



ENGENHARIA DE REQUISITOS



Professora Me. Vanessa Ravazzi Perseguine

UNICESUMAR

Av. Guedner, 1610 - Jardim Aclimação
Cep 87050-900 - MARINGÁ - PARANÁ
unicesumar.edu.br
44 3027.6360

UNICESUMAR EDUCAÇÃO A DISTÂNCIA

NEAD - Núcleo de Educação a Distância
Bloco 4 - MARINGÁ - PARANÁ
unicesumar.edu.br
0800 600 6360

as imagens utilizadas neste
livro foram obtidas a partir
do site SHUTTERSTOCK.COM

FICHA CATALOGRÁFICA

C397 **CENTRO UNIVERSITÁRIO DE MARINGÁ.** Núcleo de Educação a Distância; **PERSEGUINE**, Vanessa Ravazzi

Engenharia de Requisitos. Vanessa Ravazzi Perseguine.
(Reimpressão revista e atualizada)
Maringá-Pr.: UniCesumar, 2016.
158 p.
"Graduação - EaD".

1. Engenharia. 2. Software. 3. Gestão EaD. I. Título.

CDD - 22 ed. 620
CIP - NBR 12899 - AACR/2

Ficha catalográfica elaborada pelo bibliotecário
João Vivaldo de Souza - CRB-8 - 6828



Reitor

Wilson de Matos Silva

Vice-Reitor

Wilson de Matos Silva Filho

Pró-Reitor de Administração

Wilson de Matos Silva Filho

Pró-Reitor de EAD

Willian Victor Kendrick de Matos Silva

Presidente da Mantenedora

Cláudio Ferdinandi

NEAD - Núcleo de Educação a Distância

Direção Operacional de Ensino

Kátia Coelho

Direção de Planejamento de Ensino

Fabício Lazilha

Direção de Operações

Chrystiano Mincoff

Direção de Mercado

Hilton Pereira

Direção de Polos Próprios

James Prestes

Direção de Desenvolvimento

Dayane Almeida

Direção de Relacionamento

Alessandra Baron

Gerência de Produção de Conteúdo

Juliano de Souza

Supervisão do Núcleo de Produção de Materiais

Nádila de Almeida Toledo

Coordenador de Conteúdo

Fabiana De Lima

Design Educacional

Yasminn Zagonel

Projeto Gráfico

Jaime de Marchi Junior

José Jhonny Coelho

Editoração

Humberto Garcia da Silva

Revisão Textual

Viviane Favaro Notari/Yara Martins Dias

Ilustração

André Luís Onishi



Professor
Wilson de Matos Silva
Reitor

Viver e trabalhar em uma sociedade global é um grande desafio para todos os cidadãos. A busca por tecnologia, informação, conhecimento de qualidade, novas habilidades para liderança e solução de problemas com eficiência tornou-se uma questão de sobrevivência no mundo do trabalho.

Cada um de nós tem uma grande responsabilidade: as escolhas que fizermos por nós e pelos nossos farão grande diferença no futuro.

Com essa visão, o Centro Universitário Cesumar assume o compromisso de democratizar o conhecimento por meio de alta tecnologia e contribuir para o futuro dos brasileiros.

No cumprimento de sua missão – “promover a educação de qualidade nas diferentes áreas do conhecimento, formando profissionais cidadãos que contribuam para o desenvolvimento de uma sociedade justa e solidária” –, o Centro Universitário Cesumar busca a integração do ensino-pesquisa-extensão com as demandas institucionais e sociais; a realização de uma prática acadêmica que contribua para o desenvolvimento da consciência social e política e, por fim, a democratização do conhecimento acadêmico com a articulação e a integração com a sociedade.

Diante disso, o Centro Universitário Cesumar almeja ser reconhecido como uma instituição universitária de referência regional e nacional pela qualidade e compromisso do corpo docente; aquisição de competências institucionais para o desenvolvimento de linhas de pesquisa; consolidação da extensão universitária; qualidade da oferta dos ensinoss presencial e a distância; bem-estar e satisfação da comunidade interna; qualidade da gestão acadêmica e administrativa; compromisso social de inclusão; processos de cooperação e parceria com o mundo do trabalho, como também pelo compromisso e relacionamento permanente com os egressos, incentivando a educação continuada.



Professor

Fabrcio Lazilha

Diretoria de
Planejamento de Ensino

Professora

Ktia Solange Coelho

Diretoria Operacional
de Ensino

Seja bem-vindo(a), caro(a) acadêmico(a)! Você está iniciando um processo de transformação, pois quando investimos em nossa formação, seja ela pessoal ou profissional, nos transformamos e, consequentemente, transformamos também a sociedade na qual estamos inseridos. De que forma o fazemos? Criando oportunidades e/ou estabelecendo mudanças capazes de alcançar um nível de desenvolvimento compatível com os desafios que surgem no mundo contemporâneo.

O Centro Universitário Cesumar mediante o Núcleo de Educação a Distância, o(a) acompanhará durante todo este processo, pois conforme Freire (1996): “Os homens se educam juntos, na transformação do mundo”.

Os materiais produzidos oferecem linguagem dialógica e encontram-se integrados à proposta pedagógica, contribuindo no processo educacional, complementando sua formação profissional, desenvolvendo competências e habilidades, e aplicando conceitos teóricos em situação de realidade, de maneira a inseri-lo no mercado de trabalho. Ou seja, estes materiais têm como principal objetivo “provocar uma aproximação entre você e o conteúdo”, desta forma possibilita o desenvolvimento da autonomia em busca dos conhecimentos necessários para a sua formação pessoal e profissional.

Portanto, nossa distância nesse processo de crescimento e construção do conhecimento deve ser apenas geográfica. Utilize os diversos recursos pedagógicos que o Centro Universitário Cesumar lhe possibilita. Ou seja, acesse regularmente o AVA – Ambiente Virtual de Aprendizagem, interaja nos fóruns e enquetes, assista às aulas ao vivo e participe das discussões. Além disso, lembre-se que existe uma equipe de professores e tutores que se encontra disponível para sanar suas dúvidas e auxiliá-lo(a) em seu processo de aprendizagem, possibilitando-lhe trilhar com tranquilidade e segurança sua trajetória acadêmica.

Professora Me. Vanessa Ravazzi Perseguine

Mestre em Ciências da Computação. Especialista em Gestão e Coordenação Escolar. Especialista em Gestão de Projetos. Graduada em Ciência da Computação. Experiência na Gestão de TI órgão Público, Coordenação de cursos e projetos. Experiência na área de Ciências da Computação, atuando nas áreas: engenharia de software, lógica e algoritmos e gerencia de projetos.

SEJA BEM-VINDO(A)!

Olá! Seja bem-vindo(a) à disciplina Engenharia de Requisitos. Neste curso, iremos abordar as atividades da engenharia de requisitos e sua importância no processo de desenvolvimento de software. Nesse contexto, gostaria que considerasse o seguinte: mesmo para engenheiros de software experientes, é complexo gerir todas as especialidades envolvidas no processo de desenvolvimento de software.

Via de regra, os softwares são desenvolvidos para um conhecimento específico, em que se faz necessário um entendimento prévio do modelo de negócio para o qual o software será desenvolvido. Por exemplo, se o software é para o gerenciamento de uma locadora de vídeos, o desenvolvedor deverá entender sobre o negócio locar vídeos; se o software tem a responsabilidade de emitir demonstrativos contábeis, o responsável pelo seu desenvolvimento terá que conhecer todos os dados necessários a serem levantados para os cálculos contábeis e posterior emissão desses relatórios; ou, ainda, se o software é responsável pelo monitoramento dos dados que mantém atividades de sobrevivência que trará de volta à terra o astronauta que foi a Marte, o profissional responsável terá que aprender todas as condições e exigências que deverão ser controladas. Para a entrega de um produto de qualidade, que atenda às expectativas do cliente e que execute o que se é esperado, é de extrema importância que os responsáveis pelo desenvolvimento do software conheçam o ambiente onde o software será inserido.

Calma! Não se assuste! Não estou dizendo que terá que fazer uma faculdade sobre a especialidade de cada software que irá desenvolver, ou supervisionar o desenvolvimento, o que eu pretendo deixar claro é que tão importante quanto desenvolver o software, “botar a mão na massa”, é o planejar o que será desenvolvido. O tempo destinado ao planejamento é uma discussão antiga e a gestão de projetos está em evidência mundial. Casos de sucesso e a utilização de técnicas ágeis, modeladas para o desenvolvimento de software, estão promovendo a desburocratização do processo de software, principalmente nas fases de planejamento e controle, o que tem impulsionado as empresas para a sistematização dos processos internos e a adoção de boas práticas de desenvolvimento de software. Dentro da engenharia de software, podemos considerar como parte do planejamento o estabelecimento de um processo de software, e a engenharia de requisitos é a atividade do processo de software responsável por recolher as informações sobre o ambiente de negócio em que o software será inserido, interpretar para os desenvolvedores o propósito do software e projetar as respostas esperadas pelos usuários.

Além de entender como a engenharia de requisitos se situa nos processos de engenharia de software, no decorrer desta disciplina, estudaremos os fundamentos da engenharia de requisitos, os processos, como gerenciar requisitos e controlar mudanças e teremos um breve olhar sobre como as metodologias ágeis de desenvolvimento de software tratam as atividades da engenharia de requisitos.

Vamos começar nossos estudos?

Boa leitura!

SUMÁRIO

■ UNIDADE I

ENGENHARIA DE REQUISITOS NO CONTEXTO DA ENGENHARIA DE SOFTWARE

13 Introdução

14 Engenharia de Requisitos no Contexto da Engenharia de Software

16 Processo de Software

20 Modelo de Processo de Software

21 Atividades Fundamentais do Processo de Software

23 Importância da Engenharia de Requisitos no Processo de Desenvolvimento de Software

27 Considerações Finais

■ UNIDADE II

FUNDAMENTOS DA ENGENHARIA DE REQUISITOS

35 Introdução

36 Conceitos Fundamentais da Engenharia de Requisitos

37 Stakeholders

40 Classificação dos Requisitos

48 Documentação dos Requisitos de Software

53 Matriz de Rastreabilidade de Requisitos

54 Template da Matriz de Rastreabilidade de Requisitos

56 Considerações Finais



■ UNIDADE III

PROCESSOS DA ENGENHARIA DE REQUISITOS

67	Introdução
68	Processos da Engenharia de Requisitos
69	Levantamento e Análise de Requisitos
88	Negociação
90	Validação dos Requisitos
94	Especificação de Requisitos
105	Considerações Finais

■ UNIDADE IV

GESTÃO DE REQUISITOS

115	Introdução
116	Evolução dos Requisitos
118	Rastreabilidade
123	Controle de Mudanças
127	Considerações Finais



SUMÁRIO

UNIDADE V

REQUISITOS NAS METODOLOGIAS ÁGEIS

135 Introdução

136 Requisitos nas Metodologias Ágeis

137 Manifesto Ágil

140 SCRUM

143 Especificação de Requisitos Ágeis

146 Considerações Finais

151 Conclusão

153 Referências

158 Gabarito



ENGENHARIA DE REQUISITOS NO CONTEXTO DA ENGENHARIA DE SOFTWARE

UNIDADE

I

Objetivos de Aprendizagem

- Apresentar conceitos básicos da engenharia de software.
- Contextualizar a engenharia de requisitos no processo de software.

Plano de Estudo

A seguir, apresentam-se os tópicos que você estudará nesta unidade:

- Engenharia de requisitos no contexto da engenharia de software
- Processo de software
- Modelo de processo de software
- Atividades fundamentais do processo de software
- Importância da engenharia de requisitos no processo de desenvolvimento de software

INTRODUÇÃO

Olá, caro(a) aluno(a)! Começamos nossos estudos apresentando a engenharia de requisitos no contexto da engenharia de software. Nesta unidade, revisaremos alguns conceitos importantes, necessários para a compreensão dos objetivos da engenharia de requisitos.

Passaremos pelo software que se tornou elemento fundamental no mercado atual, a ponto de se tornar uma espécie de carteira de investimento para os empresários (Gestão de Ativos de Software) e finalizaremos com o entendimento dos processos fundamentais envolvidos nas atividades de desenvolvimento de software.

A Engenharia de Software envolve um conjunto de etapas que inclui métodos, ferramentas e procedimentos que possibilitam o controle do processo de desenvolvimento de software, ocupando-se de todos os aspectos, desde os estágios iniciais de especificação do sistema até sua manutenção (evolução). A Engenharia de Requisitos é uma espécie de subárea da Engenharia de Software, com o objetivo principal de obter um Documento de Requisitos correto e completo.

Ouso afirmar, concordando com vários estudiosos da área, que engenharia de requisitos é a disciplina do ciclo de vida do desenvolvimento de software que tem influência direta, ou uma das mais diretas, sobre o sucesso ou fracasso de qualquer projeto.

A inclusão de técnicas maduras e comprovadas de gestão de requisitos oferece, dentre outros benefícios, o aumento da satisfação do usuário, que é o principal indicador de qualidade do sistema, e, também, a redução de custos na construção e manutenção de um sistema. Se os requisitos forem bem definidos e gerenciados, existe uma real garantia de que o sistema atingirá seu objetivo em uma primeira tentativa, o que garante uma redução do custo total do sistema.

Assim, nosso objetivo nesta unidade é entender a base da engenharia de requisitos, conhecendo os processos que a sustentam. Vamos lá!?



ENGENHARIA DE REQUISITOS NO CONTEXTO DA ENGENHARIA DE SOFTWARE

Para entendermos a importância da engenharia de requisitos durante a definição de um software, é necessário identificar onde ela se encaixa no contexto da engenharia de software. Vamos refrescar a memória e contextualizar a engenharia de requisitos, relembrando rapidamente alguns conceitos fundamentais?

Primeiro, a partir da informática básica, o que é software? Fácil! Imediatamente nos vem à memória: “conjunto de instruções”; “parte lógica”; “programas para computador”; “sistemas que controlam o funcionamento de um computador”. Perfeito! Mantenha isso em mente.

Agora, da base do nosso curso, como você define engenharia de software? Quando penso em engenharia de software, lembro-me de “ferramentas”; “processo”; “sistematização”; “controle”; “metodologia”. A IEEE Standard 24765-2010, define engenharia de software como “a aplicação de uma abordagem sistemática, disciplinada e quantificável para o desenvolvimento, operação e manutenção de software”.

Correlacionando, temos, então, que software é um conjunto de instruções a ser executado por um computador, e engenharia de software é a abordagem sistemática que envolverá todos os aspectos de desenvolvimento do software: o projeto, a análise e a construção do software para algum objetivo específico.

Aprofundando um pouco, para Pressman (2010), essa abordagem sistemática a que me referi acima pode ser vista como uma tecnologia em camadas, conforme mostra a figura 1.

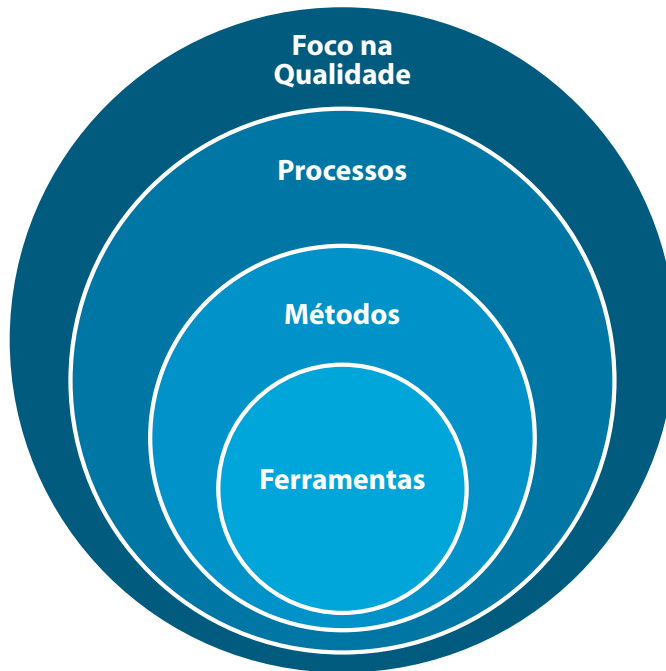


Figura 1: Engenharia de software em camadas
Fonte: adaptada de Pressman (2010).

Podemos assumir que a busca pela qualidade deve ser a base para todas as fases e atividades do processo de desenvolvimento do software, isto é, devemos manter **Foco na Qualidade**. Na camada **Métodos**, Pressman (2010) explica que estão os detalhes de como fazer para construir o software¹ e, como está acima do processo, pode ser interpretada como uma camada de suporte, pois apoia as atividades de desenvolvimento, sistematizando-as e mensurando-as, de forma a garantir a manutenção da qualidade. A última camada, **Ferramentas**, no topo, garante apoio automatizado aos métodos. São as ferramentas computacionais, eletrônicas, ou outras que permitem automatização das atividades previstas pelo Processo e definidas no Método.

No modelo de engenharia de software de Pressman (2010), apresentado

¹ Neste ponto, é importante observar que existem métodos para as diferentes atividades do processo de desenvolvimento de software, por exemplo, as atividades da fase da especificação de software terão métodos diferentes dos das atividades da fase de projeto de software, que serão ainda diferentes das atividades dos métodos da fase de validação de software e assim por diante, vocês estudarão cada fase do processo de software como disciplinas do curso.

na Figura 2, o **Processo** se destaca como o elemento principal, responsável por definir a estrutura geral para o controle gerencial do desenvolvimento do software. É a camada **processo** que vai embasar e contextualizar quais métodos e ferramentas serão aplicados para o desenvolvimento de determinado produto de software, isto é, quais serão os produtos de trabalho, quais marcos serão definidos, como gerir as modificações propostas e se a qualidade está sendo mantida. Na camada processo, é estabelecido o Projeto de Desenvolvimento do Software.



Figura 2: Engenharia de Software em Camadas
Fonte: adaptada de Roger S. Pressman (2010).

PROCESSO DE SOFTWARE

Percebemos que, tanto para o IEEE quanto para Pressman (2010), tão importante quanto o produto de software que é produzido é a forma como ele é produzido. A proposta das disciplinas da engenharia de software é nos habilitar para a criação de softwares com qualidade e que sigam as especificações de orçamento e prazos. Como? Por meio de processos adaptáveis que têm como propósitos tornar ágil o desenvolvimento, atender às particularidades de cada projeto e, principalmente, alcançar a satisfação do cliente.

REFLITA



Por que, mesmo tendo disponíveis técnicas, metodologias testadas e ferramentas automatizadas, ainda existem tantos projetos de desenvolvimento de software fracassados?

Fonte: o autor.

Então, a produção de um software bem-sucedido – assumindo somente três critérios: 1. cumpriu orçamento; 2. cumpriu prazo; 3. atendeu às expectativas do cliente – dá-se por meio do estabelecimento bem definido do processo de software.

Agora, pergunto: o que você entende por **processo**? Deixe de lado, por enquanto, seu aspecto técnico, vamos pensar na palavra processo de forma geral. Se abrirmos um browser para Internet, digitarmos na caixa de pesquisa “defina processo”, e, se você optou pela ferramenta de pesquisa mais popular, serão apresentadas duas definições antes mesmo da apresentação da lista de links com os resultados encontrados²:

processo

substantivo masculino

1.
ação continuada, realização contínua e prolongada de alguma atividade; seguimento, curso, decurso.
2.
sequência contínua de fatos ou operações que apresentam certa unidade ou que se reproduzem com certa regularidade; andamento, desenvolvimento, marcha.

Analisando essas definições, de forma geral, ambas nos remetem a uma atividade continuada, concorda? Faz-nos pensar em algo desenvolvido passo a passo. A seguir, vamos observar a definição que um dicionário nos apresenta sobre o tema.

² Resultado da busca pelo termo “defina processo” no Google.

O Aurélio³ define:

Significado de Processo

1. Método, sistema, modo de fazer uma coisa.
2. Conjunto de manipulações para obter um resultado.
3. O conjunto dos papéis relativos a um negócio.
4. Conjunto dos autos e mais documentos escritos numa causa cível ou criminal.
5. Processamento.
6. Seguimento, decurso.
7. Marcha das fases normais ou mórbidas dos fenômenos orgânicos.
8. Demanda, ação.
9. Processo sumário: aquele que, em atenção à brevidade, dispensa certas formalidades.

Deixando de lado os significados que fogem ao nosso contexto, como o terceiro e o quarto, o dicionário nos apresenta processo como uma abordagem, uma forma adotada para se desenvolver algo.

Vamos consolidar? Processo é, então, uma atividade organizada em etapas específicas a serem desenvolvidas de forma contínua, que mantenha uma unidade, a fim de alcançar um propósito final.

Considerando que o nosso propósito final é o software (enquanto produto a ser desenvolvido) e de posse da nossa própria definição de processo, fica fácil concordar com Pressman (2010, p. 16) quando nos apresenta o processo de software como “um arcabouço para as tarefas que são necessárias para construir softwares de alta qualidade”.

³ Dicionário Aurélio (online).

SAIBA MAIS



O conceito de Engenharia de Software surgiu na conferência da OTAN sobre Engenharia de Software (NATO Software Engineering Conference), em Garmisch, Alemanha, organizada a fim de propor soluções para o que ficou historicamente conhecido como “Crise de Software” no final da década de sessenta.

Hoje, mais de cinquenta anos depois da Crise do Software, percebemos que a grande maioria das empresas e dos profissionais de desenvolvimento de software ainda não consegue sucesso em seus projetos. Guinther de Bitencourt Pauli afirma em seu artigo que “o problema é que as abordagens tradicionais de engenharia de software ainda se baseiam em outros ramos de engenharia, como a civil e mecânica, onde os custos com planejamento são menores”.

Aprofunde seu conhecimento lendo o artigo disponível em:

<<http://www.ancibe.com.br/artigos%20de%20si/artigo%20-%20Sucessos%20em%20projetos%20de%20informatica%C3%A7%C3%A3o.pdf>>. Acesso em: 29 abr. 2015.

Fonte: o autor.

Ótimo! Vamos juntar tudo?

O processo de software é um conjunto de atividades que nos orienta durante o desenvolvimento do software. Um processo irá determinar as responsabilidades de cada um dos envolvidos, o prazo e quais ferramentas serão utilizadas.

Existem inúmeras propostas e sugestões para modelos de processo de softwares, todas baseadas em experiências bem ou malsucedidas. Não existe um processo ideal, o que a literatura propõe são “boas práticas” que podem ser adaptadas para cada empresa ou profissional e, ainda assim, dentro da mesma empresa, para cada projeto, deve ser estabelecido seu próprio processo, considerando suas exigências e realidade comercial. Essas propostas são conhecidas como modelo de processo de software. Um processo de software pode ser visto como um conjunto de atividades organizadas e utilizadas para definir, desenvolver, testar e manter um software. Existem várias propostas de modelos de processo de software, as atividades básicas comuns entre elas, em geral, são: levantamento de requisitos; análise de requisitos; projeto; implementação; testes; implantação.

MODELO DE PROCESSO DE SOFTWARE

Um modelo de processo de software é um conjunto de métodos e ferramentas orientados para auxiliar no planejamento, desenvolvimento, controle e manutenção de um software.

Os modelos de processos de software foram sendo desenvolvidos aos poucos, conforme os especialistas esbarravam em um problema e analisavam a melhor forma de resolvê-lo, assim, também foram sendo desenvolvidas suas atualizações, conforme a evolução do mercado de desenvolvimento de software.

Se recorrermos a uma pesquisa básica, na Internet mesmo, encontraremos vários modelos de processo de software propostos: linear (famoso cascata); incremental; orientada a reuso; baseado em componentes, espiral, processo unificado (quem nunca ouviu falar do RUP?); os, atualmente, mais explorados, os ágeis: Scrum, TDD (Test Driven Development), ATDD (Acceptance test-driven development), BDD (BDD – Behavior driven development), XP (eXtreme Programming). Enfim, são inúmeras as propostas existentes de modelos de processos para desenvolvimento de software. Qualquer modelo que você decida estudar, ainda que possua etapas e métodos diferentes, perceberá que apresenta os mesmos objetivos finais: cumprir o prazo, cumprir o custo, entregar um produto de qualidade e único; e, também, exige as mesmas condições: pessoas treinadas e motivadas, processos claros e definidos e ferramentas adequadas.

