THIAGO NOCHIMOWSKI GRACIADIO

UMA ANÁLISE E PROPOSTA DE INTEGRAÇÃO DO MÉTODO XP AO SCRUM

Monografia apresentada como trabalho final ao Programa de Educação Continuada em Engenharia (PECE) da Escola Politécnica da Universidade de São Paulo

São Paulo
2013

THIAGO NOCHIMOWSKI GRACIADIO

UMA ANÁLISE E PROPOSTA DE INTEGRAÇÃO DO MÉTODO XP AO SCRUM

Monografia apresentada como trabalho final ao Programa de Educação Continuada em Engenharia (PECE) da Escola Politécnica da Universidade de São Paulo

Área de Concentração: Tecnologia de Software

Orientador: Prof. Dr. Kechi Hirama

São Paulo

2013

FICHA CATALOGRÁFICA

Graciadio, Thiago Nochimowski
Uma análise e proposta de integração do método XP ao Scrum / T.N. Graciadio. -- São Paulo, 2013.
61 p.

Monografia (MBA em Tecnologia de Software) - Escola Politécnica da Universidade de São Paulo. Programa de Educação Continuada em Engenharia.

1. Projeto de software (Gerenciamento) 2. Desenvolvimento de software 3. Programação ágil XP 4. Scrum I. Universidade de São Paulo. Escola Politécnica. Programa de Educação Continuada em Engenharia II. t.

DEDICATÓRIA

Dedico este trabalho aos meus pais que tanto me incentivaram ao longo de todo andamento do curso. E também à minha namorada Gabriela Marques por todo auxílio durante este período.

AGRADECIMENTOS

À Escola Politécnica da Universidade de São Paulo – Por abrir espaço para este curso.

A todos os professores ao longo do curso por todo conhecimento transmitido nas aulas, trabalhos e etc.

Ao Prof. Dr. Kechi Hirama por orientar este trabalho, auxiliando intensamente para que o objetivo do trabalho fosse alcançado com sucesso.

RESUMO

Os métodos ágeis têm se difundido cada vez mais em empresas desenvolvedoras de software e muito se discute sobre suas possíveis vantagens e desvantagens se comparados a metodologias tradicionais de desenvolvimento de software. Dentre os métodos ágeis mais utilizados e populares, estão o *Scrum* e o *XP*.

Visando tornar o *Scrum* mais robusto, existem diversos relatos de combinações do *Scrum* utilizando práticas de provenientes do método XP.

Este trabalho apresenta uma análise da integração de práticas do XP no *Scrum*, determinando quais práticas são mais relevantes para estarem presentes nessa integração e posteriormente é apresentada uma proposta de integração entre os dois métodos.

ABSTRACT

The agile methods have been increasingly becoming widespread in software developtment companies and there is much discussion about the possible advantages and disadvantages when compared to traditional methods of software development. Among the agile methods, the most used and popular are Scrum and XP.

Aiming to make the Scrum more robust, there are several reports of combinations of Scrum using practices from the XP method.

This work presents an analysis of the integration of XP practices in Scrum, determining which XP practices are most relevant to be present in this integration and it proposes a integration of how the two methods could be performed.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Neste capítulo é apresentada uma introdução sobre o tema proposto, contendo as motivações existentes para a seleção deste tema, suas justificativas, assim como o objetivo e estrutura do trabalho.

1.1. Motivações

Desde meados do ano de 1970, o ciclo de vida clássico, também conhecido como modelo Cascata, já era usado em organizações desenvolvedoras de software, seguindo sua abordagem sistemática e sequencial sempre obteve melhores resultados quando usada em projetos em que seus requisitos são bem definidos e também razoavelmente estáveis (Pressman, 2001 p.34).

Com o passar dos anos, foram introduzidas novas abordagens para conduzir o processo de desenvolvimento de software, como é o caso do modelo Iterativo e Incremental, seguindo uma combinação de elementos do modelo Cascata, porém usado de forma diferente, em que no fim de cada ciclo é entregue uma parte do software, sendo essa parte incrementada nos ciclos seguintes por novas partes até o pleno atendimento do escopo definido para o projeto (Pressman, 2006 p.40).

Uma forma de se realizar desenvolvimento de forma iterativa e incremental que se torna cada vez mais popular são os chamados métodos Ágeis, que segundo Highsmith (2004, p.120), são métodos que focam principalmente em dois conceitos: a entrega de software executável, junto à eficiência no trabalho em equipe.

Dois métodos são frequentemente citados quando se discute sobre métodos ágeis, o *Scrum* e o *Extreme Programming* (XP). Ambos são muito discutidos quanto seus benefícios em comparação às abordagens tradicionais de

desenvolvimento, assim como formas de se aperfeiçoar cada método individualmente.

O *Scrum* é um método de desenvolvimento de produtos que consiste de papéis, eventos, artefatos e regras usadas para gerenciamento, a fim de maximizar a produtividade e o valor do esforço gerado no desenvolvimento (Sutherland e Schwaber, 2011, p.7).

O XP é um método ágil que consiste de práticas que visam garantir o desenvolvimento de produtos de alta-qualidade, principalmente em atividades diretamente ligadas à engenharia, como por exemplo, a Programação em Pares e o TDD (Desenvolvimento Orientado a Testes) (Mar e Schwaber, 2002, p.1).

De acordo com Mar e Schwaber (2002, p.1), havia controvérsias sobre os princípios do *Scrum* e XP no passado, sendo que equipes que usavam determinado método tendiam a depreciar o outro por detectar falta de determinada prática a qual estava habituada em outro modelo. Isso era causado por ambos os métodos possuírem áreas de atuação diferentes, sendo o *Scrum* um foco voltado para o Gerenciamento e Organização, e o XP voltado principalmente para práticas de Desenvolvimento e Codificação de Software.

Porém, com o passar do tempo, relatos de combinações entre os dois métodos começaram a surgir com bons resultados. Knickberg (2007, p. 79) relatou que o uso de programação em pares em uso conjunto com o *Scrum* traz como benefícios o aumento da qualidade do código gerado, assim como um aumento de foco na equipe.

Mar e Schwaber (2002, p.2) relataram em um estudo de caso, que determinada organização que usava o método XP possuía grandes dificuldades em gerar códigos executáveis com funcionalidade realmente aderente aos requisitos, e a partir da adição do *Scrum* em um projeto que usava unicamente o XP, as equipes passaram a produzir códigos executáveis mais estáveis e funcionais,

assim como passou a ser possível notar o progresso do projeto a cada novo incremento, o que ocasionou ganhos expressivos de visibilidade do projeto perante aos seus *Stakeholders*.

Segundo Knickberg (2007,p.79), deve-se evitar simplesmente tornar mandatórias todas as práticas do XP em uma equipe, pois podem existir pessoas que tenham uma maior rejeição a determinada prática, sendo o melhor meio de disseminar tal prática seja o encorajamento do uso e deixar a disposição todos os recursos necessários para tal.

Sendo assim, é possível identificar uma compatibilidade entre ambos os métodos, tornando possível o uso de forma conjunta a fim de se obter melhores resultados do que se aplicados de forma isolada.

1.2. Objetivo

O objetivo deste trabalho é analisar e propor uma integração do método ágil XP ao *Scrum*. Esta análise é fundamentada em discussões encontradas na literatura a fim de agregar uma maior robustez em uma proposta de modelo integrado entre os dois métodos.

1.3. Justificativa

Muito tem sido discutido sobre os benefícios gerados por esta integração, como Krishna e Basu (2011, p.2), que apontam que práticas de engenharia contidas no XP como, por exemplo, o TDD e a Refatoração podem ser incluídas dentro de cada iteração do *Scrum* proporcionando melhorias relativas à qualidade e produtividade.

Já Sutherland (2008, p.31), afirmou em um estudo de caso que equipes que usam *Scrum* e XP simultaneamente conseguem trabalhar de uma forma otimizada, assim como apontou deficiências em ambos os métodos de forma

isolada, sugerindo posteriormente a união de ambas as práticas como solução para determinados pontos.

Martin (2010) indicou que a alta-capacidade de se gerar códigos rapidamente seguindo o método *Scrum,* pode trazer problemas em sistemas a médio e longo prazo, e sugeriu também o uso de técnicas do XP para obter uma melhor escalabilidade do software.

Neste sentido existem trabalhos que indicam que a integração dos métodos *Scrum* e XP não é somente possível como recomendada, porém falta discutir como realizar esta integração, sendo este o ponto principal do trabalho.

1.4. Estrutura do Trabalho

Capítulo 1. INTRODUÇÃO
Este capítulo apresenta o objetivo geral do trabalho, assim como suas motivações, e discussões relacionadas sobre o tema propostos.

Capítulo 2. SCRUM E XP
Este capítulo apresenta um breve resumo dos pontos principais sobre os métodos *Scrum* e XP (*Extreme Programming*).

Capítulo 3. EXPERIÊNCIAS DE INTEGRAÇÃO DE SCRUM E XP
Este capítulo apresenta relatos de autores que discutem a viabilidade da integração entre o *Scrum* e XP.

Capítulo 4. ANÁLISE E PROPOSTA DE INTEGRAÇÃO XP NO SCRUM
Este capítulo apresenta uma proposta de integração entre os dois métodos, bem como os critérios usados para esta proposta.

Capítulo 5. CONSIDERAÇÕES FINAIS

Este capítulo descreve as conclusões do trabalho, as contribuições assim como uma sugestão a trabalhos futuros relacionados ao tema.

REFERÊNCIAS

Relaciona as fontes usadas ao longo do trabalho.

2. SCRUM E XP

Neste capítulo é apresentado um breve resumo sobre pontos importantes sobre os métodos *Scrum* e XP, pontos estes posteriormente discutidos durante a proposta de integração entre os métodos.

2.1. Scrum

Nesta seção será apresentado o método *Scrum,* sendo apresentados seus artefatos, papéis e principais eventos.

O *Scrum* foi criado com base em teorias de sistemas e práticas usadas pela indústria automobilística japonesa. O termo *Scrum* foi citado por Takeuchi e Nonaka (1986, p. 2) pela primeira vez no artigo chamado "The New Product Development Game" disponibilizado na revista Harvard Business Review de 1986. Os autores optaram pelo nome *Scrum* como uma alusão a uma determinada jogada que é realizada no jogo de Rúgbi, em que todo o time realiza uma formação em equipe em que todos os jogadores se movimentam juntos, significando que as equipes de desenvolvimento precisavam ser auto-organizadas e multidisciplinares (Sutherland, 2012, p. 5).

Cohn (2012) preferiu não categorizar o *Scrum* como um processo ou uma metodologia, mas sim como um *framework* (um conjunto de conceitos usado para resolver um problema de um domínio específico), que procura realizar o desenvolvimento de software usando abordagens ágeis, ou seja, ele não procura descrever como tudo precisa ser feito durante um projeto, mas sim fornecer diretrizes de como realizar o desenvolvimento ágil de software.

Segundo Sutherland e Schwaber (2011, p.4), o *Scrum* usa um modelo iterativo e incremental como forma de controlar os riscos e diminuir o impacto na alteração de requisitos ao longo do projeto. Além disso, conforme a experiência entre os membros da equipe aumenta no que se diz respeito ao

Scrum, mais eficaz o método se torna, colhendo melhores resultados do que em equipes iniciantes no método.

