IREB - Certified Professional for Requirements - Engineering - Nível Avançado RE@Agile -

Syllabus

Versão 2.0.0
Março 2023

Termos de Uso

1. Indivíduos e os provedores de treinamento podem utilizar este syllabus como base para cursos, desde que os direitos autorais sejam reconhecidos e incluídos no material do curso. Além disso, o Syllabus somente poderá ser utilizado para fins de publicidade mediante autorização por escrito do IREB e.V.

2. Qualquer indivíduo ou grupo de indivíduos poderá utilizar este Syllabus como base para artigos, livros ou outras publicações derivadas, desde que tais publicações reconheçam e citem os autores do presente documento e o IREB e.V. como fonte e detentor dos direitos autorais do mesmo.

Agradecimentos

Este programa foi escrito por: Lars Baumann, Stefan Gärtner, Peter Hruschka, Kim Lauenroth, Markus Meuten, Sacha Reis, Gareth Rogers e Hans-Jörg Steffe.

Revisão por Rainer Grau. Os comentários de revisão foram fornecidos por Jan Jaap Cannegieter, Andrea Hermann, Uwe Valentini e Sven van der Zee. Traduzido para a Língua Portuguesa por Ana Moreira, Carlos Silva, George Fialkovitz, Guilherme Siqueira Simões.

Aprovado para liberação em 18 de fevereiro de 2022 pelo Conselho do IREB, sob recomendação de Xavier Franch.

Agradecemos a todos pelo envolvimento.

Prefácio

Objetivo do documento

Este syllabus define o nível avançado da certificação "RE@AGILE" estabelecida pelo International Requirements Engineering Board (IREB). O syllabus fornece aos provedores de treinamento a base para a criação de seus materiais de curso. Os participantes dos cursos podem utilizá-lo (além de subsídios adicionais na literatura especializada) como preparo para o exame de certificação.

Nível de detalhe

O nível de detalhamento deste Syllabus possibilita uma consistência de cursos e avaliações em âmbito internacional. Para atingir este objetivo, o syllabus contém o seguinte:

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

Objetivos Educacionais/ Níveis de Conhecimento Cognitivo

A todos os módulos e objetivos educacionais deste syllabus é atribuído um nível cognitivo. Os seguintes níveis são usados:

- **N1: Conhecer** (descrever, enumerar, caracterizar, reconhecer, nomear, lembrar, ...) - lembrar ou recuperar material previamente aprendido.
- **N2: Compreender** (explicar, interpretar, completar, resumir, justificar, classificar, comparar, ...) - Entender/construir o significado de determinado material ou situações.

- **N3: Aplicar** (especificar, escrever, desenvolver, implementar, ...) - Aplicar conhecimento e habilidade em dadas situações.
- **N4: Analisar** (investigar, concluir, fornecer argumentos para, ...) - Analisar um determinado problema, discutir o que deve/pode ser feito, dividir o problema em partes, aplicar o pensamento crítico, argumentar sobre causas e efeitos.
- **N5: Avaliar** (criticar, julgar) - Fazer uma crítica bem fundamentada de um determinado artefato; fazer um julgamento profundo em um determinado caso.

Os níveis mais altos incluem os mais baixos.

Todos os termos definidos no glossário devem ser conhecidos (N1), 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/downloads/#re-agile-glossary

Este Syllabus utiliza a abreviatura "RE" para Engenharia de Requisitos.

Estrutura do Syllabus

O Syllabus consiste em **seis capítulos**. Um capítulo abrange uma unidade educacional (UE). Para cada UE, são sugeridos tempo de ensino e tempo de prática. Eles são o mínimo que um curso deve investir para essa UE. As empresas de treinamento são livres para dedicar mais tempo às UEs e aos exercícios, mas certifique-se de que as proporções entre as UEs sejam mantidas. Os termos importantes utilizados em cada capítulo estão listados no seu início. Eles são definidos no glossário ou explicados no capítulo.

Exemplo: UE2 Um Início de Projeto Limpo (N2)
Duração: 120 minutos + 60 minutos de exercício
Termos: Visão do produto, Objetivo do produto, Parte interessada, Persona, Escopo do produto, Limite do sistema

Este exemplo mostra que o capítulo 2 contém objetivos educacionais do nível N2, e que 180 minutos são previstos para o ensino desse capítulo.

Cada capítulo pode conter subcapítulos. Seus títulos também contêm o nível cognitivo de seu conteúdo.

Os objetivos educacionais (OE) são enumerados antes do texto real. A numeração mostra a que subcapítulo eles pertencem.

Exemplo: OE 3.3.1 Dominando e usando o conceito INVEST de contar histórias de usuário
Este exemplo mostra que o objetivo educacional OE 3.3.1 é descrito no subcapítulo 3.3.

O Exame

Este programa é a base para o exame para o nível RE@Agile Advanced.

O conteúdo de uma questão pode abranger material de diversos capítulos do syllabus. Todos os capítulos (UE 1 até UE 6) deste syllabus são examináveis.

O formato do exame é de múltipla escolha, bem como um trabalho escrito avaliado; os detalhes estão estabelecidos no regulamento do exame.

Os exames de certificação podem ser realizados imediatamente após um curso preparatório ou também de forma independente (em uma entidade de certificação credenciada). A lista de organizações de certificação licenciadas pelo IREB encontra-se no site: http://www.ireb.org

Histórico de versões

Versão	Data	Comentário
2.0.0	30 de março de 2023	Versão Inicial baseada no Syllabus original do IREB em inglês

Conteúdo

Sobre este Módulo de Nível Avançado **RE@Agile:**

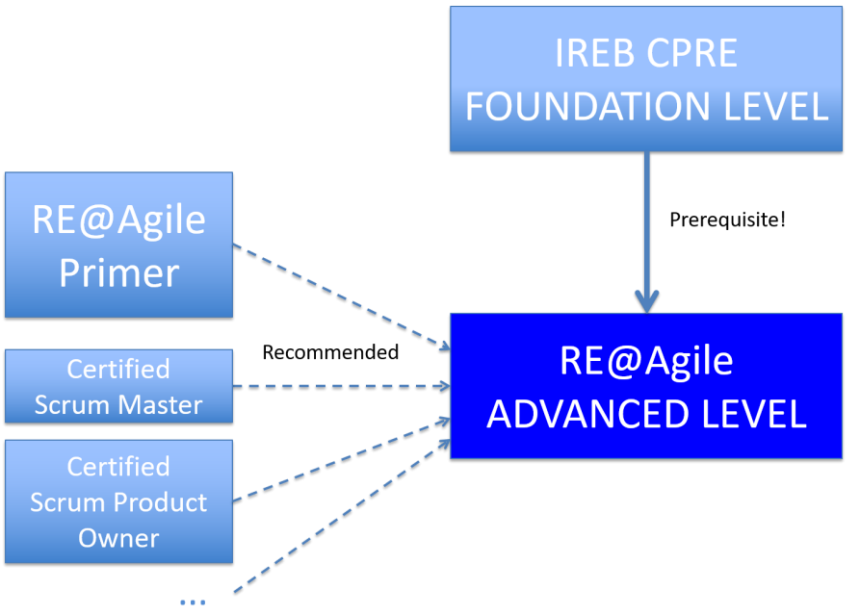
O módulo RE@Agile Advanced Level é para Engenheiros de Requisitos e profissionais Ágeis. Ele se concentra em entender e aplicar práticas e técnicas da disciplina de Engenharia de Requisitos em processos de desenvolvimento ágil, bem como entender e aplicar conceitos, técnicas e elementos de processo essenciais de abordagens ágeis em processos de Engenharia de Requisitos. Os titulares de certificados com conhecimentos de Engenharia de Requisitos poderão trabalhar em ambientes ágeis, enquanto os profissionais ágeis poderão aplicar práticas e técnicas comprovadas de Engenharia de Requisitos dentro de projetos ágeis.

Um detentor de certificado do **RE@Agile** Advanced Level:

- Está familiarizado com a terminologia de Engenharia de requisitos e agilidade
- 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
- Pode combinar abordagens ágeis e técnicas de Engenharia de Requisitos para benefício de todas as partes interessadas

Pré-requisitos

Como para qualquer outro Módulo Avançado do IREB, um pré-requisito para participar do exame do Nível Avançado **RE@Agile** é ter um certificado CPRE FL. Além disso, recomendamos fortemente ter pelo menos um dos certificados ágeis (ou seja, o RE@Agile Primer ou qualquer certificado Scrum) ou conhecimento similar sobre abordagens ágeis.



EU 1 O que é RE@Agile (N2)

Duração: 45 minutos

Termos: Cooperação das partes interessadas, Engenharia de Requisitos iterativa, Engenharia de Requisitos incremental, dono do produto

Objetivos educacionais
OE 1.1 Saber a definição de RE@Agile (N1)
OE 1.2 Entender os objetivos do RE@Agile (N2)
OE 1.3 Reconhecer que a responsabilidade por bons requisitos dentro da SCRUM é do dono do produto (N1)

O IREB define o RE@Agile [IREB2017] como uma abordagem cooperativa, iterativa e incremental com quatro objetivos:

1. Conhecer os requisitos relevantes em um nível de detalhe apropriado (a qualquer momento durante o desenvolvimento do sistema)
2. Alcançar acordo suficiente sobre os requisitos entre as partes interessadas relevantes
3. Capturar (e documentar) os requisitos de acordo com as restrições da organização
4. Realizar todas as atividades relacionadas a requisitos de acordo com os princípios do manifesto ágil

As ideais-chave desta definição estão em detalhes:

- RE@Agile é uma abordagem cooperativa:

  "Cooperativa" enfatiza a ideia ágil de interação intensiva das partes interessadas, inspecionando frequentemente o produto, fornecendo feedback sobre ele e adaptando e esclarecendo os requisitos, já que todos aprenderam mais [AgileManifesto2001].
- O RE@Agile é um processo iterativo:

  Isto sugere a ideia de requisitos "just in time". Os requisitos não precisam estar completos antes de iniciar o projeto técnico e a implementação. As partes interessadas podem definir (e refinar) iterativamente os requisitos que serão implementados em breve, até o nível de detalhe exigido [LaBai2003].
- O RE@Agile é um processo incremental:

  A implementação daqueles requisitos que oferecem o maior valor comercial, ou que mitigam os riscos mais significativos, deve formar o primeiro incremento. Os incrementos iniciais se esforçam para criar um produto mínimo viável (MVP) ou um produto mínimo comercializável (MMP). A partir daí, os incrementos subsequentes aumentam o produto, priorizando os requisitos que prometem entregar o maior valor comercial, aumentando assim o valor geral do resultado [Reinertsen2009].