Na prática, o que precisamos identificar é o processo de software que melhor se adapte à realidade do nosso negócio e que atenda às necessidades de produção do software a ser desenvolvido. Sommerville (2011, p. 19) me apoia nessa afirmação quando diz que: “não existe um processo ideal, a maioria das organizações desenvolve os próprios processos de desenvolvimento de software”.

Importante registrar que existe uma pequena distinção entre processo e modelo de software: o processo de software é intrínseco a qualquer novo projeto de software, pois trata da necessidade de um software ser desenvolvido de forma completa e que atenda a todas as expectativas do cliente, enquanto o modelo é a forma onde serão trabalhados todos os passos para atingir os objetivos de desenvolver software.

REFLITA



Uma linguagem de modelagem é suficiente no processo de desenvolvimento de um software?

Fonte: o autor.

ATIVIDADES FUNDAMENTAIS DO PROCESSO DE SOFTWARE

Todo processo de software deve cumprir todas as etapas que foram estabelecidas para ele, ainda que você desenvolva o seu próprio modelo. Você deve estar se perguntando: que etapas são essas? As etapas são as atividades estabelecidas em conjunto com sua equipe de trabalho e que serão tratadas em cada uma das fases do processo.

Pressman (2010, p. 18) diz que

um arcabouço de processo estabelece o alicerce para um processo de software completo pela identificação de um pequeno número de atividades aplicáveis a todos os projetos de software, independentemente de seu tamanho ou complexidade.

Sommerville (2011) considera as seguintes atividades como fundamentais para a engenharia de software: especificação de software; projeto de software; validação de software; e evolução de software.

- Especificação de software: define a funcionalidade do software e as restrições sobre sua operação.
- Projeto de software: define as funcionalidades que deverão ser desenvolvidas para que se obtenha como produto final o software especificado pelo cliente.
- Validação de software: o software deve ser validado para garantir que faça o que o cliente deseja.
- Evolução de software: o software deve evoluir para atender aos novos requisitos que, naturalmente, surgirão.

O software é um produto resultante de uma atividade cognitiva. A cognição está diretamente relacionada com fatores diversos, como o pensamento, a linguagem,

a percepção, a memória e o raciocínio, que fazem parte do desenvolvimento intelectual. Considerando que a atividade de desenvolver um software é intelectual, ela, então, também pode ser classificada como de natureza artística, isto é, depende de pessoas para ser realizada. Portanto desenvolver um software é um processo complexo, intelectual e criativo que apresenta, para cada requisito, diferentes propostas de solução e que dependem de pessoas, com diferentes intelectos, para criar e decidir a melhor solução para os problemas apresentados. Nesse contexto, um Processo de Software bem definido garante sistematização, regras, prazos, ferramentas e outros procedimentos padronizados que auxiliam as pessoas envolvidas na aplicação do seu intelecto para a produção das ações desejadas para o produto.

Um processo de software deve estabelecer atividades que envolvam todos os aspectos do desenvolvimento, desde a **concepção do sistema**, ou seja, sua ideia, até suas exceções, aquilo que não será resolvido pelo software, suas funcionalidades, quais são as responsabilidades e objetivos do produto a ser desenvolvido. Devem estar definidas as atividades de **modelagem do software**, em que será feita a interpretação dos requisitos para a produção de uma descrição de todos os componentes que serão necessários para a codificação do software. Acima, denominamos essa fase de projeto de software, projeto aqui, nessa fase, pode ser interpretado como arquitetural, em que técnicas e/ou ferramentas de modelagem serão utilizadas para produzir a descrição da arquitetura do produto a ser desenvolvido, definição dos módulos e seus relacionamentos, definição da interface, como será a interação do produto com o usuário, e organizado, de forma a ser entendível tanto para o cliente quanto para a equipe de desenvolvimento. Riscos, custos, prazos podem ser definidos nessa fase também.

As atividades de **codificação** são as próximas a serem contempladas no processo de software em que serão estabelecidas as iterações e interações, se serão feitos testes nessa fase, como serão tratados os impedimentos e como será executada a **validação** do software, por pacotes, ou depois de o produto pronto, ou, ainda, como uma fase anterior à codificação. O processo de software deve considerar, também, **manutenção** do produto desenvolvido, em que, praticamente, todas as fases anteriores são abrangidas quando é determinado o ciclo de manutenção. Vale registrar aqui que todas essas informações devem ser consideradas pelo modelo de processo de software, e todas as atividades estão vinculadas ao

desenvolvimento do produto final, qual ordem executá-las e quais atividades acrescentar variam para cada modelo de processo de software.

O objetivo desta disciplina é aprofundar o conhecimento na atividade de especificação de software, também conhecida como engenharia de requisitos, as demais atividades do processo de software serão trabalhadas, em momento oportuno, por outras disciplinas.

IMPORTÂNCIA DA ENGENHARIA DE REQUISITOS NO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

A Figura 3 é um clássico na área do desenvolvimento de software. Ela demonstra exatamente o que acontece durante o processo de levantamento de informações sobre um produto a ser desenvolvido e podemos abstrai-la para a engenharia de requisitos.

No tópico anterior, falamos, brevemente, sobre a conferência organizada para tratar dos assuntos relaciona-

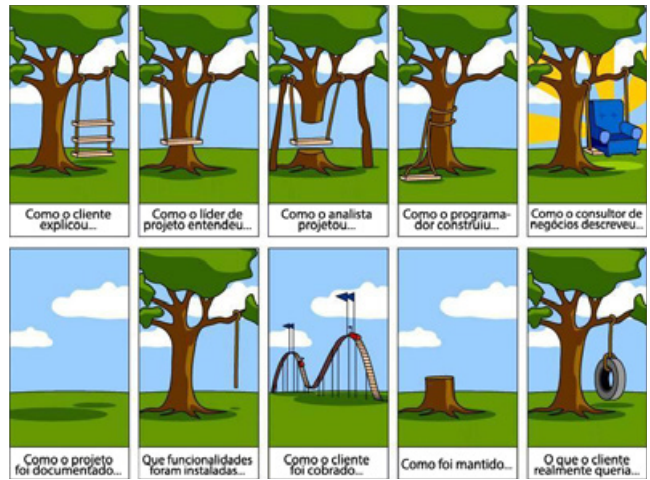


Figura 3: Processo de comunicação durante o desenvolvimento de projeto.
Fonte: Corazzin (online).

dos ao desenvolvimento de software, que ficou conhecida como Crise do Software. A conferência, que aconteceu em 1968, foi organizada para tratar de assuntos como a complexidade dos problemas do desenvolvimento do software, a ausência de técnicas bem estabelecidas e a crescente demanda por novas aplicações e teve como objetivo estabelecer práticas mais maduras para o processo de desenvolvimento. As causas da crise do software estão ligadas à complexidade do processo de software e à falta de metodologias para tratá-las durante o processo de desenvolvimento, o que promovia, principalmente, os seguintes problemas:



- Projetos estourando o orçamento.
- Projetos estourando o prazo.
- Software de baixa qualidade.
- Softwares, muitas vezes, não atingiam os requisitos.
- Projetos não gerenciáveis.
- Código difícil de manter.

Levanto novamente este assunto para propor uma analogia entre o cenário de desenvolvimento de software em 1968 e o atual. O que mudou?

Negativamente, percebemos que todas as conquistas citadas acima e mais cinquenta anos de experiência no desenvolvimento de software não bastaram para melhorar efetivamente a qualidade do software. O pmsurvey.org publica, anualmente, uma pesquisa promovida e organizada pelo *Project Management Institute*, que conta com a participação de centenas de organizações no mundo. Os resultados da pesquisa de 2012 mostram que cerca de 60% das organizações têm cumprido as metas dos fatores críticos de sucesso, conforme indicado na tabela a seguir.

FATORES CRÍTICOS DE SUCESSO EM PROJETOS	
1. Problemas de comunicação.	10. Estimativas incorretas ou sem fundamento.
2. Não cumprimento de prazos.	11. Problemas com fornecedores.
3. Escopo não definido adequadamente.	12. Retrabalho em função da falta de qualidade do produto.
4. Mudanças de escopo constantes.	13. Falta de definição de responsabilidades.
5. Recursos humanos insuficientes.	14. Falta de apoio na alta administração/patrocinador.
6. Riscos não avaliados corretamente.	15. Falta de competência para gerenciar projetos
7. Concorrências entre o dia a dia e o projeto na utilização dos recursos.	16. Falta de uma metodologia de apoio.
8. Mudanças de prioridade constantes ou falta de prioridade.	17. Falta de uma ferramenta de apoio.
9. Não cumprimento do orçamento estabelecido.	18. Clientes não satisfeitos.

Tabela 1: Fatores Críticos de Sucesso em Projetos

Fonte: [Pmsurvey.org](http://pmsurvey.org) (2012).

O *The Standish Group* publica, a cada dois anos, uma pesquisa chamada *Chaos Report*, que aponta o percentual de projetos, na área de TI, que alcançam sucesso, déficit (atraso/prejuízo) ou fracasso. O relatório de 2013 aponta que apenas 39% dos projetos de TI no mundo alcançam sucesso.

Positivamente, vários modelos de processo de desenvolvimento de software foram propostos, várias ferramentas para automação das atividades do processo foram criadas. Artefatos de gerenciamento de projeto de software foram propostos. A gestão de projetos já há algum tempo vem conquistando espaço estratégico nas organizações, como comprovado pelos estudos promovidos pelo *Project Management Institute* Brasil (PMI), entidade internacional que regula-menta os profissionais do setor.

Temos, ainda, nesse intervalo, o manifesto ágil, uma reunião de dezessete pes-soas em 2001, no *The Lodge at Snowbird Resort*, que resultou no Manifesto para o desenvolvimento Ágil de Software, o qual estudaremos mais a fundo na unidade V, e, finalmente, o cenário da crescente adoção de modelos de maturidade, como MPS. BR (Melhoria do Processo de Software Brasileiro), da SOFTEX, e CMMI (Capability Maturity Model Integration), da SEI, é um indicador de que as empresas nacio-nais estão se preocupando com a qualidade dos produtos e serviços que oferecem.

Toda essa conversa foi necessária para dar suporte à afirmação que farei a seguir: o principal causador de falhas dos projetos de software são falhas no pro-cesso de Engenharia de Requisitos.

Considere comigo: tudo – especificação, projeto, modelagem, programação, testes, implantação, manutenção - parte do entendimento e da determinação de “o que se pretende fazer”. O descompromisso com essa determinação causa incre-mentos não desejados, como: atrasos no cronograma, custos acima do previsto, falta de recursos humanos, alteração dos requisitos, alteração das especificações, flexibilização comprometedora do escopo, qualidade abaixo da esperada, frus-tração do cliente e outros tantos problemas.

Outro recurso que utilizo para ilustrar a importância da Engenharia de Requisitos no processo de desenvolvimento de software são os estudos pro-movidos pelo *Standish Group*. Eles mostram que o alto índice de manutenção, retrabalho em nível de requisitos, projeto, codificação, testes é causado por falhas de definição nas fases iniciais do processo de desenvolvimento.



	% DO CUSTO DE DESENVOLVIMENTO	% DOS ERROS INTRODUZIDOS	% DOS ERROS ENCONTRADOS	CUSTO RELATIVO DE CORREÇÃO
Análise de Requisitos	5	55	18	1
Projeto	25	30	10	1 ~ 1,5
Codificação e teste de unidade	50			
Teste	10	10	50	1 ~ 5
Validação e documentação	10			
manutenção		5	22	10 ~ 100

Tabela 2: Levantamento de custos com erros nas fases de projetos de software

Fonte: Standish Group – CHAOS Report 2013 on software project success and failure.

O que a Tabela 2 nos aponta é, primeiramente, que os custos relacionados à análise de requisitos são baixos, nessa fase, a maioria dos erros é apresentada. Observe que, se os erros apresentados fossem corrigidos nessa fase, os custos de correção ficariam baixíssimos, enquanto, se deixarmos para corrigi-los depois da validação, lá na fase da manutenção (última linha da tabela), os custos aumentariam exorbitantemente.

O mesmo estudo apresenta fatores críticos para o sucesso dos projetos de software. Os dez mais lembrados estão relacionados na tabela a seguir. Observe que três principais fatores para o sucesso estão relacionados às atividades da Engenharia de Requisitos: requisitos incompletos, falta de envolvimento do usuário, mudança de requisitos e especificações.

FATORES CRÍTICOS PARA O SUCESSO DE PROJETOS DE SOFTWARE	
Fatores	Porcentagem
Requisitos Incompletos	13,1
Falta de Envolvimento do Usuário	12,4
Falta de Recursos	10,6
Expectativas Irreais	9,9
Falta de apoio Executivo	9,3
Mudança de Requisitos e Especificações	8,7
Falta de Planejamento	8,1
Sistema não mais necessário	7,5

Tabela 3: Fatores Críticos para o Sucesso de Projetos de Software

Fonte: Standish Group – CHAOS Report 2013 on software project success and failure.

Será que alcancei meu objetivo de convencê-lo sobre a importância da Engenharia de Requisitos no processo de produção de um software? É palpável que o trabalho mais criterioso na fase inicial do processo de software garante o sucesso de todo o projeto.

Encerro o assunto citando Pressman (2010, p. 117): “engenharia de requisitos estabelece uma base sólida para o projeto e a construção. Sem ela o software resultante tem uma alta probabilidade de não satisfazer as necessidades do cliente”. Isto é, o entendimento do “o que fazer” para sua catalogação e modelagem eficiente representará a real necessidade do cliente, que se trata do ponto-chave para a qualidade do software.

REFLITA



Há mais de quarenta anos, a engenharia de software nos apresenta a importância da fase de levantamento e análise dos requisitos, para o desenvolvimento de software, pesquisas renomadas comprovam que um levantamento mal executado é a principal causa dos estouros nos projetos de desenvolvimento de software, porque, ainda hoje, enfrentamos esses mesmos problemas?

Fonte: Galeote (online).

CONSIDERAÇÕES FINAIS

Nesta unidade, revisamos alguns conceitos básicos que você, talvez, já tenha conhecido na sua formação. Este nivelamento foi necessário para podermos situar a engenharia de requisitos nas atividades da engenharia de software.

Definimos que a engenharia de software é uma abordagem sistemática que envolverá todos os aspectos de desenvolvimento do software: o projeto, a análise e a construção do software para algum objetivo específico. Formulamos, juntos, a definição de processo como uma atividade organizada em etapas específicas a serem desenvolvidas de forma contínua, que mantenha uma unidade, a fim de alcançar um propósito final, que é um produto de software de qualidade. Sendo

assim, o processo de software representa um conjunto de atividades que nos orienta durante o desenvolvimento do software.

O CMMI (*Capability Maturity Model Integration*) é um modelo de capacitação de processos de software, desenvolvido pelo SEI (*Software Engineering Institute*) e patrocinado pelo Departamento de Defesa Americano (DoD), para a avaliação da capacidade das fábricas de software de repetir uma série de resultados de uma maneira previsível e com qualidade. Como profissionais da área de desenvolvimento de software, é muito importante entender como os conceitos se encaixam na prática, o que o CMMI e outros modelos de maturidade buscam é avaliar a qualidade de gestão de processos das empresas desenvolvedoras de software, então, percebemos que, para produzir um produto de qualidade, dependemos da qualidade de seu processo de desenvolvimento, que está intimamente ligado com o processo da engenharia de requisitos.

Lembre-se de que não existe um processo ideal, deve ser estabelecido um conjunto de atividades específicas para cada projeto e, ainda que dentro de uma mesma empresa, para cada projeto, deve ser estabelecido seu próprio processo, considerando suas exigências e realidade comercial.

ATIVIDADES



1. A definição “é a abordagem sistemática que envolverá todos os aspectos de desenvolvimento do software: o projeto, a análise e a construção do software para algum objetivo específico” diz respeito a:
 - a. Processo de software.
 - b. Engenharia de software.
 - c. Software.
 - d. Engenharia de requisitos.

2. Assinale a alternativa correta sobre Processo de Software:
 - I. Conjunto de atividades bem definidas, com responsáveis, com artefatos de entrada e saída, com dependências entre elas e ordem de execução, com modelo de ciclo de vida.
 - II. Conjunto parcialmente ordenado de atividades (ou passos) para se atingir um objetivo, definindo quem está fazendo o que, quando e como para atingir um certo objetivo.
 - III. O processo de software é um conjunto de atividades que nos orienta durante o desenvolvimento do software. Um processo irá determinar as responsabilidades de cada um dos envolvidos, o prazo e quais ferramentas serão utilizadas.
 - a. Apenas I está correta.
 - b. Apenas II está correta.
 - c. Apenas III estão corretas.
 - d. I, II e III estão corretas.

3. “Um modelo de processo de software é um conjunto de métodos e ferramentas orientados para auxiliar no planejamento, desenvolvimento, controle e manutenção de um software”. Sobre modelo de processo de software, analise as alternativas abaixo:
 - I. Não existe um processo ideal, a maioria das organizações desenvolve os próprios processos de desenvolvimento de software.
 - II. Um modelo de processo de software é uma descrição simplificada do processo de desenvolvimento de software.
 - III. Os modelos de processo de software incluem as atividades, que fazem parte do processo de desenvolvimento de software.

ATIVIDADES



- a. Apenas I está correta.
 - b. Apenas I e III estão corretas.
 - c. Apenas II e III estão corretas.
 - d. I, II e III estão corretas
4. "Um arcabouço de processo estabelece o alicerce para um processo de software completo pela identificação de um pequeno número de atividades aplicáveis a todos os projetos de software, independentemente de seu tamanho ou complexidade". Essa é a definição de Pressman (2010) para Processo de Software. Em relação a Processo de Software, qual alternativa abaixo está correta?
- a. A atividade especificação de software: define as funcionalidades que deverão ser desenvolvidas para que se obtenha como produto final o software especificado pelo cliente.
 - b. A atividade de projeto de software: define a funcionalidade do software e as restrições sobre sua operação.
 - c. A atividade de validação de software: o software deve ser reescrito e mantido para garantir que faça o que o cliente deseja.
 - d. Evolução de software: o software deve evoluir para atender aos novos requisitos que, naturalmente, surgirão.
5. Os estudos promovidos pelo Standish Group mostram que o alto índice de manutenção, retrabalho em nível de requisitos, projeto, codificação, testes é causado por falhas de definição nas fases iniciais do processo de desenvolvimento. Selecione a alternativa que relaciona uma atividade inicial, em um processo de software:
- a. Levantamento de requisitos.
 - b. Teste de software.
 - c. Programação.
 - d. Implantação.



DESENVOLVIMENTO DE SOFTWARE REQUER PROCESSO DE GESTÃO.

Software, de modo genérico, é uma entidade que se encontra em quase constante estado de mudança. Essas mudanças ocorrem por necessidade de corrigir erros existentes no software e/ou de adicionar novos recursos e funcionalidades. Igualmente, os sistemas computacionais (isto é, aqueles que têm software como um de seus principais elementos) também sofrem mudanças freqüentemente. Essa necessidade evolutiva do sistema de software o torna predisposto a defeitos (isto é 'não confiável'), podendo resultar em atraso na entrega e em custos acima do estimado. Concomitante com esses fatos, o crescimento em tamanho e complexidade dos sistemas de software exige que os profissionais da área raciocinem, projetem, codifiquem e se comuniquem por meio de componentes (de software) e qualquer concepção ou solução de sistema passa então para o nível arquitetural. Nesse sentido, o objetivo deste artigo é discutir e explorar o fato de que software não é construído como uma TV ou geladeira, mas desenvolvido.

Quase cinco décadas atrás software constituía uma insignificante parte dos sistemas existentes e seu custo de desenvolvimento e manutenção eram desprezíveis. Para perceber isso, basta olharmos para história da indústria do software (www.softwarehistory.org). Encontra-se o uso do software numa ampla variedade de aplicações tais como sistemas de manufatura, software científico, software embarcado, robótica

e aplicações Web, dentre tantas. Paralelamente, observou-se o surgimento de várias técnicas de modelagem e projeto bem como de linguagens de programação. Perceba que o cenário existente, décadas atrás, mudou completamente. Antigamente, os projetos de sistemas alocavam pequena parcela ao software. Os componentes de hardware, diferentemente, eram analisados e testados quase exaustivamente o que permitia a produção rápida de grandes quantidades de subsistemas e implicava em raros erros de projetos. Entretanto, a facilidade inicial de modificar o software, comparativamente ao hardware, motivou seu uso. A intensificação do uso do software numa larga variedade de aplicações o fez crescer em tamanho e complexidade. Isto tornou proibitivo analisar e testar exaustivamente, além de impactar no custo de manutenção.

Perceba que à medida que tamanho e complexidade dos sistemas de software aumentam, o problema de projeto extrapola as estruturas de dados e algoritmos de computação. Ou seja, há necessidade de considerar o projeto da arquitetura (ou estrutura geral) do sistema, exigindo também um processo de desenvolvimento de software. O ciclo de vida de um produto compreende um conjunto de atividades que vai desde sua concepção até a entrega desse produto ao cliente, envolvendo ainda sua evolução e, eventualmente, retirada desse produto.

Fonte: Filho (2011, online).





LIVRO

Engenharia de Software - Uma Abordagem Profissional

Roger S. Pressman

Editora: Bookman

Sinopse: Versão concebida tanto para alunos de graduação quanto para profissionais da área. Oferece uma abordagem contemporânea sobre produção do software, gestão da qualidade, gerenciamento de projetos, com didática eficiente e exercícios de grande aplicação prática.



FUNDAMENTOS DA ENGENHARIA DE REQUISITOS



Objetivos de Aprendizagem

- Conhecer os fundamentos da engenharia de requisitos e definir seus principais artefatos de saída.
- Entender a importância da engenharia de requisitos no contexto de desenvolvimento de software.

Plano de Estudo

A seguir, apresentam-se os tópicos que você estudará nesta unidade:

- Conceitos fundamentais da engenharia de requisitos
- *Stakeholders*
- Classificação dos requisitos
- Documentação dos requisitos de software
- Template documento de requisitos
- Matriz de rastreabilidade de requisitos
- Template matriz de rastreabilidade de requisitos

INTRODUÇÃO

Olá! Vamos para mais uma unidade de estudos? Avançando em nosso aprendizado, estudaremos os conceitos fundamentais da engenharia de requisitos. Não é possível entender o propósito de uma atividade sem, primeiro, compreender os elementos que a compõem.

O primeiro elemento que devemos dominar seu significado é requisitos. Todo nosso curso baseia-se nele. Entender do que se trata e como ele deve ser tratado é crucial para o sucesso da engenharia de requisitos. Observaremos, no decorrer da unidade, que os requisitos delimitam o software, pois estabelecem as funcionalidades e as restrições solicitadas por um conjunto de *stakeholders*. O processo de especificação e documentação de requisitos é fundamental para o sucesso de um projeto de software e entenderemos que um bom documento de requisitos possibilita estimativas de custos mais precisas, cronogramas mais ajustados e um produto final que atende às expectativas do cliente.

Outro elemento importante na engenharia de requisitos é o *stakeholder*. Os requisitos de software são levantados e mapeados por meio da análise dos *stakeholders*. Como cada grupo de *stakeholder* tem interesses específicos e particulares, eles podem, ao invés de colaborar, atrapalhar o desenvolvimento do software, se não forem bem gerenciados. Nesta unidade, conheceremos o que significa *stakeholders* e entenderemos o tipo de influência que eles podem causar ao projeto de desenvolvimento de software.

O objetivo desta unidade, além de garantir o conhecimento sobre os principais elementos da engenharia de requisitos, é assimilar o que foi apresentado na unidade anterior sobre a importância da engenharia de requisitos enquanto principal causadora de falhas dos projetos de software.

Boa leitura!

CONCEITOS FUNDAMENTAIS DA ENGENHARIA DE REQUISITOS

A engenharia de requisitos, segundo Sommerville (2008), envolve todas as atividades de conceituação, documentação e manutenção de um conjunto de requisitos para um sistema baseado em computador.

