CARLOS CHRISTOFER KATO NAGANO

IMPLEMENTAÇÃO DE ARCABOUÇO ÁGIL PARA DESENVOLVEDORES SOLOS

São Paulo

2012

CARLOS CHRISTOFER KATO NAGANO

IMPLEMENTAÇÃO DE ARCABOUÇO ÁGIL PARA DESENVOLVEDORES SOLOS

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Especialista em Tecnologia da Informação

Área de Concentração: Engenharia de Software e Tecnologia da Informação

Orientador: Professor Eduardo Oliveira

São Paulo

2012

FICHA CATALOGRÁFICA

Nagano, Carlos Christofer Kato
Implementação de arcabouço ágil para desenvolvedores solos / C.C.K. Nagano. -- São Paulo, 2012.
85 p.

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

1.Administração de projetos 2.Tecnologia da informação 3.Metodologias ágeis I.Universidade de São Paulo. Escola Politécnica. Programa de Educação Continuada em Engenharia II.t.

DEDICATÓRIA

Dedico este trabalhão a todos que me ajudaram nesta caminhada, todos que me proveram de conhecimento seja este técnico ou sobre a vida, espero não decepcioná-los.

AGRADECIMENTOS

Ao Professor Eduardo Oliveira pela orientação e constantes correções, a todos os professores que contribuíram com minha educação seja no colégio, faculdade ou pós-graduação, a minha família que sempre me deu suporte, a minha namorada que doou vários finais de semana e a todos os meus amigos que ficaram sem me ver durante o período de execução deste trabalho.

Imaginar é mais importante que saber, pois o conhecimento é limitado, enquanto a imaginação abarca o universo.

(Albert Einstein)

Mesmo as noites totalmente sem estrelas podem anunciar a aurora de uma grande realização.

(Martin Luther King)

RESUMO

Com o mercado crescente, a busca por melhores produtos e serviços e por redução de custos torna-se maior, assim a área de tecnologia da informação ganha força com maiores investimentos sendo um dos setores que pode trazer um alto retorno sobre o investimento. Esta monografia traz como proposta um modelo de desenvolvimento de software em empresas que planejam montar, ou já possuem, uma área de desenvolvimento baseado em desenvolvedores solos. O conceito do desenvolvedor solo indica um projeto de um único desenvolvedor atuando como analista testador e programador. O foco é implantar um processo de desenvolvimento sustentável baseado em práticas ágeis, tais como Kanban, Scrum e XP. Estas metodologias apesar de descreverem pequenos times de desenvolvimento, com sete a doze pessoas cada, foram adaptadas para o cenário solo em questão. O processo adaptado irá, dentro de cada uma das metodologias, utilizar-se de práticas que possam ser aplicadas a um projeto. Estas práticas que podem ser divididas entre técnicas que irão dar suporte às tarefas de desenvolvimento ou que irão auxiliar a gerência do projeto e do processo de desenvolvimento. Criando um novo conjunto de regras ágeis mais completos para guiar o processo de desenvolvimento que irão facilitar o relacionamento com o cliente e a gestão e para oferecer ao mercado uma solução para a falta de mão de obra qualificada e baixo orçamento para contratação.

Palavras-Chave: Gerencia de Projetos. Técnicas Ágeis de Desenvolvimento de Software.

ABSTRACT

With the increasing market, the search for better products and services and for cost reduction increases as well, this way the information technology area gains strength with more investments being one of the sectors that may bring a high return over investment. This monograph brings as proposition a model of software development for companies with limited investments who are willing to set, or already have, a development area based on solo developers. The solo developer concept indicates a project of a single developer acting as analyst, tester and developer. The focus is to implant a sustainable development process based on agile practices such as Kanban, Scrum and XP. Despite of describing small developing teams of seven to twelve persons, this methodologies were be adapted to the proposed solo scenario. The adapted process, based on every cited methodology, will use of practices that may be applied for a project. These practices may be divided between techniques that will support development tasks or that will assist either the project management or the development process management. Creating a new set of agile rules to guide the development process that will make easier the customer relationship and the management and to offer to the market a solution to the lack of qualified labor and the low budget for hiring.

Key-Words: Project Management. Agile Techniques for Software Development.

LISTA DE ILUSTRAÇÕES

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

TI	Tecnologia da Informação
XP	Extreme Programming
CAGR	Taxa de Crescimento Composto Anual
UP	Unified Process
UML	Unified Modeling Language
PO	Product Owner
ROI	Retorno Sobre o Investimento
TDD	Test Driven Development
IFMA	International Facilities Management Association
BPMN	Business Process Model and Notation

SUMÁRIO

1 INTRODUÇÃO

1.1 Contexto Inicial

Atualmente o Brasil sofre uma mudança muito grande em relação à área de TI (Tecnologia da Informação), o aumento em investimentos na área de TI deve atingir os 10% de crescimento e deverá ser mantido nesta taxa nos próximos três anos segundo FERRER (2011). Este aumento nos gastos em tecnologia se justifica na medida em que as empresas entendem que este pode ser o diferencial para aumento de produtividade e automatização de processos internos, aumentando a agilidade e reduzindo custos para a corporação.

As organizações brasileiras abraçaram a recessão global como uma oportunidade e buscaram a tecnologia como um fator decisivo, o que ajudou o país (Brasil) a recuperar rapidamente a demanda e o crescimento de TI. (FERRER, 2011).

Todo este crescimento exige do mercado uma demanda maior no número de profissionais para desenvolvimento de sistemas que são cada vez mais escassos em qualidade, pois de acordo com SEPRORJ (2011), há deficiência nos cursos de formação destes profissionais, acarretando na qualidade escassa de iniciantes recém-formados e por consequência no aumento de custo para as empresas.

Em paralelo com esta vertente, uma tendência tem se tornado cada vez mais realidade globalmente, a do conceito ágil de desenvolvimento de software. O crescimento na adoção de práticas ágeis chega a 22% CAGR (Taxa de Crescimento Composto Anual) e acredita-se que esteja diretamente ligado ao fato do índice de falhas em projetos de TI ter caído de 24% para 21% e o índice de sucesso ter aumentado de 32% para 37% entre 2008 e 2010. (REDAÇÃO, 2011).

Este conceito que prioriza o cliente e entregas funcionais contínuas de pedaços de software que deem valor ao negócio. (SCHWABER et al., 2001a).

1.2 Objetivo

O objetivo deste trabalho é prover um novo arcabouço de desenvolvimento baseado nas metodologias mais aplicadas no mercado, como Scrum, Extreme Programming e Kanban, mantendo as melhores características de cada arcabouço utilizado, porém com foco no desenvolvedor solo, caracterizado por ser o único desenvolvedor no projeto em uma empresa cujo foco não é tecnologia da informação, o qual deve se autogerenciar sem comprometer os prazos e expectativas do cliente se.

1.3 Metodologia

A metodologia utilizada neste trabalho baseia-se na comparação de três arcabouços ágeis de desenvolvimento conhecidos como Scrum, XP (Extreme Programming) e Kanban. Esta comparação possibilitará a criação do arcabouço ágil para desenvolvedores solos, permitindo que este seja mais completo tanto no âmbito técnico quanto no gerencial.

Para ilustração dos processos nesta monografia, será utilizada a notação "Business Process Model and Notation (BMPN)". Esta notação não faz parte do escopo principal desta monografia, portanto, seu uso e regras não serão aprofundados. Maiores detalhes sobre esta notação podem ser obtidos em OMG (2011).

Após descrever o novo arcabouço, será feita uma simulação de implementação em um cenário proposto passando por uma iteração completa de desenvolvimento.

Com isso, será feita a conclusão e serão dadas sugestões de trabalhos futuros.

1.4 Abrangência

O trabalho limita-se a utilizar problemas reais de uma empresa, sem representar, porém, um projeto real. A aplicação abrange uma exemplificação de como implantar o arcabouço criado, servindo de base para a conclusão.

1.5 Justificativa

A demanda crescente na área de TI e a falta de preparo dos profissionais existentes geram aumento de custo e dificuldade de contratação de mão de obra qualificada. (SEPRORJ, 2011).

Por isso as metodologias ágeis têm tido uma aceitação cada vez maior na área de TI. Por se basear em equipes reduzidas e em entregas rápidas, (REDAÇÃO, D., 2011).

Apesar de o arcabouço necessitar de um desenvolvedor experiente, não há necessidade de outros desenvolvedores, pois concentram-se os papeis técnicos no desenvolvedor solo, eliminando a necessidade de uma equipe técnica.

O arcabouço também garante segurança na gestão do processo de desenvolvimento gerando menos acoplamento entre o desenvolvedor e a empresa produzindo artefatos que documentam as escolhas feitas durante o desenvolvimento reduzindo riscos quanto a perda do desenvolvedor ao mercado.

A realidade do conceito ágil está cada vez mais sólida, mas seus arcabouços são focados em empresas de software ou grandes corporações, onde se encontram times de desenvolvimento. Ainda existem, porém, muitos desenvolvedores solos dentro de pequenas empresas. O desenvolvedor solo é resultado de equipes ou projetos pequenos, quando em ambos os casos

temos apenas um desenvolvedor como responsável por entregar o produto ao cliente.

Este trabalho irá possibilitar a aplicação de um framework voltado às necessidades de um desenvolvedor solo ágil se baseando nas metodologias Scrum, Extreme Programming e Kanban, que estão entre as dez metodologias mais aplicadas no mercado de acordo com VESIONONE (p4 2012).

2 REVISÃO DA BIBLIOGRAFIA

2.1 O Manifesto Ágil

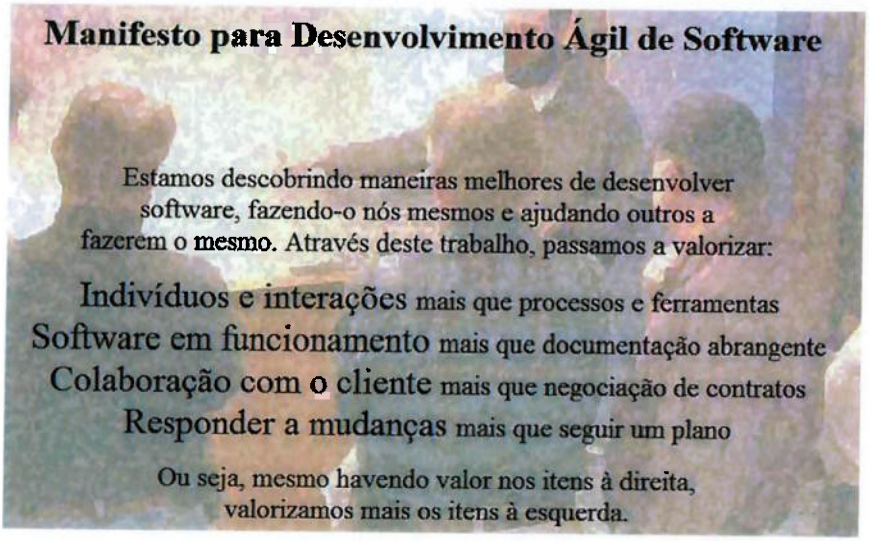

Figura 1: O Manifesto Ágil (SCHWABER et al., 2001a)

O manifesto ágil foi criado após um encontro dos principais representantes de diferentes metodologias existentes para desenvolvimento de software, o extreme Programming, Scrum, Feature-Driven Development, Adaptive-Software Development, entre outros. Cada um destes é uma metodologia criada como uma alternativa aos processos convencionais de desenvolvimento burocrático de software e voltado à documentação. (HIGHSMITH, 2001).

O quadro acima descreve o manifesto e seus valores, sendo eles:

- Dar valor a indivíduos e interações mais que processos e ferramentas.
- Dar valor a software em funcionamento mais que documentação abrangente.
- Dar valor à colaboração com o cliente mais que negociação de contratos.
- Responder a mudanças mais que seguir um plano.

Estes valores que originam os princípios do desenvolvimento ágil, a saber, satisfação do cliente, receber bem mudanças de escopo, entregas frequentes de software funcional, áreas de negócio e desenvolvimento trabalhando juntas, equipe motivada e confiante, conversas presenciais, medir progresso através de software funcional, desenvolvimento sustentável, excelência técnica, simplicidade, times auto-gerenciáveis e reflexões de melhoria do processo. (SCHWABER et al., 2001b).

Estes princípios integram as áreas envolvidas auxiliando no processo de desenvolvimento de um software, porém ainda há um paradigma muito grande a se vencer, pois de acordo com VERSIONONE (2010) o segundo maior motivo de projetos ágeis que falharam é a empresa ter uma cultura organizacional não coerente com os valores ágeis, valores descritos por SCHWABER, K. et al. (2001a), como a entrega de software funcional ao invés de documentação compreensiva. A documentação compreensiva é um dos valores do UP (Unified Process) que é outra metodologia de desenvolvimento de software, porém orientado a processos e documentos. (PRESSMAN, R. S., p52 2005). Por isso ainda é preterido nas empresas.

2.2 Elementos de um Arcabouço

Um arcabouço pode ser dividido em participantes (Atores), reuniões (Eventos), arquivos gerados (Artefatos) e atividades (Práticas de Engenharia). Estes elementos são as partes do processo de desenvolvimento de um software, definidos por iterações de reuniões com participantes gerando arquivos através das atividades descritas. Esta descrição ajuda o entendimento do framework.

2.2.1 Atores

Segundo descrito na UML um ator corresponde a um papel desempenhado por usuários ou sistemas, interagindo com outro sistema. (BOOCH, G.; RUMBAUGH, J.; JACOBSON, I., p221 2000).

Um ser humano pode representar um ator e ter vários papeis em diferentes cenários. Qualquer interessado no projeto deve ter um papel e, portanto ser descrito como um ator.

UML é uma linguagem gráfica, para visualização, especificação, construção e documentação de artefatos de sistemas de software, não será abordada toda a definição do UML (Unified Modeling Language) neste trabalho, sendo apenas utilizada para descrever a função do Ator.

2.2.2 Eventos

Os eventos são todas as partes do processo que possuem atores envolvidos com um objetivo, seja a entrega de um artefato ou alinhar expectativas. O guia de Scrum atribui eventos com duração fixa, descritas como caixas de tempo. (SCHWABER, K.; SUTHERLAND, J., p5 2011). Uma caixa de tempo designa um período de desenvolvimento ou uma reunião.

2.2.3 Artefatos

De acordo com REIS, R. Q.; REIS, C. A. L.; NUNES, D. J. (2002, p9): "Um artefato é um produto criado ou modificado durante um processo. Tal produto é resultado de uma atividade e pode ser utilizado posteriormente como matéria prima para a mesma ou para outra atividade a fim de gerar novos produtos.".

2.2.4 Práticas de Engenharia

Práticas de engenharia são o que Pressman R. S. (p18, 2006) define como métodos, ou seja, técnicas para a construção de software. Atividades de desenvolvimento em meio ao processo, estas técnicas devem auxiliar na diminuição de erros no código, facilitar a manutenção e melhoria do software e dar agilidade no processo de desenvolvimento.

2.3 Práticas do eXtreme Programming

O XP (eXtreme Programming) é uma metodologia ágil que tem como objetivo melhorar o projeto de software em cinco aspectos, comunicação, simplicidade, retorno sobre o trabalho feito, respeito e coragem. Este é o mais prescritivo dentre os frameworks propostos por prescrever dezenove regras, é também quem mais define práticas de engenharia focadas no desenvolvimento de software. (KNIBERG, H.; SKARIN, M., p31 2009).

Abaixo está representado o ciclo básico de desenvolvimento no XP como descrito por WELLS, D. (2009). Nele é feita a seleção dos requisitos mais importantes, planejamento da iteração, estimativa do prazo e desenvolvimento dos requisitos até a entrega do software funcional.

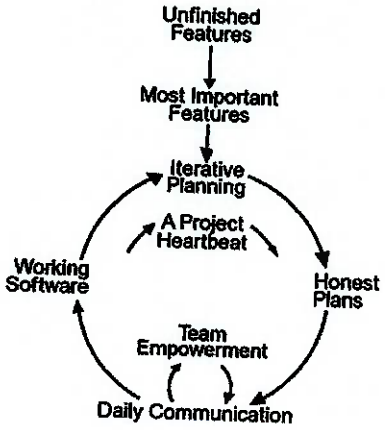

Figura 2: XP (WELLS, D., 2009)

2.3.1 Atores

O XP descreve seis atores, dos quais JEFFRIES, R. E. (1999) destaca apenas dois obrigatórios, o cliente e o programador.