O primeiro objetivo ("requisitos relevantes conhecidos no nível de detalhe apropriado") refere-se novamente à abordagem iterativa: "relevantes" são os requisitos que devem ser implementados em breve. E estes devem ser entendidos com muita precisão (incluindo seus critérios de aceite) - especialmente pelos desenvolvedores. Eles têm que estar de acordo com a "definição de preparado" (DoR). Outros requisitos - que ainda não são prioridade máxima - podem ser mantidos em um nível de abstração mais alto - apenas para serem mais detalhados assim que se tornarem mais importantes.

O pré-requisito para o segundo objetivo ("acordo suficiente entre as partes interessadas relevantes") é conhecer todas as partes interessadas e sua relevância para o sistema em desenvolvimento. Como pessoa responsável pelos requisitos (geralmente o dono do produto) você tem que negociar os requisitos com os interessados relevantes para determinar a ordem de sua implementação.

Abordagens ágeis sugerem a valorização da comunicação intensiva e contínua sobre os requisitos mais do que sua documentação. Entretanto, o terceiro objetivo enfatiza a importância da documentação em um nível de detalhe apropriado (que difere de situação para situação). Se uma organização tem que manter documentação sobre os requisitos (para fins legais, para rastreabilidade, para manutenção etc.), mesmo abordagens ágeis têm que garantir que este tipo de documentação seja produzido. No entanto, não precisa ser criada antecipadamente. Pode economizar tempo e esforço para criar essa documentação em paralelo à implementação, ou mesmo após a implementação.

O Gerenciamento de Requisitos resume todas as atividades a serem realizadas uma vez que você tenha requisitos existentes e artefatos relacionados aos requisitos. Isto inclui gerenciamento de versões e gerenciamento de configuração, bem como rastreabilidade entre os requisitos e rastreabilidade para outros artefatos de desenvolvimento. A RE@Agile sugere o gerenciamento cuidadoso do esforço gasto em tais atividades para equilibrar o custo com o benefício. Por exemplo:

- O gerenciamento extensivo de versões pode às vezes ser substituído por iterações rápidas que levam a incrementos de produto (ou seja, o histórico de mudanças de requisitos desde que foram encontrados pela primeira vez é menos interessante do que seu estado atual válido).
- O gerenciamento da configuração (baselining) está incluído na determinação iterativa dos backlogs de sprint, ou seja, agrupando aqueles requisitos que atualmente prometem o maior valor comercial.

Portanto, algumas das atividades de gerenciamento de requisitos que consomem tempo (e papel) de abordagens não ágeis são substituídas por atividades baseadas em princípios ágeis.

Como explicado no RE@Agile Primer [IREB2017], Scrum introduziu o papel de um dono de produto. Este dono de produto tem a responsabilidade de uma boa Engenharia de Requisitos em um ambiente ágil, embora outros interessados (como analistas de negócios, engenheiros de requisitos e os desenvolvedores) ajudarão o dono do produto nesse processo e talvez façam a maior parte do trabalho relacionado à Engenharia de Requisitos.

Neste syllabus - por uma questão de brevidade - nos referiremos ao dono do produto (como pessoa responsável) quando se tratar de executar tarefas relacionadas aos requisitos - não excluindo, portanto, de forma alguma, o trabalho realizado pelos outros interessados mencionados acima.

Os capítulos seguintes deste programa de estudos entrarão em mais detalhes sobre os vários aspectos do RE@Agile.

O Capítulo 2 discutirá os pré-requisitos para o desenvolvimento bem-sucedido do sistema: equilíbrio entre visões/metas, partes interessadas e escopo do sistema.

Os capítulos 3 e 4 discutirão o manuseio de requisitos funcionais, requisitos de qualidade e restrições em diferentes níveis de granularidade.

O Capítulo 5 discutirá o processo de estimativa, ordenação e priorização dos requisitos para determinar a sequência de incrementos.

Os capítulos 2 - 5 enfatizam principalmente o manuseio dos requisitos de uma única equipe de desenvolvedores (de 3 - 9 pessoas).

O Capítulo 6 discute como escalar a Engenharia de Requisitos para equipes maiores e potencialmente distribuídas, incluindo planejamento geral de produtos e road mapping.

EU 2 Um Início de Projeto Limpo (N3)

Duração: 120 minutos + 60 minutos de exercício

Termos: Visão do produto, Objetivo do produto, Parte interessada, Persona, Escopo do produto, Limite do sistema

Mesmo em abordagens ágeis, alguns pré-requisitos importantes devem ser estabelecidos antes que o trabalho de desenvolvimento iterativo e incremental do sistema possa ser iniciado com sucesso.

EU 2.1 Especificação de Visão e Objetivo (N3)

Duração: 30 minutos + 15 minutos de exercício

Objetivos educacionais
OE 2.1.1 Aplicar visão e especificação de objetivos (N3)

A visão do sistema ou visão do produto descreve o objetivo geral que deve ser alcançado com o sistema/produto. A literatura ágil frequentemente se refere à visão do produto em vez de uma visão do sistema. Ambos os termos (produto ou visão de sistema) são intercambiáveis e são usados dependendo do domínio ou contexto particular da atividade de desenvolvimento. A visão é de suma importância para cada atividade de desenvolvimento. Ela define a pedra angular e serve como uma direção geral para todas as atividades de desenvolvimento. Todas os requisitos devem apoiar a realização da visão do sistema [Scrumguide].

A diferença entre uma meta e um requisito pode ser definida da seguinte forma [Glinz2014]:

- Um objetivo é uma declaração sobre um estado de coisas desejado (que uma parte interessada deseja atingir).
- Um requisito é uma declaração sobre uma *propriedade* desejada *do sistema*.

Em certos domínios, o termo *problema* (que deve ser resolvido por um determinado sistema) é usado em vez do termo objetivo. Na perspectiva da RE, os termos estão diretamente relacionados: um problema é uma declaração sobre uma propriedade existente e indesejada, enquanto um objetivo expressa o estado futuro desejado.

Abordagens alternativas para a formulação de objetivos ou visões são:

- Formulação de metas SMART [Doran1981]. SMART é um acrônimo e significa Specific, Measurable, Achievable, Relevant, and Time-bound (Específico, Mensurável, Realizável, Relevante e Temporal).
- Formulação do objetivo PAM [Robertson2003]:
  - Qual é a finalidade (P - purpose)?
  - Qual é a vantagem comercial (A - advantage)?
  - Como mediríamos isso (M - measure)?

- Caixa de visão do produto / Projetar a caixa [Highsmith2001]. Crie um pacote para seu produto que mostre os principais benefícios/ideias de um produto para clientes potenciais.
- Notícias do futuro [HeHe2010]: Escreva um pequeno artigo de jornal que deve ser publicado no dia seguinte a um lançamento bem-sucedido do produto, descrevendo por que o produto é tão bom.
- Quadros de Visão: uma colagem gráfica de objetivos, potencialmente classificados por visões de curto, médio e longo prazo. Uma abordagem mais formal para os quadros de visão pode ser encontrada em [Pichler2011].
- Técnicas de Canvas [OsPi2010] como o Canvas de Geração de Modelos Comerciais ou o Canvas de Oportunidade (e muitas telas similares) oferecem técnicas para criar representações visuais focadas e fáceis de entender.

Qualquer que seja a forma escolhida, cada parte interessada tem o direito de saber pelo que a equipe está se esforçando. A definição da visão e dos objetivos iniciais deve, portanto, ocorrer no início de um esforço de desenvolvimento.

As mudanças na visão ou nos objetivos são raras, mas não impossíveis. Elas podem ocorrer, por exemplo, quando novos interessados são introduzidos ou porque os trabalhos no sistema revelam uma compreensão fundamentalmente nova do sistema ou de seu contexto. As mudanças na visão ou objetivos provavelmente terão um impacto significativo no desenvolvimento e devem ser tratadas com cuidado.

Uma opção é trabalhar com diferentes documentos de visão, cada uma cobrindo um horizonte de tempo diferente para o futuro. Em qualquer momento em particular, uma visão válida e precisa de seis meses, um ano e dois anos, acordada com todos os interessados relevantes, pode estar disponível para os desenvolvedores. Estes documentos de visão são revisados e atualizados em pontos regulares do ciclo.

Também não é raro focar na realização de um subconjunto particular de metas por um certo período de tempo (por exemplo, 6 meses) e depois mudar as metas para seguir outra direção. Mudanças muito frequentes da visão/ou das metas, entretanto, são um indicador de que não há uma direção clara que impulsione o desenvolvimento do produto [Reinertsen2009].

EU 2.2 Especificando o limite do sistema (N3)

Duração: 30 minutos + 15 minutos de exercício

Objetivos educacionais
OE 2.2.1 Aplique a especificação dos limites do sistema para delimitar o escopo do contexto (N3)

Um entendimento comum e compartilhado do escopo (de acordo com a IREB: "a gama de coisas que podem ser moldadas e projetadas ao desenvolver um sistema") e do contexto (de acordo com a IREB "a parte do ambiente do sistema relevante para a compreensão do sistema e seus requisitos") do sistema é um pré-requisito para um esforço de desenvolvimento eficaz e eficiente [Glinz2014].

Os mal-entendidos relacionados aos limites do sistema podem levar a:

- O desenvolvimento de funcionalidades ou componentes que não estavam sob a responsabilidade do esforço de desenvolvimento (o escopo assumido era muito grande)
- A falsa suposição de que as funcionalidades ou componentes que são de fato parte do sistema deveriam ter sido desenvolvidos fora do sistema (o escopo suposto era muito pequeno)

A definição do escopo e do contexto se conjugam na definição do limite do sistema e do limite do contexto. O sistema e o limite do contexto podem ser definidos através de discussões:

- Quais características ou funcionalidades devem ser fornecidas pelo sistema e quais devem ser fornecidas pelo contexto? (definição dos limites do sistema)
- Quais interfaces técnicas ou de usuário devem ser fornecidas pelo sistema para o contexto? (definição dos limites do sistema)
- O que é irrelevante para o sistema, ou seja, não influencia de forma alguma o seu escopo?
- Quais aspectos do contexto do sistema podem ser modificados durante o desenvolvimento do sistema? (definição do escopo)

Tenha em mente que o limite do contexto é sempre incompleto porque o limite do contexto só pode ser definido pelas coisas que se excluem explicitamente do contexto do sistema. Outros aspectos do contexto do sistema incluem, em particular, as partes interessadas. A identificação e gestão das partes interessadas é descrita em EU 2.3.