Vamos observar o significado das palavras em separado:

- Engenharia¹- substantivo feminino: aplicação de métodos científicos ou empíricos à utilização dos recursos da natureza em benefício do ser humano
- Requisito²- adjetivo: 1. p.us. que foi requisitado, requerido; 2. substantivo masculino: condição para se alcançar determinado fim

Juntado os conceitos: engenharia de requisitos pode ser definida, então, como conjunto de atividades sistemáticas como condição para se alcançar determinado fim, o que se foi requisitado. Nesse sentido, Sommerville (2003, p. 103) nos positiva com sua definição: “engenharia de requisitos é um processo que envolve todas as atividades exigidas para criar e manter o documento de requisitos de sistema”.

“Conjunto de atividades” pode ser entendido como “processo”, e sobre processo, nós entendemos! Assim, o propósito da engenharia de requisitos está relacionado com a identificação de metas a serem atingidas: qualidade de software; produtividade no desenvolvimento, operação e manutenção de software; permitir que profissionais tenham controle sobre o desenvolvimento de software dentro de custos, prazos e níveis de qualidade desejados.

1 Resultado da busca pelo termo “defina engenharia” no Google.

2 Resultado da busca pelo termo “defina requisito” no Google.

REFLITA



Qualidade de software é definida pelo IEEE (*The Institute of Electrical and Electronics Engineers*) como “o grau com que um sistema, componente ou processo atende (1) aos requisitos especificados e (2) às expectativas ou necessidades de clientes ou usuários”. Enquanto profissionais de desenvolvimento de software, carregamos nossos conhecimentos para a aplicação prática?

Fonte: Sayão, Staa e Leite (2003, p. 9).

SAIBA MAIS



A ABES Software - Associação Brasileira das Empresas de Software publica, anualmente, o Estudo do Mercado Brasileiro de Software - Panorama e Tendências. Os dados levantados para as publicações são consolidados por cinquenta escritórios da IDC - *International Data Corporation* divididos em seis regiões mundiais.

O relatório final apresenta com riqueza de detalhes dados sobre os mercados brasileiro e mundial de software, é uma ferramenta importante para os profissionais da área, é uma ferramenta importante para análise do setor.

O Relatório é disponibilizado sem custos pelo portal da ABES. Vale a pena visitar e se manter atualizado sobre o seu setor de atuação profissional.

Para saber mais, acesse o link disponível em: <<http://www.abessoftware.com.br/>>. Acesso em: 27 mar. 2015.

Fonte: autor.

STAKEHOLDERS

Vimos que a qualidade de um produto de software está diretamente relacionada ao atendimento das expectativas do cliente. Sabemos, também, que o sucesso de um projeto está intimamente relacionado à administração de problemas ligados à especificação e gerenciamento dos requisitos do software. Você percebe o elemento comum nas duas exigências? Exatamente! Tanto para qualidade quanto

para o levantamento dos requisitos, será necessária a participação do cliente, um estranho a sua organização e a você que deverá ser incluído, ouvido e interpretado, durante todo o processo de desenvolvimento de software e, principalmente, na fase da Engenharia de Requisitos.

Esse estranho na área de projetos é reconhecido como *Stakeholders*. Para o PMBOK® (online), *stakeholders* são “pessoas e organizações, como clientes, patrocinadores, organizações executoras e o público, que estejam ativamente envolvidas no projeto ou cujos interesses possam ser afetados de forma positiva ou negativa pela execução ou término do projeto”. Podemos resumir *stakeholders* como qualquer pessoa ou organização que realiza ou sofre alguma ação do projeto.



SAIBA MAIS

A partir da sua terceira edição, o gerenciamento de *stakeholders* foi abordado pelo Guia PMBOK® (guia de práticas na gestão de projetos - *Project Management Body of Knowledge* – da PMI), por meio da área de conhecimento “Gerenciamento das Comunicações do Projeto”, processo “Gerenciar as Partes Interessadas”. O enfoque do Guia PMBOK® até sua última versão foi a importância da comunicação entre os envolvidos no projeto, tanto os do ambiente interno quanto os externos, entretanto, com o desenvolvimento da área de projetos, percebeu-se que, para garantir o sucesso de um projeto, deveria ser definido um método exclusivo para gerenciar os grupos e relacionamentos que geram resultados estratégicos para os projetos, e o gerenciamento de *stakeholders* subiu de processo para área de conhecimento. Isso prova que a interferência desse elemento influencia diretamente o resultado do projeto.

Para saber mais, acesse os links disponíveis em: <<https://brasil.pmi.org/>> (PMI no Brasil); <<http://www.pmi.org/>> (*Project Management Institute*). Acesso em: 8 jul. 2015.

Fonte: o autor.

Enquanto estiver envolvido somente na engenharia de requisitos, você não terá que se preocupar com o gerenciamento e controle de todos os *stakeholders* envolvidos no projeto, mas é de extrema importância que execute o levantamento de,

no mínimo, aqueles que podem influenciar na definição e manutenção dos requisitos que definirão o escopo final do produto. Dificilmente, o usuário final do seu software será o contratante ou o patrocinador, por isso sua atenção tem que estar focada além deles, para que o produto final atenda às necessidades gerenciais e operacionais para as quais foi contratado.

O gerenciamento de identificação dos interessados é complexo e possui metodologias e ferramentas próprias, seria necessária uma disciplina só para elas. Para garantir que o projeto tenha sucesso, sugiro que, no mínimo, você execute as cinco etapas a seguir, mantendo o foco nos interesses que podem influenciar na definição e na manutenção dos requisitos e do escopo final do produto:

- 1ª Identificar os stakeholders: relacionar todas as pessoas que influenciam o projeto ou são influenciadas por ele. Mesmo que seja uma organização ou grupo, é importante identificar as pessoas que o representam.
- 2ª Caracterizar os stakeholders quanto à ação que tomam em relação ao projeto: classificar os stakeholders em quatro grupos, de acordo com a atitude que tomam em relação ao projeto:
 - Favoráveis: contribuem para o sucesso do projeto. Demonstram interesse e responsabilidade pelo projeto.
 - Neutros: não possuem influência e não estão interessados nos resultados do projeto.
 - Contrários: acreditam que o projeto prejudica seus interesses. Criam empecilhos para o andamento do projeto.
 - Sabotadores: de difícil identificação. Geralmente, espiões e concorrentes.
- 3ª Priorizar os Stakeholders: classificar os stakeholders quanto ao interesse e poder (influência) que eles podem ter sobre o projeto. A partir dessa classificação, você determinará como será sua comunicação com cada um deles. Essa priorização pode ser feita construindo uma matriz estratégica:

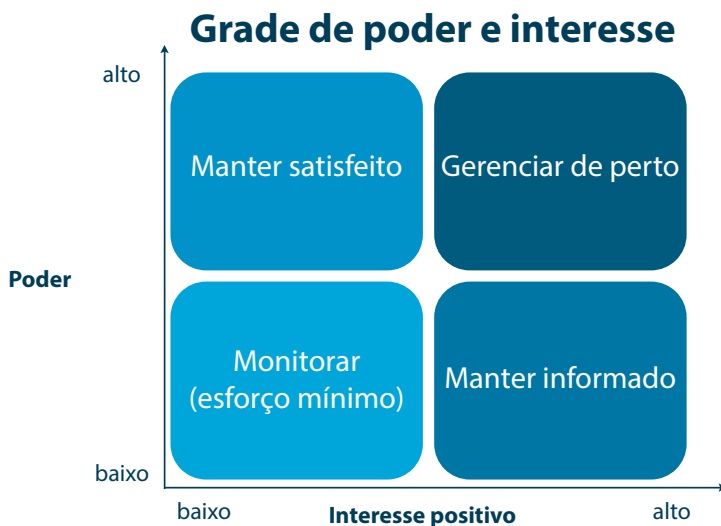


Figura 5: Matriz estratégica dos stakeholders

Fonte: PMKB (online).

- 4ª Comunicar os stakeholders: é importante manter um canal de comunicação com os interessados eleitos. Esse canal pode ser estabelecido por diversos meios: telefone, e-mail, comunicado, grupo de discussão, relatórios etc. Você define o nível e a periodicidade das informações para cada um. O Importante é que, depois de planejado, seja cumprido.
- 5ª Gerenciar os stakeholders: estar atento aos empecilhos que podem prejudicar o seu projeto e responder prontamente a cada problema identificado.

CLASSIFICAÇÃO DOS REQUISITOS

Vimos que o significado da palavra requisito é condição para se alcançar determinado fim, o que seria, então, um requisito de software?

Sommerville (2008) diz que um requisito é uma descrição detalhada de uma função do sistema. Requisitos de software já foram considerados sinônimos de funções do sistema, ou seja, tudo que o software deveria fazer era considerado requisito.

Com a evolução dos softwares e do entendimento da engenharia de sistemas, percebeu-se que requisitos são, além de funções, objetivos, propriedades e restrições que o software deve possuir para satisfazer contratos, padrões ou especificações. O importante é ter em mente que os requisitos de software devem sempre ser trabalhados para resolver os problemas do cliente e sob o ponto de vista dele.

Mantendo a linha de raciocínio, uma coleção de requisitos de software representa o produto final negociado entre as partes interessadas e o engenheiro de requisitos. Concorda? Como montar esse documento então? Primeiro passo, temos que entender os tipos de requisitos de software que existem e como eles se classificam.

Existe uma convenção, que vem da literatura clássica, definindo tipos de requisitos no que diz respeito à forma de registro, isto é, a maneira como ele será escrito no documento e a quem se destina.

TIPO DE REQUISITO		
Tipo Requisito	Definição	A quem se destina
Requisitos de Usuário	Os requisitos são escritos com alto nível de abstração, em linguagem natural, e podem conter diagramas simples e representativos das funcionalidades que o sistema deve apresentar e das restrições que ele deve operar.	<ul style="list-style-type: none"> • Gerentes de clientes. • Usuário final do sistema. • Engenheiro do cliente. • Arquiteto do sistema.
Requisitos de Sistema	O nível de detalhamento na descrição dos requisitos é aumentado. Requisitos do sistema informam, de maneira detalhada, o que o produto final deve fazer. Pode servir de contrato entre o comprador do sistema e o desenvolvedor. Aqui, um único requisito de usuário pode ser dividido em vários requisitos do sistema.	<ul style="list-style-type: none"> • Usuário final do sistema. • Engenheiro do cliente. • Arquiteto do sistema. • Desenvolvedor do software.
Requisitos de Projeto	Ou especificação de projeto, é a definição do projeto de software em nível mais técnico – modelagem.	<ul style="list-style-type: none"> • Engenheiro do cliente. • Arquiteto do sistema. • Desenvolvedor do software.

Quadro 3: Tipo de Requisitos

Fonte: o autor adaptado de Sommerville (2008) e Pressman (2010).

Nessa abordagem, os autores orientam a escrita de documentos separados para os usuários finais e para os desenvolvedores. Em qualquer prova de concurso, ou abordagem teórica, é exatamente isso que você deve responder, mas, na prática, o desenvolvimento de um único documento que seja claro, conciso e que contenha as especificações tanto para as necessidades dos usuários quanto dos desenvolvedores é o suficiente e mais eficiente.

Além dos tipos, os requisitos são classificados em requisitos funcionais, requisitos não funcionais e requisitos de domínio.

CLASSIFICAÇÃO DE REQUISITOS		
Classificação	Definição	Exemplos
Requisitos Funcionais	São os requisitos diretamente ligados à funcionalidade do software, descrevem as funções que o software deve executar. Os requisitos funcionais podem ser interpretados como um conjunto de entradas, processamento e saída. Representam cálculos, lógicas e as funcionalidades específicas que definem o que o software irá realizar. O implemento destes requisitos caracteriza um sistema.	<ul style="list-style-type: none"> • Requisitos de entrada: o Sistema deve cadastrar médicos profissionais. • Requisito de saída: o Sistema deve emitir um relatório de clientes. • Requisito de mudança de estado: O Sistema deve passar um cliente da situação “em consulta” para “consultado”, quando o cliente terminar de ser atendido.
Requisitos não funcionais	São os requisitos que expressam as condições ou especialidades que o software deve atender. São as restrições a serem impostas relacionadas a desempenho, segurança, usabilidade, portabilidade, facilidade de uso, eficiência etc.	<ul style="list-style-type: none"> • Controle de acesso somente para pessoas autorizadas. • Tempo de resposta limitado em vinte segundos.
Requisito de domínio	São aqueles derivados do domínio da aplicação, ou seja, da especialidade onde o software será implantado e, também, do próprio software, que refletem nele. Podem ser novos requisitos funcionais, ou restrições sobre requisitos existentes, ou cálculos específicos. Esses requisitos são expressos com o uso de uma linguagem específica do domínio da aplicação e, geralmente, de difícil compreensão para os engenheiros de software.	<ul style="list-style-type: none"> • Cálculo de impostos ou taxas específicas: • Área de RH possui incrementos e descontos específicos do seu domínio – férias, horas extras etc. • Área tributária, cálculos e porcentagens específicas que são inerentes a sua aplicação – IPTU, ISS etc. • Área comercial, exige cálculos que só dizem respeito a sua especialidade – PIS, COFINS etc.

Quadro 4: Classificação de requisitos

Fonte: adaptado de Sommerville (2008) e Pressman (2010).

Sommerville (2008) classifica os requisitos não funcionais em requisitos de produto, requisitos organizacionais e requisitos externos. Os requisitos de produto especificam o comportamento do produto e podem ser subdivididos em requisitos de usabilidade, de eficiência, de confiança e de proteção. Os requisitos organizacionais são elaborados a partir das políticas e procedimentos da empresa, do cliente e do desenvolvedor e são subdivididos em requisitos ambientais, operacionais e de desenvolvimento. Por último, os requisitos externos envolvem os requisitos referentes aos fatores externos ao software e seu processo de desenvolvimento, que seriam os requisitos reguladores, éticos e legais.

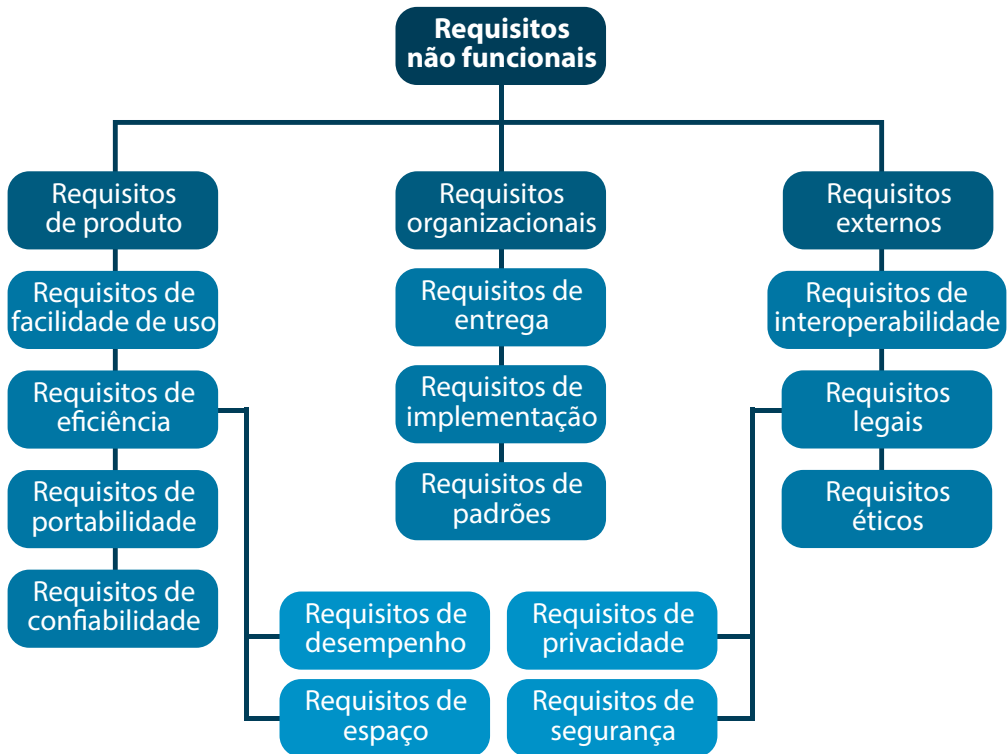


Figura 6: Requisitos Não Funcionais
Fonte: Engenharia de requisitos (online).

Para auxiliar a compreensão, alguns exemplos de requisitos não funcionais e as classificações que podem assumir:

- Requisitos da organização: padrões, infraestrutura.
- Requisitos externos: interoperabilidade, legislação, localização geográfica.
- Requisitos de facilidade de uso: usuários deverão operar o sistema após um determinado tempo de treinamento.
- Requisitos de eficiência: o software deverá processar n requisições por x tempo.
- Requisitos de portabilidade: o software deve ser compatível com qualquer plataforma.
- Requisitos de entrega: um relatório de acompanhamento do desenvolvimento deverá ser fornecido toda segunda-feira.
- Requisitos de implementação: o software será programado na linguagem Java.
- Requisitos de padrões: o software será implementado com orientação ao objeto plataforma Java.
- Requisitos de interoperabilidade: o software deve estar integrado ao Banco de dados SQL Server.
- Requisitos éticos: o sistema não apresentará aos usuários com acesso nível B quaisquer dados do nível A.
- Requisitos legais: o software deverá atender às normas legais, tais como padrões, leis etc.
- Requisitos de integração: o software deve ser integrado com outra aplicação já existente.

Higor Medeiros (online), em seu artigo *Introdução a Requisito de Software*, disponibilizado pela Devmedia, afirma que: “os requisitos não funcionais são geralmente mensuráveis e assim devemos preferencialmente associar uma medida ou referência para cada requisito não funcional” e apresenta uma tabela para ilustrar os requisitos funcionais por outra ótica.

CLASSIFICAÇÃO DE REQUISITOS FUNCIONAIS	
Requisito	Métrica
Velocidade	<ul style="list-style-type: none"> • Transações processadas por segundo. • Tempo de resposta ao usuário/evento. • Tempo de atualização da tela.
Facilidade de uso	<ul style="list-style-type: none"> • Tempo de treinamento. • Numero de telas de ajuda.
Confiabilidade	<ul style="list-style-type: none"> • Tempo médio para falhar. • Probabilidade de indisponibilidade. • Taxa de ocorrência de falhas. • Disponibilidade.
Robustez	<ul style="list-style-type: none"> • Tempo de reinício após falha. • Porcentagem de eventos que causam falhas. • Probabilidade de que os dados sejam corrompidos por falhas.
Portabilidade	<ul style="list-style-type: none"> • Porcentagem de declarações dependentes de sistema-alvo. • Número de sistemas-alvo.

Quadro 5: Classificação de Requisitos Funcionais
Fonte: Medeiros (online).

Perceba que os requisitos não funcionais descrevem as especialidades do software (como ele é), enquanto os requisitos funcionais relacionam suas funcionalidades (o que ele faz).

Sommerville (2008) classifica como requisitos de domínios aqueles requisitos que expressam a regra de negócio a ser trabalhada e interpretada pelo software a ser desenvolvido. Essa interpretação pode ser difícil para os engenheiros de requisitos, uma vez que não conhecem o ambiente e as práticas envolvidas nos processos internos do negócio que irá converter em software. “Os especialistas do domínio podem, por julgarem óbvio, deixarem de fornecer informações importantes e como resultado o requisito pode não ser implementado de forma satisfatória” (SOMMERVILLE, 2004, p. 88).

Parece complicado, mas alguns exemplos simples podem ajudar a compreender os requisitos de domínio:

- O cálculo da média final é dado pela fórmula: $(Nota1 * 2 + Nota2 * 3)/5$, em que Nota1 e Nota2 são dados do próprio processo, não são entradas para a fórmula.
- O pré-requisito para um aluno matricular-se em uma disciplina é que tenha sido aprovado nas disciplinas classificadas como pré-requisitos.
- No desenvolvimento de um software para gestão de operadora de plano de saúde, os requisitos de domínio são conhecimentos específicos desta área de atuação.

Os requisitos de usuários descrevem os requisitos funcionais e não funcionais de forma compreensível pelos futuros usuários do sistema, devem ser relacionados com uma linguagem simples e não técnica. Especificam somente o comportamento externo do sistema. Devido a problemas que podem surgir pelo uso da linguagem natural, como falta de clareza, confusão ou fusão de requisitos, é recomendado que, na formalização dos requisitos para a modelagem do sistema, essa forma de descrição não seja utilizada.

A fim de minimizar divergências na elaboração de requisitos de usuário, Sommerville (2008) recomenda que, para a descrição dos requisitos de usuário, sigam-se os seguintes critérios:

- Crie um formato-padrão e certifique que todas as definições de requisitos estejam de acordo com ele.
- Utilize a linguagem de forma consistente, distinguindo requisitos obrigatórios dos desejáveis.
- Ressalte partes importantes dos requisitos.
- Evite o uso de jargões técnicos.

Finalmente, os requisitos de sistema são descrições mais detalhadas dos requisitos de usuário, eles podem servir de base para um contrato de desenvolvimento de software. Nesse nível de descrição, os requisitos de sistema devem especificar consistentemente todo o sistema a ser desenvolvido. Normalmente, são utilizados como documentação preliminar para o projeto de um novo software.

Como comentei acima, as especificações de requisitos escritas em linguagem natural estão sujeitas a divergências e essas divergências, frequentemente, só são descobertas em fases posteriores do processo de software, o que ocasiona aumento de custo e de tempo no projeto. No caso da decisão por essa abordagem para a documentação dos requisitos, cabe aqui a mesma orientação dada por Sommerville (2008) para a descrição dos requisitos de usuário.

SAIBA MAIS



Na prática, não é fácil a distinção dos requisitos para sua classificação. Buscando uma maneira de facilitar a identificação e classificação dos requisitos, a FURPS+ foi concebida pelo Processo Unificado – UML®. Eles utilizam o acrônimo FURPS para descrever as principais categorias e sua subcategoria: *Functionality* (Funcionalidade); *Usability* (Usabilidade); *Reliability* (Confiabilidade); *Performance* (Performance ou Desempenho); *Supportability* (Suportabilidade); *Plus* (+), assim fica mais fácil lembrar cada um deles, no momento da elicitação dos requisitos. Para saber mais, acesse o link disponível em: <<http://ieeexplore.ieee.org/>>. Acesso em: 8 jul. 2015.

Fonte: o autor.

REQUISITOS NO PROCESSO DE SOFTWARE

É fácil interpretar que os requisitos serão utilizados e considerados somente na fase inicial do processo de desenvolvimento de software, uma vez que a Engenharia de Requisitos é a primeira atividade do processo, mas isso não representa a realidade. Os requisitos participarão de forma direta ou indireta, durante todas as fases do processo de desenvolvimento do software, observe como, de maneira simplificada, relaciono funções para o documento de requisitos como base para atividades em todas as fases do processo de:

- Especificação de software:
 - Definição de critérios de aceitação e validação (pelos *stakeholders*).
 - Definição sobre o que o sistema deve fazer (especificação para a equipe).

- Projeto de software:
 - Alocação de tarefas para a equipe.
 - Estimativa de custo/esforço/cronograma.
 - Acompanhamento e controle do andamento do projeto.
- Validação de software: teste e verificação do sistema desenvolvido.
- Evolução de software: informações para o gerenciamento de mudanças

Por isso, o documento de requisitos deve ser bem elaborado e estar sempre atualizado.

DOCUMENTAÇÃO DOS REQUISITOS DE SOFTWARE

DOCUMENTO DE REQUISITOS

Agora, sabemos que o principal objetivo do processo de definição de requisitos é obter um acordo entre quem solicita e quem desenvolve, estabelecendo, de forma clara e rigorosa, o que deverá ser produzido, isto é, o conjunto de requisitos levantados define o Escopo do produto que será registrado nem um documento oficial, o Documento de Requisitos.

O Documento de Requisitos deve ser um documento de consenso, como dito anteriormente, um acordo entre a equipe de desenvolvimento e o cliente e, também, um ponto de referência para validações futuras. Deve, ainda, garantir uma rastreabilidade mínima, isto é, a possibilidade de identificar, a partir de um determinado requisito, todas as dependências, diretas ou indiretas, quando, por exemplo, surge a necessidade de uma mudança.

Como documentar? Isso é um desafio!

Na prática, o que se percebe é que os procedimentos utilizados para a definição desse acordo são muito variáveis, vão desde métodos pouco estruturados até métodos sistemáticos e burocráticos. Percebe-se, também, que são vários os fatores que influenciam o nível de estruturação do documento, dentre eles, elejo três, que, no meu ponto de vista, são os mais evidentes no mercado: 1. Maturidade técnica da empresa/equipe, 2. Disciplina e 3. Cultura organizacional.