2.3.1.1 Gerente

O ator descrito como Gerente no extreme programming é opcional e deve prover os recursos necessários ao projeto, devendo também atuar na parte de comunicação externa e coordenar as atividades. (JEFFRIES, R. E., 1999). É o principal canal de comunicação entre todos os interessados externos ou internos ao projeto.

2.3.1.2 Cliente

O papel do cliente é obrigatório no projeto, é a figura dele a responsável por determinar os requisitos e priorizá-las. (JEFFRIES, R. E., 1999). É também o responsável pelo sucesso do projeto de acordo com MARTIN, A. M. (p7 2009), pois a estratégia de prioridades de lançamento norteia a aceitação da solução.

2.3.1.3 Testadores

Responsável por garantir a qualidade do código e que a funcionalidade desenvolvida de fato cumpra com os requisitos descritos pelo cliente. O testador também pode auxiliar a área de negócio a definir quais são os parâmetros de aceitação para que o desenvolvido seja aceito. (JEFFRIES, R. E., 1999).

2.3.1.4 Analistas

O analista é responsável por entender os requisitos de negócio e também entender as dificuldades técnicas envolvidas. É também quem deve auxiliar na comunicação entre cliente e programador atuando junto à área de negócios para definir os requisitos. (JEFFRIES, R. E., 1999).

2.3.1.5 Programadores

Outro papel obrigatório para qualquer arcabouço de desenvolvimento, o programador é responsável por transformar os requisitos em um produto, utilizando uma linguagem de programação para a codificação do software. (JEFFRIES, R. E., 1999).

2.3.1.6 Treinador / Técnico

A função do treinador é ajudar a manter o processo de desenvolvimento fluindo e retirar quaisquer impeditivos quanto às regras do extreme programming. (JEFFRIES, R. E., 1999).

2.3.2 Eventos

Existem dois eventos descritos na definição do XP, o planejamento do lançamento e o planejamento da iteração.

2.3.2.1 Planejamento do Lançamento

O planejamento do lançamento é o evento no qual ocorre o primeiro encontro entre a equipe de desenvolvimento e o cliente. Nesta etapa do processo, o cliente deve descrever o que entende como necessário ao produto com ajuda da equipe, e faz-se então uma lista de requisitos. A equipe deve alinhar a expectativa junto ao cliente descrevendo como será o produto e então estimar, ainda que sem precisão, a quantidade de dias para transformar todos os requisitos em produto. O planejamento deve ser atualizado após cada iteração. (JEFFRIES, R. E., 1999).

2.3.2.2 Planejamento da Iteração

O planejamento da iteração foca apenas nas atividades da iteração atual de desenvolvimento, deve-se fazer uma estimativa mais exata planejando-se a quantidade de tempo a ser gasto nas tarefas que compõe os requisitos desta iteração e ao final da iteração entrega-se a parte do produto planejada. Estimar o tempo de desenvolvimento é função da equipe. (JEFFRIES, R. E., 1999).

2.3.3 Práticas de Engenharia

Com onze práticas, o XP (extreme programming) deixa perceptível o foco maior em desenvolvimento. Estas práticas auxiliam na codificação e na qualidade do software e são práticas de implementação. (SAVOINE, M. et al., [entre 2008 e 2011]).

2.3.3.1 Testes de Aceitação

Os testes de aceitação descrevem os requisitos e regras de negócio que o sistema deve possuir. O cliente descreve junto à equipe alguns testes que serão automatizados de forma que garantam a aceitação do código gerado. Estes testes assegurarão que futuros códigos desenvolvidos sejam compatíveis com as regras de negócio já existentes. (JEFFRIES, R. E., 1999).

2.3.3.2 Lançamentos Pequenos

Com iterações de tempo curto, o time de desenvolvimento deverá manter um passo constante liberando freqüentemente pequenos lançamentos, alcançando com o tempo a publicação contínua do software. O objetivo é continuar agregando valor ao produto somando sempre pedaços funcionais de código. (JEFFRIES, R. E., 1999).

2.3.3.3 Projeto Simples

O projeto para praticantes do XP é descrito como toda a etapa de desenvolvimento do software e deve ser simples o suficiente para atender os requisitos do cliente, atentando-se que, simples não é sinônimo de pobre ou sequer fácil. O projeto ser simples apenas descreve que nenhum esforço extra deverá ser despendido se não houver real necessidade. O projeto inicia-se simples e a complexidade se dará com as novas implementações após as iterações. (JEFFRIES, R. E., 1999).

2.3.3.4 Programação em Par

Umas das principais práticas de engenharia do XP, a programação em pares prevê dois programadores por computador construindo o mesmo código e possui dois objetivos. O primeiro objetivo é revisar o código na medida em que é criado. O outro objetivo é equilibrar o conhecimento e experiência da equipe, uma vez que sempre existirão desenvolvedores melhores em um aspecto e, portanto deixá-los pareados com quem tem menos habilidade na mesma área de conhecimento permitirá uma difusão de conhecimento. (JEFFRIES, R. E., 1999).

2.3.3.5 Test Driven Development

O desenvolvimento dirigido a testes, ou TDD, é outra das principais práticas do XP e descreve que qualquer funcionalidade seja iniciada pela codificação do teste automatizado de aceitação do requisito. O teste deve garantir que o requisito está pronto, permitindo que todo código criado esteja em conformidade com os requisitos do negócio. Além de garantir que o código atenda às necessidades, isto permite que um futuro requisito, ao ser desenvolvido, não gere falhas ao código, p ois os testes automatizados

irão acusar estes erros antes de publicar o novo código. (JEFFRIES, R. E., 1999).

2.3.3.6 ***Refatoração***

Esta prática foca seu processo na eliminação de códigos que tenham a mesma função, exercitando a melhoria na coesão do código tornando-o mais claro para entendimento e reduzindo o acoplamento entre um pedaço de código e outro, o que facilita melhorias e manutenções posteriores. A cada refatoração feita, o código deve ser submetido a todos os testes de aceitação já desenvolvidos para garantir a compatibilidade com os demais códigos. (JEFFRIES, R. E., 1999).

2.3.3.7 ***Integração Contínua***

Integração contínua é o ato de manter a construção sempre atualizada em relação a novos códigos criados. O XP sugere tantas integrações quantas possíveis, como, por exemplo, para uma equipe de 40 desenvolvedores haverá no mínimo 10 compilações em um dia. Esta prática busca facilitar o processo de integração da nova característica desenvolvida evitando que problemas não identificados pelos testes de aceitação apareçam com o código já em produção. (JEFFRIES, R. E., 1999).

2.3.3.8 ***Posse Coletiva do Código***

A posse coletiva do código indica que nenhum código criado tem dono. Ou seja, qualquer código necessário estará aberto a qualquer desenvolvedor. Isto evita casos em que o criador da fonte não esta disponível para modificá-la caso necessário. O código sempre que alterado deverá passar pelos testes de aceitação, de forma que não há necessidade de um dono para gerenciar as alterações. (JEFFRIES, R. E., 1999).

2.3.3.9 Padronização do Código

A padronização do código é uma prática que regula a manutenção de um padrão de desenvolvimento na codificação cujo objetivo é fazer com que todo o código pareça ter sido desenvolvido por uma única pessoa facilitando futuras manutenções e melhorias do código. De tal forma que o XP não diz qual é o padrão, apenas que deve haver um. (JEFFRIES, R. E., 1999).

2.3.3.10 Metáfora

A técnica da metáfora utiliza-se desta figura de linguagem para descrever o funcionamento da aplicação, tornando mais didático o entendimento do sistema como um todo. Deve ser utilizada para manter um nível de comunicação comum entre as áreas de desenvolvimento e de negócio. (JEFFRIES, R. E., 1999).

2.3.3.11 Passo Sustentável

A última prática abordada cita que a equipe deve ter um ritmo de desenvolvimento sustentável, isto é, trabalhar sem deixar que isto leve os envolvidos a estafa mental. Existem momentos em que a equipe irá trabalhar além do expediente normal de trabalho e irá render acima do esperado, mas este ritmo não será e nem poderá ser constante. (JEFFRIES, R. E., 1999).

De acordo com KNIBERG, H. (p75, 2007), desenvolver incessantemente pode ser cansativo e não dá o tempo necessário à área de negócio para reprocessar as novas entregas e talvez redefinir prioridades. Uma pausa é essencial ao desenvolvedor e permitirá que entre as etapas de desenvolvimento seja possível buscar novas soluções e estudar outras tecnologias sem o compromisso e preocupação de entregar algo à área de negócios.

2.4 Práticas do Scrum

O Scrum é a metodologia ágil mais difundida, possuindo uma quantidade menor de regras em relação ao XP, todas focadas na gerencia do projeto e não define nenhuma prática de engenharia. (KNIBERG, H.; SKARIN, M., p31 2009). Abaixo encontra-se o ciclo de desenvolvimento do Scrum, com duas iterações, a iteração de desenvolvimento de requisitos do Sprint e a iteração de Sprints para o lançamento do produto.

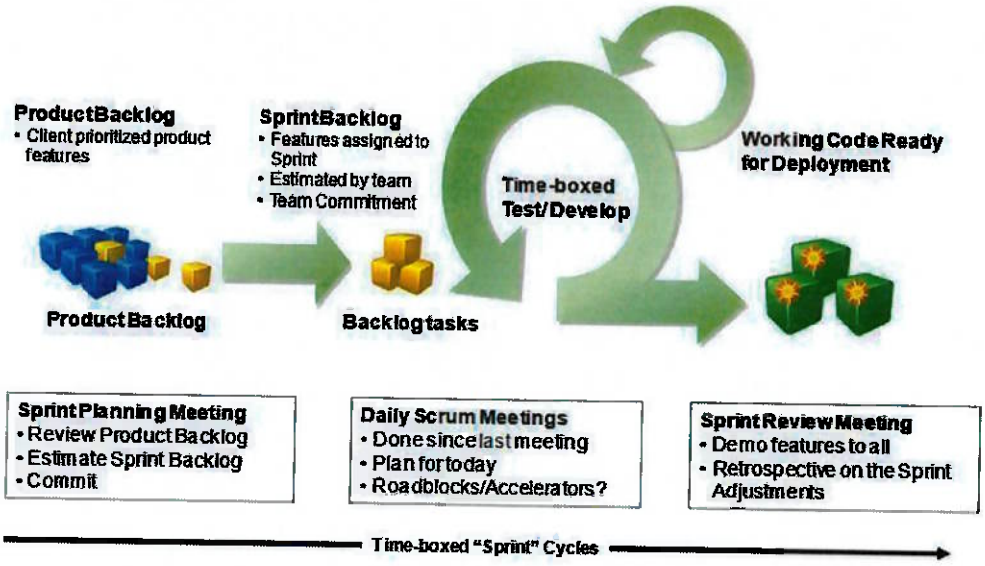

Figura 3: Scrum (AGILE Scrum, 2010)

Ao contrário do XP, o Scrum não diz como o desenvolvimento deve ser feito, apenas quando e como tornar o gerenciamento do processo melhor controlado. (SAVOINE, M. et al., [entre 2008 e 2011]).

2.4.1 Atores

Os três atores do Scrum são bem definidos e todos obrigatórios ao processo, são eles o Scrum Master, o Scrum Team Member e o Product

Owner. Eles foram descritos por SCHWABER, K.; SUTHERLAND, J. (2011) no Guia Scrum.

2.4.1.1 Scrum Master

O papel de Scrum Master é remover quaisquer impeditivos em relação às tarefas, deixando ao desenvolvedor apenas a preocupação de desenvolver e deve incorporar o papel de gerente do projeto sendo o responsável pelas reuniões e o conteúdo delas. O Scrum Master é crucial para assegurar que os valores do Scrum estão sendo seguidos e acaba muitas vezes assumindo a função do analista junto ao programador, tentando extrair mais detalhes sobre os requisitos nas reuniões com o dono do produto. (SCHWABER, K.; SUTHERLAND, J., p6-7 2011).

2.4.1.2 Product Owner

Product Owner é o responsável pelo planejamento dos requisitos por ser o único a gerenciar a lista de requisitos. Também chamado de PO, tem a liberdade de modificar as prioridades de cada requisito, exceto durante o Sprint, período em que há o desenvolvimento. A pessoa a assumir o papel deve ser alguém com poder de decisão no projeto. (SCHWABER, K.; SUTHERLAND, J., p6-7 2011).

2.4.1.3 Scrum Team Member

Scrum team member é o desenvolvedor, aquele que codifica os requisitos em produto. Para o Scrum o membro do time deve ser multidisciplinar participando dos seguintes itens:

- ✓ Levantamento de requisitos
- ✓ Elaboração da arquitetura
- ✓ Definição de pronto

- ✓ Estimativas do desenvolvimento
- ✓ Testes de aceitação
- ✓ Construção
- ✓ Integração
- ✓ Publicação em produção

Cabe ao membro do time toda a parte técnica como descrito por SCHWABER, K.; SUTHERLAND, J. (p8-9 2011).

2.4.2 Eventos

Existem seis eventos descritos no Guia Scrum por SCHWABER, K.; SUTHERLAND, J. (2011), sendo três reuniões para planejamento de etapas diferentes do:

- ✓ Lançamento
- ✓ Sprint
- ✓ Dia

E duas para melhorias no processo:

- ✓ Produto
- ✓ Desenvolvimento

E o próprio Sprint. Todos descritos por SCHWABER, K.; SUTHERLAND, J. (2011).

2.4.2.1 Reunião de Planejamento do Lançamento

Esta reunião tem como objetivo transformar a visão do PO em um produto que alcance ou supere as expectativas do cliente e o ROI (retorno sobre o investimento). Estes são os principais pontos que esta reunião deve esclarecer. Além disso, outras metas devem ser estipuladas, como uma lista de características e funcionalidades do software, data de entrega e custo. Participam desta reunião o PO, o Scrum Master e o desenvolvedor. (SCHWABER, K.; SUTHERLAND, J., p9-10 2011).

2.4.2.2 Reunião de Planejamento do Sprint

Na reunião de planejamento do Sprint deve-se planejar o que será desenvolvido no Sprint. São reuniões longas, pois os requisitos são quebrados em tarefas e a estimativa pouco precisa do planejamento do lançamento torna-se mais realista. O Scrum Guide define uma reunião de oito horas para um Sprint de um mês e devem estar presente os mesmos envolvidos no planejamento do lançamento. (SCHWABER, K.; SUTHERLAND, J., p12-14 2011).

2.4.2.3 Sprint

Sprint é o período de tempo que engloba o planejamento das etapas de desenvolvimento, revisão e retrospectiva. O desenvolvimento nesta etapa deve estar protegido contra mudanças.

A grande vantagem de frameworks ágeis é terem a iteração em um curto espaço de tempo, permitindo que o produto tenha um requisito entregue antes que possa acontecer uma mudança organizacional que mude o escopo da entrega. (SCHWABER, K., SUTHERLAND, J., p11-12 2011).

2.4.2.4 Revisão do Sprint

Revisão do Sprint é o momento em que o time se reúne para demonstrar ao PO o que foi feito, o que não foi feito e porquê. Neste momento apontam-se possíveis melhorias futuras, quais foram os problemas enfrentados e como foram resolvidos para então o PO fazer seus questionamentos, assim há um alinhamento entre o que era esperado pelo PO e o que foi entregue pela equipe de desenvolvimento. Pode-se utilizar a medição de velocidade da equipe nesta iteração para prever o restante do

Product Backlog. Para esta reunião prevê-se quatro horas para um mês de Sprint. (SCHWABER, K.; SUTHERLAND, J., p14-15 2011).

2.4.2.5 Retrospectiva do Sprint

Retrospectiva do Sprint é a reunião onde se inspeciona como ocorrera o ultimo Sprint para melhoraria do processo, seja por troca de ferramentas, meios de comunicação ou ainda de interações interpessoais. De menor duração, três horas para um mês de Sprint, nela participam o Desenvolvedor Solo e o Scrum Master. (SCHWABER, K.; SUTHERLAND, J., p15-16 2011).

2.4.2.6 Daily Scrum

Daily Scrum é uma reunião diária para os desenvolvedores compartilharem as informações quanto ao andamento da construção do projeto. Revendo quais atividades devem ser realizadas no dia. (SCHWABER, K.; SUTHERLAND, J., p16-17 2011).

