A Engenharia de Requisitos
e o Desenvolvimento Ágil
– colaborativo, just enough, just in time, sustentável -

Rainer Grau e Kim Lauenroth,
com apoio de Bogdan Bereza,
Erik van Veenendaal
e Sven van der Zee.

Tradução de Paul Tornquist
Em
02 de setembro de 2014

Patrocínio

www.tmtestes.com.br

Recognized
Training Provider

Sumário

1 Introdução

1.1 A Engenharia de Requisitos como técnica especializada

As organizações que trabalham em um ambiente ágil comprovadamente entregam produtos ou serviços mais próximos das necessidades e expectativas do cliente. Os benefícios da metodologia ágil incluem maior flexibilidade, rapidez no lançamento do produto ou serviço para o mercado e ganhos de produtividade. Esses benefícios não vêm de graça. As organizações que seguiram esse caminho realizaram grandes investimentos no desenvolvimento de pessoas, em treinamento e *coaching*, em métodos e técnicas, bem como em ferramentas de suporte para esses métodos e técnicas (ver [1]).

Um exemplo muito bom para tal investimento é o movimento em torno de conceitos como o desenvolvimento orientado a testes de aceitação (*acceptance test driven development*) e entrega contínua (*continuous delivery*). Por trás desses dois termos há um grande conjunto de técnicas e métodos, complementados por investimentos significativos em ambientes, ferramentas e no desenvolvimento de pessoas, e por último mas não menos importante, uma mudança de mentalidade em relação à colaboração. As organizações que conseguiram percorrer esse longo caminho de melhoria contínua na entrega de testes e no desenvolvimento de software nunca mais voltaram ao que faziam antes (ver [5]).

No que diz respeito à disciplina de Engenharia de Requisitos, a comunidade ágil está insastisfeita com as abordagens tradicionais, com seus documentos iniciais que podem às vezes chegar a centenas de páginas e criam "Especificações" que já estão desatualizadas quando a primeira linha de código está sendo desenvolvida. Essa insatisfação está inclusive expressa no manifesto para o desenvolvimento ágil de software, que afirma valorizar:

"Software em funcionamento, mais que documentação abrangente."

Infelizmente, isso teve como resultado que a comunidade de desenvolvimento ágil passou a considerar "*antiquadas*" as descrições da disciplina de Engenharia de Requisitos: *"Isso aí é coisa de antigamente, inútil, agora precisamos fazer de outro jeito."*

Neste ponto, no entanto – como representantes do *International Requirements Engineering Board* (IREB) – queremos expressar a nossa posição taxativa: A Engenharia de Requisitos é uma disciplina independente de qualquer modelo de processo ou abordagem de desenvolvimento de software. A Engenharia de Requisitos é definida pelo IREB nos seguintes termos(ver [2]):

> A Engenharia de Requisitos é uma abordagem sistemática e disciplinada para a especificação e o gerenciamento de requisitos, com os seguintes objetivos:
>
> [1] Conhecer os requisitos relevantes, alcançar um consenso entre os *stakeholders* sobre esses requisitos, documentá-los de acordo com determinadas normas, e gerenciá-los de forma sistemática.
>
> [2] Compreender e documentar os desejos e necessidades dos *stakeholders*.
>
> [3] Especificar e gerenciar os requisitos para minimizar o risco de entregar um sistema que não atende os desejos e necessidades dos *stakeholders*.

Além disso, acreditamos que a disciplina da Engenharia de Requisitos constitui uma competência central em qualquer processo de desenvolvimento de software, uma vez que ela é responsável pela comunicação contínua e a confirmação do que precisa ser feito (e por que) para atender as necessidades de *stakeholders* e usuários. A Engenharia de Requisitos é uma técnica especializada essencial e necessária no mundo complexo do desenvolvimento de produtos e serviços.

Nossa resposta para a objeção "Isso aí é coisa de antigamente, inútil, agora precisamos fazer de outro jeito" é a seguinte: "Coisa de antigamente?" Sim. "Inútil?" Não! "Fazer de outro jeito?" Tudo bem, sem problema! Muitos membros do IREB estão envolvidos ativamente no movimento ágil e compartilham a mentalidade das organizações ágeis, buscando aumentar a flexibilidade, a produtividade e a rapidez de entrega de produtos e serviços para o mercado.

1.2 Sobre este artigo

Este artigo discute a disciplina de Engenharia de Requisitos sob o ponto de vista de diferentes modelos de ciclo de vida. O artigo sustenta que a Engenharia de Requisitos é um fator essencial de sucesso em qualquer modelo de ciclo de vida, seja no modelo cascata ou no modelo ágil.

No item 2, o artigo reflete sobre o que leva muitas organizações a substituir modelos de ciclo de vida sequenciais por modelos iterativos e ágeis. O objetivo dessa mudança é estabelecer um fluxo contínuo de requisitos entre usuários e clientes em toda a organização. A incerteza é um aspecto importante nessa reflexão.

No item 3, o artigo discute a aplicação de atividades da Engenharia de Requisitos em ambientes ágeis. Veremos que o *know-how* da Engenharia de Requisitos é relevante tanto para modelos de ciclo de vida em cascata quanto para modelos de ciclo de vida ágil.

O item 4 aborda as diferenças de aplicação, a sequência e o momento adequado das atividades relacionadas com a Engenharia de Requisitos nos modelos de ciclo de vida em cascata e nos modelos ágeis.

O item 5 deste artigo considera se a Engenharia de Requisitos, enquanto técnica especializada, é compatível com a mentalidade ágil, analisando lado a lado a definição de Engenharia de Requisitos do IREB e o manifesto ágil. Essa validação é essencial para determinar se o conjunto de conhecimentos IREB é abrangente em ambientes ágeis.

Mas, vamos começar com a causa fundamental da maioria dos nossos problemas na engenharia de software: a incerteza.

2 A Incerteza é um Fato

2.1 A incerteza é uma característica dos modernos mercados orientados ao cliente

A demanda por maior flexibilidade e produtividade tem suas raízes em ciclos de produto dramaticamente mais curtos e na elevada pressão do mercado para entregar produtos e serviços que encantam os clientes. As pesquisas demonstram que os ciclos de produto reduziram-se drasticamente, passando de uma média de seis anos na década de 1980, para seis meses em 2010 [6].

