

Assim, esta atividade de grande importância organizacional está direcionada individualmente para os coordenadores de cada equipe. Neste contexto, as atividades dos projetos estão sendo realizadas de forma não padronizada, e questões como: (i) definição da distribuição e alocação de tarefas de cada projeto; (ii) definição e seleção de competências; (iii) definição da matriz de responsabilidades e canais de comunicação, entre outras, estão sendo tratadas unicamente com base na experiência e disponibilidade de cada coordenação.

A falta de padronização para guiar a equipe acarreta numa inevitável descentralização de informação, que é extremamente prejudicial visto que a maioria das tarefas possui interdependência e essa conexão pode envolver diferentes tipos de partes interessadas: programadores, designers, professores, profissionais da sonoplastia.

A junção desses problemas implica em repassar toda a responsabilidade de geração de artefatos – de roteiro e sonoplastia, por exemplo – para a coordenação, que não dispõe de recursos e tempo suficientes para incluir a equipe na decisão de definição das tarefas. Essa situação acarreta em outro problema: instabilidade no relacionamento da equipe.

Os aspectos visivelmente constatados relativos a esse problema são a desmotivação da equipe, por não se sentir “parte do processo”; o não reconhecimento da hierarquia do coordenador pela equipe, visto que há uma real monopolização de decisões por parte da coordenação; e a forte dependência da equipe nas direções da coordenação, requerendo constantemente instruções para prosseguir com o desenvolvimento.

Outro ponto fortemente impactante na motivação da equipe é o fato do escopo do projeto atual estar em frequente alteração, de modo que muito tempo de desenvolvimento e planejamento é de fato desperdiçado ao ser criado um produto que, de acordo com levantamento da autora no ambiente de experiência, será, na maioria das vezes, 70% a 80% alterado. Esse levantamento foi feito baseado nas subdivisões do produto – nesse caso, um objeto de aprendizagem (aula) – analisando seu critério de divisão de funcionalidades para o desenvolvimento, que será tratado no capítulo 4.

1.2 Objetivos

O objetivo deste trabalho é uma avaliação de desempenho da equipe após a implantação do método ágil Scrum, a fim de comprovar sua eficácia e validar seu uso em ambientes de desenvolvimento de tecnologias educacionais.

Entre as metas específicas por este trabalho, estão:

(i) adaptar a metodologia Scrum para a realidade do ambiente de desenvolvimento de softwares educativos;

(ii) implantar o método ágil de desenvolvimento Scrum em um ambiente de desenvolvimento de softwares educativos;

(iii) aplicar um método de adaptação, contendo boas práticas e desenvolvido pela autora, à nova metodologia de forma a minimizar o impacto da sua implantação;

(iv) testar a eficiência da metodologia ágil Scrum em ambientes de desenvolvimento de softwares educativos, no que diz respeito a entrega do produto de qualidade e em tempo hábil;

(v) projetar melhorias no ambiente de desenvolvimento, tais como: comunicação direta e sem falhas; interatividade, independência e transparência na tomada de decisões entre equipe e gerência; otimização e homogeneidade do tempo de desenvolvimento da equipe de produção da empresa.

1.3 Metodologia

A pesquisa realizada no projeto é de Natureza Aplicada, objetivando gerar conhecimentos para aplicação prática dirigida a ambientes de desenvolvimento de softwares educativos. Sua abordagem será qualitativa, de modo a analisar a interação de cada indivíduo com a metodologia e com o novo ambiente de desenvolvimento a ser criado; e quantitativa, por requerer o uso de técnicas estatísticas para gerar estimativas concretas de desenvolvimento de cada tarefa e traduzir a eficiência da metodologia.

Quanto ao seu objetivo, a pesquisa é caracterizada como exploratória, fazendo uso de procedimento técnico bibliográfico e de ação.

Desta forma, contempla as seguintes etapas:

1- **Estudo geral sobre metodologias ágeis.**

Nesse sentido, a pesquisa na literatura existente sobre as metodologias ágeis em relação a (i) histórico, (ii) propósitos individuais de cada metodologia e (iii) casos de sucesso/fracasso é necessária para embasar a escolha da metodologia aplicada.

2- **Estudo aprofundado sobre a metodologia ágil escolhida para ser aplicada: Scrum.**

Esse estudo tem enfoque no histórico e propósito principal da metodologia, bem como em suas características híbridas e casos de sucesso/fracasso, visando elaborar definições sobre a estratégia de aplicação do método, usadas na etapa seguinte.

3- **Definição da adaptação da metodologia Scrum para o ambiente a ser aplicada.**

Através da elaboração da estratégia de implantação, o projeto de adaptação prevê: (i) mudanças de papéis e responsabilidades nos recursos humanos, (ii) inclusão de equipamentos e aplicativos de suporte ao uso da metodologia – na sua forma híbrida – e (iii) alteração na distribuição de tarefas pelo tempo de desenvolvimento.

4- **Implantação do método Scrum e simultânea coleta de dados necessária para análise dos resultados.**

O projeto de coleta de dados prevê que o escopo da coleta seja composto por: (i) análise comportamental (aceitação ou recusa) do uso dos recursos físicos e softwares implantados; (ii) análise de eficiência (redução do tempo, minimização de manutenção do produto e dúvidas sobre o documento de requisitos) no desenvolvimento das tarefas; e (iii) análise de comunicação e interação entre membros da equipe.

5- **Análise e transcrição dos resultados.**

Esta etapa tem a finalidade de medir as vantagens e desvantagens no uso da metodologia Scrum em ambientes de desenvolvimentos de softwares educativos, interpretadas em documentos de análise de prazos, qualidade e custos envolvidos na implantação do método.

1.4 Estrutura do Documento

Este trabalho está dividido em seis capítulos, incluindo este introdutório, que apresentou uma breve introdução ao tema, os objetivos e a metodologia aplicada.

Capítulo 2: Metodologias Ágeis em Projetos de TI

Neste capítulo será abordado todo o embasamento teórico referente às metodologias ágeis, necessário para o desenvolvimento deste trabalho. Serão tratados conceitos, algumas abordagens existentes e técnicas; e será dada ênfase aos aspectos da implantação de metodologias ágeis em empresas de TI.

Capítulo 3: Estudo da Metodologia Scrum

Este capítulo trará informações detalhadas sobre a metodologia ágil Scrum, selecionada como ferramenta para este trabalho. Serão descritas características, utilização e práticas dessa metodologia, de modo a corroborar a decisão de sua aplicação neste trabalho.

Capítulo 4: Adaptação e Implantação do Método Scrum

Este capítulo apresentará o processo utilizado na adaptação da metodologia Scrum ao ambiente de desenvolvimento onde foi aplicado este estudo. Inicia a explanação com a contextualização do problema e do ambiente onde o estudo foi realizado, mostrando a correlação entre os vários aspectos do ambiente e as estratégias de abordagem da metodologia Scrum; sendo finalizado com o levantamento dos artefatos, ferramentas e fases utilizados na aplicação da metodologia.

Capítulo 5: Coleta e Análise de Dados do Processo de Implantação do Método Scrum

Este capítulo apresentará o resultado coletado do processo de implantação do método, bem como a análise desse resultado sob os aspectos de eficiência, adaptação e aceitação da metodologia, através da observação comportamental da equipe.

Capítulo 6: Conclusão e Trabalhos Futuros

Este capítulo traz a conclusão do trabalho, as dificuldades encontradas e as possibilidades de trabalhos futuros.

Capítulo 2

Metodologias Ágeis em Projetos de TI

O objetivo deste capítulo é apresentar os fundamentos associados ao conceito de "ágil" através do histórico e criação do Manifesto Ágil, apresentar algumas abordagens ágeis e fornecer uma definição de um método ágil de desenvolvimento de software em projetos de TI.

2.1 Histórico e Origens do Manifesto Ágil

As metodologias ágeis de desenvolvimento de software foram criadas a partir da necessidade de um desenvolvimento mais flexível a mudanças e menos custoso em relação aos métodos tradicionais – caracterizados por uma pesada regulamentação, regimentação e micro gerenciamento – que despendem muito tempo em análise e planejamento.

Em fevereiro de 2001, um grupo de desenvolvedores e consultores de software, insatisfeitos com as técnicas e métodos de desenvolvimento de sistemas usados até o momento, se juntaram para compartilhar valores e princípios comuns que eram utilizados em suas práticas e criaram uma aliança que denominaram *Agile Software Development Alliance*, mais conhecida como *Agile Alliance* [SAVOINE et al., 2009].

Estes profissionais de diversas áreas de formação, com pontos de vista diferentes sobre os modelos e métodos de desenvolvimento de software em comum, publicaram um manifesto para encorajar melhores meios de desenvolvedor software [AMBLER, 2004]. Este documento, definido como o Manifesto Ágil ou *Agile Manifesto*, teve como idealizadores: Kent Beck, Ken Schwaber e Martin Fowler, e tem como foco um conjunto de princípios, que definem critérios para o processo de desenvolvimento de software ágil, que seguem:

1. **Indivíduos e iterações acima de processos e ferramentas**

Com esse princípio, o movimento ágil enfatiza que a comunidade de desenvolvedores de software e seu relacionamento, bem como o aspecto humano, repercutem nos contratos, em oposição aos processos institucionalizados e ferramentas de desenvolvimento. Nas práticas ágeis existentes, isso se manifesta nas relações internas da equipe, nos acordos de trabalho, de ambiente e outros procedimentos, visando aumentar o espírito de equipe [ABRAHAMSSON et al., 2002].

2. **Software em funcionamento acima de documentação abrangente**

O objetivo primordial da equipe de desenvolvimento de software é continuamente entregar *releases* testados. Novos *releases* são produzidos em intervalos frequentes – em algumas abordagens, diário ou por hora –, geralmente mensal ou bimestral. Os desenvolvedores são aconselhados a manter o código simples, direto e tecnicamente o mais avançado possível, diminuindo a carga de documentação para um nível adequado.

3. **Colaboração com o cliente acima de negociação de contratos**

Nesse aspecto o movimento ágil enfatiza que ao relacionamento e à cooperação entre desenvolvedores e clientes é dada a preferência sobre os contratos restritivos; embora seja declarada a importância de contratos bem elaborados. O processo de negociação em si deve ser visto como um meio de atingir e manter uma relação viável. Do ponto de vista comercial, o desenvolvimento ágil está focado em fornecer valor ao negócio imediatamente no início do projeto, reduzindo assim os riscos de não cumprimento do contrato.

4. **Responder a mudanças acima de seguir um plano**

Segundo Highsmith e Cockburn [2001, p. 122], “o que há de novo a respeito dos métodos ágeis não são as práticas que usam, mas o reconhecimento das pessoas como os principais impulsionadores do sucesso do projeto, juntamente com um intenso foco na eficácia e capacidade de manobra. Isso gera uma nova combinação de valores e princípios que definem uma visão de mundo ágil.”.

Cockburn [2002, p. xxii] define o núcleo dos métodos ágeis de desenvolvimento de software como o uso de regras “leves-mas-suficientes” para o

comportamento do projeto. O processo é ágil leve e suficiente. “Leveza é um meio de permanecer manobrável. Suficiência é uma questão de ficar no jogo” [ABRAHAMSSON et al., 2002].

Os princípios básicos de métodos ágeis compreendem honestidade ao código de trabalho, eficácia das pessoas que trabalham em conjunto e foco no trabalho em equipe. Assim, o grupo de desenvolvimento, incluindo desenvolvedores e representantes do cliente, deve ser bem informado, competente e autorizado a considerar o eventual ajuste das necessidades emergentes durante o processo de ciclo de vida do desenvolvimento. Isto significa que os participantes estão preparados para fazer mudanças, e que também os contratos existentes são formados com as ferramentas que suportam e permitem que essas melhorias sejam feitas.

2.2 Abordagens de Metodologias Ágeis

A partir daí, surgiram várias metodologias que seguem estes valores e princípios. Algumas abordam a questão da Gerência de Projetos, como é o caso do XP, Scrum, Crystal family of methodologies, entre vários outros.

2.2.1 XP – *EXtreme Programming*

Os fundamentos da metodologia EXtreme Programming (XP) tiveram início na década de 80, baseados no desenvolvimento em SmallTalk, que praticava *pair programming*, *refactoring*, testes constantes e desenvolvimento iterativo, atualmente utilizados nas práticas da XP.

A metodologia evoluiu a partir dos problemas causados pelos longos ciclos de desenvolvimento dos modelos tradicionais de desenvolvimento [BECK, 1999a]. De acordo com Haungs [2001], iniciou como “simplesmente uma oportunidade para começar o trabalho feito”, com práticas que haviam sido eficazes no desenvolvimento de processos de software durante as décadas anteriores. Mesmo que as práticas individuais de XP não sejam novas, elas foram alinhadas para funcionar em uma maneira nova; e o termo *extreme* se deu pelo tratamento dessas práticas aos níveis extremos [BECK, 1999b].

Segundo Beck [2004], XP é uma metodologia leve, eficiente, de baixo risco e flexível, e para equipes de tamanho pequeno a médio. Além disso, possui como principal característica ser adaptativa, usada principalmente no desenvolvimento baseado em requisitos incompletos e mutáveis ao longo do ciclo de vida do projeto, sendo esse aspecto da metodologia encorajador para equipe a enfrentar as mudanças como algo natural.

A satisfação do cliente é uma preocupação constante em XP, visto que o objetivo principal da metodologia é a entrega do produto de qualidade, em tempo hábil e que satisfaça as exigências e expectativas do cliente e da equipe de desenvolvimento. Para isto, a metodologia XP segue quatro princípios de fundamental importância para seu embasamento: comunicação, *feedback*, simplicidade e coragem. Esses princípios estão representados em algumas práticas utilizadas de forma harmônica e complementar para a entrega de um software de alta qualidade, que são mostradas na Tabela 2.1, apontando seus pontos fortes e fracos [SAVOINE et al., 2009].