Os limites do sistema podem ser documentados e esclarecidos com várias técnicas, por exemplo:

- Um diagrama de contexto: documenta o sistema, elementos do contexto e sua relação
- Linguagem natural, ou seja, uma lista de características, funcionalidades, aspectos do contexto, aspectos do escopo e outros aspectos fora do contexto
- Um diagrama de caso de uso: um tipo de diagrama UML que modela os atores e os casos de uso de um sistema. Ele descreve o contexto/escopo em um nível detalhado e é especialmente útil para esclarecer o escopo e o contexto.
- Um mapa de histórias: uma disposição bidimensional das histórias de usuários em um modelo útil para ajudar a compreender a funcionalidade do sistema. Ele descreve o contexto/âmbito em um nível detalhado e é especialmente útil para esclarecer o escopo.

A definição de um escopo inicial deve ocorrer no início de um esforço de desenvolvimento. O escopo e o contexto inevitavelmente mudarão durante um esforço de desenvolvimento devido a um novo ou alterado entendimento do sistema ou contexto. A documentação do escopo e do contexto deve, portanto, ser atualizada regularmente.

EU 2.3 Identificação e Gestão de Partes Interessadas (N3)

Duração: 30 minutos + 15 minutos de exercício

Objetivos educacionais
OE 2.3.1 Aplicar a identificação e gestão das partes interessadas (N3)

A incapacidade de identificar e incluir uma parte interessada importante em um esforço de desenvolvimento pode ter um grande impacto: Descobrir tardiamente (ou faltar completamente!) os requisitos de uma parte interessada podem levar a mudanças caras ou a um sistema inútil [PoRu2015].

O *modelo de cebola* de Ian Alexander [Alexander2005] é uma ferramenta simples para a identificação e classificação das partes interessadas. O modelo consiste em três tipos de partes interessadas: Participantes do sistema, participantes do contexto circundante, e participantes do contexto mais amplo. As partes interessadas do sistema também são chamadas de *partes interessadas diretas*. As partes interessadas do contexto circundante e mais amplo também são chamadas de *partes interessadas indiretas*.

Se um sistema tem um usuário humano, o usuário é um dos mais importantes interessados diretos.

Os usuários de um sistema normalmente cobrem um amplo espectro de pessoas com expectativas, atitudes e pré-requisitos diferentes. A compreensão do espectro de usuários para um determinado sistema é um primeiro passo importante. Se o número de usuários for pequeno, é aconselhável conhecê-los (ou a seus representantes) através de entrevistas pessoais. Se o número de usuários for grande ou mesmo desconhecido, o espectro de usuários deve ser capturado com outros meios, por exemplo, personas [Cooper2004] que representam usuários exemplares com características extremas. Na era da análise de dados, análise google e grandes dados, o comportamento do usuário online muitas vezes pode ser analisado diretamente para incrementos de produtos implantados com ferramentas automatizadas.

As partes interessadas indiretas podem ser encontradas no contexto circundante do sistema e podem ser tão importantes para um esforço de desenvolvimento quanto os próprios usuários. Exemplos incluem representantes legais, órgãos governamentais ou de padronização, organizações de auditoria para verificação de conformidade em mercados regulados (médico, transporte, aviação), ou de forma ainda mais ampla ONGs, sindicatos ou concorrentes.

A identificação sistemática das principais partes interessadas deve ocorrer no início de um esforço de desenvolvimento e os resultados devem ser gerenciados ao longo do esforço de desenvolvimento como uma atividade contínua. Uma lista simples, incluindo detalhes de contato e atributos relevantes, será suficiente na maioria dos contextos. As mudanças nesta lista podem ocorrer a qualquer momento, seja porque uma parte interessada foi inicialmente negligenciada ou devido a mudanças no contexto, tais como a criação de uma nova ONG. Cada participante de um esforço de desenvolvimento (por exemplo, desenvolvedores e donos de produtos) deve estar ciente da importância das partes interessadas e procurar sinais de novos ou ausentes.

Dependendo do sistema específico e do domínio, a documentação existente e os sistemas legados também podem ser fontes importantes de requisitos, e estes devem ser sistematicamente identificados e gerenciados de forma similar às partes interessadas, ver [IREB2019].

EU 2.4 Equilíbrio de Visão e Metas, Partes Interessadas e Escopo (N3)

Duração: 30 minutos + 15 minutos de exercício

Objetivos educacionais
OE 2.4.1 Aplique o equilíbrio de visões e objetivos, partes interessadas e escopo (N3)

A definição da visão e dos objetivos, das partes interessadas e dos limites do sistema dependem uns dos outros [PoRu2015]:

- As partes interessadas relevantes formulam a visão e os objetivos. Portanto, a identificação de um novo interessado relevante pode ter um impacto sobre a visão ou os objetivos.
- A visão e as metas podem ser usadas para orientar a identificação de novos interessados perguntando: Quais partes interessadas podem estar interessadas em alcançar a visão/os objetivos ou são afetadas por alcançar a visão/os objetivos?
- Visão e objetivos podem ser usados para definir um escopo inicial, perguntando: Que elementos são necessários para alcançar a visão/os objetivos?
- A mudança dos limites do sistema (e, portanto, do escopo) pode ter um impacto sobre a visão/os objetivos. Se um aspecto for removido do escopo, é preciso verificar se o sistema ainda tem meios suficientes para atingir a visão/os objetivos.
- As partes interessadas definem os limites do sistema. Portanto, a identificação de um novo interessado relevante pode ter um impacto sobre o escopo.
- Uma mudança do escopo (por exemplo, para cumprir uma meta) requer o acordo das partes interessadas relevantes.

Estas interdependências devem ser usadas para equilibrar os três elementos e para examinar o impacto da mudança de um dos três elementos sobre o outro.

Devido a estas estreitas dependências entre visão e objetivos, partes interessadas e escopo, recomendamos tratar todos estes elementos em conjunto e de forma coerente.

EU 3 Lidando com Requisitos Funcionais (N4)

Duração: 195 minutos + 120 minutos de exercício

Termos: Requisitos funcionais, tema, épico, característica, história do usuário, critérios de aceite, técnicas de divisão e agrupamento, Definição de Preparado (DoR), INVEST

Este núcleo da UE de Nível Avançado tem uma **visão** principalmente **estática dos requisitos funcionais**, ou seja, **estruturando um conjunto maior de requisitos** em hierarquias de abstração.

EU 3.1 Diferentes níveis de granularidade de requisitos (N1)

Duração: 15 minutos

Termos: Granularidade de requisitos, hierarquias de requisitos

Objetivos educacionais
OE 3.1.1 Saber que existem requisitos funcionais em diferentes níveis de granularidade (N1)

As partes interessadas geralmente comunicam os requisitos em diferentes níveis de granularidade. Algumas vezes eles pedem grandes pedaços de funcionalidade, outras vezes eles pedem detalhes menores para serem acrescentados ou alterados. É função do dono do produto (apoiado por outras partes interessadas) elucidar todos esses requisitos e estruturá-los [Scrumguide].

Ambos, requisitos de grãos grosseiros e requisitos de grãos finos são úteis. Os requisitos de grãos grosseiros permitem ao dono do produto manter uma visão geral de todos os requisitos funcionais. Isto é especialmente necessário para planejamento a longo prazo, road-mapping, seleção de requisitos que prometem valor comercial precoce para uma investigação mais aprofundada, estimativa de alto nível e muito mais.
Os requisitos de granularidade fina são necessários para se obter uma compreensão completa dos detalhes necessários para que os desenvolvedores implementem esses requisitos [Patton2014].

Dentro do Ágil, os termos tema, épico e característica são usados para representar níveis distintos de granularidade, embora ainda não tenha surgido um consenso claro quanto à sua ordem exata. Muitas vezes é preferível a hierarquia Épico-Característica-História de Usuário, com partes da hierarquia agrupadas por temas. Veja também a Figura 1 em [IREB2017] no Capítulo 3.1.5 e a discussão no próximo capítulo.

EU 3.2 Identificação, documentação e comunicação dos requisitos funcionais (N4)

Duração: 45 minutos + 30 minutos de exercício

Termos: Baseado em valores e design e agrupamentos de requisitos, épico, tema, característica, histórias, abordagem em T

Objetivos educacionais

OE 3.2.1 Analisar e dominar uma decomposição de nível superior dos requisitos que suportam o planejamento de iteração e road-mapping (N4)

OE 3.2.2 Analisar e dominar diferentes estratégias de decomposição em grande escala (N4)

OE 3.2.3 Analisar e dominar a Identificação, documentação e comunicação de requisitos funcionais em diferentes níveis de granularidade (N4)

RE@Agile esforça-se por requisitos "just in time" [IREB2017]. Para determinar quais requisitos devem ser definidos com detalhes suficientes, é necessária uma visão geral.

A visão e os objetivos podem não ser suficientes para tomar tais decisões. Portanto, um objetivo intermediário para o dono de um produto é apresentar uma visão geral de todos os requisitos dentro do escopo de um sistema, ou seja, cobrir toda a amplitude sem muita profundidade. Isto é frequentemente chamado de abordagem T, já que esta ampla visão geral se assemelha à barra horizontal da letra T, enquanto o detalhamento daqueles requisitos que prometem o maior valor comercial se assemelha à barra vertical da letra T.

Diferentes critérios podem ser usados para decompor a visão ou as metas a fim de produzir um conjunto de requisitos de alto nível que - considerados em conjunto - abranjam o escopo pretendido. Muitos métodos sugeriram a utilização de uma decomposição orientada a processos [GoWo2006] [Lamsweerde2009], ou seja, a divisão da funcionalidade em processos comerciais, casos de uso ou grandes histórias. Este tipo de decomposição preenche os três primeiros critérios do INVEST, conforme discutido no próximo capítulo, ou seja, os processos resultantes são independentes, negociáveis e - o mais importante - valiosos.

Estratégias alternativas de decomposição poderiam ser baseadas em objetos comerciais, na decomposição de subsistemas de um sistema existente (ou seja, uma abordagem baseada em histórico ou em design), distribuição de hardware ou distribuição geográfica de subsistemas.

Enquanto os resultados de uma decomposição orientada para o processo são chamados de casos de uso ou histórias de usuários, os pedaços resultantes das decomposições alternativas são frequentemente chamados de épicos, temas ou características.