2.1.1. Artefatos do Scrum

Durante o ciclo de vida do *Scrum*, alguns artefatos são criados com o propósito de tornar as informações relativas ao projeto transparentes para todos envolvidos, sendo que deve-se manter todos os artefatos visíveis e disponíveis para toda a equipe. Sutherland e Schwaber (2011, p.12)

Segundo o *Scrum Guide* de Sutherland e Schwaber (2011, p.12) os artefatos do *Scrum* são os seguintes:

a) *Backlog* do Produto

O *Backlog* do Produto armazena uma lista de todas as funções que ainda não foram implementadas pela equipe, mas que são desejadas pelo cliente representando pelo *Product Owner*.

De acordo com Cohn (2009, p.255), diferente de modelos clássicos de desenvolvimento de software, o *Scrum* não trabalha com todos os requisitos levantados já no início do projeto. Os requisitos são vistos como dinâmicos, ou em alguns casos, são somente detalhados em fases posteriores do projeto quando estão enfim priorizados pelo cliente.

Sutherland e Schwaber (2011, p.13), afirmam que costuma-se ordenar os itens do *Backlog* por seu valor, risco, prioridade e necessidade, sendo que os itens de prioridade mais alta devem possuir um maior grau de detalhamento se comparado com os demais, pois provavelmente serão implementados primeiro que os demais.

b) *Backlog* da *Sprint*

Com base na lista de funções existente no *Backlog* do Produto, um ou mais itens são selecionados para serem construídos na próxima iteração. Todos os itens selecionados devem ser concluídos até o fim da *Sprint* (definida na p. 19), portanto, deve se ter certa cautela de quais e quantos itens serão incluídos em uma *Sprint*, pois isso representaria que a equipe está se comprometendo a entregar tudo no final desta iteração.

Para fazer parte da lista da *Backlog* da *Sprint*, os itens da *Backlog* do Produto devem ser subdivididos em tarefas. Como tais tarefas são definidas somente para a divisão de atividades da equipe durante a iteração, o *Product Owner* não deve manifestar interesse nesta lista de tarefas, tendo em vista que o foco principal dele é a priorização de funções do software. (Sutherland, 2010, p.22)

A equipe deve atualizar sempre que possível a *Backlog* de *Sprint*. Toda a equipe pode acompanhar o progresso da *Sprint* assim como estimar qual é o esforço necessário para se concluir tais atividades.

c) Incremento de Produto

No fim de toda iteração, um incremento deve ser entregue. A entrega deve conter exatamente todos os itens selecionados na *Backlog* da *Sprint* prontos. Um ponto importante dentro do *Scrum* é que a definição do termo "Pronto" deve ser discutida e compartilhada entre todos os envolvidos do projeto a fim de se evitar diferenças de critério entre as partes (Sutherland, 2010, p.22).

Segundo Sutherland e Schwaber (2011, p.15), no fim da *Sprint*, o *Product Owner* deve liberar o incremento, declarando-o pronto de fato, e assim ser adicionado junto aos demais incrementos onde tudo será posteriormente testado a fim de se garantir uma maior qualidade.

2.1.2. Papéis do Scrum

O *Scrum* trabalha com divisão de responsabilidades entre diferentes papéis ao longo de seu ciclo de vida. Segundo o *Scrum Guide* de Sutherland e Schwaber (2011, p. 5), os papéis existentes no *Scrum* são os seguintes:

a) ***Product Owner*****:** É o papel responsável por gerenciar o *Backlog* do Produto, assim como tomar decisões pertinentes a priorização de funções a serem desenvolvidas. Este papel geralmente é desempenhado pelo próprio cliente interessado no resultado do produto. Recomenda-se existir um *Product Owner* fixo junto à equipe durante o projeto para rápidas decisões que afetem a equipe de desenvolvimento.

b) ***Scrum Master*****:** É o Papel responsável por disseminar o uso correto das práticas do *Scrum* e corrigir possíveis desvios na aplicação de seus métodos. Ele realiza um elo entre os Clientes e a Equipe de Desenvolvimento, ficando responsável pela remoção de impedimentos que podem atrapalhar a equipe ao longo do projeto e consiga manter o foco em suas atividades a fim de garantir a produtividade da equipe.

c) Equipe: Papel responsável pela execução de todo trabalho a ser realizado no ciclo de desenvolvimento do produto, além de estar presente também em etapas posteriores como durante as estimativas de esforço. A equipe deve ser auto gerenciável e auto organizada, isso porque o *Scrum Master* não tende a ser um líder da equipe, sendo a própria equipe responsável pela divisão de tarefas e a decisão de qual é a melhor forma de se trabalhar.

2.1.3. Eventos do Scrum

a) *Sprint*

A *Sprint* nada mais é do que a divisão de trabalho do *Scrum*. São ciclos que ocorrem um após o outro, sempre com o objetivo de entregar os itens contidos na *Backlog* de *Sprint*.

A duração de uma *Sprint* pode variar dependendo do desejo da equipe e das características do projeto, porém uma vez este valor estabelecido, a quantidade de dias de uma *Sprint* deve ser sempre fixa, estando às atividades completas ou não. Para Cohn (2012), a duração de uma *Sprint* deve ser menor do que um mês, de preferência ser medida em semanas. Na maioria das vezes equipes optam por *Sprints* de duas semanas de duração.

Dentro da *Sprint*, é realizada grande parte das atividades ligadas ao desenvolvimento, como a codificação e testes, porém o *Scrum* não possui diretrizes de como esse trabalho deve ser realizado, deixando a cargo da equipe como todas essas atividades serão estruturadas e realizadas.

O *Scrum Guide* de Sutherland e Schwaber (2011, p. 8) determina alguns pontos importantes que devem ser seguidos dentro de uma *Sprint*:

- Mudanças de escopo não devem ser realizadas durante uma *Sprint*, elas devem sim ocorrer conforme a necessidade, porém sempre durante a etapa de formação da próxima *Backlog* de *Sprint*;
- Deve-se manter a equipe de desenvolvimento fixa durante a duração da *Sprint*;
- A qualidade deve estar de preferência contida dentro da definição de "Pronto" acordado entre todos envolvidos do projeto, ela não pode ser renegociada durante uma *Sprint*;

- Embora seja um fato relativamente raro, uma *Sprint* pode ser cancelada antes de seu período pré-determinado ser finalizado caso os itens contidos na *Backlog* da *Sprint* não sejam mais necessários. Porém, somente o *Product Owner* tem a autoridade necessária para tal.

b) Reunião de Planejamento da *Sprint*

Antes do início de toda e qualquer *Sprint*, o *Scrum* estabelece que seja realizada uma reunião entre os envolvidos do projeto (geralmente entre o *Product Owner*, o *Scrum Master* e a Equipe) Sutherland e Schwaber (2011, p.9).

Sutherland e Schwaber (2011, p.9) dividem a Reunião de Planejamento de *Sprint* em duas etapas:

- Primeiramente, são discutidos quais itens da lista de *Backlog* do Produto será incluída na *Backlog* desta *Sprint*. O *Product Owner* prioriza todos os itens, e a equipe de desenvolvimento determina quantos destes itens são de fato possíveis de serem entregues na *Sprint*.

- Na segunda etapa da reunião, ocorre uma reunião interna entre a equipe e o *Scrum Master*, a fim de segmentar os requisitos da *Backlog* da *Sprint*, e estabelecer um conjunto de tarefas que serão realizados durante esse período de tempo.

A duração desta reunião é proporcional à duração da *Sprint.* Por exemplo, em uma *Sprint* de duas semanas a duração recomendada para reunião é de quatro horas, já para uma *Sprint* de um mês, o recomendado é que a reunião seja de até oito horas (Cohn, 2012).

c) Reunião Diária

Durante todos os dias de uma *Sprint*, o *Scrum* adota que sejam realizadas reuniões como a primeira tarefa do dia. Recomenda-se que estas reuniões sejam bem dinâmicas e sejam realizadas em até 15 minutos. Para isso muitas equipes optam por realizar estas reuniões em pé, o que diminui a chance da reunião se prolongar mais do que o previsto.

Todos os membros da equipe devem basicamente responder as três seguintes perguntas:

- O que foi realizado desde a última reunião;
- O que será realizado hoje;
- Quais são os possíveis impedimentos que podem dificultar as tarefas realizadas hoje.

O *Scrum Master* é um participante primordial desta reunião, com base nos impedimentos levantados pela equipe, ele procurará remover todos os impedimentos possíveis a fim de manter o foco da equipe em realizar as tarefas acordadas (Sutherland 2010, p.16).

Para Cohn (2012) as reuniões diárias não possuem como propósito serem reuniões de reportar *status* de projeto, seu foco é unicamente em garantir o pleno andamento das atividades da equipe. Uma possível participação de gerentes pode acabar pressionando a equipe, por isso recomenda-se que as reuniões sejam realizadas somente com a presença da Equipe, do *Scrum Master* e caso possível do *Product Owner*.

Além disso, as Reuniões Diárias são primordiais para manter as equipes auto gerenciadas de forma eficaz, sendo que com base no que é reportado diariamente, a própria equipe pode detectar pessoas que estão sobrecarregadas, e também pessoas com parte do tempo vago aguardando

alguma atividade, e ir reajustando a disposição da equipe temporariamente, a fim de otimizar ainda mais o trabalho realizado dentro da *Sprint*.

d) Revisão da *Sprint*

Após o fim de toda e qualquer *Sprint* ocorre à reunião de Revisão da *Sprint*. Esta reunião tem o objetivo de inspecionar o incremento que foi criado e realizar as adaptações necessárias no *Backlog* do Produto.

Segundo Sutherland e Schwaber (2011, p. 11), a reunião de Revisão de *Sprint* deve ser realizada da seguinte maneira:

1- O *Product Owner* identifica quais itens da *Backlog* da *Sprint* foram concluídos, e quais não foram;
2- A equipe apresenta pontos positivos e negativos ao longo da *Sprint*, assim como eventuais problemas foram solucionados;
3- A equipe apresenta o incremento que foi criado na *Sprint*, e esclarece possíveis dúvidas que o *Product Owner* tenha;
4- O *Product Owner* analisa o *Backlog* do Produto após a entrega do último incremento, podendo realizar alterações no mesmo;
5- Todos os presentes na reunião expõem opiniões do que deve ser realizado a seguir, um insumo que será utilizado na próxima Reunião de Planejamento de *Sprint*.

Assim como as demais reuniões do *Scrum*, esta reunião deve se manter como um encontro informal a fim de se obter um melhor aproveitamento do tempo, e com sua duração também pré-estabelecida conforme a duração da *Sprint*.

e) Retrospectiva da *Sprint*

Logo após a Revisão da *Sprint*, e antes da próxima Reunião de Planejamento de *Sprint*, ocorre a Retrospectiva da *Sprint*.

Para Sutherland e Schwaber (2011, p.11), esta reunião visa uma melhoria contínua no processo de *Scrum* praticado pela equipe, em que o *Scrum Master* deve incentivar a equipe a planejar novos meios de aumentar a produtividade e a qualidade do produto gerado, assim como também a própria relação entre as pessoas envolvidas na *Sprint* anterior.

Cohn (2012) afirma que por mais que uma equipe pareça estar trabalhando muito bem, sempre existe espaço para possíveis melhorias em quaisquer aspectos no que se diz a respeito de processos, ferramentas e pessoas, sendo que quaisquer pontos de melhoria identificados pela equipe já podem ser aplicadas na próxima *Sprint* a ser realizada a fim de se otimizar os resultados gerados pela equipe.