Tabela 2.1 Práticas e características da metodologia ágil XP. Fonte: adaptado de Savoine et al. [2009]

Pontos fortes	Pontos fracos
Reuniões diárias, quando se tiram dúvidas de estórias complexas e planejam-se iterações do dia.	Não indicada para sistemas complexos, pois a análise não é detalhada.
Indicada quando o escopo do projeto não é bem definido: os requisitos não são totalmente conhecidos ou muito vagos.	Não indicada para equipes muito grandes, pois o escopo tem que ser definido e os membros da equipe, geralmente, ficam separados geograficamente.
Clientes presentes caracterizam a chave para o sucesso.	Não indicada em ambientes onde os clientes não podem estar presentes.

Possui foco na codificação, sendo a documentação deixada para segundo plano.	Não possui foco na gerência de projeto, sendo os resultados o principal interesse da equipe.
Flexível a mudanças, encorajando as pessoas a enfrentarem-na com naturalidade.	Não indicada em projetos que o cliente exige extensa documentação de software.
Tarefas com simplicidade e qualidade, funcionalidades realmente úteis ao usuário.	Não indicada em equipes que possuem resistências a práticas como *refactoring*, etc.
Iterações curtas, com *feedback* do usuário rápido e constante.	Não indicada em projetos em que o cliente não possa participar integralmente do desenvolvimento.

Entretanto, como afirmado por Beck [1999b], a metodologia XP não é apropriada para uso em qualquer ambiente, nem todos os seus limites ainda foram identificados. Essas definições exigem mais pesquisas empíricas e experimentais a partir de perspectivas diferentes.

Um dos limites identificados no XP é o fato de ser voltado para pequenas e médias equipes. Além disso, Beck [1999b] sugere que o ambiente físico também é importante em XP, visto que a comunicação e a coordenação entre os membros do projeto devem ser ativadas em todos os momentos. Menciona também que uma dispersão de programadores em dois andares ou até mesmo em um piso é intolerável para o XP.

Essas limitações do XP foram críticas na análise da sua adaptabilidade ao estudo de caso, visto que os membros da equipe de desenvolvimento não somente estão fisicamente, mas periodicamente, dispostos de forma dispersa, caracterizando quebra numa premissa na implantação do método. Logo, a metodologia XP não se mostrou ser a mais adequada para a solução do problema.

2.2.2 *Crystal family of methodologies*

A família Crystal de metodologias inclui uma série de metodologias diferentes para a seleção da mais adequada para cada projeto. Além das metodologias, a abordagem Crystal também inclui princípios para adaptar as metodologias às variáveis circunstâncias de diferentes projetos.

Existem certas regras, características e valores que são comuns a todos os métodos da família Crystal. Em primeiro lugar, os projetos sempre usam ciclos de desenvolvimento incremental, com um incremento máximo de 4 meses (mais comumente entre 1 e 3 meses) [COCKBURN, 2002]. Na sequência está a ênfase na comunicação e cooperação das pessoas. Além disso, as metodologias Crystal não limitam as práticas ou ferramentas de desenvolvimento, e permitem a incorporação de práticas de outras metodologias, como por exemplo, XP e Scrum [COCKBURN, 2002].

Atualmente existem três principais metodologias Crystal construídas: Crystal Clear, Crystal Orange e Crystal Orange Web. A Crystal Clear foi projetada para projetos muito pequenos, incluindo até 6 colaboradores. Uma equipe com Crystal Clear deve estar localizada em espaço físico comum, devido às limitações na sua estrutura de comunicação [COCKBURN, 2002]. Os Crystal Orange e Crystal Orange Web foram desenvolvidos para projetos de média dimensão, com um total de 10 a 40 membros na equipe de desenvolvimento e duração de 1 a 2 anos. Nesta metodologia, o projeto é dividido por várias equipes com grupos multifuncionais. Ainda assim, como o Crystal Clear, este método não tem suporte ao ambiente de desenvolvimento distribuído.

A diferença principal entre as metodologias Crystal trata-se da entrega: a Crystal Clear sugere entrega incremental num prazo 2 a 3 meses, no máximo, enquanto na Crystal Orange os incrementos podem ser estendidos para 4 meses, no máximo.

A principal vantagem adaptativa dessas metodologias é o fato de suas normas de política poderem ser substituídas por práticas equivalentes de outras metodologias, como XP ou Scrum [COCKBURN, 2002].

Bem como a metodologia XP, as limitações de espaço físico e horário de trabalho – devido à restrição na sua estrutura de comunicação – são impeditivos

para a implantação das metodologias Crystal. Além desse aspecto, os incrementos possuem menor periodicidade que o necessário para o estudo, que exige *releases* a cada 1 ou 2 semanas, no máximo. Sendo assim, as metodologias Crystal também se mostraram inadequadas para a solução do problema.

2.2.3 Scrum

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, "resultando em uma abordagem que reintroduz as ideias de flexibilidade, adaptabilidade e produtividade" [SCHWABER & BEEDLE, 2002].

A metodologia foi formalmente apresentada e publicada no OOPSLA - *The International Conference on Object Oriented Programming*, *Systems, Languages and Applications* - em 1995, por Jeff Sutherland e Ken Schwaber, trazendo alterações da metodologia elaborada por Jeff McKenna, Mike Smith e Chris Martin. Durante os cinco anos seguintes, Mike Beadle e Martine Devos fizeram contribuições significativas. Desde então tem sido usada para desenvolver produtos complexos.

A metodologia Scrum não é um processo ou uma técnica para a construção de produtos, mas sim um *framework* em que é possível empregar vários processos e técnicas [SCHWABER & SUTHERLAND, 2009]. É uma metodologia com o foco no gerenciamento da equipe, preocupada na organização dos processos, no modo como as atividades devem ser executadas, deixando a cargo dos participantes do projeto escolher a melhor maneira de concluir com sucesso as etapas do desenvolvimento, como resumido na Tabela 2.2 [SAVOINE et al., 2009].

Tabela 2.2 Práticas e características da metodologia ágil Scrum. Fonte: adaptado de Savoine et al. [2009]

Pontos fortes	Pontos fracos
Indicada quando não se tem todos os requisitos do sistema ou estes não estão totalmente definidos.	Não indicada para sistemas complexos, pois a análise não é detalhada.
Flexibilidade para mudanças constantes durante as fases do projeto.	Não indicada para projetos que exigem vasta documentação.
O cliente faz parte da equipe em tempo integral.	Não indicada quando os clientes não tem disponibilidade de tempo ou são resistentes a troca de informações.
Reuniões diárias que guiam o andamento do projeto.	Não indicada para equipes resistentes a reuniões frequentes.
Ciclos de desenvolvimento curtos e constantes.	Não indicada para projetos que necessitem de ciclos de iterações longos.
Comunicação entre os membros da equipe é frequente.	Não indicada em ambientes onde as tarefas são impostas às pessoas.
Revisões das funcionalidades realizadas acontecem no final de cada ciclo.	Não indicada em projetos que o líder não divide responsabilidades com a equipe.

O desenvolvimento é dividido em iterações, chamadas de *sprints*, de duração entre 2 e 4 semanas, no máximo. Nesse sentido, o Scrum emprega o conceito de atividades *time-boxes*, em que cada uma tem seu tempo delimitado, não podendo ser estendido, criando regularidade no ciclo de produção. Assim, também é possível um melhor controle do *release* em relação à qualidade e as funcionalidades são mais bem distribuídas por tempo de desenvolvimento.

A metodologia Scrum pode ser usada em equipes pequenas e grandes, sendo o tamanho ideal entre 5 e 9 pessoas. Entretanto, quando há menos de cinco membros da equipe há menos interação e, como resultado, menor ganho de produtividade. Além disso, a equipe pode encontrar dificuldades de habilidade durante as partes da *sprint* e ser incapaz de entregar um *release* do produto. Se houver mais de nove membros é requerida muita coordenação: grandes equipes geram muita complexidade para um processo empírico administrar. Nesse caso, é recomendado subdividir a equipe [SCHWABER & SUTHERLAND, 2009].

Diferentemente das outras metodologias abordadas, Scrum não possui as limitações de espaço físico, sendo esse impeditivo solucionado com reuniões diárias e as ferramentas próprias – entre elas, o Scrum *Board* – para integrar as atividades da *sprint*.

Pode-se perceber que essa metodologia possui muitas das características do XP, mas não possui restrições quanto à localização geográfica da equipe, por exemplo, fator este impactante na empresa onde será aplicada a metodologia; além de ser melhor aplicada com equipes ainda menores que o delimitado pelo XP e pelas metodologias Crystal, se adaptando melhor, então, ao caso estudado. Por essas razões, foi escolhido como objeto desse estudo.

2.3 Metodologias Ágeis no Mercado de TI

As metodologias ágeis têm despertado o interesse do mercado, apresentando evidências de melhoria na produtividade. Entretanto, para que possam ser efetivamente usadas em larga escala, precisam provar alguns de seus pontos de vista.

2.3.1 Aspectos da Implantação em Empresas de TI

A escolha da metodologia mais adequada para o desenvolvimento de software em uma organização não é uma tarefa trivial. Vários estudos como os de Bona [2002], Nonemacher [2003] e Harrison [2005] foram utilizados para a avaliação de uma metodologia de desenvolvimento; entretanto, a questão que será sempre levantada por qualquer organização que estude a adoção de uma nova metodologia é a quantidade de empresas que já a adotam com sucesso. Infelizmente, na

comunidade de desenvolvimento de software, pouco há de determinado estatisticamente, sendo necessário o uso desses estudos, mesmo conduzidos informalmente, para verificar as tendências do mercado.

Segundo Grinyer [2007], apesar do volume crescente de informação formada pelas diversas pesquisas empíricas já publicadas, há um consenso de que as organizações ainda tomam pouco conhecimento delas. Com respeito às metodologias ágeis, Jeffries [2005, apud GRINYER, 2007] afirma que os profissionais formam a maior parte das suas ideias pelo contato direto com outros profissionais, pelo que possam ler rapidamente e, principalmente, pela prática [BANKI & TANAKA, 2009].

Em pesquisa realizada pela empresa DigitalFocus, e apresentada no evento Agile 2006, foram coletados dados de 136 profissionais de 128 empresas de diferentes portes a fim identificar os principais aspectos envolvidos na adoção das metodologias ágeis, sob o ponto de vista técnico e gerencial. Como resultado, pôde-se observar o aumento do interesse das empresas de TI nas metodologias de desenvolvimento ágil, com 81% das empresas adotando uma metodologia ágil ou procurando por uma oportunidade para fazê-lo. As principais colocações entre os aspectos levantados estão resumidas na Tabela 2.3.

Tabela 2.3 Pesquisa sobre a aceitação das metodologias ágeis. Fonte: Projects@Work [2006]

Item da pesquisa	Resultado
Percentual de empresas que adotam metodologias ágeis em todos os projetos.	46%, nas empresas de porte médio, e 12%, nas empresas de grande porte.
Percentual de empresas que adotam metodologias ágeis em algum projeto.	44%
Principal barreira para a adoção do desenvolvimento ágil.	Falta de conhecimento (51% entre os desenvolvedores, 56% entre os executivos).

Principal motivação para a adoção do desenvolvimento ágil (para os executivos).	Melhorar a produtividade e previsibilidade no desenvolvimento de software (51%).
Principal motivação para a adoção do desenvolvimento ágil (para os desenvolvedores).	Auxiliar no gerenciamento do escopo dos projetos (47%).

Em estudo feito por Grinyer [2007], baseando-se na análise da literatura disponível e em consultas feitas a três listas de discussão populares na comunidade ágil, foram coletados 41 relatos relacionando 53 fatores que levaram à adoção de uma metodologia ágil. Esses fatores foram organizados em 7 categorias, apresentadas na Tabela 2.4.

Tabela 2.4 Fatores de adoção de uma metodologia ágil. Fonte: adaptado de Grinyer [2007]

Categoria	Número de fatores
1-Melhoria no processo de desenvolvimento, como resposta a repetidos fracassos em projetos.	15
2-Influências internas na organização (gerente, desenvolvedor sênior ou arquiteto).	11
3-Fator competitivo (prazo para colocação do produto no mercado).	10
4-Influências externas à organização (treinamento ou consultoria externa).	8
5-Resposta a requisitos em constante variação.	5

6-Acompanhamento de tendências tecnológicas e do mercado.	2
7-*Downsizing* da equipe e processo de desenvolvimento.	2
Total	**53**

Grinyer [2007] menciona no estudo que em nenhum relato foi reportada a aplicação de uma metodologia exatamente como descrita, sempre sendo feita alguma adaptação ao contexto no qual foi aplicada. É relevante também apontar que nenhum dos relatos indicou os motivos pelos quais as empresas que adotaram uma nova metodologia acreditaram que essa mudança efetivamente levaria à solução dos fatores apontados.

Além desse levantamento documental, deve-se destacar a pesquisa conduzida pela VersionOne e apoiada pela Agile Alliance, a qual atingiu mais de 700 profissionais, com a mesma intenção de determinar como os processos ágeis têm sido implementados nas diversas organizações. Dos 722 profissionais pesquisados, mais de 25% vinham de empresas com mais de 250 pessoas, mas apenas 18% deles trabalhavam em equipes de mais de 10 pessoas. Esses números comprovam que, embora o desenvolvimento ágil possa ser adaptado para empresas de maior porte, a grande maioria dos projetos é desenvolvida por pequenas equipes [BARNETT, 2006]. Essa pesquisa aponta que o desenvolvimento ágil tem garantido um significante retorno sobre o investimento (ROI) para as organizações que o adotam. Analisando os fatores que levaram à adoção de uma metodologia ágil, foram identificados os fatores presentes na Figura 2.1.

