
IREB - Certified Professional for Requirements - Engineering - RE@Agile Primer -

Syllabus e Guia de Estudo

Versão 1.1.0

1 de agosto de 2022

Termos de Uso

1. Indivíduos e provedores de treinamento podem usar este syllabus e o guia de estudos como base para seminários, desde que os direitos autorais sejam reconhecidos e incluídos nos materiais do seminário. Qualquer pessoa que utilize este syllabus e guia de estudo em publicidade precisa do consentimento por escrito do IREB para esse fim.
2. Qualquer indivíduo ou grupo de indivíduos pode usar este syllabus e guia de estudo como base para artigos, livros ou outras publicações derivadas, desde que os direitos autorais dos autores e IREB e.V. como a fonte e o proprietário deste documento sejam reconhecidos nessas publicações.

© IREB e.V.

Todos os direitos reservados Nenhuma parte desta publicação pode ser reproduzida, armazenada em um sistema de arquivamento ou transmitida de qualquer forma, ou por qualquer meio, seja eletrônico, mecânico, fotocópia, ou gravação ou qualquer outro, sem a autorização prévia e por escrito dos autores ou do IREB e.V.

Reconhecimentos

Este syllabus e guia de estudo foi escrito por: Lars Baumann, Peter Hruschka, Kim Lauenroth, Markus Meuten, Sacha Reis, Gareth Rogers, François Salazar e Thorsten Weyer.

Os comentários das análises foram fornecidos por: Bernd Aschauer, Dirk Fritsch, Rainer Grau, Andrea Hermann, Krystian Kaczor, Niko Kaintantzis, Elisabeth Larson, Ladislau Szilagyi, Daniel Tobler, Erik van Veenendaal, Arun Vetrivel, Sven van der Zee.

Revisão em português por: Ana Moreira, Bruno Favoreto, Carlos André Silva, Carlos J. Locoselli, Dagles Fernandes, George Fialkovitz, Guilherme Simões, Martin Tornquist e Stênio Viveiros.

Agradecemos a todos pelo envolvimento.

Aprovado para liberação em 2 de março de 2017 pelo Conselho do IREB, sob recomendação de Xavier Franch.

Copyright © 2016-2017 para este syllabus e guia de estudo é dos autores listados acima. Os direitos foram transferidos para o IREB International Requirements Engineering Board e.V.

Prefácio

Objetivo do documento

Este syllabus e guia de estudo define o nível básico da certificação "RE@Agile" estabelecida pelo International Requirements Engineering Board (IREB). O syllabus e o guia de estudos fornecem aos provedores de treinamento a base para a criação de seus materiais de curso. Os alunos podem usar o syllabus e guia de estudo para se preparar para o exame.

Conteúdo do syllabus e guia de estudo

O nível básico aborda as necessidades de todas as pessoas envolvidas no tópico Engenharia de Requisitos e Ágil. Isso inclui pessoas em funções como gerente de projetos ou TI, especialistas, analistas de sistemas, desenvolvedores de software, além de equipes Scrum.

Escopo

O RE@Agile é inspirado pela visão do IREB nos valores Ágeis e também pela visão do Ágil nos valores da ER. Seu conteúdo inclui a classificação e avaliação de artefatos e técnicas de Engenharia de Requisitos no contexto do Ágil, de artefatos e técnicas do Ágil no contexto da Engenharia de Requisitos e de elementos essenciais de processos no desenvolvimento Ágil de produtos. RE@Agile aponta para a motivação de usar o Ágil em um processo de desenvolvimento. Um tópico muito importante é a sinergia entre Engenharia de Requisitos e Ágil: princípios Ágeis relativos à engenharia de requisitos e mentalidade Ágil em relação aos valores principais da engenharia de requisitos.

As certificações RE@Agile do IREB visam apoiar os seguintes grupos de pessoas:

Engenheiros de requisitos que desejam se envolver no desenvolvimento Ágil e que pretendem aplicar com êxito suas técnicas neste ambiente.

Engenheiros de requisitos que desejam aplicar conceitos e técnicas estabelecidos nas abordagens Ágeis para melhorar seus processos da RE.

Profissionais Ágeis que desejam entender o valor e os benefícios da disciplina Engenharia de Requisitos em projetos Ágil.

Profissionais Ágeis que desejam melhorar o desenvolvimento Ágil usando técnicas e métodos comprovados da RE.

Pessoas de disciplinas relacionadas: gerentes de TI, testadores, desenvolvedores, arquitetos e outros representantes dos negócios envolvidos no desenvolvimento (principalmente, mas não apenas o desenvolvimento de software) - que desejam entender como combinar com êxito as abordagens de Engenharia de Requisitos e Ágil nos processos de desenvolvimento.

Nível de detalhe

O nível de detalhe deste syllabus e guia de estudo permite ensino e exames internacionalmente consistentes. Para atingir esse objetivo, o syllabus e o guia de estudo contêm:

- ▶ Objetivos educacionais gerais;
- ▶ Conteúdo com uma descrição dos objetivos educacionais;
- ▶ Referências a outras literaturas (quando necessário).

Objetivos Educacionais / Níveis de Conhecimento Cognitivo.

Cada módulo do syllabus e guia de estudo recebe um nível cognitivo. Um nível mais alto inclui os níveis mais baixos. As formulações dos objetivos educacionais são formuladas usando os verbos "saber" para o nível L1 e "dominar e usar" para o nível L2. Esses dois verbos são espaços reservados para os seguintes verbos:

- ▶ **L1 (saber):** enumerar, caracterizar, reconhecer, nomear, refletir
- ▶ **L2 (dominar e usar):** analisar, usar, executar, justificar, descrever, julgar, exibir, desenhar, desenvolver, complementar, explicar, exemplificar, elucidar, formular, identificar, interpretar, concluir, atribuir, diferenciar, comparar, compreender, sugerir, resumir



Todos os termos definidos no glossário devem ser conhecidos (L1), mesmo que não sejam mencionados explicitamente nos objetivos educacionais. O glossário está disponível para download na página inicial do IREB em <https://www.ireb.org/en/downloads/#re-agile-glossary>

Este syllabus e guia de estudo usam a abreviatura "ER" para Engenharia de Requisitos

Estrutura do syllabus e guia de estudo

O syllabus e o guia de estudo consistem em 4 capítulos principais. Um capítulo abrange uma unidade educacional (EU, Educational Unit). Cada título do capítulo principal contém o nível cognitivo do capítulo, que é o nível mais alto dos subcapítulos. Além disso, é sugerido o tempo mínimo de ensino que um curso deve investir para esse capítulo. Termos importantes no capítulo, definidos no glossário, estão listados no início do capítulo.

Exemplo: EU 2 FUNDAMENTOS DA RE@Agile (L1)

Duração: 75 min

Termos: Proprietário do Produto, Backlog do Produto, Backlog da Sprint, Épicas, Histórias de Usuários, Mapas de Histórias

Este exemplo mostra que o capítulo 2 contém objetivos educacionais no nível L1 e 75 minutos são destinados ao ensino do conteúdo deste capítulo.

Cada capítulo pode conter sub-capítulos. Seus títulos também contêm o nível cognitivo de seu conteúdo.

Os objetivos educacionais (OE) são enumerados antes do texto real. A numeração mostra a qual sub-capítulo eles pertencem.

Exemplo: OE 3.1.2

Este exemplo mostra que o objetivo educacional OE 3.1.2 é descrito no sub-capítulo 3.1.

O Exame

Este syllabus e guia de estudo são a base do exame RE@Agile Primer. Dois exames diferentes estão disponíveis:

- ▶ Exame de múltipla escolha com o certificado oficial RE@Agile Primer, semelhante aos exames de múltipla escolha do CPRE Foundation Level e Advanced Level, mas com duração de 40 minutos.
- ▶ Autoavaliação online de múltipla escolha com confirmação de participação.

Os exames podem ser realizados imediatamente após um curso de treinamento, mas também independentemente dos cursos (p. ex., em um centro de exames). Uma lista de provedores de exames reconhecidos pode ser encontrada na página inicial do IREB

<https://www.ireb.org/exams/bodies>

A autoavaliação está disponível na página inicial do IREB: <http://www.ireb.org>



Uma pergunta no exame pode abranger o material de vários capítulos do syllabus e do guia de estudo. Todos os capítulos (UE 1 a UE 4) do syllabus e guia de estudo podem ser examinados.

Histórico de versões

| Versão | Data | Comentário |
|---------------|------------------|---|
| 1.1.0 | 1 de agosto 2022 | Versão Inicial baseada no Syllabus original do IREB em inglês 1.1.0 |

Conteúdo

| | |
|---|-----------|
| Reconhecimentos..... | 2 |
| Prefácio | 2 |
| Histórico de versões | 5 |
| A visão do RE@Agile..... | 9 |
| EU 1 Motivação e Mentalidade (L1) | 13 |
| EU 1.1 Motivação para usar o Ágil (L1) | 13 |
| EU 1.2 Mentalidade e Valores na ER e Ágil (L1)..... | 14 |
| EU 1.3 Construindo uma ponte dos Princípios da ER e Ágil para o RE@Agile (L1)..... | 16 |
| EU 1.4 Benefícios, Equívocos e Armadilhas para o uso da RE@Agile (L1)..... | 18 |
| EU 1.4.1 Benefícios da RE@Agile..... | 18 |
| EU 1.4.2 Equívocos da RE@Agile..... | 19 |
| EU 1.4.3 Armadilhas da RE@Agile | 20 |
| EU 1.5 RE@Agile e o Trabalho Conceitual (L1)..... | 22 |
| EU 2 FUNDAMENTOS DA RE@AGILE (N1) | 25 |
| EU 2.1 Métodos Ágeis (Uma visão geral) (L1) | 25 |
| EU 2.2 Scrum (mais que boas práticas) como um Exemplo (L1)..... | 26 |
| EU 2.3 Diferenças e Semelhanças entre Engenheiros de Requisitos e Proprietário do Produto (L1)..... | 28 |
| EU 2.4 Engenharia de Requisitos como Processo Contínuo (L1) | 29 |
| EU 2.5 Desenvolvimento Orientado a Valor (L1) | 30 |
| EU 2.6 Simplicidade como Conceito Essencial (L1)..... | 30 |
| EU 2.7 Inspeccionar e Adaptar (L1) | 31 |
| EU 3 ARTEFATOS E TÉCNICAS NA RE@AGILE (L1)..... | 32 |
| EU 3.1 Artefatos na RE@AGILE (L1)..... | 32 |
| EU 3.1.1 Documentos de Especificação vs. Backlog do Produto | 32 |
| EU 3.1.2 Visão e Objetivos..... | 33 |
| EU 3.1.3 Modelo de Contexto..... | 34 |
| EU 3.1.4 Requisitos | 35 |
| EU 3.1.5 Granularidade de Requisitos | 35 |

| | |
|---|-----------|
| EU 3.1.6 Modelos gráficos e descrições textuais | 38 |
| EU 3.1.7 Definição de Termos, Glossários e Modelos de Informação | 38 |
| EU 3.1.8 Requisitos de Qualidade e Restrições | 39 |
| EU 3.1.9 Critérios de Aceite e de Sucesso | 40 |
| EU 3.1.10 Definição de Pronto e Concluído | 40 |
| EU 3.1.11 Protótipo vs. Incremento | 40 |
| EU 3.1.12 Relação de Artefatos | 41 |
| EU 3.2 Técnicas no RE@Agile (L1) | 42 |
| EU 3.2.1 Elicitação de Requisitos | 42 |
| EU 3.2.2 Documentação de Requisitos | 43 |
| EU 3.2.3 Validação e Negociação de Requisitos | 45 |
| EU 3.2.4 Gerenciamento de Requisitos | 45 |
| EU 4 Aspectos Organizacionais do RE@Agile (L1) | 48 |
| EU 4.1 Influência das Organizações no RE@AGILE (L1) | 48 |
| EU 4.2 Desenvolvimento Ágil em um ambiente não-Ágil (L1) | 49 |
| EU 4.2.1 Interação com stakeholders fora da organização de TI | 49 |
| EU 4.2.2 Organização do produto vs. organização do projeto | 50 |
| EU 4.2.3 O papel do gerenciamento em um contexto Ágil | 51 |
| EU 4.3 Manuseio de problemas complexos por escala (L1) | 52 |
| EU 4.3.1 Motivação para economias escalares | 52 |
| EU 4.3.2 Abordagens para a organização de equipes | 53 |
| EU 4.3.3 Abordagens para organizar a comunicação | 54 |
| EU 4.3.4 Exemplo de Frameworks para economia de escalada RE@Agile | 55 |
| EU 4.3.5 Impactos da escalabilidade no RE@Agile | 55 |
| EU 4.4 Equilibrando a ER antecipada e contínua no contexto de economia da escalabilidade (L1) | 56 |
| EU 4.4.1 Definição dos Requisitos Iniciais | 57 |
| EU 4.4.2 Nível de detalhe para itens do Backlog | 57 |
| EU 4.4.3 Validade dos itens de Backlog | 58 |
| EU 4.4.4 Feedback e atualização do Backlog | 58 |
| EU 4.4.5 Cronograma do Ciclo de Desenvolvimento | 59 |

| | | |
|------|---|----|
| EU 5 | DEFINIÇÕES DOS TERMOS, Glossário (L2) | 60 |
| EU 6 | Referências..... | 61 |

A visão do RE@Agile

Motivação e Antecedentes

A qualidade dos requisitos determina o sucesso ou o fracasso de todo o processo de desenvolvimento do produto, independentemente da metodologia de desenvolvimento aplicada. Ao contrário da crença popular, técnicas e métodos da disciplina Engenharia de Requisitos são agnósticos ao seu uso em metodologias específicas de desenvolvimento (como o modelo em cascata ou Scrum). No entanto, a Engenharia de Requisitos é mais comumente vista como uma disciplina de desenvolvimento não-Ágil, levando ao equívoco de que o corpo de conhecimento da Engenharia de Requisitos não tem relevância para o sucesso dos processos de desenvolvimento Ágil.

Em muitos casos, as abordagens de Engenharia de Requisitos e Ágil são consideradas separadamente, e não juntas. Enquanto nos processos de desenvolvimento convencionais, a Engenharia de Requisitos é estabelecida com papéis dedicados em disciplinas separadas no ciclo de vida de um sistema, no desenvolvimento Ágil, a importância da Engenharia de Requisitos é frequentemente subestimada.

As abordagens Ágeis são baseadas em comunicação direta, simplicidade de soluções e feedback. Um de seus principais valores é a rápida resposta às mudanças. Assim, alterações nos requisitos e suas prioridades representam um conceito inerente a todas as abordagens Ágeis. De fato, dar importância apropriada à competência de Engenharia de Requisitos nos processos de desenvolvimento Ágil pode alavancar o sucesso de projetos Ágeis, enquanto aumenta de maneira sustentável a qualidade dos sistemas e produtos desenvolvidos. Por outro lado, a prática de Engenharia de Requisitos pode se beneficiar significativamente de alguns princípios e técnicas Ágeis muito úteis, independentemente da metodologia de desenvolvimento específica aplicada.

Atualmente, em muitos casos, as pessoas são especialistas em Engenharia de Requisitos ou na aplicação de certas abordagens Ágeis. Como consequência, as pessoas de ambos os lados precisam encontrar seu próprio caminho para aproveitar os benefícios do uso de princípios e técnicas do outro campo de competência. Atualmente, as certificações com foco na integração de ambos os campos de competências não estão disponíveis, nem na comunidade de Engenharia de Requisitos nem na comunidade Ágil. Portanto, é altamente promissor construir uma ponte amplamente aceita entre as abordagens de Engenharia de Requisitos e Ágil e, portanto, também entre engenheiros de requisitos e especialistas em Ágil, para que ambos possam se comunicar de maneira eficiente e eficaz.

A resposta do IREB a essa demanda é a Certificação RE@Agile.

Sobre RE@Agile

Essa ponte deve ser construída a partir de duas direções diferentes: de um lado, a comunidade de Engenharia de Requisitos precisa entender como aplicar com êxito suas várias técnicas e métodos nos processos de desenvolvimento Ágil, de outro, como aplicar técnicas específicas das abordagens Ágeis, para melhorar a prática de Engenharia de Requisitos. Por outro lado, como as abordagens Ágeis visam fornecer software de valor o mais cedo possível, os profissionais Ágeis precisam entender como alavancar esta ação aplicando conceitos e técnicas comprovados da disciplina de Engenharia de Requisitos.

Os fundamentos do RE@Agile são a visão da ER sobre os valores do Ágil, bem como as visões do Ágil sobre os valores da Engenharia de Requisitos. O conteúdo inclui a classificação, avaliação e aplicação de artefatos e técnicas de Engenharia de Requisitos no contexto de abordagens Ágeis, bem como o uso de artefatos, técnicas e elementos de processos essenciais nos processos de Engenharia de Requisitos.

Os princípios fundamentais do RE@Agile são:

As abordagens de Engenharia de Requisitos e Ágil podem se alavancar

O RE@Agile analisa possíveis benefícios e armadilhas das técnicas de Engenharia de Requisitos e Ágil. Para esse fim, o RE@Agile aborda o uso de artefatos e técnicas da disciplina Engenharia de Requisitos nos processos Ágeis, bem como o uso de artefatos, funções e técnicas das abordagens Ágeis nos processos de Engenharia de Requisitos no contexto de diferentes metodologias de desenvolvimento.

Processos leves e altamente adaptáveis

Com base na filosofia do RE@Agile, a diferenciação entre processos de desenvolvimento preditivo e adaptativo é de vital importância. O RE@Agile propõe a ideia de uma abordagem leve e altamente adaptável para executar as atividades de Engenharia de Requisitos no desenvolvimento Ágil. No RE@Agile, a Engenharia de Requisitos é uma disciplina central, e não uma única etapa do processo: um processo contínuo que deve ser executado sistematicamente e que exige um alto nível de habilidade e experiência.

Estreita colaboração dentro da equipe e com os principais stakeholders, e requisitos just-in-time

A comunicação frequente e a estreita colaboração entre todos os membros da equipe e principais stakeholders são de particular importância para o sucesso dos processos de desenvolvimento Ágil. No RE@Agile, a equipe, juntamente com os principais stakeholders, elicita, analisa, refina e documenta os requisitos de uma maneira altamente interativa. O RE@Agile apoia os profissionais na seleção das atividades certas no momento certo para garantir requisitos de alta qualidade antes de serem implementados.

Elicitação, análise, especificação e refinamento de requisitos situacionais e seletivos

O RE@Agile é baseado na ideia de que nem todo requisito precisa ser especificado com precisão e com baixo nível de detalhamento antes do início da implementação do sistema. Em vez disso, apenas os requisitos que são complexos (isto é, não são compreensíveis para os stakeholders ou equipe de desenvolvimento) ou críticos (isto é, não podem ter mal-entendidos) são refinados, especificados e detalhados. O processo geral baseia-se na filosofia compartilhada de que alterações nos requisitos funcionais são bem-vindas e fáceis de acomodar.