Quaisquer que sejam os grandes pedaços de funcionalidade, eles fornecem uma boa base para estimativa (grosseira) e já podem ser priorizados, criando assim um roadmap ou um cronograma preliminar de iterações (ou sprints - na terminologia Scrum).

Todos esses requisitos de alto nível devem ser detalhados antes de serem implementados pelos desenvolvedores (ver os próximos capítulos).

Para comunicação e documentação histórias, épicos, características ou temas são muitas vezes capturados em cartões físicos, coletados no backlog do produto. Alternativamente, muitas ferramentas estão disponíveis para capturar esses requisitos eletronicamente. Assim que os requisitos de nível superior são refinados em um conjunto de requisitos de nível inferior, são criadas hierarquias de requisitos. O ideal é que o dono do produto possa gerenciar os diferentes níveis, sem perder os agrupamentos de nível superior após o refinamento.

Story Maps [Patton2014] são uma forma de organizar e visualizar cartões em duas dimensões, para que os épicos e as características ainda sejam visíveis, mesmo que tenham sido refinados para histórias de usuários.

EU 3.3 Trabalhando com histórias de usuários (N3)

Duração: 30 minutos + 30 minutos de exercício

Termos: História do usuário, critérios INVEST, Princípio 3C, critérios de aceite

Objetivos educacionais
OE 3.3.1 Aplique os conceitos de contar histórias de usuários do INVEST (N3)
OE 3.3.2 Aplique a extensão ou alinhamento de temas, épicos, características e histórias de usuários com artefatos adicionais de requisitos para melhorar a comunicação com as partes interessadas (N3)

As histórias de usuários são uma abordagem popular para estruturar os requisitos. [Cohn2004] sugere uma fórmula para captar histórias de usuários. Os três constituintes desta fórmula asseguram que

1. temos alguém que quer essa funcionalidade ("Como usuário ..."),
2. nós sabemos o que o usuário quer ("... eu quero ...") e
3. entendemos a razão ou motivação ("...").

A fórmula ajuda a pensar sobre quem quer o quê e por que, mas não é tanto o formalismo que faz com que as histórias dos usuários tenham sucesso, mas a pergunta e a resposta a estas três perguntas.

Bill Wake criou "INVEST" [Wake2003] como um bom acrônimo para discutir características de uma história de usuário. O significado das 6 letras é o seguinte:

- **I**: as histórias dos usuários devem ser *independentes*. Ou seja, deve ser possível implementar histórias de usuários em qualquer ordem, com o mínimo de dependências entre elas.
- **N**: histórias de usuários devem ser consideradas não como um contrato escrito, mas sim como uma promessa para futuras *negociações,* ou uma lembrança de discussões passadas. No papel de Engenheiro de Requisitos, os profissionais são facilitadores qualificados desta negociação entre especialistas em negócios e desenvolvedores para explorar o significado da história do usuário.

- **V**: toda história de usuário deve criar *valor*. Mais detalhes sobre o valor comercial podem ser encontrados em EU 5.
- **E**: uma história de usuário deve ser *estimável. Se o* esforço necessário para sua implementação não puder ser estimado, a história ainda não está suficientemente clara. Mais detalhes sobre estimativa seguem em EU 5.
- **S**: as histórias de usuários (no final) devem ser *pequenas* o suficiente para serem desenvolvidas dentro de uma sprint e ainda fornecer valor para a empresa ou outros interessados. Os donos de produtos têm que dividir as histórias em pedaços tão pequenos para permitir que os desenvolvedores terminem essa história em uma única iteração. As técnicas de divisão de histórias são discutidas em EU 3.4.
- **T**: As histórias de usuários devem incluir critérios de aceite para que *possam ser testadas*. Tais critérios são tipicamente escritos antes do desenvolvimento da história. As técnicas para especificar os critérios de aceite são discutidas em EU 3.5. Os donos de produtos devem entender as armadilhas das histórias de usuários não-testáveis e ser capazes de reformulá-las em histórias de usuários testáveis.

As histórias são geralmente capturadas em cartões. O princípio 3C (cartão, conversa, confirmação) [Jeffries2001] enfatiza que o cartão é apenas um meio para um fim - um símbolo físico em forma tangível e durável do que de outra forma seria apenas uma abstração. O cartão não deve ser considerado como equivalente a uma declaração de requisitos bem redigida. O *cartão* físico (ou seu equivalente em uma ferramenta) está lá para desencadear uma *conversa* sobre esse assunto, para reunir os donos do produto, outras partes interessadas e os desenvolvedores para discutir esse cartão, obter um entendimento mais profundo e esclarecer questões em aberto. Não importa a quantidade de conversa que ocorra, a *confirmação* é o teste ácido para essa história: geralmente os testes de aceite anexos a essa história, ou seja, as coisas que o dono do produto verificará quando a equipe afirmar ter terminado a implementação da história.

Embora o Ágil enfatize o uso de histórias de usuários, ele não impede o uso de artefatos tradicionais de análise como diagramas de contexto, casos de uso comercial, casos de uso de sistemas, modelos de objetos, diagramas de estados, diagramas de atividades e assim por diante, quando apropriado no contexto de um determinado projeto (cf., por exemplo, [Scrumguide]). O dono qualificado do produto deve entender como as histórias de usuários podem ser usadas juntamente com outros artefatos, e ser capaz de explicar seu significado onde for necessário para as partes interessadas.

EU 3.4 Técnicas de divisão e agrupamento (N4)

Duração: 45 minutos + 30 minutos de exercício

Termos: História composta, corte vertical, corte horizontal, padrões de divisão, padrões de agrupamento e abstração, mapa da história, spike

Objetivos educacionais

OE 3.4.1 Analisar e Dominar técnicas de particionamento para requisitos funcionais de granularidade grosseira para obter requisitos mais finos, mais precisos (N4)

OE 3.4.2 Analisar e agrupar ou abstrair os requisitos funcionais finos em requisitos mais grosseiros para lidar com a complexidade, ter uma melhor visão geral e fazer um planejamento de nível superior (N4)

A fim de gerar histórias de usuários que sejam pequenas o suficiente para caber em uma única sprint, histórias maiores podem ser quebradas em histórias de grão mais fino. Diversos padrões podem ser aplicados para este fim, desde a redução da lista de características até a redução das variações comerciais ou canais de entrada [Leffingwell2010]. Quando uma história de usuário apresenta um nível inaceitável de incerteza, spikes podem ser usados para concentrar os desenvolvedores no problema durante uma sprint dedicada.

Mesmo as histórias de usuários finas devem, no entanto, ser definidas de tal forma que tragam algum valor para pelo menos uma parte interessada.

A decomposição das histórias de usuários resultará em hierarquias de requisitos, conforme discutido em EU 3.1. Esta estrutura hierárquica e tridimensional pode ser visualizada como um mapa de história bidimensional [Patton2014]. A dimensão horizontal mostra agrupamentos maiores (como grandes histórias, épicos, características) mantendo assim uma visão geral dos requisitos; a dimensão vertical permite anexar todos os detalhes de nível inferior para os grupos maiores e ordená-los para atribuição a sprints e releases.

Tal decomposição e agrupamento de requisitos é frequentemente melhor realizada com outros membros da equipe que podem ter uma visão maior das dependências comerciais e técnicas que podem existir entre as histórias de usuários.

EU 3.5 Quando você deve parar de se decompor (N4)?

Duração: 45 minutos + 30 minutos de exercício

Termos: Definição de Preparado (DoR), Definição de Pronto (DoD), estimativa, testes de aceite

Objetivos educacionais

OE 3.5.1 Analisar e dominar o refinamento dos requisitos até um nível adequado para a implementação desses requisitos (N4)

OE 3.5.2 Analisar e Dominar a garantia de qualidade das histórias de usuários durante o desenvolvimento de software ágil (N4)

O dono do produto é responsável por continuar as discussões com os desenvolvedores até que ambos os lados tenham um entendimento comum dos requisitos [Meyer2014]. O princípio de Pareto pode ser usado para avaliar quando este ponto for atingido: os requisitos não devem ser definidos 100% perfeitamente, mas o suficiente para responder às perguntas-chave da equipe e de forma clara o suficiente para que o esforço de implementação possa ser estimado. Iniciar a implementação com demasiadas questões em aberto pode reduzir consideravelmente a velocidade de desenvolvimento e causar atrasos em relação às previsões.

Para este nível de entendimento conjunto, o Ágil definiu a Definição de Preparado (DoR) [AgileAlliance]. Uma história está pronta quando cumpre os critérios do INVEST [Wake2003], especialmente as três últimas letras:

- Os desenvolvedores foram capazes de estimar a história e o valor estimado é pequeno o suficiente para permitir que a história se encaixe em uma iteração. Lawrence sugere que a história não deve caber apenas em uma iteração, mas deve ser tão pequena que 6 - 10 histórias podem ser atribuídas à próxima iteração.
- O dono do produto forneceu critérios de aceite para a história. Com base no princípio do CCC, todos concordam que já houve *conversas* suficientes e que foram definidos critérios para a *confirmação* do sucesso em termos de testes de aceite.

Para a formulação de critérios de aceite estão disponíveis diferentes estilos [Beck2002]. Podem ser sentenças informais em linguagem natural a serem verificadas após a implementação. Eles poderiam ser um pouco mais formais usando a sintaxe Gherkin (Given/When/Then) para estruturar estas sentenças.

Alguns métodos defendem até o uso de TDD (Test Driven Development), codificando formalmente os casos de teste para que eles possam ser executados automaticamente após a implementação [Meyer2014]. Esta abordagem formal - embora muito precisa - pode ser difícil de ser feita e compreendida pelos donos de produtos e pelas partes interessadas orientadas ao negócio.

Para o dono do produto, o DoR é o equivalente à definição de pronto (DoD) dos desenvolvedores. DoD define critérios para determinar se uma história foi implementada com sucesso enquanto DoR define que os desenvolvedores têm informações suficientes sobre uma história de usuário para que ela possa ser "feita" pelos desenvolvedores dentro de uma sprint.

A discussão dos requisitos com os desenvolvedores precisa de tempo e é melhor fazer antes do planejamento da sprint. O planejamento pode então se concentrar na seleção das histórias corretas dos usuários e atribuí-las aos desenvolvedores responsáveis. Idealmente, os desenvolvedores verão os requisitos evoluir e terão ajudado o dono do produto, fazendo perguntas e fazendo estimativas.