2.4.3 Artefatos

Existem quatro artefatos gerados pelo Scrum, duas listas de requisitos para planejamento e dois gráficos para controle da velocidade do projeto, também descritos no Guia Scrum. (SCHWABER, K.; SUTHERLAND, J., 2011).

2.4.3.1 Product Backlog

O Product Backlog é o artefato gerado inicialmente a partir da reunião de planejamento do lançamento, ele possui todas as características, requisitos e regras de negócio do produto. O PO tem total liberdade para

alterar a prioridade de cada item, inclusive de adicionar novos itens ou aprovar itens sugeridos para o produto. Este artefato deve conter para cada item, o tempo de desenvolvimento estimado, ainda que sem exatidão. (SCHWABER, K.; SUTHERLAND, J., p17-20 2011).

2.4.3.2 ***Burndown do Lançamento***

O Burndown do Lançamento é o artefato para visualização do PO referente à quando o lançamento estará pronto. Somam-se todas as estimativas de cada requisito do Product Backlog e subtrai-se o que foi realizado ao longo do tempo. Como o realizado tende ao total, quando chegar à zero significa que o lançamento está pronto. (SCHWABER, K.; SUTHERLAND, J., p17-20 2011).

2.4.3.3 ***Sprint Backlog***

O Sprint Backlog consiste em uma lista de tarefas para a realização de cada item do Product Backlog escolhido para o Sprint. Cada tarefa deve ter uma estimativa muito próxima à realidade. Novas tarefas poderão surgir durante o Sprint e a equipe é responsável por adicioná-las ao Sprint Backlog. Toda alteração durante o Sprint deve ser reportada ao PO na revisão do Sprint. (SCHWABER, K.; SUTHERLAND, J., p20-21 2011).

2.4.3.4 ***Burndown do Sprint***

Similar ao Burndown do Lançamento, porém válido apenas para as tarefas do Sprint, o Burndown do Sprint deve ajudar não só ao PO, mas também ao Scrum Master a visualizar possíveis dificuldades nas tarefas. É possível ver se os requisitos foram bem mitigados em tarefas pequenas e se o planejamento está de acordo com as estimativas. (SCHWABER, K.; SUTHERLAND, J., p20-21 2011).

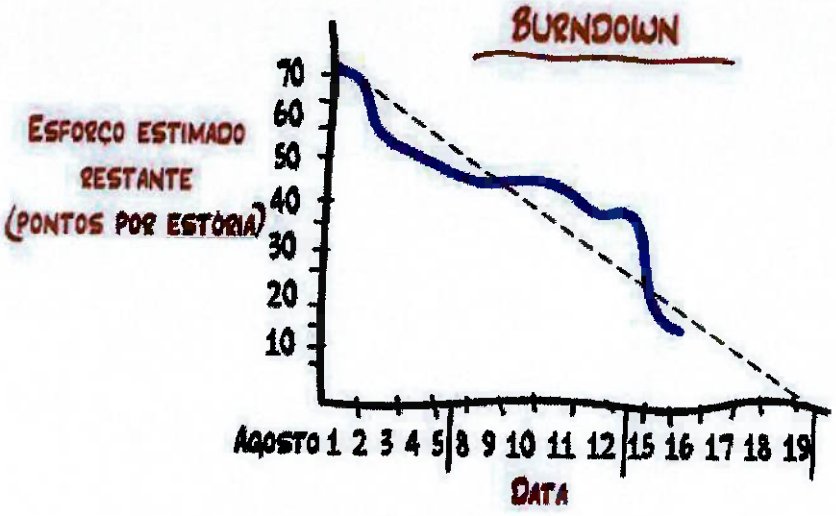

Gráfico 1: Burndown do Sprint (KNIBERG, H., p53 2007)

2.5 Práticas do Kanban

A proveniente da engenharia de processos, desenvolvido junto ao Sistema Toyota de Produção (MARTINS, P. G.; LAUGENI, F. P., p404-405 2006), o Kanban é mais adaptativo, descreve apenas três práticas de engenharia para garantir a execução contínua das tarefas, ou seja, sem gargalos. Os gargalos ocorrem sempre que o número de tarefas em uma parte do processo é maior do que o número de tarefas que pode ser executado. O Kanban basicamente auxilia, através de um quadro, a visualizar todas as tarefas e qual o status delas no processo. (KNIBERG, H.; SKARIN, M., p30-31 2009).

Abaixo encontra-se um exemplo de quadro Kanban com cinco estados, cada um com o número máximo de tarefas já delimitado.

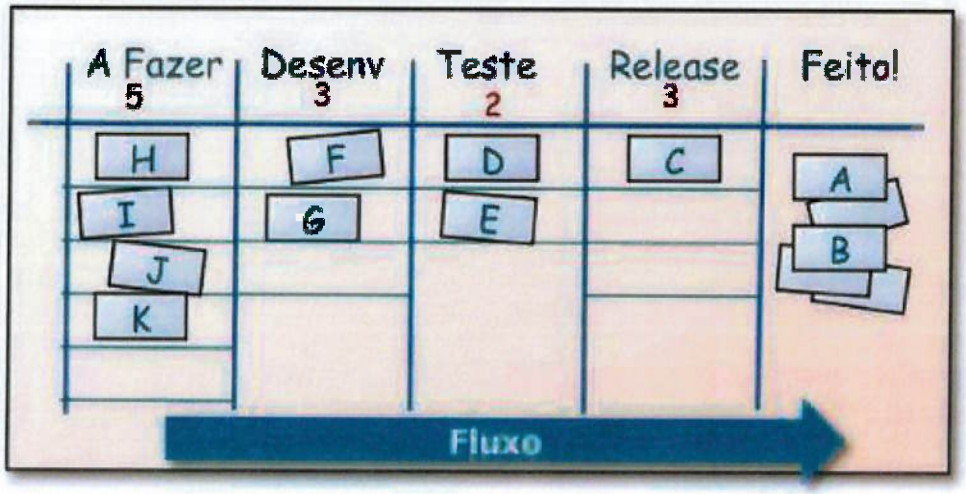

Figura 4: Kanban (KNIBERG, H.; SKARIN, M., p26 2009)

2.5.1 Práticas de Engenharia

O Kanban descreve apenas três práticas de engenharia, o suficiente para que se possa acompanhar o fluxo das tarefas durante todo o processo de desenvolvimento. (KNIBERG, H.; SKARIN, M., p25-26 2009).

2.5.1.1 Crie e visualize o fluxo de trabalho

Criar e visualizar o fluxo de trabalho é a primeira regra para se montar o quadro do fluxo das tarefas durante o processo. Coloca-se cada tarefa a ser executada em um cartão e então devem ser criadas algumas colunas para marcar o status de cada uma. Cria-se um fluxo, por exemplo, horizontal da esquerda para a direita e posicionam-se os cartões no extremo esquerdo. À medida que os trabalhos passam de um status para outro, trocamos o cartão referente ao trabalho feito da coluna. (KNIBERG, H.; SKARIN, M., p25 2009). Desta forma é possível visualizar todos os trabalhos em execução e seu status.

O Kanban não estipula nenhum status específico, porém implementações junto ao Scrum sugerem pelo menos três status, o não

iniciado, o iniciado e o pronto. Não limitados a estes três, conforme exposto por KNIBERG, H. (p49 2007).

2.5.1.2 *Limitar os trabalhos executados por estado do processo*

Devem-se impor limites de trabalho por status ou colunas, deste modo tornam-se possível identificar e evitar gargalos no processo de produção. Existem passagens de status que ocorrem de forma mais rápida e outras que demoram mais, por isso é essencial limitar corretamente a quantidade de tarefas por etapa do processo. (KNIBERG, H.; SKARIN, M., p25 2009).

2.5.1.3 *Acompanhar o tempo de execução das tarefas*

O acompanhamento do tempo de execução das tarefas tem como objetivo mapear o tempo médio de passagem pelos estados do processo. Desta forma pode-se retirar e evitar gargalos, mantendo o fluxo sempre contínuo. O Kanban prevê, com o decorrer do tempo e com a maturidade no processo, um fluxo sem gargalos, com um ritmo certo, assim como uma linha de produção fabril. (KNIBERG, H.; SKARIN, M., p26 2009).

2.6 Prescrição e Adaptação

Os autores KNIBERG, H.; SKARIN, M. (p30-31 2009), analistas de metodologia ágil de desenvolvimento, adotaram uma métrica para comparar diferentes metodologias de desenvolvimento de software, esta escala compara quanto prescritivo ou adaptativo é o framework. Um framework é mais prescritivo quanto maior o número de regras for descritas para a sua conformidade. Logicamente quanto mais prescritiva, menos adaptativa, e por isso metodologias como o UP (Unified Process), que é muito prescritivo,

tendem a possuir um processo lento e burocrático ideal para projetos e equipes grandes, pois existe um número maior de regras prescritas indicando mais eventos, artefatos, atores e práticas de engenharia resultando em um esforço maior para gerenciar o processo. Do outro lado, um processo excessivamente adaptativo não possui regras, ou seja, não é um processo.

É possível então situar as metodologias citadas. Respectivamente, o XP, o Scrum e o Kanban, nesta ordem, do mais prescritivo ao mais adaptativo. O XP é um framework voltado a atividades técnicas para melhoria do desenvolvimento. Já o Scrum define apenas regras de mote gerencial. Por ultimo o Kanban é uma metodologia com três regras, o suficiente para manter um fluxo de trabalho constante.

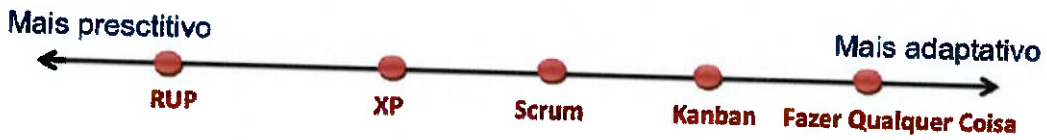

Figura 5: Métrica prescritiva adaptativa (KNIBERG, H.; SKARIN, M., p30 2009)

Um maior número de regras não faz do framework melhor ou mais completo, tão pouco faz a afirmação contrária. Ambos os extremos da prescrição ou adaptação não são viáveis, excesso de regras torna o desenvolvimento lento, já falta de regras causa desorganização no gerenciamento.

Cada metodologia ágil estudada neste trabalho oferece restrições sobre o que ser feito orientando assim quem as adota, são ferramentas diferentes para um mesmo propósito. (KNIBERG, H.; SKARIN, M., p28-29 2009).

3 PROPOSTA DO NOVO ARCABOUÇO

3.1 Comparação entre os Frameworks Estudados

A comparação dos arcabouços estudados previamente servirá análise do foco de cada metodologia e com base nisso será justificada a adaptação de um novo framework.

3.1.1 Scrum

O Scrum descreve apenas práticas para facilitar a gerência do projeto, levando em conta que o desenvolvedor possui expertise suficiente para auto gerenciar a parte técnica. Como mostra a tabela abaixo, não há nenhuma regra quanto à parte técnica do desenvolvimento do software, apenas regras que auxiliam a gerência do projeto. A seguir é possível observar através da tabela o foco do Scrum.

Metodologia	Tipo de Prática	Nome	Regra
Scrum	Ator	Scrum Master	Gerencial
	Ator	Product Owner	Gerencial
	Ator	Desenvolvedor Solo Member	Gerencial
	Eventos	Reunião de Planejamento do Lançamento	Gerencial
	Eventos	Sprint	Gerencial
	Eventos	Reunião de Planejamento da Sprint	Gerencial
	Eventos	Revisão da Sprint	Gerencial
	Eventos	Retrospectiva da Sprint	Gerencial
	Eventos	Daily Scrum	Gerencial
	Artefatos	Product Backlog	Gerencial
	Artefatos	Burndown do Lançamento	Gerencial
	Artefatos	Sprint Backlog	Gerencial
	Artefatos	Burndown da Sprint	Gerencial

Tabela 1: Regras do Scrum

3.1.2 eXtreme Programming (XP)

No XP observa-se o inverso do Scrum, existem muitas práticas de engenharia que demonstram maior preocupação na parte técnica do desenvolvimento apenas definindo alguns momentos de planejamento. Podemos verificar inclusive que os artefatos gerados são para apoio ao desenvolvimento e não para gerencia do projeto.

Metodologia	Tipo de Prática	Nome	Regra
XP	Ator	Gerente	Gerencial
	Ator	Cliente	Gerencial
	Ator	Testadores	Gerencial
	Ator	Analistas	Gerencial
	Ator	Programadores	Gerencial
	Ator	Treinador / Técnico	Gerencial
	Eventos	Planejamento do Lançamento	Gerencial
	Eventos	Planejamento da Iteração	Gerencial
	Artefatos	Testes de Aceitação	Técnica
	Artefatos	Deploy Contínuo	Técnica
	Práticas de Engenharia	Lançamentos Pequenos	Técnica
	Práticas de Engenharia	Design Simples	Técnica
	Práticas de Engenharia	Programação em Par	Técnica
	Práticas de Engenharia	Test Driven Development	Técnica
	Práticas de Engenharia	Refatoração	Técnica
	Práticas de Engenharia	Integração Contínua	Técnica
	Práticas de Engenharia	Posse Coletiva do Código	Técnica
	Práticas de Engenharia	Padronização do Código	Técnica
	Práticas de Engenharia	Metáfora	Técnica
	Práticas de Engenharia	Passo Sustentável	Técnica

Tabela 2: Regras do XP

3.1.3 Kanban

O Kanban descreve apenas regras suficientes para que se tenha um fluxo de trabalho contínuo, esta metodologia é utilizada em diversas áreas empresariais, pois não descreve atores ou eventos específicos da área de

TI. Estas práticas auxiliam o gerenciamento do projeto, pois permite rápida visualização das atividades e dos seus respectivos status.

Metodologia	Tipo de Prática	Nome	Regra
Kanban	Práticas de Engenharia	Crie um fluxo de trabalho	Gerencial
	Práticas de Engenharia	Limitar os trabalhos executados por estado do processo	Gerencial
	Práticas de Engenharia	Acompanhar o tempo de execução das tarefas	Gerencial

Tabela 3: Regras do Kanban

3.2 Metodologia

A proposta de adaptação do arcabouço baseia-se nas três metodologias ágeis estudadas, utilizando como esqueleto o Scrum, que possui mais regras gerenciais (KNIBERG, H., p84 2007) e então complementadas com praticas de engenharia provenientes do XP e do Kanban, sempre tendo em vista o cenário proposto do desenvolvedor solo dentro de uma empresa.

O arcabouço será idealizado conforme de acordo com a análise feita do Scrum, XP e Kanban, ou seja, uma soma de atores, eventos, artefatos e práticas de engenharia. O resultado segue o conceito ágil, não sendo requisito ser mais, ou menos, prescritivo em relação ao Scrum, a métrica prescritiva x adaptativa apenas será adotada para objetivo de comparação.

Ao fim da adaptação será feita a comparação com base na métrica já citada quanto á quantidade de regras e situaremos na reta prescritiva x adaptativa feita por KNIBERG, H.; SKARIN, M. (p30 2009).

3.3 Os Atores

O Scrum será utilizado como base para a adaptação Solo, pois não há necessidade da especificidade das funções técnicas do XP uma vez que no cenário solo há somente um desenvolvedor, de forma que os atores originais do Scrum serão mantidos, o Scrum Master, Product Owner, e o Scrum Team Member (que no cenário solo torna-se o Desenvolvedor Solo), porém todos os papéis descritos no XP serão absorvidos, uma vez que suas funções são importantes ao projeto.

Com base no cenário do desenvolvedor solo, todos os papéis técnicos devem pertencer ao Desenvolvedor Solo, que deve ser multidisciplinar, pois o desenvolvedor será único, de forma que tanto o papel do testador quanto ao do analista ficarão com o Desenvolvedor Solo.

O Scrum Master por sua vez acumula o papel de gerente, de treinador, e também pode atuar como analista ou consultor de desenvolvimento, ou seja, não somente assuntos relativos ao uso do framework, mas também àqueles externos à equipe que possam ser de alguma forma ser um impeditivo ao desenvolvimento e à evolução do projeto. No Scrum, o Scrum Master é o treinador, porém é comum devido à proximidade junto aos desenvolvedores que gerencie o projeto e por ter experiência em outros projetos pode ajudar na comunicação com o Product Owner nas reuniões para levantamento de requisitos.

O Product Owner do Scrum tem a mesma função que o cliente do XP, prover requisitos para os projetos e dar prioridade a eles.

