IV SINGEP
Simpósio Internacional de Gestão de Projetos, Inovação e Sustentabilidade
International Symposium on Project Management, Innovation and Sustainability
ISSN: 2317 - 8302

UTILIZAÇÃO DE PRÁTICAS ÁGEIS EM GRANDES PROJETOS DE DESENVOLVIMENTO DE SOFTWARE DE UMA INSTITUIÇÃO FINANCEIRA

ROBERTO DE SOUZA GÓES
UNINOVE – Universidade Nove de Julho
roberto.goes@gmail.com

ROSARIA DE FATIMA SEGGER MACRI RUSSO
UNINOVE – Universidade Nove de Julho
rosariarusso@r2dm.com.br

IV SINGEP
Simpósio Internacional de Gestão de Projetos, Inovação e Sustentabilidade
International Symposium on Project Management, Innovation and Sustainability
ISSN: 2317 - 8302

UTILIZAÇÃO DE PRÁTICAS ÁGEIS EM GRANDES PROJETOS DE DESENVOLVIMENTO DE SOFTWARE DE UMA INSTITUIÇÃO FINANCEIRA

Resumo

Este relato técnico tem o objetivo de descrever um projeto piloto executado na fábrica de software de uma grande instituição financeira com o objetivo de criar de um modelo de ciclo de vida de desenvolvimento de software. Este novo modelo deve ser baseado em conceitos e premissas ágeis de modo que tenha a flexibilidade necessária para responder a mudanças e entregar para o cliente funcionalidades que gerem valor o mais breve possível. O modelo também deve ter aderência ao processo de negócio da companhia atendendo a todos os requisitos legais e regras internas, ser aderente a projetos de grande porte e a complexidade técnica que existe no ambiente de uma instituição financeira. Com a conclusão do projeto, foi criado o modelo utilizando modelos ágeis de mercado para projetos em escala e seus resultados foram medidos por meio de entrevista qualitativa de modo que foi possível perceber a satisfação do cliente a qual se destina o software e também com os gestores do projeto com relação a eficiência das entregas. É recomendado estudos para definir os critérios de escolha do método tradicional ou ágil a ser utilizada na execução do projeto de modo que obtenha a melhor eficiência.

Palavras-chave

Scrum, escala, ágil, desenvolvimento de software, tradicional.

Abstract

This technical report is intended to describe a pilot project implemented in software factory of a major financial institution in order to create a software development life cycle model. This new model should be based on agile concepts and premises so that it has the flexibility to respond to change and deliver to the client features that generate value as soon as possible. The model should also be adherent to the company's business process meeting all legal requirements and internal rules, and also be suitable to large projects and the technical complexity which exist in the environment of a financial institution. At the completion of the project, the intended life cycle model was created by using existing examples for scale projects from the market. The results were measured by qualitative interview and it was possible to realize customer satisfaction with the resulted software and also managers satisfaction due to the efficiency of the deliverables. Further recommended research is to evaluate the criteria for pointing the best approach to be used in order to obtain the highest efficiency in a software development project: traditional or agile.

Keywords: Scrum, escaled, agile, development, waterfall

ISSN: 2317 - 8302

1. Introdução

Nos últimos anos o mercado tem exigido cada vez mais respostas rápidas e soluções inovadoras em um curto espaço de tempo. Isso exige das empresas evoluções tecnológicas rápidas de modo que estejam preparadas e flexíveis para mudanças. Esta flexibilidade na linha de produção dos sistemas de informática garante valor ao negócio seja agregado com mais eficiência (Michels & Ferreira, 2013).

Tradicionalmente os projetos de tecnologia da informação são executados no modelo *Walterfall.* Esse modelo desenvolvimento é amplamente e conhecido e divulgado por toda a comunidade de desenvolvimento de software e devido a isso, entre outros fatores, é bastante aderente às corporações de grande porte como no caso da instituição financeira que esta será estudada. O grande benefício que este modelo permite é a execução com grande fidelidade uma vez que já desde o início obriga que os requisitos e premissas estejam mapeados e definidos (Royce, 1970). Uma vez que não impede que a construção não seja iniciada antes que a especificação esteja pronta e da mesma forma que os testes não se iniciam antes do termino da construção (Royce, 1970). Este modelo apresenta falhas devido a todo este rigor, dificulta a inclusão de mudanças com o projeto em andamento, os testes são realizados de modo tardio o que em caso de erros irá gerar um custo elevado para correção devido aos impactos que este problema já pode ter causado na construção e por fim, apenas ao final do projeto o cliente irá receber uma versão para validação que também pode criar elevada custos de mudança ou correção.