2.1.4. Ciclo de Vida do Scrum

De acordo com o ciclo de vida representado por (Sutherland, 2010, p.11) na Figura 1, o *Scrum* acontece da seguinte maneira:

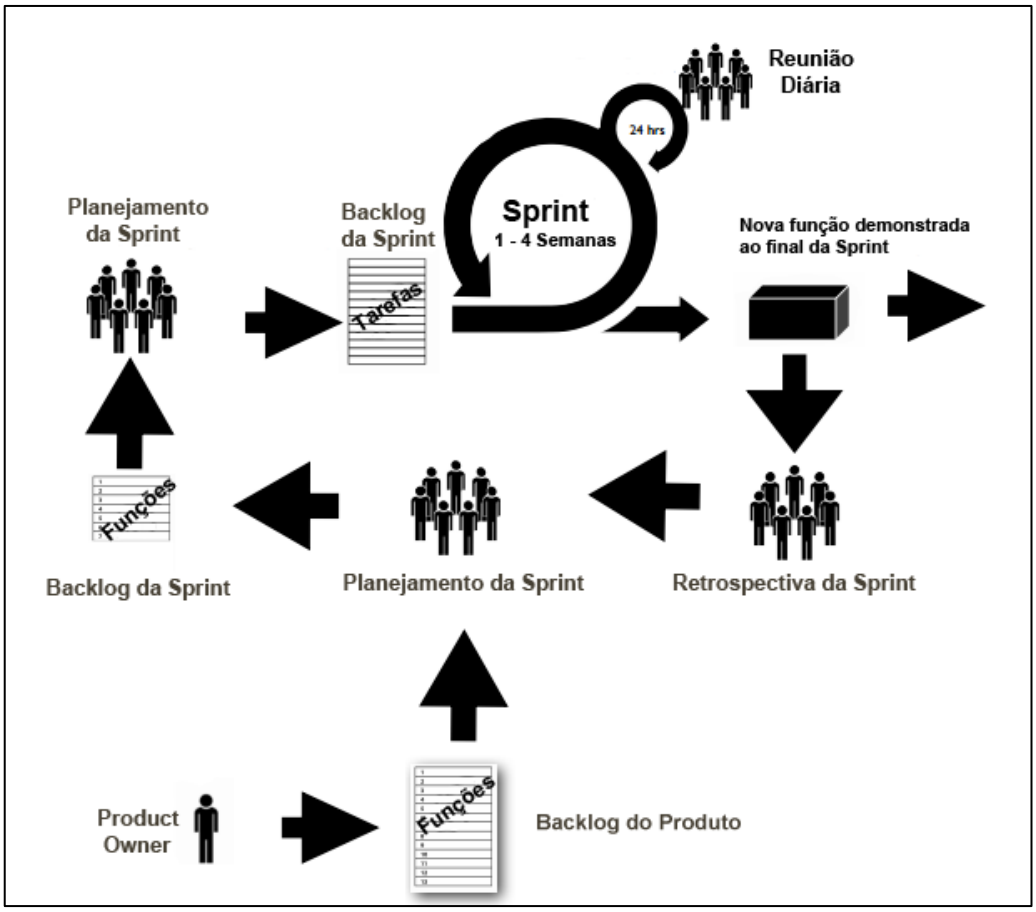

Figura 1: Ciclo de Vida do Scrum - Adaptado de (Sutherland, 2010, p.11)

1- O *Product Owner* lista as funções necessárias do produto, e as armazena na *Backlog* do Produto;
2- É realizada a Reunião de Planejamento da *Sprint* juntamente com o *Product Owner*, onde os itens contidos na *Backlog* do Produto são selecionados para constituírem o *Backlog* da *Sprint*;
3- Com o *Backlog* da *Sprint* em mãos, é realizada uma nova reunião de planejamento da *Sprint*, na qual a equipe procura discutir como será a divisão de tarefas a fim de se atender a todas as funções planejadas para a próxima *Sprint*;
4- É iniciada a *Sprint*, em que diariamente a equipe trabalha na construção do incremento do produto, além de realizar Reuniões Diárias durante o período da *Sprint*;
5- No fim da *Sprint*, é realizada a reunião de Revisão da *Sprint*, em que o incremento fruto da *Sprint* é apresentado ao *Product Owner* pela equipe;

6- É realizada a Retrospectiva da *Sprint*, a fim de se otimizar o processo do *Scrum* nas *Sprint*s seguintes.

2.2. Extreme Programming (XP)

Nesta seção é apresentado brevemente o Método Ágil XP, focando principalmente em seu conjunto de práticas, seus usos e benefícios.

O *Extreme Programming* (XP) é um método de desenvolvimento de sistemas que teve seus primeiros fundamentos baseados em técnicas de desenvolvimento na linguagem Smalltalk nos anos 80, sendo que muitas das atuais práticas do XP acabaram sendo influenciadas pela comunidade Smalltalk (Beck e Andres, 2004).

Segundo Beck (1999), o XP tem como objetivo reduzir os riscos de projetos, aumentar o grau de resposta às mudanças de negócio, aumentar a produtividade, além de tornar o desenvolvimento de software uma atividade prazerosa para a equipe. Beck (1999) também afirma que o XP é um modelo com foco em atividades ligadas ao desenvolvimento. Ele basicamente utiliza um conjunto de doze práticas interligadas que são usadas ao longo de um ciclo de vida iterativo e incremental.

Para Beck e Andres (2004), as práticas do XP não devem ser realizadas de forma mandatória, simplesmente por precisarem ser feitas devido a ordens superiores à equipe. Para que elas passem a se tornar efetivas, a equipe deve acreditar na importância e valor de que cada prática pode proporcionar de benefícios para a equipe e para a qualidade do produto a ser desenvolvido.

2.2.1. Práticas do XP

O método XP é composto através de 12 diferentes práticas, sendo que de acordo com Franco (2007,p. 32) tais práticas podem ser agrupadas em quatro

diferentes grupos de acordo com sua finalidade: Produto, Equipe, Processo e Codificação, representados através da Figura 2.

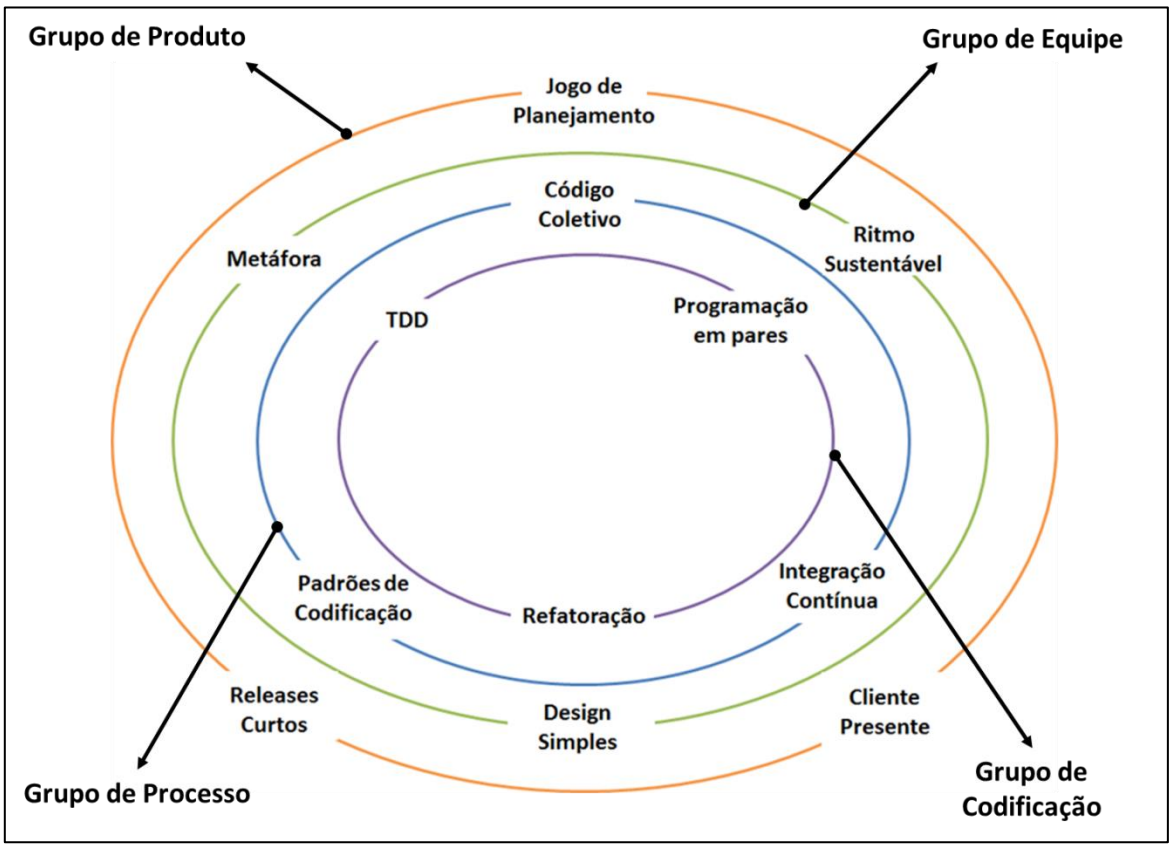

Figura 2: Agrupamento de práticas do XP - Adaptado de Franco (2007,p.32)

As definições das práticas seguintes são descritas por Beck (1999):

a) Jogo de Planejamento

O Jogo de Planejamento visa que as áreas de negócio e desenvolvimento cheguem a um acordo quanto a determinadas decisões ligadas a priorizações de requisitos.

Antes de toda iteração, o cliente se reúne com a equipe de desenvolvimento para realizar a priorização de funções. Primeiramente, o cliente estabelece as prioridades, e a equipe de desenvolvimento estima o grau de esforço para que este requisito seja entregue. Caso não exista um consenso entre ambas às

partes, o escopo original pode ser alterado a fim de permitir que a entrega possa ocorrer no prazo desejado pelo cliente, por exemplo.

b) Releases Curtos

A prática de releases curtos, visa ser uma diretriz para que os entregáveis do software sejam os menores possíveis, porém mantendo-os funcionais. O uso de Releases Curtos faz com que o cliente consiga utilizar o sistema antes de uma entrega final do software desejado; Isso possibilita que possíveis falhas sejam identificadas de forma antecipada, onde podem ser devidamente corrigidas ou mesmo ocorrer mudanças na função definida anteriormente.

c) Cliente no Local

O cliente deve fazer fisicamente parte da equipe de desenvolvimento a fim de proporcionar um *feedback* contínuo durante o ciclo de desenvolvimento. Este *feedback* em tempo real acaba diminuindo incertezas e indefinições que podem surgir na equipe de desenvolvimento durante uma iteração, assim como aumenta a confiança entre as áreas envolvidas no projeto.

d) Design Simples

O desenvolvimento deve ser realizado da maneira mais simples possível para atender os requisitos definidos pelo usuário. Caso sejam identificadas partes complexas no sistema, elas devem ser removidas o mais rápido possível.

O uso do design simples tende a diminuir o número de funções do software que sequer são utilizadas pelo cliente, agregando maior valor nos entregáveis priorizados pelo próprio cliente.

e) Metáfora

O XP utiliza o conceito de metáforas para manter a integridade conceitual do sistema. Ao invés de utilizar-se diversos termos técnicos, metáforas são definidas e utilizadas.