Diferentes formas de refinamento do backlog de produtos são possíveis. As reuniões de refinamento podem, por exemplo, ser uma forma mais eficiente de realizar refinamentos do que perturbar repetidamente os desenvolvedores de forma individual. O refinamento do backlog do produto e todas as atividades ao redor consomem tempo a partir da capacidade geral da iteração. O guia Scrum recomenda um máximo de 10% de capacidade dos desenvolvedores para refinamento: se for necessário mais tempo do que esse, a iteração deve ser replanejada para obter mais precisão no backlog do produto, pois isso é um sinal de alerta para a má qualidade dos requisitos. O dono do produto deve entender a relação entre a duração da iteração, o risco e a sobrecarga de iteração, e saber que existem loops de feedback mais curtos do que a própria sprint.

EU 3.6 Documentação de requisitos do projeto e do produto (N2)

Duração: 15 minutos

Termos: documentação do projeto e do produto, razões para preservar a documentação

Objetivos educacionais
OE 3.6.1 Entenda como distinguir entre informação/documentação do projeto e do produto (N2)
OE 3.6.2 Métodos e técnicas para preservar a informação para uso futuro (N1)

Para a equipe do projeto, os cartões de histórias são lembretes suficientes das discussões em andamento para servir de base para o desenvolvimento [Cohn2004]. Para os desenvolvedores que têm estado continuamente envolvidos em um projeto, muitas vezes o próprio código é suficiente para entender o que foi feito anteriormente.

Além daqueles diretamente envolvidos no desenvolvimento do projeto, porém, há muitos outros interessados - o cliente, a empresa em geral, as equipes de suporte, equipes de projeto relacionadas etc. – que o código e as histórias de usuários não fornecem contexto e estrutura suficientes para entender como o trabalho terá impacto sobre eles. Tais atores representam, portanto, públicos-alvo diferentes para a documentação.

Além disso, o produto que resulta de um projeto de desenvolvimento terá (espera-se) uma vida além do projeto, e poderá de fato evoluir ao longo de vários projetos de desenvolvimento. Os artefatos de engenharia de requisitos normalmente podem ser classificados como relevantes apenas para o projeto, ou como parte da documentação do produto que deve ser preservada para uso futuro. Separar as preocupações com produtos das preocupações com projetos pode representar um bom investimento em documentação sustentável.

De acordo com os princípios Ágeis, o esforço só deve ser gasto em documentação quando essa documentação tem um consumidor, e é importante obter um feedback regular desse consumidor ao desenvolver esse artefato. As técnicas para minimizar o esforço de documentação incluem a criação de sistemas de documentação dedicados e orientados a wikis, em particular para artefatos orientados a produtos que irão evoluir com o tempo [Weerd et al.2006].

EU 4 Lidando com Requisitos e Restrições de Qualidade (N3)

Duração: 90 minutos + 30 minutos de exercício

Termos: Requisitos de Qualidade, Restrições, Árvore de Qualidade, DoD

Esta UE analisa os **requisitos e restrições de qualidade em projetos ágeis**. Embora o termo "requisitos não-funcionais" ainda seja utilizado na prática como um termo guarda-chuva, utilizamos as categorias mais concretas e precisas "requisitos de qualidade" e "restrições" de acordo com [Glinz2014].

Os requisitos iniciais de qualidade são muitas vezes deliberadamente vagos. Eles têm que ser capturados em seu formato vago como um ponto de partida. Requisitos de qualidade e restrições vagos podem ser refinadas em requisitos mais precisos. Algumas vezes, requisitos funcionais concretos serão derivados deles.

EU 4.1 Entendendo a importância dos requisitos e restrições de qualidade (N2)

Duração: 15 minutos

Objetivos educacionais
OE 4.1.1 Entenda a importância dos requisitos de qualidade em projetos ágeis (N2)
OE 4.1.2 Esquemas de categorização de acordo com os requisitos e restrições de qualidade (N2)

Muitos métodos ágeis concentram-se apenas nos requisitos funcionais e não dão ênfase suficiente às qualidades e restrições [Meyer2014].

As principais restrições e qualidades previstas para o sistema devem ser explicitadas no início do ciclo de vida de um produto, uma vez que determinam as escolhas arquitetônicas fundamentais (infraestrutura, arquitetura de software e projeto de software). Ignorá-los ou aprender muito tarde no projeto pode colocar em perigo todo o esforço de desenvolvimento. Outras qualidades podem ser capturadas iterativamente, sob demanda, como com os requisitos funcionais [Meyer2014].

Os esquemas de categorização para requisitos e restrições de qualidade (por exemplo [RoRo2012], [ISO25010]) podem ser usados como listas de verificação para não esquecer categorias importantes.

EU 4.2 Acrescentando Precisão aos Requisitos de Qualidade (N2)

Duração: 30 minutos

Objetivos educacionais
OE 4.2.1 Entenda os detalhes ou a decomposição dos requisitos e restrições de qualidade (N2)
OE 4.2.2 Entenda a derivação dos requisitos funcionais a partir dos requisitos de qualidade (N2)
OE 4.2.3 Entenda a especificação dos critérios de aceite dos requisitos de qualidade (N2)
OE 4.2.4 Entenda o valor agregado de árvores de qualidade (N2)

Os requisitos de qualidade devem ser comunicados aos desenvolvedores de forma inequívoca e que suporte o refinamento e a decomposição dos requisitos, da mesma forma que para os requisitos funcionais.

Decompor (ou detalhar) um requisito de qualidade significa especificar requisitos de qualidade em um nível inferior de detalhe, por exemplo, utilizando as generalizações nos esquemas de categorização como "usabilidade" e tornando-os mais precisos ao encontrar requisitos de "facilidade de uso" e "facilidade de aprendizagem".

Derivar significa que os requisitos de qualidade podem ser alcançados definindo requisitos funcionais, ou seja, sugerindo funções que atinjam a qualidade desejada ou restrições. Um exemplo para refinar um requisito de segurança é a introdução de um conceito de papel e senhas. Árvores de qualidade ([BOSSANOVA], [Clements et al.2001]) também são uma forma comprovada de estruturar os requisitos de qualidade.

Para os requisitos de qualidade, os critérios de aceite também precisam ser definidos.

Assim como outros tipos de requisitos, os requisitos de qualidade devem ser testáveis [PoRu2015].

The type of acceptance criteria used will depend on the category of the quality: a quantifiable acceptance criterion for a usability requirement, for example, might be “90 out of 100 users must be able to enter an invoice in less than three minutes”.

EU 4.3 Requisitos de Qualidade e Backlog (N3)

Duração: 30 minutos + 30 minutos de exercício

Objetivos educacionais
OE 4.3.1 Aplicar os requisitos de qualidade aos requisitos funcionais, se aplicável (N3)
OE 4.3.2 Aplicar criando itens de backlog separados para os requisitos de qualidade (N3)
OE 4.3.3 Abordar outros requisitos de qualidade como parte do DoD (N2)

Requisitos de qualidade generalizados precisam estar ligados a requisitos funcionais mais específicos [PoRu2015], por exemplo, algum rendimento quantificável ligado a uma história de usuário, ou recursos de segurança específicos ligados a um épico.

Outras qualidades, por exemplo, escalabilidade, capacidade de manutenção ou robustez devem ser transmitidas ao desenvolvimento e verificadas em cada iteração. Uma maneira comum de conseguir isso é incluí-los na Definição de Pronto. Isto é frequentemente apoiado por testes automatizados [Leffingwell2010].

Outra abordagem é ter um registro separado (diferente do backlog do produto) de tais qualidades para mantê-los visíveis para as equipes, por exemplo, como uma lista comum ou na forma de listas de verificação; tais requisitos estão todos na mesma ordem (porque todos devem ser cumpridos) [Leffingwell2010].

Também é uma boa prática tornar visíveis os relacionamentos entre requisitos funcionais e de requisitos de qualidade afetados através da instalação de uma matriz em uma parede, indicando a relação "afetada por" com marcas nas respectivas células.

EU 4.4 Tornando as Restrições Explícitas (N2)

Duração: 15 minutos

Objetivos educacionais
OE 4.4.1 Entenda diferentes tipos de restrições em projetos ágeis (N2)
OE 4.4.2 Caracterizar a reutilização das restrições (N1)

As restrições são um tipo importante de requisitos que limitam as escolhas de projeto dos desenvolvedores [Glinz2014]. As restrições podem ser categorizadas como restrições de produto ou restrições de processo. As restrições do produto incluem o uso necessário de tecnologias, reutilização de componentes existentes, tomar ou comprar decisões ou recursos na forma de material, conhecimento e competência, enquanto as restrições do processo prescrevem tanto a organização quanto os processos de desenvolvimento. Estes incluem políticas e regulamentos organizacionais, limites financeiros, normas e padrões, regulamentos e auditorias de conformidade, restrições legais e governamentais.

É importante explicitar tais restrições para que todos na equipe estejam cientes delas. As mais limitantes devem ser conhecidas no início do projeto. Outras devem ser capturados assim que forem descobertas.

Tais restrições são normalmente aplicáveis a uma gama mais ampla de projetos. Logo, assim que são capturadas uma vez, podem ser reutilizadas facilmente.

EU 5 Priorização e Estimativa dos Requisitos (N3)

Duração: 120 minutos + 90 minutos de exercício

Termos: Ordenação e priorização do Backlog, valor comercial, MVP, MMP, ROI, custo do atraso, WSJF, tempo de comercialização, Planning Poker, Muro de estimativa, Cone de incerteza, Velocidade, Dimensionamento, histórias de referência, T-Shirt-Sizing, Sequência de Fibonacci

EU 5.1 Determinação do valor comercial (N3)

Duração: 45 minutos + 15 minutos de exercício

Objetivos educacionais
OE 5.1.1 Entenda a determinação do valor comercial (N2)
OE 5.1.2 Entenda como usar o valor comercial para priorizar itens de backlog (N2)
OE 5.1.3 Aplicar cálculos alternativos para o valor comercial (N3)
OE 5.1.4 Entender como alinhar a medição do valor comercial às metas estratégicas da organização (N2)
OE 5.1.5 Entender o conceito de MVP (produto mínimo viável) (N2)
OE 5.1.6 Entender o conceito de MMP (produto mínimo comercializável) (N2)

As abordagens ágeis visam maximizar o valor geral do negócio e otimizar permanentemente o processo geral de criação de valor comercial [Leffingwell2010]. Todos os requisitos (sejam eles grosseiros ou finos) devem ser priorizados principalmente pelo valor que podem trazer para a organização. Um pré-requisito para fazer isso é uma definição acordada do valor comercial para este produto/empresa.