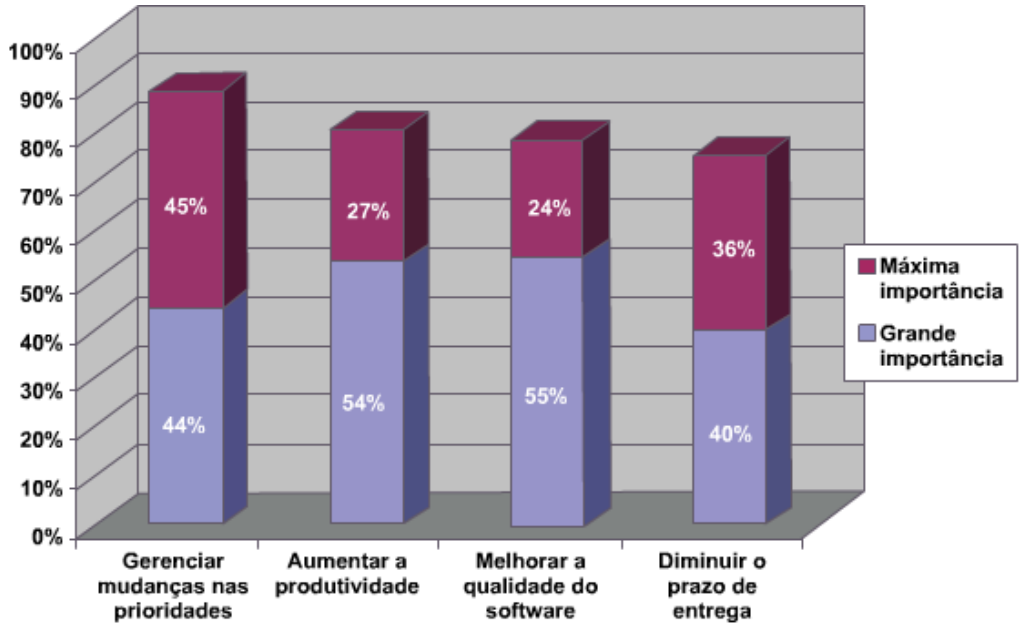

Figura 2.1 Fatores de adoção de uma metodologia ágil. Fonte: adaptado de Barnett [2006]

Comparando-se resultados apresentados na Figura 2.1 com os fatores resumidos na Tabela 2.4, é possível observar o aumento da produtividade e da qualidade do software, associado ao gerenciamento de requisitos em constante variação, como sendo o principal motivador da adoção de uma metodologia ágil de desenvolvimento.

Com relação às barreiras para a adoção do desenvolvimento ágil, Barnett [2006] comenta que, no início do Movimento Ágil, o principal motivo citado era a falta de apoio por parte das gerências e da organização como um todo. Entretanto, a falta de profissionais qualificados para o desenvolvimento ágil e a resistência dos próprios desenvolvedores à mudança são agora apontados como os principais fatores, conforme observado na Figura 2.2 [BANKI & TANAKA, 2009].

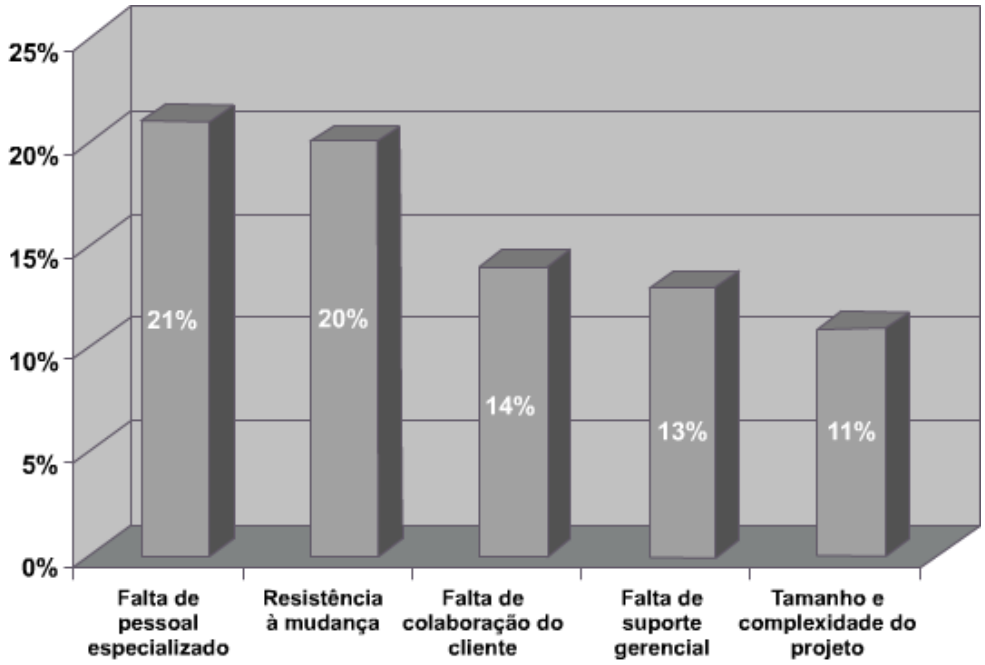

Figura 2.2 Barreiras para adoção do desenvolvimento ágil. Fonte: adaptado de Barnett [2006]

Segundo Barnett [2006] a comunidade técnica precisa ser capaz de demonstrar quantitativamente os benefícios atingidos pela adoção do desenvolvimento ágil, a fim de convencer as empresas mais tradicionais a fazer a mesma mudança. Os resultados apresentados na Figura 2.3 mostram que os desenvolvedores que adotam as metodologias ágeis estão bastante satisfeitos com os resultados que têm encontrado [BANKI & TANAKA, 2009].

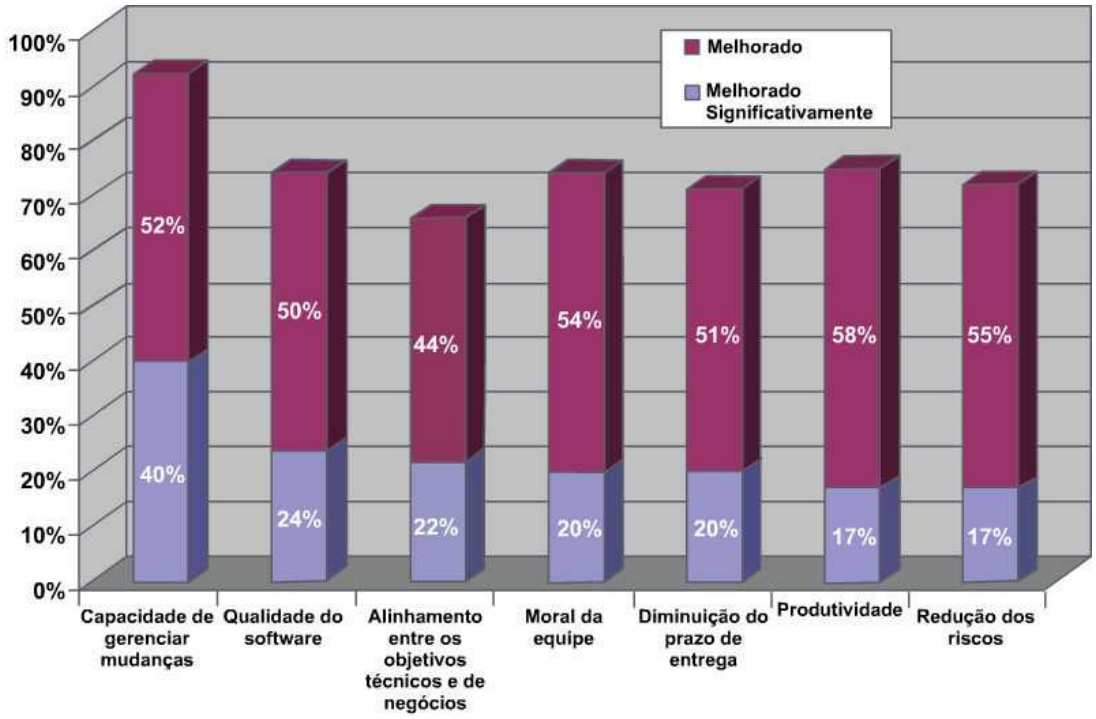

Figura 2.3 Valores obtidos pela adoção de uma metodologia ágil. Fonte: adaptado de Barnett [2006]

Pode-se concluir que o desenvolvimento ágil está sendo adotado por um número crescente de empresas, permitindo que um número maior de equipes possa quantificar seus resultados. É esperado, para um futuro próximo, que informações mais consistentes sejam levantadas e apresentadas à comunidade de desenvolvimento.

2.4 Resumo do Capítulo

Neste capítulo foi definido o conceito de agilidade e apresentado o histórico do Manifesto Ágil, bem como seus princípios, visando fornecer uma visão teórica da metodologia utilizada nesse trabalho. Além disso, foram mostradas três abordagens de metodologias ágeis – XP, *Crystal family of methodologies* e Scrum – que foram avaliadas para serem empregadas nesse estudo, sendo justificados o seu uso ou não. Por fim, foi apresentado o cenário das metodologias ágeis em empresas de TI, levantando, através de pesquisas elaboradas por empresas conceituadas, os aspectos de implantação de metodologias ágeis nessas empresas.

Capítulo 3

Estudo da Metodologia Scrum

O objetivo desta seção é caracterizar a metodologia Scrum; apresentar seu *framework* visando fornecer a informações suficientes para justificar a escolha de sua utilização como metodologia adequada para desenvolvimento de softwares educativos.

3.1 Origem e Conceitos

As primeiras referências na literatura ao termo “Scrum” apontam para o artigo de Takeuchi e Nonaka [1986], *The New Product Development Game*, um estudo de caso da indústria de computadores e impressoras e automobilística [TAKEUCHI & NONAKA, 1986] que relata o processo de desenvolvimento adaptativo, rápido e auto-organizado iniciado por dez empresas inovadoras japonesas. O termo “Scrum” – que deriva de uma estratégia no jogo de rúgbi – foi introduzido para definir práticas adaptativas utilizadas em times auto-gerenciáveis.

Após a introdução do conceito japonês, Jeff Sutherland e Ken Schwaber, em 1994, se reuniram para formalizar, refinar e implantar o processo Scrum dentro da sua organização, produzindo o artigo *The Scrum Development Process*, primeira referência formal ao processo ágil de desenvolvimento de software. Em 2001, Ken Schwaber e Mike Beedle editaram o primeiro livro sobre Scrum e, em 2004, foi lançada uma nova apresentação da metodologia.

Assim, a metodologia Scrum pôde ser formalmente definida como um processo de gerenciamento de projetos que pode ser utilizado em diversas áreas, se concentrando no atendimento às necessidades do negócio [SCHWABER & BEEDLE, 2002].

Pode ser considerado mais como um *framework* que uma metodologia, cujas práticas são aplicadas em um processo empírico iterativo e incremental, para desenvolvimento de qualquer produto e gerenciamento de qualquer trabalho. A sua aplicação provê a agilidade necessária para desenvolver projetos adaptados às

novas realidades organizacionais em ambientes de constante mudança. Além disso, Scrum foca-se, a partir de princípios herdados de *Lean* [POPPENDIECK & POPPENDIECK T., 2003], na ampla comunicação da equipe e cooperação dos envolvidos no projeto.

É suportado por três pilares: transparência, inspeção e adaptação. A **transparência** assegura que os aspectos do processo que afetam o resultado devem ser visíveis para aqueles que administram os resultados [SCHWABER & SUTHERLAND, 2009].

Os vários aspectos do processo devem ser **inspecionados** com frequência suficiente para que as variações inaceitáveis no processo possam ser detectadas. A frequência de inspeção tem que levar em consideração que todos os processos são alterados pelo ato da inspeção [SCHWABER & SUTHERLAND, 2009].

Assim, se o inspetor determina que um ou mais aspectos do processo estejam fora dos limites aceitáveis, e que o produto resultante será inaceitável, o inspetor deve ajustar o processo ou o material a ser processado, e essa **adaptação** deve ser feita o mais rápido possível [SCHWABER & SUTHERLAND, 2009]. Não há especificações quanto às práticas de construção; as equipes são auto-organizadas e o processo é orientado a objetivos e resultados [SCHWABER & BEEDLE, 2002]. Decisões de como usá-la e criação de estratégias para obter produtividade e realizar entrega de artefatos, ficam por conta de quem está aplicando o processo [SCHWABER, 2004].

Há três pontos de inspeção e adaptação em Scrum: a *Daily* Scrum *Meeting* é usada para inspecionar o progresso em direção à meta da *sprint* e fazer as adaptações de modo a otimizar o valor do dia de trabalho seguinte. Além desta, a *Sprint Planning Meeting* e a *Sprint Review* são usadas para inspecionar o progresso em direção à meta do *release* e para fazer as adaptações que aperfeiçoam o valor da próxima *sprint*. Por fim, a *Sprint Retrospective* é utilizada para analisar a *sprint* passada e determinar adaptações para a *sprint* seguinte ser mais produtiva, gratificante e agradável.

3.2 Framework

Nesta seção serão detalhados os papéis e responsabilidades da metodologia Scrum, bem como seu ciclo desenvolvimento, os elementos *time-boxed* e os artefatos e ferramentas que compõem esse ciclo.

3.2.1 Papéis e Responsabilidades

O Scrum possui três papéis: o Scrum *Master*, o *Product Owner* e o time. Os membros do Scrum são chamados de "porcos" (*pigs*); todo o resto é "galinha" (*chicken*). *Chickens* não podem dizer aos *pigs* como fazer seu trabalho, ou seja, não é permitida interferência externa no Scrum. Essas denominações vêm da história mostrada na Figura 3.1, que versa sobre envolvimento e comprometimento no negócio.