Isso teve um forte impacto sobre as organizações. Como os círculos de inovação são muito rápidos, a pressão para encurtar o tempo de entrega para o mercado é alta, e os horizontes de planejamento de longo prazo perdem relevância frente a um mercado e uma economia em transformação constante. A confiabilidade do planejamento de médio e longo prazo é baixa devido ao aumento da incerteza sobre o desenvolvimento do mercado. Uma organização será bem sucedida sempre que conseguir adaptar rapidamente seu modelo de negócio e seu portfólio de produtos e serviços em resposta às forças dinâmicas do mercado.

O desenvolvimento ágil e *lean* é uma abordagem que favorece a performance em condições de maior incerteza. As empresas que apresentam um bom desempenho nessas condições muitas vezes adotam métodos ágeis e outras técnicas iterativas em toda a organização. Em mercados caracterizados por mudanças rápidas e orientados para o consumidor, os modelos de ciclos de vida ágeis e iterativos comprovadamente aumentaram a taxa de sucesso dos projetos [1].

2.2 Por que as organizações insistem no modelo em cascata

Dada a evidência de que um modelo de ciclo de vida ágil aumenta a taxa de sucesso em mercados em rápida mudança, por que muitas organizações ainda insistem com abordagens clássicas, caracterizadas por grandes fases iniciais de elicitação de requisitos, resultando em enormes especificações e terminando em processos em cascata?

Entre outros, os motivos mais importantes são:

A ilusão da previsibilidade: A gestão clássica trata a incerteza como uma questão pessoal. Sua auto-percepção é: "Precisamos saber exatamente para onde estamos indo." Nessa mentalidade, a gestão define as metas. Como resultado, as metas, o plano e o orçamento geralmente são fixos. Quaisquer alterações nas metas, planos e orçamentos são vistas como problemas e riscos para o atingimento das metas, e não como aspectos "normais" da vida sob as condições de um mercado dinâmico. A gestão orientada a metas, planos e orçamentos identifica nos modelos de ciclo de vida ágil um risco adicional para atingir as metas que foram acordadas, em vez de um mecanismo inteligente de adaptação a mudanças.

Projetos (empresariais) possuem marcos (milestones**) fixos**: As organizações que oferecem produtos e serviços não precisam apenas lidar com mercados em constante mudança. Outro aspecto igualmente importante é o cumprimento de normas legais e outros regulamentos. Temos como exemplo: bancos, seguradoras, serviços médicos, agências ambientais, distribuidoras de energia, entre outros setores. Se uma norma muda, a organização tem um certo prazo para entrar em conformidade com este regulamento. Assim, tanto a meta (assegurar conformidade com a norma) quanto o componente de planejamento (até o prazo “x”) são determinações externas. Em organizações onde grandes investimentos são direcionados para questões de conformidade e regulamentos, a mentalidade focada em metas, planos e orçamentos é predominante. A fatia mais dinâmica e orientada para o mercado, do orçamento de investimento - muitas vezes por razões de padronização - é tratada com a mesma mentalidade: metas, planos, orçamentos.

As organizações com outro histórico ou que operam em um ambiente diferente assumem o risco de agir de outra forma. As histórias de sucesso da Amazon, Facebook ou Google demonstram que a adaptação rápida para mercados em constante mudança, com base no *feedback* do usuário, é uma abordagem válida para lidar com a incerteza. Muitos serviços bem sucedidos do Google começaram com uma versão beta e surgiram a partir do *feedback* dos usuários. Os usuários apreciam esse comportamento. A gestão nessas organizações é do tipo: *"Não sabemos exatamente qual é a meta. Na verdade, são os nossos usuários que sabem. Mas sabemos como chegar lá o mais rapidamente possível"*. Essa é uma abordagem muito diferente das organizações clássicas, uma abordagem que incorpora princípios ágeis e *lean*.

3 A Engenharia de Requisitos e a Agilidade

Seguir a ideia da inovação incremental através do desenvolvimento iterativo realizado em pequenos passos, como a Amazon, o Facebook ou o Google, implica em mudanças fundamentais no modelo operacional de uma organização. As coisas que antes eram feitas em sequência precisam ser executadas paralelamente. Isso se aplica a todos os diferentes aspectos do desenvolvimento de produtos e serviços, desde o nível estratégico até o operacional.

3.1 A agilidade promove um fluxo contínuo de informações nas organizações

O fluxo clássico de informação sequencial, começando com a definição de uma estratégia, decompondo essa estratégia *top-down* na definição de um portfólio e um plano para desenvolver um conjunto de produtos com um roteiro fixo, não vai funcionar nesses cenários. A construção de um plano de negócios fixo, abrangendo um período de um a cinco anos, não existe mais.

Em vez disso, um fluxo de trabalho contínuo, de caráter marcadamente paralelo em todos os níveis de uma organização, promove a influência e o *feedback* recíproco. O fluxo de informações do nível estratégico até o operacional é tão importante quanto o fluxo de *feedback* dos usuários, seja a nível de produto ou do portfólio, para as atividades de definição de estratégia, refletindo mudanças do mercado atual.

Os indivíduos em uma organização trabalham em paralelo na adoção da estratégia (baseado em *feedback bottom-up*), no desenvolvimento do portfólio de produtos e serviços (em alinhamento com a nova estratégia e para criar o portfólio existente com base no *feedback* dos clientes), bem como no desenvolvimento e operação (a entrega da próxima versão do produto).

3.2 O impacto sobre a Engenharia de Requisitos

Os modelos de ciclo de vida em cascata ordenam as atividades em sequência com funções predefinidas. Cada etapa do processo tem *inputs* e *outputs* previamente definidos, e cada etapa tem duração definida. A experiência industrial[1] demonstra que um processo sequencial como esse geralmente não alcança a meta almejada, pois a equipe de desenvolvimento e os *stakeholders* aprimoram a sua compreensão do sistema ao longo de todo o processo de desenvolvimento. As lições aprendidas com a melhor compreensão do âmbito do problema precisam voltar para o projeto em forma de *feedback*.