Evite atividades e funcionalidades menos relevantes e garanta o produto mínimo viável

Um dos princípios do Ágil é a "Simplicidade". De acordo com esse princípio, o primeiro estágio do desenvolvimento do sistema ou produto nos processos Ágil é frequentemente o MVP (Produto Mínimo Viável). O MVP é um sistema distinto e implementável que oferece apenas um conjunto básico de funcionalidades/características que fornece valor comercial suficiente para que os usuários finais permitam um aprendizado válido. O escopo mínimo do MVP permite a eliminação de desperdícios durante o desenvolvimento e oferece uma oportunidade para feedback rápido dos clientes. Um dos próximos estágios do produto geralmente é o MMP (Produto Mínimo Comercializável) - um produto com o menor conjunto de funcionalidades que atende às necessidades dos usuários e, portanto, tem valor de mercado. O RE@Agile fornece respostas para duas perguntas muito importantes, que são bastante relevantes mesmo em processos de desenvolvimento não-Ágil: "Como simplificar o processo de gerenciamento de liberação e definição de produto?" e "Como definir o MVP ou MMP com base nos requisitos?"

Sobre a certificação IREB RE@Agile:

O IREB RE@Agile fornece duas Certificações diferentes e uma autoavaliação abordando diferentes níveis de habilidades:

Autoavaliação do RE@Agile (aberta a todos os participantes)

A autoavaliação do RE@Agile é dirigida a pessoas que desejam mostrar seu conhecimento, mas não estão interessadas no certificado Primer ou Advanced Level ou não cumprem os pré-requisitos.

Uma pessoa com o resultado da autoavaliação do RE@Agile :

- ▶ demonstrou conhecimento fundamental relevante e medido (Escala 0 a 100%) da terminologia das abordagens da Engenharia de Requisitos e Ágil.

Certificado RE@Agile Primer (recomendado ter CPRE FL e conhecimento em Ágil)

O certificado RE@Agile Primer é para profissionais de disciplinas relacionadas: gerentes de projeto, analistas de negócios, arquitetos, desenvolvedores, testadores e também pessoas de negócios. Este certificado concentra-se na comunicação entre especialistas em Engenharia de Requisitos e Ágil, bem como na compreensão dos termos de ambas as áreas. Os titulares de certificados podem conversar com especialistas em Ágil sobre Engenharia de Requisitos e com especialistas em Engenharia de Requisitos sobre abordagens e desenvolvimento Ágil.

Uma pessoa com o certificado RE@Agile Primer:

- ▶ está familiarizado com a terminologia relevante das abordagens de Engenharia de Requisitos e Ágil;
- ▶ entende o papel e a importância da Engenharia de Requisitos nos processos Ágeis, bem como o valor do Ágil na Engenharia de Requisitos

Certificado RE@Agile Advanced Level (o pré-requisito é o certificado CPRE FL. No entanto, o Certificado Primer ou outro certificado Ágil reconhecido também é altamente recomendado.

[Nota: Certificados Ágeis reconhecidos incluem aqueles fornecidos pela Scrum Alliance e pela Scrum Org.]

O módulo RE@Agile Advanced Level é para engenheiros de requisitos e profissionais Ágeis. O módulo AL RE@Agile se concentra no entendimento e aplicação de métodos e técnicas da disciplina Engenharia de Requisitos nos processos de desenvolvimento Ágil, bem como no entendimento e aplicação de conceitos, técnicas e elementos essenciais do processo das abordagens Ágeis nos processos de Engenharia de Requisitos. Os titulares de certificados com conhecimento em Engenharia de Requisitos podem trabalhar em ambientes Ágeis, enquanto os profissionais Ágeis podem tornar os instrumentos da Engenharia de Requisitos acessíveis em projetos Ágeis.

Um detentor de certificado do RE@Agile Advanced Level:

- ▶ está familiarizado com a terminologia adicional da Engenharia de Requisitos em um contexto Ágil, conforme coberto pelo módulo AL RE@Agile;
- ▶ pode planejar, implementar e executar com êxito técnicas e métodos de Engenharia de Requisitos em projetos Ágeis;
- ▶ pode planejar, implementar e executar com êxito técnicas e métodos das abordagens Ágeis nos processos de Engenharia de Requisitos.

EU 1 Motivação e Mentalidade (L1)

Duração: 90 min

Termos: Valores, Manifesto Ágil, Práticas, Atividades, Sprint, Ágil

Objetivos educacionais:

- OE 1.1.1 Conhecer a motivação para usar métodos Ágeis;
- OE 1.2.1 Conhecer os valores de ER do IREB;
- OE 1.2.2 Conhecer os valores essenciais do manifesto Ágil e os princípios dele derivados;
- OE 1.3.1 Conhecer a diferença entre princípio, prática e atividade;
- OE 1.3.2 Conhecer as diferenças entre as mentalidades Ágil e ER;
- OE 1.3.3 Conhecer a sinergia das mentalidades e valores em relação ao RE@Agile ;
- OE 1.3.4 Saber o que "Documentação" significa em um Contexto Ágil (em alinhamento com o Manifesto Ágil);
- OE 1.4.1 Conhecer os benefícios, as armadilhas e os conceitos errados para o uso do RE@Agile;
- OE 1.4.2 Conhecer exemplos equivocados;
- OE 1.5.1 Saber que valores Ágeis podem ser transferidos para o trabalho conceitual;
- OE 1.5.2 Conhecer abordagens exemplares que permitem o Ágil no trabalho conceitual

EU 1.1 Motivação para usar o Ágil (L1)

Vários estudos (ver [MeMi2015]) mostram que o negócio de TI como um todo está passando por uma mudança essencial: ela está se tornando um fator importante em várias áreas de negócios (p. ex., comércio eletrônico, mídia social) e domínios técnicos (p. ex., automotivo ou indústria aeronáutica). Conseqüentemente, os sistemas e produtos nas empresas orientadas por TI têm que passar por uma constante adaptação para acompanhar as mudanças das necessidades dos clientes ou do mercado. Assim que ocorre uma mudança no mercado, os sistemas precisam ser adaptados de acordo com estas mudanças.

Os métodos de desenvolvimento existentes que se concentram na previsibilidade e estabilidade a longo prazo não foram desenvolvidos para tais circunstâncias e muitas vezes falham em negócios ou situações de projetos em rápida mudança. Métodos Ágeis, impulsionados pelo manifesto Ágil (ver EU 1.2), surgiram para preencher essa lacuna. O Ágil em si é um termo difícil e pode ser definido da seguinte forma (ver [ShYo2006]):

Um movimento rápido do corpo inteiro com mudança de velocidade ou direção em resposta a um estímulo sem perder o controle.

Essa definição não se origina da engenharia de software, mas no esporte, refletindo a motivação essencial para o uso de um método Ágil: se a situação do negócio ou do projeto exigir mudanças rápidas e controladas, esse método é o adequado. Um método Ágil é, obviamente, mais do que apenas desenvolvimento rápido (ver EU 1.2), mas, em essência, todos os princípios focam na entrega frequente de uma dada qualidade.

É importante reconhecer que os métodos Ágeis não se constituem, em si próprios, um fim [Meyer2014]. Uma organização deve ser capaz de selecionar a abordagem de desenvolvimento adequada que atenda às necessidades de seu mercado, clientes e organização. O Gartner ainda afirma que a capacidade de desenvolver TI com a abordagem correta é o principal fator de sucesso para os negócios digitais [MeMi2015].

EU 1.2 Mentalidade e Valores na ER e Ágil (L1)

A mentalidade e os valores da ER estão determinados na definição de Engenharia de Requisitos do IREB (ver [IREB2015]).

Uma abordagem sistemática e disciplinada para a especificação e gestão de requisitos com os seguintes objetivos:

1. Conhecer os requisitos relevantes, obter consenso entre os stakeholders sobre esses requisitos, documentá-los de acordo com os padrões definidos e gerenciá-los sistematicamente;
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 atenda aos desejos e necessidades dos stakeholders.

Na ER, falamos de um sistema em vez de um software ou um produto. O uso do termo sistema não se destina a excluir produtos, outros tipos de software ou até mesmo outras coisas (p. ex., processos de negócios ou hardware). A ER prefere o termo sistema porque ele enfatiza o fato de que um sistema é um grupo de partes ou elementos que trabalham juntos em um ambiente. A ER chama este ambiente de contexto do sistema. No syllabus e guia de estudo, sempre usaremos o termo sistema, que inclui produtos e quaisquer outros tipos de elementos relacionados ao software.

O IREB FL [IREB2015] também define um conjunto de quatro atividades principais de ER: elicitação, documentação, validação/negociação e gerenciamento de requisitos. Essa lista de atividades não denota um conjunto específico de etapas ou a sequência na qual essas atividades são executadas. Um valor central do IREB FL é que a ER é uma abordagem agnóstica do processo: ela fornece um rico corpo de conhecimento que consiste em vários métodos e uma rica coleção de técnicas que podem ser aplicadas em qualquer abordagem de desenvolvimento. Ela não recomenda nem especifica nenhum processo.

Este syllabus e guia de estudo usará o termo “Métodos Ágeis” para se referir ao rico conjunto de abordagens que surgiram no campo do Ágil (ver EU 2). Para distinguir métodos Ágeis de outros métodos de desenvolvimento (p. ex., orientados a planos ou estilo cascata), este syllabus e guia de estudo usa o termo “métodos não-Ágeis”. Esses dois termos justificam uma avaliação de qual deles é melhor - o IREB está convencido de que os dois tipos de métodos (Ágil e não-Ágil) têm seu valor.

A mentalidade do Ágil é definida pelo Manifesto Ágil e os doze princípios por trás dele (veja [AgileMan2001]):

Manifesto Ágil

Estamos descobrindo maneiras melhores de desenvolver software.
fazendo-o nós mesmos e ajudando outros a fazerem 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.

Princípios Ágeis

- 1) Nossa maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de software de valor.
- 2) Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos Ágeis se adequam a mudanças, para que o cliente possa obter vantagens competitivas.
- 3) Entregar software em funcionamento com frequência, na escala de semanas até meses, de preferência em períodos mais curtos.
- 4) Pessoas relacionadas aos negócios e desenvolvedores devem trabalhar em conjunto e diariamente, ao longo do projeto.
- 5) Construir projetos ao redor de indivíduos motivados. Dando a eles o ambiente e suporte necessários, e confiar que farão seu trabalho.
- 6) O método mais eficiente e eficaz de transmitir informações para e dentro de uma equipe de desenvolvimento é a conversa presencial.
- 7) Software em funcionamento é a medida primária de progresso.
- 8) Processos Ágeis promovem um desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários, devem ser capazes de manter um ritmo constante indefinidamente.
- 9) Atenção contínua à excelência técnica e bom desenho aumenta a Agilidade.

- 10) Simplicidade - a arte de maximizar a quantidade de trabalho que não precisou ser feito - é essencial.
- 11) As melhores arquiteturas, requisitos e desenhos emergem de times auto-organizáveis.
- 12) Em intervalos regulares, a equipe reflete sobre como se tornar mais efetiva, e então ajusta e otimiza seu comportamento de acordo.

Se compararmos os valores e a mentalidade da ER e Ágil, não podemos encontrar qualquer parte que contradiga os valores ou mentalidade do outro. O valor mais importante compartilhado pela ER e Ágil é tornar o usuário final do produto feliz porque a solução atende às suas necessidades ou resolve seus maiores problemas. O valor mais importante é compartilhado pela ER e Ágil e é tornar o usuário final do produto feliz porque a solução atende às suas necessidades ou resolve seus maiores problemas. No entanto, temos que reconhecer que a mentalidade e os valores da ER e Ágil são, até certo ponto, separados. A interseção da ER e Ágil é definida pelo campo da RE@Agile, que será explicado na próxima EU.

EU 1.3 Construindo uma ponte dos Princípios da ER e Ágil para o RE@Agile (L1)

Antes de nos aprofundarmos nos detalhes da RE@Agile, precisamos esclarecer algumas terminologias. Dentro da indústria de software e pesquisa, há um extenso corpo de conhecimento sobre como trabalhar e se comportar ao desenvolver software. Este conhecimento existe em vários níveis de abstração. A seguir, apresentaremos a diferença de princípios, práticas e atividades, como três níveis de abstração, para falar sobre desenvolvimento de software (ver [Meyer2014]):

- Um princípio é uma declaração prescritiva que é abstrata e falseável;
- Uma prática/técnica é uma instancia de um princípio para um determinado contexto;
- Uma atividade é uma execução real ou planejada de uma prática.

Os termos importantes para diferenciar as três definições são “prescrição”, “abstração” e “falseabilidade”. A prescrição significa que a declaração direciona a ação em vez de declarar um fato ou uma propriedade. A abstração distingue um princípio de uma prática. Por exemplo, “teste uma característica do software antes da entrega” é prescritivo e abstrato, enquanto “crie um teste unitário para cada característica/funcionalidade de software” é uma prática baseada no princípio dado. Uma atividade, para este exemplo, seria a criação de um teste unitário para a função de pesquisa de um sistema de biblioteca. A falseabilidade significa que uma pessoa com conhecimento suficiente pode discordar de um princípio. O princípio declarado acima (“teste um ...”) satisfaz este critério. Pode-se argumentar que o teste pode não ser suficiente para características críticas de segurança e, em vez disso, deveriam ser verificadas por meios matemáticos. Uma declaração que não é falseável (“esforçar-se por alta qualidade”) não deve ser considerada um princípio para orientar o comportamento.

Conhecer os princípios por trás de nossas práticas (que em si é um princípio) leva a decisões conscientes sobre nossas ações. Conhecer diferentes práticas para cumprir os princípios nos dá a capacidade de reagir de maneira diferente, dependendo da situação dada. A falseabilidade promove discussões sobre a aplicabilidade de um princípio ou prática num determinado contexto e, portanto, ajuda a decidir se um princípio deve ou não ser aplicado.

Conhecer princípios e práticas é apenas um primeiro passo: a aplicação adequada de uma prática é uma competência por si só. Por exemplo, a definição de casos de uso é uma prática amplamente conhecida para o princípio "especificar requisitos funcionais para características importantes". No entanto, escrever um caso de uso de alta qualidade é uma competência em si e conhecer os elementos de um modelo (template) de caso de uso não é suficiente para isso.

Em essência, agora definimos três níveis de competências:

1. Conhecer princípios (corresponde aos OEs do L1)
2. Conhecer práticas para cumprir os princípios dados (corresponde aos OEs do L1)
3. Dominar uma prática em um determinado contexto (capacidade de realizar uma atividade com alta qualidade) (corresponde aos OEs do L2)

Comparando a disciplina Ágil com a disciplina ER (ver EU 1.2), é possível ver porque as duas disciplinas são às vezes percebidas como em conflito: A ER se preocupa com a elicitação e documentação sistemáticas de requisitos como artefatos em si mesmos, enquanto o Ágil enfatiza a importância do software funcionando mais que uma documentação abrangente, e valoriza os indivíduos e a comunicação mais que os processos e ferramentas.

Uma implementação exagerada de ambas as mentalidades pode, na prática, levar a um conflito: uma falsa interpretação de que com a ER é possível criar um documento de requisitos completo, consistente e acordado, que pode ser implementado sem modificação adicional. Uma interpretação igualmente falsa do Ágil é que um projeto de desenvolvimento pode começar sem qualquer trabalho preparatório e ter sucesso apenas fornecendo software, em intervalos regulares, que é revisado pelos stakeholders [*Nota: os clientes são um subconjunto de stakeholders do ponto de vista do ER.*] e aprimorado com base no feedback.

É nossa afirmação que as mentalidades da ER e do Ágil não estão de fato em conflito: ambas as abordagens compartilham o mesmo objetivo da entrega de software em um nível de qualidade bem definido. Os métodos Ágeis podem fornecer software funcional de maneira eficiente e rápida (tempo de ciclo reduzido). A ER fornece as técnicas adequadas para entender os desejos e necessidades dos stakeholders para desenvolver o software correto.

A ER facilita:

- ▶ No entendimento relevante dos desejos e necessidades dos usuários para desenvolver software com valor (1º princípio Ágil)

- Nas ferramentas adequadas para reconhecer as mudanças no mercado para a vantagem competitiva dos stakeholders (2º Princípio Ágil)
- Nas ferramentas e técnicas adequadas para promover uma colaboração eficiente entre os stakeholders e os desenvolvedores (4º Princípio Ágil)
- Nas ferramentas e técnicas adequadas para apoiar a comunicação verbal (6º Princípio Ágil)
- Na compreensão relevante dos desejos e necessidades dos stakeholders para minimizar o desenvolvimento desnecessário de software (10º Princípio Ágil).

A diferença importante entre a aplicação da ER no Ágil e outros métodos de desenvolvimento é o tempo e o processo aplicados. Com este syllabus e guia de estudos, o IREB define o campo RE@Agile, que mostra como aplicar a ER no contexto de métodos Ágeis. O IREB prefere o termo RE@Agile ao termo "Engenharia de Requisitos Ágil" para deixar claro que a ER é independente do processo.

O manifesto para desenvolvimento Ágil de software enfatiza o valor de um incremento de software em funcionamento (ou produto em funcionamento) mais que uma documentação abrangente. Uma interpretação exagerada levou à falsa impressão de que os métodos Ágeis abandonaram completamente a documentação. Esta interpretação está errada: a documentação que tem uma finalidade ainda é bem-vinda e recomendada no Ágil, mas apenas aquela que suporta desenvolvimento ou faz parte do produto. Uma questão percebida no passado foi que muitos projetos criaram documentação sem um propósito claramente declarado ou valor agregado - esse tipo de documentação deve ser evitado, de acordo com os princípios.

EU 1.4 Benefícios, Equívocos e Armadilhas para o uso da RE@Agile (L1)

A RE@Agile oferece vários benefícios. No entanto, esses benefícios não são gratuitos: há equívocos e armadilhas que devem ser evitados.

EU 1.4.1 Benefícios da RE@Agile

As competências da ER & Dev na mesma equipe podem reduzir as transferências de entregáveis: A prática de equipes multifuncionais em métodos Ágeis requer que a equipe tenha todas as habilidades necessárias para o desenvolvimento de um incremento de produto, com base nos requisitos selecionados. Realizar tarefas da ER na equipe pode reduzir a necessidade de criar uma documentação abrangente dos requisitos antecipadamente, pois os membros da equipe podem explicar certos detalhes diretamente aos outros membros. O benefício dos documentos neste contexto será a documentação dos resultados das discussões e a preservação do conhecimento.

O desenvolvimento incremental permite a otimização de ideias existentes: O princípio essencial dos métodos Ágeis é o desenvolvimento incremental de software em iterações. O processo iterativo cria artefatos (p. ex., um modelo de processo de negócios, um épico, uma história de usuário, um caso de uso, um protótipo de interface de usuário, uma descrição de processo ou o software) e os aprimora em uma sequência de atividades de desenvolvimento e revisão. O benefício de tal procedimento é que a qualidade do artefato e/ou do software é continuamente aprimorada e otimizada. Além disso, incrementos menores permitem discussões mais cedo com o cliente e minimizam o risco de grandes lacunas entre as expectativas dos clientes e o desenvolvimento.