Existem vários modelos sugeridos para documentação, o RUP sugere modelos para documento de requisitos mais complexo – com mais controles – e mais simples, dependendo do tamanho do projeto. O Guia PMBOK® tem a sua sugestão para a documentação de requisitos e, se fizer uma busca pela Internet usando as palavras “documento de requisitos”, irá se deparar com inúmeras formas de apresentação. O desafio aqui é decidir qual o modelo que melhor se adapta ao seu projeto, a sua empresa, a sua equipe.

Independente da abordagem escolhida, é de extrema importância para o projeto que ele exista e que seja útil. Não adianta desenvolver um artefato com trezentas páginas de um único requisito com protótipos, diagramas, descritivas detalhadas e técnicas, *wireframes* que ninguém vai ler ou seguir, ou, no outro extremo, documentar em poucas linhas algo que não representa o problema real.

Como chegar ao nível certo de documentação? Na minha opinião, primeiro, seu documento deve ser capaz de garantir o entendimento dos *stakeholders*. O documento deve demonstrar tanto as funcionalidades do sistema, de forma interpretável por esse público, quanto deve deixar claro que é ele quem garantirá que o que foi descrito ali será contemplado no produto final. Segundo, a equipe técnica deve estar ciente de que esse documento é a única garantia de que está atendendo, de forma completa, às solicitações do cliente, que qualquer solicitação que não esteja documentada ali será considerada mudança e que terão que despende tempo extra para produzi-la.

E agora? Como montar o documento de requisitos?

Várias são as formas usadas para documentar requisitos. Por exemplo, nos anos setenta, usava-se o DFD – Diagrama de Fluxo de Dados - complementado por material descritivo, também, documentavam-se requisitos usando fluxogramas; até o final dos anos noventa, alguns ainda chamavam o documento de requisito de especificação funcional. Depois, veio a linguagem *UML*[®] - *Unified Modeling Language* e, então, as metodologias baseadas nela apresentaram seus modelos de artefatos: UP – *Unified Process*, RUP – *Rational Unified Process*, para eles, a descrição de requisitos é feita pelos casos de uso. Temos, ainda, a versão de boas práticas para a documentação e desenvolvimento de projetos da *Microsoft* - *Microsoft Solutions Framework* (MSF).

Não se aflija! O ponto aqui é que todas oferecem modelos, e modelos são adaptáveis, portanto, você pode definir junto com sua equipe qual o padrão que utilizarão para a documentação dos seus requisitos.

O Documento de Requisitos deve: 1. estabelecer o escopo do software - conjunto de funcionalidades que ele deverá oferecer - e também deve especificar os atributos de qualidade suportados – conjunto de características técnicas - (fácil, não é? Percebeu que estamos falando dos requisitos funcionais e não funcionais); 2. ser elaborado de maneira precisa, completa, consistente e, como já dito, deve ser compreensível aos *stakeholders* - todos eles: este documento será lido por vários interessados no projeto, por exemplo, o cliente, o gerente de projeto, o engenheiro de testes, os programadores etc.; 3. servir de referência para testes, manutenção e evolução do sistema. Percebemos que alguns itens são imprescindíveis:

- Definição dos requisitos de usuário.
- *Stakeholders*.
- Especificação de requisitos de sistema.
- Arquitetura do sistema.

Com base nessas premissas, definiremos um padrão para o registro dos requisitos para o nosso curso. Para isso, utilizaremos as sugestões do Guia PMBOK[®] e as recomendações IEEE de documentação padrão: IEEE-Std 830-1998. A Figura 7 traz um *template* desse documento.

TEMPLATE DOCUMENTO DE REQUISITOS

DOCUMENTO DE REQUISITOS

Histórico de Revisões

DATA	VERSÃO	DESCRIÇÃO	RESPONSÁVEL
01/01/2015	1.0	Elaboração do documento	Vanessa Chain
...
28/04/2015	3.1	Modificação na seção Descrição da interface com o usuário e nos casos de uso do sistema.	Rogério Matos

1. OBJETIVOS

Descrição dos objetivos do documento e o público ao qual ele se destina.

2. ESCOPO GERAL DO PRODUTO

Definir, em linhas gerais, o propósito e escopo do produto a ser desenvolvido.

3. CONVENÇÕES, TERMOS E ABREVIações

Relação dos termos e abreviações usadas, tipos de prioridades atribuídas aos requisitos, além de informar como o documento deve evoluir.

3.1 Identificação dos requisitos e casos de uso

3.2 Prioridades dos requisitos

4. REQUISITOS

Os requisitos podem ser registrados utilizando várias técnicas, de forma individual ou agregada, para facilitar a compreensão da rotina a ser implementada. Neste documento, sugiro uma planilha para o registro, com a opção de utilização da representação gráfica por meio de casos de uso. A coluna UC identifica o caso de uso referente àquele requisito.

4.1 Requisitos funcionais

Esta seção descreve, de maneira sumarizada, as funcionalidades do software. O engenheiro de requisitos deve organizar o conjunto de funcionalidades, de modo a torná-las compreensíveis a todos os stakeholders envolvidos.

RF	UC	PR	Descrição/ Ação	Entrada	Saída	pré- condição	pós- condição	Stakeholder

4.2 Requisitos não funcionais

Esta seção considera os requisitos do produto, do processo, da interface gráfica e da plataforma tecnológica empregada e demais requisitos limitadores, isto é, requisitos de segurança, confiabilidade, timeout de sessão de usuário, usabilidade, e outros a serem identificados.

RF	UC	PR	Descrição/ Ação	Entrada	Saída	pré- condição	pós- condição	Stakeholder

5. ESCOPO NÃO CONTEMPLADO

Contém descrição das funcionalidades que não serão desenvolvidas. Outra denominação dada a esta seção é escopo negativo. Isto garante aos stakeholders o que não faz parte do conjunto a ser implementado.

6. APROVAÇÃO

Nesta sessão, registre o nome, função na empresa e/ou no projeto, os envolvidos, a data e a assinatura.

Figura 7: Template de Documento de Requisitos

Fonte: o autor.

Cabem, ainda, no documento de requisitos outras duas sessões, que são opcionais e podem ficar a critério do engenheiro de requisitos sua inclusão no documento formal:

- Documentos Complementares, em que você pode relacionar atas de reuniões nas quais ocorrerão levantamento e validação de requisitos, o plano de projeto, ou, ainda, a matriz de rastreabilidade dos requisitos.
- Apêndice, em que você poderá acrescentar documentos que não fazem parte do documento de requisitos, mas podem colaborar com informações de apoio para os leitores; documentos, por exemplo, de levantamento de perfil dos usuários do sistema a ser desenvolvido, descrição do problema a ser automatizado pelo sistema de software, descrição da plataforma sobre a qual o sistema será desenvolvido etc.

MATRIZ DE RASTREABILIDADE DE REQUISITOS

Perfeito! Sabemos onde registrar nossos requisitos. Agora, falaremos um pouco sobre como ligar os requisitos as suas origens e os rastrear durante todo o ciclo de vida do projeto.

Uma das maiores dificuldades para a engenharia de requisitos é o controle e o registro de novos requisitos impostos ao projeto durante o seu desenvolvimento que podem ser gerados por diversos motivos, por exemplo: demanda de programação, por solicitação de *stakeholders*, por mudança de contexto, ou, ainda, correção de erros detectados.

A Gerência de Requisitos é o processo da Engenharia de Requisitos que trata esse problema por meio de um processo com três fases: Identificação, Gestão de Mudanças e Rastreabilidade, e tem como principais objetivos: o gerenciamento das mudanças, o gerenciamento entre requisitos relacionados e o gerenciamento das dependências entre a documentação de requisitos e outros documentos originados durante outros processos da engenharia de software.

Na unidade IV, trataremos a gestão de requisitos com mais propriedade e detalhes, para o momento, o que importa é como elaborar o documento para registrar as mudanças e permitir o rastreamento dos requisitos do início ao fim do projeto.

De acordo com o guia Guia PMBOK®, o uso da matriz de rastreabilidade tem as seguintes funções: a) garante que cada requisito adiciona valor de negócio por meio da sua ligação aos objetivos funcionais (da regra de negócio) e aos objetivos do projeto de desenvolvimento; b) fornece um meio de rastreamento do início ao fim do ciclo de vida do projeto; c) garante que os requisitos validados no documento de requisitos sejam entregues no final do projeto; e d) fornece uma estrutura de gerenciamento das mudanças do escopo do produto.

A matriz de rastreabilidade é uma tabela que relaciona os atributos associados a cada requisito e os liga a sua origem. Os atributos que devem ser registrados, no mínimo, são:

- Identificador de requisito.
- Descrição do requisito conforme documento de requisitos.
- Solicitante.
- Prioridade.
- Situação (aprovado, em desenvolvimento, entregue, entre outros).
- Critério de aceitação.

O modelo sugerido para o uso durante nosso curso foi elaborado com base no proposto pelo Guia PMBOK® para matriz de rastreabilidade de requisitos.

TEMPLATE DA MATRIZ DE RASTREABILIDADE DE REQUISITOS

MATRIZ DE RASTREABILIDADE DE REQUISITOS

Histórico de Revisões			
Data	Versão	Descrição	Responsável
01/01/2015	1.0	Elaboração do documento	Vanessa Chain
...
09/04/2015	3.1	Inserido novo requisito (RF152-UC89)	Rafaela Rodrigues
Legenda			
Prioridade	Tipo do Requisito	Status	Complexidade
Altíssima	Funcional	Validado	Alta
Alta	Técnico	Em desenvolvimento	Média
Média	Qualidade	Teste	Baixa
Baixa		Qualidade	
Muito Baixa		Concluído	

Id Req.	Prioridade	Descrição	Complexidade	Critérios Mínimos para Aceitação	Solicitante	Data Validação	Status
Inserir identificação do Requisito registrado no Documento de requisitos.	Estabelecer a prioridade de desenvolvimento para os requisitos, considerando as abstrações e dependências.	Inserir descrição do Requisito conforme registrado no Documento de requisitos.	Estabelecer o nível de complexidade para o desenvolvimento (programação da função estabelecida pelo requisito).	Critérios/exigências externas que devem estar concluídos para o aceite do requisito. Por exemplo: Emissão de NF Eletrônica. Critério mínimo: conjunto de Schemas XML com as definições das mensagens e regras de validação dos Web. Services (WS) da NF-e disponível.	Registrar o stakeholder solicitante. Por exemplo: Usuário; Gerente Projeto; Qualidade.	Data da validação do requisito, ou da assinatura do documento de aceite de inclusão de requisito, ou do documento de aceite de inclusão de re-quisito (Registro de Solicitação de Mudanças).	Status.
...							

Figura 8: Matriz de Rastreabilidade de Requisitos
Fonte: o autor.



SAIBA MAIS

A Devmedia é um site de conteúdo para área de desenvolvimento de software. Conta com vários especialistas focados na melhoria dos processos de desenvolvimento de software e disponibiliza artigos, cursos, links e opiniões sobre as vertentes da área.

O link a seguir oferece um curso que objetiva introduzir os conceitos-base da Engenharia de Requisitos.

Acesse e participe: <<http://www.devmedia.com.br/curso/introducao-a-engenharia-de-requisitos/132>>. Acesso em: 22 jun. 2015

Fonte: o autor.

CONSIDERAÇÕES FINAIS

Nesta unidade, analisamos a importância da engenharia de requisitos dentro do processo de desenvolvimento de software. Percebemos que um documento de requisito pode ser considerado o motivo do sucesso ou do fracasso do projeto, uma vez que é nele que estão contidas as especificações operacionais do software a ser desenvolvido.

Aprendemos que o propósito da engenharia de requisitos está relacionado com a identificação de metas a serem atingidas: qualidade de software; produtividade no desenvolvimento, operação e manutenção de software; permitir que profissionais tenham controle sobre o desenvolvimento de software dentro de custos, prazos e níveis de qualidade desejados.

Estudamos ainda que, enquanto estiver envolvido somente na engenharia de requisitos, você não terá que se preocupar com o gerenciamento e controle de todos os *stakeholders* envolvidos no projeto, mas é de extrema importância que execute o levantamento, no mínimo, daqueles que podem influenciar na delimitação e manutenção dos requisitos que definirão o escopo final do produto.

Nesta unidade, desenvolvemos um modelo de Documento de Requisitos e entendemos que esse documento é responsável por delimitar o escopo do conjunto de funcionalidades que um sistema deve prover, bem como descrever os

atributos de qualidade que devem ser suportados pelo sistema a ser desenvolvido. Montamos, também, uma Matriz de Rastreabilidade de Requisitos que nos apoiará na gerência de requisito durante todo o ciclo de vida do projeto.

Percebemos que a engenharia de requisitos se relaciona com praticamente todas as áreas do processo de desenvolvimento de software: especificação, projeto, validação, testes e implementação e, como primeiro passo no processo de desenvolvimento de software, tem um impacto significativo sobre o sucesso do projeto. Ela delimita o escopo do projeto e estabelece uma base comum de comunicação para todas as disciplinas envolvidas no projeto de desenvolvimento.

ATIVIDADES



1. São os requisitos que expressam as condições ou especialidades que o software deve atender. São as restrições a serem impostas relacionadas a desempenho, segurança, usabilidade, portabilidade, facilidade de uso, eficiência etc. Com base nessa afirmação, **identifique a alternativa que apresenta a descrição de um requisito não-funcional:**

- a. O software deve calcular os salários dos diaristas e mensalistas.
- b. Os relatórios onde constam os salários dos funcionários devem ser emitidos quinzenalmente para considerar os adiantamentos e vales que recebem.
- c. O tempo de resposta das consultas não deve superar quinze segundos.
- d. O software deve apresentar uma tela informando ao usuário que seu acesso não é permitido.

2. Afirmativa 1: conjunto de atividades sistemáticas como condição para se alcançar determinado fim, o que se foi requisitado.

Afirmativa 2: conjunto estruturado de atividades para extrair, validar e manter um documento de requisitos.

Sobre os requisitos de software, em relação às afirmações acima:

- a. Somente a afirmativa 1 é verdadeira.
- b. Somente a afirmativa 2 é verdadeira.
- c. Ambas as afirmativas são falsas.
- d. Ambas as afirmativas são verdadeiras.

3. Dada a descrição abaixo, identifique a alternativa que relaciona os principais stakeholders:

“A pizzaria “Pizza a Pezzi” funciona com entrega de pizzas “in-loco”, ou seja, não há entrega em domicílio e o cliente deve vir buscar a pizza. É uma empresa familiar com atualmente 6 funcionários, além dos 2 proprietários. A pizzaria gostaria de melhorar sua organização e tem como objetivo principal maximizar sua produção e, portanto, a entrada bruta e o lucro líquido. Deseja-se também melhorar o atual serviço de pedido e entrega das pizzas. O horário de abertura da pizzaria ao público é das 18:00 às 22:00. Os pedidos podem ser feitos por telefone ou pessoalmente na pizzaria. Os pedidos feitos por telefone tem mais prioridade. As pizzas são enfiadas em grupos de no máximo nove pizzas e cada fornada precisa de 5 minutos para um cozimento apropriado das pizzas. O processo de preparação das pizzas consiste de seis fases: (1) o cliente faz o pedido; (2) a pizza (base, molho, cobertura) é preparada; (3) a pizza é enfiada; (4) se necessário são colocados ingredientes crus depois do cozimento; (5) a pizza é preparada para entrega; (6) a pizza é entregue ao cliente, que efetua o pagamento. A pizzaria abre de terça a domingo. Durante final de semana e ferias-

ATIVIDADES



dos, a carga de trabalho é maior do que nos dias úteis. A organização do trabalho é feita da seguinte maneira: os dois proprietários trabalham todos os dias; dos outros seis funcionários, cinco trabalham somente durante o final de semana, enquanto o sexto ajuda o proprietário durante os dias úteis e fica à disposição dos proprietários também para ajudar nos feriados e eventualmente substituir algum outro funcionário que precisou faltar. Existem três papéis principais na gestão da pizzeria: quem prepara a pizza, quem trabalha na cozinha preparando os ingredientes e quem gerencia os pedidos, entregas e pagamentos. A gestão dos recursos necessários para a pizzeria é efetuada inteiramente pelos dois proprietários. Os ingredientes que faltam (acabam) são comprados no supermercado. Outros recursos, como por exemplo as caixas para entrega das pizzas, são pedidos ou comprados em lojas especializadas.

Fonte: Souza (online).

- a. Os proprietários da pizzeria; os funcionários da pizzeria (o pizzaiolo, o cozinheiro e o atendente); os clientes da pizzeria.
 - b. O Proprietário do supermercado; os clientes da pizzeria; lojas especializadas.
 - c. O governo; os funcionários da pizzeria (o pizzaiolo, o cozinheiro e o atendente); o proprietário do supermercado.
 - d. Os proprietários da pizzeria; os funcionários da pizzeria (o pizzaiolo, o cozinheiro e o atendente); o proprietário do supermercado.
4. Os requisitos de usuário são escritos com alto nível de abstração, em linguagem natural, e podem conter diagramas simples e representativos das funcionalidades que o sistema deve apresentar e das restrições que ele deve operar. Assinale a alternativa que contém a quem se destinam as informações nesse nível de abstração:
- a. Engenheiro do cliente; arquiteto do sistema.
 - b. Desenvolvedor do software; gerentes de clientes.
 - c. Gerentes de clientes; usuário final do sistema.
 - d. Engenheiro do cliente; desenvolvedor do software.
5. Os requisitos não funcionais são classificados em requisitos de produto, requisitos organizacionais e requisitos externos, os requisitos de produto especificam o comportamento do produto e podem ser subdivididos em:
- a. Requisitos de usabilidade, de eficiência, de confiança e de proteção.
 - b. Ambientais, operacionais e de desenvolvimento.
 - c. Reguladores, éticos e legais.
 - d. Requisitos de usabilidade, reguladores, éticos e legais, ambientais.

ATIVIDADES



6. Assinale a alternativa correta, marcando com (V) a assertiva verdadeira e com (F) a assertiva falsa, sobre os tipos de requisitos de software:

() Requisitos de facilidade de uso: usuários deverão operar o sistema após um determinado tempo de treinamento.

() Requisitos de eficiência: o software deverá processar n requisições por x tempo.

() Requisitos de portabilidade: o software deve ser compatível com qualquer plataforma.

() Requisitos de domínio: um relatório de acompanhamento do desenvolvimento deverá ser fornecido toda segunda-feira.

Assinale a opção com a sequência CORRETA.

a. V, F, V, V.

b. F, V, V, F.

c. V, V, V, F

d. F, V, F, F.

7. A matriz de rastreabilidade é uma tabela que relaciona os atributos associados a cada requisito e os liga a sua origem. Assinale a alternativa que relaciona, de acordo com o guia Guia PMBOK®, o objetivo da utilização da matriz de rastreabilidade:

a. Garante que cada requisito adiciona valor de negócio por meio da sua ligação aos objetivos não funcionais (da regra de negócio).

b. Fornece um meio de rastreamento do início ao fim do ciclo de vida do projeto.

c. Garante que os requisitos não validados no documento de requisitos sejam tratados no final do projeto.

d. Fornece uma estrutura de planejamento do escopo do produto.



DOCUMENTO DE REQUISITOS - ESSENCIAL AO DESENVOLVIMENTO DE SOFTWARE

Um engenheiro de software é um profissional que deve ter a habilidade de antecipar e gerenciar mudanças de requisitos de um produto de software. Além disso, ele precisa saber se expressar e comunicar-se bem a fim de capturar e registrar adequadamente o documento de requisitos. Os principais problemas no desenvolvimento de um sistema de software decorrem do entendimento errado entre engenheiro de software (produtor), responsável em apresentar o documento de requisitos, e usuário (consumidor). Um documento de requisitos de software precisa ser claro, consistente e completo, porque esse documento servirá de referência aos desenvolvedores, gerentes de projeto, engenheiros de software (responsáveis pelos testes e manutenção do sistema), além de servir de base para definir o escopo das funcionalidades a serem registradas num contrato. Perceba que os requisitos compreendem o cerne de qualquer produto e mudanças sobre eles podem ocorrer ao longo do ciclo de vida de um software.

Desenvolver um sistema de software requer um processo, o qual informa um conjunto de atividades a serem realizadas, quem as executam, quais artefatos de entrada são necessários e quais artefatos de saída são produzidos. Nesse sentido, detectar erros ou quaisquer outros problemas como, por

exemplo, inconsistência e falta de clareza é de suma importância de modo a tornar o processo mais efetivo sob o ponto de vista de custo. Adicionalmente, envolver o usuário no processo é determinante para o sucesso do produto e do processo. Dentro deste contexto, entender adequadamente o requisito é essencial e essa é tarefa do engenheiro de software. Um requisito compreende uma característica ou funcionalidade que o sistema deve possuir ou uma restrição que deve satisfazer para atender uma necessidade do usuário. Dessa forma, o engenheiro de software, desempenhando o papel de engenheiro de requisitos, deve executar duas atividades essenciais para a elaboração do documento de requisitos: Elicitação de requisitos – atividade na qual os requisitos do sistema a ser desenvolvido são levantados; Análise de requisitos – atividade na qual os requisitos são analisados e confirmados pelos principais interessados do projeto (isto é, os stakeholders) que incluem cliente, usuário final e gerente de projetos, dentre outros.

Considera-se ainda que a elicitação de requisitos objetiva definir características do sistema sob a perspectiva do cliente, enquanto que a análise de requisitos visa obter a especificação de requisitos, do ponto de vista técnico, conforme entendimento dos desenvolvedores.





Durante a realização destas atividades, o engenheiro de software está preocupado em levantar, entender, analisar e, por fim, documentar os requisitos. Para tanto, ele deve concentrar-se nas características do sistema e atributos de qualidade, e não em como obtê-los. Aqui, é preciso identificar quais requisitos fazem parte ou não do escopo do sistema a ser desenvolvido, ou em outras palavras, entender a interface do sistema considerado e o ambiente externo.

Fonte: Filho (online).

É importante ressaltar a necessidade de definir o 'limite', ou também denominado escopo do sistema, a fim de tratar os requisitos funcionais e não funcionais do sistema. Além disso, quando da elaboração do documento de requisitos, o engenheiro de software deve levar em consideração os diferentes pontos de vistas dos stakeholders de modo que o documento resultante possa comunicar adequadamente o conjunto de requisitos do sistema a ser construído.



NA WEB

O vídeo a seguir apresenta, de forma simples, todas as áreas de conhecimento do Guia PMBOK® e como se integram durante o gerenciamento de um projeto.

Disponível em: <<http://www.ricardo-vargas.com/pt/pmbok5-processes-flow/>>. Acesso em: 01 jul. 2015.

PROCESSOS DA ENGENHARIA DE REQUISITOS



Objetivos de Aprendizagem

- Conhecer os processos da engenharia de requisitos.

Plano de Estudo

A seguir, apresentam-se os tópicos que você estudará nesta unidade:

- Processos da engenharia de requisitos
- Levantamento e análise de requisitos
- Técnicas para coletar requisitos
 - Entrevistas
 - Brainstorming
 - Técnica de grupo nominal (TGN)
 - Oficinas
 - Questionários
 - Pesquisas
 - Etnografia
- Negociação
- Validação de requisitos
- Especificação de requisitos

INTRODUÇÃO

Muito bem! Já entendemos onde a engenharia de requisitos se encaixa nas atividades da engenharia de software, já observamos os principais fundamentos, estudamos o que é requisito e sua classificação e, ainda, modelamos os documentos que deverão ser o produto final da engenharia de requisitos e a base para todo o projeto de software de gerenciamento de requisitos. Agora, vamos conhecer os processos e atividades que são responsáveis por garantir a especificação de um bom requisito.

Começo com uma pergunta: qual é o objetivo da engenharia de requisitos? A engenharia de requisitos tem o objetivo de fornecer a todos os envolvidos no projeto de desenvolvimento de software um artefato escrito que promova o entendimento do que se deseja produzir. Para ter certeza de que esse artefato reproduz, de fato, o produto final desejado, ele deve ser revisado e validado com o cliente, usuário, patrocinador, enfim, com todos os *stakeholders* que foram identificados no projeto e durante todo o processo de desenvolvimento. Nesta unidade, estudaremos o conjunto de atividades que nos auxiliam na construção do documento que será o ponto-chave, o produto final.