[1] Essa experiência já foi apresentada por Winston W. Royce em seu artigo “Gerenciando o Desenvolvimento de Grandes Sistemas de Software" (muitas vezes considerado como a origem do conceito “cascata”).

Nos modelos de ciclo de vida em cascata, esse *feedback* toma a forma de um processo de gerenciamento de mudanças, com medidas explícitas para gerenciar o fluxo de informações que retornam para os projetos a partir das lições aprendidas, dos problemas encontrados e de mudanças externas no contexto de trabalho. Em muitas organizações, o processo de gestão de mudança é uma estrutura pesada, com alto impacto em termos de custo e trabalho.

A ordem das atividades e a forma de colaboração é diferente no modelo de ciclo de vida ágil. Ao longo de todo o projeto, a equipe trabalha os requisitos em estreita colaboração com os *stakeholders* e usuários, desenvolvendo o sistema produtivo de forma incremental. O objetivo central do modelo de ciclo de vida ágil é obter *feedback* de maneira rápida e direta dos *stakeholders* e usuários, de preferência através do uso do próprio sistema. O *feedback* direto obtido a partir do uso do sistema mostrou ser a melhor evidência de que o desenvolvimento do sistema está seguindo a direção pretendida pelos objetivos do sistema. Na disciplina da Engenharia de Requisitos, as equipes ágeis trabalham continuamente com os *stakeholders* e usuários:

- Na elicitação de requisitos, ou seja, na identificação e especificação de novos épicos e histórias de usuários alinhadas aos objetivos dos *stakeholders* e dos usuários, ou em atividades de refinamento do *backlog*.
- No detalhamento maior de épicos e histórias de usuários até que sua definição esteja confirmada como completa, bem como na especificação de critérios de aceitação para que a equipe tenha todas as informações necessárias para desenvolver, a partir desses requisitos, uma parte produtiva do sistema que forneça o valor esperado para o negócio.
- No desenvolvimento de requisitos de *design*, para atender requisitos adicionais implícitos e não-funcionais provenientes de áreas como usabilidade (aspectos como atratividade, facilidade de aprendizagem, compreensibilidade).
- Nos testes de verificação e validação da Engenharia de Requisitos executados nas partes do sistema desenvolvidas até agora com qualidade e status potencial de entrega.

A equipe executa todas essas atividades em paralelo. Cada membro da equipe atua em mais de uma função. Ao discutir a história do usuário e definir critérios de aceitação com um *stakeholder*, um membro da equipe desempenha o papel de engenheiro de requisitos. Ao implementar a história do usuário, preferencialmente após um processo de design orientado por testes de aceitação, o membro da equipe atua como desenvolvedor e testador.

A diferença entre os dois modelos de ciclo de vida está na maneira em que os papéis são atribuídos às pessoas. As organizações que preferem o modelo em cascata muitas vezes atribuem a uma pessoa exatamente um papel. O efeito desse engessamento de papéis é que ao longo do processo de desenvolvimento, muitos indivíduos trabalham em uma função do produto. Assim, a responsabilidade pelas funcionalidades do produto é distribuída entre vários indivíduos. A documentação inicial substitui a comunicação direta e o *feedback* com os *stakeholders*, resultando em suposições que podem mais tarde revelar-se equivocadas e acabam gerando solicitações de mudança. A organização de um projeto como esse funciona como uma espécie de "telefone sem fio" entre os vários papéis que participam no desenvolvimento de um requisito. Nas organizações que preferem o modelo de ciclo de vida ágil, cada membro da equipe desempenha mais de um papel e troca de papéis conforme o trabalho exige. A comunicação direta com os *stakeholders* e o *feedback* rápido mantêm atualizado e ativo o conhecimento da equipe, o que reduz a necessidade de documentação.

3.3 O impacto das suposições

Outro fator importante nos mercados em rápida mudança é a confiabilidade e a durabilidade das suposições. Um requisito que não é uma meta predefinida, não é uma restrição fixa e não foi validado ainda pelo usuário pode ser considerado uma suposição. Tal requisito descreve uma característica do produto ou do sistema que virá a existir no futuro. Em última análise, no entanto, devemos confessar que todas essas afirmações são suposições até que tenham sido validadas pelo produto desenvolvido. Suposições podem ser válidas, mas podem também estar incorretas. Falhas baseadas em suposições incorretas ocorrem tanto em projetos em cascata quanto em projetos ágeis e precisam ser enfrentadas. O importante é entender as implicações do respectivo modelo de ciclo de vida ao lidar com suposições inválidas:

- O período entre a formulação inicial da suposição e sua validação pelo sistema desenvolvido é muito maior em projetos em cascata do que em projetos ágeis. Em projetos em cascata, as suposições são validadas dentro de uma fase explícita de integração e testes de aceitação pelo usuário. Projetos ágeis, com um ciclo curto de iteração, validam as suposições em cada iteração a partir de testes de aceitação do usuário e sessões de revisão com os *stakeholders*.

- Quanto maior o período para validar uma suposição no confronto com a realidade, maior é a probabilidade de mudanças, especialmente em ambientes onde o contexto em torno do próprio projeto sofre mudanças. Mesmo estando correta no momento em que foi formulada, é mais provável que uma suposição venha a se tornar inválida durante o tempo de ciclo mais longo de um projeto em cascata, do que no curto tempo de ciclo de um sprint em um projeto ágil.

- A intenção dos projetos em cascata de gerar inicialmente uma especificação "completa" de requisitos aumenta o número de suposições. Isso agrava ainda mais as dificuldades mencionadas acima com relação a suposições. Requisitos abstratos são decompostos em um conjunto de requisitos mais concretos. Caso um requisito abstrato tenha sido baseado em uma suposição inválida, a cadeia completa de requisitos derivados dessa suposição será inválida também. Tais cadeias de dependência podem aumentar consideravelmente o impacto de uma suposição falsa. O potencial de retrabalho em projetos em cascata é maior do que em projetos ágeis, resultando em mais pedidos de alterações relacionados a suposições inválidas.