Por ultimo, apesar de não ser descrito no Guia Scrum por SCHWABER, K.; SUTHERLAND, J. (2011), será adotado o ator Outros para descrever qualquer interessado que sugerir um novo requisito para o produto. Isto não significa que será feito, pois somente o PO tem poder para decidir isto, mas os interessados podem adicionar itens como sugestão conforme descrito por KNIBERG, H. (p11 2007).

3.4 Os Artefatos

Serão mantidos os quatro artefatos gerenciais que o Scrum contém, além de mais um artefato do XP e outro do Kanban. As listas e gráficos gerados pelo Scrum e o quadro gerado pelo Kanban auxiliam o controle do processo de desenvolvimento, já o artefato do XP auxilia na apresentação do produto.

Deve-se manter o Product Backlog do Scrum contendo todos os requisitos de negócio extraídos do cliente. Esta lista de itens contém tudo que será desenvolvido em relação ao projeto e com base nela será possível estipular, ainda que sem precisão, quando o lançamento estará pronto e qual o custo do projeto a cada etapa.

Também do Scrum deve-se manter o Burndown do Lançamento que informa a velocidade da equipe no projeto, auxiliando a prever a data de entrega de novos requisitos. Com ele será possível obter alguns indicadores de requisitos lançados em um período de tempo.

O artefato Sprint Backlog será mantido, pois contém a lista do que será desenvolvido na iteração atual, com cada requisito já mitigado em tarefas pequenas com duração de um dia, permitindo assim prever a data de entrega. Esta lista é fundamental para saber o tamanho do esforço a ser feito pelo desenvolvedor no ciclo atual de desenvolvimento.

Ainda do Scrum, deve-se aproveitar o Burndown do Sprint com a velocidade do desenvolvedor no Sprint que permite melhor controle sobre o andamento das tarefas e identificação de dificuldades. Na retrospectiva deve-se utilizar este artefato para que futuros erros não sejam cometidos e para que futuras dificuldades não existam mais.

Do XP devem-se aproveitar os testes de aceitação que são montados com base na definição de pronto de cada item dos Backlogs. Os testes direcionam o desenvolvimento uma vez que o requisito só está pronto se passar pelos testes de aceitação.

Por fim, proveniente do Kanban, deve-se ter um Quadro Kanban exibindo o fluxo de trabalho e os limites de trabalho por estado. O quadro é feito por iteração e deve ser atualizado ao fim do dia, sendo utilizado diariamente pelo Scrum Master que deve analisar o andamento das tarefas e se alguma há demora em alguma tarefa.

3.5 Os Eventos

Dentre os frameworks estudados, o Scrum é quem melhor controla o processo de iterações de desenvolvimento e por isso será utilizado como base para a adaptação com a adição de duas práticas de engenharia, uma do XP, o passo sustentável, e outra do Kanban, a criação do quadro Kanban.

Os eventos determinam o processo adotado e suas etapas, que no total são oito etapas. O processo inicia-se planejando o lançamento e com os requisitos em mãos faz-se o quadro Kanban, quando então se iniciam as iterações de Sprints.

Planejam-se cada iteração do Sprint iniciando-se com o desenvolvimento. Ao se finalizar a construção do produto, faz-se a revisão e a retrospectiva do Sprints e finalmente, antes de se iniciar um novo Sprint, baseado no principio do passo sustentável do XP, faz-se um intervalo, que tem como função marcar o fim de uma iteração de desenvolvimento.

3.5.1 Planejamento do Lançamento

O inicio do projeto ocorre na Reunião de Planejamento do Lançamento, onde participam o Scrum Master, o Desenvolvedor Solo e o PO (Product Owner). O PO deve revelar todos os requisitos desejados para o próximo lançamento, já o Desenvolvedor Solo, que no cenário é um único desenvolvedor, e o Scrum Master devem não só entender o que é pedido, atuando como analistas de negócio, mas também garantir que todos tenham entendido o requisito de forma igual.

O PO (Product Owner) pode coletar de outros interessados requisitos não identificados. No Scrum somente o PO pode solicitar novos requisitos, porém na prática, é comum existirem diversos interessados no produto que irão opinar e dar sugestões de forma que o PO deve coletar e filtrar estes requisitos.

Com os requisitos em mãos inicia-se a construção do Product Backlog, que deve ser resumidamente descrito com a participação dos três papeis envolvidos na reunião de forma que a definição do item esteja clara e tenha o mesmo significado para todos.

Após definir cada requisito, o desenvolvedor deve prever junto ao Scrum Master o tamanho do esforço em dias para a entrega, não sendo esta técnica objetivo desta monografia.

As estimativas em dias mostram a diferença entre um requisito e uma tarefa que deve ser menor ou igual a um dia para que se tenha melhor controle sobre o fluxo de trabalho. Em casos de requisitos menores que um dia, assume-se um requisito com uma única tarefa. As estimativas devem também ser menores que semanas, pois a entrega do Sprint é calculada em semanas para justificar a agilidade proposta pela metodologia.

A definição do requisito inclui saber a definição de pronto que deve contemplar o comportamento do requisito quando finalizado, ou ainda, como demonstrar que está pronto. Isto irá facilitar o desenvolvimento do requisito e também ajudará a montar os testes de aceitação.

A demonstração que o requisito atende à solicitação dá-se em sua entrega e baseia-se na definição de pronto, que por isso é de alta importância para o projeto, pois se mal feita o requisito entregue pode não atender as expectativas.

É através desta definição que o desenvolvedor consegue seguir o conceito de projeto simples proposto pelo XP, pois somente deve-se desenvolver o suficiente para que o requisito esteja pronto de acordo com definição, sem esforço extra. Qualquer característica do produto, por menor que seja deve estar incluída na definição de um requisito ou no Backlog para que seja desenvolvido, caso contrário torna-se esforço extra e portanto desnecessário.

O requisito deve contemplar somente o que for solicitado pelo PO, o desenvolvedor deve estar ciente que qualquer hora de desenvolvimento gasta com algo a mais poderia ter sido gasta com outro requisito do Backlog e que somente o PO poderia ter trocado a prioridade entre o algo a mais e o próximo item do Backlog.

O Guia Scrum (SCHWABER, K.; SUTHERLAND, J., 2011) cita a reunião de planejamento do lançamento como opcional, pois a falta de seus artefatos surgirá como impeditivos para o desenvolvimento que deverão ser solucionados, porém ela será obrigatória para o arcabouço solo. Nesta reunião oficializa-se o inicio de um projeto, introduzem-se os envolvidos e obtém-se uma visão geral do que será desenvolvido. Deixar como opcional apenas criaria problemas futuros, pois não existirá uma lista geral de requisitos para o projeto, o que impede uma previsão para o final do projeto.

O tempo de reunião irá depender da quantidade de requisitos identificados pelo Product Owner. Por isso é necessária a agenda livre de todos os interessados possivelmente para o dia todo.

Após a reunião o Scrum Master deverá criar o artefato de Burndown do Lançamento, que deverá ser atualizado a cada Semana. Através do Burndown do Lançamento podem-se extrair informações de complexidade por item, velocidade de desenvolvimento e uma previsão aproximada da data de entrega do lançamento.

Abaixo segue o diagrama BPMN do planejamento descrito.

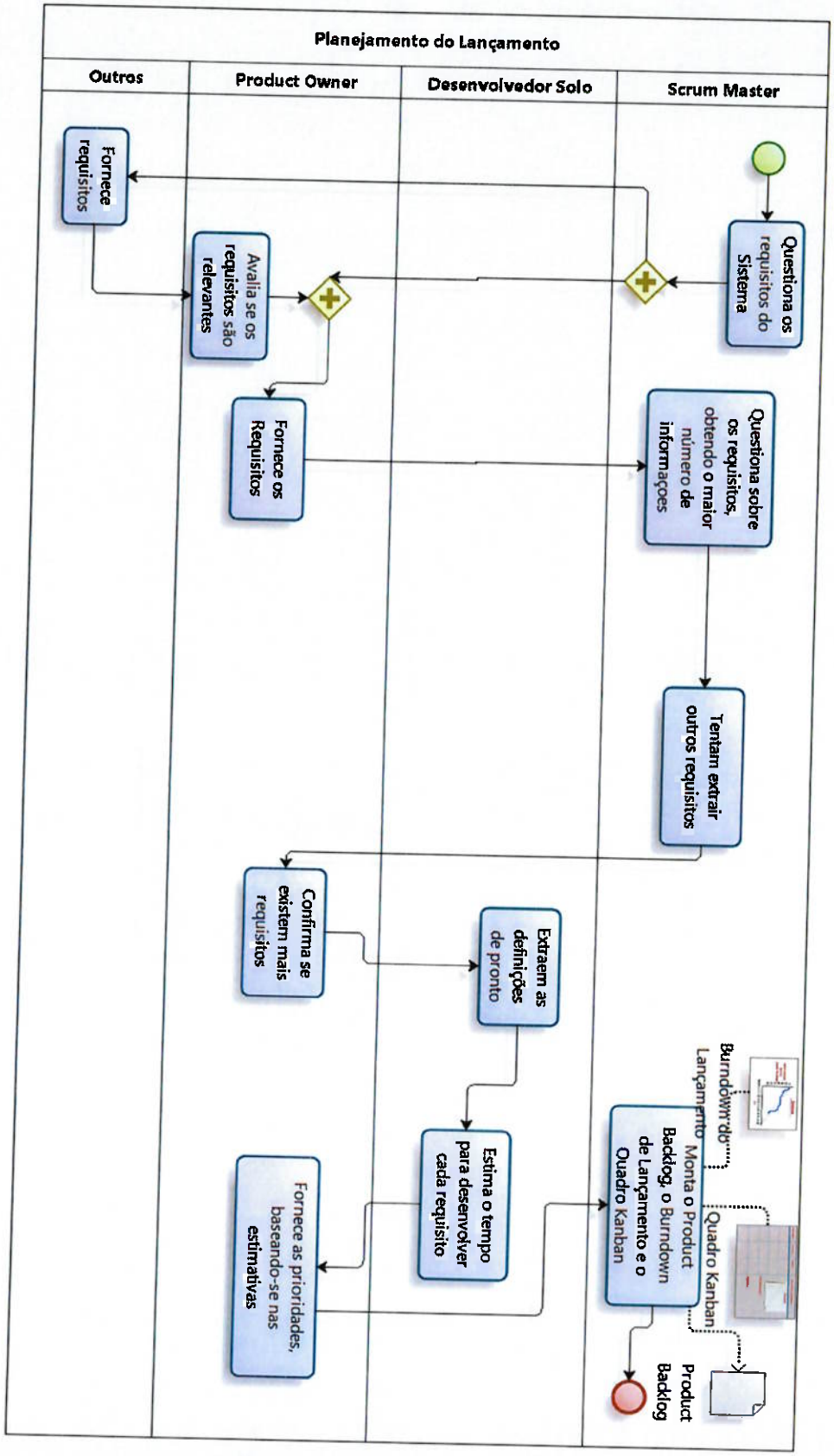

Figura 6: BPMN do Planejamento do Lançamento

3.5.2 Criação do Quadro Kanban

O quadro Kanban tem a função de tornar facilmente visível as tarefas e o estado da execução de cada uma. Isto permite ao PO verificar o andamento do desenvolvimento em tempo real, permite ao Scrum Master controle sobre impeditivos durante o Sprint ao ver tarefas paradas em um estado por mais tempo que o planejado e permite ao desenvolvedor visualizar suas tarefas.

Após planejar o lançamento define-se como será montado o quadro Kanban. Primeiramente devem-se identificar quais são os estados de trabalho para então identificar quem participa de cada estado e a complexidade de cada um. Com isso é possível realizar uma previsão do limite de trabalho por estado.

O Quadro deve ser ajustado com o tempo. Ele deve ser um painel de controle do Sprint e para se aproveitar de seus benefícios, como o controle dos estados das tarefas, deve estar sempre atual.

É comum em equipes que nunca tenham trabalhado com Kanban que se definam quadros simples com estados como "Não Feito", "Fazendo" e "Feito" utilizando um limite igual ao número de responsáveis. Para estes estados, como o responsável é o próprio desenvolvedor, que no cenário estudado é solo, então o limite seria um, pois o desenvolvedor poderá executar apenas uma tarefa por vez, mais tarefas causariam um gargalo no processo.

O quadro pode, porém, ser alterado adicionando-se um estado aumentando assim o nível de controle sobre o processo de desenvolvimento, como por exemplo, "Retrospectiva Realizada" tendo como responsável o Scrum Master. Desta forma ele poderia gerir mais de um desenvolvedor / projeto ao mesmo tempo, colocando-se um limite maior de tarefas neste estado sem que existam gargalos no fluxo.

3.5.3 O Sprint

O Sprint engloba tudo que acontece dentro da iteração, planejamento, desenvolvimento, revisão, retrospectiva e descanso. A definição do tempo de duração do Sprint pode mudar, sendo mais adequado defini-lo na fase de planejamento.

No cenário do desenvolvedor solo, o Sprint fica em sua maioria sob responsabilidade deste, pois ele é o único que participa de todas as etapas, no planejamento é quem dá os prazos e quem estipula as tarefas, no desenvolvimento é quem gera o produto, na revisão é quem entrega e demonstra a entrega, na retrospectiva é quem dá as informações necessárias para futuras melhorias e no descanso é quem procurará novas tecnologias para serem aplicadas no futuro.

Fica claro então que a peça chave do Sprint é o desenvolvedor, o Scrum Master deve fazer com que o desenvolvedor não se preocupe com nada além do descrito acima, mantendo o foco nos objetivos destinados ao papel do Desenvolvedor Solo.

3.5.3.1 Planejamento do Sprint

O planejamento do Sprint é a reunião onde se determinam quais requisitos do Product Backlog serão feitos e quais são as tarefas para que cada requisito escolhido seja desenvolvido. Isto inclui uma reanálise das estimativas, uma vez que divididos em tarefas as estimativas podem mudar.

O arcabouço solo faz com que o desenvolvedor seja diretamente responsável pelas estimativas e se comprometa com elas, pois somente ele pode definir os prazos.

O planejamento do Sprint é a reunião que envolve Scrum Master, Desenvolvedor Solo e o Product Owner, inicia-se combinando uma duração para o Sprint. O recomendado por KNIBERG, H.; SKARIN, M. (p21 2009) são de três semanas sendo recomendado por SCHWABER, K.; SUTHERLAND, J. (p12 2011) duas horas de reunião por semana de Sprint.

Com o Product Backlog em mãos, já ordenado por importância, o desenvolvedor solo deve selecionar os primeiros itens por ordem de prioridade que conseguir entregar no período definido e começar a refinar os

itens em tarefas, sendo ideal que cada tarefa tenha no máximo um dia de duração, pois quanto menor a tarefa melhor é o controle de tarefas executadas sendo mais fácil identificar gargalos e dificuldades no desenvolvimento.

Para cada tarefa é dada uma estimativa sendo que a estimativa do requisito é atualizada para ser igual à soma das estimativas das tarefas. Conforme descrito anteriormente, a previsão para o desenvolvimento do requisito deve ser modificada nesta reunião.

Nesta reunião a previsão criada é mais precisa do que a feita no planejamento do lançamento e o Product Owner pode mudar a prioridade do item por causa da nova estimativa, podendo inclusive trocar por um item que anteriormente ficaria fora deste Sprint.

Feita a nova estimativa para os itens, escolhe-se tantos itens quantos forem possíveis de ser entregues na duração combinada do Sprint e então deve-se selecionar também os dois próximos itens que ficariam para o próximo Sprint, para o caso da estimativa estar errada e os itens ficarem prontos muito antes da previsão.

O Sprint Backlog, que é o artefato com a seleção dos itens e definição das tarefas com seus respectivos prazos, pode ser gerado para então se montar o fluxo no quadro Kanban.

Cada requisito é uma linha do quadro Kanban, com todas as suas tarefas anexadas à coluna "Não Feito". O limite de tarefas para o desenvolvedor solo na coluna "Fazendo" é de uma tarefa apenas, o que significa que somente uma tarefa poderá ser executada por vez.

Na medida em que o desenvolvedor assume uma tarefa a ser feita, e somente uma, o cartão referente a ela vai para o próximo estado, "Fazendo" e por fim quando termina o cartão vai para a ultima coluna "Feito".