Frequentemente os requisitos definidos pelos usuários são escritos no formato de estórias, que representam literalmente pequenas estórias de como o cliente utilizaria o sistema.

f) Ritmo Sustentável

O XP define que cada membro da equipe não deva trabalhar mais do que quarenta horas por semana.

Além do aumento do custo de projeto devido horas-extras, muitos são os relatos de que horas-extras contribuem diretamente para uma má qualidade do software entregue.

g) Integração Contínua

Equipes de desenvolvimento normalmente possuem programadores trabalhando em paralelo em diferentes partes do sistema, sendo que após suas respectivas partes terem sido desenvolvidas e testadas, são realizados os testes de integração entre as diferentes partes do sistema. A Integração Contínua do XP determina que esporadicamente (geralmente diversas vezes ao dia), sejam geradas compilações do sistema com as últimas versões dos módulos que vão sendo implementados pela equipe.

Esta integração contínua antecipa possíveis falhas de desenvolvimento evitando que tais falhas sejam encontradas somente em etapas finais de desenvolvimento, etapas estas em que o risco de se atrasar o projeto é consideravelmente maior.

h) Padrão de Codificação

A adoção de padrões de codificação dentro do XP visa obter uma maior consistência dentro do código produzido pela equipe, proporcionando uma simplificação na compreensão do código pelos demais membros da equipe, assim como facilitará futuras manutenções do sistema.

Não existe um padrão de código específico indicado pelo XP, ele deve ser escolhido pela equipe visando às características do projeto.

i) Código Coletivo

Esta prática tem como objetivo tornar aberto a toda a equipe o repositório de códigos fontes, e não somente isso, mas permitir que todos possam realizar alterações e melhorias em códigos escritos por outro membro.

O código coletivo auxilia a equipe se tornar mais completa, pois assim todos os desenvolvedores se habituam a diferentes partes do sistema, proporcionando um compartilhamento do conhecimento. Outro ponto importante é que para manter a qualidade do sistema ao utilizar o código coletivo em uma equipe, é altamente recomendado existir um mecanismo de testes automatizados para garantir que não ocorram efeitos colaterais.

j) Refatoração

A prática de refatoração do XP tem como objetivo otimizar a estrutura de código fonte a fim de se remover partes desnecessárias diminuindo a complexidade do código, facilitando futuras manutenções, assim como diminuindo a probabilidade do aparecimento de falhas de sistema devido a trechos de códigos não utilizados.

A refatoração realiza alterações em códigos existentes sem alterar seu comportamento, ou seja, toda e qualquer alteração realizada pela refatoração não deve ser notada pelo usuário do sistema.

Equipes de XP que utilizam a refatoração normalmente não separam um tempo durante projetos somente para realizar a refatoração, pequenas ações de refatoração ocorrem durante todo o tempo. Por exemplo, durante o desenvolvimento de novas funções no sistema, podem ser detectados códigos que realizam a mesma função em diferentes pontos do sistema, portanto o programador que estiver trabalhando neste ponto do código pode unificar reaproveitar o código já existente e utilizar o mesmo método em ambos os pontos do código.

Apesar de todos os benefícios da refatoração, existe sempre o risco de que alterações causem efeitos colaterais no sistema. Para isso é de suma

importância que a equipe de desenvolvimento tenha como realizar testes automatizados, para sempre se certificar de que as refatorações não estejam criando novos defeitos no sistema.

k) Programação em Pares

A prática de Programação em Pares consiste em que dois desenvolvedores trabalhem juntos durante as etapas de codificação e teste em um mesmo computador. Com o uso desta prática, todo e qualquer código desenvolvido no XP acaba sendo revisado em tempo real, auxiliando na redução de possíveis falhas durante o desenvolvimento.

Durante a Programação em Pares, um desenvolvedor fica responsável pelo desenvolvimento, ou seja, ele que acaba de fato codificando todo o código, já o segundo desenvolvedor é responsável pela inspeção, sendo que periodicamente os papéis são invertidos entre os desenvolvedores.

Embora a uma primeira vista, alocar dois desenvolvedores para uma mesma atividade de desenvolvimento pareça ser custoso para o projeto, à identificação de defeitos antecipadamente em um sistema acaba reduzindo consideravelmente futuros gastos com manutenções e correções.

Além de diminuir consideravelmente a quantidade de defeitos durante o desenvolvimento, a programação em pares também acaba gerando soluções mais simples e estruturadas diminuindo o tempo de desenvolvimento posterior.

Outro benefício do uso desta prática é a distribuição de conhecimento entre a equipe, conhecimentos relativos desde a própria tecnologia e linguagem no qual o sistema é desenvolvido, até mesmo particularidades do sistema, acaba sendo distribuído de forma transparente pela equipe durante a programação em pares, evitando um gasto de tempo exclusivo com dois recursos parados em projetos para realizar uma passagem de conhecimento.

l) TDD

Visando diminuir a quantidade de defeitos, o XP possui dentre suas práticas o *TDD* (*Test Driven Development)* ou também Desenvolvimento Orientado a Testes. Os testes contemplados pelo XP se resumem a dois tipos: os testes de unidade e os testes de aceitação.

Os testes de unidade visam garantir que os componentes do sistema estejam funcionando da maneira esperada. Estes testes devem ser escritos antes que o desenvolvimento do componente em questão seja iniciado. Isso é necessário para que no próprio momento da codificação o desenvolvedor já saiba se o componente está de acordo com o especificado ou não.

A abordagem de testes automatizados também faz parte do *TDD* proposto pelo XP. Eles devem garantir que todos os componentes continuem funcionando após toda e qualquer alteração realizada.

Já os testes de aceitação são escritos pelo próprio cliente a fim de certificarem de que as funções implementadas no sistema estejam de acordo com o que foi solicitado previamente. Estes testes devem serem planejados com o sistema sendo uma "caixa-preta" e com base em determinados parâmetros de entrada, existe um resultado esperado para o teste.

O *TDD* está fortemente ligado com as práticas de Refatoração e Design Simples, isso porque durante a etapa de codificação, o desenvolvedor deve codificar da maneira mais simples possível até que o resultado do teste planejado pelo *TDD* seja atendido, após isso a refatoração deve ser realizada para uma melhor estruturação e reusabilidade do código fonte.

2.2.2. Inter-relação entre práticas

Segundo Beck (1999), o uso de práticas do XP de forma isolada, ou seja, sem o uso conjunto das demais práticas do próprio XP, acaba diminuindo sua eficiência. Isso é explicado devido a que todas as práticas do XP possuírem

relações com outras práticas. O uso de práticas relacionadas de forma conjunta reforça a estrutura base do XP.

Na figura 3 de Beck (1999), está representada a relação entre todas as práticas do XP, onde a linha entre duas práticas representam que ambas as práticas se reforçam mutuamente.

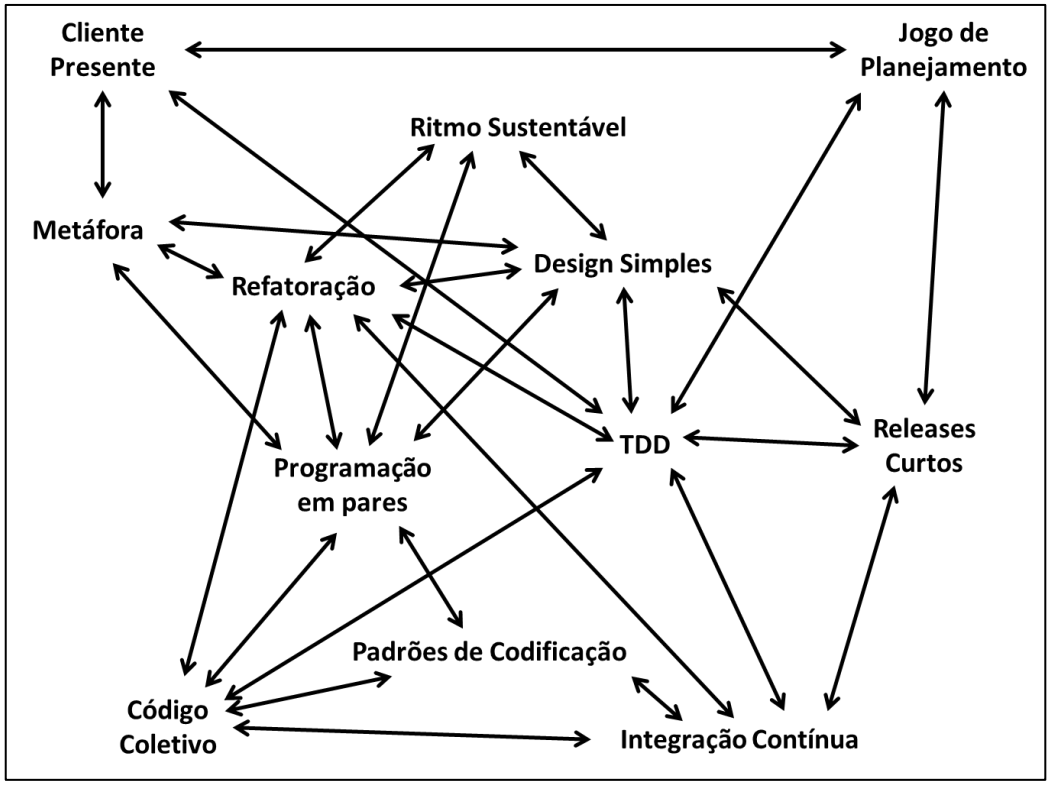

Figura 3: Relação entre práticas do XP – Adaptado de BECK (1999)

2.3. Considerações do Capítulo

Baseado nas definições dos métodos *Scrum* e XP, pode-se notar que embora ambos utilizem uma abordagem voltada ao desenvolvimento ágil, existe uma diferença no escopo tratado em cada método.

O Scrum tem como seu maior foco, disciplinas ligadas ao gerenciamento de projetos de Software, utilizando uma abordagem ágil o que proporciona uma maior facilidade e maleabilidade ao lidar com alterações de requisitos, dentre demais vantagens citadas anteriormente.

Já o XP, embora também possua certas definições ligadas a gerenciamento de projetos, possui claramente um foco diferente do Scrum, sendo as disciplinas ligadas ao desenvolvimento bastante especificadas e detalhadas.

Partindo destas duas afirmações, torna-se coerente uma possível integração de áreas de conhecimento diferentes criando um modelo mais robusto e definido. Porém, é preciso identificar quais práticas do XP serão de fato integradas ao Scrum, evitando o uso de práticas sobrepostas e conflitantes entre os dois métodos. Para isso, no Capítulo 3 são apresentados trabalhos de autores com relatos de experiências de integrações de ambos os métodos que servem de base para a análise e a proposta de integração dos métodos neste trabalho.

3. EXPERIÊNCIAS DE INTEGRAÇÃO DE SCRUM E XP

Diversos autores já realizaram análises sobre a viabilidade do uso de práticas do XP ao *Scrum*. Neste capítulo, são apresentados pontos de vista de alguns destes autores a fim de posteriormente identificarem-se elementos que indiquem a possibilidade de uma proposta de integração.

Segundo Knickberg (2007, p.77), afirmar que *Scrum* e XP podem ser combinados trazendo bons resultados é discutível, devido ao enfoque de cada um dos métodos. O *Scrum* possui uma abordagem de gerenciamento e organização do projeto e o XP possui um foco em práticas diretamente ligadas ao desenvolvimento de software. Para Salo e Abrahamsson (2008, p.59), ambos os métodos seguem os princípios centrais de desenvolvimento ágil, o que facilitaria uma possível integração.