Modelos de ciclo de vida ágil procuram evitar isso, limitando o esforço de Engenharia de Requisitos realizado em um único requisito ao mínimo necessário para entender os requisitos em um nível de detalhamento exigido naquele momento. Se o requisito estiver sendo implementado hoje, o nível de detalhamento será alto. Se, por outro lado, a previsão de implementação do requisito é dentro de 3 meses, o nível mais abstrato de detalhes, ou seja, apenas o suficiente para entender seus atributos centrais, pode ser suficiente. As atividades ágeis de planejamento do *release* e a elaboração de *roadmaps*, muitas vezes baseados em *backlogs*, são responsáveis pela organização e direcionamento do esforço investido na Engenharia de Requisitos.

3.4 Gerenciar a incerteza através do *feedback* e do aprendizado

Por exemplo: Vamos supor que uma empresa decide oferecer serviços de localização pela nuvem, acessíveis para os usuários através de um conjunto de aplicativos em qualquer dispositivo móvel.

- A abordagem clássica é sequencial: pesquisa de mercado, segmentação de clientes, definição de funcionalidades do serviço, desenvolvimento das funcionalidades, início da campanha de marketing, lançamento. Mas a abordagem provavelmente irá fracassar, pois o mundo se transformou nesse meio tempo, assim como também mudaram os segmentos de clientes e as funcionalidades exigidas.
- Um modelo de ciclo de vida ágil é iterativo: ter uma visão, dar início ao marketing, desenvolver o produto viável mínimo, lançar no mercado o mais rapidamente possível, obter *feedback*, detectar o próximo conjunto mínimo de funcionalidades exigidas, mais uma vez lançar muito rapidamente, obter *feedback,* e assim por diante.

A diferença entre as duas abordagens é o fato que um grupo limitado de pessoas não é capaz de representar ou antecipar a demanda de um público heterogêneo de usuários. Falando em termos de habilidades cognitivas, um grupo fixo de pessoas apenas tem uma capacidade limitada para prever desenvolvimentos futuros (veja, por exemplo, [3]). A única maneira de identificar a demanda é obter *feedback* de verdade, seja ele fornecido diretamente pelos usuários, ou através de métodos de análise embutidos no serviço, que medem o comportamento do usuário. O *feedback* é um tipo de comunicação com a multidão. Fazendo um paralelo com a Engenharia de Requisitos, essa é uma atividade de elicitação de requisitos. O mecanismo de avaliação do aplicativo em uma loja de aplicativos como o iTunes é, na verdade, um mecanismo de *feedback* e uma forma transparente de análise.

O *feedback* qualificado da multidão apoia o processo de aprendizagem organizacional. Os resultados da aprendizagem são os seguintes:

- Recebemos *feedback* real de clientes reais e não as nossas próprias (e potencialmente falsas) crenças sobre o que o cliente possivelmente pensa
- Identificamos os segmentos de clientes com mais precisão
- Entendemos as necessidades do segmento de clientes com mais precisão
- Podemos aprender a prever o futuro até certo ponto: podemos antecipar a próxima inovação incremental (funcionalidades) que irá melhorar o nosso produto

Essa abordagem pressupõe um processo de desenvolvimento iterativo e incremental. O tempo de ciclo entre as etapas de fornecimento de um produto depende do mercado. Nos mercados em transformação constante, esse intervalo pode ser de um dia. Em outros mercados, quatro implementações por ano podem ser suficientes. O fato é que o tempo de ciclo está se tornando cada vez mais curto em todos os mercados. Se hoje o ciclo é de seis meses, amanhã ele poderá ser de seis semanas.

4 A Engenharia de Requisitos é Independente do Modelo de Processo

A constatação fundamental: não importa se o projeto para desenvolver um serviço na nuvem utiliza um processo em cascata ou um processo ágil, a Engenharia de Requisitos é igualmente importante em qualquer um dos casos. Porém, no exemplo que mencionamos acima, o desenvolvimento de um serviço de localização em nuvem, o processo ágil muito provavelmente vai gerar um resultado que melhor atende as necessidades do usuário, e sendo assim, os objetivos da organização desenvolvendo o serviço.

4.1 A aplicação da disciplina é diferente

Em perspectiva global, as atividades de Engenharia de Requisitos em projetos cascata ou em modelos de ciclo de vida ágil são comparáveis. Para demonstrar isso, vamos analisar as atividades centrais, conforme definidas no *Syllabus* IREB [4]:

- ***Elicitação***: *Durante a elicitação de requisitos, diversas técnicas são utilizadas para obter os requisitos de stakeholders e outras fontes, bem como para definir os requisitos em maior detalhe* [4].

  Tanto projetos em cascata quanto projetos ágeis realizam essa atividade. Os métodos e técnicas utilizados são idênticos. Exemplos de tais métodos e técnicas incluem: entrevistas, pesquisas de campo (*apprenticing*), elaboração de *storyboards*. O responsável pela elicitação exerce o papel de engenheiro de requisitos. Os projetos ágeis realizam essa atividade de forma contínua. Em projetos em cascata, ela é realizada principalmente durante uma fase de análise de requisitos no início de um projeto.

- ***Documentação***: *Durante a documentação de requisitos, os requisitos elicitados são descritos de forma adequada. Diferentes técnicas são utilizadas para documentar os requisitos, seja por meio de linguagem natural ou modelos conceituais* [4].

  Essa atividade é a mesma, tanto em projetos em cascata quanto em projetos ágeis. Os métodos e técnicas para a criação da documentação são idênticos ou semelhantes. Os critérios de qualidade para requisitos no *Syllabus* IREB podem ser comparados aos princípios SMART e INVEST das histórias de usuários [7]. No *Syllabus* IREB, épicos e histórias de usuários encontram-se na categoria de cenários para documentação de requisitos em linguagem natural.