O primeiro passo, para o sucesso do projeto, é o levantamento de requisitos, ele deve ter características que o torne conciso, claro e completo em sua unidade. Para que isso ocorra, várias técnicas de elicitação podem ser utilizadas. Nesta unidade, serão apresentadas algumas, as mais populares e de aplicação prática no contexto do desenvolvimento de software.

Durante o processo de levantamento e mesmo depois de decidido todo o conjunto de requisitos, necessários para a composição do sistema que atenderá à necessidade do cliente, é importante analisar o que foi especificado e, então, corrigir, caso não se tenha escrito exatamente o que era esperado. Esse processo da engenharia de requisitos é denominado análise dos requisitos. Na sequência, vem a etapa da negociação de requisitos com *stakeholders*.

Perceberemos que os processos de levantamento, análise e negociação devem andar juntos e permanecer durante todo o ciclo de vida do projeto, o que irá dar garantias da elaboração de um bom Documento de Requisitos.

PROCESSOS DA ENGENHARIA DE REQUISITOS

Processos da engenharia de requisitos podem variar muito em função das características dos projetos e, até mesmo, das organizações. Pressman (2010) organiza as atividades de engenharia de requisitos montando o seguinte processo: Levantamento, Elaboração, Negociação, Especificação, Validação e Gestão. Sommerville (2008) nos apresenta: Levantamento, Análise, Documentação, Verificação/Validação e Gerência, e ajusta em sua publicação de 2011, organizando as atividades da seguinte forma: Estudo de Viabilidade, Levantamento e Análise, Documentação, Validação e Gestão.

Também é possível encontrar, na literatura, as atividades organizadas como elicitación, análise, negociação, documentação, verificação e validação. Analisando as propostas, podemos perceber que, apesar de estarem organizadas ou, até mesmo, nomeadas de formas diferentes, elas apresentam em comum a necessidade do levantamento e do registro dos requisitos.

Fato é, como afirma Márcio Andrade Silva em seu artigo: *A importância do levantamento de requisitos no sucesso dos projetos de software*, que toda metodologia de desenvolvimento de software propõe uma série de fases e atividades dentro do seu ciclo de vida, porém, independente do nome dado a cada fase, é extremamente recomendável que o processo contemple, no mínimo, dois grupos de atividades:

Especificação de requisitos: que representa todas as atividades realizadas para identificar, analisar, especificar e definir as necessidades que o sistema deverá prover para solução do problema levantado.

Gestão de requisitos: atividades responsáveis pela documentação, versionamento, controle de mudanças e qualidade dos requisitos levantados na fase de especificação de requisitos.

Para o nosso estudo, organizei uma compilação dos principais autores e suas representações genéricas e mesclei com as propostas práticas apresentadas no mercado atual. Assim, seguiremos o seguinte processo: Levantamento e Análise, Negociação, Documentação, Validação e Gerenciamento de requisitos.

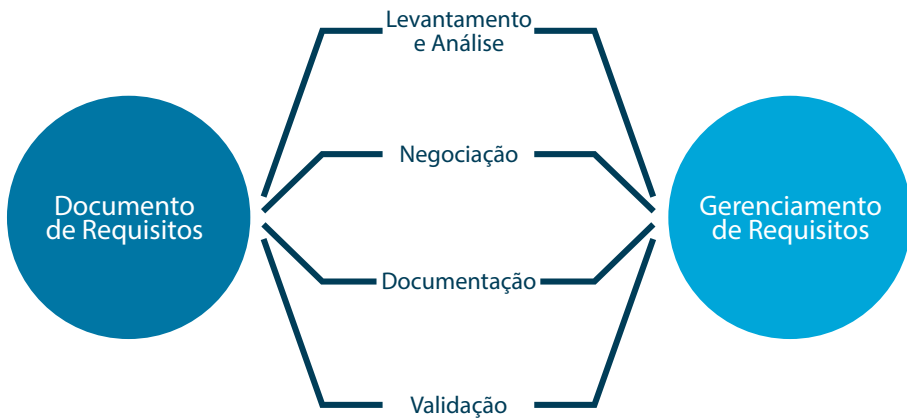


Figura 9: Processo da Engenharia de Requisitos
Fonte: o autor.

LEVANTAMENTO E ANÁLISE DE REQUISITOS

Esta etapa inicia o processo de engenharia de requisitos. Nesse ponto, você já deve ter identificado ou já foi disponibilizado para você a relação dos *stakeholders*, considerando seu grau de interesse, poder e influência no seu produto, bem como aqueles que conhecem a regra de negócio e sejam capazes de auxiliá-lo nas atividades de levantamento de requisitos. Essa fase também é conhecida como Elicitação de requisitos, elicitacão quer dizer ato ou efeito de eliciar; confrontar; aliciar; conseguir obter resposta ou informação (PRIBERAM, online).

Summerville (2008) considera que a definição do escopo geral deve ser feita em uma fase anterior à de Levantamento e Análise que ele intitula Estudo de Viabilidade, e Pressman (2010) concorda com ele, adicionando uma fase anterior no seu processo, a qual nomeia Concepção. Ambos afirmam que, nessa fase, o engenheiro de requisitos irá reunir-se com os clientes e usuários – patrocinadores e usuários finais do sistema – para obter a noção geral sobre as funcionalidades do sistema a ser desenvolvido ou mantido e, a partir dessa noção, define-se o escopo geral e se existe viabilidade para o desenvolvimento do software.

Em uma abordagem mais prática, considero que as etapas de concepção ou estudo de viabilidade foram desenvolvidas em um momento anterior do projeto, em que as partes se reúnem para decidir pelo desenvolvimento e contratação dos serviços, e que a fase de engenharia de requisitos já está dentro do projeto de desenvolvimento de software acordado, então, nesse momento – do Levantamento – o engenheiro de requisito estará com a responsabilidade de definir o escopo e os requisitos para o desenvolvimento do software, podendo, assim, aproveitar a reunião tanto para o levantamento do escopo quanto para levantamento dos requisitos.

O levantamento de requisitos é o início da atividade de desenvolvimento de software, é o ir a campo e compreender o que o cliente deseja e como será desenhada sua proposta para atender às expectativas dele. Sommerville (2008) propõe um processo genérico de levantamento e análise que contém as seguintes atividades:

1. **Compreensão do domínio:** os engenheiros de requisitos devem desenvolver sua compreensão do domínio da aplicação para que a transcrição seja eficiente e clara para os demais profissionais envolvidos no processo de desenvolvimento de software, principalmente em equipes em que engenheiro de requisitos não é necessariamente o programador. A compreensão do domínio irá se desenvolver mais conforme as atividades a seguir forem sendo desenvolvidas.
2. **Coleta de requisitos:** é o processo de interagir com os *stakeholders* do sistema para descobrir seus requisitos.
3. **Classificação:** organização dos requisitos em grupos coerentes.
4. **Resolução de conflitos:** como existem muitos *stakeholders* diferentes envolvidos, os requisitos serão apresentados a partir de pontos de vista diferentes. Os requisitos podem conflitar uns com os outros, nessa fase, o engenheiro de requisitos irá categorizar todas as informações para, posteriormente, decidir pelo conjunto de requisitos mais consistente.
5. **Definição das prioridades:** com o conjunto de requisitos em mãos, juntamente com os *stakeholders* tomadores de decisão, são definidos os requisitos mais importantes.

- 6. Verificação de requisitos:** os requisitos são verificados para descobrir se estão completos e consistentes e se estão em concordância com o que os *stakeholders* desejam do sistema.

A obtenção de requisitos não é um processo formal, por isso não permite sua automatização. O engenheiro de requisitos deve contar com técnicas de entrevista, questionário, mapeamento, entendimento do processo-alvo etc.

TÉCNICAS PARA COLETAR REQUISITOS

O objetivo é reunir-se com os *stakeholders* e conquistá-los, trazê-los para o projeto e fazê-los entender que são coautores, e não meros participantes ou telespectadores. As características do projeto e do cliente determinarão a técnica mais adequada para a coleta de requisitos.

Fornier (1994) apresenta alguns procedimentos a serem estabelecidos durante o levantamento de requisitos que independem da técnica adotada para a coleta deles:

Antes:

- I. Identificar as áreas envolvidas, explicar a finalidade e obter aprovação das gerências apropriadas para o trabalho.
- II. Obter nome e função dos *stakeholders* chaves e que participarão da coleta. Confirmar e solicitar concordância quanto a papéis, responsabilidades e disponibilidades previstas.

Durante:

1. Familiarizar-se com o local de trabalho que está sendo estudado.
2. Coletar amostras de documentos e procedimentos escritos.
3. Acumular informações estatísticas a respeito das tarefas: frequência com que ocorrem, estimativas de volume, tempo de duração para cada tarefa e assim por diante.
4. Enquanto interage com as partes interessadas, tentar ser objetivo e não comentar as formas de trabalho de maneira não construtiva.

5. Além das operações normais de negócio, identificar, também, as exceções.
6. Tão logo o levantamento tenha sido completado, agradecer às pessoas pelo apoio.

Após:

1. Documentar os requisitos e consolidar os resultados.
2. Caso seja necessário, contatar os próprios informantes para esclarecer dúvidas.
3. Rever os resultados consolidados com os próprios informantes e/ou superiores.
4. Atribuir uma prioridade para cada requisito identificado.



SAIBA MAIS

A coleta de requisitos é um processo meticuloso que demanda recursos e tempo. A EMC desenvolveu uma nova abordagem para a coleta de requisitos que foca nos dados. Desenvolveram um mapa de percurso para a governança e qualidade de dados empresariais que utiliza uma abordagem colaborativa e consistente para definir a qualidade de dados.

Fonte: Karel (online).

Existem inúmeras técnicas para interagir com os *stakeholders* a fim de coletar dados e informações. Novamente, recomendo que decida qual metodologia utilizar em acordo com a necessidade do seu projeto. Isso significa que deve conhecer sua organização, seus colaboradores e seu cliente.

A seguir, analisaremos algumas técnicas para coletar requisitos, que selecionei tomando por base tanto a teoria quanto a minha experiência prática: entrevistas, *brainstorming*, técnica de grupo nominal (TGN), técnica Delphi, oficinas, questionários, pesquisas e etnografia.

ENTREVISTAS

Sotille et al (2014) afirmam que a entrevista é o método mais utilizado para coletar requisitos. As entrevistas acontecem em reuniões para sanar dúvidas, levantar problemas e buscar soluções. Essa técnica, normalmente, é utilizada quando a equipe de desenvolvimento não conhece os problemas do cliente ou da organização. É necessário que se faça uma preparação prévia para a realização dela.

Antes da entrevista:

- Algumas questões devem ser consideradas: o que se deseja saber? O que se aceita saber? (Mínimo necessário). Quem será o entrevistado? Quais perguntas serão feitas? Qual é a sequência de execução das perguntas? Quando finalizar?
- Divulgue com antecedência a agenda e pauta da entrevista (reunião). Dessa forma, o entrevistado pode se preparar e buscar informações/documentos necessários.
- Agendar a entrevista/reunião em um horário conveniente para o entrevistado e, sempre que possível, em local sossegado, que elimine distrações.

Durante a entrevista:

- Ter em mente que pode haver resistência às mudanças e que seu trabalho pode ser considerado uma ameaça pelo entrevistado.
- Fazer perguntas.
- Ouvir e responder de forma cordial às perguntas do entrevistado.
- Tomar notas.
- Não ser tendencioso.
- Determinar quem ficará responsável pelas atividades a serem realizadas e o respectivo prazo de conclusão.

Quanto ao tipo de perguntas, para elaborar o questionário, você pode compor com duas categorias gerais: abertas (subjetivas) e objetivas. As perguntas abertas permitem que o interlocutor dê uma resposta descritiva. Por exemplo: como o usuário se comporta nessa situação? As perguntas objetivas, por outro lado,

apenas requerem um sim ou não como resposta, por exemplo: o usuário geral terá acesso a esta rotina? Uma pergunta objetiva é indicada quando uma resposta precisa é necessária.

A estrutura da entrevista pode ser organizada das seguintes formas:

- Estrutura Pirâmide: começa com questões detalhadas (objetivas/finas) e termina com questões mais gerais (subjetivas/largas).
- Estrutura Funil: começa com questões mais gerais (subjetivas/largas) e termina com questões mais detalhadas (objetivas/finas).
- Estrutura de Diamante: agrega a estrutura funil mais a estrutura pirâmide. Isto é, começa com questões específicas, avança para questões mais gerais e finaliza com questões específicas. Geralmente, é a melhor forma de se entrevistar, mas tende a ser mais longa.
- Não-Estruturada: não há uma sequência de como as questões serão abordadas, porém as questões são definidas com antecedência.



REFLITA

A comunicação é um processo que envolve a troca de informações e utiliza os sistemas simbólicos como suporte para este fim. Por meio da internet, novos meios de comunicação foram criados, e-mail, chat, fóruns, comunidades virtuais, webcam, entre outros, revolucionaram os relacionamentos humanos. Você considera que qualquer rotina que exija interação e troca de informações hoje em dia é possível ser realizada pelos meios virtuais?

Fonte: o autor.

BRAINSTORMING

Brainstorming significa tempestade cerebral ou tempestade de ideias. É uma expressão inglesa formada pela junção das palavras “brain”, que significa cérebro, intelecto, e “storm”, que significa tempestade (SIGNIFICADOS, online).

O brainstorming é uma dinâmica de grupo que é usada para desenvolver novas ideias ou projetos, para juntar informação e para estimular o pensamento criativo. Foi criado pelo publicitário Alex Osborn para testar e explorar a capacidade criativa de indivíduos ou grupos. A técnica de brainstorming propõe que um grupo de pessoas se reúna para expor seus pensamentos e ideias sobre um determinado problema. É intencionalmente não limitador e projetado para deixar a mente criativa fluir livremente. O foco é na quantidade de ideias expostas.

- Os procedimentos para a implantação do *brainstorming* é composto pelas seguintes etapas: definição do tema (defina claramente o problema, saiba o que você quer de resultado antes de iniciar o *brainstorming*), escolha da equipe (recomenda-se até 12 participantes e o encorajamento para a participação de representantes de várias áreas diferentes), definir o tipo de brainstorming, geração de ideias em reunião (encoraje a participação voluntária e a geração entusiasmada e divertida de muitas ideias, anote todas as ideias em um flip-chart), seleção das ideias (elaboração de um relato final sobre o tema que pode, inclusive, incluir oportunidades de melhoria no processo de realização de um *brainstorming*). A Figura 10 demonstra as etapas para a implantação do *brainstorming*:



Figura 10: Etapas do brainstorming
Fonte: o autor.

Para uma sessão de brainstorming, devem ser seguidas algumas regras básicas:

- Garantir que todos compreendam o objetivo do brainstorming antes de iniciá-lo.
- Decidir que método de brainstorming irá utilizar antes de iniciar a reunião. Normalmente, começa de maneira estruturada e, então, muda para o não estruturado, quando a maioria dos integrantes não tem mais colaboração a fazer.
- É proibido debates e críticas às ideias apresentadas. O objetivo é deixar a criatividade fluir e alcançar o máximo possível de ideias, críticas causam inibições. A discussão deve começar depois que o brainstorming acabar.
- Nenhuma ideia deve ser desprezada. Deve ser dada liberdade total para que o grupo fale sobre o que quiser.
- Modificação ou combinação de ideias. Prosseguir com base na ideia de outros e tentar criar ideias mais livres, novas e desembaraçadas.
- Deve-se garantir igualdade de oportunidade. Todos os participantes devem ter chance de expor suas ideias.

Escolha um lugar para a reunião que seja do tamanho exato para acomodar a equipe e crie uma atmosfera relaxada, que promova segurança aos participantes. Existem várias formas de promover o brainstorming, pode ser estruturado ou não estruturado, pode, ainda, utilizar anotações anônimas para a divulgação de ideias, quando perceber que o grupo não é comunicativo, ou não se sente a vontade em expor suas opiniões abertamente.

Lembra o que Fornier (1994) recomenda? Para qualquer técnica utilizada, antes de aplicá-la, identifique os *stakeholders*, reconheça o ambiente. Observe as pessoas, pois esse processo poderá ser útil para resolver impasses que podem surgir durante o andamento do projeto.

No brainstorming estruturado, cada membro deve dar uma ideia sobre o tema escolhido a cada rodada. O número de rodadas pode ser definido em função do tempo disponível ou você pode deixar que o brainstorming prossiga

até que todos não tenham mais ideias para apresentar. Essa forma de condução tem a vantagem de evitar que os indivíduos mais falantes predominem sobre os mais tímidos e, também, colabora no impedimento da inibição, em que colaboradores de diferentes níveis hierárquicos da organização participam juntos.

No brainstorming não estruturado, qualquer participante lança ideias à medida que vão surgindo, não existe vez, como no brainstorming estruturado. A vantagem é que se cria um ambiente mais relaxado e se torna mais fácil para alguns integrantes aproveitar as ideias de outros para formular suas próprias. É menos metódico e mais dinâmico, ideal quando as pessoas do grupo são todas comunicativas e pouco tímidas, mas há o risco de os integrantes mais falantes dominarem o ambiente.

TÉCNICA DE GRUPO NOMINAL (TGN)

O brainstorming não utiliza, em sua estrutura, votação ou priorização de ideias, a técnica de grupo nominal adiciona ao processo de brainstorming uma etapa de votação para ordenar as melhores ideias e priorizá-las.

O site escritório de projetos¹ apresenta as seguintes etapas para a realização da técnica de grupo nominal:

- Gere ideias: cada participante escreve suas ideias em um papel.
- Recolha o material escrito por cada participante e registre as ideias: pode-se usar um flip chart ou um quadro negro.
- Reveja e discuta as ideias.
- Vote as ideias: cada participante prioriza as ideias que são ordenadas conforme votação.

¹ <<http://escritoriodeprojetos.com.br/>>.

TÉCNICA DELPHI

O método Delphi foi originalmente criado nos anos 50 do séc. XX. O método consiste em um inquérito efetuado a especialistas e realizado em dois ou mais ciclos. Como em cada ciclo os participantes têm acesso aos resultados do ciclo anterior, as suas respostas podem ser mantidas ou alteradas de forma a incorporar o conhecimento e as opiniões expressas pelos demais.

Esse modelo não reúne os participantes, é uma técnica não interativa e utilizada como meio de alcançar um consenso quando há dificuldade em reunir os participantes.

Sotille et al (2014) nos apresentam o seguinte processo que pode ser utilizado para a organização do levantamento de requisitos utilizando o método Delphi:

1. Designa-se um facilitador e escolhem-se os participantes.
2. De acordo com o problema, um questionário é elaborado e distribuído aos participantes. Apenas o facilitador tem acesso sobre os participantes e suas respostas.
3. O facilitador distribui as informações sobre o projeto e solicita aos participantes que respondam ao questionário e gerem uma lista de requisitos e individual e anonimamente o encaminhe para ele.
4. O facilitador consolida as diversas listas em uma única e redistribui para os participantes revisarem/complementarem. Esse é o momento em que cada participante considera o que foi proposto pelos demais e mantém, complementa ou acata outras ideias.
5. A nova lista é devolvida ao facilitador que, novamente, a consolida.

Esse processo pode se repetir quantas vezes forem necessárias, até que o facilitador resolva o consenso e decisão final. A Figura 11 representa o processo.

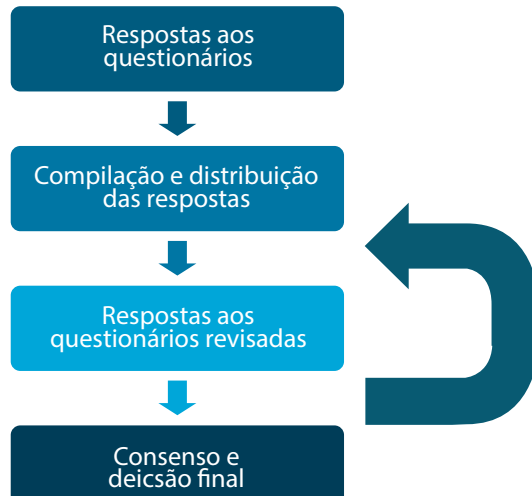


Figura 11: Processo método Delphi para coleta de requisitos
Fonte: o autor.

OFICINAS

As oficinas reúnem partes interessadas multifuncionais para definir os requisitos do produto a ser desenvolvido ou melhorado. As oficinas têm como objetivo acionar o trabalho em equipe. Deve existir sempre um facilitador cujo papel é conduzir a equipe e promover a discussão entre os vários *stakeholders*. As tomadas de decisão são baseadas em processos bem definidos e com o objetivo de obter um processo de negociação, mediado pelo facilitador. Algumas abordagens que podem ser utilizadas em oficinas são exemplificadas a seguir:

- QFD – *quality function deployment*. O método desdobramento da função de qualidade foi criado na década de 60 pelo japonês Yoji Akao e tem como objetivo principal permitir que a equipe de desenvolvimento do produto incorpore as reais necessidades do cliente. A primeira indústria a aplicá-lo foi a Mitsubishi Heavy em 1972. A Ford e a Xerox também utilizam esse método. O site Infoescola afirma que o QFD é uma ferramenta que possibilita “ouvir” a voz do cliente e ordená-la, de modo a facilitar a análise de suas necessidades

que são transformadas em requisitos. Em geral, o QFD obedece a quatro fases: planejamento do produto, desenvolvimento dos componentes, planejamento do processo e planejamento da produção. A técnica original desenvolvida por Akao, Figura 12, sofreu algumas adaptações ao longo do tempo, havendo, portanto, algumas variações da matriz original.

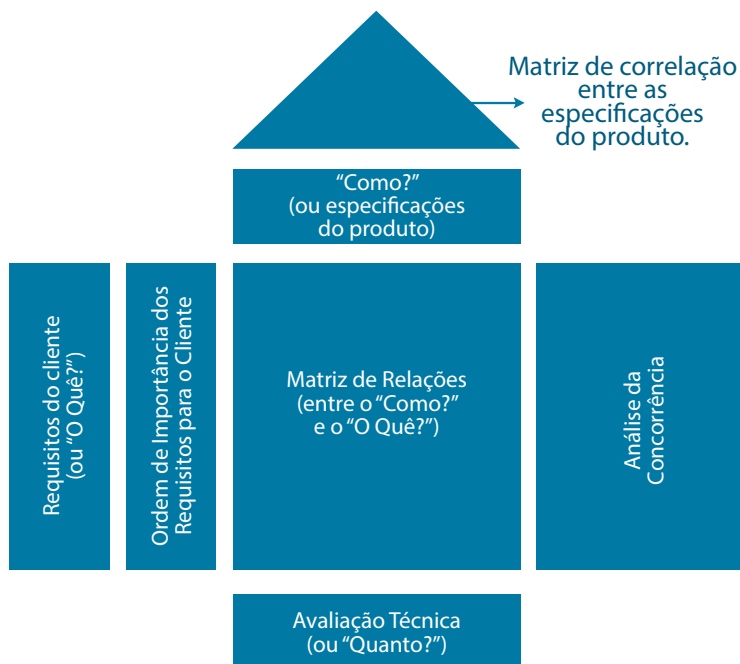


Figura 12: Matriz QFD original

Fonte: Faria (online).

- **JAD - joint application design.** É uma metodologia para definição de requisitos criada pela IBM do Canadá em 1977 e adaptada para o Brasil em 1982, por Hugo Gattoni. É uma oficina que busca promover que todos os participantes compartilhem uma mesma visão do produto. No JAD, todos os participantes são coautores da solução e devem deter o mesmo sentimento de responsabilidade sobre o seu desenvolvimento. A técnica JAD pode ser utilizada em vários momentos durante o ciclo de vida de um projeto: no planejamento, na definição dos requisitos e do escopo e na etapa de estimativas de esforço e horas para o desenvolvimento. A técnica JAD se divide em duas etapas, representadas pela Figura 13: Dados sobre o projeto e Resultados Revisados. Cada etapa possui três fases: (1) Adaptação - preparar o material que será utilizado durante as reuniões,

alocar e convidar os participantes selecionados para as reuniões, adaptar o processo JAD ao produto que será desenvolvido; (2) Sessão - são as reuniões propriamente ditas. Nessa fase, os requisitos são desenvolvidos e documentados; (3) Finalização - converter os requisitos extraídos em um documento de especificação de requisitos.