O refinamento é um princípio para amadurecer e validar requisitos: No desenvolvimento Ágil, uma prática muito boa desenvolvida foi: refinamento contínuo. Aqui são realizadas reuniões regulares de refinamento para que a equipe de desenvolvimento revise e detalhe continuamente os requisitos, tudo em estreita comunicação com os stakeholders. Além disso, a boa prática da Definição de Pronto é usada como um portal de qualidade para validar que um requisito está pronto para implementação em uma iteração subsequente.

A ER ajuda a definir um backlog inicial do produto: A ER fornece várias técnicas para obter uma compreensão adequada dos requisitos dos stakeholders para o produto desejado. A ER fornece, assim, uma compreensão mais profunda dos requisitos necessários para a definição inicial do backlog do produto. É importante reconhecer que tal atividade da ER não cria uma especificação detalhada. Em vez disso, o objetivo de tais atividades é focar em uma compreensão completa do produto, em um determinado nível de abstração (p. ex., compreender e definir os casos de uso essenciais ou épicos e histórias de usuários). Por exemplo, um complicado processo de negócios de alto nível pode ser decomposto em épicos e histórias de usuário, usando métodos bem estabelecidos da ER para criar um backlog inicial.

EU 1.4.2 Equívocos da RE@Agile

No mundo do desenvolvimento (principalmente desenvolvimento de software), existem alguns equívocos e armadilhas em relação a ER que precisam ser discutidos:

Equívoco: a ER é apenas uma análise antecipada: Muitas vezes, a ER é considerada possível apenas como uma atividade inicial. A ER, como disciplina, é de fato, independente do processo e não impõe um trabalho inicial completo. Em vez disso, as atividades da ER podem estar inseridas dentro de um método Ágil da mesma maneira que outras atividades (p. ex., codificação ou teste) são realizadas. A ER é uma atividade incorporada dentro de cada iteração.

Equívoco: Antecipar é ruim: A preparação para o desenvolvimento iterativo é uma parte essencial de qualquer empreendimento de desenvolvimento não trivial. Pensar em antecipar não implica em nenhum ciclo de vida de software específico ou nenhuma fase longa de análise ou documentação: precisamente como e quando acontece o pensamento inicial será determinado pelo contexto do projeto. Os praticantes não devem interpretar a literatura Ágil para sugerir que o pensar antecipadamente é em si próprio algo ruim; o manifesto e os princípios apresentados acima não suportam tal entendimento.

As práticas Ágeis que refletem o valor do pensar antecipadamente são o mapeamento de histórias (veja [Patt2014]), prototipagem (veja [Martin1991]) e desenvolvimento orientado a testes (TDD) (veja [Beck2003]).

Equívoco: a ER é igual a documentação: A ER é frequentemente associada apenas aos documentos que produz. No entanto, os documentos são um resultado potencial de uma atividade que cria conhecimento. Uma boa ER inclui estar ciente do fato de que mesmo o melhor documento nunca é totalmente auto-contido. Em vez disso, um documento serve para apoiar os propósitos definidos na EU 3.2: conformidade legal, preservação de informações valiosas, facilitação de comunicação e apoio a processos de pensamento.

Equívoco: histórias de usuários são suficientes: As histórias de usuários são um método popular para capturar as necessidades dos stakeholders. No entanto, eles pretendem iniciar a comunicação e não representar a especificação completa. O caminho de uma necessidade não específica para um requisito completo é resumido na prática "3C - Cartão, Conversação, Confirmação". Um quadro muito mais completo de requisitos pode ser obtido por meio de uma combinação de histórias de usuário com outras técnicas da ER aprovadas, como diagramas de contexto, protótipos, casos de uso e jornadas de usuários.

Equívoco: a documentação é inútil, somente o código tem um valor duradouro: Embora seja perfeitamente possível que, em alguns projetos particularmente carregados de processos, a especificação excessiva possa ser um problema, não é correto concluir que toda a documentação é inútil. A documentação dos requisitos, assim como a documentação do desenho, a documentação de teste ou os documentos operacionais são todos, em certos contextos, artefatos igualmente válidos e necessários resultantes do processo de desenvolvimento de qualquer produto de software sustentável.

Equívoco: o software funcionando é a única maneira de validar os requisitos: O manifesto Ágil valoriza "software em funcionamento mais que documentação abrangente". Quando se trata da validação de requisitos, uma conclusão incorreta desta afirmação é que a validação de requisitos baseados no funcionamento do software é sempre preferível à validação de uma forma documentada de requisitos. O software como meio de validar os requisitos é preferível se os custos e riscos associados a essa abordagem de validação forem aceitáveis em comparação com o resultado. Se os custos e/ou riscos são altos, a ER fornece várias ferramentas que permitem feedback rápido e validação de requisitos, antes de uma única linha de código ser escrita, por exemplo: maquetes ou storyboards da interface do usuário (ver EU 3.1.10)

EU 1.4.3 Armadilhas da RE@Agile

Armadilha: tratar os requisitos como um tipo uniforme de informação: Um erro típico relacionado à ER em todos os contextos/métodos de implementação é considerar o "requisito" como tipo uniforme de informação. Com base nesse mal-entendido, a documentação dos requisitos é normalmente considerada uma perda de tempo, pois os requisitos mudam tão rapidamente que são inválidos logo após terem sido escritos.

Os requisitos NÃO são um tipo uniforme de informação; os requisitos podem ser definidos em vários níveis de detalhe, abstração e formatos. Por exemplo, a visão do sistema ou os objetivos de um sistema são requisitos em um alto nível de abstração, normalmente com uma vida útil longa; um protótipo executável é um meio para validar um conjunto de requisitos ou para elicitar novos requisitos.

Armadilha: perder a visão do todo: Métodos Ágeis são muitas vezes mal compreendidos e implementados de uma forma que se concentram apenas nos tópicos que estão imediatamente à frente da equipe. Do ponto de vista de um desenvolvedor, isso pode ser tratado como um princípio útil, porque o desenvolvedor pode concentrar sua energia mental no trabalho em questão e não se distrair com tópicos de longo prazo. No entanto, se todos se concentrarem apenas no trabalho em questão, a visão do todo e a perspectiva de longo prazo se perderão. Métodos Ágeis sustentáveis abordam perspectivas de longo e médio prazos em sessões dedicadas (p. ex., sessões de refinamento, sessões de roadmaps ou workshops visuais). Uma armadilha relacionada é prosseguir com a solução, sem definir completamente o problema do negócio (geralmente chamado de necessidade).

Armadilha: sobrecarregar os stakeholders com informações: Os artefatos da ER podem ter uma alta densidade de informações e serem criados muito rapidamente em uma equipe Ágil. Essa abordagem é frequentemente usada em projetos "de ponta" ou de alta tecnologia, em que os especialistas no assunto são difíceis de encontrar. No entanto, utilizar a iteração completa para trabalhar em artefatos da ER coloca uma alta carga de revisão nos stakeholders, exigindo que eles digiram especificações extensas. Melhores resultados podem ser obtidos com uma boa combinação de especificação e desenvolvimento (prototipagem).

Armadilha: elaborar todos os tópicos de forma incremental e iterativa: Nem todos os tópicos de requisitos para um sistema devem ser desenvolvidos de maneira detalhada e incremental. Tópicos com complexidade aditiva (ver [Meyer2014]) são adequados para desenvolvimento incremental. Exemplos típicos de tópicos são processos que podem ser separados em elementos independentes (p. ex., processo de compra em uma loja online). Os tópicos com complexidade completa (ver [Meyer2014]) não são adequados para o desenvolvimento incremental, porque cada novo insight sobre um tópico levará a um entendimento completamente novo das informações já conhecidas. Exemplos aqui são cálculos com parâmetros de entrada complexos e parâmetros de saída simples (como apólices de seguro ou componentes de controle de um motor).

Armadilha: o desenvolvimento incremental pode não incentivar a inovação radical ou disruptiva: O processo incremental do Ágil pode não encorajar o desenvolvimento de ideias inovadoras e/ou disruptivas, pois um determinado artefato (p. ex., o software ou uma característica/função do software) é tipicamente melhorado localmente (p. ex., corrigindo erros ou adicionando elementos ausentes) uma vez que tenha sido definido. Embora o Manifesto Ágil aceite explicitamente a mudança, os processos incrementais do Ágil normalmente oferecem suporte à inovação contínua de produtos e serviços. Inovações radicais ou disruptivas surgem através da consideração de múltiplas ideias e a recombinação de ideias existentes [LiOg2011]. O desenvolvimento de ideias alternativas em termos de software é normalmente considerado

como desperdício em ambientes Ágeis (veja o 10º princípio - maximizar o trabalho não realizado). Para inovações radicais ou disruptivas, práticas adicionais precisam ser incorporadas, como ideias de startups enxutas ou abordagens de design thinking apresentadas na EU 1.5.

Maior e mais importante Armadilha: Mudanças Ágeis e culturais não caminham juntas: Os valores Ágeis promovem mudanças na forma como as organizações estão trabalhando, uma perda de propriedade sobre alguns produtos, e responsabilidade coletiva. O Ágil promove ainda retrospectivas contínuas sobre o comportamento da equipe para melhorar sua maneira de trabalhar: a mudança contínua da equipe e, eventualmente, de toda a organização é inevitável. Tais mudanças culturais (organizacionais) exigem tempo e pessoas qualificadas para liderar as equipes em uma nova direção.

EU 1.5 RE@Agile e o Trabalho Conceitual (L1)

O desenvolvimento Ágil surgiu no mundo da engenharia de software para enfrentar os desafios que vieram do mundo fora da engenharia de software (ver EU 1.1). No entanto, esses desafios não foram experimentados exclusivamente pelo mundo da engenharia de software. Outros ramos da indústria e da sociedade sofriam com desafios semelhantes, como clientes exigentes e ciclos de inovação mais rápidos. Outros campos desenvolveram abordagens para o trabalho conceitual (ou seja, criar conceitos ou especificações de sistemas) que eram bastante semelhantes ao desenvolvimento Ágil. Vários deles são especialmente úteis a partir de uma perspectiva de ER para o desenvolvimento de inovações e visões de produtos. Eles serão apresentados aqui sucintamente, incluindo uma discussão de sua correspondência com a mentalidade Ágil (ver EU 1.2). A seguir, apresentaremos três abordagens como exemplos Ágeis no trabalho conceitual.

Design Thinking (veja [Dsch2015], [LiOg2011]) é um método para resolver os chamados problemas perversos (isto é, por fracamente definidos aparentarem ser de difícil ou até impossível solução). De uma perspectiva da ER, o design thinking é uma combinação de técnicas de elicitação e validação. No centro deste método estão (a) uma equipe multidisciplinar que trabalha com o problema e representa uma ampla gama de conhecimentos necessários para resolver o problema; (b) um ambiente de trabalho no qual a equipe pode trabalhar nas ideias; e (c) um processo iterativo que consiste nas seguintes fases:

- Empatizar: nesta fase, a equipe desenvolve empatia para entender as pessoas por trás do problema que tem que ser resolvido.
- Definir: nesta fase, a equipe reformula o problema para obter um entendimento compartilhado sobre os detalhes do problema que precisa ser resolvido.
- Idealizar: nesta fase, a equipe se concentra na geração de ideias. O objetivo aqui não é desenvolver a ideia. Em vez disso, a equipe desenvolve o máximo de ideias possíveis. No final desta fase, a equipe seleciona as ideias mais promissoras para prototipagem.
- Prototipar: nesta fase, a equipe cria protótipos muito simples (não necessariamente software!) das ideias desenvolvidas. O princípio aqui é que o protótipo deve ser o mais realista e barato possível.

- ▶ Testar: nesta fase, a equipe testa o protótipo com clientes reais para obter feedback sobre suas ideias. Um princípio principal para a fase de teste é “mostrar e não falar”, ou seja, que o protótipo deve ser capaz de falar por si mesmo, para que o usuário possa fornecer feedback genuíno.

As fases do design thinking são escalonáveis e podem ser executadas em projetos que variam de alguns dias a várias semanas. Além disso, as fases não definem uma sequência estrita. Sempre que for necessário, a equipe pode decidir voltar ou avançar no processo. Com isso em mente, o design thinking está alinhado com o valor Ágil “responder à mudança mais que seguir um plano”. O resultado final de um processo de design thinking é um conjunto de protótipos que representam soluções validadas e inovadoras para o problema definido no início. Assim, o design thinking pode ser usado para desenvolver ideias para software de alto valor e, portanto, suporta o primeiro princípio Ágil. Como dito acima, a equipe multidisciplinar e o ambiente de trabalho são elementos centrais do processo, alinhados com o 5º princípio Ágil. Um dos principais objetivos da fase de protótipo é produzir protótipos baratos e simples, o que está de acordo com o princípio Ágil de simplicidade.

Design Sprint [KnZK2016] é um processo de cinco dias para o desenvolvimento de ideias baseadas nos princípios de desenho, prototipagem e testes com os clientes finais. Do ponto de vista da ER, o design sprint também é uma combinação de técnicas de elicitação e validação. O centro deste método é a forma de trabalhar com o tempo; cada dia é dedicado a uma das seguintes atividades: desempacotar o conhecimento da equipe, esboçar idéias, decidir quais ideias prototipar, prototipar as ideias selecionadas, e finalmente testar as ideias com clientes reais. A diferença importante em relação ao desenvolvimento Ágil é que o protótipo não precisa ser software. É importante reconhecer que o termo “sprint” não se refere ao sprint do Scrum.

Lean Startup [Ries2011] é uma abordagem para o desenvolvimento de negócios e gerenciamento de startups que é muito bem aceita na comunidade Ágil. De uma perspectiva da ER, também contém várias ideias que são muito interessantes. Dois exemplos são a abordagem especial de desenvolvimento de produtos e o produto mínimo viável. A abordagem de desenvolvimento de produtos é chamada de construir-medir-aprender (build-measure-learn) e concentra-se principalmente no aprendizado contínuo sobre as necessidades do cliente. O produto mínimo viável (MVP) é “uma versão de um novo produto, que permite a uma equipe coletar o máximo de aprendizado validado sobre clientes com o mínimo de esforço” [Ries2011]. Outro conceito importante do Lean Startup é o pivô, “uma correção de curso estruturada, desenhada para testar uma nova hipótese fundamental sobre o produto, a estratégia e o mecanismo de crescimento” [Ries2011]. De uma perspectiva da ER, essas ideias são uma combinação de técnicas de elicitação e validação. Em vez de elicitar e validar requisitos com base em conceitos ou documentos, a elicitação e a validação são realizadas com o produto real, que é preferível de acordo com Ries, sob circunstâncias de extrema incerteza. Ries enfatiza que o MVP não deve, necessariamente, ser um software totalmente funcional e completo. Em vez disso, o livro mencionou uma página web muito simples para vender sapatos com um processo de envio manual, para validar a necessidade de compras online de sapatos.

Essas abordagens mostram que o Ágil é realmente mais do que o Scrum. Os exemplos de design thinking, design sprint e lean startup mostram que há abordagens para o trabalho conceitual na RE@Agile que compartilham a mentalidade dos Métodos Ágeis e, portanto, são totalmente compatíveis com organizações que desejam desenvolver software de maneira Ágil. Essas e outras abordagens não devem ser desconsideradas como uma nova forma de abordagem cascata ou planejamento antecipado. Elas podem e devem ser usadas dentro da estrutura do desenvolvimento Ágil (p. ex., dentro de uma ou mais iterações) para desenhar aspectos dedicados de um sistema. Alternativamente, elas podem ser usadas como atividades que precedem o desenvolvimento Ágil (p. ex., uma fase rápida de pensar de antemão). Assim, estas abordagens ajudam a superar o potencial limitado de inovação no desenvolvimento Ágil (ver EU 1.4.2).

EU 2 FUNDAMENTOS DA RE@AGILE (N1)

Duração: 90 min

Termos: Proprietário do Produto, Backlog do Produto, Backlog da Sprint, Épicos, Histórias de Usuários, Mapas de Histórias

Objetivos educacionais:

- OE 2.1.1 Conhecer exemplos de métodos Ágeis
- OE 2.2.1 Conhecer o Scrum como exemplo: suas funções, processos, artefatos e sua relevância para a Engenharia de Requisitos
- OE 2.2.2 Conhecer as responsabilidades de um Proprietário do Produto no Scrum
- OE 2.2.3 Conhecer o conceito de backlog do produto com épicos e histórias de usuário
- OE 2.3.1 Conhecer a diferença entre um engenheiro de requisitos clássico e um Proprietário do Produto no Scrum
- OE 2.4.1 Conhecer boas razões pelas quais a Engenharia de Requisitos deve fazer parte de um processo contínuo
- OE 2.5.1 Conhecer o desenvolvimento orientado a valor (isto é, priorização de requisitos)
- OE 2.5.2 Saber que valor é risco E gerenciamento de oportunidades
- OE 2.5.3 Conhecer exemplos de valor em organizações com e sem fins lucrativos
- OE 2.6.1 Conhecer como simplificar o Processo de Definição do Produto e como definir um produto mínimo viável
- OE 2.7.1 Conhecer o valor dos processos contínuos e sua curva de aprendizado

EU 2.1 Métodos Ágeis (Uma visão geral) (L1)

Muitos métodos foram desenvolvidos compartilhando os valores do manifesto Ágil. Esta unidade educacional lhe dará uma visão geral de alguns deles, a fim de ver a diversidade de abordagens. A lista não pretende ser exaustiva e irá discutir os métodos, principalmente do ponto de vista da Engenharia de Requisitos.

Crystal é uma família de métodos desenvolvida por Alistair Cockburn [Cock1998]. Ele sugere que cada projeto precisa de seu próprio modelo de processo adaptado. Dependendo do tamanho, complexidade e criticidade, Alistair sugere funções, atividades e artefatos adequados. Crystal Clear - o método para projetos pequenos e menos críticos - é muito semelhante ao XP. Crystal Orange e Crystal Red adicionam um pouco mais de formalismo para lidar com projetos maiores. Do ponto de vista de requisitos, Alistair sugere (entre outras coisas) trabalhar com modelos de casos de uso menos frequentes e maquetes (mock-ups).

Lean Development [Popp2003] e **Kanban** baseiam-se em princípios usados pela primeira vez na produção automotiva na década de 1940. Eles foram adaptados para projetos de TI no contexto de métodos Ágeis [Ande2012]. Eles se esforçam para descobrir 7 tipos de desperdícios no processo de produção (produtos intermediários inacabados, superprodução, defeitos, ...) e gradualmente eliminam cada um deles para acelerar a entrega final.

Scrum [Scrum2016] é um framework para desenvolver e sustentar produtos complexos. O framework concentra-se no desenvolvimento incremental e iterativo. Ele identifica apenas três funções principais: um Proprietário do Produto (para gerenciar o backlog do produto, ou seja, definir a visão e todos os requisitos relevantes de um produto), uma equipe de desenvolvimento para implementar esses requisitos em sprints curtas e um Scrum Master para monitorar o processo e mediar entre os outros papéis. Discutiremos o Scrum em mais detalhes na próxima seção.