  O responsável pela elaboração de épicos e histórias de usuário desempenha o papel de engenheiro de requisitos. Em um modelo de ciclo de vida ágil, a ordem das atividades é completamente diferente da documentação de requisitos em um modelo de ciclo de vida em cascata. Na "Fase Conceitual", os projetos em cascata decompõem todos os requisitos relevantes para um nível maior de detalhamento (granularidade fina). Os projetos ágeis começam com uma visão consensual representada por um pequeno conjunto de requisitos globais. A equipe somente decompõe um requisito no momento em que ele vai ser desenvolvido. Os requisitos globais, muitas vezes chamados de "características do produto" em projetos ágeis [8], são decompostos em épicos e histórias de usuários. A especificação de requisitos em um projeto ágil é um documento vivo enquanto o desenvolvimento estiver ativo. Comparar o esforço investido na especificação de requisitos nos dois modelos de ciclo de vida seria uma investigação interessante. Provavelmente os projetos ágeis investem mais esforço na Engenharia de Requisitos do que os projetos em cascata, pois sabem que o esforço resulta em alto valor agregado para o projeto [1].

- ***Validação e Negociação****: Para assegurar que os critérios de qualidade previamente definidos sejam cumpridos, os requisitos documentados devem ser validados e verificados em um estágio inicial. Além disso, o acordo dos stakeholders é um pré-requisito para a aceitação do novo produto / sistema, portanto, os requisitos precisam ser negociados* [4].

  Em projetos em cascata, essa atividade se limita às fases relacionadas aos requisitos. Os requisitos são validados e negociados com base em documentos (por exemplo, a especificação de requisitos). Uma vez concluídas essas atividades, um erro ou um conflito entre requisitos inicia um processo de mudança que reativa essas atividades para corrigir o erro ou resolver o conflito. No modelo de ciclo de vida ágil, a validação e a negociação de requisitos estendem-se desde a primeira ideia até a entrega final. A validação e negociação não são realizadas apenas em requisitos documentados. Em projetos ágeis, esse é um princípio intrínseco, implementado por um conjunto de meios, entre outros: *feedback* rápido através de iterações curtas, a definição de pronto (*ready*); critérios de aceitação; definição de concluído (*done*); design orientado a testes, sessões de revisão com os *stakeholders* e muitos mais. O IREB codifica essa abordagem como o quinto princípio da validação: a validação baseada na construção de artefatos de desenvolvimento [4].

- ***Gerenciamento***: *O gerenciamento de requisitos é independente de todas as outras atividades e compreende quaisquer medidas necessárias para estruturar requisitos, para prepará-los de modo que possam ser usados por pessoas desempenhando diferentes papéis, para manter a consistência entre os requisitos após suas alterações, bem como para assegurar sua implementação* [4].

  Essa atividade é executada tanto em projetos em cascata, quanto em projetos ágeis. Nos projetos em cascata, essa atividade é geralmente atribuída a uma função (gerente de projetos, engenheiro de requisitos, ...) desempenhada pelo responsável pela gestão dos documentos de requisitos. Em projetos ágeis, o gerenciamento de requisitos é incorporado em várias atividades e distribuído entre os membros da equipe. Uma equipe realizando o refinamento do *backlog* em uma reunião com os *stakeholders* está gerenciando os requisitos ao decompor os requisitos. A priorização é uma atividade importante e requer ações de gerenciamento de requisitos. Na metodologia Scrum, o *Product Owner* é responsável por determinar prioridades entre os requisitos de diferentes fontes, tais como os *stakeholders*, as organizações de apoio ou a arquitetura do sistema. Outro exemplo de gerenciamento de requisitos é a referência a histórias de usuários no código fonte (o código de produção e o código do teste de aceitação) para mostrar qual parte do software implementa uma história de usuário, ou seja, a rastreabilidade ou o acompanhamento do progresso através da atualização do gráfico de *burn-up*.

De modo geral, as práticas de Engenharia de Requisitos em modelos em cascata e modelos ágeis são comparáveis. Quando entramos nos detalhes sobre como a Engenharia de Requisitos é aplicada, os modelos são muito diferentes. Os dois modelos de ciclo de vida são relevantes, porém para diferentes cenários. Para o desenvolvimento de um serviço em nuvem, como mencionado acima, o modelo de ciclo de vida ágil muito provavelmente é a melhor escolha. Projetos como a transferência de um centro de dados, ou a construção de um túnel ferroviário, no entanto, talvez sempre exigem um modelo de ciclo de vida sequencial em cascata.

Como reflexão sobre a discussão até agora, as diferenças fundamentais na aplicação da Engenharia de Requisitos entre os modelos em cascata e os modelos de ciclo de vida ágil são:

- A ordem das atividades de Engenharia de Requisitos
- A atribuição do papel de "engenheiro de requisitos" aos membros de uma equipe multifuncional, com uma pessoa atuando em mais de uma função
- Um foco maior na colaboração e na comunicação contínua entre os membros das equipes ágeis e os *stakeholders* e usuários, com o objetivo de receber *feedback* sobre o que foi desenvolvido até agora e o que está em desenvolvimento.

Assim, em termos de Engenharia de Requisitos, a principal diferença entre o modelo em cascata e o modelo de ciclo de vida ágil é a aplicação da disciplina e não seu contéudo.

4.2 A Engenharia de Requisitos em projetos ágeis complexos e extensos

A unidade típica para estruturar os projetos ágeis é o *sprint*, geralmente definido em termos de semanas, por exemplo, 2 a 4 semanas. A ideia e o conceito de uma equipe ágil, tal como definida na metodologia Scrum, trabalhando em *sprints* para implementar os itens de um *backlog* é suficiente para pequenos projetos. Se um projeto requer mais trabalho a ser feito, por duas, três ou dez equipes em paralelo, o conceito de Scrum precisa crescer em escala. Além da metodologia Scrum, há várias abordagens no mercado que lidam com essa questão. Uma, por exemplo, é a estrutura SAFe (*Scaled Agile Framework*) [8], utilizada para gerenciar roteiros de produtos mais longos, projetos com centenas de desenvolvedores e organizações inteiras, do marketing e vendas até entrega e operações. Essencialmente, essas abordagens ampliam a escala de conceitos como cadências, ritmos e cerimônias do nível de equipe para o nível da organização e aumentam o nível de abstração dos artefatos. É interessante observar as consequências sobre a Engenharia de Requisitos [8].