Ainda no quadro, desenha-se o gráfico Burndown do Sprint, colocando-se no eixo das ordenadas a soma das estimativas e na abscissa os dias do Sprint. A equação da reta é de primeiro grau com o coeficiente angular negativo, ou seja, isto significa dizer que a soma das estimativas tende a zero no decorrer do desenvolvimento, pois a quantidade de tarefas não feitas deve se esgotar ao final do Sprint.

O quadro deve sempre estar atualizado em tempo real, ou seja, toda vez que uma nova tarefa mudar de estado, ou que haja uma tarefa não planejada, ou que haja uma correção de estimativa o Desenvolvedor Solo ou o Scrum Master devem atualizá-lo.

O quadro deve representar o andamento do Sprint e qualquer interessado deve conseguir esta informação do quadro sem ter que perguntar nada ao desenvolvedor.

Abaixo o BPMN do planejamento do Sprint descrito.

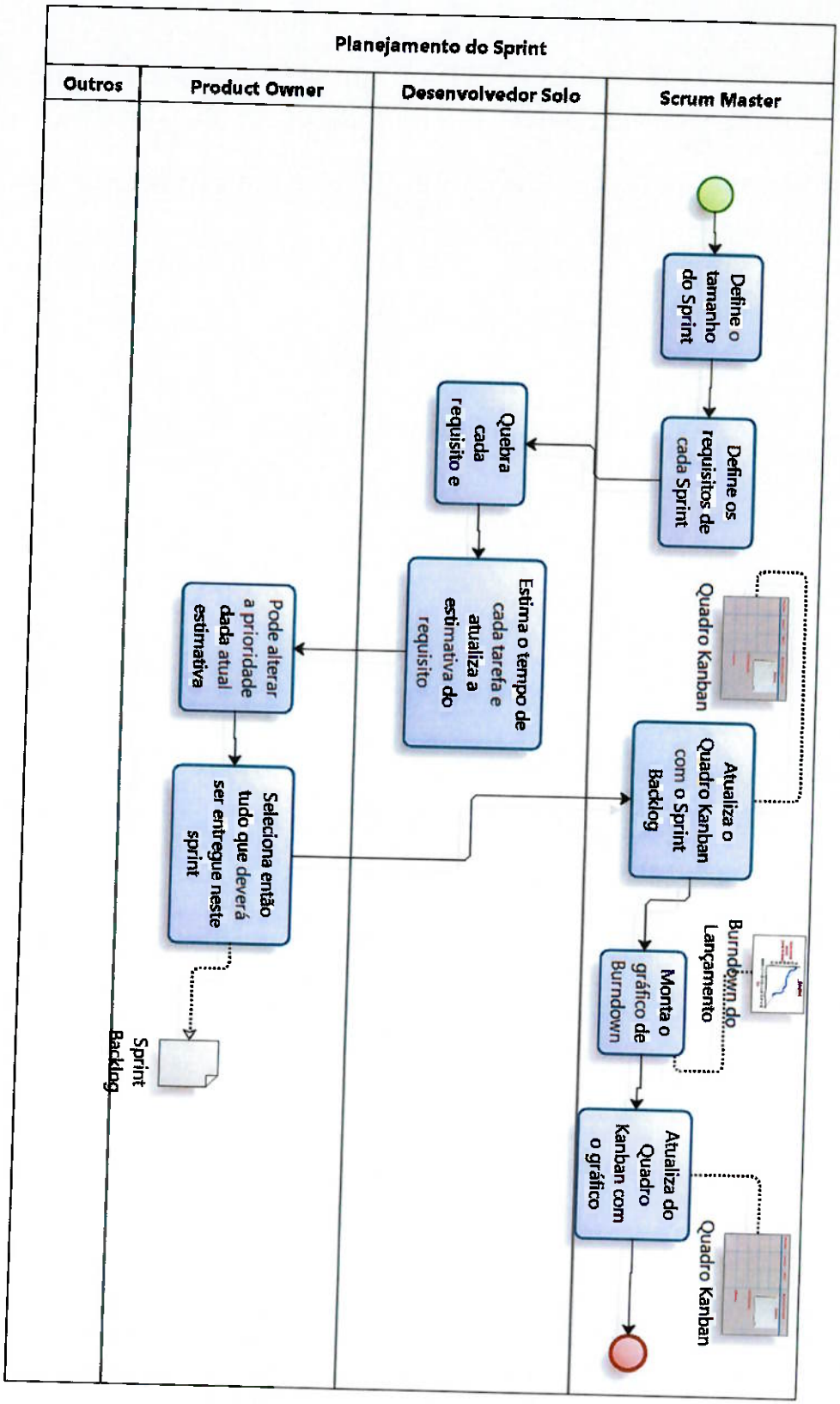

Figura 7: BPMN do Planejamento do Sprint

3.5.3.2 Desenvolvimento

Com o quadro de tarefas montado, inicia-se o desenvolvimento. E para tanto será adotada uma técnica do XP chamada Test Driven Development, ou TDD.

Esta técnica dá mais segurança ao desenvolvedor no decorrer do projeto, pois garante que novos códigos não gerem falhas em outras partes do software, uma vez que todo novo código deve passar pelos testes de aceitação já criados, estes erros são corrigidos ainda em tempo de desenvolvimento.

Por estar desenvolvendo sem ajuda, a segurança que o TDD traz ao Desenvolvedor Solo, faz com que seja essencial.

Esta técnica baseia-se nos testes de aceitação especificados no Backlog, para que assim as regras de negócio estejam asseguradas por testes automatizados.

Primeiramente desenvolve-se o teste que garantirá que o código atenda o propósito, então deve-se desenvolver o requisito até que ele passe pelo teste. Enquanto não houver aceitação no teste imposto pelo TDD, o código não estará pronto.

O hábito de desenvolver iniciando pelos testes unitários não é comum e tão pouco natural, por isso o Scrum Master deve ajudar o desenvolvedor a adquirir este hábito. Sempre que necessário, o Scrum Master pode juntar-se com o desenvolvedor e sugerir alterações no código, revisando quaisquer condutas que não condigam com os valores do framework adotado.

Outra prática é a integração contínua, sempre que terminar de desenvolver um novo requisito o Desenvolvedor Solo deve integrá-lo a um ambiente de homologação, ambientes de homologação deve ter aplicação rodando com sua ultima versão sendo utilizada para aprovação do produto.

Apesar de consumir mais tempo, manter o ambiente atualizado servirá também para o Product Owner conferir o andamento do produto, de forma que na próxima reunião de revisão o PO poderá ir já com os comentários prontos e pontuais, isto fará com que a reunião seja mais objetiva e rápida.

Não serão impostas tantas práticas de engenharia quantas são no XP, pois tornaria o desenvolvimento muito burocrático para apenas um desenvolvedor, estas práticas podem ser assimiladas eventualmente, porém forçá-las desde o início tornaria o processo menos ágil. As práticas serão citadas por serem consideradas boas práticas. Manter lançamentos pequenos para identificar alguns erros imperceptíveis aos testes unitários, projeto simples para não criar nada desnecessário e padrão no código.

Abaixo o BPMN do Desenvolvimento.

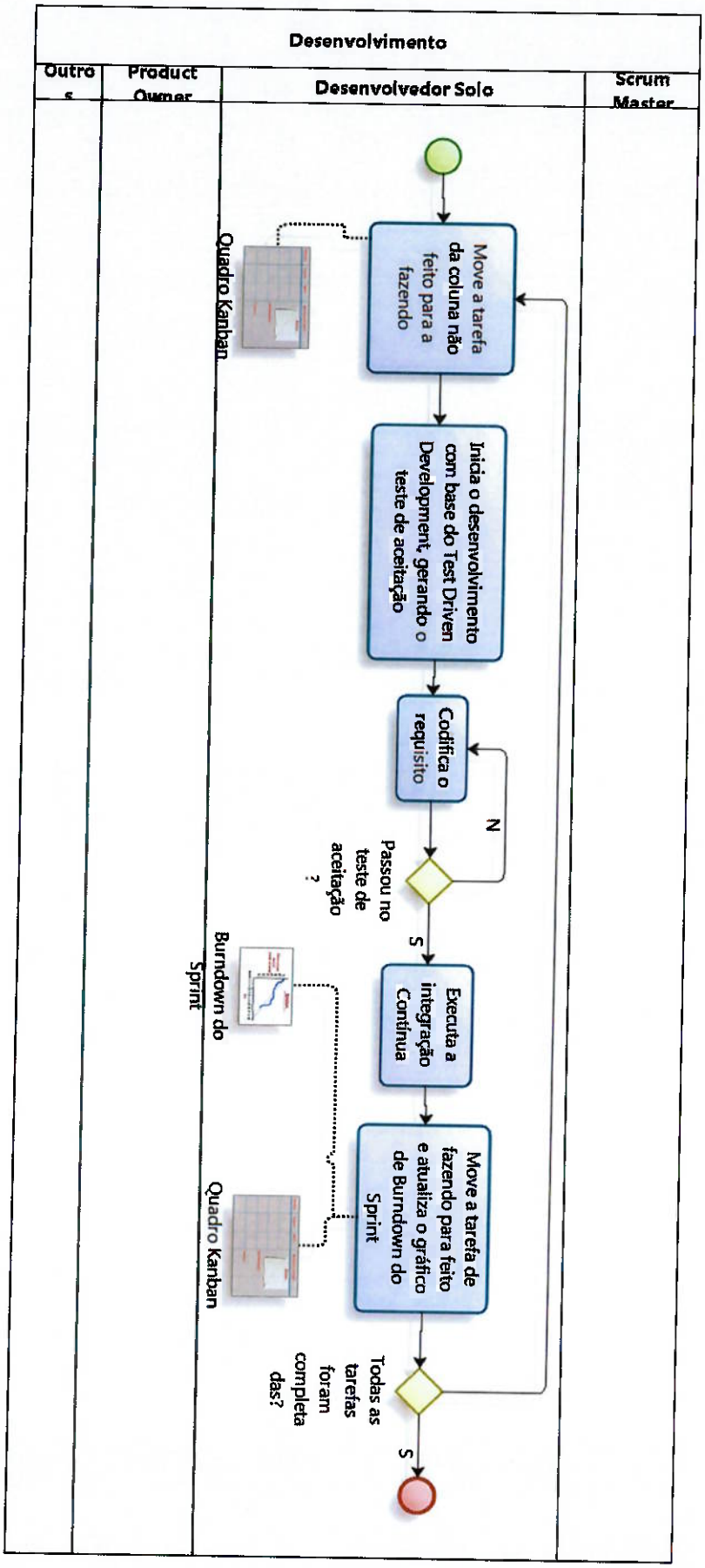

Figura 8: BPMN do Desenvolvimento

3.5.3.3 ***Revisão do Sprint***

Após o desenvolvimento é feita uma reunião para revisão do Sprint, a função é apresentar o trabalho feito. Este é o momento em que o desenvolvedor solo terá o retorno do cliente sobre o trabalho feito, e por ser o único responsável pela criação, todo e qualquer comentário, elogio ou critica será direcionado a ele e deverá aproveitá-las para seu crescimento profissional.

Nesta reunião o quadro Kanban deve estar presente e através dele deve-se demonstrar como as tarefas foram executadas, se houve algum erro na estimativa, alguma tarefa não prevista ou se a velocidade planejada foi atingida e devem ser apresentados os requisitos desenvolvidos e após, realiza-se uma apresentação de cada característica desenvolvida, tomando por base os testes de aceitação, mostrando que de fato o requisito está pronto.

O Scrum Master deve garantir que a apresentação seja preparada pelo desenvolvedor. Isto criará maior compromisso por parte do Desenvolvedor Solo.

Abaixo o BPMN da Revisão do Sprint.

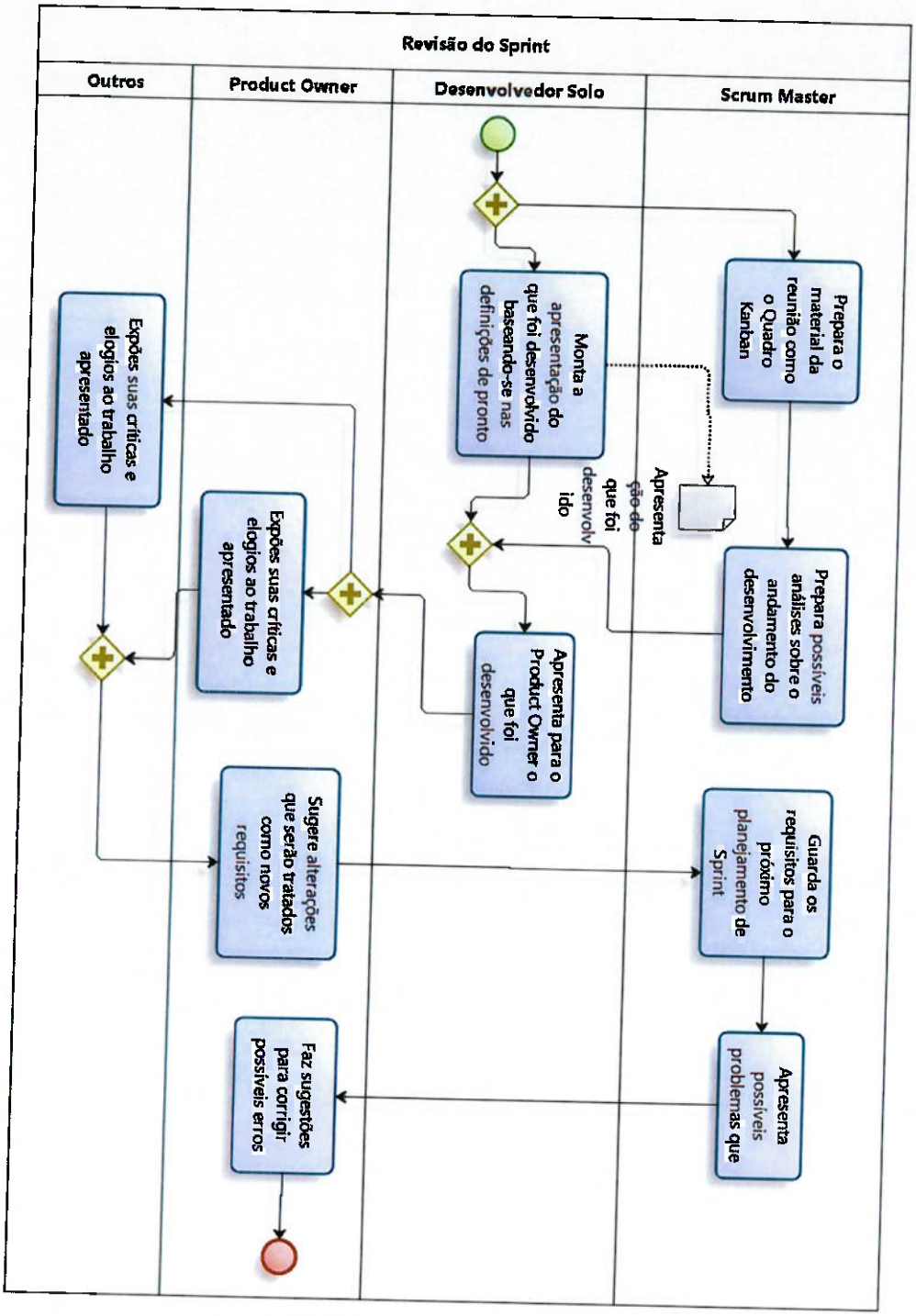

Figura 9: BPMN da Revisão do Sprint

3.5.3.4 Retrospectiva do Sprint

A retrospectiva do Sprint é uma reunião entre Scrum Master e Desenvolvedor Solo tem a função de auxiliar a melhoria do processo de desenvolvimento. O desenvolvedor deve ter ciência dos pontos que funcionaram e pontos que podem ser melhorados, sejam eles tempo de Sprint, ferramentas ou técnicas de programação.

A retrospectiva é o momento para sugerir aumento ou diminuição do período de Sprint, para modificações no quadro Kanban, sugerir ferramentas diferentes, ou qualquer melhoria no processo.

3.5.4 Intervalo entre Sprints

Este evento tem como função aplicar um intervalo entre os Sprints e apesar do Scrum Guide (SCHWABER, K.; SUTHERLAND, J., 2011) não incluir intervalos entre Sprints realizar Sprint após Sprint pode ser extremamente cansativo ao desenvolvedor solo.

Após a reunião de retrospectiva do Sprint o próximo evento é o planejamento do próximo Sprint que para um Sprint de três semanas ocuparia seis horas de planejamento de acordo com SCHWABER, K.; SUTHERLAND, J. (p12-14 2011), o que somada à reunião de retrospectiva, completaria praticamente um dia inteiro de reunião.