TDD (Test driven development) [Beck2003] baseia-se na ideia de escrever o teste antes de codificar a funcionalidade correspondente. Os casos de teste são uma especificação exata e detalhada dos requisitos que o produto deve cumprir.

XP (eXtreme Programming) [Beck2004]: O XP enfatiza a comunicação direta entre um cliente e um programador ("o cliente no local" sentado ao lado do programador, discutindo constantemente os requisitos e obtendo feedback imediato na forma de funcionalidades implementadas).

EU 2.2 Scrum (mais que boas práticas) como um Exemplo (L1)

O Scrum é o framework Ágil mais popular e mais adotado. O Scrum é um framework leve para desenvolver produtos em ambientes complexos. O Scrum Guide [Scrum2016] fornece uma definição do Scrum e seus componentes essenciais.

O Scrum propõe 3 funções (Scrum Master, Proprietário do Produto, Equipe de Desenvolvimento), 4 eventos (Planejamento da Sprint, Reunião Diária de Scrum, Revisão da Sprint e Retrospectiva da Sprint) e 4 artefatos (Backlog do Produto, Backlog da Sprint, Objetivo da Sprint e Definição de Concluído). O Scrum não recomenda nenhuma prática de engenharia.

O Scrum sugere o desenvolvimento de produtos de forma iterativa e incremental em uma série de Sprints, limitadas no tempo a um mês ou menos. Cada Sprint resulta em um Incremento no Produto: um produto parcial que poderia ser usado por usuários finais em um ambiente de produção.

A Sprint

A Sprint é o condutor essencial para o desenvolvimento, pois é um processo iterativo de planejar-fazer-verificar-agir que permite ciclos de feedback curtos. Cada Sprint começa com o Planejamento da Sprint, um evento onde a Equipe do Scrum colabora para definir o que pode ser entregue no próximo Incremento do Produto e como alcançá-lo (decomposição). A origem desse plano é extraída do Backlog do Produto (uma lista ordenada e dinâmica de tudo o que pode ser necessário no produto). Os resultados do Planejamento da Sprint são: a Meta da Sprint e o Backlog da Sprint (seleção de itens e sua decomposição para a Sprint).

A Equipe do Scrum começa a implementar o Incremento do Produto. A Equipe de Desenvolvimento é responsável por transformar itens selecionados em resultados tangíveis, trabalhando com o Proprietário do Produto para refinar os itens do Backlog do Produto e compartilhar feedback sobre a implementação atual. A equipe de desenvolvimento se sincroniza todos os dias durante a reunião diária do Scrum para avaliar o progresso e atualizar o Backlog da Sprint.

O trabalho da Equipe de Desenvolvimento é guiado pela “Definição de Concluído”: uma definição do que deve ser alcançado para que um incremento seja completo. A “Definição de Concluído” ajuda os stakeholders a entender de forma inequívoca o progresso do produto.

Quando o limite do tempo da Sprint termina (4 semanas ou menos), o Incremento do Produto é inspecionado na Revisão da Sprint pela Equipe Scrum e pelos principais stakeholders para avaliar os resultados da Sprint e discutir o que pode ser feito a seguir. O Backlog do Produto é atualizado de acordo.

A Sprint termina com a Retrospectiva da Sprint, durante a qual a Equipe do Scrum inspeciona como foi a última Sprint e como melhorar a eficiência. Imediatamente após o fim da Retrospectiva da Sprint, a próxima Sprint é iniciada.

Boas Práticas

Mesmo que o Scrum não venha com técnicas de engenharia de requisitos, a comunidade Ágil desenvolveu algumas boas práticas que se encaixam bem no Scrum.

O Guia do Scrum usa o termo Itens do Backlog para os itens listados no Backlog do Produto. Este é um termo genérico para identificar qualquer tipo de informação sobre o produto a ser desenvolvido, mas muitos itens do Backlog do Produto são de fato requisitos ou podem ser refinados para se tornarem requisitos.

A comunidade Ágil propõe:

- ▶ Decompor os requisitos em: épicos, opcionalmente funcionalidades/características e histórias de usuário (nível de granularidade);
- ▶ Distinguir entre Requisitos Funcionais (capacidade), Requisitos de Qualidade (comportamento) e Restrições (ambiente);
- ▶ Agrupar requisitos por temas.

Uma boa prática (adicional) foi desenvolvida para descrever as características do backlog do produto: o backlog deve ser “DEEP” (Detalhado apropriadamente, Estimado, Emergente, Priorizado) [Cohn2004]. Dentro da Sprint, o backlog é tratado e ajustado, que é uma parte essencial do processo de desenvolvimento do Ágil.

Com a “Definição de Concluído” (DoD), a equipe desenvolve um entendimento comum de quando um Item do Backlog do Produto está completo e pronto para ser lançado em produção [Scrum2016].

Outra boa prática é aplicar a regra INVEST [Wake2003]. A sigla cobre os seguintes critérios:

I: Independente um do outro

N: Negociável

V: Valioso

E: Estimável

S: Simples o suficiente para caber em uma sprint

T: Testável

A ER oferece critérios correspondentes a bons requisitos (ver [IREB2015]):

- Acordado
- Não ambíguo
- Necessário
- Consistente
- Verificável
- Viável
- Rastreável
- Completo
- Compreensível

EU 2.3 Diferenças e Semelhanças entre Engenheiros de Requisitos e Proprietário do Produto (L1)

Depois de discutir os princípios do framework Scrum, vamos comparar o papel do tradicional engenheiro de requisitos com o papel do Proprietário do Produto.

As principais atividades de um engenheiro de requisitos são [IREB2015]:

- Elicitação de Requisitos
- Documentação de Requisitos
- Validação de requisitos.
- Gerenciamento de Requisitos

Conforme explicado acima, as principais responsabilidades de um Proprietário do Produto são:

- ▶ Garantir que a equipe de desenvolvimento ofereça valor constante ao negócio: isso significa que o Proprietário do Produto deve equilibrar a visão de longo prazo do produto com necessidades de curto prazo, priorizar o backlog do produto do ponto de vista de negócio e inspecionar os resultados da equipe de desenvolvimento junto com os stakeholders no final de cada sprint.
- ▶ Gerenciar todos os stakeholders: o Proprietário do Produto é responsável por encaminhar requisitos consistentes para a equipe. Ele precisa coletar requisitos de todos os stakeholders e garantir que eles não se contradigam. Qualquer disputa entre os stakeholders deve ser resolvida, para liberar a equipe de desenvolvimento de tais disputas.
- ▶ Fornecer continuamente à equipe de desenvolvimento o item de maior prioridade do backlog: A granularidade desses requisitos deve ser pequena o suficiente para caber em uma única sprint. Para quaisquer questões que surjam após a reunião de planejamento da sprint, o Proprietário do Produto deve estar disponível para esclarecê-las rapidamente.

Comparando essas duas funções, verifica-se que tanto os engenheiros de requisitos quanto os Proprietários do Produtos (juntamente com todos os outros stakeholders) precisam executar as principais tarefas de elicitar, documentar, validar e gerenciar requisitos. As notações e ferramentas usadas, no entanto, são geralmente menos formais em ambientes Ágeis:

- ▶ cartões de história em vez de documentos de requisitos;
- ▶ mais conversa e menos escrita;
- ▶ mais ênfase no estado atual dos requisitos, menos ênfase na versão e no histórico.

Embora continue a manter uma responsabilidade geral pela alta qualidade dos requisitos, o papel do Proprietário do Produto é mais amplo do que o de um engenheiro de requisitos tradicional, pois ele(a) é responsável pelo sucesso do produto como um todo, coletando continuamente feedback do negócio e atualizando e priorizando o backlog de acordo.

EU 2.4 Engenharia de Requisitos como Processo Contínuo (L1)

No Ágil, então, a Engenharia de Requisitos é menos uma fase distinta durante o desenvolvimento e mais uma atividade contínua e iterativa. Não é uma meta ter todos os requisitos elicitados e analisados antes que o desenho e a implementação possam começar; requisitos e produtos são criados de forma iterativa e incremental.

A Engenharia de Requisitos é, portanto, uma atividade contínua que dura tanto quanto o produto em si. No entanto, este processo tem resultados intermediários bem definidos: os requisitos previstos que prometem o maior valor ao negócio devem estar “prontos para implementação” (no sentido da “definição de pronto” descrita acima). Outros requisitos menos urgentes do ponto de vista de negócio só serão refinados quando os urgentes forem concluídos.

O “Processo contínuo” não exclui algumas atividades iniciais importantes. Mesmo que os requisitos sejam esclarecidos com base na “necessidade de saber”, existem alguns aspectos dos requisitos que, mesmo assim, devem ser abordados no início do ciclo de vida. Exemplos são a definição de visões ou metas, conhecimento dos stakeholders e o estabelecimento do escopo do produto. Iniciar a implementação sem tais atividades aumenta consideravelmente o nível de risco.

EU 2.5 Desenvolvimento Orientado a Valor (L1)

Os métodos Ágeis se esforçam para fornecer continuamente valor do negócio ao usuário final. Muitas vezes, o valor do negócio pode ser expresso diretamente em termos financeiros, em aumento de participação de mercado ou em termos de satisfação do cliente. Essa abordagem é frequentemente usada em organizações orientadas a lucros, mas não é tão claro como definir o valor para organizações sem fins lucrativos. Aqui, medidas como taxa de uso ou de satisfação de um produto (ou seja, cliques em websites ou doações de uma organização sem fins lucrativos) podem ser mais relevantes.

Um tipo diferente de valor é a redução de risco. Boas abordagens Ágeis tentam equilibrar o valor do negócio e a redução de riscos nas iterações.

A fim de determinar quais requisitos levam ao valor ideal, os métodos Ágeis frequentemente se esforçam para obter produtos mínimos viáveis (MVPs) ou produtos mínimos comercializáveis (MMPs), como explicado na próxima seção.

EU 2.6 Simplicidade como Conceito Essencial (L1)

Num mundo complexo, a simplicidade é uma forma de abraçar a complexidade pelo processo de:

- ▶ Criar resposta simples e potencialmente incompleta para um problema, criando um incremento de valor.
- ▶ Ganhar a capacidade de aprender mais sobre o contexto, com base na experiência do mundo real,
- ▶ Adaptar e iterar na criação e entrega de valor em um ritmo sustentável.
- ▶ Poupar recursos de uma ideia não lucrativa para redistribuí-los em uma nova ideia, falhando rápido e aprendendo rapidamente.

Simplicidade é, de certa forma, oposta à “perfeição”. Dito isto, a simplicidade, na maior parte do tempo, não significa “má qualidade”, mas sim “escopo mínimo” ou “serviço mínimo” - mas sempre com alta qualidade. Qualidade não é negociável.

Dois tipos de produtos “mais simples”: MVPs e MMPs

Um Produto Mínimo Viável (MVP) é um conceito de lean start-up (ver [Ries2011]) e é definido como o menor produto que pode criar uma experiência do usuário final e fornecer feedback para a equipe. Esse feedback é uma entrada importante para evoluir o produto.

Muitas startups estão alinhadas nessa forma de trabalho, pois permitem um rápido retorno sobre o investimento com baixo nível de risco.

O objetivo dos Produtos Mínimos Comerciais (MMP) vai um passo além. A questão não é apenas fornecer feedback antecipado, na condução dos próximos passos dos requisitos, mas criar valor imediatamente. Muitos produtos podem ser usados em uma “versão 1” sem ter todos as funcionalidades e qualidades desejadas, mas, gerando receita para pagar pela melhoria contínua do produto.

EU 2.7 Inspeccionar e Adaptar (L1)

Muitos métodos Ágeis (incluindo o Scrum) enfatizam a importância do feedback frequente e rápido. Após cada iteração (ou até com mais frequência), a equipe deve discutir se o processo de desenvolvimento está funcionando ou se deve ser melhorado.

Nesse processo de feedback, todos devem inspecionar o processo de desenvolvimento atual em um estágio inicial. A equipe deve questionar os métodos utilizados, as ferramentas, a cooperação na equipe etc. Todos devem responder a perguntas como: O que funcionou bem? O que não funcionou bem? O que devemos tentar na próxima iteração?

É importante que o produto ou solução seja revisado ao final de cada sprint. A equipe deve examinar a velocidade da equipe e alterar ou ajustar a velocidade planejada de implementação para a próxima iteração.

Isso também se aplica ao processo de requisitos. As consequências desses insights devem ser a adaptação a curto prazo das etapas de melhoria do processo.

EU 3 ARTEFATOS E TÉCNICAS NA RE@AGILE (L1)

Duração: 90 min

Termos: Visão do Produto, Roteiro do Produto, Backlog do Produto, Backlog da Sprint, História de Usuário, Critério de Aceite, Função, Requisito Funcional, Requisito de Qualidade, Épico, Modelo de Contexto, Mapa de História, Definição de Concluído, Definição de Preparado

Objetivos educacionais

- OE 3.1.1 Conhecer a diferença entre documentos de especificação tradicionais e backlogs
- OE 3.1.2 Conhecer o valor das visões e metas
- OE 3.1.3 Conhecer o valor agregado para o uso de modelos de contexto em ER.
- OE 3.1.4 Saber distinguir os três tipos de requisitos
- OE 3.1.5 Conhecer os diferentes níveis de granularidade nos requisitos
- OE 3.1.6 Conhecer os diferentes formatos de especificação para os diferentes tipos de artefatos nos processos Ágeis (ou seja, textuais versus baseados em modelos versus diagramados)
- OE 3.1.7 Conhecer o valor dos termos, glossários e modelos de informações
- OE 3.1.8 Conhecer a especificação de Requisitos de Qualidade e Restrições em Processos de Engenharia de Requisitos Ágil
- OE 3.1.9 Critérios de Aceite e de Adequação
- OE 3.1.10 Conhecer o uso da Definição de Pronto e Definição de Concluído em Processos de Engenharia Requisitos Ágil
- OE 3.1.11 Conhecer a diferença entre Protótipo e Incremento
- OE 3.1.12 Conhecer os diferentes tipos de artefatos nos processos RE@Agile (Modelo de Contexto, Épico, História de Usuário, Backlog, Plano Estratégico, Requisito, Definição de Concluído e Definição de Pronto)
- OE 3.2.1 Saber como elicitar requisitos em Engenharia de Requisitos Ágil
- OE 3.2.2 Saber como criar e manter backlogs em processos de Engenharia de Requisitos Ágil
- OE 3.2.3 Saber validar e negociar requisitos em Engenharia de Requisitos Ágil
- OE 3.2.4 Saber gerenciar os requisitos em RE@AGILE

EU 3.1 Artefatos na RE@AGILE (L1)

EU 3.1.1 Documentos de Especificação vs. Backlog do Produto

Para criar produtos ou soluções a partir de requisitos, os requisitos não podem ficar sozinhos, mas precisam ser organizados e documentados como uma lista ordenada de tudo o que pode ser necessário no produto [Scrum2016]. Independente de qualquer metodologia, essa coleção de requisitos é considerada o principal artefato para engenheiros de requisitos, analistas de negócios ou Proprietário do Produtos. Diferentes métodos sugerem diferentes formas e nomes para tais coleções de requisitos.

A ER geralmente chama esse artefato, o resultado do processo de elicitação, de Especificação de Requisitos do Usuário, Especificação de Requisitos do Sistema ou Especificação de Requisitos de Software, dependendo de quem o escreve e do seu nível de detalhamento. Não é necessariamente baseado em documentos, mas simplesmente uma coleção de requisitos em qualquer forma física (papel, repositório, banco de dados ...).

Métodos Ágeis, especialmente o Scrum, preferem o termo Backlog do Produto para essa coleção geral de requisitos (e outras informações relacionadas ao produto, consulte EU 2.1) a serem implementadas no futuro e Backlog da Sprint para aqueles requisitos que foram selecionados para a próxima iteração (Sprint no Scrum) [Scrum2016]. Novamente, a forma física desses backlogs não importa. Eles podem consistir em cartões de índice ou notas adesivas em uma parede ou ser capturados em alguma ferramenta de software apropriada. Embora os nomes e o tratamento possam diferir, todos os artefatos seguem as mesmas ideias e oferecem uma base para elicitação, documentação, negociação, validação e gerenciamento de requisitos.

Em EU 2.2 aprendemos que o DEEP é uma boa prática para que o backlog de produtos seja detalhado de forma apropriada, estimado, revelado e priorizado. Estas características são importantes para o acúmulo de produtos e estão ligadas ou dependem uma da outra. Para o Proprietário do Produto como "otimizador de valor", a classificação ou ordenação do produto é a mais valiosa e importante a ser alcançada. A estimativa apoia isto e o desprendimento apropriado ajuda a se concentrar nas partes importantes, de modo que elas apoiem o pedido. Independentemente do nome para a coleta dos requisitos, certos elementos devem ser definitivamente capturados. Isso inclui todos os tipos de requisitos: metas e visões, a definição do escopo de um sistema ou produto, requisitos funcionais, requisitos de qualidade e restrições, e um glossário (ou seja, definições de termos e abreviações relevantes). Como discutido mais adiante, as abordagens podem variar em notações, sintaxe e nível de detalhe para tais especificações de requisitos. Não ter nenhuma especificação (ou seja, confiar apenas na comunicação verbal com os stakeholders sem nenhum requisito escrito) não é uma opção normal, pois os documentos escritos são frequentemente a base para negociação, testes de aceite, propósitos legais e muito mais. Quanto mais os stakeholders se comunicarem, menos a escrita será necessária, mas os resultados, os requisitos, ainda devem ser capturados em uma forma bem conhecida (escrita ou desenhada). Nos parágrafos a seguir, as diferentes partes desta coleção de requisitos gerais serão discutidas em detalhes.

EU 3.1.2 Visão e Objetivos

Cada processo de desenvolvimento deve ser guiado por visões ou objetivos que definam a capacidade do produto que, se implementadas, significaria que a solução é considerada bem-sucedida. Ter tais visões ou objetivos, acordadas por todos os stakeholders relevantes, o mais cedo possível, é de extrema importância para qualquer atividade relacionada aos requisitos do respectivo sistema ou produto. Nos processos de desenvolvimento Ágil, o termo "visão do produto" é comumente usado para enfatizar que cada resultado do processo de desenvolvimento deve ter um nítido valor para o negócio, relacionado à visão do produto.

Objetivos de diferentes stakeholders podem ser contraditórios, o que significa que os stakeholders relacionados devem negociar para chegar a uma visão ou conjunto de objetivos acordados. Alternativamente, isso pode indicar que são necessárias variantes do produto (p. ex., um pequeno e um grande iPhone), ou mesmo produtos diferentes (um iPhone e um iPad).

O desenvolvimento Ágil geralmente usa metas com granularidades diferentes, como metas para diferentes horizontes temporais ou intervalos de planejamento. Por exemplo, pode haver metas de um ano para permitir negociações sobre as entregas (tempo e conteúdo), metas de três meses para planejamento de release e metas de sprint para a próxima iteração / sprint [High2009].