Figura 3.1 História sobre o comprometimento do Scrum *Team*.

Fonte: Vizdos [2006]

Cada membro do Scrum assume uma responsabilidade importante no desenvolvimento do produto, conforme apresentado a seguir:

Product Owner: É o único responsável por definir as características do produto e a prioridade de execução dos requisitos, de acordo com o valor para o negócio. Gerencia o ROI, garantindo a lucratividade do produto ao aceitar/recusar os resultados do trabalho desenvolvido; além de garantir que os especialistas de domínio estejam disponíveis para o time. Para o *Product Owner* para ter sucesso, todos na organização têm que respeitar suas decisões. Ninguém está autorizado a

dizer à equipe para trabalhar a partir de um conjunto diferente de prioridades, e as equipes não estão autorizados a escutar qualquer um que diga o contrário [SCHWABER & SUTHERLAND, 2009]. Por fim, é o responsável por concentrar as informações vindas de usuários, *stakeholders* ou do mercado de maneira de modo a se obter uma visão única dos requisitos do sistema [SCHWABER & BEEDLE, 2002].

Scrum *Master*: Ele é responsável pelo gerenciamento do projeto, devendo garantir que o trabalho da equipe seja funcional e produtivo, sempre acompanhando o desenvolvimento e removendo os impedimentos; além de garantir o uso do Scrum de maneira correta e participar de todas as reuniões. Trabalha próximo ao *Product Owner*, gerenciando seus os interesses mediante a equipe a fim de maximizar o ROI. O Scrum *Master* é o facilitador e o mediador da equipe, responsável por habilitar a cooperação entre todos os papéis e funções, e funciona como um escudo para o time das interferências externas (*chickens*).

Scrum *Team*: É a equipe auto-organizada, pois a organização é feita de forma participativa, auto-gerenciada e multifuncional. É composta por 5 a 9 pessoas, sem os papéis tradicionais da Engenharia de Software, como programadores, analistas de qualidade, engenheiros de software, executores de testes, etc., sendo todos os membros responsáveis por atingir juntos os objetivos definidos em cada *sprint* [SCHWABER, 2004]. É responsável por selecionar os itens priorizados que irão ser executados em cada iteração, com total liberdade para cumprir seus objetivos e por demonstrar o trabalho desenvolvido ao *Product Owner*.

3.2.2 *Time-Boxes*

Scrum utiliza “caixas de tempo” (*time-boxes*) para criar regularidade no processo de desenvolvimento. Os elementos, ou etapas, de Scrum que são *time-boxed* compreendem o *Release Planning Meeting*, a *Sprint Planning Meeting*, a *Sprint*, a *Daily* Scrum *Meeting*, a *Sprint Review* e a *Sprint Retrospective*. O núcleo do Scrum é a *sprint*, que é uma iteração de um mês ou menos – geralmente entre 2 a 4 semanas. Todas as *sprints* usam o *framework* Scrum e devem entregar um incremento do produto final que é potencialmente entregável. Uma *sprint* começa imediatamente após outra.

Em seguida será detalhado o ciclo de desenvolvimento Scrum e o uso dos elementos *time-boxed*, bem como seus artefatos.

3.2.3 Ciclo de Desenvolvimento

Em Scrum, projetos progridem numa série de iterações de 2 a 4 semanas chamadas de *sprints*, focadas na entrega incremental de um produto. O ciclo de desenvolvimento do Scrum começa com uma visão do produto que será desenvolvido, chamado de *Product Backlog*, que contém as características definidas pelo *Product Owner*, assim como as tecnologias necessárias para sua implementação. A equipe de desenvolvimento se reúne com *Product Owner* antes do início de cada *sprint* para priorizar o trabalho a ser feito e selecionar as tarefas que a equipe pode terminar na *sprint*; as tarefas selecionadas compõem o *Sprint Backlog*. Nessa fase também são definidas as ferramentas a serem utilizadas e detectados os possíveis impedimentos. A Figura 3.2 ilustra, de forma resumida, o ciclo de desenvolvimento do Scrum.

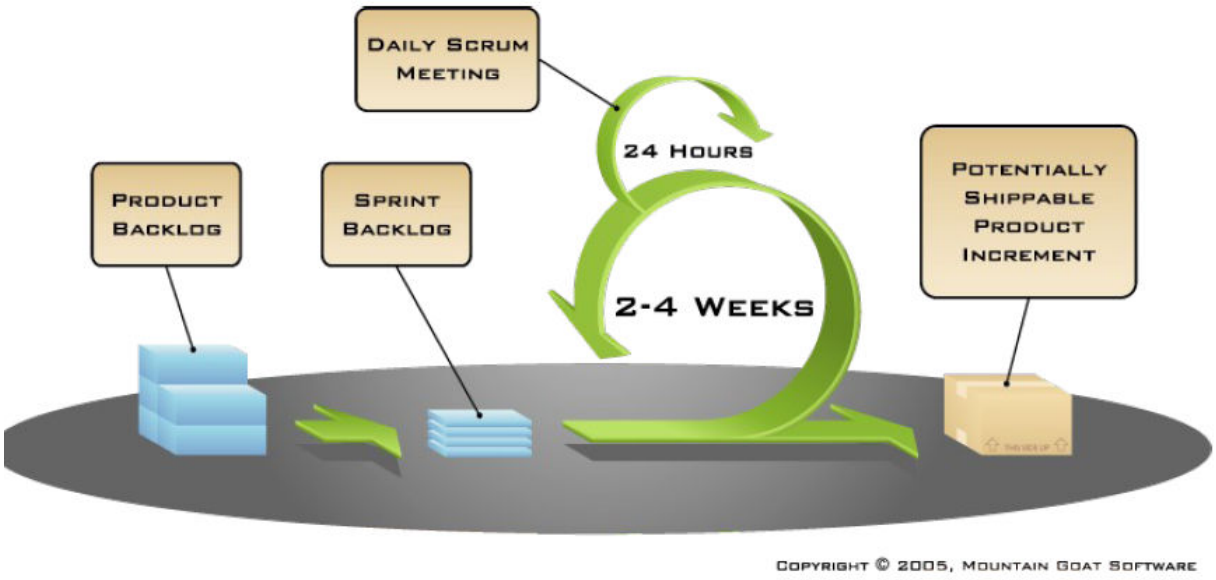

Figura 3.2 Ciclo de desenvolvimento em Scrum. Fonte: www.mountaingoatsoftware.com/scrum [2010]

Durante a execução de cada *sprint*, a equipe realiza reuniões diárias de aproximadamente 15 minutos de duração denominadas *Daily* Scrum *Meeting*, com o

objetivo de acompanhar o andamento do projeto. Ao final cada iteração é realizada uma reunião de revisão, chamada de *Sprint Review Meeting*, para que o time apresente um incremento potencialmente entregável do produto ao *Product Owner*.

Em seguida, o Scrum *Master* realiza uma reunião com a equipe chamada de *Sprint Retrospective Meeting*, como ilustra a Figura 3.3.

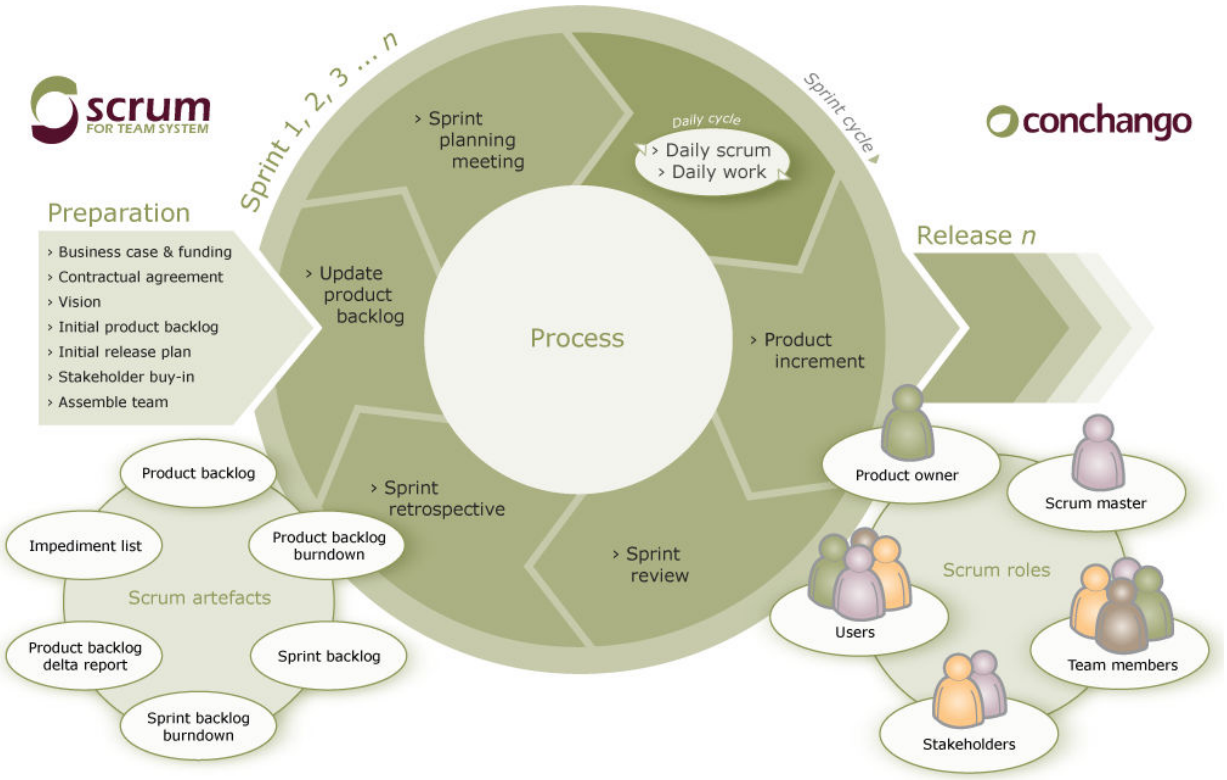

Figura 3.3 Ciclo de desenvolvimento em Scrum. Fonte: http://consultingblogs.emc.com/colinbird/ [2010]

A *Sprint Retrospective Meeting* tem o objetivo de analisar a execução do progresso do projeto e a versão do produto gerada, visando melhorar a equipe, o processo ou o produto para a *sprint* seguinte.

3.2.4 Artefatos e Ferramentas

Como citado anteriormente, dentre os artefatos do Scrum estão o *Product Backlog* e o *Sprint Backlog* e, além destes, também são usados como ferramentas o gráfico *Burndown* e o método *Planning Poker*, mostrados em seguida.

O ***Product Backlog*** trata-se de uma lista de itens priorizados elencando o que deve ser desenvolvido na *sprint*, associada a um valor de negócio, *e* que pode ser composta de requisitos funcionais ou não. Desse modo, o *Product Backlog* permite controle e gerenciamento do processo.

O *Product Owner* tem a função de manter a listagem do *Product Backlog* com suas prioridades atualizadas e sempre visíveis à equipe. Da mesma forma que o valor de negócio de cada tarefa é de responsabilidade do *Product Owner*, a complexidade das mesmas é definida pela equipe.

O ***Sprint Backlog*** é composto por uma lista de tarefas extraídas do *Product Backlog*, com as quais a equipe se compromete a fazer durante uma *sprint*, ou seja, a equipe determina a quantidade de itens do *Product Backlog* que serão executados, pois ela estará responsável em se comprometer por sua implementação e entrega de suas respectivas funcionalidades [SCHWABER, 2004]. As tarefas devem ser divididas de modo que cada uma tenha de 4 a 16 horas de atividade para sua finalização e somente a equipe pode alterar o *Sprint Backlog*, não sendo aceita a intervenção de clientes, gerentes, usuários (*chickens*) durante o curto tempo de uma *sprint*.

O **gráfico *Burndown*** (*Burndown Chart*) contribui para proporcionar visibilidade do progresso da equipe e do trabalho restante ao longo do tempo, mostrando a correlação entre a quantidade de trabalho restante em qualquer ponto no tempo e o progresso das equipes na redução deste trabalho. É formado por duas retas: uma de tendência, que representa uma iteração ideal onde o trabalho é concluído de forma uniforme e estável, e a reta real, que representa o trabalho realizado pela equipe ao longo do tempo estipulado. A reta real decresce conforme as tarefas vão sendo finalizadas ao longo da *sprint* [SCHWABER, 2004], como mostrado na Figura 3.4.