3.1. Experiência de Mar e Schwaber (2002)

Mar e Schwaber (2002, p.1) descrevem os resultados obtidos em um determinado projeto realizado para uma grande empresa do ramo energético. Esse projeto já havia falhado anteriormente algumas vezes, isso devido a grande complexidade de requisitos e informações consideradas.

Como o cliente anteriormente havia tido algumas experiências bem sucedidas usando algumas práticas do XP em projetos de engenharia e infraestrutura, foi vista uma oportunidade de se usar os métodos ágeis *Scrum* e XP juntos neste projeto.

O projeto usou uma combinação de práticas de ambos os métodos (Mar e Schwaber, 2002, p.2), conforme a Tabela 1.

Tabela 1: Práticas Selecionadas a partir da experiência de Mar e Schwaber

Scrum	**XP**
Sprints e Planejamento da *Sprint*	Design Simples
Reuniões Diárias	TDD
Backlog do Produto e *Product Owner*	Integração Contínua
Backlog da *Sprint*	Refatoração
	Programação em Pares

O processo *Scrum* foi usado como base para gerenciar as fases necessárias para o gerenciamento do projeto, além disso foram também inseridas algumas práticas do XP neste processo a fim de obter uma maior qualidade no software através da adoção de diretrizes de como se realizar o desenvolvimento dentro das *Sprints* do projeto.

Como a área cliente estava acostumada somente com o uso do XP em seus projetos anteriores, o uso das práticas do XP pelo *Scrum* acabou produzindo uma maior produtividade em comparação a uma implementação puramente XP. Para Mar e Schwaber (2002, p. 03), a grande razão para isso, foi realizar a seleção dos requisitos priorizados pelo cliente através do *Backlog* da *Sprint* a cada iteração, o que acaba diminuindo a quantidade de funções e artefatos desnecessários a serem gerados pela equipe de desenvolvimento.

Ao analisar inserção das práticas do XP no processo *Scrum*, foi realmente notada uma melhoria notável na qualidade da codificação se comparando a projetos puramente *Scrum*. Por se tratar de um desenvolvimento iterativo e incremental, o uso constante da Refatoração acabou trazendo um ganho gradual de qualidade à medida que cada *Sprint* era concluído.

Todas as práticas selecionadas do XP para o projeto visavam uma otimização na qualidade do produto, porém Mar e Schwaber (2002, p. 03) salientaram que cada uma das práticas possuía um impacto em somente determinado aspecto da qualidade do código.

Ainda segundo Mar e Schwaber (2002, p. 03), durante a Revisão da *Sprint*, a própria área cliente questionou como a equipe de desenvolvimento conseguiu realizar tantos progressos durante aquela *Sprint*, sendo que a equipe atribuiu esse progresso à melhora na comunicação, existindo uma maior proximidade junto à área cliente devido à participação do *Product Owner*, papel este contido originalmente no *Scrum*.

Como resultado final do projeto, foi notado que o uso combinado do *Scrum* juntamente a práticas do XP tiveram um impacto significativo na produtividade da equipe assim como na qualidade dos incrementos de software que foram gerados ao longo do projeto. Mar e Schwaber (2002, p.05), afirmaram que ambos os métodos são complementares, sendo que a integração de suas práticas acaba também causando impactos positivos na disciplina da equipe, desenvolvendo valores de gestão.

3.2. Experiência de Knickberg (2007)

Já Knickberg (2007, p.78) afirmou que a integração do *Scrum* com o XP, além de ser possível, era altamente recomendada. Ele também relatou que suas equipes usavam algumas práticas consideradas como chaves do XP juntamente com o *Scrum*. Mas também complementou dizendo que nem todas suas equipes usavam todas as práticas do XP devido às particularidades de cada projeto.

Ao se tratar de Programação em Pares, por exemplo, Knickberg (2007, p.78) alertou que deve-se evitar a imposição da prática em seu formato original a todo custo, isso porque tanto a Programação em Pares como outras práticas do XP podem não parecer efetivas em um primeiro momento para uma equipe não acostumada com elas. Para ele, o melhor meio de se amadurecer determinada prática em uma equipe é fornecer as ferramentas e recursos necessários para que a própria equipe crie a curiosidade e experimente por si mesma.

Knickberg (2007, p.79) também relatou que o TDD em sua visão, era em sua visão a prática do XP mais relevante em prol de qualidade, embora também tenha citado que o uso do TDD levava certo tempo a ser aprendido pela equipe, porém o resultado acabou sempre sendo recompensado através da qualidade obtida através da aplicação de testes automatizados.

3.3. Experiência de Willians et al. (2011)

Já Willians et al. (2011, p.1) apresentaram experiências de três equipes de desenvolvimento da Microsoft que incluíram práticas de desenvolvimento em um processo seguindo o *Scrum*, sendo que entre as práticas selecionadas estavam algumas práticas do XP.

Dentro do Planejamento da *Sprint*, originalmente do *Scrum*, foi inserida a prática do Jogo de Planejamento, sendo necessária a participação de todas as áreas envolvidas no desenvolvimento do produto. Além de um representante da área cliente, gerentes de projeto, desenvolvedores, analistas de testes, e engenheiros de usabilidade, participaram do Jogo de Planejamento a fim de se obter uma estimativa mais precisa do esforço necessário por funcionalidade. (Willians et al. , 2011, p. 2)

Quando o Jogo de Planejamento entrava em um estado de *deadlock*, ou seja, quando ambas as partes não chegavam a um consenso, era sinal de que o *Product Owner,* ainda não havia descrito totalmente o trabalho a ser desenvolvido, sendo um forte indicativo para este item ficar de fora da próxima *Sprint* até que as equipes de negócio e desenvolvimento tivessem um melhor entendimento do requisito.

Willians et al. (2011, p.2) relataram também que o Jogo de Planejamento diminuiu significativamente o erro nas estimativas de esforço das equipes além

de aumentar o conhecimento compartilhado entre os envolvidos no projeto e também identificar possíveis alternativas para se realizar o mesmo objetivo.

Outra prática do XP usada pelas equipes analisadas foi a Integração Contínua. Ao menos uma vez ao dia, cada desenvolvedor deveria enviar suas alterações para o repositório oficial do projeto, cada envio de código fonte para o repositório automaticamente iniciava um processo de geração de uma nova versão do sistema, sendo que todas as versões também passavam por uma série de testes de unidade automatizados. No fim deste processo, os desenvolvedores recebiam e-mails com os resultados dos testes de unidade relativos a ultima versão gerada. (Willians et al. , 2011, p.5)

Para possibilitar que os testes automatizados fossem realizados no processo de Integração Contínua, o desenvolvimento dos projetos em questão abordados por Willians et al. (2011, p.5) foi sempre baseado nas diretrizes do TDD.

3.4. Experiência de Sutherland et al. (2006)

Em uma análise do uso do *Scrum* em ambientes distribuídos, Sutherland et al. (2006, p.1), relataram uma integração de práticas do XP em um processo também baseado no *Scrum*.

As práticas do XP não faziam parte do processo padrão seguido pela empresa, ou seja, não eram usadas em todos os projetos e em todas as ocasiões, e sim usadas sob demanda em situações especificas ao longo dos projetos relatados.

No estudo de caso de relatado por Sutherland et al. (2006, p.7), somente as três práticas do XP abaixo foram utilizadas:

Programação em Pares: Era realizada somente em trechos de desenvolvimento mais complexos de determinadas funções de software consideradas como vitais para o projeto.

Refatoração: É planejado para ocorrer somente quando o projeto está próximo de ser concluído, e não em todas *Sprints*.

TDD: A abordagem do TDD era recomendada aos desenvolvedores, porém não era uma prática mandatória de ser seguida.

Embora não tenha sido o foco do trabalho discutir sobre a junção do XP ao Scrum, Sutherland et al.(2006) afirmaram que uma integração do Scrum ao XP pode ser realizadas até mesmo em grandes equipes, trabalhando inclusive de forma distribuída, proporcionando uma maior produtividade, maior qualidade e reduções nos riscos do projeto.

3.5. Experiência de Dinakar (2009)

Dinakar (2009, p.579) procurou em seu trabalho citar algumas boas práticas de desenvolvimento ágil a serem usadas juntamente com o *Scrum*. Sendo que dentre estas recomendações podem-se identificar princípios ligados diretamente a práticas originalmente contidas no XP.

Antes da adoção de boas práticas pela equipe analisada, houve uma tentativa inicial de se disseminar padrões de codificação entre a equipe, porém, Dinakar (2009, p.582) relatou que o controle da padronização de código fonte, que era realizado sempre em revisões em pares (um segundo analista verificando o trabalho realizado por outro membro da equipe) entre dois desenvolvedores, foi se tornando menos aderente aos padrões estabelecidos. Isto estava diretamente ligado ao fato da revisão em pares ser sempre realizada entre os mesmos desenvolvedores, o que acabou criando uma espécie de vício no propósito da revisão do código, diminuindo significativamente sua eficácia.

A mesma equipe em questão decidiu então incluir uma série de novas boas-práticas de desenvolvimento em seus projetos, começando com a

automatização de seus testes de unidade através das ferramentas PHPUnit e JUnit.

Este fato acabou indiretamente melhorando a aderência do código-fonte desenvolvido aos padrões de codificação estabelecidos pela organização, isto devido às alterações na estrutura do código a ser desenvolvido, para que o mesmo pudesse ser utilizado juntamente com as ferramentas de automação. Os benefícios destas mudanças foram altamente positivos e imediatos, o que acabou diminuindo significativamente o número de defeitos em testes de regressão. (Dinakar, 2009, p.582)

Já todos os novos códigos que eram desenvolvidos seguiam um estilo uniforme e padronizado independentemente do desenvolvedor que o criava. As revisões em pares passaram a ser possíveis de ser realizadas entre quaisquer desenvolvedores, já que todos estavam familiarizados com o estilo de codificação da equipe. (Dinakar, 2009, p.580)

Posteriormente, foi identificada uma oportunidade de se automatizar o processo de geração de versões de software, já que esta atividade vinha se tornando cada vez mais trabalhosa e custosa para a equipe de desenvolvimento devido ao crescente número de novas versões liberadas para cada componente do software.

Foi então que a ferramenta *CruiseControl* (ferramenta voltada à automação de geração de versões) passou a realizar este processo dentro do ciclo de desenvolvimento da equipe. Nas primeiras *Sprints* realizadas com o auxílio desta nova ferramenta, muitos problemas acabaram ocorrendo com as versões de software que passaram a ser geradas diariamente, isso devido a alguns erros ainda encontrados em padrões de codificação e nos testes automatizados, o que levou uma queda significativa na qualidade e estabilidade do software. Porém, foi consenso na equipe que estes problemas seriam temporários e de curto prazo, sendo então decido por manter-se a ideia de gerações diárias de

novas versões de software. (Dinakar 2009, p.581)

Segundo Dinakar (2009, p.583), o resultado passou a ser notado já nas *Sprints seguintes*, onde a automação de geração de versões de software juntamente com testes de regressão automatizados possibilitaram um ganho valioso de tempo que era anteriormente gasto pelos desenvolvedores. Os resultados obtidos foram tão significativos, que acabou influenciando a equipe de Garantia de Qualidade da mesma organização a adotar uma abordagem de testes automatizados em suas atividades pouco tempo depois.