O valor comercial não é definido apenas pelo lucro: os cálculos alternativos incluem Retorno do Investimento, Período de Retorno do Investimento, Valor Presente Líquido, Emprego Mais Curto Ponderado Primeiro (WSJF), Custo do Atraso e Balanced Scorecard. Valor de mercado, tempo para comercializar e redução de riscos potenciais, todos representam potencialmente tipos de valor comercial, assim como a excelência operacional e organizacional [Reinertsen2009].

De fato, a definição de valor comercial pode ser diferente em cada organização, cada projeto, e da perspectiva de diferentes partes interessadas. Os profissionais devem entender como alinhar a medição do valor comercial aos objetivos estratégicos da organização, e ser capazes de adaptar este alinhamento à medida que estes objetivos mudam.

Uma técnica para identificar o valor comercial dentro dos requisitos é o modelo KANO. Em geral, o dimensionamento relativo do valor comercial às vezes é preferível ao cálculo do valor comercial absoluto.

Os termos chave dentro do Ágil são MVP (produto mínimo viável) e MMP (produto mínimo comercializável) [IREB2017]. O MVP pode reduzir drasticamente o esforço enquanto o valor comercial permanece aproximadamente o mesmo. Os MVP's podem ser definidos não apenas para o produto inteiro, mas também para uma característica particular.

Embora um MVP seja ótimo para validar o aprendizado, não é suficiente para uma operação a longo prazo; para uma implantação real no negócio, o MMP é importante. MVP é principalmente sobre redução de escopo: produzir o valor comercial com o mínimo possível de recursos e expandir mais tarde se o recurso for usado com sucesso.

EU 5.2 Valor de Negócio, Riscos e Dependências (N3)

Duração: 45 minutos + 45 minutos de exercício

Objetivos educacionais
OE 5.2.1 Aplicar priorização do backlog (N3)
OE 5.2.2 Aplicar diferentes estratégias básicas de priorização (N3)
OE 5.2.3 Aplicar a determinação das dependências de requisitos (N3)
OE 5.2.4 Entender a modificação da prioridade dos itens de backlog, considerando as dependências (N2)
OE 5.2.5 Entender as dependências entre o valor comercial potencial e os riscos relacionados (N2)

Diferentes estratégias básicas de priorização nos diferentes tipos de Backlog (empresa, produto, sprint) são possíveis [Leffingwell2010]. Uma empresa poderia adotar uma estratégia de ganho de valor comercial precoce, por exemplo, se seu objetivo principal fosse entregar um produto antecipadamente e estabelecer participação de mercado. Uma estratégia de redução precoce do risco poderia ser preferida se um fornecedor quisesse a todo custo evitar o retorno de um produto devido, por exemplo, a um desempenho ou segurança inadequados.

Muitas vezes o valor comercial potencial e os riscos são interdependentes. O foco em um valor comercial específico pode aumentar riscos específicos, mudando o foco do valor comercial pode mudar os riscos também [Reinertsen2009].

Em cada caso, a priorização dos requisitos deve ser ajustada de acordo com a estratégia selecionada, levando em conta as dependências entre os requisitos.

EU 5.3 Estimativa de histórias de usuários e outros itens de Backlog (N3)

Duração: 30 minutos + 30 minutos de exercício

Objetivos educacionais
OE 5.3.1 Aplicar previsões e estimativas (N3)
OE 5.3.2 Entender como obter uma previsão de médio prazo (N2)
OE 5.3.3 Entender a vantagem das estimativas relativas, categorizadas e de grupo (N2)
OE 5.3.4 Técnicas de estimativa (N2)

Mesmo em um mundo ágil são necessárias previsões perfeitas a fim de determinar quanto trabalho pode ser "feito" dentro de uma iteração previamente especificada (timebox). Nenhum elemento não estimado é permitido entrar em um sprint em Scrum por duas razões [Cohn2005]:

1. Não está claro se o elemento pode ser completado dentro da sprint e assim impede a entrega do software operacional no final da sprint.

2. Sem discussão e estimativa, a equipe não teria nenhum ponto de referência (planejamento vs. realização real) para aprender para as próximas sprints.

Além disso, as organizações de desenvolvimento que excedem uma equipe geralmente precisam de previsões para priorizar e planejar adequadamente o trabalho.

A previsão de desenvolvimento se dá em várias escalas de tempo. Como o entendimento de quais requisitos são atribuídos à próxima iteração deve ser bastante detalhado, os planos de iteração de longo prazo terão menos detalhes, mas devem permitir a estimativa e o planejamento em larga escala.

Tal planejamento também permite relatar o progresso, ou seja, quais épicos, características e histórias de usuários foram de fato entregues nas datas de entrega previstas. Ele apoia também a detecção de requisitos que necessitam ser quebrados porque são muito grandes para caber em uma estimativa [Leffingwell2010].

As estimativas iniciais do projeto são frequentemente imprecisas, mas se tornam cada vez mais precisas à medida que a atividade é iterada (um princípio conhecido como o Cone da Incerteza). Ao analisar o que foi entregue em iterações anteriores, a velocidade da equipe pode ser calculada permitindo uma melhor estimativa da capacidade das futuras iterações [McConnel2006].

Para apoiar melhores estimativas a médio prazo, grandes requisitos como épicos são divididas em características para realizar estimativas em nível de características e somá-las ao épico. Essas estimativas ainda não são muito precisas, mas são úteis para a estimativa do épico e servem adicionalmente como uma espécie de verificação em relação ao conhecimento do épico (suposições etc.).

Os métodos ágeis definem regras que ajudam a fazer estimativas melhores e mais precisas:

- Todos os envolvidos na estimativa devem ter a mesma compreensão do trabalho que precisa ser "feito".
- As estimativas devem ser feitas por aqueles que fazem o trabalho, a equipe multifuncional (Desenvolvedores no Scrum). Isto ajuda a colocar todas as pessoas envolvidas no mesmo nível de conhecimento, trocando conhecimentos e suposições sobre o trabalho a ser feito.
- As estimativas devem ser feitas relativamente / relativamente ao trabalho já realizado (Estimativa por analogia), uma vez que é mais provável que essas estimativas sejam precisas do que estimativas absolutas.
- As estimativas devem ser feitas em uma unidade artificial que represente esforço, complexidade e risco em uma só. O uso de uma unidade artificial é necessário para trazer a todos a nova forma de estimativa porque o uso de horas / dias em estimativas tende a reforçar os padrões de comportamento existentes e tradicionais.
- O uso do extrato da sequência Fibonacci ou tamanhos de camisetas ajuda a suportar essa forma de estimativa (uso de termos como "maior/menor que" ao invés de "exatamente 2 vezes mais que ...").

Várias técnicas suportam a estimativa relativa. A técnica mais famosa é a chamada Planning Poker [Cohn2005]. Em Planning Poker, os desenvolvedores discutem o requisito, e todos selecionam uma carta do conjunto Planning Poker que é baseada na sequência Fibonacci para representar o tamanho relativo do novo requisito em relação a uma base comum. Após cada um ter selecionado sua carta de pôquer, os membros da equipe com a estimativa mais baixa e mais alta discutem as razões de suas estimativas e tentam convencer os outros membros da equipe de sua argumentação. Em seguida, é iniciada a próxima rodada de estimativa.

Se a equipe não conseguir chegar a um valor comum em três rodadas, o requisito é enviado de volta para o dono do produto. Se a equipe puder concordar, o valor comum é atribuído.

A vantagem do Planning Poker é que é uma técnica muito boa para as equipes novas e inexperientes encontrarem suas estimativas porque evita a ancoragem por um único membro da equipe. A desvantagem é que isso consome muito tempo. *[Nota: O livro Thinking, Fast and Slow de D. Kahneman [Kahneman2013] dá uma grande introdução à ancoragem e outros efeitos psicológicos relacionados ao pensamento e ao julgamento.]*

Para superar esta desvantagem para equipes mais experientes, são utilizadas técnicas aprimoradas.

Uma simplificação da técnica do Planning Poker é baseada nos mesmos princípios do Planning Poker, mas utiliza uma maneira diferente de determinar a estimativa correta. Ao invés de cada membro da equipe fazer uma estimativa pessoal, um conjunto de cartas de pôquer é distribuído por uma mesa e os requisitos de referência são colocados no "recipiente" correspondente representado pela carta de pôquer. Em seguida, os requisitos são selecionados pelos membros da equipe em uma abordagem 'round-robin' onde os membros da equipe podem colocar um novo requisito no "recipiente" correspondente ou redesignar um já colocado em um recipiente diferente. Se um requisito for reatribuído várias vezes, ele será removido e enviado de volta para o dono do produto. Esta abordagem é muito mais rápida, mas precisa de uma equipe que seja madura o suficiente para discordar das tarefas feitas por outros membros da equipe, em vez de concordar facilmente ("ancorar").

O próximo passo da evolução é normalmente chamado de "Estimativa de Afinidade" e é usado para estimar quantidades maiores de requisitos, por exemplo, para uma estimativa aproximada na preparação de Planejamento de Releases. A diferença em relação à abordagem anterior é que os requisitos não serão atribuídos pela abordagem de 'round-robin', mas cada membro da equipe recebe uma parte dos requisitos e a atribui silenciosamente aos "recipientes" representados pelo conjunto de cartas de pôquer. Após a tarefa silenciosa, todas as pessoas envolvidas podem inspecionar os requisitos designados e marcar aqueles que são questionados. Normalmente isto leva a uma cota de 20-30% de requisitos que precisam ser discutidos e 70-80% que são aceitos por todos os membros da equipe.

EU 6 Escalando RE@Agile (N2)

Duração: 105 minutos

Termos: Estruturas de escalonamento, Requisitos de escalonamento e equipes, Estruturando requisitos e equipes em grande escala, Roadmaps e planejamento em grande escala, Validação do produto

EU 6.1 Escalando Requisitos e Equipes (N2)

Duração: 30 minutos

Objetivos educacionais
OE 6.1.1 Descrever exemplos comuns de estruturas de escalonamento (N1)
OE 6.1.2 Entender os desafios e o mecanismo para escalar os requisitos ágeis (N2)

Existem diferentes estruturas e implementações para o escalonamento da agilidade. As estruturas são generalizações de implementações particulares, conduzidas por especialistas estabelecidos e empresas de consultoria. Todas as estruturas de escalonamento são baseadas em valores e princípios ágeis. Cada estrutura específica é baseada em uma seleção de boas práticas ágeis e métodos particulares, técnicas e outros elementos combinados dentro de um conceito geral coerente. As estruturas de escalonamento variam em seu nível de maturidade, o número de boas práticas, diretrizes e regras, e o grau de adaptabilidade às necessidades específicas de uma organização.