Visões de produtos de longo prazo e metas de sprint de curto prazo são úteis para enfatizar as realizações mais importantes que devem ser alcançadas dentro de um determinado período de tempo e ajudar a alinhar todos os stakeholders em uma “missão comum”. Todos esses objetivos podem ser representadas de maneira útil em uma linha do tempo no plano estratégico.

A visão ou objetivos do produto são a forma mais abstrata de requisitos e não podem ser levados ao desenvolvimento sem mais refinamento. Eles fornecem orientação fácil de entender e abrangente para todo o processo de desenvolvimento Ágil. Cada requisito deve ser checado em relação às metas para verificar a contribuição do requisito para as diferentes metas. Um requisito sem relação com o conjunto de metas pode ser um indicador do valor de negócio ausente. Conseqüentemente, a declaração de visão do produto e as metas acordadas dos stakeholders são artefatos muito importantes para o sucesso dos processos de desenvolvimento Ágil, pois eles definem a estrutura para todas as atividades de desenvolvimento, sem restringir desnecessariamente a criatividade dos desenvolvedores.

EU 3.1.3 Modelo de Contexto

A declaração de visão, juntamente com os objetivos dos stakeholders, especifica as exigências gerais que o sistema ou produto deve satisfazer para cumprir o seu propósito. Modelos de contexto, por outro lado, representam um ponto de vista diferente, já que seu objetivo é descrever propriedades particulares do ambiente (contexto) em que o sistema ou produto irá operar.

Os requisitos do sistema ou produto são frequentemente especificados considerando-se as suposições sobre o ambiente. Os modelos de contexto são uma maneira estruturada de documentar suposições relevantes sobre o contexto. É benéfico explicitar tais suposições para estabelecer uma visão compartilhada e acordada sobre o ambiente operacional do sistema ou produto para toda a equipe de desenvolvimento e outros stakeholders relevantes. Em caso de incerteza, os requisitos especificados somente estarão corretos se as suposições documentadas nos modelos de contexto forem verdadeiras, significando que os modelos de contexto representam corretamente o contexto operacional real do sistema ou produto.

Os modelos de contexto são artefatos poderosos, pois diferenciam claramente entre o sistema ou produto a ser desenvolvido e seu contexto, consistindo, por exemplo, em sistemas adjacentes e usuários humanos (ver [IREB2015]).

Ao usar essa diferenciação, a funcionalidade pode ser alocada para diferenciar as responsabilidades do sistema ou do próprio produto e as responsabilidades dos sistemas adjacentes ou usuários humanos no contexto - colaborando durante uma operação para atender à visão geral. Portanto, os modelos de contexto também podem ser usados para esclarecer e especificar as interfaces externas do sistema ou produto a ser desenvolvido.

Tais modelos podem ser documentados em diferentes formatos, como diagramas de contexto propostos para análise estruturada de sistema, diagramas de casos de uso, diagramas SysML de definição de blocos, diagramas UML de componentes ou diagramas UML de classes. Qualquer notação é adequada, se diferenciar claramente entre o sistema ou produto a ser desenvolvido e interfaces externas com pessoas ou sistemas no ambiente. Além disso, deve-se documentar as relações entre esses elementos e o sistema ou produto a ser desenvolvido em um nível apropriado de detalhes. Mesmo um simples diagrama com apenas caixas e linhas traçadas à mão pode ser pragmático e útil.

Independentemente da forma de documentação, um modelo de contexto é um artefato muito valioso e é recomendado durante um processo de desenvolvimento Ágil. Ele delimita a área (dentro do escopo) onde os analistas estão livres para tomar decisões, enquanto as interfaces externas (ou seja, o limite entre o escopo e o contexto) precisam ser negociadas com os sistemas adjacentes.

EU 3.1.4 Requisitos

Os requisitos, então, precisam ser capturados com base na visão e nos objetivos e confinados pelo modelo de contexto. O IREB define três tipos diferentes de requisitos: requisitos funcionais, requisitos de qualidade e restrições, que serão discutidos nas seções a seguir.

EU 3.1.5 Granularidade de Requisitos

Muitas vezes os stakeholders transmitem suas necessidades com diferentes níveis de granularidade: desde alto nível, como objetivos gerais de negócio; até refinados, que especificam detalhes da funcionalidade esperada do sistema. Os requisitos funcionais e de qualidade (ver EU 3.1.8) podem (e devem!), portanto, ser discutidos e documentados em diferentes níveis de abstração.

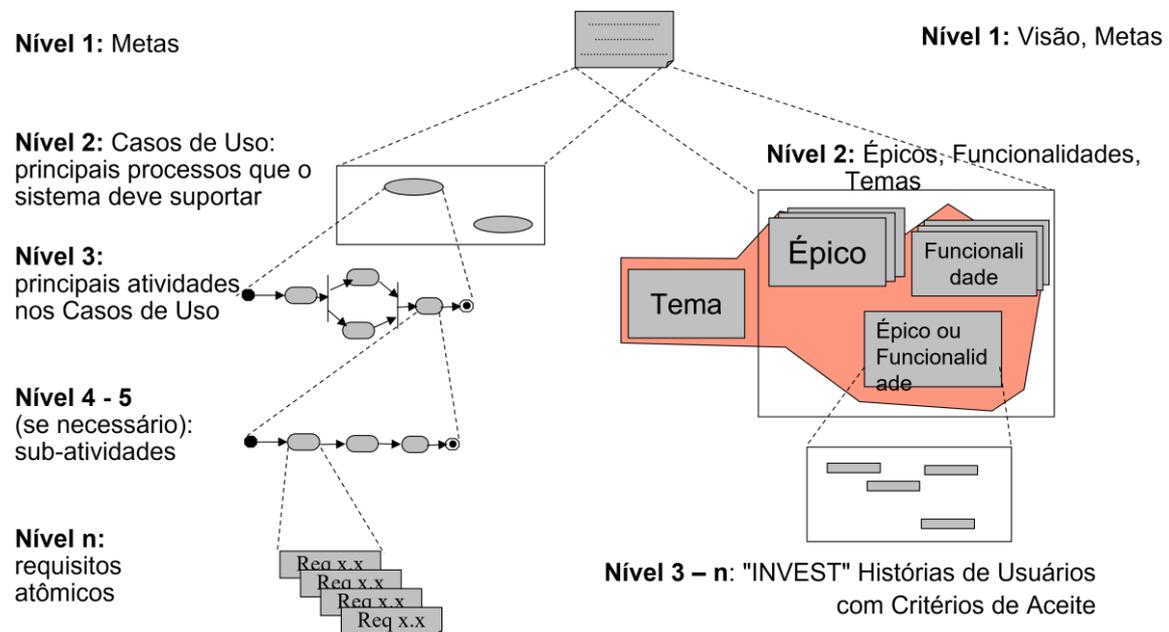


Figura 1: Granularidade dos Requisitos: decomposição de caso de uso e variantes Ágeis, Casos de Uso, Especificações de Casos de Uso e Diagramas de Atividades

Uma abordagem típica das metodologias não-Ágeis é representada no lado esquerdo da Figura 1. Nesta abordagem, os casos de uso são usados primeiramente para refinar visões e metas e fornecer a subdivisão da funcionalidade esperada do sistema (ver Figura 1, nível 2). A figura mostra apenas um exemplo de um agrupamento de requisitos de alto nível. Outros exemplos incluem agrupar por características/funcionalidades, objetos de negócios, fluxos de dados ou componentes de soluções existentes.

No nível seguinte de refinamento, cada caso de uso é elaborado para descrever os caminhos envolvidos no cumprimento dessa função. Várias alternativas de acordo com o nível de complexidade estão disponíveis aqui:

1. Descrição verbal (texto simples)
2. Modelos de caso de uso (formato semiestruturado ou fluxos narrativos)
3. Diagrama de atividade UML (ou qualquer outro tipo de modelo funcional gráfico, como diagramas de fluxo de dados, BPMN, diagrama de contexto, gráfico de estado, modelo de objetos de negócios etc.)

O nível 3 na Figura 1 mostra, como exemplo, os casos de uso elaborados com diagramas UML.

Baseado na complexidade do problema, outros níveis granulares podem ser fornecidos, por exemplo, atividades decompostas em subatividades (ver o Nível 4 na Figura 1) ou com especificações de requisitos textuais de atividades individuais (ver o Nível 5 na Figura 1).

Assim, os requisitos alto nível do cliente são refinados sucessivamente em especificações de funcionalidade esperada da solução.

Observe que a abordagem acima implica um processo top-down de análise e decomposição dos requisitos. Essa visão um tanto idealizada nem sempre é realista. Os requisitos podem, de fato, serem expressos primeiramente no nível do detalhe da solução, e metas gerais estabelecidas mais tarde. Assim, o processo no mundo real pode ser top-down, bottom-up ou alguma combinação destes.

O ponto-chave é que existem métodos e técnicas que apoiam essas discussões em seus diferentes níveis de abstração e estabelecem uma hierarquia significativa de requisitos à medida que o conhecimento geral sobre eles aumenta.

Uma opção para ser usada para descrever casos de uso de baixa complexidade, é usar poucas frases para esclarecer os passos necessários a serem executados. Para casos de uso de complexidade média, outras opções podem ser uma boa escolha, pois os gabaritos de caso de uso permitem uma descrição mais sofisticada dos passos dos processos, que fazem parte do caso de uso específico. Esse formato semiestruturado ajuda a entender os requisitos capturados em um caso de uso com uma diretriz clara. No entanto, se houverem atividades paralelas e alta complexidade com vários cenários o modelo de casos de uso terá limitações.

Por fim, os diagramas de atividades permitem que o engenheiro de requisitos use um formato gráfico no qual vários cenários possam ser exibidos. O diagrama pode representar vários fluxos paralelos e sequenciais de etapas. Essa forma de descrição de casos de uso permite que fluxos de casos de uso complexos sejam descritos em um diagrama.

Terminologia Ágil: Épicos, Temas, Funcionalidades e Histórias de Usuários

No lado direito da Figura 1, uma abordagem equivalente é mostrada usando termos Ágeis, como Épicos, Temas, Funcionalidades e Histórias de Usuário.

Aqui, objetivos de negócios de alto nível e funcionalidades complexas são capturados como épicos ou funcionalidades e agrupados por temas. Esses tópicos podem ser suficientemente complexos, exigindo mais de um sprint para uma equipe Scrum desenvolvê-los [Griff2015].

Esses tópicos complexos são então decompostos em histórias de usuários mais refinados. Os critérios para "boas" histórias de usuários foram discutidos anteriormente (EU 2.2). Eles devem seguir a "Definição de Pronto" [Kron2008], como cumprir os critérios do INVEST [Wake2003]. Especialmente o "E", "S" e "T" são importantes: devem ser estimáveis, pequenos o suficiente para caber em uma iteração e testáveis.

Mais uma vez, se um projeto segue um processo estritamente top-down de refinar épicos em funcionalidades em histórias de usuários, ou se as histórias de usuários são primeiro reunidas individualmente com temas comuns e épicos surgem posteriormente bottom-up, dependerá da natureza do projeto e dos stakeholders. De qualquer forma, o Ágil fornece artefatos para discutir e priorizar tanto o quadro geral quanto os requisitos prontos para desenvolvimento.

EU 3.1.6 Modelos gráficos e descrições textuais

Independentemente da granularidade, há sempre uma escolha sobre a notação de requisitos funcionais. Eles podem ser expressos escrevendo-os em forma de texto semi-estruturado (como histórias de usuários de acordo com um modelo) ou como uma linguagem natural ou formal (como Gherkin) que afirma claramente o que a solução deve fazer.

Como é sabido que a linguagem natural às vezes é imprecisa e unívoca quanto necessária, muitas notações gráficas foram desenvolvidas para superar essa ambiguidade ou mostrar interdependências, como entre dados e processos. Os exemplos incluem diagramas de atividades UML, diagramas BPMN, fluxogramas, gráficos de estados, diagramas de sequência etc.

A maioria destas notações gráficas enfatiza diferentes aspectos. Diagramas de atividade ou BPMN são bem adequados para processos bastante lineares, mostrando passos consecutivos, alternativas ou loops. As diagramas de estado, pelo contrário, são perfeitos sempre que eventos assíncronos influenciam o fluxo. Os diagramas de sequência são perfeitos para "especificação por exemplo", pois mostram cenários concretos de interações sem tentar ser completos.

Há aspectos positivos e negativos na modelagem de processos e dados, ou de dados e interações, ou de processos e interfaces. Eles estão inter-relacionados (processo de suporte de dados) portanto nunca podendo ser uma coisa ou outra, ou sugerindo apenas a utilização deste ou daquele modelo. Uma tal abordagem seria bastante contraproducente. Embora os requisitos textuais sejam mais fáceis de entender por muitos stakeholders, eles também podem ser facilmente mal interpretados ou incompreendidos pelos mesmos leitores. Os modelos gráficos proporcionam mais formalidade, evitando assim diferentes interpretações ou mal-entendidos. A escolha de estilo deve ser determinada pelos principais objetivos da Engenharia de Requisitos, assegurando um entendimento comum entre todos os stakeholders de um lado e fornecendo proteção suficiente contra a incompletude e mal-entendidos do outro lado [GoAk2003].

EU 3.1.7 Definição de Termos, Glossários e Modelos de Informação

Os requisitos funcionais são incompletos sem uma compreensão clara de todos os termos utilizados em tais frases e modelos gráficos.

Portanto, uma coleção de todos os termos de negócio relevantes usados em qualquer artefato de requisitos é um artefato necessário, mas facilmente negligenciado quando se concentra apenas nos requisitos do negócio.

A forma mínima deste artefato é uma lista (textualmente descrita) de termos e abreviações de negócio, geralmente ordenadas alfabeticamente para facilitar a leitura. Isto é às vezes chamado de dicionário, glossário, ou lista de definições.

Quando o negócio é muito complicado, este glossário pode ser estruturado agrupando termos simples em classes (ou entidades) e mostrando a visão geral de todos os termos em formato gráfico. Tais artefatos são então chamados modelos de dados, modelos de informação, diagramas de entidade-relacionamento, ou diagramas de classe UML.

Além da definição de termos, esses modelos também incluem associações relevantes entre essas entidades, ou seja, relações estáticas entre os termos.

Como mencionado acima, o backlog de produtos consiste frequentemente em épicos e histórias de usuários, enfatizando a funcionalidade necessária e desenfaticando definições de termos de negócio. No entanto, mesmo em abordagens Ágeis, é necessária uma compreensão clara dos termos de negócio para criar uma compreensão compartilhada e alinhada dos termos usados em artefatos como épicos e histórias de usuários.

EU 3.1.8 Requisitos de Qualidade e Restrições

Além dos requisitos funcionais (especificando funções que um sistema ou produto precisa fornecer), os requisitos de qualidade e restrições são de importância crucial para o sucesso do sistema ou produto que está sendo desenvolvido. Tradicionalmente, os requisitos de qualidade e restrições estão incluídos no termo geral "requisitos não-funcionais" [CNYM2000].

Os requisitos de qualidade dizem respeito a qualidades específicas que um sistema ou produto deve exibir, por exemplo, em relação ao desempenho, confiabilidade, segurança, proteção ou usabilidade [ISO25010]. Com uma ênfase nos requisitos funcionais do cliente na forma de histórias de usuários, há um risco com o Ágil de que os requisitos de qualidade não sejam declarados explicitamente. Os requisitos de qualidade não podem ser definidos como histórias de usuários, que podem ser desenvolvidas dentro de um sprint: eles descrevem antes um atributo emergente do produto que está sendo desenvolvido e, portanto, devem ser testados continuamente para todas as histórias de usuários. As listas de verificação dos aspectos de qualidade da ER (ver [IREB2015]), por exemplo, podem ser úteis. Os requisitos de qualidade são notoriamente difíceis de incorporar nos sistemas existentes através da refatoração, por isso é ainda mais valioso considerar tais aspectos no início do processo.

Restrições definem as limitações gerais no espaço da solução do sistema ou produto a ser desenvolvido [Glinz2014]. As restrições vêm de várias formas: restrições organizacionais (limitações orçamentais, prazos apertados, um processo de desenvolvimento prescrito etc.), restrições técnicas (exigindo um determinado sistema de BD, o uso de uma linguagem de programação específica, frameworks escolhidos etc.) ou restrições do próprio ambiente dentro do qual o sistema irá operar (padrões, normas, regulamentos etc.).

Semelhante aos requisitos de qualidade, aprender demasiadamente tarde sobre as principais restrições pode ser muito caro, uma vez que muitos desses aspectos não podem ser acrescentados de forma incremental. O planejamento e as decisões de concepção dependem de uma boa compreensão destas questões e, por isso, mais uma vez, é crucial que as principais restrições sejam identificadas no início do processo.

Similar aos requisitos funcionais, requisitos de qualidade e restrições podem existir em diferentes níveis de abstração que devem ser documentados no backlog do produto, tornados visíveis para a equipe (p. ex., colocados no quadro de tarefas do Scrum) e testados em cada sprint. Portanto, pode ser útil adicioná-los à "Definição de Concluído" e implementar sua validação na forma de testes de regressão automatizados.

EU 3.1.9 Critérios de Aceite e de Sucesso

Todos os tipos de requisitos são de valor limitado se o seu cumprimento não puder ser validado, verificado ou testado. Portanto, cada requisito necessita de um conjunto de critérios que podem ser testados para verificar se o requisito foi de fato cumprido. Tais critérios ajudam adicionalmente a compreender (e encorajam o feedback), descrevendo em termos concretos o que se espera.

O tipo de critério utilizado corresponde ao nível de granularidade do requisito:

- ▶ Em níveis superiores de abstração (visão, objetivos, capacidades e épicos), os Critérios de Sucesso ([SAFe]) são geralmente definidos porque só é possível medir se uma funcionalidade ou capacidade necessária é fornecida ou não.
- ▶ Em níveis de abstração mais baixos, os Critérios de Aceite podem ser usados para descrever como a solução será testada contra os requisitos, para obter o aceite.

Os métodos não-Ágeis e Ágeis concordam que os requisitos têm de ser verificáveis. Métodos não-Ágeis frequentemente usam termos como "critérios de qualidade", "critérios de adequação" ou apenas "casos de teste"; no Ágil os termos "Critério de Aceite" (para histórias de usuários) ou "Critério de Sucesso" (para épicos ou temas) são mais comuns.

EU 3.1.10 Definição de Pronto e Concluído

Enquanto os Critérios de Aceite e Adequação pertencem e completam os requisitos de negócio, os artefatos, a "Definição de Preparado" (DoR) e a "Definição de Concluído" (DoD), apoiam o processo formal de desenvolvimento e asseguram a qualidade dos requisitos e incrementos do produto. O DoD é um artefato oficial do Guia Scrum ([Scrum2016]) e funciona como um portal de qualidade para o processo de desenvolvimento Ágil, enquanto o DoR é uma boa prática que deve ajudar a criar requisitos valiosos e evitar sobrecarregar a equipe com requisitos não-qualificados. O DoR ajuda a trazer os requisitos ao nível certo de detalhe e entregar informação suficiente para a negociação que se pretende entre o Proprietário do Produto ou Engenheiro de Requisitos e a Equipe de Desenvolvimento.