3.6. Pesquisa de Salo e Abrahamsson (2008)

Salo e Abrahamsson (2008, p. 58) apresentaram uma pesquisa realizada entre profissionais ligados a empresas de desenvolvimento de software embarcado. Nesta pesquisa, Salo e Abrahamsson (2008, p. 58) procuravam mostrar através de um questionário, qual era o grau de adoção e experiência dos entrevistados em relação às práticas do *Scrum* e do XP.

Para cada prática, foi atribuída uma classificação definida pela seguinte escala: "usada sistematicamente", "usada na maioria da vezes", "usada às vezes", "usada raramente", "nunca usada", "não se aplica" e "não sei".

Salo e Abrahamsson (2008, p. 61) classificaram cada prática agrupando as duas melhores e as duas piores escalas citadas acima, sendo que o resultado da pesquisa gerou a Tabela 1.

Tabela 2: Relação de Práticas do XP mais usadas – Adaptado de Salo e Abrahamsson (2008, p.61)

Prática XP	Sistematicamente ou na maioria das vezes	Raramente ou nunca
Padrões de Codificação	60%	21%
Ritmo Sustentável	59%	26%
Integração Contínua	44%	31%
Código Coletivo	42%	48%
Refatoração	30%	18%
Jogo de Planejamento	28%	50%
Cliente Presente	24%	42%
Design Simples	22%	41%
TDD	18%	56%
Programação em Pares	15%	51%

Pode-se notar que duas dentre as mais relevantes práticas do XP, o TDD e a Programação em Pares figuraram entre as duas últimas posições desta pesquisa, o que significou que estas práticas não costumam ser usadas pelos profissionais entrevistados. Segundo Salo e Abrahamsson (2008, p. 61), isso se deve a tais práticas requererem uma maior mudança cultural no processo de desenvolvimento ao se comparar com as demais práticas que podem simplesmente serem adotadas sob demanda sem maiores dificuldades. A adoção em curto prazo de práticas mais complexas como o TDD e a Programação em Pares podem causar dificuldades para a equipe e na própria organização até que sejam plenamente compreendidas e seu resultado seja reconhecido entre todos os envolvidos no projeto.

3.7. Considerações do Capítulo

Diversos autores discutiram a viabilidade da integração entre o *Scrum* e o XP, assim como seus ganhos e possíveis dificuldades que foram encontradas especialmente em curto prazo em determinadas práticas.

Dentre todos os trabalhos analisados, foi detectada um padrão de que esta

integração é possível de ser realizada, porém pode-se notar que algumas das práticas do XP mesmo não sendo fortemente discutidas, acabam sendo mais citadas do que outras, sendo que algumas outras práticas não foram citadas por nenhum dos autores.

Com base nos trabalhos analisados acima, a Tabela 03 apresenta a quantidade de citações que cada prática do XP obteve. Somente no trabalho de Salo e Abrahamsson será utilizado o critério das 4 práticas com maiores índices de uso de acordo com sua pesquisa realizada, sendo estas 4 práticas consideradas como uma citação perante os demais trabalhos.

Este número de citações será um insumo para determinar em quais das práticas do XP podem ser incluídas em uma proposta de integração ao *Scrum* no Capítulo 4.

Tabela 3: Quantidade de Referências às práticas do XP

Prática do XP	**Mar e Schwaber (2002)**	**Knickberg (2007)**	**Willians et al. (2011)**	**Sutherland et al. (2006)**	**Dinakar (2009)**	**Salo e Abrahamsson (2008)**	**Total**
Integração Contínua	X	X	X	X	X	X	**6**
Desenvolvimento Dirigido a Testes		X	X	X	X		**4**
Código Coletivo		X	X	X		X	**4**
Programação em Pares	X	X		X			**3**
Refatoração	X	X		X			**3**
Padrões de Codificação		X			X	X	**3**
Design Simples	X	X					**2**
Ritmo Sustentável		X				X	**2**
Jogo de Planejamento			X				**1**
Metáfora							**0**
Releases Curtos							**0**
Cliente Presente							**0**

4. ANÁLISE E PROPOSTA DE INTEGRAÇÃO DE XP NO SCRUM

Com base nos relatos dos autores citados no capítulo 3, neste capítulo são definidas quais práticas do XP podem ser inseridas no *Scrum*, assim como é realizada uma proposta de como este modelo integrado de desenvolvimento pode ser realizado.

4.1. Seleção de Práticas do XP

Com base na Tabela 2 apresentada no Capítulo 3 é possível ter uma visão inicial de quais práticas foram mais relevantes dentre os trabalhos analisados devido ao seu número de citações.

Porém, foi notado que não seria adequado atribuir o mesmo peso de uma experiência relatada em um trabalho que visa discutir exclusivamente a possibilidade de integração entre o *Scrum* e o XP, do que outros trabalhos que somente citaram casos onde as práticas do XP foram inseridas no *Scrum*.

A fim de se possuir uma lista ordenada dentre as práticas do XP com maior probabilidade de serem elegíveis para serem incluídas na proposta de integração ao *Scrum*, os trabalhos analisados foram agrupados em quatro diferentes grupos de acordo com seu objetivo:

> **Grupo 1:** Neste grupo estão considerados todos os trabalhos que possuem como objetivo sugerir e analisar uma integração de práticas do XP ao *Scrum*, citando as vantagens proporcionadas do uso deste método integrado.
>
> **Grupo 2:** Neste grupo estão trabalhos que possuem uma abordagem utilizando diversas outras práticas, métodos ou processos de melhoria a serem incorporados em um modelo de desenvolvimento originalmente *Scrum*. Sendo que foram detectados neste grupo de trabalhos a

aplicação de uma ou mais práticas do XP na proposta dos autores selecionados.

Grupo 3: Neste grupo estão trabalhos que possuem uma análise do grau de adoção de práticas relativas a métodos ágeis, assim como também uma classificação da utilidade de cada prática. Os resultados foram obtidos de pesquisas quantitativas realizadas entre diversos profissionais de empresas ligadas diretamente ao desenvolvimento de software.

Grupo 4: Neste grupo estão trabalhos em que não possuem a intenção de realizar alguma análise de integração de práticas ao *Scrum*, porém são usados em suas experiências, exemplos de processos parcialmente híbridos que por ventura usam práticas do XP dentro de um modelo baseado no *Scrum*.

Com a definição de grupos de trabalhos realizada, são aplicadas as seguintes regras de pesos de acordo com o grau de relevância ao tratar sobre a junção entre os métodos dos grupos de trabalhos pesquisados originando a Tabela 4:

Tabela 4: Pontuação de Grupo de Trabalhos Analisados

Grupo	**Peso**
Grupo 1	3
Grupo 2	2,5
Grupo 3	2
Grupo 4	1

Sendo assim, os trabalhos coletados e analisados foram distribuídos da seguinte maneira, segundo a Tabela 5.

Tabela 5: Distribuição de Autores por grupo de trabalho

Autor	**Grupo**
Knickberg (2007)	1
Mar e Schwaber (2002)	1
Willians et al. (2011)	2
Dinakar (2009)	2
Salo e Abrahamsson (2008)	3
Sutherland et al. (2006)	4

Com esta distribuição, a pontuação de cada prática possuirá uma variação de 0 pontos (nenhuma citação) até 14 pontos (prática citada por todos os trabalhos).

Aplicando-se os critérios definidos acima, uma nova tabela é obtida (Tabela 6) em que cada prática agora possui uma pontuação que possui um maior peso dependendo do grupo no qual o autor foi enquadrado.

Tabela 6: Pontuação de práticas do XP ponderadas

Prática do XP	**Mar e Schwaber (2002)**	**Knickberg (2007)**	**Willians et al. (2011)**	**Sutherland et al. (2006)**	**Dinakar (2009)**	**Salo e Abrahamsson (2008)**	**Total**
Integração Contínua	3	3	2,5	1	2,5	2	**14**
Desenvolvimento Dirigido a Testes (TDD)		3	2,5	1	2,5		**9**
Código Coletivo		3	2,5	1		2	**8,5**
Padrões de Codificação		3			2,5	2	**7,5**
Programação em Pares	3	3		1			**7**
Refatoração	3	3		1			**7**
Design Simples	3	3					**6**
Ritmo Sustentável		3				2	**5**
Jogo de Planejamento			2,5				**2,5**
Metáfora							**0**
Releases Curtos							**0**
Cliente Presente							**0**

A Figura 4 apresenta o resultado obtido na Tabela 6 agrupando-se as práticas do XP por suas subdivisões: Práticas de Produto, Práticas de Equipe, Práticas de Processo e Práticas de Codificação.

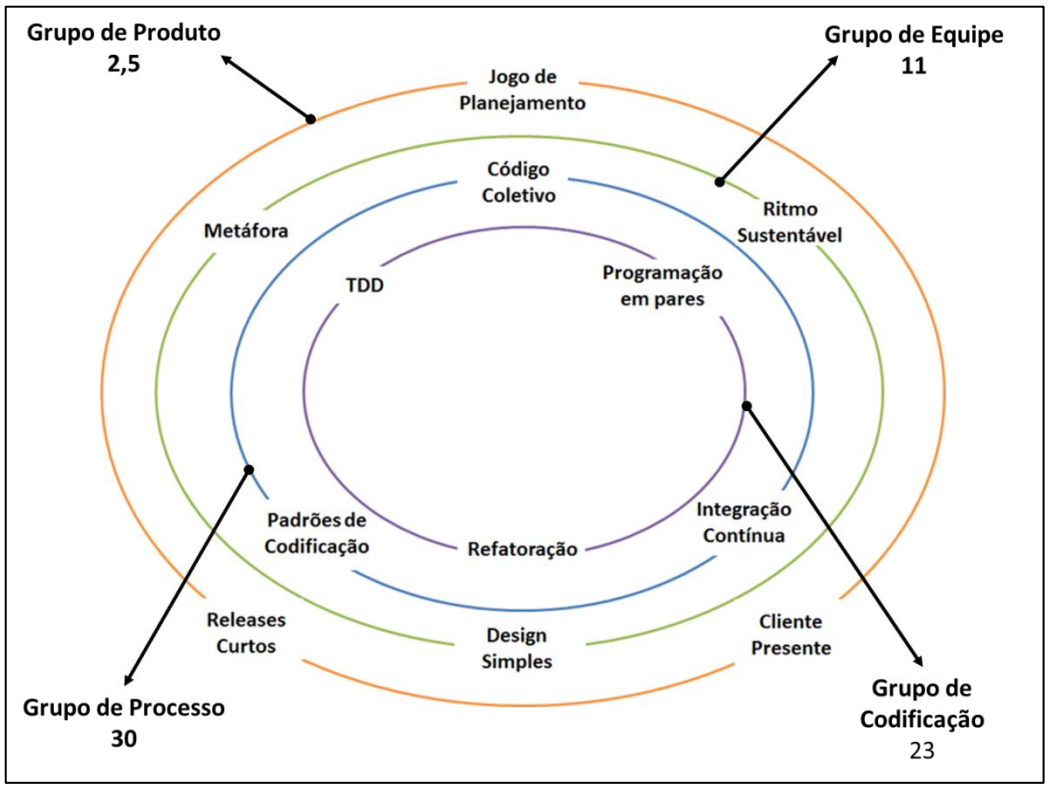

Figura 4: Distribuição de Pontos por área de conhecimento de práticas do XP

Analisando a Figura 4, é possível detectar que as práticas que obtiveram a pior pontuação estão diretamente ligadas a tarefas de Produto e Equipe, obtendo respectivamente 2,5 e 11 pontos. Entretanto, as práticas relacionadas a Processo e Codificação obtiveram 30 e 23 pontos respectivamente.