Além do tempo excessivo gasto com estas reuniões, as pessoas envolvidas não teriam tempo suficiente para assimilar todas as informações apresentadas pós Sprint.

O intervalo de pelo menos um dia entre um Sprint e o próximo será tratado como dia de laboratório, sugestão dada por KNIBERG, H. (p75-77 2007), inspirado pelo Google e tratado como um benefício para contratação de desenvolvedores.

Este intervalo tem o custo de um dia de desenvolvimento, porém pode-se gerar benefícios como melhor análise do Product Owner sobre o que foi feito e o que deverá ser feito no próximo Sprint.

KNIBERG, H. (p75-77 2007) indica que seja feita uma apresentação sobre o tópico estudado entre os desenvolvedores solos, assim garante-se que a realização da pesquisa e compartilha-se o novo conhecimento adquirido, permitindo valiosas trocas de conhecimento.

O intervalo possibilita também a busca por novas técnicas para melhorias fora do contexto do Sprint, assim as estimativas podem ser mais realistas do que se estas técnicas tivessem que ser assimiladas durante o Sprint e não tivessem sido identificadas durante o planejamento.

3.6 O Arcabouço SOLO

Abaixo segue o BPMN do arcabouço adaptado, o Processo de desenvolvimento Solo está descrito em atores, artefatos, eventos e práticas de engenharia, descritos anteriormente de acordo com a metodologia solo.

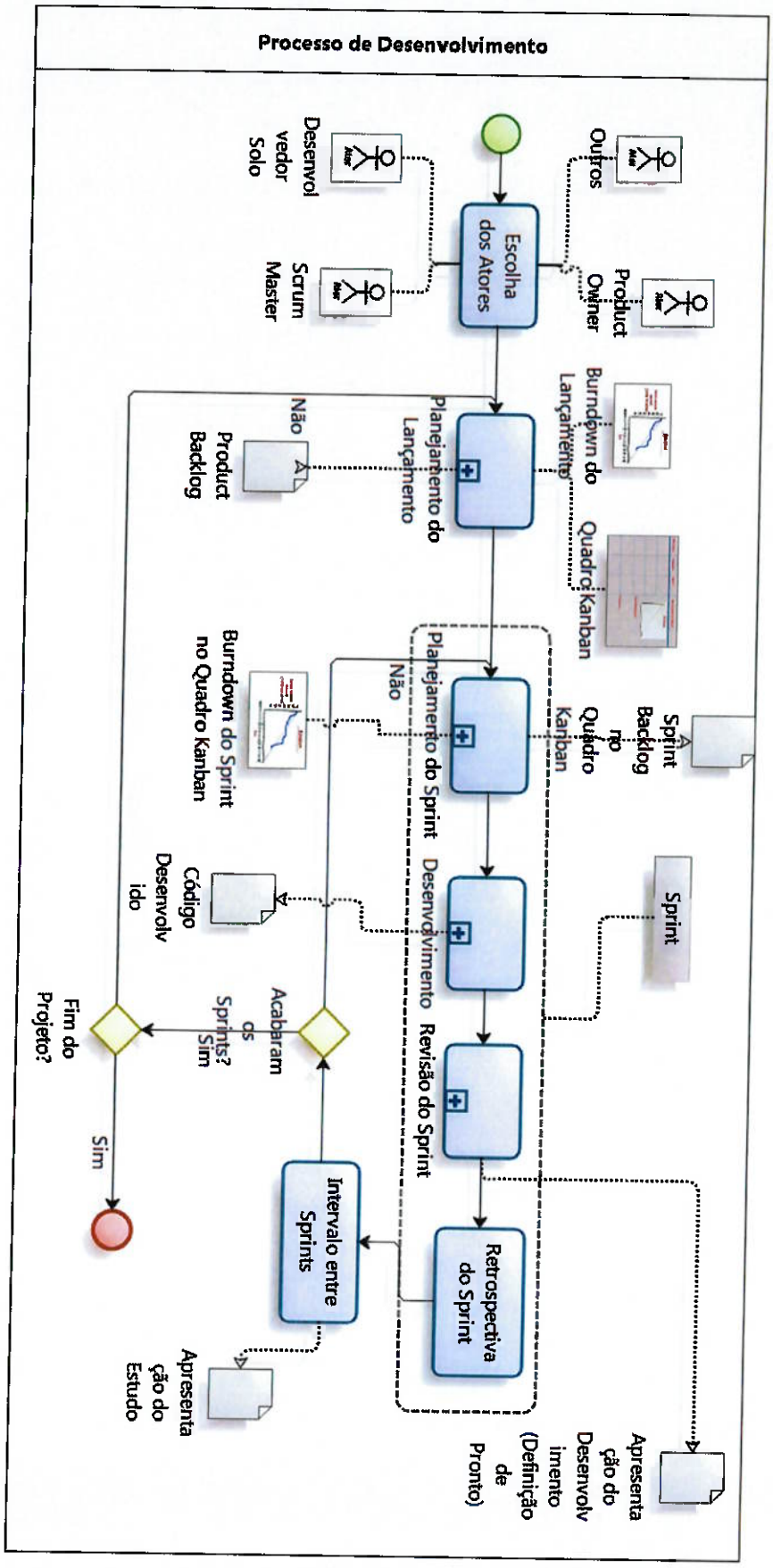

Figura 10: Solo

O arcabouço SOLO moldado nos três arcabouços supracitados possui um número de regras entre o Scrum e o XP, juntando as práticas gerenciais e técnicas para manter controle sobre o desenvolvimento sem deixar o processo lento. A metodologia auxilia equipes a adotar práticas reconhecidas mesmo contando com um único desenvolvedor por projeto, mantendo os valores de projetos ágeis.

Através da tabela abaixo é possível perceber a estrutura do Scrum com apenas algumas adições de práticas de engenharia provenientes do XP e do Kanban. O resultado é um arcabouço que manteve o foco gerencial do Scrum e Kanban, mas que contém processos para auxiliar o desenvolvimento e facilitar a manutenção e melhoria do produto.

Metodologia	Tipo de Prática	Nome	Regra
Solo	Ator	Scrum Master	Gerencial
	Ator	Product Owner	Gerencial
	Ator	Desenvolvedor Solo	Gerencial
	Ator	Outros	Gerencial
	Eventos	Reunião de Planejamento do Lançamento	Gerencial
	Eventos	Sprint	Gerencial
	Eventos	Reunião de Planejamento da Sprint	Gerencial
	Eventos	Revisão da Sprint	Gerencial
	Eventos	Retrospectiva da Sprint	Gerencial
	Eventos	Intervalo entre Sprints (Passo Sustentável)	Técnica
	Artefatos	Product Backlog (Testes de Aceitação)	Gerencial e Técnica
	Artefatos	Burndown do Lançamento	Gerencial
	Artefatos	Quadro Kanban (Sprint Backlog + Sprint Burndown)	Gerencial
	Práticas de Engenharia	Crie um fluxo de trabalho	Gerencial
	Práticas de Engenharia	Limitar os trabalhos executados por	Gerencial

		estado do processo	
	Práticas de Engenharia	Acompanhar o tempo de execução das tarefas	Gerencial
	Práticas de Engenharia	Test Driven Development	Técnica
	Práticas de Engenharia	Integração Contínua	Técnica

Tabela 4: Regras do Arcabouço Solo

No total o arcabouço SOLO possui dezoito regras, quatro a mais que o Scrum e duas a menos que o XP. Ele foi adaptado para possuir os mesmo controles gerenciais do Scrum com algumas técnicas de desenvolvimento do XP para dar mais segurança ao desenvolvedor e as técnicas gerenciais do Kanban para melhor controle do fluxo de tarefas.

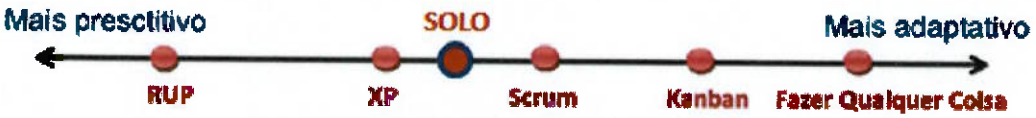

Figura 11: Métrica prescritiva adaptativa SOLO

4 IMPLEMENTAÇÃO

4.1 Cenário Proposto

O cenário proposto irá demonstrar a melhoria no processo de desenvolvimento com a implantação do arcabouço SOLO em uma pequena empresa especializada em gestão de facilidades, que possui mais de cento e cinquenta funcionários e necessita de sistemas de informação para as áreas internas de manutenção, operação de utilidades e segurança, cujo nome será mantido em sigilo por questões de confidencialidade.

Este cenário é baseado em problemas reais, porém não representa exatamente um conjunto de atividades ou projeto real desta empresa.

A gestão de facilidades é descrita pelo IFMA (International Facilities Management Association) em IFMA (2012) como um integrador de pessoas, lugares, processos e tecnologias com a característica de ser multidisciplinar abordando diversos setores de um empreendimento.

No cenário em questão serão abordados os três principais serviços fornecidos pela empresa, operação de utilidades, operação de segurança e manutenção.

A operação de utilidades atua sobre os sistemas de automação de ar condicionado, preocupando-se primeiramente com o conforto dos condôminos e em um segundo plano em diminuir custos através do melhor controle sobre o sistema.

A operação de segurança engloba os sistemas de vídeo e alarme de intrusão. O propósito é captar situações de risco para o empreendimento e seguir o fluxo de trabalho necessário para cada situação.

A manutenção é o principal braço de apoio de segurança e utilidades, pois é quem mantém os sistemas em funcionamento.

Com a ampla atuação da gestão de facilidades, a área de TI tem ganhado força, pois softwares podem reduzir custos mantendo e

automatizando processos e oferecendo serviços diferenciados dentro de cada segmento, seja de utilidades, manutenção ou segurança.

No cenário proposto trabalham na empresa, dois desenvolvedores, um engenheiro de software e o diretor geral da empresa que representa as áreas de negócio concentrando as decisões de escopo e prioridades dos projetos.

Sem uma metodologia de desenvolvimento, as melhorias nos sistemas são feitas por demanda e novos projetos demoram a agregar valor ao negócio, uma vez que a primeira entrega acontece somente com o software funcional e pronto.

4.2 A Escolha do Arcabouço

A necessidade de entregas rápidas e equipes pequenas são cenários comuns às metodologias ágeis, tanto o Scrum quanto o XP resolveriam o proposto.

Porém o Scrum demanda muita disciplina dos programadores, já que não especifica regras de desenvolvimento, já o XP demandaria muita disciplina do gestor da área, pois apesar de especificar um processo, ainda que pequeno, não fornece ferramentas para identificar possíveis problemas e gargalos no processo como fazem os gráficos do Scrum e o quadro do Kanban.

Ainda deve-se levar em consideração um excesso de projetos de diferentes áreas de forma que para não deixar nenhuma área em deficiência atribui-se a cada desenvolvedor um projeto.

Temos então um cenário com dois desenvolvedores solos, ou seja, atuando isoladamente no desenvolvimento do produto de forma ágil, aderindo assim ao conceito do arcabouço solo.

4.3 O Projeto

Será definido como escopo desta implementação o projeto para a operação de segurança, cujo seu objetivo é aumentar a eficiência dos sistemas já existentes de captação de vídeo e alarmes de intrusão.

O sistema de captação de vídeo possui uma lista de alarme muito grande e por isso não é possível utilizá-la devendo-se então captar de forma filtrada estes alarmes.

Os alarmes de intrusão devem se integrar aos alarmes de vídeo para que exista somente um único software de alarmes.

4.4 A Escolha dos Atores

O ator de Scrum Master será função do engenheiro de computadores, que deverá atuar diretamente na gerencia dos projetos, agendar as reuniões, ajudar os desenvolvedores e controlar o processo.

O PO (Product Owner) será função do diretor, que no cenário proposto possui conhecimento sobre o negócio e tem poder para priorizar os requisitos.

O Desenvolvedor Solo ficará sendo a função dos programadores, cada um em seu projeto, responsáveis por toda a parte técnica.

Qualquer outro interessado poderá participar fornecendo requisitos que serão analisados pelo PO.

4.5 O Inicio do Processo – Planejamento do Lançamento

É função do Scrum Master, agendar e iniciar a reunião com o PO (Product Owner) e o Desenvolvedor Solo. A reunião inicia-se com o PO explicando o que motiva o projeto e qual o objetivo a ser atingido.

A motivação e o objetivo não são obrigatórios para o framework, porém podem sanar alguma futura dúvida.

O motivo do projeto é a dependência humana para captação de falhas de segurança nos empreendimentos.

O objetivo é criar um sistema de captação de alarmes provenientes do sistema de vídeo dos empreendimentos e tratá-los de forma unificada com os alarmes de intrusão.

A partir daí o PO inicia a descrição de todos os requisitos do projeto e faz-se uma lista que recebe o nome de Product Backlog.

4.5.1 Identificação do Product Backlog

A partir da conversa, forma-se uma tabela contendo o requisito, a estimativa para entrega, a definição de pronto, uma prioridade e algumas observações que o desenvolvedor identificar necessário.

- O software deve captar do sistema de vídeo as seguintes situações: Falha de Gravação e Perda de Vídeo
- O software deve gerenciar sua conexão com a central
- O software deve integrar o sistema de recepção de alarmes de intrusão
- Cada evento captado pelo software deve ser tratado de acordo com um fluxo de trabalho específico
- O software terá três diferentes níveis de acesso: Administrativo, Supervisor e Operador

A partir daí ambos Scrum Master e Desenvolvedor Solo devem entender o que são cada requisito questionando sempre o que for necessário.

Requisito - O software deve captar do sistema de vídeo as seguintes situações: Falha de Gravação e Perda de Vídeo	
Exemplo de Questões	**Exemplo de Resposta**
O sistema de vídeo armazena algum indício destas falhas? Como?	Para "Falha de Gravação" não, porém para "Perda de Vídeo" sim é gerado algum tipo de arquivo que contém.
Quando se dá cada uma delas "Falha de Gravação" e "Perda de Vídeo"?	O primeiro deve ocorrer se não houver gravações na última hora, o segundo se o sistema perder o sinal de vídeo da câmera.

Qual é a arquitetura do Sistema de Vídeo? Como funciona?	Há um computador no cliente com uma placa de captura de vídeo e as imagens trafegam para central via rede TCP/IP.
Será possível acesso ao Sistema de Vídeo?	Sim, é possível liberar um de teste.
Requisito - O software deve gerenciar sua conexão com a central	
Exemplo de Questões	**Exemplo de Resposta**
O sistema de vídeo armazena algum indício desta falhas? Como?	Sim, porém se a conexão tiver caído seria impossível obter tal informação na hora.
Trata-se de uma conexão TCP/IP?	Sim.
Atualmente como é percebido este tipo de problema?	A reprodução da imagem na central para.
Requisito - O software deve integrar o sistema de recepção de alarmes de intrusão	
Exemplo de Questões	**Exemplo de Resposta**
Já existe um software para recepção dos alarmes de intrusão?	Sim, ele recebe alarmes via linha telefônica ou via rede TCP/IP.
O Acesso aos dados dele é liberado?	Sim, temos usuário e senha da base de dados.
Requisito - Cada evento captado pelo software deve ser tratado de acordo com um fluxo de trabalho específico	
Exemplo de Questões	**Exemplo de Resposta**
Este fluxo já existe?	Sim, temos o fluxo documentado.
Quem participa do fluxo?	O fluxo de atendimento é basicamente feito por um operador. O que muda são as tratativas para cada evento capturado.
As tratativas devem ser configuráveis?	Sim, o supervisor deve ter acesso à configuração.
Requisito - O software deverá ter três diferentes níveis de acesso: Administrativo, Supervisor e Operador	
Exemplo de Questões	**Exemplo de Resposta**
Poderão existir mais níveis futuramente?	Não.

Tabela 5: Questionamento sobre os requisitos

Após análise dos requisitos, monta-se uma tabela identificando os demais pontos, como a estimativa a ser feita em dias, a prioridade que deve ser um número diretamente proporcional à prioridade e a definição de pronto. Não faz parte de esta proposta estudar metodologias para estimar, dar prioridade e identificar a definição de pronto. Ao montarmos o documento podemos utilizar de observações às questões e respostas levantadas na tabela 5.