- Histórias de usuários. Planejamento de curto prazo de *sprint* (no máximo três *sprints* para a frente): a equipe planeja o próximo *sprint* com base no *backlog* do produto / nível de detalhe que pode ser implementado pela equipe.
- Característica do produto (épicos). Planejamento de médio prazo (3 a 6 meses): características mais gerais do produto (granularidade grossa). Reflete o planejamento de *release* do produto. A decomposição das características (épicos) exige um esforço adicional, o chamado refinamento do *backlog*.
- Objetivos e metas. Planejamento de longo prazo (3 meses a 1 ano): descreve as aplicações do produto e as oportunidades da empresa (*business cases*). Reflete as atividades de elaboração de roteiros (*roadmaps*).

Da perspectiva da Engenharia de Requisitos, essa abordagem utiliza níveis de abstração previamente definidos para estruturar o desenvolvimento e a discussão de conceitos de especificação. A aplicação de níveis de abstração não é de forma alguma limitada a projetos ágeis. Níveis de abstração são um conceito conhecido e bem estabelecido para decompor informações. Alistair Cockburn, por exemplo, define vários níveis de abstração para casos de uso (nível nuvem, nível pipa, nível do mar ...) [9]. Outro exemplo é o modelo de abstração de requisitos [10]. Esses conceitos são válidos e usados tanto em projetos em cascata quanto em projetos ágeis. Existem até mesmo mapeamentos de nomes e conceitos entre esses níveis de abstração de requisitos, comparando seu uso em projetos clássicos em cascata e projetos ágeis [14].

Podemos portanto afirmar que ampliar a escala da agilidade do nível de equipe para o nível da organização envolve a reutilização de conceitos da Engenharia de Requisitos sob novos nomes, já bem estabelecidos no mundo ágil.

4.3 A evolução dos formatos de documentação da Engenharia de Requisitos

O que não difere muito – surpreendentemente – são os formatos de documentação utilizados na Engenharia de Requisitos em ambientes cascata e ágeis. Por formato de documentação, entendemos o tipo de artefato utilizado para documentar requisitos. Por exemplo, uma história de usuário é um formato de documentação que reúne um conjunto de informações em uma combinação muito específica. Qual mídia usar para manter um documento em um formato de documentação específico é um aspecto de menor importância. Uma equipe ágil pode decidir prender cartões sobre um quadro ou usar uma ferramenta de software como *Greenhopper* para preservar histórias de usuários de forma permanente. Seja qual for a mídia, do ponto de vista da Engenharia de Requisitos, o formato da documentação continua sendo uma "história de usuário".

É interessante observar que formatos de documentação de aparência tão radicalmente diferentes e supostamente inventados pelos métodos ágeis, como, por exemplo, "as histórias de usuários", são na realidade uma evolução gradual e contínua de conceitos conhecidos da Engenharia de Requisitos. Os cenários são um elemento importante na Engenharia de Requisitos há mais de vinte anos. Os cenários expressam objetivos abstratos através de exemplos concretos [15]. Informações tipicamente encontradas em cenários são as pessoas ou sistemas que interagem entre si; os objetivos que um cenário expressa; os locais virtuais ou concretos onde um cenário ocorre. Se investigamos as histórias de usuários, descobrimos as mesmas informações: uma pessoa ("como <papel/função>") atuando em um ambiente concreto (o local) com o interesse de alcançar um objetivo ("para que <razão/benefício>"). Também podemos observar uma evolução contínua na definição de atributos de qualidade para os requisitos. As definições de critérios de qualidade SMART e INVEST são siglas para especificações de atributos de qualidade para histórias de usuários.

Muitas vezes em projetos ágeis complexos e extensos como os discutidos na seção anterior, as equipes ágeis ainda trabalham com formatos de documentos que já estavam estabelecidos muito antes que os modelos de ciclo de vida ágil se tornaram tão populares. Muitos projetos ágeis de grande escala recorrem a formatos de documentação como casos de uso, modelos de dados, especificações de regras de negócio, funcionalidades e especificações de requisitos contendo requisitos de qualidade e restrições, bem como requisitos técnicos detalhados, tais como a definição de um algoritmo matemático. A estrutura SAFe [8], por exemplo, usa os seguintes formatos de documentação: épicos, visão, funcionalidades e histórias de usuários. Esses conceitos coincidem com os conceitos de objetivos e cenários na Engenharia de Requisitos [15]. O conceito Scrum, com um objetivo e histórias de usuários para cada sprint, também coincide com os conceitos de objetivos e cenários na Engenharia de Requisitos. Compreender os conceitos da Engenharia de Requisitos, portanto, aprofunda a compreensão dos mecanismos de ação que estão por trás dos conceitos ágeis equivalentes e suas evoluções. Assim, os formatos de documentação utilizados nos modelos de ciclo de vida ágil representam uma evolução dos modelos de objetivos e cenários. A diferença fundamental e inovadora é que a comunidade ágil introduziu novos nomes para evoluções concretas de conceitos, tais como as histórias de usuários como uma evolução dos casos de uso. Isso é positivo, pois novos nomes expressam uma intenção, neste caso, a intenção de usar esses formatos de documentação em modelos de ciclo de vida ágil em uma combinação específica, com propósitos específicos e movidos por uma mentalidade diferente.

Com o foco no fluxo contínuo de informações e a abordagem *lean* de detalhar requisitos apenas quando necessário, a especificação de requisitos está mais para uma documentação que vai surgindo aos poucos do que o resultado de uma atividade inicial. Assim, uma especificação ágil contém informações úteis apenas para a equipe e os *stakeholders*. A especificação muitas vezes está no nível mais abstrato de metas e objetivos. Essas informações fornecem subsídios para que os *stakeholders* e a equipe possam alinhar o desenvolvimento de médio prazo com a direção estratégica, tomar decisões fundamentais sobre o roteiro de desenvolvimento e o planejamento do *release* do sistema e documentar importantes conhecimentos básicos (como algoritmos) que de outra forma poderiam se perder ao longo do tempo. Essa documentação existe sob forma de artefatos independentes de um backlog.