Uma hipótese para esse resultado está ligada diretamente aos autores citados procurarem incorporar práticas do XP preferencialmente de domínios não cobertos pelo *Scrum*. Quanto às práticas de produto como o Jogo de Planejamento, Releases Curtos e Cliente Presente, acabam se sobrepondo às práticas já existentes no *Scrum*.

Por exemplo, o uso da prática Cliente Presente já é de certa forma citada pelo *Scrum*, onde o cliente no papel de *Product Owner* é encorajado a possuir uma participação ativa a constante no ciclo de desenvolvimento, sendo recomendável se possível até ser um membro fixo da equipe de desenvolvimento até a conclusão do projeto em questão.

De acordo com Sutherland (2010, p.32), o *Scrum* enfatiza a produção de códigos potencialmente entregáveis ao final de cada *Sprint*, ao analisar esta afirmação, nota-se que a prática de *Releases* Curtos do XP acaba de fato não trazendo ganhos se utilizada junto ao *Scrum* por serem extremamente parecidas por seguirem princípios básicos dos métodos ágeis.

O Jogo de Planejamento foi à única dentre as práticas do XP ligadas a processo a receber uma pontuação, e essa pontuação por mínima que seja, é coerente, pois o *Scrum* de fato não determina de qual maneira deve ser mensurado o esforço relativo às funções que serão incluídas na *Backlog* da *Sprint*, ficando este ponto em aberto para a equipe.

Já as práticas do grupo de equipe, embora não sejam tratadas pelo *Scrum* em seu formato original, acabaram obtendo apenas uma pontuação intermediária, tornando evidente uma alta pontuação nos grupos de Processo e Codificação.

Com base nos resultados gerados através da Tabela 6. Os seguintes critérios para determinar quais práticas do XP são usadas na proposta de integração com o *Scrum*:

- A prática em questão deverá alcançar ao menos metade da nota máxima possível de 14 pontos (7 pontos).
- A prática em questão deverá ser citada ao menos pela metade dos autores pesquisados (3 citações).
- A prática deve atender obrigatoriamente aos dois critérios acima para ser considerada.

Com estes critérios devidamente aplicados, as práticas Integração Contínua, TDD, Código Coletivo, Padrões de Codificação, Programação em Pares e Refatoração foram selecionadas para serem utilizadas junto ao *Scrum* (práticas em verde na Tabela 7).

Já as demais práticas, mesmo não sendo selecionadas, foram classificadas em três outros grupos de acordo com a pontuação recebida, sendo as em amarelo (*Design* Simples e Ritmo Sustentável) consideradas como práticas optativas de serem incluídas em uma integração, o Jogo de Planejamento (em laranja) já entra para o grupo de práticas passíveis de serem agregadas caso exista uma necessidade específica de se otimizar o esforço atribuído a cada requisito solicitado pelo cliente.

Já as práticas restantes, Metáfora, Releases Curtos e Cliente Presente por não terem obtido nenhuma pontuação em nossa análise, entram para o grupo em vermelho, onde são tratadas como fora do escopo para uma possível integração com o *Scrum* neste trabalho.

Tabela 7: Classificação final de Práticas do XP

Prática do XP	Pontos	Citações
Integração Contínua	14	6
TDD	9	4
Código Coletivo	8,5	4
Padrões de Codificação	7,5	3
Programação em Pares	7	3
Refatoração	7	3
Design Simples	6	2
Ritmo Sustentável	5	2
Jogo de Planejamento	2,5	1
Metáfora	0	0
Releases Curtos	0	0
Cliente Presente	0	0

4.2. Inter-relações entre as Práticas do XP

Segundo Beck (1999), uma prática do XP usada de forma isolada não consegue manter sua plena eficiência ao comparar-se com a mesma prática utilizada dentro do método completo do XP, elas necessitam de outras práticas complementares para que se mantenham devidamente balanceadas.

Ao analisar a Figura 5, também de Beck (1999), pode-se ver que as práticas dentre as selecionadas na Tabela 6 estão inter-relacionadas, representando uma forte sinergia entre elas, o que de fato possibilita que a integração destas práticas ao *Scrum* sem uma perda significativa de eficiência. Essa interligação é

explicada por tratar-se de práticas voltadas para atividades de desenvolvimento e diretrizes de como realizar o desenvolvimento pela equipe.

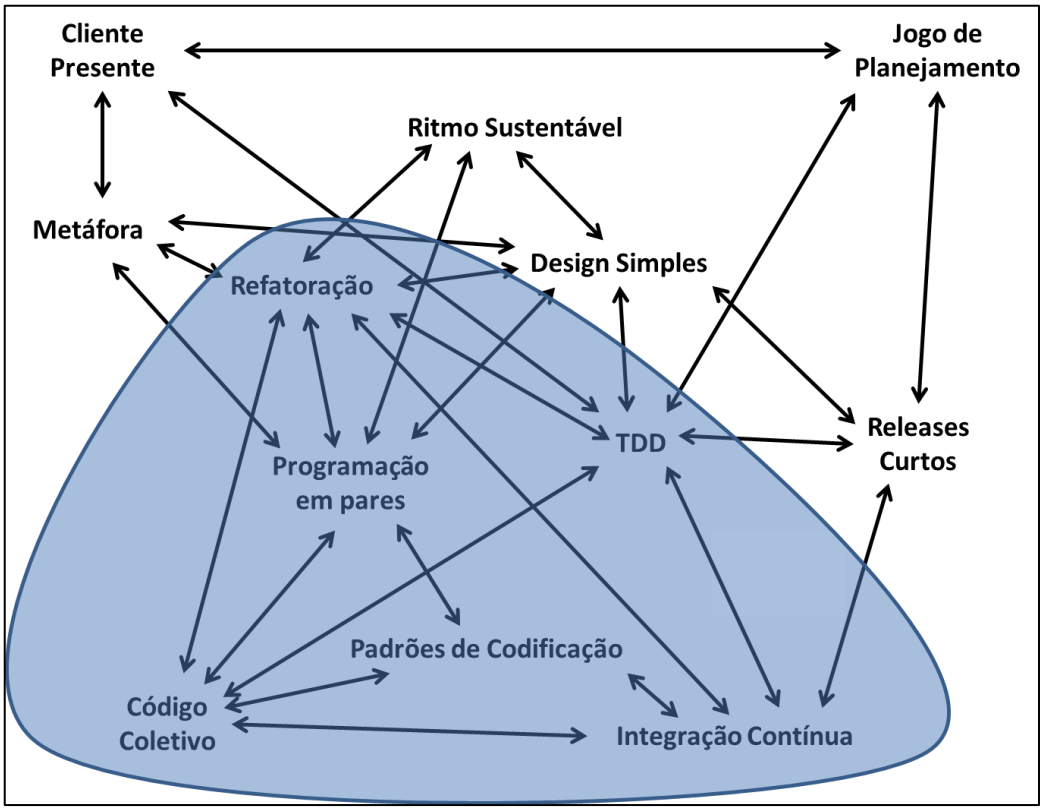

Figura 5: Interligação de Práticas XP - Adaptada de BECK (1999)

4.3. Proposta de Integração do XP ao Scrum

Tendo as práticas selecionadas e verificada a possibilidade de seu uso fora do modelo original proposto no próprio XP, é necessário verificar como cada prática é inserida no modelo do *Scrum*.

Supondo que uma integração total dos dois métodos fosse realizada, certamente haveria inclusões de práticas do XP em diversas etapas do ciclo proposto no *Scrum*, porém como verificado anteriormente, as práticas selecionadas tratam de atividades ligadas diretamente à codificação e também de diretrizes de como a equipe deve atuar durante o desenvolvimento, desenvolvimento esse realizado diariamente dentro das *Sprints* do *Scrum*.

Sendo assim, o modelo proposto representado na Figura 6, altera o *Scrum* somente em um ponto de seu ciclo, mantendo sua estrutura original intacta.

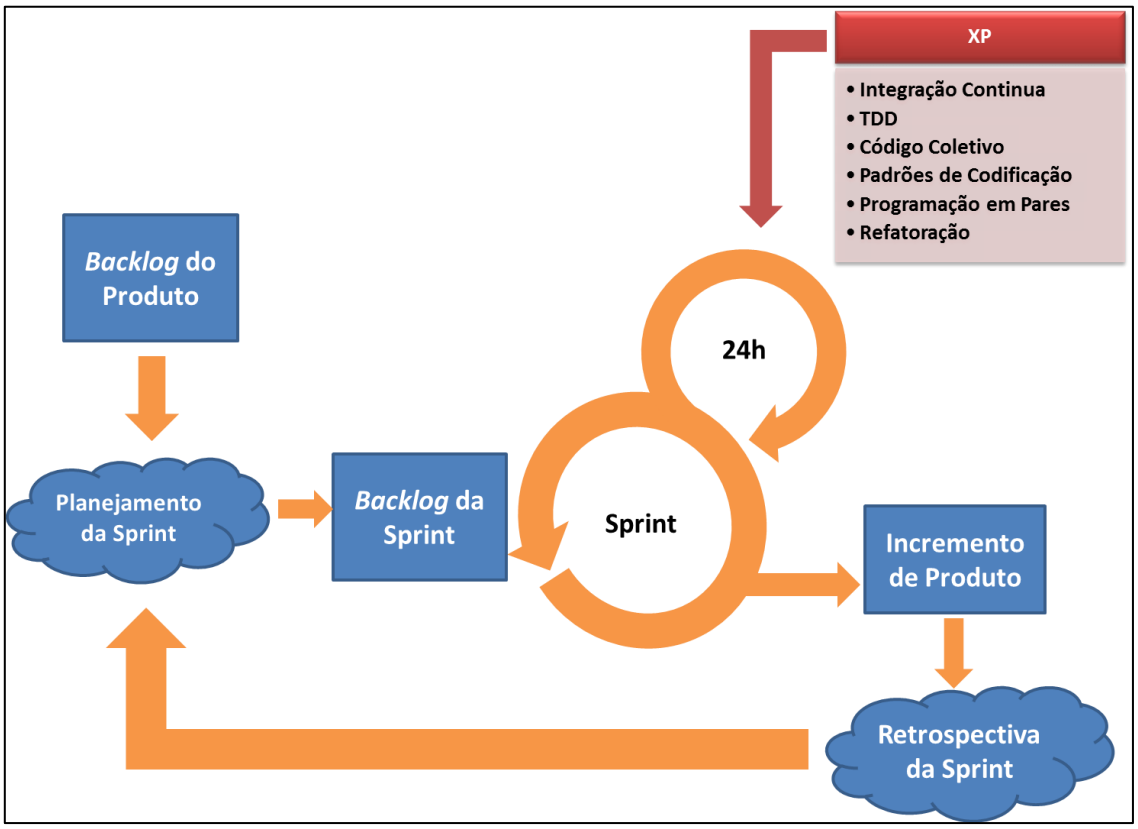

Figura 6: Representação do Modelo Integrado Scrum com XP Proposto

Para ilustrar o funcionamento de uma *Sprint* utilizando o modelo integrado proposto, a seguir são apresentados quais pontos do *Scrum* devem passar por alterações e inclusões para que a integração das práticas do XP ocorra de forma organizada para a equipe.