O desenvolvimento de software é muito importante no mundo moderno, porém não existe um processo perfeito. (Chow & Cao, 2008). Durante a execução dos projetos de TI, esta instituição encontrava diversos problemas característicos na gestão de projetos tradicional que envolve esta área, como insatisfação do cliente, sem retorno financeiro em curto prazo, alto custo de gestão e pouca flexibilidade para mudanças e principalmente a dificuldade em definir os escopos e requisitos de maneira que o cliente tenha a visão de valor no produto que está sendo construído. Estes problemas ocasionam problemas de desgaste entre todos os *stakeholders* envolvidos no projeto, atrasos e entregas diferentes da expectativa do cliente o geram grande insatisfação e a sensação de projeto não entregue ou com problemas.

Diante deste cenário a empresa seguindo orientação de sua matriz localizada no continente europeu, iniciou o processo de inovação para criar uma metodologia adaptada originada de frameworks que são tendências no mercado de fábrica de software para o gerenciamento dos projetos. Esta metodologia tem como base os conceitos ágeis descritos em frameworks conhecidos como *SCRUM* DEVELOPMENT com o objetivo de unir as equipes, entregando mais valor ao cliente, com facilidade para acomodar as incertezas e indefinições encontradas nos projetos de desenvolvimento de software. Desta maneira, ela reduz o *time to market* e o ROI dos projetos além de diminuir a distância criada entre a fábrica software e a área de negócios mostrando transparência e fazendo com que o cliente participe de todas as etapas de construção mantendo assim o alinhamento da expectativa com a entrega. As metodologias ágeis em questão, quando aplicadas de maneira pura não atende as necessidades da empresa devido às questões legais e *compliances* que existem na companhia que não são aderentes aos *frameworks* utilizados como as questões referentes a qualidade de software, auditoria, confidencialidade, tamanho e complexidade dos projetos. Devido a estes pontos este projeto possui características de inovação para o modo de gestão atual e por este motivo a instituição optou em contratar uma consultoria especializada.

Questão de pesquisa: Como aplicar a metodologias ágeis em projetos de larga escala em instituições financeiras.

Objetivo: Descrever a metodologia criada com princípios ágeis para desenvolvimento de software em projetos de larga escala.

2. Referencial Teórico

2.1. Histórico - *Manifesto Agile*

No início dos anos 2000 a maioria dos projetos de desenvolvimento de software de sucesso utilizavam o modelo tradicional chamado de "Waterfall" criado no ano de 1970 que consiste em segmentar cada fase do projeto em etapas independentes e padronizadas permitindo que o projeto inteiro seja planejado de maneira organizada e programada, mantendo a rastreabilidade entre todos os artefatos (Royce, 1970).

Também nesta mesma época um grupo de metodologistas chamado de "A Agile Alliance" se reuniram para discutir a situação que se encontrava a matéria de desenvolvimento de software e entre eles estavam Jeff Sutherland, Ken Schwaber e Mike Beedle que mais tarde seriam os percursores do framework *Scrum*. Este grupo escreveu um manifesto de intenção de promover um novo modelo de desenvolvimento de software influenciado pela indústria japonesa e princípios enxutos ("Manifesto for Agile Software Development," 2015). Como resultado deste encontro foram descritos os princípios e valores ágeis que diz que indivíduos, interações, software em funcionamento, respostas rápidas a mudanças e colaboração com o cliente são prioridades ("Manifesto for Agile Software Development," 2015).

2.2. *Scrum*

Criado por Ken Schwaber e Jeff Sutherland que compuseram e estiveram presentes no encontro que resultou no "Manifesto Agile", o *Scrum* é um framework que tem como objetivo o desenvolvimento e a manutenção de produtos complexos. Atualmente é framework mais conhecido e utilizado (Vaidya, 2014). Utilizando um modelo adaptativo, criativo para resolução de problemas e, sobretudo produtivo durante o processo de fabricação de software buscando sempre entregar o mais alto valor ao cliente. O *Scrum* não é um processo ou técnica de desenvolvimento e sim um modelo que permite empregar vários processos e técnicas de gerenciamento de produtos e projetos e deixa sempre a opção de melhoria. Este framework é composto por artefatos, papeis e eventos com o proposito essencial e especifico para o sucesso. (Schwaber, K.; Sutherland, 2013)

O *Scrum* é fundamentado em teorias empíricas de controle de processo sustentado por três pilares básicos: (i) Transparência onde os resultados são visíveis em linguagem pública a todos os envolvidos. (ii) Inspeção e (iii) adaptação de todos os componentes do framework além do plano inicial de desenvolvimento.

O time é composto por três partes: O *Product Owner* que é o dono do produto é responsável pela qualidade, prioridade e orientação do que deve ser feito, o *Scrum Master* que é o dono do processo *Scrum* e pela remoção de impedimentos que estão impactando a produtivi*DAD*e do time e por fim o time de desenvolvimento que são as pessoas que efetivamente vão produzir deve ter tamanho entre 3 e 9 pessoas que possuem características

ISSN: 2317 - 8302

próprias como auto gerenciamento, multidisciplinaridade e também não é permitido ter sub times (Schwaber, K.; Sutherland, 2013).