Os quatro princípios do método JAD são:

- Dinâmica de grupo - facilita o entendimento do problema e das necessidades das áreas envolvidas no projeto. Os participantes compartilham ideias e exploram a criatividade na criação da solução.
- Recursos visuais – usar técnicas visuais que facilitam a comunicação e o entendimento, por exemplo, utilizar na sala de reunião painéis contendo referências e conceitos do projeto, indicadores magnéticos, protótipos e transparências que servem para melhor comunicar e validar ideias durante a definição do projeto.
- Processo organizado e racional – o JAD emprega análise top-down com atividades bem definidas, a fim de atingir a definição de objetivos, especificações e projetos externos.
- Documentação padrão – cada sessão JAD possui um documento de saída padrão que tem como objetivo registrar formalmente os resultados do processo para que todos os participantes entendam as decisões tomadas. Geralmente, utiliza-se para esse documento a abordagem WYSIWYG (do inglês *what you see is what you get* ou, em português, o que você vê é o que você obtém).

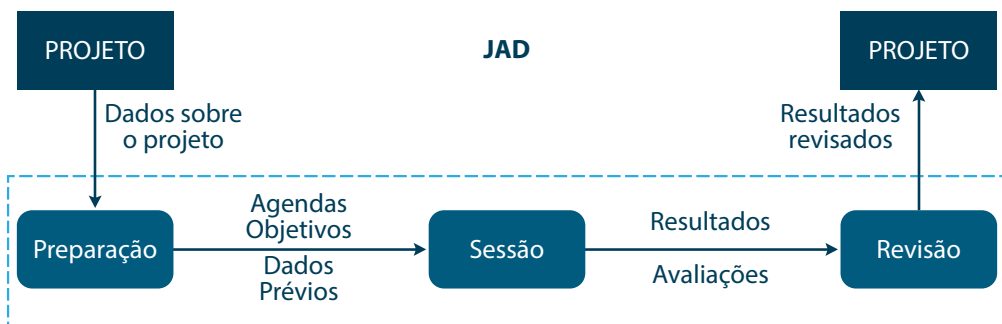
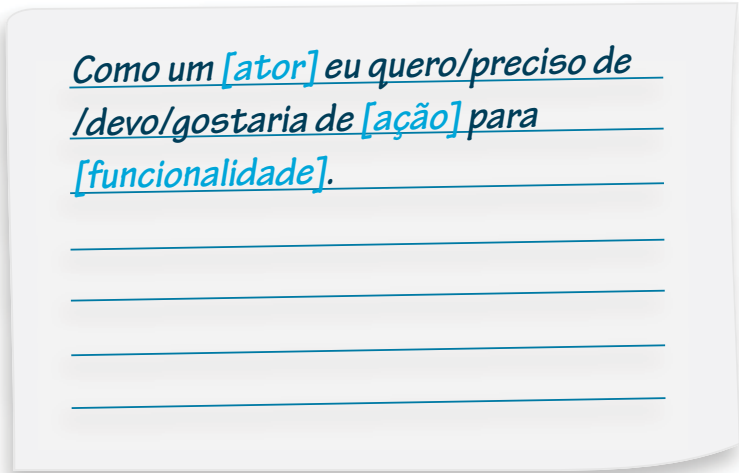


Figura 13: Visão geral do JAD
Fonte: Luiz David (2012).

- História de usuário (*user stories*) – são descrições textuais de uma funcionalidade requerida escrita do ponto de vista do usuário. As *user stories* possuem uma característica que deriva da metodologia ágil de desenvolvimento de sistemas que focam mais na comunicação do que na documentação. As metodologias ágeis fornecem uma abordagem iterativa e incremental para desenvolver e implementar projetos, com entregas contínuas, flexibilidade de escopo e participação da equipe. Na prática, se o seu projeto, por alguma razão, necessita de uma documentação formal que contemple todas as informações referentes aos requisitos, sozinha, as *user stories* não trarão o resultado esperado. O foco das *user stories* é a discussão em torno dos itens, e não na escrita dos cartões. São descrições curtas de uma funcionalidade de um sistema. Os requisitos aqui tratados como Histórias devem ser pensados sempre em nível de negócio, por isso, devem ser escritos sempre com o cliente presente. O cliente deve focar nos objetivos do negócio. “Por que você quer que seja realizada essa história?” – essa deve ser a pergunta da equipe ao tentar determinar o objetivo de uma história. Histórias, geralmente, são escritas como índices ou títulos para blocos de atividades, elas, usualmente, são colocadas em quadros para planejamento e discussão. É recomendado que cada história seja composta, no mínimo, por um ator, uma ação e uma funcionalidade. A Figura 14 ilustra a composição de uma *user story*.

A user story card template is shown as a light gray rectangular card with rounded corners and a subtle drop shadow. It features a series of horizontal blue lines for writing. The first line contains the text 'Como um [ator] eu quero/preciso de', the second line contains '/devo/gostaria de [ação] para', and the third line contains '[funcionalidade]'. Below these three lines are four more empty blue lines for additional text.

Como um [ator] eu quero/preciso de
/devo/gostaria de [ação] para
[funcionalidade].

Figura 14: Exemplo de user story card padrão
Fonte: Primo (online).

- a. Ator: é o proprietário da história de usuário. É o interessado naquela funcionalidade. É recomendado descrever de forma específica quem é o ator para ser mais fácil identificar o contexto da história dentro do sistema.
- b. Ação: representa o que o ator quer fazer. Realizando aquela ação, ele espera alcançar seu objetivo dentro do sistema.
- c. Funcionalidade: é o resultado esperado após a execução da ação pela perspectiva do ator. Também pode ser visto como justificativa. A Figura 15 apresenta um exemplo de *user story*.

Como um vendedor responsável pelo
setor de livros eu quero procurar por
livros filtrando por nome para que seja
possível verificar se o livro X está
disponível para pronta entrega.

Figura 15: Exemplo de user story card
 Fonte: Primo (online).

QUESTIONÁRIOS

O uso de questionário é indicado, por exemplo, quando há diversos grupos de usuários que podem estar em diversos locais diferentes. Nesse caso, elaboram-se pesquisas específicas direcionadas aos *stakeholders* identificados previamente.

Vários tipos de questionários podem ser utilizados: múltipla escolha, lista de verificação e questões abertas com espaços em branco para uma descrição mais informal. Lembre-se de que ninguém gosta de ficar respondendo questionários complexos e, o mais importante, ninguém quer perder tempo respondendo

perguntas, por isso, o questionário deve ser construído de forma a minimizar o tempo gasto em sua resposta.

Na fase de preparação do questionário, deve ser indicado o tipo de informação que se deseja obter. Assim que os requisitos forem definidos, o analista deve elaborar o questionário com questões de forma simples, clara e concisa, deixar espaço suficiente para as repostas que forem descritivas e agrupar as questões de tópicos específicos em um conjunto com um título especial. O questionário deve ser acompanhado por uma carta explicativa, redigida por um alto executivo, para enfatizar a importância dessa pesquisa para a organização.

Deve ser desenvolvido um controle que identifique todas as pessoas que receberão os questionários. A distribuição deve ocorrer junto com instruções detalhadas sobre como preenchê-lo e ser indicado claramente o prazo para devolução do questionário. Ao analisar as respostas dos participantes, é feita uma consolidação das informações fornecidas no questionário, documentando as principais descobertas e enviando uma cópia com essas informações para o participante, como forma de consideração pelo tempo dedicado à pesquisa.

Sotille et al (2014) recomendam alguns procedimentos para garantir uma aplicação eficiente do questionário:

Antes:

- Definir objetivos do questionário.
- Decidir o que é preciso saber para atingir os objetivos.
- Expor claramente os objetivos pretendidos e a importância da contribuição de cada *stakeholder*.
- Desenvolver perguntas representativas e objetivas. Evitar perguntas discursivas, mas permitir que os usuários deixem comentários gerais sobre o projeto no final do questionário.
- Definir método de tabulação para respostas.
- Validar o questionário em piloto com usuário de fácil acesso. Essa validação permite testar o questionário e o método de tabulação.

Durante:

- Distribuir o questionário com prazo suficiente para que o usuário responda sem pressão.
- Estar disponível para dirimir eventuais dúvidas.
- Monitorar o recebimento dos questionários.

Após:

- Tabular os dados.
- Analisar o resultado da tabulação.
- Remeter os dados tabulados para cada participante que respondeu ao questionário.

PESQUISAS

Como vimos, elicitar é uma expressão normalmente atribuída à atividade de descobrir, identificar, deduzir, extrair ou obter informações. Pesquisas em documentos organizacionais podem ser uma maneira de se conseguir identificar os requisitos necessários para determinar as funcionalidades de um novo produto de software. Exemplos de documentos que podem ser analisados incluem plano de negócio, registro de marketing, acordos, requisições de propostas, fluxo de negócio, modelos lógicos de dados, repositório de regras de negócio, documentação, processos de negócio, casos de uso, registro de questões públicas, políticas, procedimentos, leis, códigos etc.

ETNOGRAFIA

Etnografia é uma técnica de observação usada para compreender os requisitos sociais e organizacionais. O engenheiro de requisitos se insere no ambiente de trabalho onde o sistema será usado. Ele observa o trabalho do dia a dia e anota as tarefas reais nas quais os participantes estão envolvidos. A Wikipedia afirma

que, por meio de uma observação direta das atividades realizadas durante um período de trabalho de um funcionário, é possível encontrar requisitos que não seriam observáveis usando técnicas convencionais. Nessa técnica, assume-se que o representante do cliente observado desempenha as suas funções corretamente.

Enfim, encontraremos diversas técnicas, como já comentei: métodos em que o foco é a conversação, por exemplo, grupo focal, mapas mentais, diagrama de afinidade, análise de decisão multicritérios; outras técnicas em que o foco é a observação, etnografia, protocolo de análise; ainda, temos os métodos analíticos, como: reuso de requisitos, estudo de documentação/análise de conteúdo, *laddering*, sorteio de cartões; também, existem as técnicas sintéticas, como: prototipação, questionário de ambiente, *storyboards*. Todas elas possuem vantagens e desvantagens a serem consideradas em relação à complexidade do ambiente de negócio onde seu projeto estará inserido. O importante é conhecê-las e saber seus pontos fortes e fracos para aplicá-las adequadamente em cada situação.

Uma vez levantados os requisitos, passamos para a atividade de análise. Nessa etapa, os requisitos levantados são usados como base para a modelagem do sistema. Conforme vimos anteriormente, os requisitos, quando levantados diretamente dos usuários, são escritos tipicamente em linguagem natural, pois eles são fornecidos e devem ser compreendidos por pessoas que não são especialistas técnicos. Na atividade de análise, os requisitos podem ser detalhados de maneira mais técnica, utilizando, para isso, diversos tipos de modelos que, por utilizarem representações gráficas, são, geralmente, mais compreensíveis do que descrições detalhadas em linguagem natural (SOMMERVILLE, 2008).

Em essência, a fase de análise é uma atividade de modelagem. A modelagem nessa fase, é dita conceitual, pois se preocupa com o domínio do problema, e não com soluções técnicas para ele. Esse é o momento em que a equipe de desenvolvimento irá interpretar as necessidades do cliente e desenhar o sistema. Esses modelos ajudam o analista a entender a informação, a função e o comportamento do sistema.

Parece confuso quando descritas linearmente todas as atividades do processo, na prática, as atividades da engenharia de requisitos ocorrem em paralelo. Como? Explico: durante o levantamento, uma análise e uma especificação já estarão sendo executadas de maneira indireta e alguns problemas são identificados e tratados,

entretanto outros somente serão identificados por meio de uma análise mais detalhada. A Figura 16 nos apresenta um modelo da espiral da interação entre levantamento e análise de requisitos, com o propósito de facilitar a compreensão.

O levantamento é o ato de identificar as necessidades do usuário, isto é, os requisitos do sistema. A análise trata da constante observação e dos elementos do ambiente onde o software será implantado, checando todas as atividades que serão envolvidas no sistema: quem faz, por que faz e se existem outras formas de ser feito, considerando os dados e informações que são gerados por essas atividades.

A especificação é a descrição sistemática e abstrata do que o sistema deve fazer a partir dos requisitos levantados. Ela apresenta a solução de como os problemas levantados na análise devem ser resolvidos pelo software em desenvolvimento, fechando o primeiro ciclo da engenharia de requisitos com a produção do Documento de Requisitos e da Matriz de Rastreabilidade de Requisitos. Naturalmente, os requisitos passarão por uma negociação e uma prévia validação tanto dos *stakeholders* do lado do cliente quanto da equipe de desenvolvimento. Esse é o processo de gerência de requisitos! A gerência de requisitos será estudada na unidade IV.

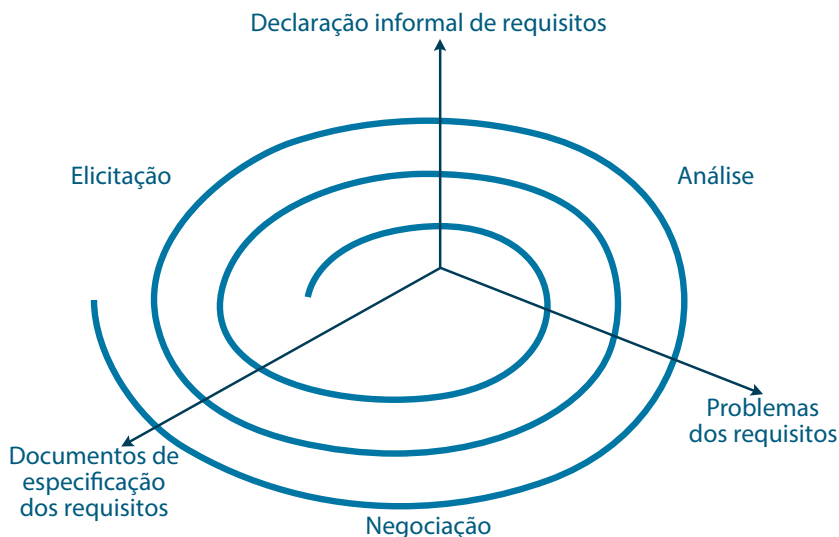


Figura 16: Modelo da espiral da interação entre levantamento e análise de requisitos
Fonte: Kotonya (1998).

A seguir, vamos estudar as demais atividades da engenharia de requisitos!

NEGOCIAÇÃO

Com a lista geral de requisitos em mãos, vamos, agora, para a fase de negociação e priorização dos requisitos junto aos *stakeholders* do lado do cliente.

Pressman (2010) afirma, e concordo com ele, que é muito comum que clientes e usuários peçam mais do que pode ser conseguido, e também é muito comum que diferentes clientes ou usuários proponham requisitos conflitantes e argumentem que a sua versão é a essencial para as necessidades do sistema.

Embora exista a máxima “o cliente tem sempre razão”, alguns pontos devem ser considerados e discutidos para garantir que o produto final atinja seu objetivo, as restrições de prazo, custos, recursos, quase sempre vêm de encontro com a realização dos desejos do cliente, nesse ponto, a equipe de desenvolvimento deve deixar as limitações do projeto claras e iniciar um processo de negociação, com o intuito de acordar entre as partes, quanto aos requisitos que serão considerados prioritários e quais participarão do escopo do projeto.

A priorização dos requisitos pode determinar tanto o sucesso do projeto como o seu fracasso e essas decisões não são simples ou binárias. Pressman (2010) sugere que clientes, usuários e demais interessados ordenem os requisitos e discutam os conflitos e prioridades até se chegar a um consenso geral e o escopo geral do produto ser definido. Sylvia (2008) sugere que, dentre os principais problemas, os mais críticos são os relacionados a: negócio, clientes e implementação.

De forma geral, o que se sugere é que os riscos associados a cada requisito sejam identificados e analisados para, então, elaborar estimativas de esforço de desenvolvimento, de forma a avaliar o impacto de cada requisito no custo do projeto e no prazo de entrega. Dessa forma, requisitos podem ser eliminados, combinados ou modificados para se alcançar o consenso entre cliente e equipe de desenvolvimento (PRESSMAN, 2010).

Existem várias técnicas para priorização dos requisitos, por exemplo:

- Atribuição numérica - propõe a atribuição de uma escala de notas para cada requisito. Os requisitos são agrupados em diferentes grupos, nos quais os *stakeholders* podem relacionar uma classificação confiável.
- Árvores de Busca Binárias (BSTs) - é um algoritmo comumente usado em busca de informação e que pode ser facilmente adaptado para priorizar uma grande quantidade de requisitos.

- *Analytic Hierarchy Process* (AHP) – é uma técnica de tomada de decisões em situações de múltiplos objetivos. A técnica provê um índice de consistência utilizado para verificar a fidelidade dos resultados. Esse índice é calculado de acordo com o número de requisitos e com os valores atribuídos em uma matriz. Por se tratar de uma técnica complexa e de difícil rastreabilidade dos resultados, a AHP não é bem aceita comercialmente.
- Abordagem Custo-Valor (*Cost-Value Approach*) - proposta por Karlsson (1997), ela prioriza os requisitos de acordo com seus custos e valores relativos. Essa técnica simplifica os resultados providos pela aplicação da técnica AHP, de forma clara e objetiva. Assim, ela une a precisão da AHP com resultados mais transparentes e visíveis aos *stakeholders*. Também, não é bem aceita comercialmente, pois exige a aplicação de AHP em dois momentos, tornando-se muito custosa na medida em que o número de requisitos aumenta.
- *Value-Oriented Prioritization* (VOP) - avalia e ordena requisitos de acordo com o impacto que eles terão em valores de negócios específicos da organização e, a partir daí, escolhem-se quais requisitos serão mais benéficos para a organização como um todo, e não para um *stakeholder* em particular. Para a priorização, são atribuídos pesos a cada área da organização de acordo com seu nível de importância, pode-se também atribuir pesos para os riscos dos requisitos e considerá-los no resultado final.

O levantamento, a análise e a negociação de requisitos estão intimamente ligados e fazem parte de um processo em espiral. Esse processo deve ser seguido até a satisfação de todas as partes envolvidas, sendo um processo influenciado não só pela lógica e argumentos técnicos, mas também pela política da organização e pela personalidade dos *stakeholders* (SOMMERVILLE; SAWYER, 1997).

REFLITA



A mudança não virá se esperarmos por outra pessoa ou outros tempos. Nós somos aqueles por quem estávamos esperando. Nós somos a mudança que procuramos.

Fonte: Barack Obama.

VALIDAÇÃO DOS REQUISITOS

Depois de levantados e negociados os requisitos, chegou o momento de revisar tudo quanto à ambiguidade, às omissões e às inconsistências. A validação dos requisitos, segundo Sommerville (2008), garante que a necessidade real do usuário esteja descrita corretamente no Documento de Requisitos. Isso está parecendo redundante para você? Você deve estar se perguntando: não foi exatamente isso que fizemos durante a negociação? Vamos esclarecer.

Análise e Negociação respondem à pergunta: foram elicitados os requisitos corretos? A Validação responde: os requisitos foram elicitados corretamente? Isto é, em um primeiro momento, você irá certificar se todos os requisitos que devem ser considerados foram realmente, eles garantem a completude do sistema que será desenvolvido, se não faltou nada a ser levantado. Na segunda, validação, você os corrige.

O objetivo da validação de requisitos é descobrir erros nos requisitos documentados. Como já estudamos, o Documento de Requisitos é referência para todas as demais atividades de desenvolvimento. Nesse sentido, a validação é extremamente importante, pois o custo para correção de um requisito nessa fase é bem inferior ao custo nas fases posteriores, implementação ou testes, por exemplo.

Nessa fase, se uma divergência for encontrada, levanta-se novamente o requisito, sem parar o desenvolvimento ou os testes, o que comprometeria custos e prazos do projeto. Carla Oliveira (2014) sustenta essa teoria afirmando que erros afetam negativamente todas as atividades posteriores de desenvolvimento. Um erro de requisito descoberto quando o sistema já está em produção ou em operação exige a revisão de todos os artefatos: código fonte, artefatos de testes e descrições de arquitetura, o que implica custos significativos.

Para tal atividade, Sommerville (2008) propõe algumas técnicas de validação, tais como revisões técnicas de requisitos, prototipação e geração de casos de teste.

A revisão técnica de requisitos pode ser formal ou informal. A revisão técnica formal é o principal mecanismo para validação de requisitos, trata-se de um processo no qual uma equipe formada por engenheiros de software, clientes, usuários ou outros interessados examinam as especificações, procurando por erro de registro ou de interpretação, por requisitos que precisem de mais esclarecimento, ou por aqueles em que faltam informações. Nessa reunião, devem também ser considerados os requisitos conflitantes e os intangíveis. A revisão de requisitos informal é um debate que ocorre entre equipe técnica e cliente com o objetivo de identificar problemas e propor soluções.

Na prototipação, os requisitos são apresentados de maneira mais lúdica, por exemplo, no formato de telas para o usuário, facilitando a interpretação das rotinas a serem desenvolvidas.

A técnica de validação geração de casos de testes verifica se o requisito é implementável. Isto significa: fácil de ser interpretado e codificado. Caso contrário, ele deve ser reconsiderado. Parece complicado testar um requisito antes da codificação, mas não é impossível, considerando a prática e experiência da equipe de desenvolvimento.

Por exemplo, nas metodologias ágeis de desenvolvimento de software – que estão dominando as fábricas de software atuais –, os requisitos são expressos em cenários, também conhecidos como *users stories* (histórias de usuários), conversamos um pouco sobre isso quando estudamos as técnicas de levantamento de requisitos. Partindo das informações adquiridas, a equipe de desenvolvimento estima o esforço e recursos necessários para a implementação e o cliente atribui uma prioridade para cada história, tendo como referência a importância da funcionalidade para o sistema.

Por fim, Oliveira (2014) propõe um *checklist* para a validação de requisitos. A técnica *checklist* deve ser usada durante a revisão técnica formal de requisitos e tem como objetivos:

- Descobrir erros de função, de lógica ou de implementação para qualquer representação do software.
- Verificar que o software em questão atende aos requisitos especificados.
- Garantir que o software foi representado de acordo com padrões pré-definidos.
- Garantir que o software seja desenvolvido de maneira uniforme.
- Desenvolver projetos mais gerenciáveis.

Para isso, podemos considerar a lista de verificações que Sommerville (2008) propõe:

1. Verificação de validade. Um sistema possui diversos usuários com necessidades diferentes, nessa validação, a equipe irá identificar se certas funções citadas como necessárias por um usuário são, realmente, indispensáveis para o todo.
2. Verificação de consistência. Identificar se existem requisitos que dizem a mesma coisa de uma forma diferente, se descrevem uma mesma função sob pontos de vista de usuários diferentes.
3. Verificação de completeza. O documento de requisito deve descrever todas as funções e todas as restrições exigidas pelos usuários do sistema.
4. Verificação de realismo. Validar se existe tecnologia para a implementação dos requisitos. Nessa validação, deve-se, também, considerar os custos e o prazo determinado para o desenvolvimento do sistema.
5. Facilidade de verificação. Observar se os requisitos estão escritos de forma compreensível para todas as partes envolvidas no processo de desenvolvimento do sistema, o que os torna verificáveis.

Oliveira (2014) afirma que o *checklist* auxilia o analista de requisitos ou o representante da equipe de desenvolvimento a coordenar a reunião de revisão formal de requisitos, colaborando para o foco dos revisores nas características importantes. Os *checklists* devem ser aplicados a documentos de análise, projeto, codificação e testes. Um exemplo de checklist é apresentado na Figura 17.

ITEM	ITEM PARA VERIFICAÇÃO	SIM	NÃO
Não ambíguo: É não ambíguo se, e somente se, cada requisito declarado seja suscetível a apenas uma interpretação.			
1	Cada requisito está descrito com clareza, concisão e sem ambiguidade?		
Consistente: É consistente se, e somente se, nenhum dos requisitos do documento, tomado individualmente, está em conflito com qualquer outro requisito do mesmo documento.			
2	Existem requisitos conflitantes?		
Completo: É completo se, e somente se, conter toda e apenas a informação necessária para que o software correspondente seja produzido.			
3	Existem requisitos implícitos?		
4	Os requisitos exibem a distinção clara entre funções, dados e restrições?		
5	As restrições e dependências foram claramente descritas?		
6	Existem requisitos que contêm algum nível desnecessário de detalhe do projeto?		
7	Os requisitos definem todas as informações a serem apresentadas aos usuários?		
8	Os requisitos descrevem as respostas do sistema ao usuário devido às condições de erro?		
9	Existem situações não tratadas pelos requisitos que precisam ser consideradas?		
10	O documento contém realmente toda a informação prometida em sua introdução?		