Requisito	Estimativa	Prioridade	Definição de Pronto	Observações

O software deve captar do sistema de vídeo as seguintes situações: Falha de Gravação e Perda de Vídeo	8	2	Quando o software captar tais eventos do sistema de vídeo.	
O software deve gerenciar sua conexão com a central	8	1	Quando o software for capaz de identificar a desconexão com o sistema de vídeo.	
O software deve integrar o sistema de recepção de alarmes de intrusão	15	4	Quando for possível trocar a operação de um software pelo outro.	
Cada evento captado pelo software deve ser tratado de acordo com um fluxo de trabalho específico	12	3	Quando dado um evento, forçar a tratativa de acordo com o fluxograma.	
O software terá três diferentes níveis de acesso: Administrativo, Supervisor e Operador	8	5	Quando provar-se que cada um tem acesso a um conjunto de itens diferentes.	

Tabela 6: Product Backlog

4.5.2 *Montagem do Burndown do Lançamento*

Com o Product Backlog monta-se o gráfico de Burndown do Lançamento conforme abaixo.

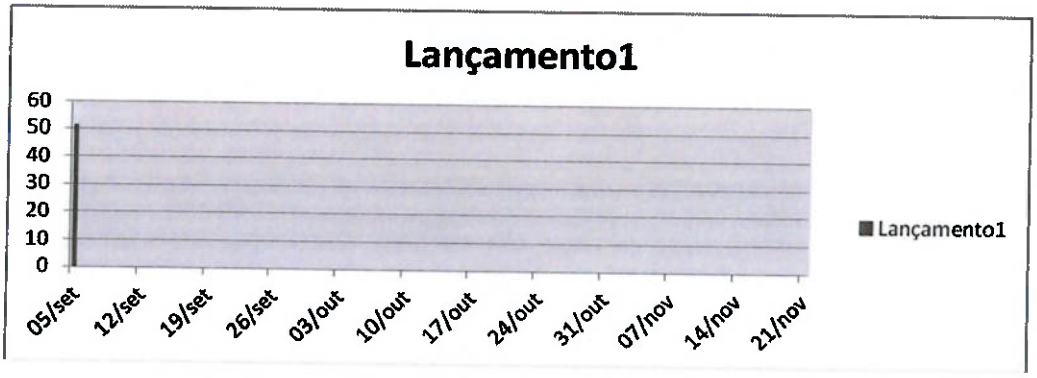

Gráfico 2: Burndown do Lançamento

O gráfico inicialmente possui o total de dias estimados para a conclusão de todos os itens do Product Backlog e com o decorrer do projeto, a cada semana, atualiza-se o gráfico. No decorrer do projeto ele deverá ficar como exibido abaixo.

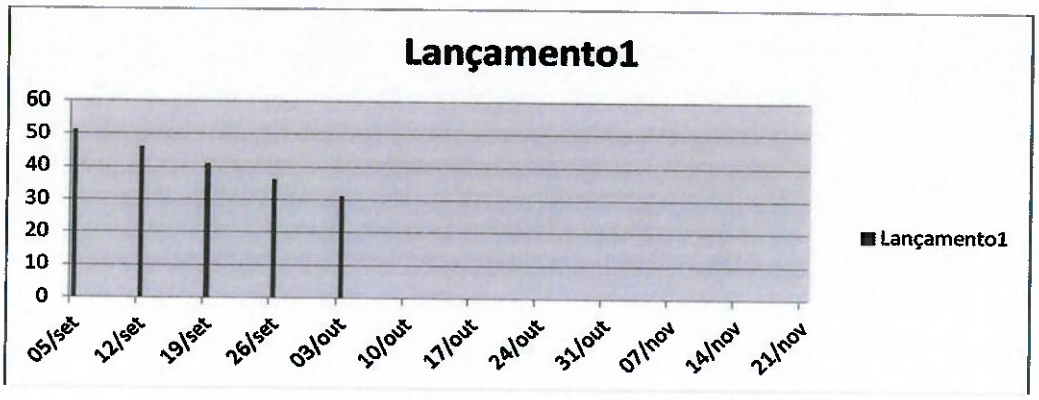

Gráfico 3: Burndown do Lançamento Atualizado

Percebe-se então que podemos traçar uma reta de tendência indo a zero. Com isso podemos atualizar a expectativa de conclusão e entrega do lançamento. A reta de tendência deve ser aproximadamente igual à reta formada após conclusão do lançamento.

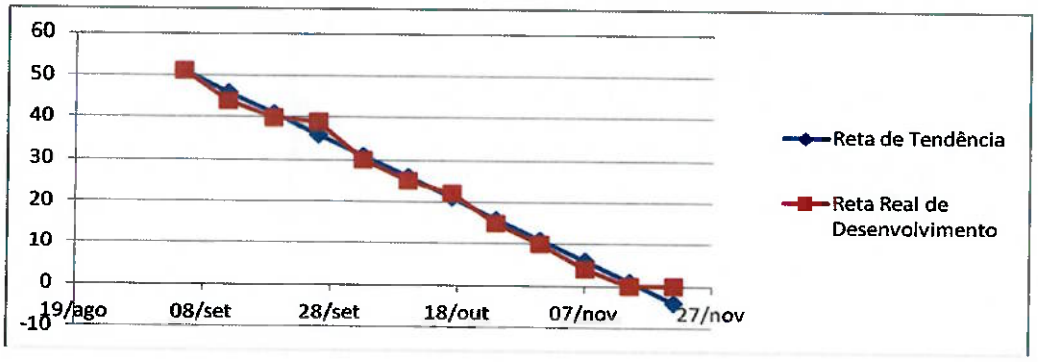

Gráfico 4: Burndown Real x Teórico

4.6 A Criação do Quadro Kanban

Para a montagem do quadro definem-se os estados possíveis de cada trabalho. Inicialmente serão definidos apenas os estados considerados básicos, "Não Feito", "Fazendo" e "Feito".

Nos estados citados o único participante é o Desenvolvedor Solo, e o limite de fluxo de trabalho é um para "Fazendo" e "Feito", e não há limites para "Não Feito".

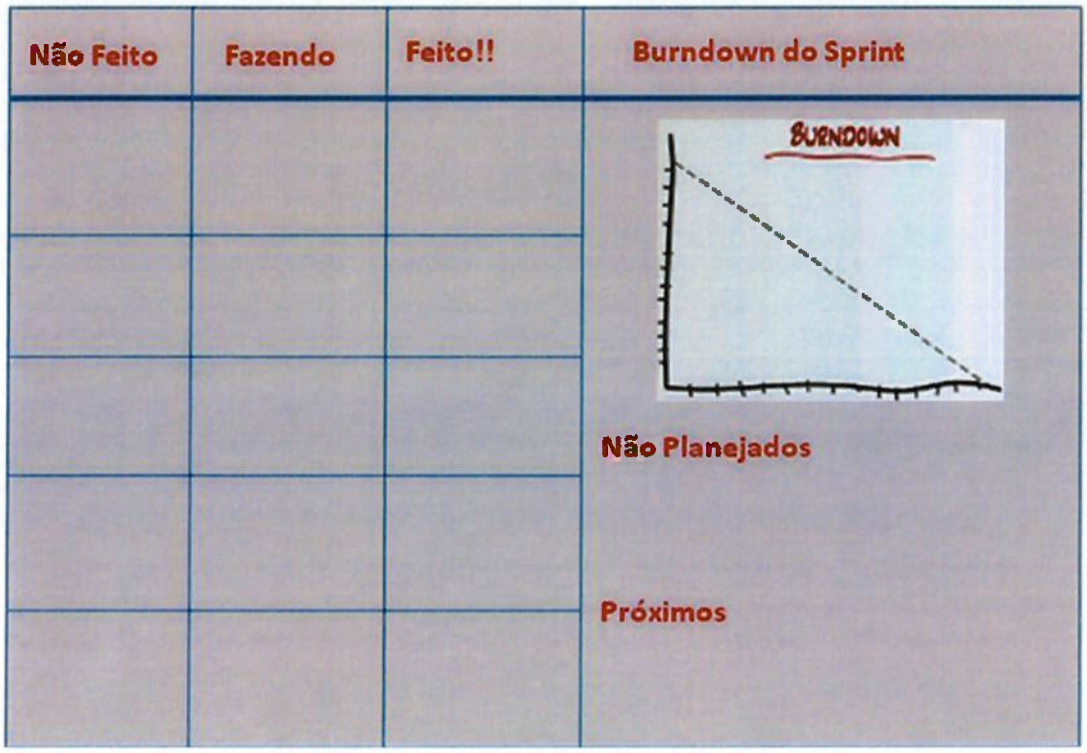

Figura 12: Montagem do Quadro Kanban

O quadro deve ser atualizado diariamente durante o Sprint. Cada tarefa aceita pelo desenvolvedor deve ser retirada de "Não Feita" para "Fazendo" e assim que estiver concluída, deve ser colocada sobre a coluna "Feita". Desta forma o quadro é mantido sempre atualizado.

4.7 O Sprint

4.7.1 O Planejamento do Sprint

Com o projeto já iniciado e com os artefatos Product Backlog, Burndown do Lançamento e Quadro Kanban já criados, o Scrum Master deverá garantir a agenda dos envolvidos para o dia inteiro.

Com o Product Backlog em mãos o Scrum Master sugere o tamanho do Sprint baseando-se no tamanho de cada requisito. O tamanho é dado em semanas, lembrando que a semana possui cinco dias de trabalho e a estimativa é dada em dia x homem.

Ordenado por prioridade, temos os dois primeiros itens somando 16 dias x homem, ou seja, este é um trabalho de dezesseis dias para um homem e será o primeiro Sprint. O segundo Sprint será o item com doze dias x homem estimados, o terceiro Sprint será o quarto item da prioridade, sobrando o ultimo item para um quarto Sprint.

	Sprint	Estimativa	Inicio em	Requisito
Lançamento1	Sprint1	51	05/set	1) O software deve gerenciar sua conexão com a central 2)O software deve captar do sistema de vídeo as seguintes situações: Falha de Gravação e Perda de Vídeo
		46	12/set	
		41	19/set	
	Sprint2	36	26/set	Cada evento captado pelo software deve ser tratado de acordo com um fluxo de trabalho específico
		31	03/out	
		26	10/out	
	Sprint3	21	17/out	O software deve integrar o sistema de recepção de alarmes de intrusão
		16	24/out	
		11	31/out	
	Sprint4	6	07/nov	O software terá três diferentes níveis de acesso: Administrativo, Supervisor e Operador
		1	14/nov	
		-4	21/nov	

Tabela 7: Os Sprints

Os Sprints deverão sofrer alterações no decorrer do projeto, este é um dos conceitos que tornam as metodologias ágeis tão flexíveis, as mudanças são previstas no processo.

Definidos os requisitos do primeiro Sprint, o Desenvolvedor Solo deve então listar as tarefas necessárias para a construção de cada requisito. Cada tarefa deverá durar um dia preferencialmente, isto torna o controle da gestão mais simples.

O software deve captar do sistema de vídeo as seguintes situações: Falha de Gravação e Perda de Vídeo	Estudar o sistema de vídeo e identificar onde são armazenados os eventos.
	Estudar onde ficam as gravações e montar a lógica para captar falhas de gravação.
	Montar a arquitetura para a captura dos eventos.
	Modelar as Classes do sistema.
	Modelar as Tabelas da base de dados.
	Montar os testes de aceitação para a captura dos eventos.
	Desenvolver a captura dos eventos.

	Terminar o desenvolvimento e verificar o desempenho.
O software deve gerenciar sua conexão com a central	Estudar a conexão.
	Montar uma arquitetura estilo "Keep Alive"
	Modelar as Classes do sistema.
	Modelar as Tabelas da base de dados.
	Montar os testes de aceitação para o gerenciamento da conexão.
	Montar um teste de arquitetura para certificar a conexão simultânea de todos.
	Desenvolver.
	Desenvolver.

Tabela 8: Criação das Tarefas

A tabela acima possui para cada requisito um conjunto de oito tarefas estimadas em um dia de trabalho. Com isto pode-se atualizar o quadro Kanban onde para cada tarefa coloca-se uma nota adesiva com o nome da tarefa e caso a estimativa seja diferente de um dia, coloca-se também a estimativa.

As notas são dispostas no quadro na linha de seu requisito, e devem seguir sempre esta linha. A movimentação é somente entre as colunas. Sendo que foi estipulada uma única tarefa por coluna, uma vez que o Desenvolvedor Solo só poderá fazer uma tarefa por vez.

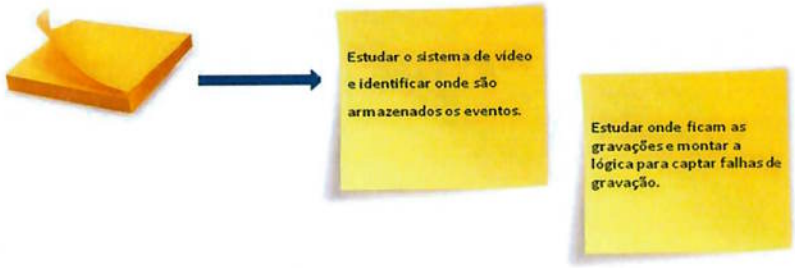

Figura 13: As Tarefas

Também no quadro é desenhado o gráfico de Burndown do Sprint. Com ele o Scrum Master deve acompanhar o andamento de cada tarefa, como, por exemplo, se a tarefa permaneceu mais de um dia parada na coluna "Fazendo" é sinal de que há algum impeditivo que deve ser investigado e resolvido.

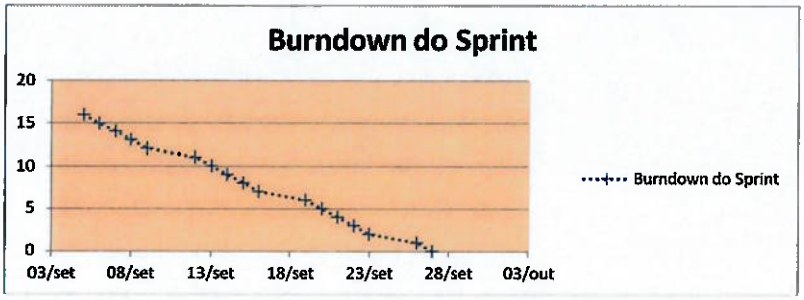

Gráfico 5: Burndown Estimado do Sprint

O gráfico do Burndown do Sprint é desenhado na parte superior direita do quadro.

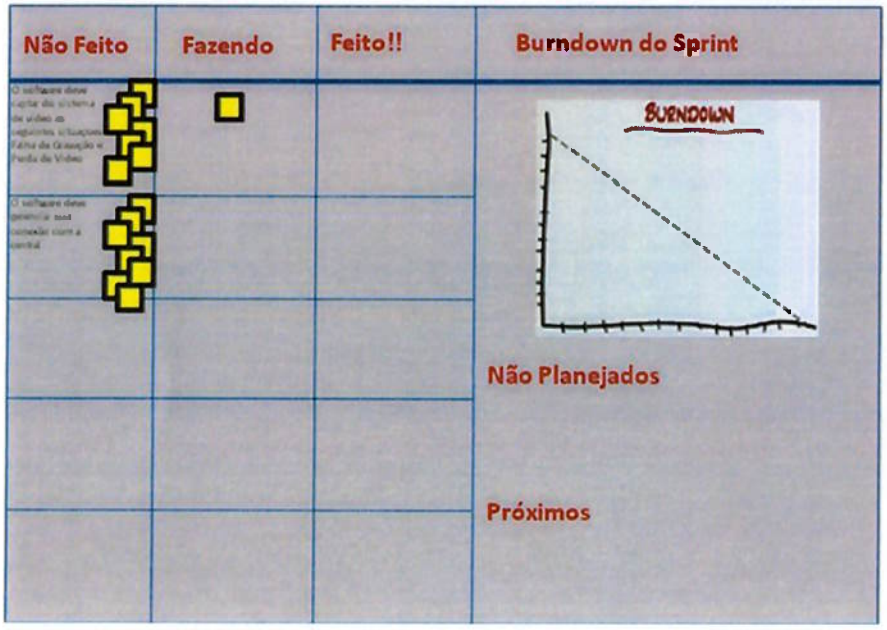

Gráfico 6: Quadro Kanban Inicial

4.7.2 O Desenvolvimento

Para o inicio do desenvolvimento o Desenvolvedor Solo move uma nota de tarefa para a coluna "Fazendo".