O modelo também é composto por eventos, o principal deles é a "*Sprint*" que é o coração e a base do framework que tem tamanho (prazo) previamente definido antes do inicio da execução da construção com tempo que pode variar entre 7 a 30 dias, este é prazo para o atingimento da meta estabelecida para um time fixo igualmente predefinido. Estas duas características principais garantem que o projeto será entregue com qualidade, dentro do prazo e do custo esperado e partido destes princípios o que será clarificado e renegociado é o escopo. O andamento da *Sprint* é avaliado diariamente através do evento "daily meeting" que possui duração máxima de 15 minutos. Neste evento que tem como objetivo o alinhamento do status da Sprint, cada integrante do time, incluindo o *Scrum Master* e o *Product Owner,* posiciona o andamento das atividades que estão em execução e possíveis impedimentos. de modo que todos estejam alinhados e antecipem possíveis problemas. As metas da *Sprint* são definidas em outro evento chamado "*Sprint* Planning" que em conjunto com o *Product Owner* e o *Scrum* Master o time define quais são os itens a serem concluídos no termino da *Sprint* e para manter uns dos pilares do *Scrum* ao final da *Sprint* e o do projeto é realizada outro evento chamado de retrospectiva onde todos os envolvidos demostram as iniciativas que estão em pleno funcionamento e aqueles que precisam ser melhoradas, possíveis erros e acertos que impactaram a meta estabelecida. (Schwaber, K.; Sutherland, 2013)

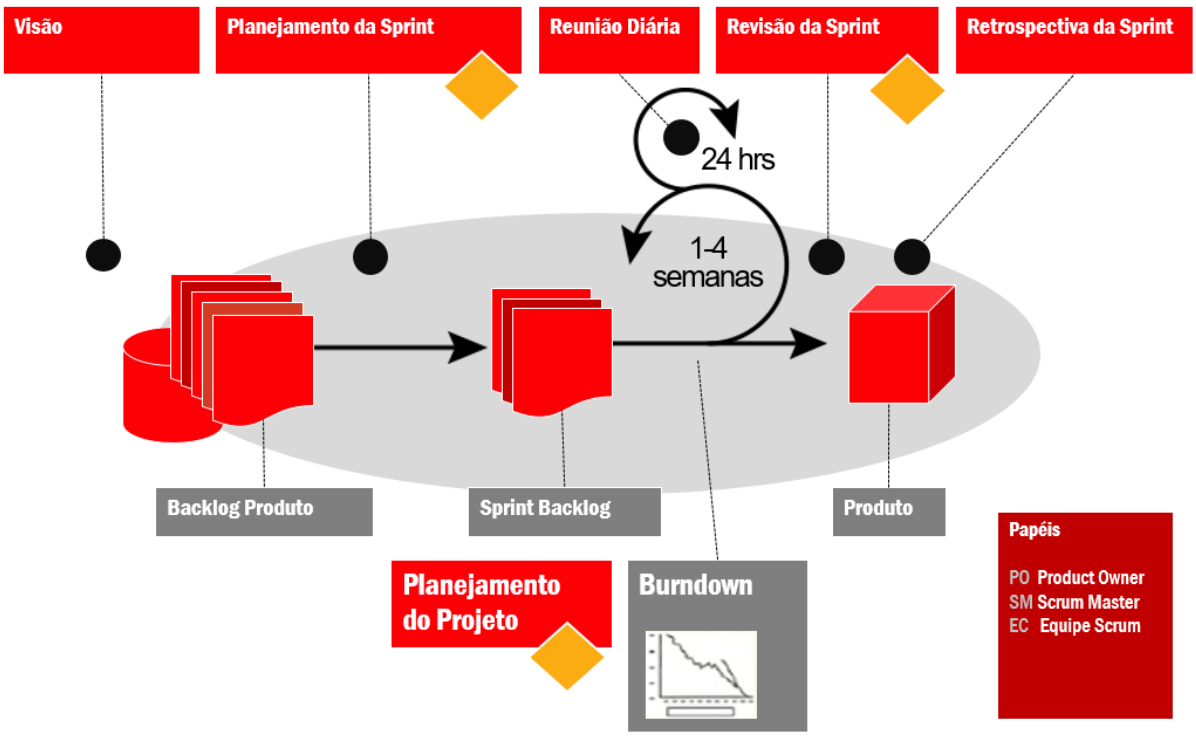

Figura 1 - Adaptado de "The Scrum Development Process"

2.3. *SAFe*® - Scaled Agile Framework ®

A adoção do framework *Scrum* em sua forma original não fornece suporte para estruturas de projeto mais amplas e complexas. Também é frágil para projetos grandes que

ISSN: 2317 - 8302

exigem grande número de times de desenvolvimento trabalhando de forma sincronizada como fornecedores externos, concorrência de *Product Owners*, integração com times de componente e orçamento além de não fazer nenhuma recomendação para o nível acima da equipe de forma escalável, apenas espera uma liderança ativa e comprometida. (Vaidya, 2014).