Schwaber e Beedle (2002, p. 47) retrataram que antes de cada *Sprint*, a equipe deve realizar inicialmente o Planejamento da *Sprint* junto ao *Product Owner* para determinar quais funções serão construídas nesta *Sprint*. Após isso é realizada um novo planejamento de *Sprint*, porém, este somente com a equipe de desenvolvimento, para ser discutido como será realizada a construção destas funções assim como, como será a organização da equipe nesta *Sprint,* algo que já é discutido no formato original desta reunião no Scrum.

Como parte desta segunda etapa do Planejamento da *Sprint*, os seguintes itens necessitam ser incluídos na pauta original do *Scrum* na primeira *Sprint* Planning do Projeto para possibilitar uma melhor disseminação das práticas do XP selecionadas para o modelo proposto:

- Discutir e definir pela equipe, um determinado **padrão de codificação** a ser utilizado como diretriz pela equipe durante as demais *Sprints*, e preferencialmente nos demais projetos.
- Tornar de conhecimento comum da equipe o uso das ferramentas de versionamento de códigos fontes assim como incentivar a aplicação do **Código Coletivo,** para que toda a equipe colabore com novas ideias e melhorias nos códigos existentes.
- Definir como será realizada a integração dos diversos componentes a serem construídos pela equipe, qual será a frequência de geração de novas versões de software e caso seja possível, estabelecer o uso de alguma aplicação de automatização do processo de geração de versões para diminuir o tempo gasto com esta atividade, a fim de tornar esse processo automático e transparente, poupando um esforço valioso da equipe ao longo do tempo **(Integração Contínua)**.
- Orientar a equipe para o uso de uma abordagem de **TDD**, escrevendo casos de teste para todas as funções deste *Backlog* da *Sprint*, que auxiliará o programador a considerar o que realmente precisará ser construído, assim como possibilitará uma futura automatização destes testes com o auxílio de ferramentas de automação de testes.
- Definir quais funções serão consideradas como necessárias de se utilizar a abordagem da **Programação em Pares**, podendo ser somente funções críticas do sistema, ou até mesmo todas as funções dependendo do projeto em questão (seguindo os relatos dos trabalhos analisados). Também será necessário determinar quais duplas trabalharão juntas nesta *Sprint*, a fim de se obter uma codificação de maior qualidade além de auxiliar que toda a equipe tenha um bom conhecimento em todos os

módulos a serem desenvolvidos no sistema, para isso é importante que periodicamente os pares sejam revezados de preferência a cada nova *Sprint*.

- Disseminar a importância da **Refatoração** pela equipe, assim definir em quais momentos ele será realizado nesta *Sprint,* a fim de eliminar redundâncias de código e funções não usadas, proporcionando uma maior reusabilidade dos métodos já desenvolvidos, assim como aumentar a escalabilidade e manutenibilidade da solução desenvolvida devido à diminuição da complexidade do código fonte criado.

4.4. Considerações do Capítulo

Com o resultado obtido através da Figura 6, a constatação de Knickberg (2007, p.77), de que uma possível integração dentre os métodos *Scrum* e XP além de possível, como altamente recomendada, é justificada de fato por cada método ágil ter uma ênfase a uma área de conhecimento em particular (O *Scrum* em questões de gerenciamento e XP em atividades de desenvolvimento e codificação).

Isso pode ser notado neste trabalho com a ilustração ainda da Figura 6, onde todas as práticas do XP selecionadas foram inseridas no fluxo de trabalho diário da *Sprint* onde acabam sendo realizadas todas as tarefas de desenvolvimento.

Torna-se evidente uma lacuna existente dentro do *Scrum* em que atividades ligadas ao desenvolvimento devem ser inseridas, sendo que as práticas do XP selecionadas trouxeram resultados positivos no relatos de diversos autores discutidos anteriormente, proporcionando um ganho de qualidade no software construído, assim como uma maior produtividade da equipe de desenvolvimento.

Um fator de suma importância durante a análise da integração do XP no *Scrum*, foi que em momento algum foi necessário realizar alterações do *Scrum* em seu

fluxo estrutural para a inserção das práticas do XP, fator esse que agrega uma maior robustez no modelo integrado proposto por não desfigurar nenhum ponto chave do *Scrum*.

5. CONSIDERAÇÕES FINAIS

Muitos estudos e materiais apontam que uma junção entre o *Scrum* e o XP é recomendada, proporcionando ganhos expressivos de qualidade e produtividade, Porém é detectada uma dificuldade em entender como essa junção deve ser realizada, e qual foi o critério utilizado para que determinada prática do XP fosse escolhida para fazer parte ou não desta junção.

Com base nos autores estudados, foi notado que certo grupo de práticas do XP eram mais citadas do que as demais, sendo assim, estas consideradas como práticas mais relevantes de serem integradas ao *Scrum*. Porém, para obter-se a lista de práticas elegíveis para uma proposta de integração, as práticas ainda passaram por uma reclassificação de acordo com o grau de relevância dos trabalhos dos autores analisados, a fim de se obter uma classificação mais coerente em termos de relevância para o trabalho.

Um aspecto que ficou evidente dentro do trabalho foi que o *Scrum* não possui orientações dentro de certas área de conhecimento, dentre elas o processo de desenvolvimento e codificação, sendo que este espaço detectado foi o mais indicado para receber as práticas selecionadas do XP.

Outro fator importante para agregar robustez ao trabalho, foi que dentre as práticas selecionadas para a integração, todas tratam diretamente de atividades de desenvolvimento, assim como são consideradas como práticas complementares, diminuindo significativamente possíveis perdas em sua eficiência ao serem utilizadas fora do desenho original do XP.

5.1. Contribuições do Trabalho

Entende-se que o objetivo proposto inicialmente pelo trabalho foi cumprido, através da análise de integração entre o *Scrum* e XP, foram estabelecidos critérios para justificar a inclusão das práticas usadas na proposta de junção realizada, critérios estes embasados em estudos previamente realizados por outros autores.

Também foi apresentado de forma mais clara, como uma integração dentre os dois métodos pode ser realizada, assim como pode-se detectar quais práticas do XP são mais relevantes de serem integradas ao *Scrum*. Isso foi realizado determinando um critério para que cada prática do XP fizesse ou não parte da proposta de integração ao *Scrum*. Tal informação foi consolidada na tabela 7, onde cada prática do XP recebeu certa pontuação conforme sua necessidade de ser integrada ao *Scrum*.

Também foi apresentado como funcionaria o modelo integrado proposto conforme a Figura 6, em que são sinalizadas quais partes do *Scrum* passariam por alterações para que as práticas do XP fossem inseridas.

5.2. Trabalhos Futuros

Como este trabalho foi focado unicamente em uma proposta teórica de uma integração do XP ao *Scrum*, dentro os trabalhos futuros recomendados estão à aplicação da proposta de integração realizada neste trabalho, assim como uma análise de integração utilizando o XP como método base, e ser feita a integração de práticas dos *Scrum* necessárias.

As demais práticas que ficaram de fora desta proposta de integração também poderiam ser mais bem discutidas e analisadas em quais situações elas poderiam contribuir ao *Scrum*.

REFERÊNCIAS

BECK, K. **Extreme Programming Explained: Embrace Change**, 1.ed. Addison-Wesley Professional, 1999. 224p.

BECK, K.; ANDRES, C. **Extreme Programming Explained: Embrace Change**, 2.ed. Addison-Wesley Professional, 2004. 224p.

COHN, M. **Succeeding with Agile: Software Development Using Scrum**. 1.ed. Addison-Wesley Professional, 2009. 504p.

COHN, M. **What is Scrum?**. 2012. Disponível em: http://www.mountaingoatsoftware.com/topics/scrum. Acessado em 23 jun. 2013.

DINAKAR, K. Agile Development: Overcoming a Short-Term Focus in Implementing Best Practices. **In Proceedings of the 24th ACM SIGPLAN conference companion (OOPSLA '09),** 2009. p. 579-588.

FRANCO, E. **Um Modelo de Gerenciamento de Projetos Baseado nas Metodologias Ágeis de Desenvolvimento de Software e nos Princípios da Produção Enxuta**. São Paulo, 2007. 107p. Dissertação de Mestrado – Universidade de São Paulo.

HIGHSMITH, J. **Agile Project Management. Creating Inovative Products**. 1.ed. Addison-Wesley Professional, 2004. 448p.

KRISHNA, V.; BASU, A. Scrum+: Is it ScrumBut or ScrumAnd. **In Proceedings of: India Conference (INDICON)**, 2011 Annual IEEE, 2011. 4p.

KNICKBERG, H; **Scrum and XP from the Trenches: How we do Scrum**. 1.ed. Prentice-Hall, 2007. 131p.

MAR, K.; SCHWABER, K. **Scrum with XP**. InformIT (22 Mar.2002). 2002. Disponível em: http://faculty.salisbury.edu/~xswang/research/papers/serelated/scrum/scrumxp.pdf, Acessado em 23 jun. 2013.

MARTIN, R. **The Land That Scrum Forgot**. Publicado em 14/12/2010 em ScrumAlliance. Disponível em: http://www.scrumalliance.org/community/articles/2010/december/the-land-that-scrum-forgot - Acessado em 23 jun. 2013.

PRESSMAN, R. **Software Engineering: A Practicioner's Approach**. McGraw-Hill, 5.ed. 2001. 853p.

PRESSMAN, R. **Engenharia de Software**. McGraw-Hill, 6.ed. 2006. 752p.

SALO, O.; ABRAHAMSSON, P. **Agile methods in European embedded software development organizations: a survey on the actual use and usefulness of Extreme Programming and Scrum**. Software, IET (Volume:2 , Issue: 1). 2008. p. 58-64.

SCHWABER, K.; BEEDLE. **Agile Software Development with Scrum**. 1.ed. Pearson Education International, 2002. p.158.

SUTHERLAND, J. Pretty Good Scrum: **Secret Sauce for Distributed Teams**. 2008. Disponível em: http://confluence.agilefinland.com/download/attachments/884822/Pretty+Good+Scrum+v6+CSM.pdf?version=1, Acesso em 24 jun. 2013. 58p.

SUTHERLAND, J. **Scrum Handbook**. The Scrum Training Institute. 2010. Disponível em: www.jeffsutherland.com/scrumhandbook.pdf. Acessado em 23 jun. 2013. 66p.

SUTHERLAND, J. **The Scrum Papers: Nut, Bolts, and Origins of an Agile Framework**. Scrum, Inc. 2012. Disponível em: http://jeffsutherland.com/ScrumPapers.pdf. Acessado em 24 jun. 2013. 217p.

SUTHERLAND, J.; SCHWABER, K. **The Scrum Guide**. 2011. Disponível em: http://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum_Guide.pdf. Acessado em 24 jun. 2013. 16p.

SUTHERLAND, J.; VIKTOROV, A.; BLOUNT, J.; KNICKBERG, H. Distributed Scrum: Agile Project Management with Outsourced Development Teams. **In Proceedings of: HICSS 2007. 40th Annual Hawaii International Conference. 2006. 274p.**

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

WILLIANS, L.; BROWN, G.; MELTZER, A.; NAGAPPAN, N. Scrum + Engineering Practices: Experiences of Three Microsoft Teams. **In Proceedings of: ESEM '11 Proceedings of the 2011 International Symposium on Empirical Software Engineering and Measurement.** 2011. p. 463-471.