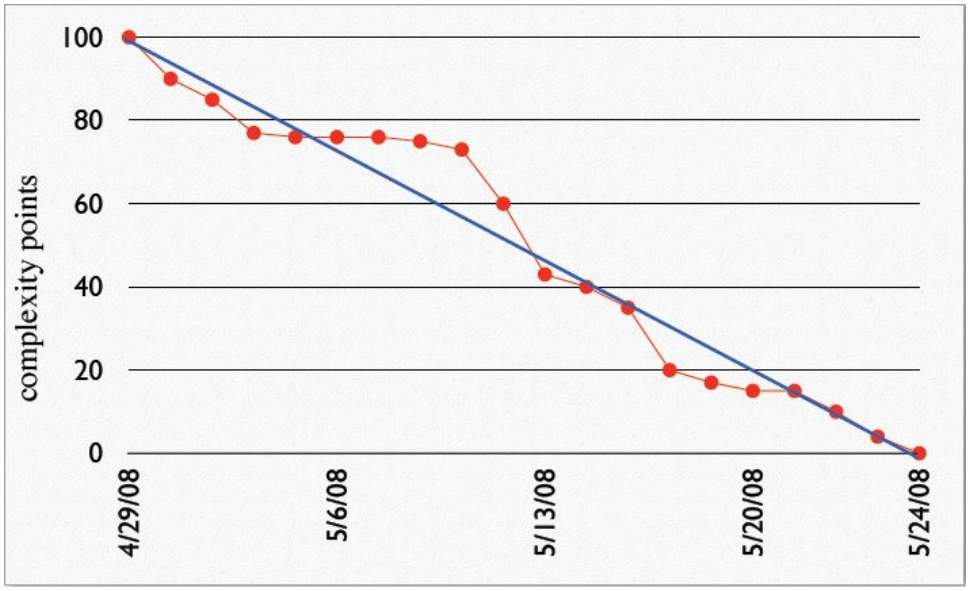

Figura 3.4 Exemplo de Scrum *Burndown Chart*

Para montar o *Burndown Chart* é utilizado o método de ***Planning Poker***. Esse método foi primeiramente descrito por James Grenning [2002] e popularizado nas metodologias ágeis através de Mike Conh no seu livro *Agile Estimating and Planning* [COHN, 2005]. O *Planning Poker* não é nativo de Scrum, mas é largamente utilizado no seu processo de estimativa, que utiliza cartas de baralho que somente são exibidas após a descrição da atividade a ser executada. Usualmente se utiliza a escala de Fibonacci (1, 2, 3, 5, 8, 13, 21,...) para a avaliação de complexidade; assim, o valor quantitativo escolhido para um item através da escala é diretamente proporcional a sua complexidade.

No *Planning Poker* cada tarefa é discutida de modo sucinto. Cada participante dá sua nota de complexidade com base na escala definida para cada tarefa e, caso haja consenso, a complexidade é validada e atribuída à tarefa. Se houver divergência se abre espaço para discussão com as pontas de maior e menor valor e são reapresentadas as escolhas do valor de complexidade da equipe após a discussão. Caso ainda não haja consenso o Scrum *Master* tem a responsabilidade de intervir [BENTZEN, 2009].

3.2.5 Etapas da *Sprint*

Os artefatos detalhados na seção anterior são utilizados nas etapas da *sprint*, também supracitadas, que compreendem o *Release Planning Meeting*, a *Sprint Planning Meeting*, a *Daily* Scrum *Meeting*, a *Sprint Review Meeting* e a *Sprint Retrospective*, explicados nessa seção.

O ***Release Planning Meeting*** é uma reunião cujo objetivo é estabelecer plano e metas que a equipe Scrum e o resto das organizações possam compreender e se comunicar. São respondidas as seguintes questões que guiam a reunião: “Como podemos transformar essa visão em um produto vencedor da melhor maneira possível?” e “Como podemos atender ou exceder a satisfação do cliente e o Retorno sobre o Investimento?”. Também estabelece uma data provável de entrega e custo que deve manter, se nada mudar. Entretanto, é totalmente opcional.

Em seguida é realizada a ***Sprint Planning Meeting***. É nesta etapa que a iteração é planejada, sendo selecionadas as estórias a serem implementadas durante a *sprint* baseando-se num *Product Backlog* pré-definido e priorizado. Essa reunião consiste em duas partes, cada uma com duração média de 4 horas.

Na primeira parte é decido o que será feito na *sprint*. A equipe, o *Product Owner* e o Scrum *Master* planejam as funcionalidades que serão feitas durante a *sprint.* O artefato de entrada usado nessa primeira parte é o *Product Backlog*. Nesta etapa a equipe avalia a complexidade dos itens priorizados nesse artefato.

Na segunda parte da *Sprint Planning Meeting* é quando a equipe descobre como vai construir as funcionalidades selecionadas no *Sprint Backlog* em um incremento do produto durante a *sprint*. A equipe tem como objetivo quebrar as estórias em tarefas que serão cumpridas durante a *sprint*, gerando o *Sprint Backlog.* Desse modo, a equipe verifica se a estimativa inicial dada está de acordo com todas as atividades existentes.

Para que o processo possa ser acompanhado sem atrasos são realizadas reuniões diárias de 15 minutos, chamadas de ***Daily* Scrum *Meetings***. Essas reuniões acontecem no mesmo local e horário durante toda a *sprint* e visam melhorar a comunicação, eliminar outras reuniões, identificar e remover obstáculos ao desenvolvimento, destacar e promover a rápida tomada de decisões e melhorar o

nível de conhecimento de todos sobre projeto. A *Daily* Scrum *Meeting* não é uma reunião de status, mas inspeção do progresso em direção à meta da sprint.

O Scrum *Master* é o moderador da reunião e deve garantir sua realização diária, e a equipe deve relatar o progresso do desenvolvimento respondendo a três questões, que guiam a reunião: “O que tem realizado desde a última reunião?”, “O que pretende fazer antes da próxima reunião?” e “Quais são os impedimentos para realizar seu trabalho com eficácia?”. O Scrum *Master* também tem a responsabilidade de reforçar a regra de que os *chickens* não estão autorizados a falar ou de qualquer forma interferir com a reunião. Todos os membros devem participar dessa reunião e devem ser sucintos, evitando problemas específicos.

Quando é relatado um problema na reunião diária de interesse de outros membros da equipe ou quando há a necessidade de assistência de outros membros da equipe, qualquer membro de equipe pode se organizar imediatamente após a *Daily* Scrum *Meeting* [SCHWABER, 2004].

A ***Sprint Review Meeting*** é um ponto de inspeção ao fim de cada iteração, em que é feita uma reunião de revisão com duração máxima de 4 horas, monitorada pelo Scrum *Master*. É uma reunião informal, onde a equipe exibe ao *Product Owner* e interessados os itens de *Backlog* considerados prontos e inicia-se uma nova *sprint*.

Após a *Sprint Review* e antes da *Sprint Planning Meeting* seguinte, o Scrum *Team* realiza a ***Sprint Retrospective***. Nesta reunião de cerca de 3 horas, o Scrum *Master* incentiva a equipe rever, no âmbito do processo Scrum e práticas, seu processo de desenvolvimento a fim de para torná-lo mais eficaz e agradável para a próxima *sprint*. Isso se dá por Scrum possuir conjunto de práticas focadas em melhoria adaptativa do processo, sendo feita uma busca por melhorias na produtividade ou qualidade do produto final. O objetivo principal da *Sprint Retrospective* é a inspecionar como se desenvolveu a *sprint* passada no que diz respeito às pessoas, relações de processos e ferramentas.

3.3 Resumo do Capítulo

Neste capítulo foi apresentada a metodologia selecionada para a utilização neste trabalho, apresentado sua origem e termos das práticas, bem como seu *framework*, detalhando os papéis e responsabilidades, as etapas e ciclo de desenvolvimento, e os artefatos e as ferramentas definidas pela metodologia Scrum.

Capítulo 4

Adaptação e Implantação do Método Scrum

O objetivo deste capítulo é descrever adaptação do método Scrum ao ambiente de estudo e sua implantação, através da caracterização do ambiente quando ao seu processo de desenvolvimento e equipe de produção, relacionando-os às práticas, artefatos e ferramentas da metodologia Scrum, a fim de apresentar as adaptações executadas no processo de implantação do método e comprovar a simplicidade, praticidade e adaptabilidade da metodologia.

4.1 Caracterização do Ambiente

O estudo de caso deste trabalho trata-se de um ambiente de desenvolvimento de softwares educativos apresentados em forma de aulas multimídia. O produto é dividido por matérias escolares (matemática, física, química, entre outros), de acordo com o projeto em execução, que define para qual faixa etária será destinado o material. O projeto vigente trata-se de material para ensino fundamental, necessitando-se então das matérias matemática, ciências e português.

4.1.1 Processo de Desenvolvimento

A partir da divisão do produto por matérias são formadas as estruturas de desenvolvimento dos conteúdos: grupos de equipes responsáveis por cada matéria e compostas por coordenador de equipe, designers, programadores e professor roteirista. Assim, o processo de desenvolvimento atual está disperso por essas equipes que, através do seu coordenador, determinam como sua subdivisão do produto deve ser produzida, partindo de premissas básicas definidas pela empresa a respeito de identidade visual e tecnologia utilizada, por exemplo – nesse caso, Adobe Flash CS4 com ActionScript 3.0. São produzidas em média 3 aulas por mês, por equipe, de acordo com a experiência da autora.

É possível manter a uniformidade do produto devido às definições básicas para cada entrega (aula) do projeto; entretanto, é praticamente impossível integrar prazos entre as equipes e prever problemas e impedimentos nas diversas subdivisões do projeto. Além disso, numa análise interna de cada grupo, questões como definição da distribuição e alocação de tarefas, definição e seleção de competências, definição da matriz de responsabilidades e canais de comunicação, entre outras, estão sendo tratadas de acordo com a experiência e disponibilidade de cada coordenação.

Outro aspecto que fica a critério da coordenação das equipes é geração dos artefatos durante o desenvolvimento, sendo atualmente obrigatórios somente os documentos de roteiro da aula e de solicitação de áudios, visto que a produção da aula depende de instruções mínimas a respeito do conteúdo e sonoplastia. Sendo assim, não é gerada nenhuma documentação de levantamento de requisitos ou mesmo diagramas de desenvolvimento para os programadores.

4.1.2 Equipe de Produção

A equipe analisada neste trabalho, sob a coordenação da autora, é composta por dois designers, dois desenvolvedores e um professor roteirista, todos trabalhando 20 horas por semana e responsáveis pela subdivisão de matemática. Compõem também a equipe dois membros em comum com as outras equipes, que são responsáveis pela sonoplastia e revisão ortográfica/gramatical.

Devido à falta de padronização no desenvolvimento atual para guiar a equipe, os membros possuem uma forte dependência das direções da coordenação, requerendo constantemente instruções para prosseguir com o desenvolvimento, bem como ocorre uma inevitável descentralização de informação, que é extremamente prejudicial visto que a maioria das tarefas executadas em cada aula possui interdependência, ou seja, é comum existir tarefas dependentes – e pertencentes a diferentes membros – sendo executadas simultaneamente.

Os principais impactos da ausência de metodologia de desenvolvimento no relacionamento interno da equipe são a desmotivação, por não se sentir “parte do processo”, e a não aceitação da forte hierarquia da estrutura atual pela equipe, visto que há uma real monopolização de decisões por parte da coordenação.

4.2 Análise e Correlação da Metodologia Scrum com o Ambiente

Após a análise da empresa foco deste trabalho foi possível elencar opções de metodologias aplicáveis para o contexto do problema. E, depois de avaliadas as opções, citadas no capítulo 2, pôde-se concluir ser a metodologia Scrum a mais adequada para a implantação nesse caso.

4.2.1 Processo de Desenvolvimento do Ambiente

Como não há diretrizes superiores para execução de um método de desenvolvimento, se faz necessária a escolha de uma metodologia simples de ser rapidamente assimilada e aplicada pelos coordenadores, visto que seriam eles os principais responsáveis pela implantação e manutenção do processo. Devido a sua adaptabilidade, a metodologia Scrum atende a essas exigências de simplicidade e rápida aplicabilidade, de modo que as coordenações de equipes não necessitam alterar suas práticas de desenvolvimento em função da implantação do método, e sim adaptar o método às atuais etapas do processo de desenvolvimento, visando sua organização e padronização.

Quanto às questões de distribuição e alocação de tarefas e seleção de competências, a coordenação passará a ser ainda mais habilitada a desempenhar estas funções, devido à praticidade em como são tratadas pela metodologia Scrum. Além dessas questões, a definição da matriz de responsabilidades e os canais de comunicação também são abordados pela metodologia e, se implementados corretamente, serão ferramentas facilitadoras ao trabalho da coordenação.

Em relação à geração dos artefatos, estes são limitados e pré-definidos de modo que sejam elaborados os estritamente necessários para orientar o desenvolvimento e manter equipe e cliente informados durante toda a iteração. Entretanto, se fazem indispensáveis para o bom funcionamento da metodologia. A vantagem na produção dos artefatos é novamente sua capacidade de adaptação à realidade do projeto no qual está sendo usado: a documentação pode abranger, de forma simples e objetiva, a finalidade das outras documentações já existentes de roteirizarão e sonoplastia.

Tabela 4.1 Resumo dos relacionamentos quanto ao processo de desenvolvimento na visão de ambiente *versus* Scrum

Aspecto Analisado	Ambiente	Scrum
Distribuição e alocação de tarefas e seleção de competências	Executado pelo coordenador de equipe	Executado pelo Scrum *Master*
Definição da matriz de responsabilidades e os canais de comunicação	Executados pelo coordenador de equipe	Inerentes à metodologia, representados pelos recursos sugeridos: Scrum *Board*, método *Planning Poker* para estimativa de tarefas, estruturas de *Backlog* (*Product* e *Sprint*)
Geração dos artefatos	Executados sem pré-definição sob a responsabilidade do coordenador de equipe	Definidos pela metodologia, implementados e adaptados pelo Scrum *Master*

4.2.2 Equipe de Produção do Ambiente

Com o uso da metodologia Scrum, igualmente às práticas, os papéis da equipe de produção também podem ser relacionados aos definidos pela metodologia.

Como citado no capítulo 3, o Scrum possui três papéis: o Scrum *Master*, o *Product Owner* e o Scrum *Team* (equipe). O papel do Scrum *Master* equivale ao do coordenador de equipe, sendo somente necessário ao coordenador assumir a postura de facilitador, não só de líder, como é sua atual função. Ao assumir a função de Scrum *Master*, o coordenador adiciona às suas atribuições a garantia do uso do Scrum de maneira correta, a remoção dos impedimentos e a participação de todas

as reuniões definidas pela metodologia, a fim de assegurar a eficácia da metodologia.

O *Product Owner*, no estudo de caso, está representado pela alta gerência da empresa, que é responsável pelo contato com o cliente. Nesse caso específico, o Scrum *Master* não entra em contato com o cliente, devendo executar os requisitos repassados pela gerência da empresa.