O modelo criado pela equipe liderada por Dean Leffingwell denominado *SAFe* (Scaled Agile Framework), é uma estrutura enxuta composta de três níveis na organização Team, Programa e Portfólio que funcionam com propostas independentes e com rituais próprios, porém de maneira integradas. Criado para absorver até 12 (doze) times de até 10 (dez) membros, totalizando até aproximadamente 130 (cento e trinta) indivíduos dedicados a entregar o maior valor possível para o cliente de maneira mais rápida (Leffingwell, 2007).

No nível de equipe o modelo está preparado para diferentes equipes formatadas com outros modelos e não necessariamente ágeis, é possível possuir times independentes onde cada um adota o modelo desejado. Já no nível de programa foram criados novos personagens chaves como o "Gerente de Produto" que é responsável pela priorização do Program Backlog e atua junto aos líderes dos times na otimização das funções e entregáveis. O *System Architect* é responsável pela orientação dos times referente arquitetura emergente do produto. O RTE (Release Train Engineer) é responsável pelo gerenciamento do conteúdo do release e novamente fazendo comparação ao *Scrum*, seria o *"Master Chief Scrum"*, em suas atribuições está também à gestão do risco, impedimentos e a melhoria continua. Em um nível superior existe a equipe de gerenciamento de portfólio que é o mais alto nível de gestão que é composto por executivos da empresa e estão alinhados com a estratégia e restrições em nível corporativo ("Scaled Agile Framework ® (*SAFe*®)," 2015).

Na operação o *SAFe* cria o conceito de *"Train"*, que possui o conceito bastante semelhante ao conceito de release, no nível de programa com o objetivo de desenvolver programas de larga escala, onde o programa é composto por múltiplos time atuando em sprints com interação continua e forte dedicação a sincronização das atividades mantendo a cadencia do desenvolvimento que é o objetivo do *"Train"*. Já no nível de portfólio o fluxo de valor para o usuário de negócios que é definido como uma sequência de "*trains"* desenvolvidos em etapas de longa duração. ("Scaled Agile Framework ® (*SAFe*®)," 2015)