No conceito mais abstrato de documentação, os testes de aceitação implementados e automatizados também fazem parte da especificação de requisitos. Nas especificações de requisitos de estilo clássico, sempre havia um capítulo ou documento "testes de aceitação". Hoje, uma especificação executável ou uma série de testes de aceitação podem substituir esse (capítulo do) documento. No entanto, é importante observar que a capacidade de entender os requisitos do sistema e de "escrever" testes de aceitação de alta qualidade continua sendo necessária. A pessoa que escreve esses testes de aceitação está desempenhando os papéis de engenheiro de requisitos e testador.

Um ponto interessante é que a equipe cria essa documentação de forma contínua e não como um tarefa inicial. A especificação de requisitos em um ambiente ágil é um objeto vivo, representado parcialmente em diferentes *containers*, até mesmo como uma série de testes de aceitação executáveis. Essa maneira diferente de realizar atividades de Engenharia de Requisitos em um ambiente ágil resulta em uma especificação de requisitos viva, atualizada, correta e útil. Uma equipe ágil disciplinada e madura dotada de todos os recursos necessários desenvolve a documentação de requisitos do sistema enquanto o próprio sistema é desenvolvido. Esse tipo de especificação de requisitos contém apenas:

- as informações exigidas pela equipe para construir o sistema alinhado a um roteiro de produto
- as informações exigidas pelos novos membros da equipe para compartilhar o conhecimento e aumentar sua produtividade pessoal o mais rapidamente possível
- as informações exigidas pela equipe para fazer a manutenção e dar suporte para a capacidade operacional do sistema
- as informações exigidas pela organização para melhorar continuamente o produto alinhado com a estratégia organizacional, ou seja, para desenvolver um roteiro de produto e um planejamento de *release* transparente.

5 O Manifesto Ágil e a Engenharia de Requisitos

Até agora discutimos os valores da Engenharia de Requisitos a partir da perspectiva da comunidade ágil. Constatamos que o *know-how* da Engenharia de Requisitos é importante e relevante tanto para modelos de desenvolvimento em cascata quanto modelos de ciclo de vida ágil. Também constatamos que muitos elementos aparentemente inventados pela comunidade ágil são, na verdade, evoluções de conceitos já existentes. Sim, há diferenças fundamentais na maneira em que esses modelos de ciclo de vida aplicam as atividades. Estamos convencidos, no entanto, que uma melhor compreensão dos conceitos básicos da Engenharia de Requisitos aprofunda a compreensão dos mecanismos nos projetos ágeis.

Queremos agora, em retrospectiva final, comparar a Engenharia de Requisitos, conforme definida pelo IREB, com os valores da metodologia ágil. O objetivo é demonstrar que as declarações do IREB não estão em conflito com os princípios ágeis. Para tal, analisaremos se a definição IREB de Engenharia de Requisitos contradiz o Manifesto para Desenvolvimento Ágil de Software:

Engenharia de Requisitos: Uma abordagem sistemática e disciplinada para a especificação e gerenciamento de requisitos, com os seguintes objetivos: (1) (a) Identificar os requisitos relevantes, (b) alcançar um consenso entre os *stakeholders* sobre esses requisitos, (c) documentar os requisitos de acordo com determinadas normas (d) e gerenciá-los de forma sistemática, (2) (a) Compreender (b) e documentar os desejos e necessidades dos *stakeholders* (3) (a) Especificar e gerenciar os requisitos (b) para minimizar o risco de entregar um sistema que não atende os desejos e necessidades dos *stakeholders*. Glossário IREB	Estamos descobrindo maneiras melhores de desenvolver software, fazendo-o nós mesmos e ajudando outros a fazer o mesmo. Através deste trabalho passamos a valorizar: **Indivíduos e interações** mais que processos e ferramentas **Software em funcionamento** mais que documentação abrangente **Colaboração com o cliente** mais que negociação de contratos **Responder a mudanças** mais que seguir um plano Ou seja, mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda. http://manifestoagil.com.br/

Indivíduos e interações mais que processos e ferramentas

Os *stakeholders* (e os usuários são *stakeholders*) são a principal fonte de requisitos. Um objetivo central da Engenharia de Requisitos é compreender os desejos e necessidades (2a) dos *stakeholders* e para identificar requisitos (1a). Esse processo cognitivo ocorre através de intensa comunicação e interação entre os *stakeholders* e a equipe. Processos e ferramentas são importantes para lidar com conteúdos complexos e oferecem apoio para esse processo cognitivo. No entanto, processos e ferramentas não são um fim em si mesmos. Eles precisam ser introduzidos de forma racional e cuidadosa, devendo ser utilizados com atenção e pragmatismo.

Software em funcionamento mais que documentação abrangente

A Engenharia de Requisitos, a implementação e os testes dos requisitos podem ser vistos como as principais competências no desenvolvimento de software. A Engenharia de Requisitos é responsável por compreender as necessidades do usuário (3a) e para transferir essa compreensão à equipe de desenvolvimento, para que a equipe possa desenvolver o que o usuário necessita (3b). A documentação não é um fim em si mesmo. O valor central é conhecer as necessidades dos *stakeholders* e do usuário (1a). A interação entre os membros da equipe para compartilhar o conhecimento é uma maneira de preservar o conhecimento. Restrições de conformidade e de regulamentação determinam padrões de especificação para domínios específicos (a indústria farmacêutica, aeroespacial, automotiva, por exemplo) que exigem uma documentação escrita em relação aos requisitos. De acordo com o IREB, um membro da equipe, desempenhando o papel de engenheiro de requisitos, é responsável por conhecer e cumprir as normas e regulamentos aplicáveis no âmbito de desenvolvimento de um produto (1c), bem como criar a documentação para atender as restrições de conformidade e de regulamentação.