Caso a tarefa esteja relacionada com a codificação do software inicia-se pelo teste automatizado de aceitação.

No caso do primeiro requisito temos as cinco primeiras tarefas não relacionadas à codificação, somente as demais. Assim a partir da sexta faz-se uso do Test Driven Development.

A cada requisito desenvolvido, uma versão nova deve ser construída e publicada no ambiente de homologação e assim o Scrum Master deve divulgar ao Product Owner.

A etapa de desenvolvimento é finalizada quando todas as tarefas forem cumpridas e os requisitos passarem pelos testes de aceitação

4.7.3 A Revisão do Sprint

Antes de convocar uma reunião com o Product Owner, o Desenvolvedor Solo deve gerar uma apresentação, ainda que simples, para demonstrar o trabalho feito. A tabela abaixo descreve o mínimo que a apresentação deste Sprint deve conter, sendo que para cada requisito entregue a apresentação deve demonstrar que o teste de aceitação foi cumprido.

Requisito	Teste de Aceitação	Apresentação
1) O software deve gerenciar sua conexão com a central	Quando o software for capaz de identificar a desconexão com o sistema de vídeo.	Na apresentação leva-se o sistema de vídeo teste utilizado no desenvolvimento, de modo que seja possível simular a desconexão exibindo no software a desconexão.
2)O software deve captar do sistema de vídeo as seguintes situações: Falha de Gravação e Perda de Vídeo	Quando o software captar tais eventos do sistema de vídeo.	Aproveita-se o sistema de vídeo para exibir a captura dos eventos. Mostra-se a lista de eventos no sistema de vídeo com os eventos desejados e também a base de dados vazia. Após execução do software a base de dados deve contar os mesmos eventos, demonstrando assim o funcionamento dele.

Tabela 9: Apresentação da Revisão do Sprint

Feita a apresentação, o Product Owner deve comentar se a entrega está alinhada com a expectativa.

Após os comentários, utiliza-se o quadro Kanban para análise da execução das tarefas, devem-se levantar quais demoraram mais do que o esperado e por que, assim é possível alguma ação para que o erro não ocorra mais.

Inicialmente observa-se o gráfico de Burndown do Sprint, o tracejado representa a linha estimada. Ao fim de todo dia o Scrum Master marca o ponto real que representa a estimativa atualizada para o final do Sprint. Assim a reta contínua representa a realidade.

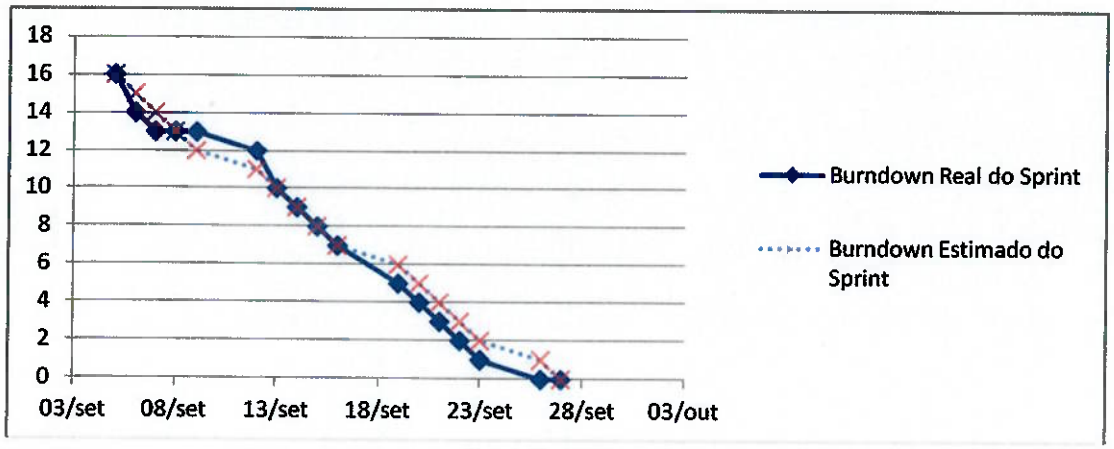

Gráfico 7: Gráfico Burndown Real x Estimado

No exemplo acima é possível observar que o gráfico estagnou em três dias, entre os dias sete e nove de setembro nenhuma tarefa foi concluída de modo que esta tarefa que deveria ter demorado apenas um dia.

Com esta observação feita, identifica-se no Backlog do Sprint a terceira tarefa em ordem de prioridade. Montar a arquitetura para a captura dos eventos.

Questiona-se o Desenvolvedor Solo sobre o motivo deste erro de estimativa, mesmo que no geral a entrega tenha sido desenvolvida antes do prazo.

Esta tarefa teria representado um erro de estimativa, sendo que uma possível resposta do desenvolvedor poderia ser uma mudança de arquitetura não planejada.

4.7.4 Retrospectiva do Sprint

Terminada a revisão do Sprint, Scrum Master e Desenvolvedor Solo têm uma conversa rápida para sugestões de melhoria no processo de desenvolvimento.

O Scrum Master poderia sugerir que fosse notificado sempre que alguma mudança na arquitetura no desenvolvimento fosse necessária ou ainda o Desenvolvedor Solo poderia sugerir um software para integração contínua e controle de versão dos arquivos.

Em ambos os casos são sugestões de melhoria no processo de desenvolvimento.

4.7.5 Intervalo entre Sprints

O intervalo deve servir de laboratório para novas técnicas, o Scrum Master deve sugerir alguns tópicos que identificar como possíveis soluções para dificuldades técnicas encontradas nos desenvolvimentos passados como:

- Novas arquiteturas Cliente – Servidor
- Push Server
- Enterprise Java Beans
- Padrões de Desenvolvimento

Algumas técnicas podem ser aplicadas durante futuros Sprints não ocasionando talvez, erros de estimativas. No cenário proposto foi citado um possível erro na escolha da arquitetura que poderia não ocorrer caso houvesse um estudo prévio sobre novas arquiteturas cliente – servidor.

O Desenvolvedor Solo tem neste dia a oportunidade de trabalhar em um ritmo reduzido uma vez que não há cobranças sobre o que é explorado no intervalo.

5 CONCLUSÃO

Com o crescimento do mercado brasileiro, a demanda de projetos na área de TI tem crescido, porém a qualidade da mão de obra não consegue acompanhar o ritmo.

Conforme constatado por SEPRORJ (2011) cursos de tecnologia são muito caros e as faculdades não oferecem profundidade suficiente para que um formando tenha capacidade de ingressar no mercado de trabalho então com este cenário, acarreta em pouca mão de obra em TI no Brasil o que ocasiona também em um elevado custo para contratar a pouca mão de obra qualificada existente.

O cenário do Desenvolvedor Solo oferece uma solução para empresas que não tem orçamento para possuir grandes equipes de desenvolvimento e necessitam desenvolver vários projetos em um curto espaço de tempo, uma vez que reduz o tamanho da equipe ao máximo, mesmo que no fim seja mantido apenas aquele de mais experiência e, portanto de maior custo.

O arcabouço também mantém um baixo nível de acoplamento entre a empresa e o desenvolvedor, ou seja, o conjunto de artefatos gerados facilita uma possível mudança de desenvolvedor no projeto, pois deixa rastros claros do que foi desenvolvido, problemas resolvidos e mudanças realizadas.

Originado no Scrum, a necessidade de desenvolvedores multidisciplinares evita a segmentação de obrigações técnicas, o que pode ser visto como perda de qualidade, porém com a aplicação do TDD (desenvolvimento dirigido a testes) torna-se inviável separar o desenvolvimento do teste uma vez que se inicia o desenvolvimento com o teste.

Outro ponto positivo deste acúmulo de responsabilidades é tornar o desenvolvedor mais preocupado, pois é o único responsável técnico, sendo que nunca será preocupação de outro a integração com códigos já existentes ou com desempenho.

Esta visão tem como principio o cenário proposto, ou seja, empresas com equipes reduzidas, não sendo aplicável para equipes de centenas de

desenvolvedores em projetos extremamente complexos e demorados. Nestes casos existem arcabouços mais adequados, onde a burocracia e o excesso de regras tornam-se necessários para que tenha controle sobre o processo.

Outra observação que deve ser feita em relação a estes diferentes cenários, é sobre extensas documentações que para as metodologias ágeis aplicadas em pequenas equipes serão geradas somente se forem um requisito para o cliente, então se entende que o documento será mais útil do que outro requisito, assim não se faz documentos desnecessários ao cliente. Porém para grandes equipes e grandes projetos existem documentos essenciais para o controle do projeto.

Estes dois cenários de dimensões diferentes são fundamentais para a escolha da metodologia, os arcabouços ágeis são ideais para pequenas equipes, já metodologias como o UP (Unified Process) comportam um número muito maior de pessoas por ter um processo muito mais controlado e burocrático.

5.1 Resultados Obtidos

Baseado no Scrum, a técnica de gestão do projeto do arcabouço solo é simples e eficaz, dá confiança ao desenvolvedor e comprometimento com as estimativas. Os interessados no projeto têm visão clara do andamento do projeto e se houver qualquer problema de estimativa logo se identifica o motivo, resolvendo o problema sem que a culpa seja do desenvolvedor.

Esta visão do andamento do projeto também é responsabilidade do quadro Kanban adotado, o Kanban fornece um artefato de fácil entendimento para o controle das tarefas do Sprint do projeto.

O XP contribui com algumas técnicas de desenvolvimento para que haja alguma regra de desenvolvimento. Não existir regra alguma faz com que não haja padrão e dificulta estimativas.

As regras adotadas, porém, tem objetivos considerados essenciais que dão maior segurança ao programador e maior visibilidade ao trabalho feito. Uma vez implementado, o arcabouço permite que o conhecimento sobre o

projeto esteja claro e documentado através dos artefatos gerados durante os ciclos.

Este trabalho apresenta um arcabouço de desenvolvimento ágil baseado no Scrum, XP e Kanban para solucionar a escassez de desenvolvedores na empresa, bastando somente um desenvolvedor, um engenheiro de software e um responsável da área de negócio.

Além manter a equipe reduzida, o arcabouço solo auxilia, através de seus artefatos, o dinamismo de desenvolvedores nas empresas, pois deixa rastros claros sobre como o desenvolvimento evoluiu, reduzindo a dependência que o projeto possa ter sobre o desenvolvedor.

O desenvolvedor deve ser multidisciplinar, ou seja, deve conseguir reagir bem à adversidade de qualquer problema, não sendo requisito que seja um especialista técnico.

O engenheiro de software deve ter habilidades técnicas, porém também não necessita ser um especialista. Ele deve ter fácil acesso ao responsável da área de negócio.

Não está prevista participação de consultores de fora da empresa, embora caso necessário, não há restrições quanto a sua participação que pode ocorrer por alguma dificuldade técnica ou por implementação de alguma tecnologia nova.

A soma das metodologias utilizadas como base fornece um arcabouço mais completo tanto para a área de gestão do projeto, quanto para o desenvolvimento.

O arcabouço foi projetado para atender um cenário específico do desenvolvedor solo dentro de uma pequena empresa, nada impede, porém que seja implementado em equipes com mais desenvolvedores ou ainda em desenvolvedores solo autônomos.

Espera-se com a implementação deste framework que seja estabelecido um processo ágil de desenvolvimento que guie o projeto de forma que todos os envolvidos tenham ciência das etapas do projeto e que a estimativa esteja sempre alinhada com a expectativa.

5.2 Recomendações para futuros trabalhos

Para futuros trabalhos recomenda-se a aplicação do arcabouço solo em equipes maiores com mais desenvolvedores e coletar os resultados de uma implantação maior.

Ainda seria interessante aplicar o framework para desenvolvedores autônomos que trabalham solo. Existem muitos profissionais que trabalham em pequenos projetos para sua própria empresa onde é o único funcionário.

Outra recomendação é realizar uma implantação real e adicionar a este trabalho como um estudo de e caso assim seria possível realizar uma revisão das regras adotadas.

Também é recomendado que seja feito um trabalho sobre técnicas e processos para se estimar tempo de desenvolvimento dos requisitos para ser aplicado nas reuniões de planejamento.

Podem-se adicionar ao trabalho possíveis ferramentas eletrônicas para auxiliar o processo de gerenciamento e desenvolvimento.

6 REFERÊNCIAS BIBLIOGRÁFICAS

AGILE Scrum. Proven Practical Tactics For Agile IT Lançamento Management – Lessons Learned. 2010. Disponível em: <http://agileScrum.biz/proven-practical-tactics-for-agile-it-lançamento-management-lessons-learned/>. Acesso em: 03 jan. 2012.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. UML, Guia do Usuário. Rio de Janeiro: Campus, 2000.

FERRER, R. TI no Brasil cresce 10% até 2012,diz Gartner. Info Online, 2011. Disponível em: <http://info.abril.com.br/noticias/ti/ti-no-brasil-cresce-10-em-2011-diz-gartner-26102011-11.shl>. Acesso em: 02 nov. 2011.

HIGHSMITH, J. History: The Agile Manifesto. Agile Manifesto, 2001. Disponível em: <http://agilemanifesto.org/history.html>. Acesso em: 27 dez. 2011.

IFMA, International Facilities Management Association, Disponível em: <http://www.ifma.org/resources/what-is-fm/default.htm> Acesso em: 25 jan. 2012

JEFFRIES, R. E. What is Extremme Programming? 1999. Disponível em: <http://xprogramming.com/what-is-extreme-programming>. Acesso em: 27 dez. 2011.

KNIBERG, H. Scrum e XP direto das trincheiras, como nós fazemos Scrum. EUA: InfoQ, 2007.

KNIBERG, H.; SKARIN, M. Kanban e Scrum obtendo o melhor de ambos. EUA: InfoQ, 2009.

MARTIN, A. M. The Role of Customers in Extreme Programming Projects. Nova Zelândia: Victoria University of Wellington, 2009.

MARTINS, P. G.; LAUGENI, F. P., Administração da Produção. São Paulo: Editora Saraiva, 2006.

OMG Business Process Model Notation (BPMN). Version 2.0, 2011. Disponível em: <http://www.omg.org/spec/BPMN/2.0/PDF/>. Acesso em: 23 mai. 2012.

PRESSMAN, R. S. Engenharia de Software. São Paulo: McGraw-Hill Interamericana do Brasil Ltda., 2005.

REDAÇÃO, D. Métodos ágeis criam nova geração de profissionais. IT Careers, 2011. Disponível em: <http://convergenciadigital.uol.com.br/cgi/cgilua.exe/sys/start.htm?infoid=28182&sid=57>. Acesso em: 27 dez. 2011.

REIS, R. Q.; REIS, C. A. L.; NUNES, D. J. Automação no Gerenciamento do Processo de Engenharia de Software. Belém: Universidade Federal do Pará, 2002.

SAVOINE, M. et al. Análise de Gerenciamento de Projeto de Software Utilizando Metodologia Ágil XP e Scrum: Um Estudo de Caso Prático. Araguaína: Instituto Tocantinense Presidente Antônio Carlos, [entre 2008 e 2011].

SCHWABER, K. et al. Manifesto para Desenvolvimento Ágil de Software. Agile Manifesto, 2001a. Disponível em: <http://agilemanifesto.org/iso/ptbr/>. Acesso em: 27 dez. 2011.

SCHWABER, K. et al. Princípios por trás do Manifesto Ágil. Agile Manifesto, 2001b. Disponível em: <http://agilemanifesto.org/iso/ptbr/principles.html>. Acesso em: 27 dez. 2011.

SCHWABER, K.; SUTHERLAND, J. Scrum tradução: José Eduardo Deboni. Disponível em: <http://www.Scrum.org/storage/Scrum%20Guide%202011%20-%20PTBR.pdf>. Acesso em: 02 nov. 2011

SEPRORJ Falta de profissional capacitado aumenta custo de produção e pode afetar crescimento do setor de TI. Rio Info 2011, 2011. Disponível em: <http://www.rioinfo.com.br/site/index.php/imprensa/1342>. Acesso em: 27 dez. 2011.

VESIONONE, Leading Causes of Failed Agile Projects. State of Agile Survey 2010, p.5, 2010. Disponível em: <http://www.versionone.com/pdf/2010_State_of_Agile_Development_Survey_Results.pdf>. Acesso em: 27 dez. 2011.

WELLS, D. A gentle introduction. Extreme Programming, 2009. Disponível em: <http://www.extremeprogramming.org>. Acesso em: 27 dez. 2011.