ISSN: 2317 - 8302

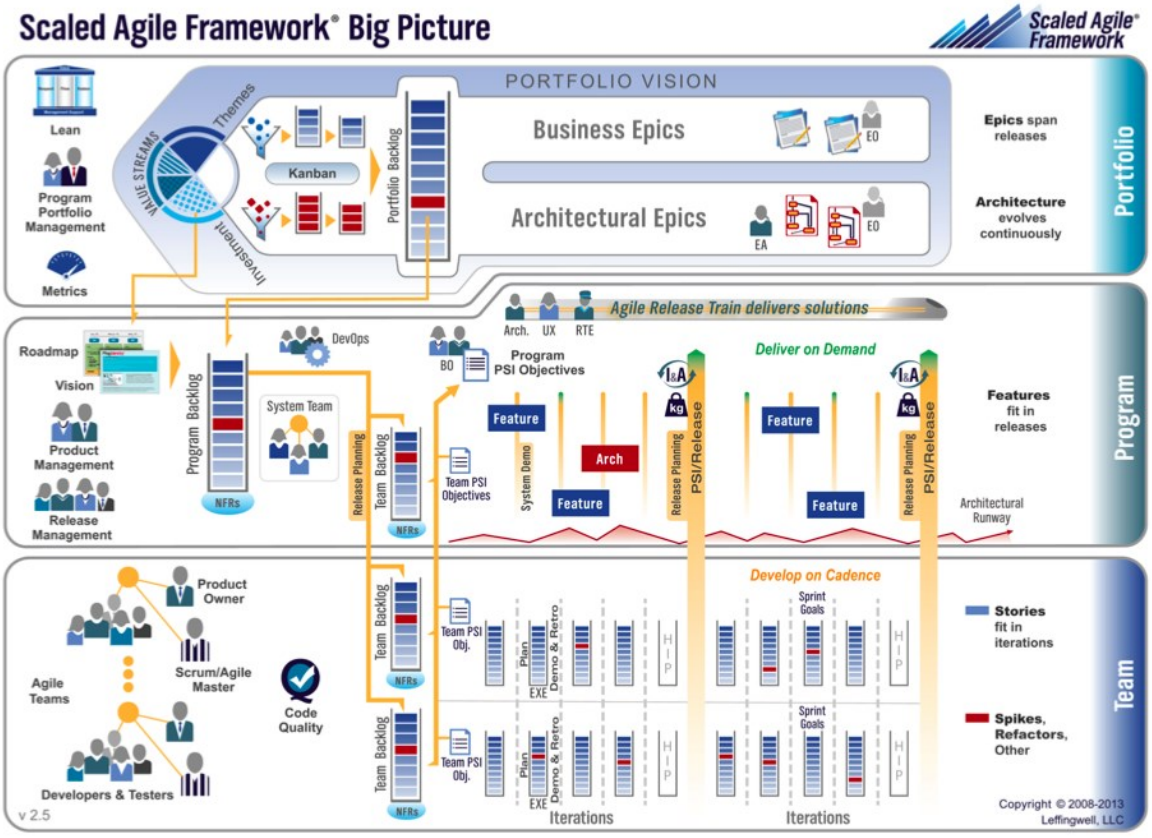

Figura 2 - Scaled Agile Framework - Big Picture

3. Disciplined Agile Delivery (*DAD*)

O modelo Disciplined Agile Delivery (*DAD*) foi criado pela IBM Rational e tem sua estrutura muito semelhante ao *Scrum*, porém muito mais robusto ou ainda é reconhecido por ser uma extensão do modelo de Ken Schwaber e Jeff Sutherland. O *DAD* é um modelo de desenvolvimento de software com uma abordagem iterativa e incremental atrás de um ciclo de vida orientado ao risco e ao valor. A execução e realizada de modo colaborativo e disciplinado, tendo como premissa a colaboração constante dos envolvidos na solução (Silva, 2015).

Os principais papeis contidos no *DAD* são os mesmos contidos no *Scrum* (*Scrum* Master, *Product Owner* e DevTeam) e além disso é introduzido um novo personagem reconhecido como "Team Lead" que é bastante semelhante ao *Scrum* máster mas agrega a função de proprietário arquitetura. Além dos papeis principais existem outros secundários que são introduzidos na aplicação dos modelos em projetos de escala como o Specialist, Tester Independent, Domain Expert, Especialista Técnico e Integrador (Ambler, 2013).

As práticas do *DAD* são divididas em quatro ciclos de vida distintos integrados entre eles. O primeiro ciclo chamado de Agile / Basic é subdivido em três fases: Inception que consiste na modelagem inicial e visão arquitetônica, a Construção trata-se do *Scrum* propriamente dita e Transição que é utilizada para fazer a implantação em produção. O segundo ciclo de vida é chamado de Advanced / Lean que é bastante semelhante ao modelo

ISSN: 2317 - 8302

de Kanban. O terceiro ciclo é chamado de "Continuous Delivery" que se trata de uma versão resumida do ciclo de vida do Lean. Já no quarto ciclo de vida que é chamado de "Exploratory" onde são implementadas soluções iniciais para facilitar os stackholders a fornecer informações sobre o novo produto (Ambler, 2013).

4. Metodologia de Pesquisa

Mudar o modelo de gestão de projetos dentro de uma empresa envolve diversos fenômenos sociais. Estes fenômenos são cultura, psicológicos, educacionais, políticos entre outros na qual se enquadram as ciências sociais, pois englobam relações humanas e sociais (Godoy, 1995a).

Seguindo a tendência de pesquisas que envolver as ciências sociais, optamos pela aplicação do método de pesquisa qualitativa. Este método de estudo é caracterizado por não ter um plano estabelecido e objetivando a precisão dos números como sugere o modelo quantitativo (Godoy, 1995a). O modelo qualitativo parte de um foco definido e ampla área de analise de modo que a definição das perguntas é feita conforme o andamento da entrevista ou do estudo através de obtenção de informações colhidas previamente ou ainda durante a própria entrevista objetivando um melhor entendimento sob o olhar do entrevistado dos fenômenos em estudo (Godoy, 1995a, 1995b).

Uma vez definido o modelo de pesquisa, foram definidos os agentes que foram entrevistados, um CIO (Chief Information Officer), dois gerentes de projetos, um diretor de metodologia e dois clientes. As entrevistas possuem questões abertas com o objetivo de analisar a percepção extraindo resultados detalhados as vezes não cobertos pelos números de uma pesquisa quantitativa. As entrevistas foram realizadas desde o início do projeto de modo a identificar a expectativa de cada um dos stackholders, colhendo informações de maneira informal durante o andamento do projeto com o objetivo de medir a satisfação e os ganhos com a utilização do método.

Foi escolhido um projeto com características importantes para a metodologia: (i) alta complexidade uma vez que envolve várias plataformas e integração de sistemas de diferentes áreas de negócio da instituição; (ii) escopo aberto e estimado em cerca de 400.000 (quatrocentas mil horas) incluindo horas de desenvolvimento, gestão e planejamento e (iii) envolvendo uma equipe fixa de 60 pessoas diretamente e outros parceiros prestadores de serviços como equipe de componente. Estas características foram importantes para que o modelo que será criado, fosse abrangente e absorvesse as características contidas nos demais projetos da empresa, assim como as regras contidas no negócio.

5. Analise dos resultados

A empresa já havia experimentado em outras oportunidades o desafio de se criar este modelo de desenvolvimento e mudar a sua cultura referente a maneira de se fazer software, todas sem sucesso por motivos desconhecidos. Nesta nova oportunidade optou-se em contratar uma empresa de consultoria com expertise em metodologia ágil em seu portfólio. Para o sucesso deste projeto foi necessário quebrar alguns paradigmas dentre eles podemos citar o principal como sendo: fazer com que o usuário participe do processo de construção desde o momento da concepção do projeto.