Desde cerca de 2010, uma série de diferentes estruturas de escalonamento ágeis foram desenvolvidas. Entre eles estão Nexus [NEXUS], SAFe [SAFe], LeSS [LeSS], Scrum@Scale [S@S], BOSSA Nova [BOSSANOVA], Scrum of Scrums [SofS]e Spotify [Spotify2012].

As estruturas de escalonamento também são usadas porque às vezes não é suficiente trabalhar com apenas uma equipe: se o produto for muito complexo ou se for necessário um tempo mais curto para o mercado, o desenvolvimento ágil tem que ser escalado para envolver várias equipes. Além disso, a distribuição geográfica ou a distribuição de habilidades poderia levar a equipes distribuídas. Assim que tivermos várias equipes trabalhando em paralelo, o desafio é que os requisitos devem ser coordenados e a comunicação entre as equipes deve ser definida.

Os requisitos têm de ser agrupados para que possam ser atribuídos a diferentes equipes. Para estes agrupamentos, as estruturas sugerem nomes diferentes para diferentes níveis de abstração. Recomendamos enfaticamente que qualquer organização deve concordar com os nomes para estes diferentes níveis de requisito e definir diretrizes para os diferentes níveis, e então se ater a eles. Esses nomes são frequentemente predefinidos pela estrutura de escalonamento, ou pelas ferramentas usadas para capturar os requisitos.

Toda equipe precisa de alguém que assuma a responsabilidade pelo gerenciamento de requisitos. Para uma única equipe esta pessoa é normalmente chamada de dono do produto, como já explicamos nos capítulos anteriores. Se os requisitos tiverem que ser gerenciados para múltiplas equipes, algumas estruturas sugerem diferentes legendas de trabalho para gerenciar os requisitos em equipes maiores. Independente do cargo: certifique-se de que haja uma clara responsabilidade pela gestão de requisitos em todos os níveis organizacionais.

As equipes individuais trabalham em pequenas iterações, muitas vezes chamadas sprints (como discutido nos capítulos anteriores). No início de uma sprint, um conjunto de requisitos é selecionado a partir do backlog de produto e, no final da sprint, sua implementação é verificada. Ao escalar, as estruturas muitas vezes agrupam essas iterações curtas em iterações mais longas (por exemplo, com duração de dois ou três meses em vez de 2 a 4 semanas), chamadas de releases. Isto cria a necessidade de liberar os backlogs, estabelecendo o contexto e as metas para as equipes participantes organizarem seus sprints e a integração dos resultados das sprints.

EU 6.2 Critérios para estruturar Requisitos e Equipes em Grande Escala (N2)

Duração: 30 minutos

Objetivos educacionais
OE 6.2.1 Saber os princípios para organizar um backlog e para comunicar sobre os requisitos dentro de uma rede de equipes (N1)
OE 6.2.2 Entender requisitos divididos em diferentes configurações (de projeto) (N2)

De uma perspectiva de requisitos, temos que "fechar o ciclo" do requisito inicial (empresarial) exigido pelas partes interessadas, através da divisão de requisitos complexos em peças menores gerenciáveis pelos desenvolvedores, e depois assegurar que os resultados montados se combinem para formar uma solução que possa ser liberada para o negócio.

São necessárias estruturas e práticas sofisticadas a fim de apoiar a colaboração da equipe, mudanças de requisitos e entrega rápida de produtos no desenvolvimento de produtos em larga escala. Caso contrário, os desenvolvedores podem desperdiçar esforços coordenando com equipes que não são relevantes para seu trabalho.

Requisitos complexos devem ser compreendidos e gerenciados por vários donos e desenvolvedores de produtos. Para este fim, os requisitos são recursivamente divididos em outros requisitos até que sejam suficientemente simples para serem desenvolvidos por uma equipe, estabelecendo assim uma hierarquia de requisitos [GoWo2006]. A comunicação é necessária tanto para cima quanto para baixo nesta hierarquia, por exemplo, entre os donos de produtos que trabalham com requisitos relacionados (ou que concordam em prioridades) em diferentes níveis de abstração, bem como entre os desenvolvedores, por exemplo, na organização de entregas para formar incrementos de produto ponta a ponta.

Para apoiar adequadamente a colaboração e a comunicação, os requisitos devem ser gerenciados usando um único backlog lógico. A ideia chave é que cada requisito seja mantido em um único lugar, evitando redundâncias ou contradições. Isto ainda pode ser alcançado mesmo quando se subdividir ainda mais o backlog em equipes de trabalho de backlog.

Enquanto refinam os requisitos de granularidade grosseira, os donos de produtos podem trabalhar em itens de backlog ainda não associados a nenhuma equipe, ou podem dividir requisitos complexos e entregar os itens de backlog resultantes às equipes para refinamento adicional (ver [NEXUS], [SAFe], [LeSS] para mais informações). Para garantir a rastreabilidade entre os requisitos em diferentes níveis de abstração, os donos dos produtos devem relacionar os respectivos itens de backlog.

O desenvolvimento de produtos terá dificuldade de reagir às mudanças em tempo hábil se cada equipe depender de uma teia complicada de interações com outras equipes para aprovar qualquer decisão. É necessária uma estrutura de equipe que permita que as equipes se auto-organizem em torno da criação de valor: para responder melhor ao feedback das partes interessadas, para tomar decisões razoáveis de forma independente e para fornecer recursos de ponta a ponta [Anderson2020], [Reinertsen2009].

A divisão de requisitos é necessária para quebrar os requisitos de granularidade grosseira, para que possam ser atribuídos aos desenvolvedores. Para entregar incrementos de produtos liberáveis com dependência mínima de outras equipes, as equipes ágeis devem trabalhar com recursos de ponta a ponta, acoplados fracamente (compare com as equipes de recursos, conforme introduzido em [Larman2016]). Para conseguir isso, o escopo do produto é dividido em unidades menores, fracamente acopladas, de funcionalidade interna consistente (ou seja, divisão de requisitos baseada em recursos). Cada unidade tem limites funcionais bem definidos. Os limites devem ser claros para permitir uma colaboração eficaz. Se os requisitos também forem divididos de acordo com estes limites funcionais, então os donos de produtos atribuídos a uma unidade específica podem trabalhar em características com maior grau de independência.

Infelizmente, em muitos casos, não é tão fácil decompor os requisitos com base em unidades fracamente acopladas de funcionalidade ponta a ponta. Devido ao projeto arquitetônico (por exemplo, tecnologia, infraestrutura, componentes do sistema, plataforma comum, camadas arquitetônicas como front e backend), bem como considerações organizacionais (habilidades especializadas, localização da equipe, subcontratados), as unidades de funcionalidade podem se sobrepor. Isto significa que diferentes equipes ágeis devem trabalhar em conjunto para implementar características específicas e seus respectivos donos de produtos precisam colaborar mais de perto com os requisitos.

Para implementar recursos de forma colaborativa, equipes ágeis requerem uma compreensão compartilhada dos requisitos e de seu contexto comercial. Eles também devem concordar sobre requisitos transversais, restrições e interfaces técnicas comuns para que os produtos de diferentes equipes possam ser integrados aos incrementos de trabalho. A integração e os testes de características tornam-se mais complexos e a sincronização de equipes usando backlogs e roadmaps é ainda mais crítica. Outra abordagem é formar uma equipe ágil separada em torno de certas características ou emprestar os engenheiros necessários para desenvolver características específicas de forma independente. Veja também [Anderson2020] para mais detalhes sobre desenho e práticas organizacionais ágeis.

EU 6.3 Roadmaps e Planejamento em Grande Escala (N2)

Duração: 30 minutos

Objetivos educacionais
OE 6.3.1 Caracterizar a diferença entre um roadmap e um backlog (N1)
OE 6.3.2 Entender a criação e a gestão de um roteiro (N2)

No desenvolvimento de produtos em larga escala, os donos de produtos gerenciam os requisitos no backlog focado no produto. Em contraste com o backlog, um roadmap é utilizado para o planejamento do desenvolvimento de produtos de forma incremental.

Um roadmap é uma previsão de como o produto irá crescer [Pichler2016]. Os roadmaps não alteram o conteúdo dos itens de backlog, mas os organizam em uma linha de tempo. Ele responde à pergunta quando podemos esperar, grosso modo, quais características.

Um roadmap é um meio útil para comunicar metas e decisões (estratégicas) aos desenvolvedores e outras partes interessadas. Ele divide uma meta de longo prazo em iterações gerenciáveis, representa dependências entre as equipes e fornece direção e transparência às partes interessadas.

No início do desenvolvimento ágil do produto, pouco se sabe sobre o produto, ou sobre o trabalho feito pelas equipes. Assim, o escopo do produto, bem como as estimativas de custo, estão sujeitos a um alto nível de incerteza. À medida que mais iterações são concluídas e mais feedback é coletado das partes interessadas, a incerteza diminui gradualmente levando a um planejamento mais confiável e a um roadmap estável. Este princípio é conhecido como o *cone de incerteza* [Boehm1981]. Embora este princípio seja geralmente verdadeiro para todos os projetos de desenvolvimento ágil, ele se torna ainda mais importante no desenvolvimento de produtos em larga escala, pois os riscos devidos à complexidade do produto e ao potencial de desalinhamento entre várias equipes - e consequentemente a necessidade de mais planejamento - são ainda maiores.

Um roadmap mostra objetivos estratégicos, marcos e requisitos de granularidade grosseira. Marcos importantes podem ser internos ou determinados por eventos externos, como uma feira ou a introdução de nova regulamentação no mercado. A representação de um roadmap depende de seu objetivo, grupo alvo e horizonte de planejamento.

Para os clientes, patrocinadores da administração e a empresa, um **roadmap de produtos** a longo prazo contendo metas estratégicas e requisitos de produtos de granularidade grosseira é muitas vezes suficiente [Pichler2016]. Por outro lado, os desenvolvedores precisam conhecer os detalhes adicionais representados pelos itens de backlog refinados (por exemplo, histórias e tarefas), bem como as dependências entre eles. Estas informações são fornecidas pelos **roadmaps de entrega** de médio prazo [SAFe].