A equipe, como delimitada pelo Scrum, é composta por 6 pessoas, entre programadores, designers e pedagogo, sendo todos os membros responsáveis por atingir juntos os objetivos definidos em cada *sprint*. A única ressalva em relação à equipe é sua jornada de trabalho, que é 20 horas por semana, de todos os membros. E, devido ao fato de serem todos estagiários e, por isso, ainda universitários, e ao fato da empresa em questão trabalhar com banco de horas, os horários em que estão em desenvolvimento é muito variado. Assim, alguns aspectos como a *Daily* Scrum *Meeting* e as outras reuniões da *sprint* devem ser adaptados em relação à flexibilidade de horário, por exemplo, para que seja possível a participação de todos.

Embora a maior dificuldade encontrada num processo de implantação de uma metodologia seja a característica comportamental de resistência à mudança, a equipe se mostrou interessada em adquirir um processo de desenvolvimento e cooperativa nas diversas etapas da implantação.

Tabela 4.2 Mapeamento entre papéis e responsabilidades na visão de ambiente *versus* Scrum

Ambiente	Scrum
Coordenador de equipe, líder, responsável unicamente pelo desenvolvimento da equipe.	Scrum *Master*, facilitador, mantenedor do desenvolvimento, contato com cliente.
Alta gerência, diretoria.	*Product Owner*, cliente.

Equipe limitada entre 5 a 9 membros, trabalhando em horários divergentes, atendida pelo coordenador.	Equipe limitada entre 5 a 9 membros, sempre presentes nas etapas/reuniões definidas pela metodologia, e estimulados pelo Scrum *Master*.

4.3 Planejamento de Implantação do Método Scrum

Após analisadas as características do ambiente e correlacionadas ao método Scrum foi possível definir uma planejamento para a implantação do método, no que diz respeito às etapas, artefatos e ferramentas.

A etapa inicial do planejamento foi definir a amostragem da experiência, delimitada em cerca de 50 dias, entre as fases de treinamento da equipe, aplicação do método e coleta de dados e resultados. De acordo com o cronograma da empresa foi possível selecionar duas aulas para serem produzidas na fase de aplicação do método, que corresponde a um mês de produção, sendo cada aula produzida em 2 semanas, período esse caracterizado pela *sprint*.

Em seguida a definição da *sprint*, foi realizado o levantamento referente aos artefatos e ferramentas do Scrum a serem utilizados no desenvolvimento e às etapas de desenvolvimento definidas pela metodologia Scrum a serem atribuídas ao processo de desenvolvimento atual da empresa, como descritos nas próximas seções.

4.4 Levantamento de Artefatos e Ferramentas

A implantação do método Scrum inclui uma nova realidade na dinâmica de desenvolvimento da empresa foco deste trabalho: a geração de artefatos. Mesmo tendo consciência da necessidade de documentar o processo de desenvolvimento,

esta não é uma prática adotada pelos coordenadores, devido à falta de instruções superiores e também à ausência de uma rotina que demonstrasse a real utilidade dessa prática.

Com o uso do Scrum é possível formalizar toda a documentação necessária, sem que haja geração de artefatos em excesso e sendo feitas as devidas adaptações para que esses artefatos sejam de fato úteis.

Como detalhado no capítulo 3, dentre os artefatos do Scrum estão o *Product Backlog* e o *Sprint Backlog*, e, além destes, também são usados como ferramentas o Scrum *Board*, o *Burndown Chart* e o *Planning Poker*.

Todos os artefatos e ferramentas citados foram utilizados neste trabalho, de forma a comprovar o nível de adaptação da metodologia ao ambiente de desenvolvimento de softwares educativos, como segue abaixo.

4.4.1 *Product Backlog*

Como definido anteriormente, o *Product Backlog* é uma lista de itens priorizados elencando o que deve ser desenvolvido na *sprint* que, nesse estudo de caso, possui 2 semanas de duração.

Para a associação entre as práticas foi determinado que o *Product Backlog* corresponde à matriz de temas de aulas a serem produzidas durante toda a duração do projeto, a qual já foi refinada a fim de excluir as aulas que já estariam prontas, produzidas em projetos anteriores.

4.4.2 *Sprint Backlog*

Como dito no capítulo 3, o *Sprint Backlog* é composto por uma lista de tarefas extraídas do *Product Backlog*, as quais a equipe se compromete a fazer durante uma *sprint*.

Para compor o *Sprint Backlog*, nesse estudo de caso, foram selecionadas duas aulas do *Product Backlog*, para serem produzidas em duas *sprints*, cada uma de duração de 2 semanas. Essas aulas são subdivididas em páginas, e a listagem dessas páginas compõem o *Sprint Backlog* e as tarefas afixadas no Scrum *Board*, como será descrito na próxima seção.

Embora a amostragem da experiência seja limitada a duas *sprints*, ou seja, a um mês de desenvolvimento, foi feita uma seleção de aulas para a elaboração de um cronograma em longo prazo, que seria concluído em julho/2010. Nessa etapa de seleção estavam presentes o Scrum *Master*, o *Product Owner*, o professor roteirista e membros da equipe. Assim, para cada *sprint* até julho/2010, já existe um *Sprint Backlog* pronto.

4.4.3 Scrum *Board*

O Scrum *Board* é a principal ferramenta utilizada pela equipe para manter-se atualizada do progresso da iteração, seja de atribuições individuais ou como um panorama geral da *sprint*. Foi dividido em 8 seções, como mostra a Figura 4.1.

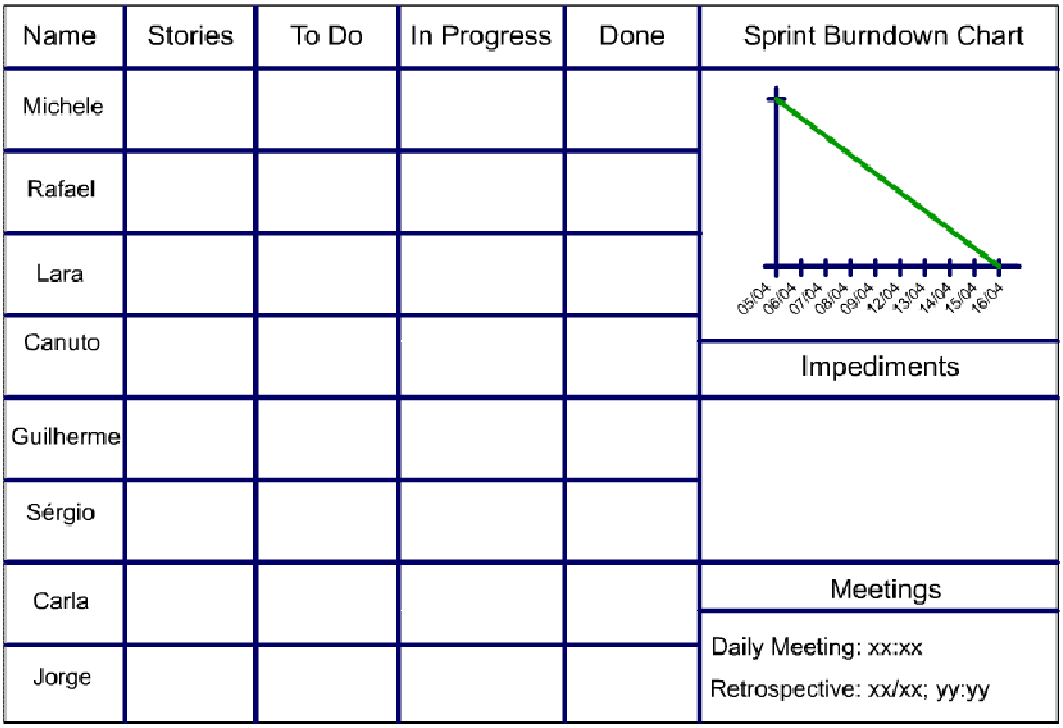

Figura 4.1 Figura ilustrativa do Scrum *Board* utilizado pela equipe.

No Scrum *Board* estão presentes também os membros Carla e Jorge, que são os responsáveis pela correção do roteiro e sonoplastia, respectivamente. Mesmo não sendo membros fixos do Scrum *Team*, suas atribuições possuem ligações de dependências com as atribuições dos outros membros, por isso são expostas no quadro, para que os membros que dependem de suas tarefas concluídas possam ter esse *feedback*.

As aulas em desenvolvimento são divididas em páginas, e estas são inseridas na coluna *Stories*, para determinar qual membro será responsável pelo desenvolvimento – seja de visual ou de funcionalidade – da referida página. Na coluna *To Do* estão listadas as atribuições, em mais alto nível, que cada membro deve executar, item a item, como mostrado na Tabela 4.3.

Tabela 4.3 Relação de atribuição alto nível por desenvolvedores, designers e Scrum *Master*

TO DO List	
Programadores	XML
	Som e Sincronizar personagem
	Funcional
	Documentar
	Redimensionar balões
Designers	Animação Geral: -Externa ao personagem -Zoom
	Animação de Personagem: - Interna ao personagem - Movimento de falas
	Cenário
	Elementos
	Criação de personagem
	Organizar biblioteca

Scrum *Master*	Gerar XML
	Elaborar Roteiro v2
	Elaborar Solicitação sons
	Gerar Diagramas

Na coluna *In Progress* estarão os *post-its* das tarefas listadas na coluna *To Do*, à medida que são executadas. Ao fim de alguma das tarefas, quando esta é testada e comprovadamente finalizada, é transferida para a coluna *Done*, para ser computada e registrada no *Burndown Chart*, como mostrado na Figura 4.2.

As seções *Impediments* e *Meetings* são opcionais e foram inseridas para cumprir os seguintes propósitos: a seção *Impediments* serve para evitar que os problemas encontrados no desenvolvimento permaneçam, atrasando a *sprint*; assim, sempre que há algum impedimento, o membro da equipe deve relatá-lo nesse quadro para que o Scrum *Master* tenha o registro do obstáculo e possa removê-lo. Já a seção *Meetings* foi inserida visando à melhoria na comunicação interna da equipe, visto que, como já comentado, os horários de trabalho não coincidem sempre e, quando acontece, devem ser registrados para que todos lembrem os compromissos da *sprint*, como a *Daily* Scrum *Meeting* e a *Sprint Retrospective*.

4.4.4 *Burndown Chart*

O gráfico *Burndown* é uma das seções dos Scrum *Board* e, como já dito, tem o objetivo de mostrar a correlação entre a quantidade de trabalho restante em qualquer ponto no tempo e o progresso da equipe na redução deste trabalho.

Há diversas formas de registrar o trabalho executado no gráfico, sendo as duas principais por quantidade de tarefas e por pontos de complexidade dessas tarefas. No estudo de caso deste trabalho foi usada esta última opção, que utiliza o *Planning Poker* para efetuar as métricas de complexidade das tarefas.

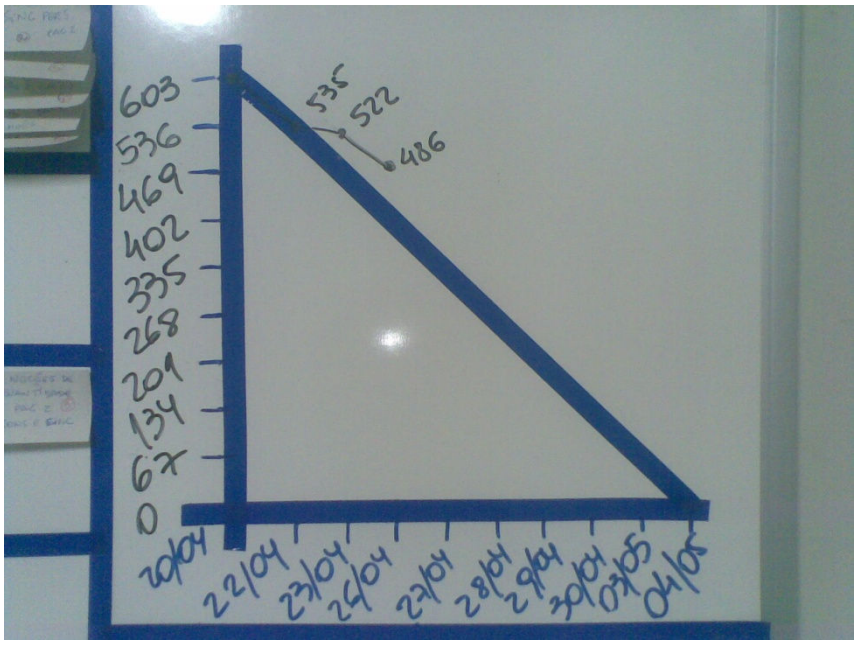

Figura 4.2 Imagem do *Burndown Chart* da segunda *sprint*, mostrando a evolução do desenvolvimento no quarto dia da iteração.

Todas as tarefas listadas na seção *To Do* são mensuradas, de forma que somadas preenchem o valor total de pontos no gráfico. À medida que são executadas são subtraídas do total de pontos até que o gráfico chegue ao ponto zero. A elaboração do *Planning Poker* será descrita na próxima seção.

4.4.5 *Planning Poker*

O *Planning Poker* foi o método utilizado para estimar o desenvolvimento da *sprint*. Como explicado anteriormente, esse método utiliza cartas de baralho valoradas, que somente são exibidas após a descrição da atividade a ser executada, para a avaliação de complexidade. A escala utilizada no baralho é a seguinte: 1, ½, 2, 3, 5, 8, 13, 21, 40 e 100, como mostrado na Figura 4.3; assim, o valor quantitativo escolhido para um item através dessa escala é diretamente proporcional a sua complexidade.

Figura 4.3 Figura ilustrativa do *Planning Poker* personalizado.