Colaboração com o cliente mais que negociação de contratos

Em uma de suas publicações [2], o IREB aborda a colaboração com o cliente no capítulo sobre elicitação e consolidação de requisitos. Além disso, o IREB disponibiliza um módulo de nível avançado [11] especificamente para lidar com a elicitação e consolidação dos requisitos dos *stakeholders* e usuários. Esse módulo de nível avançado representa a importância dada pelo IREB aos valores ágeis. Os objetivos de aprendizagem apresentados nesse módulo englobam importantes princípios e elementos utilizados em projetos ágeis, tais como a comunicação direta, personagens, métodos de criatividade, a investigação contextual, histórias de usuários, o princípio CCC (cartão, comunicação, confirmação) [12] e outros. O módulo IREB de nível avançado, Elicitação e Consolidação de Requisitos, pode ser considerado um módulo central na abordagem de técnicas e métodos ágeis – assim como outros módulos IREB cobrem princípios que são válidos e importantes tanto para projetos em cascata quanto para projetos ágeis.

Responder a mudanças mais que seguir um plano

Métodos ágeis, tais como Scrum ou Kanban, apresentam a capacidade de responder a mudanças como um dos principais valores da metodologia ágil através de técnicas, cerimônias e métodos específicos. Responder a mudanças não é um atributo da Engenharia de Requisitos. A Engenharia de Requisitos oferece métodos específicos e técnicas de suporte para responder a mudanças. Os modelos de ciclo de vida ágil fornecem orientações sobre como e quando aplicar esses métodos e técnicas.

O gerenciamento de mudanças é um elemento do *know-how* IREB [2]. Essa forma de aplicar o gerenciamento de mudanças é claramente voltada para processos em cascata ou para ambientes regulamentados que exigem uma gestão de mudanças explícita, tais como o desenvolvimento de um software médico sujeito às regras da FDA. Entretanto, mesmo os capítulos sobre gerenciamento de mudanças podem ser relevantes no âmbito de projetos ágeis. Uma equipe que está desenvolvendo um sistema de software médico pode, como parte das prestações fornecidas no quadro do projeto, consultar esses capítulos para saber como executar o gerenciamento de mudanças em conformidade com os regulamentos da FDA. Nada impede que o processo de desenvolvimento do software propriamente dito seja Scrum.

6 Resumo

Este artigo apresenta a convicção do *International Requirements Engineering Board* (IREB) de que nosso *know-how* não ficou ultrapassado a partir do advento do desenvolvimento ágil. Ao contrário, estamos convencidos que as técnicas e os métodos promovidos pelo IREB são parte natural do trabalho diário em ambientes ágeis – embora provavelmente sejam aplicados de forma diferente em termos de sequência e ênfases.

Na nossa visão, a Engenharia de Requisitos é um conhecimento especializado que utiliza um conjunto de "ferramentas" apropriadas, a caixa de ferramentas da Engenharia de Requisitos. As "ferramentas" são os métodos, as técnicas e ferramentas (concretas) para executar atividades de Engenharia de Requisitos. Projetos de todos os tipos podem beneficiar-se dessa caixa de ferramentas, escolhendo o subconjunto de ferramentas que leva ao sucesso em seu contexto específico. Muitas das ferramentas são aplicáveis tanto em projetos ágeis quanto em projetos clássicos em cascata. Algumas são restritas apenas a projetos ágeis, outras somente a projetos em cascata.

Estamos convencidos também que a compreensão desses métodos e técnicas pode beneficiar todos os profissionais envolvidos no desenvolvimento de novos produtos e serviços, independente do modelo de ciclo de vida adotado por sua equipe de desenvolvimento. Um objetivo central do IREB é promover a disciplina da Engenharia de Requisitos, para que profissionais altamente qualificados possam atuar na construção e utilização de boas práticas de desenvolvimento de software, baseadas em métodos e técnicas estabelecidas de comum acordo e que agregam valor a seus projetos.

Referências

[1] Chaos Manifest 2011, Standish Group International

[2] Requirements Engineering Fundamentals; 1st Edition; Klaus Pohl, Chris Rupp; Rocky Nook Inc. (April 2011); English, 184 pages Paperback; ISBN-13: 978-1933952819

[3] Kahnemann, D.: Thinking, Fast and Slow. PENGUIN Psychology, 2011

[4] IREB Certified Professional for Requirements Engineering, Foundation Level, Syllabus Version 2.1, 1st March 2011

[5] Supporting Agile Teams of Teams via Test Driven Design; Kris Read; Thesis submitted to the faculty of graduate studies in partial fulfillment of the requirements for the degree of Master of Science; Department of Computer Science; Calgary University, Alberta; Feb 2005

[6] Creative Process and Product Life Cycle of High-Tech Firms – Creativity and Innovation, key drivers for success; Verna Lu and Cédric Marjot; Master Thesis; Baltic Business School, University of Kalmar; Sweden; June 2008

[7] Blog of Bill Wake, 2003: INVEST in Good Stories, and SMART Tasks, see: http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/

[8] The Scaled Agile Framework SAFe, http://www.scaledagileframework.com/, as well as the book: Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise; Dean Leffingwell; Addison-Wesley Professional; 1st edition (January 6, 2011); ISBN-13: 978-0321635846

[9] Writing Effective Use Cases; Alistair Cockburn; Addison-Wesley Professional; 1st edition (October 15, 2000); ISBN-13: 978-0201702255

[10] Requirements Abstraction Model; T. Gorschek and C. Wohlin; Requirements Engineering Journal, Vol. 11, No. 1, pp. 79-101, 2006

[11] Syllabus CPRE Advanced Level Requirements Elicitation and Consolidation; version 1.0; Dec 2012; see:

http://www.ireb.org/fileadmin/IREB/Lehrplaene/CPRE_Elicitation_and_Consolidation_Syllabus_Version_1.0.pdf

[12] Three C's; Ron Jeifries; Aug 30, 2001; card, conversation, confirmation; see: http://xprogramming.com/articles/expcardconversationconfirmation/

[13] Extreme Programming Explained: Embrace Change; Kent Beck; Addison Wesley; 2nd edition (November 26, 2004); ISBN-13: 978-0321278654

[14] Agile Requirements Abstraction Model — Requirements Engineering in a Scrum Environment; Richard Berntsson Svensson, Sigurdur Örn Birgisson, Christian Hedin, Björn Regnell; Paper published at the Department of Computer Science, Lund University, Sweden

[15] Requirements Engineering, Fundamentals, Principles, and Techniques; Pohl, Klaus; 2010, XVIII, 814 pages, ISBN 978-3-642-12577-5