A consultoria foi iniciada com um contrato com prazo inicial de 1 (um) ano e as atividades foram iniciadas através da realização do mapeamento do processo atual de

produção do software em todas as etapas desde a aprovação do orçamento até a implantação em produção incluindo todo o ciclo de vida do software. Nesta etapa foram desprezados projetos com demandas pontuais ou pequenas melhorias, somente foram admitidos projetos em que fosse produzido um novo produto ou que o projeto seja de construção de uma nova solução dentro dos produtos existentes. Verificou-se que nesta análise que o processo possuía características peculiares de grande impacto para o framework *Scrum*, entre os mais importantes destaca-se: (i) contratação de empresa terceirizada para execução do projeto, (ii) processo redundante e burocrático de controle de documentação, sendo que estas passavam por muitos ajustes ao longo da execução, (iii) equipes virtuais, (iv) equipe de componentes (v) equipes grandes chegando uma única equipe possuir 60 (sessenta) pessoas, (vi) grande número de gerentes de projetos envolvidos em um único produtos.

Nesta primeira fase do projeto que compreendia todo o contrato, o foco foi nas etapas de designer, codificação e homologação da construção do software desprezando as etapas de aprovações, aquisições, treinamento e *rollout* do produto desenvolvido. O objetivo era demostrar de maneira rápida as vantagens e benefícios do modelo e após isso escalar o modelo para outros projetos do portfólio da instituição.

Figura 3 - Ciclo de Vida do Software

Após a definição do escopo onde seria realizado o trabalho, o passo seguinte foi teve como objetivo mensurar a maturidade da fábrica de software frente as metodologias ágeis. Como não existe um modelo de maturidade padrão de mercado, foi elaborada uma pesquisa quantitativa por meio de um questionário que contém perguntas agrupadas por papeis, eventos, artefatos, práticas e questões gerais baseadas nos modelos sugeridos por Benefield e Ramachandran (Benefield, 2010; Patel & Ramachandran, 2009) de modo que seja possível medir a cobertura e aderência do modelo ágil no processo atual. Estas perguntas foram feitas aos líderes de áreas de alguns projetos em que já havia sido experimentado no modelo ágil sem sucesso. A conclusão foi que não havia uma definição clara dos processos ágeis, tão pouco os artefatos e funções, alguns itens que foram apontados como aderentes não traduz a aderência e sim a semelhança ao modelo tradicional.

O projeto no qual seria realizada a criação do modelo foi escolhido arbitrariamente pela gestão estratégica da empresa onde os critérios não foram divulgados. O projeto escolhido não poderia ser desenvolvido com o modelo ágil devido ao tamanho, riscos e impacto relacionados (Boehm & Turner, 2003). A posição de Boehm & Turner é questionada após os modelos, complementares ao *Scrum* com o *SAFe* e o *DAD*. Estes modelos dão a abrangência e sofisticação faltantes no *Scrum*.

A primeira definição foi o foco principal da atuação era a fase de definição e construção do produto definido no modelo de Dean Leffingwell, o *SAFe*, (Leffingwell, 2007) como "*Team*". Este *"team"* irá atuar na definição e construção do software além de interagir com os PMO (Project Management Officer) tradicionais e demais executivos da companhia, desprezando questões como aquisições, gerenciamento de mudança, implantação em ambiente de produção e treinamentos.

Diante do cenário encontrado no *assessment* (pesquisa inicial de maturidade do ágil), a estratégia traçada foi a de fazer diversos *workshops* com todos os possíveis *stakeholders* do projeto, mesmo antes do início formal do projeto. Através destes workshops foram avaliadas as competências de algumas pessoas que compunham a equipe do projeto e destas foram destacadas 3 (três) *Scrum Masters* e 4 (quatro) *Product Owners*.

A empresa optou por comprar o projeto de uma empresa terceirizada para a desenvolvimento deixando apenas os assuntos relacionamentos a área de negócios sob sua responsabilidade. A contratação e remuneração da empresa contratada são feita por entrega dos itens do *Product Backlog* que são previamente orçados e aprovados antes da construção. Este modelo de contratação não é o apropriado e sim a contratação realizada no modelo custos reembolsáveis (Project Management Institute, 2014) uma vez que os projetos ágeis atuam com custo, prazo e qualidade fixo (Schwaber, K.; Sutherland, 2013) e também com a alocação full time. O próximo passo foi à definição da estrutura das equipes e como elas iriam interagir. Nesta etapa, foram definidos três times de desenvolvimento compostos cada um por: um arquiteto de soluções, responsável por criar a solução técnica, os modelos de banco de dados e arquitetura de software; Quatro desenvolvedores especialistas em Java que seriam responsáveis pelo desenvolvimento das telas que os usuários iriam operar; três testadores de software em nível operacional conforme modelo *Scrum*. (Schwaber, K.; Sutherland, 2013). O analista de negócios, *Product Owner* e *Scrum Master* seriam utilizados de modo compartilhado entre os times assim como outros profissionais especializados. Estes profissionais são os recursos de manutenção e operação de bancos de dados (DBA's), time de infraestrutura, desempenho e data-center além do time de desenvolvimento de baixa plataforma ou que envolviam sistemas críticos para a operação comercial da empresa (Leffingwell, 2007).