Figura 17: Modelo Checklist de Validação de Requisitos

Fonte: Oliveira (online).

Para facilitar o entendimento, acompanhe o processo abaixo, representado pela Figura 18, que demonstra o que seriam as possíveis entradas e saídas da etapa de validação.



Figura 18: Entradas e Saídas do Processo de Validação de Requisitos

Fonte: adaptada de Kotonya (1998).

A entrada 1 Especificação de Requisitos deve ser uma versão completa do Documento de Requisitos, organizada de acordo com os padrões da organização, contemplando todos os requisitos que comporão o software a ser

desenvolvido. O Conhecimento da Organização (entrada 2 no processo) não é tangível, mas, na prática, é muito importante ser considerado, uma vez que as pessoas envolvidas na validação de requisitos devem participar por deterem o conhecimento sobre as terminologias particulares, a execução prática das rotinas que serão automatizadas pelo novo sistema e, principalmente, o perfil dos usuários que ofereceram as informações durante o levantamento de requisitos e que irão utilizar o sistema depois de pronto. A entrada 3, Padrões Organizacionais, garante que a validação irá considerar que todos os padrões relevantes para o documento de requisitos devem ser entrada para o processo de validação.

Como saídas do processo, teremos: 1. Lista de Problemas, é recomendado que seja organizada por tipo de problema (inconsistentes, incompletos, intangíveis etc.) para facilitar as soluções e agilizar a reunião técnica de revisão; e 2. Lista de Ações, que representa a lista de atividades em resposta aos problemas listados, acordada com os envolvidos no processo.

No final da validação, temos praticamente o documento de requisitos pronto, que corresponde à entrega final da engenharia de requisitos. No próximo item, falaremos mais desse documento. Vamos lá?



REFLITA

Um projeto é uma verdadeira guerra de interesses. Descobrir, desde o começo, como gerenciá-los é a chave para o sucesso.

Fonte: *Stakeholder* (online).

ESPECIFICAÇÃO DE REQUISITOS

Quando conversamos sobre os conceitos fundamentais da engenharia de requisitos, falamos sobre o Documento de Requisitos, lá, tínhamos como foco a estrutura de apresentação do documento, agora, estudaremos um pouco mais sobre como escrever os requisitos para documentá-los.

Vimos, na unidade dois, que essa documentação descreve os requisitos que foram coletados e como eles permitirão atingir os objetivos do sistema a ser desenvolvido. Aprendemos, também, que, inicialmente, os requisitos podem ser descritos de um modo geral (alto nível) para que sejam detalhados posteriormente.

Primeiro, vamos considerar o público-alvo dos requisitos, a quem se destinam, quem serão os leitores e quais serão os propósitos da leitura:

- Clientes: leem para validar.
- Gerentes: utilizam para solicitar uma proposta e para planejar o processo de desenvolvimento.
- Programadores: utilizam para entender o que deve ser desenvolvido.
- Testes: utilizam para planejar os testes de validação do sistema.
- Manutenção: utiliza para compreender o sistema e suas relações.

Agora, considere os parâmetros sugeridos para o registro dos requisitos e os definidos para a sua validação. Releia o requisito e aplique um teste rápido para verificar se os critérios foram atingidos:

1. Necessidade. Depois de escrito, pergunte: qual será a pior coisa que poderá acontecer se esse requisito não for incluído? Se não encontrarmos qualquer problema, então, é porque, provavelmente, não precisamos do requisito (esta função é necessária?).
2. Verificável. Durante e após a escrita, leia o que está produzindo e observe se está entendível para todos os leitores, como estudamos acima. Esse aspecto é importante porque ajudará a assegurar que o requisito é verificável (como posso testar essa função?).
3. Realizável. Em algumas literaturas, poderá encontrar este parâmetro nomeado como atingível. Observe se o requisito registrado pode ser desenvolvido (programado) em relação à tecnologia disponível, orçamento e prazo. No caso de dúvida, investigue, converse, se ainda assim persistir a dúvida, descreva junto com o requisito o objetivo que deseja alcançar com a execução da rotina ali descrita, porque, ainda que ele seja exequível tecnicamente, poderá não ser atingível devido a outros critérios (é atingível?).

4. Claro: cada requisito deve expressar uma única ideia, geralmente frases simples especificam um bom requisito (está claro o que precisa ser feito?).

É frequente a utilização da linguagem natural para a descrição dos requisitos, principalmente quando a especificação está relacionando os requisitos de usuário. Para esse nível de especificação, Sommerville (2008) recomenda algumas regras básicas:

- a. Crie um formato padrão para sua escrita e garanta que todos os requisitos serão escritos seguindo esse padrão, assim, omissões se tornam menos frequentes e facilita a verificação dos requisitos.
- b. Utilize uma linguagem coerente, use variações de tempo de verbos para indicar o que é obrigatório ser feito e o que é desejável. Por exemplo, a utilização do verbo deve para requisitos obrigatórios, e sua variação deveria para aqueles requisitos que são desejáveis. Com isso estabelece-se uma regra ou padrão em que os requisitos obrigatórios são especificados com o verbo deve à frente, enquanto aqueles requisitos que, se não forem implementados, não causarão impacto no objetivo do produto final, serão especificados com o verbo deveria na sua descrição.
- d. Destaque as partes mais importantes da descrição do requisito.
- e. Não utilize jargões de informática ou da regra de negócio para a qual o requisito será desenvolvido, quando termos técnicos inerentes ao domínio da aplicação forem inevitáveis, procure detalhá-lo da forma mais completa possível. Por exemplo:
 - RF001 – O sistema deve validar os dados informados no campo destinado ao número do CPF.
 - RF002 – O sistema deve classificar a apresentação em tela da lista de clientes pelo campo nome.
 - RF003 – O sistema deveria classificar a apresentação em tela da lista de clientes pelo campo sobrenome.
 - RF004 – O sistema deve exibir a mensagem de alerta “ATENÇÃO! Quantidade em estoque insuficiente” quando o campo quantidade contiver valor mais alto que o campo quantidade de itens em estoque.

Os exemplos nos mostram que os requisitos funcionais 001, 002 e 004 são obrigatórios, enquanto o 003 é somente desejável. Procure especificar as ações em sua sequência e registre, sempre que possível, uma ação por requisito.

A descrição dos requisitos de sistema, em que o nível de especificação é mais técnico também pode ser feita utilizando linguagem natural, entretanto, quando a especificação exige mais detalhamento, podem surgir problemas.

A linguagem natural possui uma ambiguidade intrínseca, a interpretação de um conceito depende tanto do interlocutor quanto do receptor da mensagem. Ela, também, é muito flexível, é possível dizer a mesma coisa de muitas maneiras diferentes, além de não possibilitar uma padronização.

Sommerville (2008) sugere, para a especificação de requisitos de sistema, algumas linguagens padronizadas:

- Linguagem natural estruturada: os requisitos são escritos em linguagem natural em um formulário padrão ou template. Cada campo fornece informações sobre um aspecto do requisito.
- Linguagem de descrição de projeto: essa abordagem usa uma linguagem como de programação, mas com características mais abstratas para especificar os requisitos, definindo um modelo operacional do sistema. Essa abordagem é pouco usada atualmente, embora possa ser útil para as especificações de interface.
- Notações gráficas: para definição dos requisitos funcionais do sistema, são usados modelos gráficos, suplementados por anotações de texto; diagramas de caso de uso e sequência da UML®.
- Especificações matemáticas: essas notações são baseadas em conceitos matemáticos, como máquinas de estado finito ou conjuntos. Embora essas especificações possam reduzir a ambiguidade de um documento de requisitos, a maioria dos clientes não entende uma especificação formal. Por ser muito técnica, não é possível que o cliente verifique o que as informações representam e automaticamente ficam relutantes em aceitá-las.

Para nossos estudos, adotaremos a abordagem da Linguagem Natural Estruturada, que, segundo Sommerville (2008), mantém a facilidade de expressão da linguagem natural, mas garante um nível de padronização. Na utilização da linguagem natural estruturada, podem-se estabelecer terminologias e utilizar *template* padrão

para a especificação dos requisitos do sistema, possibilitando um agrupamento mais eficiente dos requisitos.

Essa abordagem elimina alguns problemas da linguagem natural, mas é necessário, ainda, um cuidado quanto à ambiguidade no momento da descrição do requisito. Deve-se criar um ou mais *templates* padrão em função do sistema que será desenvolvido e das necessidades, do detalhamento necessário. Quando um formulário padrão é usado para especificar os requisitos funcionais, as seguintes informações devem ser consideradas:

- Descrição da função ou entidade.
- Descrição de suas entradas e de onde elas vieram.
- Descrição de suas saídas e para onde elas irão.
- Informações sobre os dados necessários para o processamento e outras entidades usadas.
- Descrição da ação a ser tomada.
- Condições pré e pós (se for o caso).
- Os efeitos colaterais (se houver) da função.

Para a atividade de levantamento de requisitos do nosso curso, foi definido o seguinte formulário-padrão:

Função		RF		UC		PR	
Descrição / Ação							
Entrada							
Saída							
Pré-condição							
Pós-condição							
Stakeholder							

Em que:

- **Função:** identifica a Função do Requisito.

- **RF:** identificação do Requisito. Pode ser numérico, ou outro código que decida utilizar. Por exemplo: RF001, RF002, RNF003 etc.
- **UC:** representa, o Caso de Uso, método de especificação de requisito da UML®, uma representação gráfica, que pode ser utilizada para auxiliar na especificação de requisitos mais complexos e de difícil compreensão somente descritiva. Também corresponde a um código de identificação do *Use Case*. Por exemplo: UC001, UC002 etc.
- **PR:** representa a prioridade estipulada pelo stakeholder para o desenvolvimento do requisito.
- **Descrição/Ação:** campo destinado à especificação da função do requisito e da ação a ser tomada.
- **Entrada:** entradas necessárias para o processamento da função.
- **Saída:** saídas/resultados produzidos com a realização da função do requisito.
- **Pré-condições:** o que já deve ter ocorrido ou estar disponível antes da execução do requisito, por exemplo, <criar nota fiscal> deve preceder <modificar nota fiscal>.
- **Pós-condições:** que devem ser verdadeiras quando a execução do requisito se completar, por exemplo, <nota fiscal é modificada e consistente>.
- **Stakeholder:** identificar o tipo (grupo) ou nome do *stakeholder* solicitante desse requisito. Esse formulário é incorporado ao Documento de Requisitos no formato de tabela.

A seguir, para ilustrar e auxiliar no aprendizado, apresento um exemplo do professor Bruno Calegari, da Universidade Federal de Santa Maria. Ele demonstra a especificação de requisitos para um sistema de Bomba de Insulina, primeiro na versão em linguagem natural e, na sequência, na versão linguagem natural padronizada:

Linguagem Natural:

3.2 O sistema deve medir o açúcar no sangue e fornecer insulina, se necessário, a cada dez minutos. (Mudanças de açúcar no sangue não relativamente lentas, portanto, medições mais frequentes são desnecessárias; medições menos frequentes podem levar a níveis de açúcar desnecessariamente elevados.)

3.6 O sistema deve, a cada minuto, executar uma rotina de autoteste com as condições a serem testadas e as ações associadas definidas no Quadro 4.3 (A rotina de autoteste pode descobrir problemas de hardware e software e pode alertar o usuário para a impossibilidade de operar normalmente.)

Fonte: Calegaro (2013, p. 10).

Linguagem Natural Estruturada:

BOMBA DE INSULINA/SOFTWARE DE CONTROLE/SRS/3.3.2	
Função	Calcular as doses de insulina: nível seguro de açúcar
Descrição	Calcula a dose de insulina a ser fornecida quando o nível de açúcar esta na zona de segurança entre três e sete unidades
Entradas	Leitura atual de açúcar(r2), duas leituras anteriores (r0 e r1)
Fontes	Leitura atual da taxa de açúcar pelo sensor. Outras leituras de memória
Saídas	CompDose – a dose de insulina a ser fornecida
Destino	Loop principal de controle
Ação	CompDose é zero se o nível de açúcar esta estável ou em queda ou se o nível esta aumentando, mas a taxa de aumento esta diminuindo. Se o nível esta aumentando e a taxa de aumento esta aumentando, então CompDose é calculado dividindo-se a diferença entre o nível atual de açúcar e o nível anterior por quatro e arredondando-se o resultado. Se o resultado é arredondado para zero, então CompDose é definida como a dose mínima que pode ser fornecida
Requisitos	Duas leituras anteriores, de modo que a taxa de variação do nível de açúcar pode ser calculada
Pré-Condições	O reservatório de insulina contem, no mínimo, o Maximo da dose única permitida de insulina
Pós-Condições	R0 é substituída por r1 e r1 é substituída por r2
Efeitos colaterais	Nenhum

Fonte: Calegaro (2013, p. 13).

Quando se tem pouco acesso ou informação nenhuma sobre o que se deseja desenvolver/produzir, inevitavelmente você tenderá a produzir definições com base em deduções sobre o assunto. Essas definições podem não representar o que o cliente deseja para o seu produto.

Para o problema de insuficiência de informação, a solução é: busque-as. Visite a organização do cliente, converse com os usuários, levante documentos, desenhe esquemas, pesquise atividades semelhantes ou, até mesmo, sistemas concorrentes, fato é, será necessário ir a campo e interagir. Para o caso de não existir nenhuma informação, registre suas hipóteses associadas a cada requisito, isso ajudará nas etapas de negociação e validação.

Outra dica para escrever seus requisitos: separe funcionalidade de implementação. É frequente a especificação ter a ver com a implementação e não com a necessidade. As especificações devem referir o que é necessário para o sistema e não como vai ser disponibilizado ou programado. Para evitar a especificação de implementações, pergunte-se: por que precisamos deste requisito? Por exemplo:

- Projeto: Desenvolver um sistema para gerenciamento de requisitos.
- Requisito Funcional: RF001 – Fornecimento de uma base de dados.

Pergunta:

Por que precisamos fornecer uma base de dados?

Respostas:

1. Disponibilizar capacidade para o acompanhamento dos requisitos.
2. Disponibilizar capacidade para adicionar atributos aos requisitos.
3. Disponibilizar a possibilidade de ordenar os requisitos.

Observe que as respostas são claramente os requisitos reais, isto é, são as necessidades, elas que deveriam representar o RF001. Já a descrição: “Fornecimento de uma base de dados” é problema para a implementação.

Sempre pergunte ao seu requisito para que você precisa dele e verifique se a resposta te remete para uma ou mais necessidades. Se isso ocorrer, refine o requisito, separando a necessidade da implementação, senão você tem a garantia de que definiu a função da maneira correta.

Por fim, lembre-se sempre: os requisitos não devem ser dúbios, devem ser mensuráveis, devem possibilitar testes, devem ser investigáveis, completos, consistentes e aceitáveis para todas as partes envolvidas.

Os atributos de qualidade aplicados nos requisitos são, basicamente, os mesmos aplicados na fase de Validação dos requisitos:

1. Não Ambiguidade: cada requisito deve possuir, apenas, uma interpretação.
2. Completude: é a capacidade de cada requisito especificar claramente seu assunto, objetivo.
3. Consistência: é a capacidade de cada requisito não ser contraditório a outro.
4. Correção/Exatidão: é a capacidade do requisito de estar correto.

Escrevendo assim, parece uma tarefa fácil, não se engane, não subestime este trabalho. Pronto. Sabemos como ele tem que ser! Agora, é só praticar!

ESPECIFICAÇÃO DE REQUISITO POR CASO DE USO

Caso de Uso é um diagrama da UML®, Figura 17, utilizado para elicitación de requisitos. Por ser gráfico, pode facilitar na compreensão de alguns requisitos. Pode ser utilizado sozinho ou acompanhado de especificação detalhada dos casos de uso em linguagem natural.

Para aprender de forma satisfatória como especificar um requisito utilizando o diagrama de Caso de Uso, seria necessária uma disciplina específica para isso, o objetivo desta seção é apresentar, de forma generalizada, o Caso de Uso e relacionar uma técnica de modelagem com o conteúdo apresentado.

Um modelo Caso de Uso descreve uma função proposta para o sistema a ser desenvolvido. Ele deve representar uma interação entre um utilizador (humano ou máquina) e o sistema. Essa interação deve representar uma única funcionalidade, por exemplo: faz pagamento ou ver detalhes do pagamento.

Cada Caso de Uso descreve a funcionalidade a ser construída no sistema proposto; o que pode incluir a funcionalidade de outro Caso de Uso ou estender outro Caso de Uso com seu próprio comportamento.

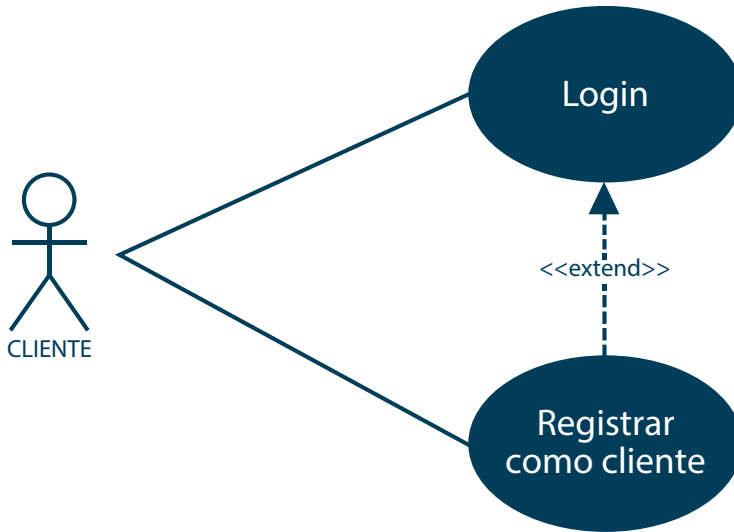


Figura 18: Exemplo de Caso de Uso
Fonte: o autor.

Casos de Uso são tipicamente relacionados com “atores”, Figura 19, que são entidades, pessoas ou não, que usam ou interagem com o sistema para realizar alguma rotina.



Figura 19: Representação da entidade Ator
Fonte: o autor.

Um Caso de Uso pode incluir a funcionalidade de outro como parte de seu processamento. Geralmente, esse relacionamento é denominado *Include*. O processo *Include* é chamado toda vez que o caminho básico é executado.

Como no exemplo da Figura 20, para executar a função do requisito: *o sistema deve permitir ao usuário vendedor escolher na lista de pedidos de cliente a ordem de classificação por cliente*. Antes de alterar a ordem, o caso de uso <ordem de classificação> irá incluir (*Include*) o caso de uso <alterar ordem de classificação> toda vez que for executado.



Um caso de uso pode ser incluído por um ou mais outros casos de uso. Esse procedimento ajuda a reduzir redundância, reutilizando um componente que é comum e que pode ser reutilizado. O Include é denominado como um tipo de relacionamento do diagrama de caso de uso.

Outro tipo de relacionamento é o *Extend*. O Atributo *Extend* possibilita que um caso de uso estenda o comportamento de outro, nesse caso, diferente do *Include*, quando circunstâncias excepcionais são encontradas. Por exemplo, o requisito: *o sistema deve exigir a aprovação de usuário máster para alterar dados do pedido de cliente*. Para elicitare esse requisito, o caso de uso <obter a aprovação> deverá, quando necessário, estender o comportamento normal do caso de uso <alterar pedido de cliente>, permitindo o acesso. Como apresentado na Figura 20.

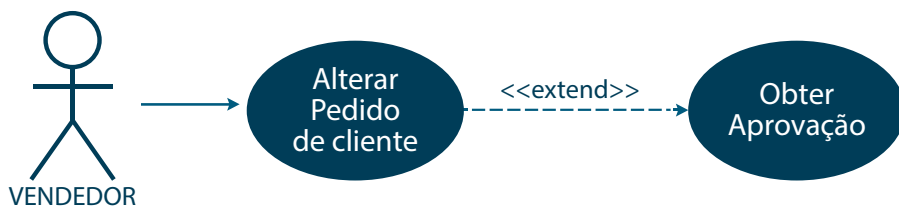


Figura 20: Diagrama de Caso de Uso: Alterar Pedido de Cliente

Fonte: o autor.

Include e *Extend* são relacionamentos que podem ocorrer somente entre casos de uso. A modelagem por caso de uso permite, além do relacionamento entre casos de uso, relacionamento entre atores e entre atores e casos de uso.

Os relacionamentos entre atores são comunicação (ou associação), representado por uma seta de base aberta, e a especialização (ou generalização), representado por uma seta de base fechada.

Um relacionamento de comunicação representa que os dois atores, de forma uni ou bidirecional, realizam uma troca de informação ou de mensagem, que tem significado formal para o sistema a ser desenvolvido.

O relacionamento entre um ator e um caso de uso expressa sempre uma comunicação. Um ator, enquanto entidade externa, não pode assumir nenhum relacionamento estrutural com o sistema computacional.

SAIBA MAIS

O Documento de Requisitos é a especificação das características operacionais do sistema de software a ser produzido, permite construir modelos que descrevem atividades funcionais, classes de dados, relacionamentos e fluxo de dados. Existem várias técnicas para a modelagem dos requisitos de software, a UML® (*Unified Modeling Language* ou Linguagem de Modelagem Unificada) é uma delas. A UML® permite uma modelagem orientada a objetos, que possibilita, por meio de diagramas, planejar e definir toda a estrutura de fluxo de dados do sistema a ser desenvolvido.

Aprenda mais sobre a UML® acessando os links a seguir:

- Devmedia - Introdução a UML® – Unified Modeling Language ou Linguagem de Modelagem Unificada: <<http://www.devmedia.com.br/introducao-a-uml-unified-modeling-language-ou-linguagem-de-modelagem-unificada/6928#ixzz3ZeYZMXWJ>>. Acesso em: 24 jul. 2015.
- OMG® - The Object Management Group® is an international, open membership, not-for-profit technology standards consortium: <<http://www.uml.org/>>. Acesso em: 24 jul. 2015.

Fonte: o autor.

CONSIDERAÇÕES FINAIS

Estudamos, nesta unidade, o processo de engenharia de requisitos, nesse ponto, entendemos por que é considerado um dos mais importantes processos da engenharia de software.

Requisitos bem especificados favorecem todas as outras atividades do processo de desenvolvimento de software, garantindo, ainda, que seja realizado dentro do custo e prazo estabelecido e, mais importante, que satisfaça a necessidade do cliente.

Estudamos que são quatro as principais atividades para realização da engenharia de requisitos. A primeira delas é o levantamento de requisitos, o momento em que o cliente vai declarar sua expectativa quanto ao sistema que será desenvolvido e a equipe de desenvolvimento levanta, em um primeiro momento, em linguagem natural e não tão formal, os requisitos funcionais e suas restrições. Aprendemos que existem várias técnicas para o levantamento de requisitos, por exemplo, a entrevista é uma técnica na qual o engenheiro de requisito realiza uma conversa com os diferentes *stakeholders* envolvidos no processo de desenvolvimento, com o objetivo de definir quais são os requisitos necessários para a fabricação do software. Em seguida, as fases de negociação e validação dos requisitos, o que garantirá que sejam coesos, claros, não ambíguos e rastreáveis. Na última etapa, verificamos formas de especificar os requisitos no Documento de Requisitos. Os requisitos devem ser descritos de uma maneira que possibilite que sejam verificados e em uma linguagem ou padronização que possa ser compreendida por leitores técnicos e não técnicos.

Depois de conhecer todos os processos e entender por que são necessários, não podemos mais subestimar a importância do requisito no desenvolvimento de software. Sabemos que, na prática do desenvolvimento de software, temos a tendência de não nos prolongarmos na fase do levantamento e partir logo para a implementação, mas, considerando tudo que foi apresentado até agora, você percebe quais seriam os impactos no projeto? Consegue perceber por que tantos projetos de software não cumprem prazos, nem custos, muito menos atendem às necessidades do cliente? Vai, agora, aplicar toda essa teoria na prática?