EU 3.1.11 Protótipo vs. Incremento

Outra forma de lidar com os requisitos são os protótipos, uma vez que muitos stakeholders que possuem requisitos para um sistema ou produto, não querem escrever ou ler documentos a fim de defini-lo - eles querem resultados imediatos. Para essas pessoas, a melhor maneira de entender suas necessidades e obter feedback é demonstrar as funcionalidades ou capacidades do sistema sob a forma de um sistema em funcionamento.

Uma maneira de fazer isso é através do uso de produtos mínimos e incrementais. A ER propõe dois tipos: incrementos "horizontais" de produto para mostrar mais variedades para validação ("faça mais da coisa certa") e incrementos "verticais" de produto para verificar se a abordagem de desenvolvimento está certa ("faça a coisa certa").

Ao proporcionar uma interação direta com um sistema em funcionamento, os protótipos podem ser muito úteis para obter feedback. As propriedades de usabilidade, como tempo de reação, por exemplo, são difíceis de especificar, mas fáceis de identificar com o software em funcionamento. Os protótipos também podem, no entanto, serem frustrantes para os usuários, porque creem que o desenvolvimento já está terminado; e para os desenvolvedores porque muitas vezes são descartados para serem substituídos por uma tecnologia melhor.

Métodos Ágeis tentam evitar protótipos descartáveis, desenvolvendo imediatamente incrementos de boa qualidade do sistema ou do produto "real". O Ágil se esforça para iterações curtas, fornecendo incrementos demonstráveis de produto que são usados para aprender mais sobre os requisitos. Embora a intenção não seja jogar fora o código desenvolvido, a refatoração é uma boa prática Ágil como resposta à mudança de requisitos funcionais ou de qualidade com base no feedback do usuário.

O termo "spike" no Ágil é usado para se referir a uma iteração de desenvolvimento realizada com o propósito explícito de entender uma área de complexidade (p. ex., a arquitetura do sistema) e assim reduzir o risco. O termo, embora não definido com precisão, pode se referir à validação de uma tarefa ou iteração inteira. A prototipagem é uma técnica válida nos spikes onde, ao contrário de outras iterações Ágeis, o objetivo é o conhecimento adquirido e não o código em funcionamento.

EU 3.1.12 Relação de Artefatos

Como já mencionado, alguns artefatos são muito importantes para o desenvolvimento bem sucedido:

- ▶ É uma boa prática começar com visões ou objetivos.
- ▶ É uma boa prática identificar e conhecer sempre os stakeholders mais importantes.
- ▶ É uma boa prática definir explicitamente o escopo e delimitá-lo a partir do contexto.

Mesmo que ER não-Ágil e ER Ágil possam usar termos diferentes, eles concordam que é necessário entender tanto a funcionalidade como os termos de negócio e capturar os requisitos funcionais (incluindo funcionalidade e dados).

Além disso, os métodos Ágeis e não-Ágeis devem identificar requisitos de qualidade e restrições, uma vez que estes podem influenciar fortemente as decisões de desenho. Ignorá-los ou aprender sobre eles tarde demais pode levar a muito retrabalho e a um produto que não atende às expectativas dos clientes.

Uma boa Engenharia de Requisitos garante que nenhum problema relevante seja esquecido. Métodos não-Ágeis, portanto, muitas vezes insistem em documentar muitas áreas potenciais de interesse relacionadas com os requisitos de um sistema.

As abordagens Ágeis utilizam artefatos menos formais e substituem a documentação em falta por comunicação direta e loops de feedback rápido, possibilitados pelo desenvolvimento incremental de produtos ou protótipos.

O segundo princípio frequentemente citado do manifesto Ágil [AgileMan2001] explica exatamente o que acabamos de discutir:

"... através deste trabalho, chegamos a valorizar: ...software em funcionamento sobre documentação abrangente ..."

Estamos convencidos de que uma consideração deliberada do que precisa ser capturado por escrito, do que pode ser discutido e do que pode ser prototipado ou mostrado em incrementos é muito proveitoso para qualquer organização. Os melhores resultados serão alcançados quando a documentação, comunicação e prototipagem ou desenvolvimento incremental forem equilibrados de acordo com as restrições e cultura da empresa. O Capítulo 4 abordará mais sobre o tema do equilibrar atividades antecipadas de requisitos (e de arquitetura) e as atividades iterativas.

EU 3.2 Técnicas no RE@Agile (L1)

Em EU 3.1 você aprendeu sobre importantes artefatos de requisitos. Nesta unidade educacional, você aprenderá sobre as principais atividades a serem realizadas na Engenharia de Requisitos. O IREB estrutura estas atividades da seguinte forma:

- ▶ Elicitação de Requisitos
- ▶ Documentação de Requisitos
- ▶ Validação e Negociação de Requisitos
- ▶ Gerenciamento de Requisitos

As seguintes subseções discutem estas atividades a partir de uma perspectiva Ágil.

EU 3.2.1 Elicitação de Requisitos

A ER em Desenvolvimento Ágil baseia-se na comunicação intensiva entre todas os stakeholders (incluindo a equipe de desenvolvimento) para elicitar os requisitos. A comunicação informal e direta entre os membros da equipe pode ser considerada uma boa mistura de entrevistas e brainstorming. Técnicas como o "cliente onsite" do XP podem ser tão bem-sucedidas quanto sessões entre o Proprietário do Produto e os stakeholders (normalmente a equipe de desenvolvimento que refina itens do backlog). O objetivo em todos os casos é obter uma maior percepção do que é realmente necessário.

A ER fornece uma gama muito mais ampla de técnicas para descobrir e elicitar requisitos do que os normalmente discutidos no contexto do desenvolvimento Ágil. Estas incluem técnicas de Q/A (não apenas entrevistas, mas também questionários), técnicas de observação, técnicas baseadas em artefatos (reutilização, arqueologia de sistemas etc.) e técnicas de criatividade, como brainstorming ou design thinking [IREB2015]. Estas técnicas apoiam a ideia de requisitos em formato de Histórias de Usuários, que são uma base para uma discussão estruturada em vez de uma prescrição para implementação.

Como mencionado na seção EU 3.1.11, protótipos e incrementos de produtos são outra forma de aprender mais sobre os requisitos. Assim que um incremento de produto é apresentado na reunião de revisão ou demonstração de sprint, novas idéias podem surgir, as quais podem ser levadas diretamente para o backlog de produto e priorizadas pelo Proprietário do Produto.

A Engenharia de Requisitos em desenvolvimento Ágil pode se beneficiar muito da Engenharia de Requisitos não-Ágil estudando as diferentes técnicas de elicitación e tomando uma decisão deliberada sobre qual combinação de técnicas deve ser escolhida. Embora a comunicação intensiva entre os stakeholders e o feedback rápido através de incrementos de produto sejam excelentes ideias, há mais na elicitación do que isso. Se existem centenas ou milhares de stakeholders, por exemplo, a comunicação verbal por si só não é suficiente. Ao tentar descobrir requisitos inovadores e subconscientes, técnicas de criatividade podem ser necessárias. Quando se trabalha sob restrições de tempo (ou seja, não há tempo suficiente para discussões aprofundadas), então técnicas como "snow cards" podem ser mais eficazes. Ao explorar novas tecnologias, os spikes (incrementos simples, limitados no tempo, para explorar soluções potenciais) são uma grande ideia.

EU 3.2.2 Documentação de Requisitos

No Ágil, os requisitos são organizados dentro de um ou mais backlogs. Os dois principais tipos de backlog são o Backlog de Produtos e o Backlog da Sprint. Os requisitos podem ser documentados sob a forma de cartões de histórias anotados com valores de negócio e prioridade. Estes podem ser organizados dentro de mapas de histórias ou decompostos em histórias mais simples. O princípio por trás dos cartões é que o tamanho do cartão restringe o que pode ser escrito, e ajuda a focar nos principais detalhes.

No entanto, a documentação dos requisitos ainda é considerada uma atividade importante para promover a comunicação entre todos os stakeholders. O formalismo para documentação pode ser minimizado se os detalhes forem comunicados verbalmente.

A definição de um grau adequado de documentação depende de muitos fatores como tamanho dos projetos, quantidade de stakeholders envolvidos, restrições legais, ou a criticidade de segurança do projeto. Com base nesses fatores, os projetos Ágeis tentam evitar o exagero de documentação e encontrar um conjunto mínimo de conteúdo útil da documentação.

Apesar de trabalhar com um backlog de produto "vivo" ser uma forma eficiente de lidar com a documentação, nem sempre ela é suficiente. Portanto, vamos dar uma olhada em outros tipos de documentação.

De uma perspectiva ER, distinguimos quatro tipos de documentação:

1. **Documentação para fins legais:** Determinados domínios ou contextos de projeto (p. ex., software no setor da saúde ou aviação) requerem uma documentação com informações específicas (p. ex., requisitos e casos de teste para um sistema) para que um determinado público obtenha aprovação legal.

O princípio da criação da documentação para fins legais é: a documentação legalmente necessária tem de ser derivada das leis ou normas correspondentes e é uma parte inseparável do produto.

2. **Documentação para fins de preservação:** Certas informações sobre um sistema têm um valor duradouro para além do esforço inicial de desenvolvimento. Exemplos incluem os objetivos que o sistema foi construído para alcançar, os casos de uso central que ele suporta ou as decisões que foram tomadas durante seu desenvolvimento, por exemplo, para excluir certas funcionalidades. A documentação para fins de preservação pode se tornar o arquivo compartilhado da equipe, de um produto ou de uma organização. Pode aliviar a dependência da capacidade de memória dos membros individuais da equipe e pode facilitar as discussões sobre decisões anteriores (p. ex., "por que decidimos não implementar isto?").

O princípio é: a equipe decide sobre o que documentar para fins de preservação.

3. **Documentação para fins de comunicação:** A comunicação eficaz e eficiente é uma ferramenta importante nos métodos Ágeis devido à sua interatividade e ciclos curtos de feedback. Na prática, existem várias situações que podem dificultar a comunicação verbal direta: equipes distribuídas, barreiras linguísticas ou restrições de tempo das pessoas envolvidas. Além disso, a informação é por vezes tão complexa que a comunicação direta pode ser ineficiente ou enganadora. Um protótipo em papel ou um diagrama de um algoritmo complicado pode, por exemplo, ser relido mais tarde. Às vezes os stakeholders simplesmente preferem a comunicação escrita à leitura do código fonte ou à revisão do software. Nestes casos, a documentação facilita o processo de comunicação entre todas as partes envolvidas e conserva os resultados.

O princípio para criar a documentação para fins de comunicação é: um documento é criado como um meio de comunicação adicional se os stakeholders ou a equipe de desenvolvimento perceberem um valor na existência da documentação. O documento deve ser arquivado quando a comunicação tiver sido bem sucedida.

4. **Documentação para fins de reflexão:** Um aspecto frequentemente esquecido da escrita de um documento é que escrever sempre será um meio para melhorar e apoiar os processos de pensamento do escritor. Mesmo que o documento seja jogado fora mais tarde no processo, o benefício de melhorar e apoiar o pensamento é duradouro. Por exemplo, escrever um caso de uso força o escritor a pensar em interações concretas entre o sistema e os atores, incluindo, por exemplo, exceções e cenários alternativos. Escrever um caso de uso pode, portanto, ser entendido como uma ferramenta para testar o seu próprio conhecimento e compreensão de um sistema.

O princípio para criar documentação para fins de reflexão é: o pensador decide sobre a forma de documento que suporta o seu pensamento da melhor forma. O pensador não precisa justificar a sua escolha da forma de documentação para refletir. O documento pode ser descartado quando o processo de reflexão estiver terminado.

Os métodos Ágeis podem beneficiar-se significativamente se estes quatro tipos de documentação forem identificados e aplicados no contexto adequado. Resumindo, podemos dizer que documentar requisitos não é um fim em si mesmo, mas que deve facilitar a comunicação entre os stakeholders, especialmente entre o requerente (frequentemente substituído pelo Proprietário do Produto) e a equipe de desenvolvimento.

EU 3.2.3 Validação e Negociação de Requisitos

Enquanto a ER enfatiza a validação de requisitos por meio de métodos como revisões, walktroughs, inspeções ou leitura baseada em perspectiva, os métodos Ágeis se esforçam para validar os requisitos por meio de feedback precoce e frequente sobre os incrementos valiosos do produto. Uma boa prática para suportar isso é o teste de regressão automatizado, que fornece uma validação contínua do desenvolvimento e dos requisitos relacionados. O objetivo da validação de requisitos inclui a identificação de requisitos ausentes, ambíguos ou incorretos, bem como os controversos ou conflitantes, onde técnicas de negociação e resolução de conflitos podem ser aplicadas.

Como o desenvolvimento iterativo e incremental é uma estratégia chave nos métodos Ágeis, a necessidade de validação formal dos documentos diminui. Ela é substituída por negociações constantes entre todos os stakeholders sobre os requisitos, de modo que os conflitos sejam descobertos e resolvidos com antecedência. Outra forma de validação dos requisitos é o teste automatizado (de regressão).

A validação formal também é reduzida ao mostrar resultados rápidos na forma de incrementos integrados do produto. Se o incremento não atender todos os requisitos de todos os stakeholders, a diferença é colocada de volta na backlog de produtos na forma de novos requisitos e avaliada e priorizada com todos os outros itens do backlog.

Ainda assim, os acompanhamentos do backlog de produtos, discussões de valor do negócio, discussões de riscos e negociações imediatas de requisitos são todas técnicas valiosas na Engenharia de Requisitos Ágil. Todas essas técnicas podem ser usadas em reuniões de refinamento onde o Proprietário do Produto e a equipe de desenvolvimento (e stakeholders, se disponíveis) trabalham juntos para encontrar o detalhamento necessário para a implementação, usando os princípios do refinamento contínuo.

Nas primeiras versões do Guia Scrum, a chamada reunião de refinamento de backlog de produtos só foi mencionada indiretamente. Com a nova versão [Scrum2016], foi explicitada sendo uma boa prática para validar os requisitos, descobrindo os problemas antecipadamente e reduzindo o tempo necessário para a reunião de planejamento de sprint mais tarde.

EU 3.2.4 Gerenciamento de Requisitos

Na ER tradicional, a gestão de requisitos preocupa-se com todas as atividades para lidar com os requisitos ao longo do tempo. Isto inclui o gerenciamento de versões, gerenciamento de mudanças, gerenciamento de configuração, rastreabilidade, bem como a adição de atributos

como status, estimativas, prioridades, links para requisitos conflitantes e as pessoas envolvidas na captura, verificação, assinatura, implementação ou teste do requisito.

Como discutido anteriormente (ver EU 3.1.1), os principais artefatos para manter os requisitos dentro do Ágil são um ou mais backlogs. Ao contrário do gerenciamento tradicional de requisitos, os backlogs são desenhados para manter apenas a melhor e mais recente versão de todos os requisitos ainda a serem implementados. Os itens do backlog são normalmente eliminados assim que o produto que preenche esses requisitos é entregue.

As atividades de gerenciamento de requisitos que acontecem no backlog incluem:

1. **Priorização de requisitos:** determinar seus valores negociais para decidir quando implementá-los. Quanto maior o valor para o negócio, mais importante será um requisito, uma vez que os projetos Ágeis tentam entregar primeiro o maior valor para o negócio. Frequentemente existem outros fatores influenciadores do que o valor para o negócio e que serão discutidos no Nível Avançado de RE@Agile.
2. **Estimativa de requisitos:** determinar quanto trabalho está envolvido no seu cumprimento. Estimativas muito grandes são uma mensagem clara para um Proprietário do Produto, de que mais trabalho tem que ser feito para levá-los para a definição de pronto (DoR, veja EU 2).

Isto não quer dizer que outras atividades relacionadas com os aspectos históricos da gestão de requisitos não possam ter lugar no Ágil. No entanto, tais atividades serão tipicamente registradas fora do backlog.

Ao decidir quais atividades de gerenciamento de requisitos são apropriadas em um determinado contexto, o engenheiro de requisitos deve buscar um equilíbrio entre a minimização das despesas não operacionais, permitindo a entrega antecipada de soluções em funcionamento, e as necessidades de longo prazo da organização, tais como conformidade legal, documentação operacional ou transferência para novos membros da equipe de desenvolvimento.

Conclusão

Atividades de requisitos como elicitação, documentação, validação e negociação, assim como gerenciamento de requisitos ainda têm que ser realizadas em geral no desenvolvimento Ágil. As técnicas preferidas para elicitação e documentação podem diferir entre desenvolvimento Ágil e não-Ágil, mas aprender um com o outro revela a melhor solução, pois combina a força, mas reduz o desperdício ou as despesas gerais. Esta forma de trabalhar representa os cinco valores do Scrum (compromisso, coragem, foco, abertura e respeito) e sua representação nos três pilares do Scrum (transparência, inspeção e adaptação) [Scrum2016].

Especialmente a abertura e o respeito são úteis ao trazer as habilidades da ER para o mundo Ágil, trazendo ideias e princípios do Ágil para os especialistas em ER.

Nesta unidade educacional, você aprendeu que mesmo na Engenharia de Requisitos no Ágil, há mais artefatos do que apenas histórias de usuários no backlog de produtos e que as principais atividades de Engenharia de Requisitos não devem ser esquecidas - mas podem ser realizadas com ênfase e métodos diferentes - com base nos princípios Ágeis explicados no EU 1.

EU 4 Aspectos Organizacionais do RE@Agile (L1)

Duração: 90 min

Objetivos educacionais

- OE 4.1.1 Conhecer o relacionamento entre a estrutura organizacional e a RE@AGILE
- OE 4.2.1 Conhecer a interação com os stakeholders nos processos de Engenharia de Requisitos no Ágil
- OE 4.2.2 Saber como a Comunicação e a Colaboração podem ajudar a melhorar os resultados
- OE 4.2.3 Conhecer o Papel da Gestão Ágil
- OE 4.3.1 Conhecer a motivação para economias de escala
- OE 4.3.2 Conhecer as dimensões para a organização das equipes
- OE 4.3.3 Conhecer abordagens para organizar a comunicação entre as equipes
- OE 4.3.4 Conhecer exemplos de estruturas para economias de escala
- OE 4.3.5 Conhecer os principais impactos da economia de escala na ER
- OE 4.4.1 Conhecer os critérios para decidir sobre o nível de ER antecipada e contínua
- OE 4.4.2 Conhecer o nível de detalhe certo para itens de backlog
- OE 4.4.3 Conhecer o valor da validação em RE@Agile
- OE 4.4.4 Conhecer os ciclos corretos de atualização para o backlog de produtos
- OE 4.4.5 Saber como encontrar o momento certo no ciclo de desenvolvimento

EU 4.1 Influência das Organizações no RE@AGILE (L1)

O Ágil tem suas raízes na manufatura e no controle empírico de processos (ver [TaNo1986]) Princípios Ágeis são fáceis de entender e práticas e frameworks Ágeis como Scrum são fáceis de aplicar em startups e pequenas empresas completamente novas. É difícil implementá-los em organizações maiores que se comportam como um organismo vivo, que se defende de cada intruso. Mas, por outro lado, à medida que as organizações descobrem os benefícios da aplicação desses princípios e práticas, elas tentam misturá-los com seu próprio DNA, como descrito na Lei da Conway [Conw1968] [Conw1968]. Fazer isso resulta em um interesse crescente em tópicos como gestão Ágil (ver [ACP/PMI]) e organizações Ágeis (ver [Denn2015] e [Appel2011]), o que leva a discussões sobre Ágil que frequentemente vão além do desenvolvimento (de software).