Durante o *Planning Poker*, cada tarefa listada na seção *To Do* é discutida sempre se baseando no documento de roteiro. As reuniões são feitas em duplas, visto que a equipe é composta por 2 programadores e 2 designers. Embora os programadores só valorassem as tarefas contidas no *To Do* dos programadores, poderiam opinar no *Planning Poker* dos designers, por exemplo, no caso de empate, e vice-versa. Foi definido o menor valor do baralho para uma tarefa simples, já antes executada em outros projetos, como meio de comparação.

Assim, cada participante dá sua nota de complexidade com base nessa escala definida para cada tarefa e, quando há consenso, a nota é validada e atribuída à tarefa. No caso de divergência se abre espaço para discussão de toda a equipe e são reapresentadas as escolhas do valor de complexidade da dupla após a discussão. Caso não houvesse ainda consenso, o Scrum *Master* teria a responsabilidade de intervir; entretanto, essa situação não ocorreu.

4.5 Levantamento das Etapas e Processos

Analogamente aos artefatos e ferramentas, as etapas e processos da metodologia também são facilmente adaptáveis ao processo encontrado no estudo de caso. Todas as etapas no processo de desenvolvimento foram utilizadas e devidamente adaptadas para obter melhor uso da metodologia, como descrito nas próximas subseções.

4.5.1 *Sprint Planning Meeting*

Como citado anteriormente, é nesta etapa que a iteração é planejada, sendo selecionadas as estórias a serem implementadas durante a *sprint*, baseando-se no *Product Backlog*. Essa reunião consiste em duas partes, cada uma com duração média de 4 horas.

Na primeira parte a equipe, o *Product Owner* e o Scrum *Master* planejaram as funcionalidades que a serem executadas durante a *sprint*, ou seja, selecionaram as estórias do cronograma contendo as aulas pré-selecionadas do *Product Backlog*, representado nesse caso pela matriz de aulas.

Na segunda parte do primeiro *Sprint Planning Meeting*, a equipe definiu como registrar a aula no *Sprint Backlog* durante a *sprint*. Nesta etapa foram definidas pela equipe tarefas-padrão para compor as atribuições de cada membro da equipe, seja programador, designer ou o próprio Scrum *Master*, como resumidas na Tabela 4.3. Após a definição inicial dessas atribuições, todas as *sprints* seguiram o mesmo padrão, de modo que essa etapa passou a ser mais prática e clara para todos os membros da equipe.

Assim, é seguida a etapa em que a equipe mensura as funcionalidades observadas para cada página através do *Planning Poker*, como descrito na seção 4.5.5.

Figura 4.4 Imagem do Scrum *Board* montado após o *Sprint Planning Meeting*.

Depois de estimadas todas as tarefas, o Scrum *Board* é montado com os *post-its* de cada funcionalidade e estórias, com suas devidas pontuações de complexidade, como mostrado na Figura 4.4.

4.5.2 *Daily* Scrum *Meeting*

As *Daily* Scrum *Meetings* se mostraram as mais complicadas de adaptar, para esse caso específico, devido às divergências de horários entre os membros da equipe. Durante o desenvolvimento das *sprints* foi possível analisar a inviabilidade da realização dessa etapa, de modo que foi necessária uma medida paliativa para a coleta dessas importantes informações do processo de desenvolvimento: foi acordado entre os membros da equipe que aquele que não pudesse estar presente no horário acordado – e comum à maioria – relataria as três questões em forma de arquivo de texto, de forma sucinta como seria pessoalmente, visando manter a ligação entre os dias da *sprint*.

Este documento foi disponibilizado a todos da equipe, em diretório comum do projeto, com o propósito que todos pudessem estar a par dos impedimentos e andamento da *sprint* pela visão de toda a equipe.

4.5.3 *Sprint Review Meeting*

Como relatado anteriormente, a *Sprint Review* trata-se de uma reunião informal de revisão da *sprint* com duração máxima de 4 horas, monitorada pelo Scrum *Master*, onde a equipe exibe ao *Product Owner* e interessados os itens do *Product Backlog* considerados prontos, e inicia-se uma nova *sprint*.

Devido aos pequenos atrasos ocorridos nas iterações e às interferências do *Product Owner*, somente uma *Sprint Review Meeting* foi realizada, resumindo os release de duas *sprints*.

Apesar de trabalhando num mesmo projeto – a aula –, a equipe não possui uma visão do produto completo, sendo a principal função desta reunião mostrar o produto da *sprint* e servir como estímulo para a continuação de mais *sprints* bem sucedidas. A única ressalva da reunião foi a ausência do *Product Owner*, que obteve a ata da reunião posteriormente.

4.5.4 *Sprint Retrospective*

A *Sprint Retrospective* tem a função de incentivar a equipe reavaliar o desenvolvimento, no âmbito do processo Scrum e práticas, a fim de detectar aspectos de pessoal, relacionamentos, processo e ferramentas. É a oportunidade que o time tem para discutir sobre o que funcionou ou não durante a *sprint*, levantando as questões: “O que funcionou bem?” e “O que pode ser melhorado?”.

No caso estudado, esta foi a mais produtiva das reuniões que, com apenas 30 min. de duração, permitiu levantar inúmeras vantagens do uso da metodologia, bem como sugestões de melhorias para as etapas que não foram imediatamente adaptadas, como mostrado na Tabela 4.4.

Tabela 4.4 Relação de vantagens e melhorias do processo de implantação do método Scrum elencadas pela equipe de produção durante a *Sprint Retrospective*.

Vantagens	Melhorias
Manter o controle das tarefas a serem feitas, evitando esquecimento.	Alterar as dimensões das áreas de *To Do* e *Done* no Scrum *Board*.
Organizar as etapas do desenvolvimento, auxiliando determinar o início, meio e fim.	Definir um padrão para determinar as dependências entre as tarefas nos *post-its*.
Prover uma visão do projeto todo para a equipe inteira.	Permitir maior flexibilidade nos horários das *Daily* Scrum *Meetings*, visando minimizar o uso dos arquivos de texto.
Permitir toda a equipe de ter noção da complexidade de todas as tarefas, auxiliando o desenvolvedor a entender o tempo de produção do designer, e vice-versa.	
Inserir uma melhor divisão do desenvolvimento, permitindo detectar um padrão.	
Perceber o andamento do projeto – através do *Burndown Chart* – permitindo a equipe manter o controle do prazo “antes de sair dele”.	

4.6 Documentação do Processo Utilizado

Além dos artefatos demandados pela metodologia, que foram elaborados com as devidas adaptações, foi feito uso de um artefato da Engenharia de Software para minimizar o trabalho da coordenação no processo de formulação do roteiro e auxiliar os membros da equipe no fluxo de desenvolvimento de cada página (estórias da aula).

Assim, foram elaborados pela coordenação de equipe diagramas de estado de cada página da aula em execução na *sprint*, detalhando o fluxo de ocorrência dos eventos e animações definidas para cada página. Este artefato não visa a substituição dos outros artefatos mencionados, mas sua complementação, sendo indicado seu uso paralelamente a estes.

O uso dos diagramas de estado minimizou de forma efetiva a dependência da equipe com relação ao coordenador, eliminando as dúvidas frequentes no roteiro, acelerando o desenvolvimento e reduzindo o número de manutenções e alterações feitas por página.

4.7 Resumo do Capítulo

Neste capítulo foi abordado o processo de adaptação e implantação do método ao ambiente de estudo, caracterizando o ambiente – seu processo de desenvolvimento e equipe de produção – e relacionando com as práticas da metodologia Scrum. Além disso, foi feito um levantamento de artefatos e ferramentas utilizadas na implantação do método, bem como as etapas e processos adaptados. Por fim, foi listada a documentação utilizada no processo além da pré-definida pela metodologia.

Capítulo 5
Coleta e Análise de Dados do Processo de Implantação do Método Scrum

O objetivo deste capítulo é descrever o resultado do estudo sob o aspecto comportamental da equipe de produção e sob as características do processo de desenvolvimento quanto ao prazo, à qualidade e aos custos relacionados às práticas da metodologia Scrum, visando apresentar o produto das adaptações executadas no estudo de caso.

5.1 Análise Comportamental da Equipe

Diferentemente de outras metodologias, a implantação do Scrum não exige esforços de adaptação para geração de documentação, hierarquia de equipe ou divisão tradicional de papéis da Engenharia de Software, por exemplo; entretanto, demanda rigor no cumprimento das etapas de desenvolvimento, sendo mandatórias as reuniões de planejamento, revisão e retrospectiva por iteração.

Isto posto, neste estudo foram feitas as devidas alterações na implantação, já citadas no capítulo anterior, para que fossem seguidas todas etapas determinadas pela metodologia, sendo todas questionadas, verificadas e aprovadas por toda a equipe de produção.

Assim, com essa abordagem de integração inicial da equipe com o processo de implantação foi possível atingir o máximo de aceitação da metodologia, como observado no interesse pelas reuniões diárias (apesar das limitações de horário), na participação ativa na *Sprint Planning Meeting* e na pontualidade e comprometimento com as *Sprint Review* e *Sprint Retrospective*. Nestas últimas foram levantados aspectos positivos do produto e processo e sugestões de melhorias, como a inserção de uma legenda para o *post-it* para representar o fluxo de dependências

entre as tarefas – artifício que, quando implantado, ajudará a equipe na ordem de execução de suas próprias tarefas.

Quanto aos recursos físicos e *softwares* implantados foi feito uso do Scrum *Board* para a afixação dos *post-its*, do baralho do método *Planning Poker* para a métrica das estórias e do Violet UML para a geração dos diagramas de estado. A análise do Scrum *Board* feita pela equipe o descreve como "uma ferramenta que fornece uma visão global do projeto". A equipe também entrou em consenso de que, além de fornecer um panorama completo em relação a todas as tarefas do projeto em execução, contribui para que cada membro oriente seu próprio desenvolvimento de acordo com (i) as tarefas que dependem de outras (suas ou de outros membros) para serem concluídas; (ii) as tarefas que afetam o desenvolvimento de outros membros; e (iii) as tarefas prioritárias, representadas pelas maiores complexidades estimadas no *Planning Poker*.

Além de mensurar a complexidade das tarefas e, consequentemente, detectar as prioridades dentre elas, outro ponto positivo relatado pela equipe é a cumplicidade que o *Planning Poker* gerou entre os membros da equipe, visto que, como a estimativa não é feita de forma arbitrária pelo Scrum *Master*, cada membro contribui com a estimativa e se sente responsável por ela.

O uso do Violet UML ficou restrito ao Scrum *Master*, tendo a equipe acesso às imagens geradas dos diagramas de estado, bem como seus arquivos fonte. A utilização dessa ferramenta proporcionou à equipe mais independência na escolha do procedimento a ser usado para cada funcionalidade, e clareza quanto ao fluxo de eventos contido no roteiro em forma textual.

5.2 Análise de Eficiência da Metodologia Scrum no Desenvolvimento das Tarefas

Além de projetar melhorias no ambiente de desenvolvimento, como comunicação direta e sem falhas, e interatividade, independência e transparência na tomada de decisões entre equipe e gerência; a implantação do método Scrum propunha a otimização e homogeneidade do tempo de desenvolvimento da equipe, visando à entrega do produto de qualidade e em tempo hábil.

Isto posto, foram tomados como critérios os aspectos de prazo, qualidade e custos para a avaliação da eficiência da metodologia Scrum quando aplicada ao ambiente de desenvolvimento de softwares educativos.

5.2.1 Prazo

Quanto ao prazo, foi possível observar que as aulas, após a implantação do método passaram a ser produzidas 2 por semana. Embora a priori fossem desenvolvidas numa média de 3 semanalmente, após a implantação as aulas passaram a ser mais completas, exigindo menos correções – em uma das iterações, nenhuma correção foi feita – enquanto as aulas produzidas antes da implantação exigiam cerca de 2 semanas a mais para manutenção.

Assim, pode-se concluir que a implantação do método contribuiu para minimizar o tempo gasto com correções, reduzindo o prazo total de desenvolvimento de cada objeto de aprendizagem.

5.2.2 Qualidade

Como dito na seção anterior, após a implantação do método foi possível entregar um produto aparentemente mais robusto, adequado ao uso (cumprindo as requisições de usabilidade) e satisfazendo os requisitos do *Product Owner*; correspondendo assim, de acordo com os conceitos de Gerenciamento de Qualidade do PMBok, a um produto de qualidade.

Foi possível notar a redução de erros e adequação da usabilidade das aulas desenvolvidas após a implantação do Scrum em detrimento às anteriores através das documentações de revisão elaboradas pelo *Product Owner* (também responsável por analisar o produto em relação à qualidade e requisitos do cliente final); estas passaram a relatar menor quantidade de pontos de correção e, quando relatados, correspondem a tarefas de menor complexidade e que não são relacionadas à usabilidade do produto.

Entretanto, foi detectada a necessidade de ser implantando um processo de Gerenciamento de Qualidade, de forma a quantificar – não somente qualificar – o produto nos padrões de qualidade. Como trabalhos futuros é recomendada a aplicação dos conceitos de Gerenciamento de Qualidade do PMBok, para assegurar

que as necessidades que deram origem ao desenvolvimento projeto sejam atendidas. Para se adequar à metodologia Scrum, é sugerido que o controle da qualidade seja feito através de avaliação periódica do desempenho geral do projeto, sendo baseada nas informações coletadas nas *Daily* Scrum *Meetings* e nos dados registrados diariamente no *Burndown Chart*.

5.2.3 Custos

Os custos na implantação do método se resumem a dois aspectos: de tempo de produção, devido aos dias de trabalho usados para a apresentação da metodologia e para as *Sprint Review* e *Sprint Retrospective*, quando não há produção; e de investimento financeiro relacionado à aquisição de ferramentas como o baralho do *Planning Poker*, o quadro Scrum *Board* e os *post-its*, visto que até mesmo a ferramenta usada na geração dos diagramas de estado é *open source*.