Para desenvolver um roadmap de produtos a longo prazo, um dono de produto deve primeiro definir uma visão e estratégia do produto. Posteriormente, os donos de produtos devem então elucidar os requisitos de granularidade grosseira, engajando-se com as partes interessadas necessárias. Neste momento, não há necessidade de investir tempo em requisitos detalhados. Embora os requisitos estejam sujeitos a um alto nível de incerteza nesta fase, o roadmap do produto como um plano de iteração rudimentar e inicial é suficientemente bom para apoiar o planejamento e a sincronização.

Para criar um roadmap de entrega a médio prazo, os donos de produtos devem refinar e priorizar os itens de backlog a partir do roadmap de produto existente.

Estes itens precisam ser grosseiramente estimados pelos desenvolvedores, mesmo que as estimativas ainda sejam imprecisas (por exemplo, tamanhos de camisetas) nesta fase. A estimativa só tem que ser boa o suficiente para fornecer uma visão geral das próximas iterações.

A criação e atualização de roadmaps de entrega normalmente acontece em eventos de planejamento presencial conhecidos como planejamentos de sala grande (ou PI Planning in SAFe), realizados em intervalos regulares.

EU 6.4 Validação do produto (N2)

Duração: 15 minutos

Objetivos educacionais

OE 6.4.1 Entender métodos concretos para validar os requisitos do produto em desenvolvimento ágil (N2)

A ideia principal do desenvolvimento ágil é desenvolver uma pequena fatia do produto, gerar feedback envolvendo as partes interessadas e adaptar o desenvolvimento do produto de acordo com as percepções e descobertas. Para este fim, os donos do produto devem usar uma versão recém-lançada do produto para verificar seu valor comercial e para examinar se os requisitos do produto foram entendidos corretamente.

Uma revisão de sprint é um meio adequado para apresentar aos interessados um incremento de produto desenvolvido na última sprint. No desenvolvimento de produtos em larga escala, a mesma ideia pode ser usada também. Mas ao invés de revisar uma única fatia de produto desenvolvida por uma única equipe, todos os produtos entregues pela equipe são integrados a um incremento de produto que vale a pena validar. O incremento é demonstrado em uma **análise do produto (demonstração)** aos interessados levando a uma melhor impressão de todo o produto [SAFe], [Larman2016], [LeSS].

Outra abordagem para a validação de produtos no desenvolvimento de produtos em larga escala é **análise de dados** [Maalej et al.2016]. O incremento integrado do produto é entregue aos usuários e, com base em seu comportamento, são feitas medições sobre se as características do produto têm um impacto positivo, neutro ou negativo. Os donos de produtos podem usar os resultados para identificar características potencialmente mal projetadas. Para entender melhor os problemas identificados, eles podem precisar aplicar novamente técnicas regulares de elicitação e análise de requisitos.

DEFINIÇÕES DOS TERMOS, Glossário

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

Referências

[AgileAlliance] Glossary of the Agile Alliance: Definition of term “Definition of Ready”: https://www.agilealliance.org/glossary/definition-of-ready. Última visita em janeiro de 2018.

[AgileManifesto2001] Agile Manifesto: http://www.Agilemanifesto.org, 2001. Última visita em janeiro de 2017.

[Alexander2005] Alexander, I. F.: A Taxonomy of Stakeholders – Human Roles in System Development. International Journal of Technology and Human Interaction, Vol 1, 1, 2005, pages 23-59.

[Anderson2020] Anderson, J.: Agile Organizational Design – Growing Self-Organizing Structure at Scale. Leanpub, 2020.

[Beck2002] Beck, K.: Test Driven Development: By Example. Addison-Wesley 2002.

[Boehm1981] Boehm Barry W.: Software Engineering Economics. Prentice Hall, 1981.

[BOSSANOVA] BOSSA nova https://www.agilebossanova.com/#bossanova , última visita em maio de 2021

[Clements et al.2001] Clements, P., Kazman, R., Klein, M.: Evaluating Software Architectures. SEI Series in Software Engineering, 2001.

[Cohn2004] Cohn, M.: User Stories Applied For Agile Software Development. Addison-Wesley, 2004.

[Cohn2005] Cohn, M.: Agile Estimating and Planning. Prentice Hall, Nov 2005.

[Conway1968] Conway, M.E.: How Do Committees Invent? Datamation Magazine , 1968. http://www.melconway.com/Home/Committees_Paper.html. Última visita em maio 2021.

[Cooper2004] Cooper, A.: The Inmates are Running the Asylum: Why High Tech Products Drive Us Crazy and How to Restore the Sanity. Pearson Education, 2004.

[Doran1981] Doran, G.T.: There's a S.M.A.R.T. way to write management's goals and objectives. Management Review, 1981. AMA FORUM. **70** (11): 35–36.

[Glinz2014] Glinz, M.: A Glossary of Requirements Engineering Terminology. Standard Glossary for the Certified Professional for Requirements Engineering (CPRE) Studies and Exam, Version 1.6, 2014. https://www.ireb.org/en/downloads/#cpre-glossary**.** Última visita em maio de 2021.

[GoWo2006] Gorschek, T., Wohlin, C.: Requirements Abstraction Model. Requirements Engineering Journal, Vol. 11, No. 1, pp. 79-101, 2006.

[HeHe2010] Heath, C., Heath, D.: Switch: How to Change Things When Change Is Hard. Crown Business, 2010.

[Highsmith2001] Highsmith, J.: Design the Box. Agile Project Management E-Mail Advisor 2001, http://www.joelonsoftware.com/articles/JimHighsmithonProductVisi.html. Última visita em maio de 2021.

[IREB2019] IREB e.V.: CPRE Advanced Level Elicitation Syllabus, 2019. https://www.ireb.org/en/downloads/tag:al-e-c#top. Última visita em maio de 2021.

[IREB2017] IREB e.V.: CPRE – RE@Agile Primer – Syllabus and Study Guide, 2017 https://www.ireb.org/en/downloads/tag:re-agile-primer#top. Última visita em maio de 2021.

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

[Jeffries2001] Jeffries, R.: Essential XP: Card, Conversation, Confirmation, 2001, https://ronjeffries.com/xprog/articles/expcardconversationconfirmation/. Última visita em maio de 2021.

[Kahneman2013] Kahneman D.: Thinking, Fast and Slow. Farrar, Straus and Giroux, 2013.

[Kniberg2012] Kniberg, H.: Scaling Agile @ Spotify with Tribes, Squads, Chapters & Guilds https://blog.crisp.se/2012/11/14/henrikkniberg/scaling-agile-at-spotify. Última visita em maio de 2021.

[LaBai2003] Larman, C., Basili, V. R.: Iterative and Incremental Development: A Brief History. IEEE *Computer,* Vol 36, No. 6, 2003, 47-56.

[Lamsweerde2009] van Lamsweerde, A.: Requirements Engineering: From System Goals to UML Models to Software Specifications. Wiley, 2009.

[Larman2016] Larman, C.: Large-Scale Scrum: More with LeSS. Addison Wesley, 2016.

[Leffingwell2010] Leffingwell, D.: Agile Software Requirements – Lean Requirements Practices for Teams, Programs, and the Enterprise. Addison Wesley, 2010.

[LeSS] Large-Scale Scrum: https://less.works/. Última visita em maio de 2021.

[Maalej et al.2016] Maalej, W., Nayebi, M., Johann T., Ruhe, G.: Toward Data-Driven Requirements Engineering. IEEE Software (Volume 33, Issue 1), 2016.

[McConnel2006] McConnel, S.: Software Estimation, Demystifying the Black Art. Microsoft Press, 2006.

[Meyer2014] Meyer, B.: Agile! The Good, the Hype and the Ugly. Springer, 2014.

[NEXUS] Scaling Scrum with Nexus™: https://www.scrum.org/resources/scaling-scrum. Última visita em maio de 2021.

[OsPi2010] Osterwald, A., Pigneur, Y.: Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers. John Wiley and Sons, 2010.

[Patton2014] J. Patton, J.: User Story Mapping –Discover the Whole Story, Build the Right Product. O'Reilly, 2014.

[Pichler2011] Pichler, R.: Product Vision Board, 2011 http://www.romanpichler.com/blog/the-product-vision-board/. Última visita em maio de 2021.

[Pichler2016] Pichler, R.: Strategize: Product Strategy and Product Roadmap Practices for the Digital Age. Pichler Consulting, 2016.

[PoRu2015] Pohl, K., Rupp, C.: Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam - Foundation Level. Rocky Nook, 2015.

[Reinertsen2009] Reinertsen, D.G.: The Principles of Product Development Flow – Second Generation Lean Product Development. Celeritas Publishing, 2009.

[SAFe] Scaled Agile Framework 4.5 ® http://www.scaledagileframework.com/. Última visita em maio de 2021.

[S@S] Scrum at Scale™: https://www.scruminc.com/scrum-scale-case-modularity/. Última visita em maio de 2021.

[Scrumguide] The Scrum Guide ™ : http://www.scrumguides.org/scrum-guide.html. Última visita em maio de 2021.

[SofS] Scrum of Scrums https://scrumguide.de/scrum-of-scrums, última visita em maio 2021

[Spotify2012] Scaling Agile @ Spotify https://blog.crisp.se/wp-content/uploads/2012/11/SpotifyScaling.pdf , última visita em maio 2021

[Sterling2012] Sterling C.: Affinity Estimating – A How-To https://scrumology.com/guest-post-affinity-estimating-a-how-to/. Última visita em maio de 2021.

[RoRo2012] Robertson J., Robertson S.: Mastering the Requirements Process – Getting Requirements Right, 3rd edition. Addison Wesley, 2012.

[Robertson2003] Robertson, S.: Stakeholders, Goals, Scope: The Foundation for Requirements and Business Models, 2003, https://www.volere.org/wp-content/uploads/2018/12/StkGoalsScope.pdf. Última visita em junho de 2021.

[Wake2003] Wake, B.: INVEST in Good Stories, and SMART Tasks, 2003, https://xp123.com/articles/invest-in-good-stories-and-smart-tasks/. Última visita em maio de 2021.

[Weerd et al.2006] van de Weerd, I., Brinkkemper, S., Nieuwenhuis R., Versendaal, J., Bijlsma, L.: On the Creation of a Reference Framework for Software Product Management: Validation and Tools Support. International Workshop on Software Product Management (IWSPM 2006).