As ideias de colocar o cliente no centro, a auto-organização das equipes, a capacitação e fortalecimento dos indivíduos e a melhoria contínua têm ressonância no mundo mais amplo dos negócios. No entanto, o foco do RE@Agile, em consonância com as principais disciplinas dos métodos ER e Ágeis, reside firmemente no desenvolvimento e, portanto, olhamos principalmente para esse quando consideramos o impacto do Ágil na organização (como demandante e fornecedor).

No mundo do desenvolvimento de software, muitas implementações Scrum falham em desenvolvimento porque o resto da organização não foi capaz de mudar para suportar as equipes Scrum.

Por que essa mudança organizacional é necessária?

Vamos dar dois exemplos de porque o resto da empresa também deve mudar para apoiar o desenvolvimento Ágil:

1. A (sub)organização demandante deve ser capaz de entregar requisitos suficientemente bons (detalhados o suficiente, seguindo o princípio INVEST) a fim de manter o ritmo de desenvolvimento (fornecedor) constante. Isto, em combinação com a mudança frequente de requisitos, requer um fluxo de demanda contínuo.
2. O departamento de RH precisa entender quais pessoas contratar para apoiar corretamente as equipes Scrum. Muitas vezes podem ser encontradas ofertas de emprego para Scrum Master em que habilidades de programação e certificações são necessárias, o que mostra que os princípios das três funções Scrum são mal compreendidos.

Em EU 4.2 discutimos os aspectos organizacionais ao incorporar uma unidade organizacional Ágil, como uma equipe Scrum, em um ambiente não-Ágil. Em muitos casos, embora a introdução da Agilidade comece no desenvolvimento de produtos, a empresa inteira pode não seguir necessariamente os princípios Ágeis.

A influência na organização, no caso de ser necessário mais do que uma equipe Ágil para resolver um problema complexo é discutida em EU 4.3 O foco principal é novamente a organização de TI e suas interfaces com o negócio. A transição de uma empresa para uma organização totalmente Ágil não é explicitamente considerada aqui, pois está além do escopo de uma discussão sobre ER.

EU 4.4 foca em aspectos organizacionais na linha do tempo, analisando especialmente a questão de quando as atividades da ER devem ser executadas.

EU 4.2 Desenvolvimento Ágil em um ambiente não-Ágil (L1)

EU 4.2.1 Interação com stakeholders fora da organização de TI

O papel da organização de desenvolvimento dentro da empresa é fornecer soluções e serviços a clientes empresariais (tanto dentro como fora da organização).

O Ágil coloca o cliente no centro do desenvolvimento de produtos. Isto significa que o cliente está envolvido durante todo o ciclo de vida de desenvolvimento do produto, que o feedback sobre entregas incrementais é ativamente buscada e que novos requisitos de acordo com as necessidades do negócio são acomodados. Com o envolvimento contínuo do cliente, a ER também se torna um processo contínuo (ver EU 2.4). A pessoa responsável, como o Proprietário do Produto no Scrum, deve engajar seus clientes em uma comunicação aberta e direta, ouvindo as novas necessidades e mudando as expectativas e capturando-as dentro do backlog.

Na prática, esta comunicação pode assumir muitas formas. Se o cliente está de fato disponível para trabalhar com a equipe diariamente, então a comunicação pode de fato ser direta e principalmente informal por natureza, onde apenas os resultados e decisões são documentados. É importante reconhecer fatores fora do controle da equipe de desenvolvimento (p. ex., separação geográfica ou simplesmente falta de disponibilidade), o que significa que a interação direta nem sempre é prática. Aqui, uma forma mais eficiente de comunicação deve ser planejada, seja convidando representantes de clientes para reuniões regulares de planejamento e revisão, ou realizando sessões intensivas e limitadas no tempo (p. ex., design thinking ou design sprint, ver EU 1.5) em um estágio inicial, com o resultado capturado no backlog.

EU 4.2.2 Organização do produto vs. organização do projeto

O Ágil promove relações diretas, abertas e não-hierárquicas dentro da organização e flexibiliza precisamente como os produtos evoluem ao longo do tempo. Organizações maiores, tradicionalmente baseadas em estruturas de gestão top-down, atribuem um elevado valor ao planejamento e previsibilidade. O planejamento de projetos e recursos, por exemplo, são realidades das quais as empresas não podem simplesmente optar por não participar.

O desenvolvimento de software tem sido tradicionalmente baseado em projetos, o que significa que ele ocorre como uma série de empreendimentos temporários para produzir produtos, serviços ou resultados únicos (ver [PMI]). Grupos de projetos relacionados com objetivos ou metas compartilhadas são normalmente chamados de programas, enquanto o planejamento e controle da totalidade de projetos e programas dentro de uma organização é chamado de gerenciamento de portfólio.

Fiel às suas raízes no desenvolvimento de produtos, a abordagem Ágil é mais centrada no produto. Um backlog de melhorias nos produtos é mantido e estes são implementadas em um processo iterativo de melhoria contínua. O Ágil por si só não define uma data final como tal. Enquanto houver melhorias a serem feitas ou benefícios a serem realizados, o trabalho deve, em princípio, ser contínuo se os benefícios superarem o esforço/custo.

Embora essas abordagens não sejam mutuamente exclusivas (o escopo de um projeto pode ser a entrega de um produto específico e projetos adicionais são estabelecidos para melhorias posteriores), as diferenças de perspectiva e terminologia podem ser uma fonte de tensão e mal-entendidos entre o desenvolvimento de software Ágil e organizações não-Ágeis.

Uma abordagem para resolver tais tensões é que enquanto o desenvolvimento de software em si ocorre de forma Ágil, as funções de gerenciamento de portfólio e programas fornecem um nível mais alto de planejamento e controle que utiliza abordagens de ambos os mundos (veja também EU 4.3). A chave para fazer tal abordagem funcionar é a capacidade de conectar lacunas conceituais entre o cumprimento planejado de objetivos e funcionalidades negociais no nível do portfólio e programa e uma entrega iterativa e flexível de funcionalidades individuais de software.

A ER fornece conceitos e métodos necessários para diferenciar entre requisitos nesses diferentes níveis de abstração com requisitos de nível de negócio nos níveis de portfólio e programa, e requisitos de software derivados e detalhados adequados para o backlog de desenvolvimento. A abordagem de requisitos muda de requisitos mais detalhados e precisos utilizados como ordens exatas para o desenvolvimento, para requisitos que são utilizados como base comum para discussão e alinhamento just-in-time, e tão detalhados quanto necessário para o nível atual de planejamento.

EU 4.2.3 O papel do gerenciamento em um contexto Ágil

Disciplinas & Equipes: O pessoal dos departamentos de TI tem sido tradicionalmente organizado por disciplina: desenvolvedores, testadores, engenheiros de requisitos, analistas de negócios, gerentes de projeto etc. As equipes de projeto são reunidas a partir destes vários conjuntos de habilidades para a duração fixa do empreendimento. O Ágil, e o Scrum em particular, promove a ideia de equipes mais inter-funcionais. Além das funções especializadas de Proprietário do Produto e Scrum Master, todos os membros da equipe devem ser capazes de atuar em diferentes capacidades, com diferentes indivíduos apoiando a Engenharia de Requisitos ou atividades de teste, conforme necessário. O objetivo da equipe é ser capaz de entregar os requisitos do cliente na sua totalidade, quaisquer que sejam os elementos tecnológicos ou organizacionais envolvidos. Isto pode ser alcançado através de competências da ER dentro da equipe de desenvolvimento (solução preferida) ou fora (muitas vezes usada na realidade como suporte para o Proprietário do Produto) que então oficialmente não faz parte da estrutura Scrum.

Gerentes de TI: É dever dos gestores de TI (chamados "desenvolvedores de pessoas" em [SAFe]) encontrar o equilíbrio na organização entre o conjunto de especialistas e generalistas e ajudar a organizar essas competências em quantidades apropriadas nas equipes. Com respeito à Lei Conway [Conw1968], a estrutura da equipe será um espelho da estrutura do produto (componentes) ou da estrutura do sistema em TI. Isto é ainda mais importante ao escalar o número de equipes. Se uma empresa organiza suas equipes por componentes ou sistemas, escalar aumentando o número de equipes não ajuda, pois cria ainda mais dependências. Economias de escala só seriam possíveis se escalássemos o número de membros da equipe, mas apenas até certo ponto, pois o esforço de comunicação também aumentará drasticamente. As equipes inter-funcionais que herdaram todas as capacidades de fornecer incrementos inteiros do frontend para o backend podem ser escaladas facilmente - mas não são fáceis de construir e pode ser demorado fazê-lo.

Proprietário do Produto vs. Gerente do Projeto: Como discutido anteriormente EU 2.2, os Proprietários de Produtos ampliam as responsabilidades dos engenheiros de requisitos, assumindo a responsabilidade pelas prioridades negociais dos requisitos fornecidos às equipes de desenvolvimento de produtos. Os Proprietários de Produto devem ser capacitados (conhecimento) e autorizados à tomar decisões de negócio. Algumas decisões tomadas anteriormente pelos gerentes de projeto não são mais necessárias quando se trabalha com abordagens Ágeis, pois as equipes de desenvolvimento são auto-organizadas e só precisam do trabalho a fazer (requisitos em um nível de detalhe que permita o desenvolvimento dentro de

um sprint) para organizar seu trabalho por conta própria (divisão de tarefas e atribuição dentro da equipe).

O que se exige dos gestores de TI Ágil é uma definição clara da visão para o desenvolvimento Ágil e uma comunicação clara dos pré-requisitos culturais para que esta abordagem seja bem sucedida.

Obter o equilíbrio enquanto se atende às expectativas do negócio não é uma questão simples; alguns critérios para o sucesso são discutidos abaixo em EU 4.4

Papel da Engenharia de Requisitos (ER)/ Gestão de Requisitos (GR): Os padrões anteriores também se aplicam ao Scrum como exemplo. Engenheiros de Requisitos podem trabalhar diretamente com a equipe como parte do time de desenvolvimento e, portanto, não serão nomeados separadamente ou podem formar eles mesmos uma equipe que suporte um ou vários product owners como uma equipe. Ambas as abordagens têm suas vantagens e podem ser usadas em combinação sem violar os princípios Ágeis. A ER se tornará a espinha dorsal de um desenvolvimento Ágil bem sucedido.

EU 4.3 Manuseio de problemas complexos por escala (L1)

EU 4.3.1 Motivação para economias escalares

Os valores Ágeis da comunicação direta, diária e não hierárquica são tipicamente representados por equipes pequenas e próximas, como a do Scrum com seus 5 a 11 membros recomendados (3 a 9 membros da equipe de desenvolvimento + Proprietário do Produto + Scrum Master). Preferencialmente, os membros da equipe devem ser fisicamente co-localizados e serem multifuncionais em termos de habilidades de negócios e técnicas.

Em organizações maiores, esta visão ideal sobre o desenvolvimento Ágil pode não ser viável por muitas razões:

- ▶ problemas complexos podem envolver stakeholders e conhecimentos de diferentes partes do negócio que não podem ser facilmente acomodados em uma única equipe.
- ▶ problemas complexos podem envolver uma série de especialistas técnicos e conhecimentos que não podem ser facilmente acomodados em uma única equipe.
- ▶ o escopo exigido por uma data de implementação definida está simplesmente além da velocidade alcançável por uma única equipe.
- ▶ o pessoal dentro das empresas globais pode estar geograficamente distribuído.

O termo Scaled Agile é usado para descrever situações em que várias equipes (a maioria equipes Scrum) são necessárias para trabalhar unidas em um produto/solução compartilhando objetivos comuns. As abordagens Scaled Agile requerem decisões sobre como as equipes Scrum são organizadas e como a comunicação entre elas deve ser coordenada. O objetivo é alcançar uma abordagem eficaz para lidar com problemas complexos, mantendo o maior número possível de vantagens do Ágil.

EU 4.3.2 Abordagens para a organização de equipes

A questão a ser respondida é como uma organização deve se estabelecer em equipes de um tamanho que permita que as elas sejam multifuncionais (tamanho mínimo) mas ao mesmo tempo eficazes (tamanho máximo) no que diz respeito à comunicação e alinhamento. A organização em linhas funcionais (p. ex., uma equipe especializada em uma área de negócio definida, ou mesmo em uma funcionalidade dentro de uma área de negócio) tem a vantagem de concentrar o conhecimento do negócio dentro de uma única equipe. Isto pode facilitar a elicitación dos requisitos, por exemplo, ao reduzir o número de stakeholders diretamente envolvidos com a equipe.

Uma desvantagem desta abordagem é que a entrega de ponta-a-ponta da funcionalidade em uma área de negócios provavelmente envolverá várias especialidades técnicas diferentes, tais como desenho de interface de usuário, motores de processo, bancos de dados e plataformas centrais, tais como ERP ou mainframes, que podem facilmente sobrecarregar o tamanho máximo de uma equipe eficaz.

Uma alternativa para organizar o desenvolvimento é através de linhas técnicas, com equipes especializadas em componentes técnicos ou plataformas.

A vantagem das equipes de componentes ou plataformas é o profundo conhecimento em seu respectivo campo tecnológico. A desvantagem é a dependência que tal abordagem cria entre equipes que trabalham em conjunto para entregar todo o incremento do produto de acordo com o cronograma.

As estruturas para economia de escala (como as denominadas em EU 4.3.4) mostram formas de lidar com esta situação. As atividades de engenharia de requisitos devem se alinhar com a estrutura para alcançar uma visão comum do que pode ser entregue.

Neste cenário, a RE tem um papel particularmente importante a desempenhar em primeiro lugar na divisão das metas e requisitos de negócio em requisitos de subsistemas que podem então ser atribuídos a equipes individuais e, em segundo lugar, no acompanhamento do desenvolvimento para assegurar que o resultado seja uma solução integrada e que o benefício do negócio seja entregue.

Uma mistura de todos os tipos de equipes pode fornecer a melhor e mais pragmática solução para se beneficiar das ideias, levando-se em consideração as restrições específicas da empresa.

EU 4.3.3 Abordagens para organizar a comunicação

Ao responder à questão de quantas equipes podem trabalhar juntas de forma eficiente, podemos distinguir entre duas abordagens diferentes:

1. Utilizar métodos de abordagem de equipes únicas
2. Introduzir conceitos adicionais para organizar a comunicação e as responsabilidades

Utilizar métodos de abordagem de uma equipe única: a primeira abordagem segue a ideia de que artefatos e papéis adicionais não são necessários e a comunicação entre várias equipes deve ser apoiada com as técnicas existentes a partir das abordagens de equipes únicas. Papéis e artefatos adicionais seriam contrários ao pensamento Ágil e levariam a uma complexidade adicional na organização. Nenhuma sobrecarga baseada em novas funções deve ser criada, seguindo o princípio básico de se "manter a simplicidade". A comunicação e coordenação entre as equipes é geralmente iniciada pelos proprietários dos produtos das equipes, mas feita pelos delegados da equipe. Construções como Comunidades de Prática permitem aos membros das equipes compartilhar experiências e coordenar os processos globais.

Introduzir conceitos adicionais: a segunda abordagem recomenda a divisão de problemas maiores em problemas menores e a gestão de responsabilidades por diferentes papéis para diferentes abstrações. Assim, devem ser introduzidos artefatos adicionais para os diferentes níveis de abstração (p. ex., Épicas de Negócios, Épicas Arquitetônicas, Temas de Investimento, Funcionalidades, Histórias de Usuários). Dependendo da estrutura utilizada, são também introduzidas funções adicionais com responsabilidade pelos diferentes níveis de abstração (p. ex., Gestores de Portfólio, Gestores de Produto e Proprietários de Produto). Devido à crescente complexidade, também são necessários artefatos e papéis adicionais para gerenciar o planejamento e a comunicação entre as diferentes equipes e para alcançar resultados integrados por iteração (p. ex., Mapas Estratégicos e Gerentes de Liberação). Além disso, é necessário organizar reuniões específicas para promover a comunicação entre as novas funções e as já estabelecidas. O ER fornece muitas técnicas que podem ajudar a subdividir problemas maiores e a apoiar os novos papéis nos diferentes níveis de abstração (p. ex., modelagem de contexto, modelagem de objetivos).

Ambas as abordagens têm suas vantagens e desvantagens e cada usuário ou empresa precisa encontrar seu melhor caminho com base no caso de negócios.

EU 4.3.4 Exemplo de Frameworks para economia de escalada RE@Agile

Frameworks que suportam economia de escalada do Scrum e Ágil: Existem muitos e diferentes frameworks disponíveis que suportam estas abordagens e devido à crescente importância da economia de escalar o Ágil, este número também está crescendo rapidamente. A seguir, você encontrará uma seleção dos frameworks mais conhecidos:

- ▶ Scaled Agile Framework (SAFe)
SAFe é uma base de conhecimento de padrões de sucesso comprovados, para a implementação do software e desenvolvimento de sistemas lean Ágil em escala empresarial. [SAFe]
- ▶ Large-Scale Scrum (LeSS)
Scrum aplicado a muitas equipes que trabalham em conjunto num só produto. [Less]
- ▶ Nexus
Uma estrutura que vai ao cerne da economia de escalabilidade: os problemas de interdependência e integração entre equipes. [Nexus2015]
- ▶ Scrum@Scale:
A estrutura Scrum at Scale é uma extensão mínima da estrutura central Scrum que mantém a estrutura modular no núcleo da estrutura Scrum, e permite escalar uma implementação Scrum adaptada às necessidades únicas de sua empresa. [SatS]
- ▶ Disciplined Agile Delivery (DAD)
O Disciplined Ágil (DA) é um framework de decisão de processo para empresas lean. Permite economias de escala táticas no nível da equipe e estratégicas em toda a empresa [AmLi2012]

EU 4.3.5 Impactos da escalabilidade no RE@Agile

As camadas de abstração discutidas anteriormente significam que as atividades da ER são gerenciadas por mais funções, tais como Proprietário do Produto, Gerente de Produto, Gerente de Portfólio, Analista de Negócios. Cada função irá criar artefatos da ER correspondentes para suas respectivas áreas de preocupação (Épicos, Funcionalidades, Histórias de Usuários e a rastreabilidade entre eles). São necessárias reuniões adicionais para atividades da ER (p. ex., tempos de histórias, tempos de funcionalidades, demonstrações do sistema), enquanto a comunicação com os stakeholders é realizada por diferentes funções em vários níveis dentro da organização.