A agenda do ciclo de vida do release foi definida da seguinte forma. Dez dias uteis para definições, esclarecimentos de dúvidas e elaboração das especificações técnicas e funcionais. A construção do software foi planejada em forma de *Sprint*s de 10 (dez) dias uteis de codificação além de outros 10 (dez) dias uteis de homologação (Schwaber, K.; Sutherland, 2013). O processo de certificação e auditoria foi deslocado do release devido a criticidade do processo e também por ser uma área extremamente regulamentada.

O ciclo de vida de um release se inicia quando o *Product Backlog* for priorizado pelo *Team Product Management*. Estes itens são agrupados conforme o conceito de *"train"* do framework *SAFe* (Leffingwell, 2007) e subdivididos de modo que seja absorvido dentro de uma *Sprint.* O trabalho é iniciado por meio de uma reunião conduzida pelos *Product Owners* e todo o time designado participam discutindo o funcionamento e os detalhes de cada um dos itens. A saída desta reunião são anotações e definições que servirão de insumos para a produção da especificação técnica e funcional.

A questão da especificação técnica e funcional merece um destaque especial. Diferente do modelo sugerido no *Scrum* (Schwaber, K.; Sutherland, 2013), os gestores optaram em produzir este documento antes da construção do software com o objetivo de obter a aprovação prévia dos *Product Owners* devido à necessidade contratual e controle de mudanças que quando impactar a entrega da *Sprint*s necessita de renegociação comercial. O modelo original da especificação foi alterado por um modelo mais simples baseado na proposta orientada a comportamento (Haumer, 1998; Leite et al., 1997).

Ao final do primeiro *time-box* de 10 (dez) dias úteis previstos para o release, as especificações devem estar prontas e aprovadas, os times novamente são reunidos para a realização da cerimonia de *release planning* onde é definido qual dos times ficará responsável por item e a priorização de desenvolvimento além das metas diárias dos times. Esta cerimônia

possui duração de 4 horas e está alocada no primeiro horário do primeiro dia. O segundo time-box, também de 10 dias uteis, além da cerimonia já citada, compreende a construção do software e demais artefatos e rotinas que compõe o processo de fabricação como, por exemplo, os testes unitários e integrados, construção de manuais e documentos exigidos pela legislação vigente. Todo este processo é acompanhado pelo *Product Owner* e quando necessário é criado um pacote funcional chamado *"system demo"* para avaliação dos usuários finais de modo que eventuais mudanças sejam implementadas sem grande impacto. Durante todos os dias é realizada também outra cerimonia denominada "*Daily Meeting*" com duração de 15 a 20 minutos com todos os envolvidos com o objetivo de alinhamento do andamento das tarefas e esclarecimentos de dúvidas e identificação ou solução de impedimentos. No último dia é realizada geração do pacote de entrega e disponibilizada no ambiente de certificação. Como última tarefa deste time-box é realizado a cerimonia de fundamental importância chamada de "Sprint Review" onde são debatidas todas as ações que impactariam positivamente ou negativamente o andamento da sprint (Abrahamsson, Salo, Ronkainen, & Warsta, 2002; Chow & Cao, 2008; Leffingwell, 2007; Schwaber, K.; Sutherland, 2013).

No último time-box com 5 dias uteis é onde são executados os testes de performance, auditoria de qualidade e onde acontece a entrega formal para o cliente que neste período realiza testes funcionais e planeja a estratégia de implantação com os gestores envolvidos. O processo de implantação pode ser realizado a qualquer momento limitado apenas pela geração do próximo pacote. Eventuais erros e mudanças encontradas nesta fase irão ser direcionados para o time de desenvolvimento como novo item de *program backlog* que deverá seguir novamente todo o fluxo normal.

6. Conclusão

Não foi possível implantar um framework puro como já havia sido experimentado anteriormente a este projeto. Os frameworks utilizados possuíam lacunas que impactavam de alguma maneira o andamento do projeto. Devido a isso se optou em criar um modelo proprietário da empresa onde este modelo possui características principais do *Scrum* para a linha operacional e do *SAFe* Agility para a gestão do programa e interação com as demais áreas da companhia.