Logo, é possível avaliar que o custo com a implantação do método Scrum não impacta no orçamento da empresa e tem pequena influência no cronograma de produção da empresa.

5.3 Resumo do Capítulo

Neste capítulo foi apresentada a coleta e análise dos dados do processo de implantação do método Scrum, abordando o aspecto comportamental da equipe e a eficiência da metodologia quanto ao prazo de entrega, à qualidade do produto e ao custo de produção, como resumido na Tabela 5.1.

Tabela 5.1 Resumo comparativo da avaliação do ambiente antes e após a implantação do método Scrum.

Ambiente Pré-Implantação	Ambiente Pós-Implantação
Hierarquia da equipe composta pelas três instâncias: alta gerência, coordenador de equipe, equipe.	Hierarquia da equipe composta pelas três instâncias: *Product Owner*, Scrum *Master* e Scrum *Team*.

Papel da alta gerência: gerenciar os coordenadores de equipe e contatar os clientes. A alta gerência não participa do desenvolvimento em nenhuma etapa.	Papel do *Product Owner*: colaborar com o Scrum *Master* e a equipe na seleção e manutenção das prioridades do desenvolvimento de acordo com o valor de negócio da empresa. Além de contatar os clientes.
Papel do coordenador de equipe: líder. Responsável por guiar a equipe para obter resultados de acordo com suas próprias definições do produto e premissa básicas determinadas pela empresa. Responsável por remover as dúvidas frequentes da equipe quanto ao processo de desenvolvimento, visto que este é definido por sua experiência do coordenador.	Papel do Scrum *Master*: facilitador. É responsável por remover os impedimentos da equipe no processo de desenvolvimento, não sendo responsável por definir esse processo, mas por assegurar que a metodologia Scrum seja seguida quanto às etapas, aos artefatos e papéis.
Papel da equipe: desenvolver os objetos de aprendizagem de acordo com a documentação de roteiro gerada e as instruções do coordenador e reportar todas as dúvidas e problemas ao coordenador, sempre que surgirem. Possui total dependência do coordenador para sua organização.	Papel do Scrum Team: É responsável por ser auto-organizada e por selecionar os itens priorizados que irão ser executados em cada sprint, com total liberdade e comprometimento para desenvolver os objetos de aprendizagem de acordo com as etapas definidas pela metodologia Scrum; bem como responsável por reportar os impedimentos encontrados (através do Scrum Board) para que o Scrum Master os remova. É também papel da equipe participar ativamente das reuniões diárias e de revisão e retrospectiva.

Processo de desenvolvimento: iterações sem etapas definidas ou delimitadas.	Processo de desenvolvimento: *sprints* com etapas pré-definidas e obrigatórias.
Ciclo de desenvolvimento: produção do roteiro, seguida do desenvolvimento (com testes periódicos, mas sem padronização) e publicação do objeto de aprendizagem.	Ciclo de desenvolvimento: produção do roteiro, seguida da *Sprint Planning Meeting* (para validação do roteiro com equipe e *Product Owner*) e do desenvolvimento (com verificações diárias – *Daily* Scrum *Meetings*). O fim do desenvolvimento é seguido pela execução da *Sprint Review* (para validação do *Product Owner*) e a publicação do objeto de aprendizagem.

Capítulo 6

Conclusão e Trabalhos Futuros

Analisando o cenário de produção de software é possível perceber que os maiores problemas enfrentados pelas empresas de desenvolvimento de software – atrasos na entrega do projeto, produtos de baixa qualidade e aumento significativo dos custos, por exemplo – são causados principalmente pela falta de gerenciamento nos processos. Isso justifica a necessidade da adoção de processos que utilizem práticas ágeis que visem uma redução de desperdício e que sobrevivam diante de um ambiente em que as mudanças ocorrem com grande frequência.

Através da análise dos resultados pôde-se observar que a metodologia ágil Scrum se mostrou adequada para o uso em ambientes de desenvolvimento de softwares educativos, e que sua aplicação engloba todas as etapas do desenvolvimento, através de pequenas e médias adaptações.

Apesar de exigir rigor no cumprimento das etapas e reuniões, remove a obrigatoriedade de geração de vasta documentação, sendo necessária somente a escolhida pelo Scrum *Master* ou *Product Owner*, como foi o caso da geração de diagramas de estado e roteiros de aulas, respectivamente, justificado pela dinâmica de desenvolvimento do negócio em questão.

Assim, quando cumpridas, as etapas definidas fornecem importantes dados relativos à produtividade das equipes e são importantes elementos na construção de um processo adaptativo com constantes melhorias e foco na comunicação, como o caso da *Daily* Scrum *Meeting*, que relata o progresso do desenvolvimento, e da *Sprint Retrospective*, que identifica falhas e lacunas presentes no ciclo de desenvolvimento da metodologia Scrum.

O estudo também mostrou a boa adaptabilidade e aceitação da equipe à metodologia, não sendo a característica comportamental da equipe um obstáculo à implantação do método. Por ter sido rapidamente assimilada pela equipe, a queda de produtividade inicial à implantação do método não se caracterizou como alto impacto no cronograma do projeto.

6.1 Dificuldades Encontradas

Como já mencionado, devido às divergências entre os horários de cada membro da equipe, as *Daily* Scrum *Meetings* ficaram comprometidas, visto que a troca de experiências durante a iteração não foi inteiramente compartilhada por todos. Para contornar esse obstáculo foi proposto e aprovado pela equipe o uso de arquivos de texto, disponibilizados a todos, para registrar as questões respondidas na *Daily* Scrum *Meeting*, da mesma forma que feita pessoalmente.

Outra dificuldade encontrada no processo de implantação do método se deu por conta dos atrasos e interferências causadas pelo *Product Owner*, que impediram as *Sprint Review* e *Sprint Retrospective* de ocorrerem no prazo correto, visto que alguns requisitos foram adicionados pelo *Product Owner* e estes implicaram no cancelamento da *sprint*, sendo uma nova criada para abordá-los. Desta forma, somente aconteceu uma vez cada uma dessas reuniões, quando foram discutidos os produtos das 2 *sprints* desenvolvidas neste trabalho.

6.2 Trabalhos Futuros

Para trabalho futuro, é sugerido implantar o processo de Gerenciamento de Qualidade, no qual o controle da qualidade seja feito através de avaliação periódica do desempenho geral do projeto, sendo baseada nas informações coletadas nas *Daily* Scrum *Meetings* e nos dados registrados diariamente no *Burndown Chart*. Além disso, que seja desenvolvido um artifício de indicação de dependências entre as tarefas afixadas no Scrum *Board*, que, quando implantado, ajudará a equipe na ordem de execução de suas próprias tarefas.

Além disso, visando minimizar o impacto causado pela divergência de horários entre os membros da equipe, é proposto o uso de ferramentas case, como a sugerida *open source* FireScrum – desenvolvida no Programa de Mestrado de Engenharia de Software do CESAR.EDU –, para prover suporte ao desenvolvimento remoto.

Por fim, é esperado também que, depois de feitas essas alterações, o método Scrum seja implantado em todas as equipes da empresa, de modo a analisar as outras formas de adaptação da metodologia e poder, assim, quantificar resultados e

levantar informações mais consistentes sobre o uso da metodologia Scrum, visando apresentá-las à comunidade de desenvolvimento e fomentar sua implantação em outras empresas.

Bibliografia

[ABRAHAMSSON et al., 2002] ABRAHAMSSON, P. et al. **Agile software development methods: Review and analysis**. VTT Publications 478. Finlândia. 2002. Disponível em <http://www.vtt.fi/inf/pdf/publications/2002/P478.pdf>. Acesso em 20/04/2010.

[AMBLER, 2004] AMBLER, Scott W. **Lessons in Agility from Internet-Based Development**. IEEE Software. p. 66-73. 2004.

[BANKI & TANAKA, 2009] BANKI A.; TANAKA S., **Metodologias Ágeis: Uma Visão Prática**. Engenharia de Software Magazine, Rio de Janeiro. v. 04. p. 22-29. 2008. Disponível em <http://www.devmedia.com.br/assgold/listmag.asp?site=48>. Acesso em 14/04/2010.

[BARNETT, 2006] BARNETT, L. **Agile Survey Results: Solid Experience And Real Results**. Agile Journal. 2006.

[BECK, 1999a] BECK, K. **Embracing Change With Extreme Programming**. IEEE Computer. p. 70–77. 1999.

[BECK, 1999b] BECK, K. **Extreme programming explained: Embrace change**. Boston, Addison-Wesley, 1999.

[BECK, 2004] BECK, K. **Programação Extrema (XP) explicada: acolha as mudanças**. Bookman, Porto Alegre, 2004.

[BENTZEN, 2009] BENTZEN, Hélio. **Avaliação Qualitativa Da Utilização Da Abordagem Scrum Em Um Ambiente Real Através Do Método Nokia Test**. 2003. 68f. Tese (Conclusão de Curso) - Escola Politécnica de Pernambuco da Universidade de Pernambuco, Pernambuco.

[BONA, 2002] BONA, C. **Avaliação de Processos de Software: Um Estudo de Caso em XP e ICONIX**. 2002. Tese (Mestrado) - Universidade Federal de Santa Catarina, Santa Catarina.

[COCKBURN, 2002] COCKBURN, A. **Agile Software Development Joins the "Would-Be" Crowd**. Cutter IT Journal 15. p. 6-12. 2002.

[COCKBURN, 2002] COCKBURN, A. **Agile Software Development**. Boston, Addison-Wesley, 2002.

[COHN, 2005] COHN M. **Agile Estimating and Planning.** Addison-Wesley, 2005.

[GRENNING, 2002] GRENNING, J. **Launching XP at a Process-Intensive Company**. IEEE Software. p. 3-9. 2002.

[GRINYER, 2007] GRINYER, Antony R. **Investigating the Adoption of Agile Software Development Methodologies in Organizations**. 2007. Faculdade de Matemática e Informática. Universidade Aberta, Reino Unido.

[HARRISON, 2005] HARRISON, Warner. **Skinner Wasn't a Software Engineer**. IEEE Software. 2005.

[HAUNGS, 2001] HAUNGS, J. **Pair programming on the C3 project**. Computer 34. p. 118-119. 2001.

[HIGHSMITH e COCKBURN, 2001] HIGHSMITH, J.; COCKBURN, A. **Agile Software Development: The Business of Innovation**. Computer 34. p. 120-122. 2001.

[JEFFRIES, 2005, apud GRINYER, 2007] JEFFRIES, R., et al. **Extreme Programming Installed**. Addison-Wesley, Nova Jersey, 2005.

[NONEMACHER, 2003] NONEMACHER, Marcos L. **Comparação e Avaliação entre o Processo RUP de Desenvolvimento de Software e a Metodologia Extreme Programming**. 2003. Tese (Mestrado). Universidade Federal de Santa Catarina, Santa Catarina.

[PLAYFAIR, 2008] PLAYFAIR, K. **When 'General Agile' Isn't Enough - Why Scrum Wins in the Enterprise**. ǁ Agile Journal, 07/07/2008. Disponível em: <http://www.agilejournal.com/content/view/808/111/>. Acesso em 29/12/2008.

[POPPENDIECK & POPPENDIECK T., 2003] POPPENDIECK M.; POPPENDIECK T. **Lean Software Development**. Addison Wesley, 2003.

[PRIKLADNICKI, 2003] PRIKLADNICKI, R. **MuNDDoS - Um Modelo De Referência Para Desenvolvimento Distribuído De Software**. 2003. 144f. Tese (Mestrado) - Faculdade de Informática da Pontifícia Universidade Católica do Rio Grande do Sul, Rio Grande do Sul.

[PROJECTS@WORK, 2006] PROJECTS@WORK. **Agile 2006 Roundup**. Agosto, 2006. Disponível em <www.projectsatwork.com/article.cfm?ID=232426>. Acesso em 03/05/2010.

[SAVOINE et al., 2009] SAVOINE, M.; et al. **Análise de Gerenciamento de Projeto de Software Utilizando Metodologia Ágil XP e Scrum: Um Estudo de Caso Prático**. In: XI Encontro de Estudantes de Informática do Tocantins, 2009, Palmas. Anais do XI Encontro de Estudantes de Informática do Tocantins. Palmas: Centro Universitário Luterano de Palmas, 2009. p. 93-102. Disponível em: <http://tinyurl.com/yfsvnwz>. Acesso em 17/05/2010.

[SCHWABER, 2004] SCHWABER, K. **Agile Project Management with Scrum**. Microsoft Press, 2004.

[SCHWABER & BEEDLE, 2002] SCHWABER, K.; BEEDLE, M. **Agile Software Development With Scrum**. Prentice-Hall, Nova Jersey, 2002.

[SCHWABER & SUTHERLAND, 2009] SCHWABER K.; SUTHERLAND J. **Scrum Guide: Developed and sustained**. Scrum.org. 2009.

[SZIMANSKI, 2009, apud PLAYFAIR, 2008] SZIMANSKI, F. **Extensão Do Scrum Segundo As Áreas De Processo Do MPS.BR Nível G**. 2009. Tese (Mestrado). Centro de Estudos e Sistemas Avançados do Recife – CESAR, Pernambuco.

[TAKEUCHI & NONAKA, 1986] TAKEUCHI, H.; NONAKA, I. **The New Product Development Game**. Harvard Business Review. p. 137-146. 1986.

[TAPSCOTT & CASTON, 1993] TAPSCOTT, D.; CASTON, A. **Paradigm Shift – The New Promise of Information Technology**. McGraw-Hill, Nova Iorque, 1993.