Como o próprio Scrum não escala facilmente em seus próprios termos, o ER desempenha um papel crítico na determinação de como os requisitos gerais são decompostos e distribuídos adequadamente entre várias equipes Ágeis, a fim de manter viva qualquer abordagem de escalabilidade. As técnicas ER ajudam a estruturar a análise de problemas e a refinar requisitos de alto nível para requisitos mais detalhados como funcionalidades e histórias de usuário

apropriadas para equipes individuais. Uma abordagem estruturada, aplicando abstrações apropriadas e diferentes perspectivas de análise, fornecerá uma base sólida para uma subdivisão de tarefas entre as equipes de desenvolvimento Ágil semi-autônomas.

Isto aplica-se especialmente às dependências dentro dos requisitos que precisam ser encontrados e discutidos o mais cedo possível para evitar desperdícios no desenvolvimento.

Um desafio adicional ocorrerá se as equipes ou outras funções estiverem trabalhando em locais diferentes. A complexidade da gestão da comunicação aumenta não só devido a mudanças de horário ou de línguas diferentes. Na maioria dos casos, o principal desafio é baseado em diferenças culturais. Como a comunicação é uma das principais tarefas dos Proprietários do Produto e Gerentes de Produto, isto influencia significativamente nas habilidades necessárias dos papéis relevantes da ER.

EU 4.4 Equilibrando a ER antecipada e contínua no contexto de economia da escalabilidade (L1)

Em um nível muito abstrato, os métodos Ágeis podem ser caracterizados como um processo contínuo e iterativo no qual o sistema é desenvolvido de forma incremental com base nos itens de backlog. Da perspectiva da Engenharia de Requisitos, cinco parâmetros (ver subcapítulos seguintes) podem ser identificados e que impulsionam este processo:

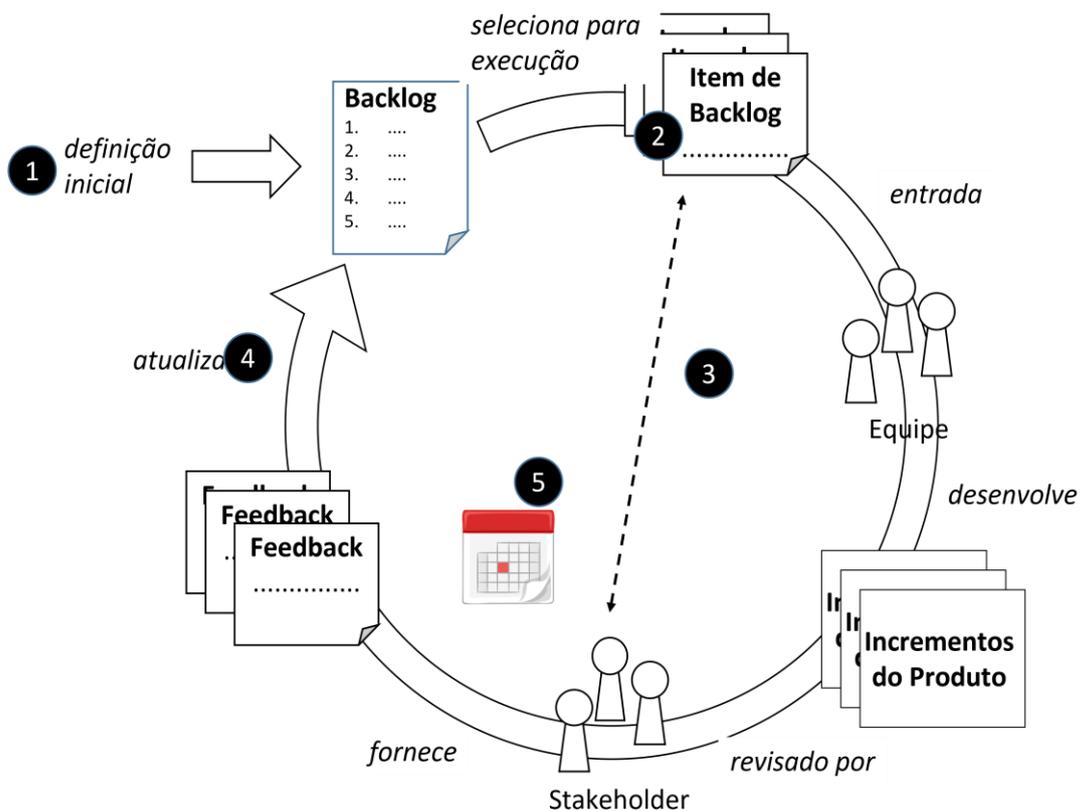


Figura 2: Processo Contínuo de Engenharia de Requisitos

EU 4.4.1 Definição dos Requisitos Iniciais

Antes de iniciar o processo de desenvolvimento contínuo, deve ser criado um backlog inicial (ver [TaNo1986]) Esta definição inicial do backlog é frequentemente referida como "definição inicial". Às vezes, a ER é entendida como que estipulando que esta "definição inicial" seja uma especificação completa e detalhada de todos os requisitos. Este pode ter sido o caso de algumas metodologias específicas, mas nem sempre deve ser assim.

A linha de base para a definição inicial do backlog é a quantidade de informação necessária para iniciar a primeira iteração do processo de desenvolvimento contínuo. Justamente quando iniciar o processo de iteração é uma decisão chave. Dependendo do sistema e do contexto, a incerteza relativa aos requisitos (iniciar sem conhecimento suficiente) pode levar a atrasos adicionais, custos ou potencialmente a falhas no projeto. Por outro lado, iniciar tardiamente pode ter um impacto negativo no tempo de chegada ao mercado e na adequação às necessidades dos usuários finais em um determinado momento.

A ER nos diz que os requisitos que podem ser identificados como tendo um alto impacto sobre a arquitetura, sobre a viabilidade geral da solução ou sobre as principais escolhas relativas à infraestrutura e ao hardware devem ser elucidados e detalhados mais cedo. Tais questões relevantes para a arquitetura devem, de fato, serem levantadas e analisadas antes da primeira iteração do desenvolvimento. Os requisitos de menor impacto podem então ser refinados durante as iterações como um processo contínuo.

Dentro dos processos iterativos e incrementais do Ágil, a ER torna-se um processo contínuo que entrega e aperfeiçoa os requisitos no momento exato e com detalhes suficientes para alimentar o ciclo de desenvolvimento. O processo assemelha-se a um funil para triturar pedras, onde pedras grandes entram no funil e são trituradas em pedras de tamanho médio e, finalmente, em pedras pequenas quando o processo é concluído.

EU 4.4.2 Nível de detalhe para itens do Backlog

O nível de detalhe de um item de backlog limita a liberdade da equipe de desenvolvimento para a realização desse item (ver [Pich2010]). Todos os aspectos de um item de backlog que não foram definidos são deixados como decisões para a equipe, o que deve permitir que a equipe seja mais criativa, mas dentro dos limites definidos do negócio.

As competências disponíveis da equipe de desenvolvimento podem servir como uma regra geral. Se a equipe de desenvolvimento tiver competência suficiente para decidir sobre os detalhes de um item de backlog (p. ex., um especialista em autenticação & autorização faz parte da equipe), a decisão sobre os detalhes deve ser deixada para a equipe.

EU 4.4.3 Validade dos itens de Backlog

Conhecer a validade de um item de backlog antes da implementação ajuda a minimizar o trabalho desnecessário de implementação (ver [Denn2015]). Esperar para reconhecer um requisito errado ou incompleto no software implementado é uma abordagem muito cara para a validação de requisitos.

A determinação da validade de um item de backlog está intimamente relacionada com o nível de detalhe desse item em particular. A validade de um item de backlog só pode ser determinada quando esse item possui detalhes suficientes. Portanto, o esforço de elaboração e validação de um requisito antes da implementação deve ser comparado com o esforço assumido para implementar o requisito e validá-lo posteriormente no software fornecido.

Quando a preferência dos stakeholders em um item de backlog pode ser determinada com esforço aceitável, essa validação de requisitos deve ocorrer antes que o item seja desenvolvido. Exemplos típicos para esses tipos de requisitos (incluindo uma abordagem exemplar de validação) são: desenho geral da interface do usuário (p. ex., com protótipos de IU), mecanismos de autorização e autenticação (p. ex., com revisões de casos de uso), estruturas de dados que devem ser armazenadas no sistema (p. ex., com revisões de modelos de dados) e requisitos para interfaces com sistemas existentes (p. ex., com revisões de diagramas de atividades).

Itens de backlog que são de alto risco (p. ex., funções comerciais críticas, funções críticas de segurança, funcionalidades inovadoras) ou que têm um alto custo de teste para a implementação (p. ex., o software tem que ser testado em um protótipo caro) devem ser validados antes de sua implementação.

EU 4.4.4 Feedback e atualização do Backlog

Os Backlogs são frequentemente atualizados com base no feedback de uma atividade de inspeção, como a revisão de sprint. Tal abordagem é possível para requisitos de pequena escala ou detalhados para os quais o impacto de uma mudança pode ser analisado e compreendido rapidamente em um ambiente com uma ou duas pequenas equipes de desenvolvimento. Não é aconselhável modificar os requisitos que são de maior complexidade ou que têm múltiplas dependências sem aviso prévio. Nessas situações, a modificação do backlog leva mais tempo, os stakeholders podem não estar disponíveis e será necessária uma análise adicional.

Outro fator que pode ter um impacto na modificação dos itens de backlog é o processo de tomada de decisão da organização. Em organizações onde decisões significativas podem levar algum tempo (p. ex., o conselho responsável só se reúne uma vez a cada três meses), o princípio do refinamento contínuo precisa tomar a forma de reuniões concretas envolvendo todos os stakeholders afetados. Tais reuniões precisam da ER como preparação e apoio à decisão.

EU 4.4.5 Cronograma do Ciclo de Desenvolvimento

O parâmetro final para o processo de desenvolvimento é o cronograma ou duração da iteração. Isso tem um efeito significativo nas atividades da ER que precisam ser realizadas enquanto se trabalha com itens do backlog que não estão atualmente em desenvolvimento. Além disso, a duração da iteração determina a frequência na qual os resultados são entregues aos participantes do negócio ou clientes disponíveis para revisão.

Consequentemente, durações mais curtas de iteração aumentam a carga de trabalho dos participantes do negócio por três razões (ver exemplo [Rein1997])

1. Os stakeholders do negócio devem estar disponíveis durante toda a iteração para trabalhar no backlog para criar as demandas para a próxima iteração.
2. Os stakeholders de negócio devem rever os resultados criados pela equipe para fornecer feedback.
3. Os stakeholders de negócio sofrem por várias mudanças de contexto entre suas atividades diárias e o trabalho do projeto.

Tamanhos de iteração mais longos reduzem a pressão, mas também reduzem a capacidade de influenciar o backlog para o desenvolvimento de produtos.

A definição da duração da iteração tem de ser feita tendo em mente a disponibilidade dos stakeholders do negócio. Os stakeholders do negócio normalmente não estão 100% disponíveis para as atividades de desenvolvimento, uma vez que têm outras funções na organização para a qual o sistema é desenvolvido.

Como regra geral, um tempo de iteração mais curto fornece feedback frequente e mais oportunidades para descobrir erros precocemente, assim iterações mais curtas tendem a acelerar as atividades de desenvolvimento. Se um tempo de iteração curto representa uma carga inaceitável para os stakeholders, então deve-se acordar uma duração de iteração adequada, que, dependendo da prioridade do projeto, poderá até ser menor que a iteração curta original.

Outro fator que pode ter um impacto no tempo do ciclo é o tamanho médio e a complexidade dos itens no backlog. Itens do backlog maiores ou mais complexos consomem mais tempo para a compreensão e análise. Portanto, o tempo de ciclo pode ser aumentado para lidar com itens do backlog maiores ou mais complexos dentro de uma única iteração. Por exemplo, se o sistema em desenvolvimento estiver numa fase inicial, um ciclo mais longo pode ser aconselhável para dar à equipe mais tempo para obter uma compreensão inicial do sistema. No entanto, este fator deve ser equilibrado com o objetivo de utilizar tempos de ciclo mais curtos para obter um feedback mais frequente. A decisão de ter tempos de ciclo mais longos ou mais curtos tem de ser tomada em conjunto como uma equipe que pondera a necessidade de análise com o objetivo de obter um feedback antecipado. A mudança dos tempos de ciclo é sempre baseada no princípio "inspecionar & adaptar", tendo em conta que a experiência passada nem sempre é uma garantia para o futuro. A mudança do tempo de ciclo deve ser feita antes do sprint, nunca dentro dele.

EU 5 DEFINIÇÕES DOS TERMOS, Glossário (L2)

O glossário define os termos que são relevantes no contexto do RE@Agile Primer. O glossário está disponível para download na página inicial do IREB em <https://www.ireb.org/en/downloads/#re-agile-glossary>

EU 6 Referências

- [ACP/PMI] Agile Certified Practitioner: <http://www.pmi.org/certification/Agile-management-acp.aspx>. Última visita em janeiro de 2017.
- [AgileMan2001] Agile Manifesto: <http://www.Agilemanifesto.org>, 2001. Última visita em janeiro de 2017.
- [AmLi2012] Ambler S.; Lines, M.: Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise. IBM Press, 2012
- [Ande2012] Anderson, D.J.: Lessons in Agile Management: On the Road to Kanban. Blue Hole Press, 2012
- [Appel2011] Appelo, J.: Management 3.0 Addison-Wesley Professional, 2011
- [Beck2003] Beck, K.: Test-Driven Development by Example. Addison Wesley – Vaseem, 2003
- [Beck2004] Beck, K.: Extreme Programming Explained: Embrace Change. Addison-Wesley Professional, 2004
- [CNYM2000] Chung, L.; Nixon, B. A.; Yu, E.; Mylopoulos, J. : Non-Functional Requirements in Software Engineering. Springer Science & Business Media, 2000
- [Cock1998] Cockburn, A.: Surviving Object-Oriented Projects. Addison-Wesley, 1998
- [Cohn2004] Cohn, M.: User Stories Applied: For Agile Software Development. Addison Wesley Professional, 2004
- [Conw1968] Conway, M.: How Do Committees Invent? Datamation 14(4):28–31, 1968. Artigo disponível no site http://www.melconway.com/Home/Conways_Law.html. Última visita em 2017.
- [Denn2015] Denning, S.: How To Make The Whole Organization Agile. <http://www.forbes.com/sites/stevedenning/2015/07/22/how-to-make-the-whole-organization-agile/#658d3f65135b>, 2015. Última visita em janeiro de 2017.
- [Dscl2015] d.school: An Introduction to Design Thinking – Process Guide. <https://dschool-old.stanford.edu/sandbox/groups/designresources/wiki/36873/attachments/74b3d/ModeGuideBOOTCAMP2010L.pdf>, 2015. Última visita em janeiro de 2017.
- [Glinz2014] Glinz, M.: A Glossary of Requirements Engineering Terminology, Version 1.6. <https://www.ireb.org/downloads/#cpre-glossary>, 2014. Última visita em janeiro de 2017.
- [Griff2015] Griffiths, M.: PMI-ACP Exam Prep. Rmc Publications, 2015

[GoAk2003] Gordijn, J.; Akkermans, J.M.: Value-based Requirements Engineering : exploring innovative e-commerce ideas. Springer, 2003

[High2009] Highsmith, J.: Agile Project Management: Creating Innovative Products. Addison-Wesley Professional, 2009

[ISO25010] ISO/IEC Systems and software engineering -- Systems and software Quality Requirements and Evaluation. ISO/IEC Standard 25010:2011

[IREB2015] IREB e.V.: Syllabus CPRE Foundation Level, version 2.2.
<https://www.ireb.org/downloads/#syllabus-foundation-level>, 2015. Última visita em janeiro de 2017.

[KnZK2016] Knapp, J.; Zeratsky, J; Kowitz, B.: Sprint: How to Solve Big Problems and Test New Ideas in Just Five Days. Simon & Schuster, 2016

[Kron2008] Kronfaelt, R.: Ready-ready: the Definition of Ready for User Stories going into sprint planning. <http://scrumftw.blogspot.de/2008/10/ready-ready-definition-of-ready-for.html>, 2008. Última visita em janeiro de 2017.

[Less] Large Scale Scrum: <http://less.works>. Última visita em janeiro de 2017.

[LiOg2011] Liedtka, J.; Ogilvie, T.: Designing for Growth: A Design Thinking Tool Kit For Managers. Columbia Business School Publishing, 2011

[Martin1991] Martin, J.: Rapid Application Development. Macmillan Coll Div, 1991

[Meyer2014] Meyer, B.: Agile! – the good, the hype, and the ugly. Springer, 2014

[MeMi2015] Mesaglio, M., Mingay, S.: Bimodal IT: How to Be Digitally Agile Without Making a Mess, Gartner 2015 <https://www.gartner.com/doc/2798217/bimodal-it-digitally-agile-making>. Última visita em janeiro de 2017.

[Nexus2015] Nexus Guide
<https://www.scrum.org/Portals/0/NexusGuide%20v1.1.pdf>, 2015. Última visita em janeiro de 2017.

[Patt2014] Patton, J.: User Story Mapping. O'Reilly, 2014

[Pich2010] Pichler, R: Make the product backlog deep.
<http://www.romanpichler.com/blog/make-the-product-backlog-deep/>, 2010. Última visita em janeiro de 2017.

[PMI] PMI Project Management Institute. <http://www.pmi.org/>. [PMI] PMI Project Management Institute. <http://www.pmi.org/>. Última visita em janeiro de 2017.

[Popp2003] Poppendieck, M.: Lean Software Development: An Agile Toolkit. Addison-Wesley Professional, 2003

[Rein1997] Reinerstsen, D. G.: Managing the Design Factory – A Product Developer’s Toolkit. Simon & Schuster, 1997

[Ries2011] Ries, E.: The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses, Crown Publishing Group, 2011

[SAFe] SAFe – Scaled Agile Framework. <http://www.scaledagileframework.com>. Última visita em janeiro de 2017.

[SatS] Scrum at Scale Framework: <https://www.scruminc.com/scrum-incs-scrum-at-scale-framework/>. Última visita em janeiro de 2017.

[Scrum2016] Schwaber, K. & Sutherland, J.: The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game, July 2016.
<http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf>. Última visita em janeiro de 2017.

[ShYo2006] Sheppard, J. M.; Young W. B.: Agility literature review: Classifications, training and testing. Journal of sports sciences 24(9): 919-932, 2006

[TaNo1986] Takeuchi, H.; Nonaka, I.: The new new product development game. Harvard Business Review 64(1), January/February 1986, p.137-146

[Wake2003] Wake, B.: Invest in Good Stories and Smart Tasks,
<http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/>. Última visita em janeiro de 2017.