Utilizando estes modelos de maneira única ou combinada teremos um fluxo de trabalho que permite a antecipação do termino mais cedo, redução do *lead-time* de construção, antecipação de erros e validação constante por parte do usuário o que gera menos impacto na construção e entrega e mantem o time ocupado de maneira linear durante todo o tempo de execução do projeto. (Wang, Conboy, & Cawley, 2012)

Após a conclusão da primeira fase do projeto, observou se uma redução em até 42% no tempo de entrega dos itens de backlog se comparado a gestão tradicional, equipes técnicas e de negócios mais alinhadas, com expectativa e objetivos claros, com menor custo de gestão e facilidade na comunicação. Com relação à equipe de projeto observou certo nivelamento dos recursos ao longo de todo o projeto com menor custo de gestão. Os resultados apresentados não seriam possíveis se não existisse apoio da alta gestão da empresa e do engajamento das pessoas envolvidas.

As limitações do relato e do método de pesquisa devem-se a opção de realizar o projeto utilizando apenas um único projeto com características peculiares e também por não alterar algumas características da empresa que teriam potencial para revisão de processo e otimização.

7. Referencias

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). Agile software development methods. *Vtt Publications*, *478*, 167–168. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.161.5931&rep=rep1&type=pdf

Ambler, S. W. (2013). Going Beyond Scrum Disciplined Agile Delivery, (October). Retrieved from http://disciplinedagiledelivery.wordpress.com/2013/10/24/beyondscrum/

Benefield, R. (2010). Seven dimensions of agile maturity in the global enterprise: A case study. *Proceedings of the Annual Hawaii International Conference on System Sciences*, 1–7. http://doi.org/10.1109/HICSS.2010.337

Boehm, B., & Turner, R. (2003). Observations on balancing discipline and agility. *Proceedings of the Agile Development Conference, 2003. ADC 2003*. http://doi.org/10.1109/ADC.2003.1231450

Chow, T., & Cao, D.-B. (2008). A survey study of critical success factors in agile software projects. *Journal of Systems and Software*, *81*(6), 961–971. http://doi.org/10.1016/j.jss.2007.08.020

Godoy, A. S. (1995a). Introdução à Pesquisa Qualitativa e Suas Possibilidades. *Revista de Administração de Empresas*, *35*(2), 57–63.

Godoy, A. S. (1995b). Pesquisa Qualitativa Tipos Fundamentais. *Revista de Administração de Empresas*, *35*(3), 20–29.

Haumer, P. (1998). Requirements elicitation and validation with real world scenes. *IEEE Transactions on Software Engineering*, *24*(12), 1036–1054. http://doi.org/10.1109/32.738338

Leffingwell, D. (2007). *Scaling Software Agility: Best Practices for Large Enterprises* (1st ed.). Addison-Wesley Professional.

Leite, J. C. S. P., Rossi, G., Balaguer, F., Maiorana, V., Kaplan, G., Hadad, G., & Oliveros, a. (1997). Enhancing a requirements baseline with scenarios. *Proceedings of ISRE '97: 3rd IEEE International Symposium on Requirements Engineering*, 184–198. http://doi.org/10.1109/ISRE.1997.566841

Manifesto for Agile Software Development. (2015). Retrieved January 1, 2015, from http://www.agilemanifesto.org/

Michels, E., & Ferreira, M. G. G. (2013). Gerenciamento Ágil no Processo de Desenvolvimento de Produtos Inovadores: Uma Análise Bibliográfica Sistemática. *Revista de Gestão E Projetos - GeP*, *04*(01), 52–76. http://doi.org/10.5585/gep.v4i1.119

Patel, C., & Ramachandran, M. (2009). Agile maturity model (AMM): A Software Process Improvement framework for agile software development practices. *Int. J. of Software Engineering, IJSE*, *2*(I), 3–28. http://doi.org/10.4304/jsw.4.5.422-435

Project Management Institute. (2014). *Um Guia do Conhecimento Em Gerenciamento de Projetos - Guia Pmbok®* (5ª Edição).

Royce, W. W. (1970). Managing the development of large software systems. *Electronics*, *26*(August), 1–9. http://doi.org/10.1016/0378-4754(91)90107-E

Scaled Agile Framework ® (SAFe®). (2015). Retrieved January 1, 2015, from http://www.scaledagileframework.com/

Schwaber, K.; Sutherland, J. (2013). Guia do SCRUM. *Harvard Business Review, Boston*, *IV*, 163–179. Retrieved from https://www.scrum.org/Portals/0/Documents/Scrum Guides/Scrum_Guide.pdf

Silva, D. L. (2015). IBM Developerworks - Disciplined Agile Delivery. Retrieved from https://www.ibm.com/developerworks/mydeveloperworks/blogs/rationalbrasil/entry/uma_introdu_c3_a7_c3_a3o_ao_disciplined_agile_delivery_parte_i19?lang=en

Vaidya, A. (2014). Does DAD Know Best , Is it Better to do LeSS or Just be SAFe ? Adapting Scaling Agile Practices into the Enterprise. In *PNSQC.ORG* (pp. 1–18).

Wang, X., Conboy, K., & Cawley, O. (2012). "Leagile" software development: An experience report analysis of the application of lean approaches in agile software development. *Journal of Systems and Software*, *85*(6), 1287–1299. http://doi.org/10.1016/j.jss.2012.01.061