ATIVIDADES



1. Processos da engenharia de requisitos podem variar muito em função das características dos projetos e, até mesmo, das organizações. Pressman (2010) organiza as atividades de engenharia de requisitos montando o processo representado pelas atividades constantes em qual das alternativas abaixo?
 - a) Levantamento, Elaboração, Negociação, Especificação, Validação e Gestão.
 - b) Levantamento, Análise, Documentação, Verificação/Validação e Gerência.
 - c) Análise, Levantamento, Documentação e Gerência.
 - d) Levantamento, Elaboração, Negociação, Especificação e Gestão.

2. A Elicitação de requisitos, fase na engenharia de requisitos, envolve técnicas de coleta de informações. Uma delas ocorre em ambiente informal, onde toda ideia é considerada como uma possível solução de um problema, nessa técnica, críticas são proibidas e todas as sugestões apresentadas são encorajadas, mesmo que pareçam, em um primeiro momento, estranhas e fora de contexto. Assinale a alternativa que contenha a técnica com essas características:
 - a) Prototipação.
 - b) Entrevista.
 - c) Questionário.
 - d) *Brainstorming*.
 - e) Análise de protocolos.

3. São técnicas para a elicitação de requisitos de um software:
 - I. Joint Application Development (JAD).
 - II. Questionário.
 - III. Entrevista.

Quais estão corretas?

 - a) Apenas I está correta.
 - b) Apenas III está correta.
 - c) Apenas I e II estão corretas.
 - d) I, II e III estão corretas.

ATIVIDADES



4. Qual ou quais alternativas representam a etapa que inicia o processo de engenharia de requisitos:
 - I. O levantamento de requisitos é o início da atividade de desenvolvimento de software.
 - II. Esta fase também é conhecida como Elicitação de requisitos, em que elicitacão quer dizer ato ou efeito de eliciar; confrontar; aliciar; conseguir obter resposta ou informação.
 - III. É o ir a campo e compreender o que o cliente deseja e como será desenhada sua proposta para atender às expectativas dele.
 - a) Apenas I está correta.
 - b) Apenas III está correta.
 - c) Apenas I e II estão corretas.
 - d) I, II e III estão corretas.
5. A obtenção de requisitos não é um processo formal, por isso, não permite sua automatização. O engenheiro de requisitos deve contar com técnicas de entrevista, questionário, mapeamento, entendimento do processo-alvo. São técnicas de elicitação de requisitos.
 - a) Questionário, Entrevista, Brainstorming.
 - b) Questionário, Scrum, Oficinas.
 - c) Negociação, Questionário, *Brainstorming*.
 - d) Scrum, Questionário, Entrevista.
6. “De forma geral o que se sugere é que os riscos associados a cada requisito sejam identificados e analisados para então elaborar estimativas de esforço de desenvolvimento de forma a avaliar o impacto de cada requisito no custo do projeto e no prazo de entrega”. Essa afirmação diz respeito a qual fase do processo de engenharia de requisito.
 - a) Elicitação de requisitos
 - b) Validação de requisitos.
 - c) Negociação de requisitos.
 - d) Especificação de requisitos.
7. Sobre a validação de requisitos, analise as afirmações e indique a alternativa correta:
 - I. O objetivo da validação de requisitos é descobrir erros nos requisitos documentados.
 - II. Nessa fase, se uma divergência for encontrada, é feita uma pausa no desenvolvimento e nos testes e se levanta, novamente, o requisito.

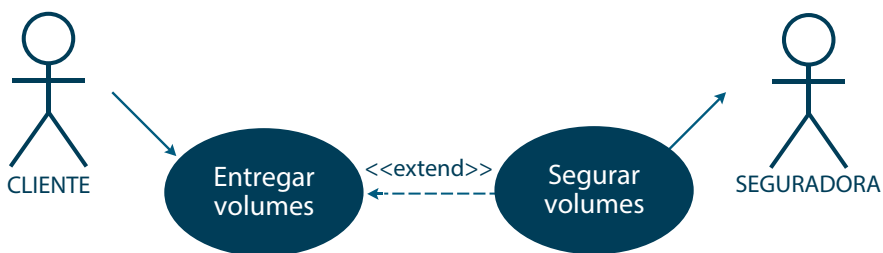
ATIVIDADES



III. Nesse sentido, a validação é extremamente importante, pois o custo para correção de um requisito nessa fase é bem inferior ao custo nas fases posteriores, implementação ou testes, por exemplo.

- a) Apenas I está correta.
- b) Apenas II e III estão corretas.
- c) Apenas I e III estão corretas.
- d) I, II e III estão corretas.

8. Qual das alternativas é melhor representada pela figura a seguir:



- a. O cliente tem autorização para solicitar entrega de volumes. Alguns volumes de maior valor podem ser assegurados, os clientes podem solicitar isso do sistema e uma companhia de seguro foi contratada para segurar os volumes de maiores valores.
- b. O relacionamento *extend* é usado em casos em que comportamento opcional ou excepcional é inserido em um caso de uso existente.
- c. O cliente deve solicitar autorização para um usuário máster para solicitar entrega de volumes. Alguns volumes de maior valor podem ser assegurados, os clientes podem solicitar isso do sistema e uma companhia de seguro foi contratada para segurar os volumes de maiores valores.
- d. A seguradora pode informar ao cliente que ele deve contratá-la para assegurar cargas de maiores volumes.



A IMPORTÂNCIA DA ENGENHARIA DE REQUISITOS PARA O CICLO DE DESENVOLVIMENTO DE SOFTWARE DE TEMPO REAL

A análise e a especificação de requisitos são atividades fundamentais no processo de desenvolvimento de software, influenciando diretamente no desenvolvimento satisfatório de sistemas. Por se tratarem de atividades de grande importância no ciclo de vida do software, e que se relacionam diretamente com a qualidade do produto a ser desenvolvido, a Engenharia de Requisitos precisa ser devidamente planejada. Sendo assim, deve ser aplicada de forma abrangente para assegurar que um conjunto completo das necessidades e requisitos dos usuários sejam capturados, e posteriormente, transformados em um conjunto validado de requisitos em todo o ciclo de vida. A Engenharia de Requisitos (RE) é uma abordagem disciplinada e sistemática para elicitar, especificar, analisar, confirmar, validar e gerenciar requisitos enquanto considera usuários, objetivos e necessidades técnicas, econômicas e de negócio. Ela abrange todo o processo de desenvolvimento do software e é caracterizada, em diferentes bibliografias, como o processo mais crítico e complexo do ciclo de desenvolvimento de software. A principal razão é que o processo de engenharia de requisitos tem impacto dominante sobre as capacidades do produto resultante.

A crescente complexidade dos sistemas de software tornam as atividades de Engenharia de Requisitos tanto mais importantes quanto difíceis. Para minimizar a complexidade dos sistemas de tempo-real (STR) (Real-Time Systems), que são encontrados em muitos setores, como plantas industriais, telecomunicações, transporte, militar e de saúde, são utilizados modelos que de forma gráfica ou textual auxiliam na compreensão e representação desses sistemas. Por sua própria natureza, projetar softwares de tempo-real traz exigências específicas de análise, projeto e teste, exigindo assim habilidades/metodologias especiais para seu projeto e desenvolvimento. Com isso, a análise de STR requer modelos de software que possibilitem avaliar questões de tempo, sincronização e dimensionamento destes softwares. A modelagem de aplicações de tempo-real não é um processo trivial, pois o projeto das mesmas deverá conter, na maioria dos casos, uma análise distribuída em rede do comportamento e disposição dos diversos componentes do sistema, como por exemplo, os sensores e atuadores, que estão aptos, respectivamente, a captar estímulos gerados externamente e a responder a tais estímulos dentro de um intervalo de tempo finito





e especificável. Dada a grande quantidade de problemas já relatados na literatura em relação ao desenvolvimento de software, torna-se necessário aplicar metodologias que possibilitem tratar e manipular coerentemente as características inerentes aos softwares em geral, como por exemplo, a intangibilidade e o alto grau de abstração. A complexidade de softwares de tempo-real demonstra-se crescente quando de sua especificação e análise. Os sistemas de

Fonte: Ribeiro e Souza (2012).

tempo-real possuem requisitos específicos, e dada a grande importância deste tipo de sistema, devem ser claramente expressados. Por esta razão, e para minimizar as dificuldades para a modelagem de requisitos de sistemas de tempo-real, propõem-se, neste trabalho, utilizar o profile SysML, que estende a UML, em conjunto com estereótipos do profile MARTE para representar requisitos não-funcionais de sistemas de tempo-real.



NA WEB

Para entender um pouco mais sobre a necessidade de se especificar de forma clara e objetiva os requisitos, assista ao vídeo disponível em: <<https://www.youtube.com/watch?v=fHjc3CJ6m00>>. Acesso em: 06 jul. 2015.



LIVRO

Engenharia de Software - Qualidade e Produtividade com Tecnologia

Kechi Hirama

Editora: Elsevier

Sinopse: Neste livro, o professor Kechi Hirama, da Escola Politécnica da USP, apresenta uma visão moderna e inovadora do conceito de Engenharia de Software. Em vez de explorar à exaustão tópicos tradicionais da disciplina, como fazem os títulos considerados clássicos da área, dá ênfase aos conceitos requisitados pelos entrantes nesse universo, tais como documentação de processos, novas formas de padrões arquiteturais, gerência de projetos OO, gerência de requisitos e, por fim, o novíssimo manifesto ágil (Scrum) e os projetos e implementação de serviços SOA. A importância do livro se concretiza pelo fato de apresentar todos os temas de maneira direta e totalmente voltada para a realidade dos estudantes e professores brasileiros.



GESTÃO DE REQUISITOS

UNIDADE

IV

Objetivos de Aprendizagem

- Entender como gerenciar mudanças em requisitos já definidos.
- Verificar como gerenciar os relacionamentos entre requisitos e entre o Documento de Requisitos e demais documentos produzidos durante o desenvolvimento.

Plano de Estudo

A seguir, apresentam-se os tópicos que você estudará nesta unidade:

- Evolução dos requisitos
- Rastreabilidade
- Controle de mudanças
 - Planejamento do gerenciamento
 - Gerenciamento de mudança de requisitos
- Ferramentas para o gerenciamento de requisitos

INTRODUÇÃO

Quando começamos nossos estudos na unidade I, conversamos sobre o quão complexo é o processo de desenvolvimento de um software. A construção de um sistema não é algo físico, não é palpável, não é como construir uma casa, ou um computador, exige interpretação, envolve dedução. Várias pesquisas e estudos, como os executados pelo Pulse of the Profession® do PMI, apontam como principal fator pelas falhas em Projetos o inadequado Gerenciamento de Requisitos. Isso porque os programadores devem seguir especificações para o desenvolvimento de um produto de software, quando nem sempre eles participam da fase em que elas são elicitadas.

O gerenciamento de requisitos deve iniciar nessa fase, durante o levantamento dos requisitos, ali, as políticas de gestão dos requisitos devem ser definidas considerando as características e necessidades do software a ser desenvolvido. Terá início efetivo, a partir da liberação da primeira versão do Documento de Requisitos, e permanece ativo durante todo o ciclo de desenvolvimento do produto de software. A gestão de requisitos ultrapassa o ponto de entrega do produto ao cliente, por se manter necessário o controle de mudanças quando o produto desenvolvido entra na fase de manutenção.

Uma gestão de requisitos mal planejada e mal executada provoca impactos e, como já foi dito, tem relação direta com o sucesso ou com o fracasso do projeto. O PMI apresenta pesquisas demonstrando que, dos projetos que fracassam, 47% deles têm como causa base uma gestão de requisitos mal feita.

Nesta unidade, estudaremos a gestão de requisitos e serão apresentadas formas de lidar com a volatilidade dos requisitos, o que nos possibilitará entender que a evolução é uma característica inerente aos sistemas computacionais e deve ser levada em conta durante todas as fases do projeto de desenvolvimento do software.

Para entender do que se trata a gestão de requisitos, vamos retomar nossa abordagem de relacionar o significado das palavras com a atividade que elas nomeiam. Vamos lá! Busque no seu navegador ou no seu dicionário o significado da palavra gestão. O Michaelis (online) define: “ges.tão *sf (lat **gestione**)* **1** Ato de gerir. **2** Administração, direção. **G. de negócio:** administração oficiosa de negócio alheio, feita sem procuração”.

Se gestão significa administrar, então, o primeiro passo é definir quais são os objetivos e criar políticas de identificação, controle e rastreamento das informações necessárias para atingi-los.

Assim, a gestão de requisitos começa com a identificação. Cada requisito deve receber um atributo responsável por identificá-lo e torná-lo único em um processo de busca. Pense nessa identificação como o número do seu RG (Registro Geral). Depois de identificados, segundo Pressman (2010), tabelas de rastreamento podem ser desenvolvidas relacionando com um ou mais aspectos do sistema ou seu ambiente.

Cada vez mais fábricas de software buscam pela maturidade nos seus processos de desenvolvimento, a fim de finalizarem seus projetos dentro do prazo e com os menores custos. Sayão e Koogan, em artigo publicado pela PUC-RIO, relatam que a gerência de requisitos é vista como um dos principais problemas a serem superados para que as organizações cheguem ao nível 2 de maturidade do modelo CMMI (*Capability Maturity Model Integration*) do SEI (*Software Engineering Institute*). O propósito do processo Gerência de Requisitos, segundo o Programa MPS.BR nível G (Parcialmente Gerenciado), é gerenciar os requisitos do produto e dos componentes do produto do projeto e identificar inconsistências entre os requisitos, os planos do projeto e os produtos de trabalho do projeto.

Pronto, essa é a visão geral da gestão de requisitos, agora, vamos estudar com um pouco mais de detalhes o conteúdo envolvido na gestão de requisitos.

EVOLUÇÃO DOS REQUISITOS

Requisitos mudam, essa é uma certeza no desenvolvimento de software, Pressman (2010) complementa afirmando que o desejo de mudar dos requisitos persiste ao longo da vida do sistema. A evolução é uma característica inerente aos sistemas computacionais. O surgimento de novos requisitos ou a necessidade de alterar um já existente acontece durante o processo de desenvolvimento e, até mesmo, depois que o sistema tiver em operação, isso se chama evolução de sistemas.

Você já deve ter ouvido falar sobre o “bug do milênio”, ou então sobre o frenesi que o lançamento do sistema operacional Windows provocou no mercado de desenvolvimento de software para interface gráfica, quando lançado. Esses são exemplos clássicos de que simples alterações no macrosistema afetam os sistemas de software, que ou evoluem para se adaptarem ou se tornarão inúteis.

Outros fatores de mudanças nos requisitos são: especificação mal descrita ou errada de requisito (inconsistências, conflitos); mais conhecimento adquirido pelo cliente ou usuário com o decorrer do projeto (mudam de ideia, quando entendem o que realmente querem do sistema); mudança de prioridade do cliente; mudança de tecnologia; mudança de leis, políticas e processos internos ou externos à organização vinculada ao projeto.

A literatura nos apresenta duas classificações de requisitos a partir da perspectiva de evolução do requisito. Sommerville (2008) apresenta os requisitos Permanentes e os Requisitos Voláteis:

- **Requisitos Permanentes:** são os requisitos criados a partir do domínio da aplicação, são relativamente estáveis, mudam mais lentamente que os requisitos voláteis. Por exemplo, em um sistema de recursos humanos, sempre existirão os requisitos relacionados à documentação dos funcionários, ou referências da organização, ou, ainda, referentes ao calendário básico. É padrão os sistemas informatizados disponibilizarem uma aba/janela/menu referente aos Cadastros Básicos.
- **Requisitos Voláteis:** são requisitos que irão mudar durante o tempo de desenvolvimento do sistema, provavelmente, depois, quando já estiverem em operação. Por exemplo, as regras de pagamento do funcionário podem sofrer alterações por imposição de leis ou por mudança na política da empresa.

Sommerville (2010) divide os requisitos voláteis em quatro classes:

- **Mutáveis:** são os requisitos que se modificam devido às mudanças no ambiente em que a organização está inserida.
- **Emergentes:** são os requisitos que surgem conforme a compreensão do cliente referente ao sistema. Dessa compreensão, novos requisitos podem surgir. São emergentes, também, aqueles requisitos que o processo do projeto revela.

- **Consequentes:** são os requisitos que surgem após a inserção do sistema na organização. São as solicitações de adaptações ou criação de novas rotinas desencadeadas pelo uso do sistema até sua perfeita adaptação às rotinas da organização.
- **Compatibilidade:** são os requisitos que dependem de outro sistema, equipamento ou processo. Conforme esse sistema, equipamento ou processo os requisitos também mudam.

RASTREABILIDADE

Não é tópico para os nossos estudos, mas o quesito qualidade de software deve ser mencionado quando falamos em gerenciamento de requisitos. Analise o seguinte: uma denominação simplista do objetivo da qualidade de software é entregar o que o cliente espera receber.

Qualidade para o CMMI é medida por níveis de maturidade nos processos de desenvolvimento de software. Se observar as exigências, irá encontrar que o conjunto de atividades que compõe o processo de desenvolvimento de requisitos é exigido no nível 3 de maturidade, já as atividades de Gerenciamento de Requisitos estão no nível 2 de maturidade. Percebe o que isso quer dizer? Isso significa que uma organização que busque certificação de qualidade pelo CMMI deve, primeiro, focar sua atenção nos processos de Gerenciamento de Requisitos. Gerir requisitos, então, é o primeiro passo para um processo de desenvolvimento de software maduro e que garanta um produto final com qualidade.

O que garante um gerenciamento de requisitos bem executado? Um processo de controle de mudanças conciso e um atributo de rastreabilidade bem definido. A rastreabilidade de requisitos é uma das atividades aconselhadas por vários modelos de qualidade, por exemplo, o CMMI, MPS-BR e ISO 9001.

Rastreabilidade pode ser definida como o conjunto de ligações entre os requisitos, com as fontes dos requisitos, com eles mesmos e com outros artefatos. A rastreabilidade pode ser conseguida vinculando o requisito ao *stakeholder* que trouxe a informação, ou vinculando o requisito aos requisitos relacionados, ou ainda vinculando o requisito aos padrões do projeto. A condição de rastreamento deve ser atribuída ao requisito no momento do seu levantamento e ela deverá refletir a facilidade de se encontrar os requisitos relacionados. Sommerville (2008) relaciona três tipos de rastreabilidade de requisitos:

1. Por *stakeholders*: vincula o requisito ao *stakeholder* que o propôs. Essa informação é utilizada, quando uma mudança é proposta, para descobrir os *stakeholders*, para que possam ser consultados.
2. Requisitos dependentes: quando o requisito é vinculado a outro requisito na sua concepção, essa informação pode ser utilizada para avaliar quantos requisitos serão afetados pela mudança proposta e a extensão das mudanças consequentes dela que serão necessárias.
3. Modelos de projeto: vincula o requisito ao modelo de projeto onde foi implementado. Essa informação é utilizada para avaliar o impacto das mudanças para o projeto e implementação.

É preciso decidir qual critério de rastreabilidade será utilizado para atribuir ao requisito já no momento da sua especificação.

Para deter a habilidade de acompanhar e descrever a vida de um sistema, tanto fazendo o caminho da concepção do requisito até sua implementação quanto voltando da função em execução até seu requisito gerador, as matrizes de rastreabilidade podem ser implementadas por meio de uma ferramenta simples, como um editor de textos ou uma planilha eletrônica (muitas das ferramentas comercialmente disponíveis para a fase de requisitos utilizam alguma forma de matriz de rastreabilidade).

MATRIZ DE RASTREABILIDADE			
Identificação do Projeto	Nome do projeto	Data:	Data de criação da matriz
Gerente do Projeto	Responsável pelo Projeto		
Resumo do Projeto	Descrição resumida do produto final		

Id.	Requisito	Doc. Fonte	Arquitetura	Componente	Caso de teste

Figura 22: Exemplo de Matriz de Rastreabilidade por Requisito e Caso de teste

Fonte: o autor.

A matriz deverá ser preenchida com os requisitos, expressos em linguagem natural e numerados sequencialmente. As colunas subsequentes representam os documentos gerados durante o desenvolvimento do software. Nesse formato, a correspondência nem sempre é da ordem de um para um, isto é, um mesmo requisito pode ser verificado por vários casos de teste e, por outro lado, vários casos de teste podem estar verificando um único requisito. A Figura 22 representa uma matriz de rastreabilidade por requisito e caso de teste. A Figura 23 representa uma matriz de rastreabilidade por requisitos funcionais e não funcionais.

MATRIZ DE RASTREABILIDADE			
Identificação do Projeto	Nome do projeto	Data:	Data de criação da matriz
Gerente do Projeto	Responsável pelo Projeto		
Resumo do Projeto	Descrição resumida do produto final		

Requisitos Funcionais	Requisitos não funcionais					
	RNF01	RNF02	RNF03	RNF04	RNF05	RNF06
RF01						
RF02		↙				↙
RF03			↙		↙	
RF04	↙					
RF05				↙		

Figura 23: Exemplo de Matriz de Rastreabilidade por Requisitos Funcionais e Requisitos Não funcionais

Fonte: o autor.

Nossa Matriz de Rastreabilidade desenvolvida na unidade II representa um modelo simples de uma ferramenta de rastreabilidade de requisitos. A seguir, relaciono alguns exemplos de matriz de rastreabilidade:

- Rastreamento das fontes de requisitos (relaciona o requisito, *stakeholders* e documento de requisitos).
- Rastreamento da razão dos requisitos (relaciona o requisito com a descrição do porquê o requisito foi especificado).
- Rastreamento requisitos-requisitos (relaciona requisitos com outros requisitos que são dependentes uns dos outros).
- Rastreamento requisitos-arquitetura (relaciona os requisitos com os subsistemas onde foram implementados).
- Rastreamento requisitos-projeto (relaciona os requisitos com o hardware específico ou componentes de software que são usados para implementá-los).
- Rastreamento requisitos-interface (relaciona os requisitos com a interface externa do sistema que será usada para apresentar os requisitos).

Vários caminhos de rastreamento são possíveis:

- Rastreamento *Backward-from*: vincula requisitos a suas fontes em outros documentos ou *stakeholders*.
- Rastreamento *Forward-from*: vincula requisitos ao projeto e componentes de implementação.
- Rastreamento *Backward-to*: vincula o projeto e os componentes de implementação aos requisitos.
- Rastreamento *Forward-to*: vincula outros documentos (que possam ter precedido o documento de requisitos) aos requisitos relevantes.

Existem, no mercado, várias ferramentas para o gerenciamento de requisitos, proprietárias e livres. Por exemplo: IBM® Rational® da IBM; EasyRM Version 1.06 da Cybernetic Intelligence GmbH; Borland Caliber Analyst da Borland; Caliber da Microfocus, OSRMT (*Open Source Requirements Management Tool*) ferramenta livre, DotProject, distribuído sob a licença GNU-GPL (*GNU General Public License*) e outros.

Para sistemas de pequeno porte, geralmente, o rastreamento é suportado por ferramentas simples, como processadores de texto e planilhas de cálculo. Nos sistemas de grande porte, a utilização dessas ferramentas fica comprometida devido à quantidade de informação gerada. Nesse caso, a utilização de ferramentas automatizadas de gerenciamento de requisitos é recomendada.

Temo me considerar repetitiva, mas, novamente, a definição dos elementos que comporão seu processo de rastreabilidade deve ser adaptada às necessidades específicas do seu projeto, ou melhor, de cada projeto a ser desenvolvido. Não adianta rastrear uma quantidade excessiva de informações se não for viável a sua manutenção e utilização. A definição dos elementos a serem relacionados para o rastreamento deve considerar prazos e custos do projeto, além dos processos e dos padrões em uso na organização.



SAIBA MAIS

O site IBM Developer Works postou uma notícia falando sobre a importância de se capturar e gerenciar requisitos.

A notícia retrata como uma falha clássica, para conceituar uma arquitetura física de um produto, resultou em problemas de integração no tempo de execução, gerando um diagnóstico inicial que designava o software como culpado do mau funcionamento, já que era difícil para os mecânicos testar o componente fisicamente. A falta de critério na especificação, no momento do levantamento, provocou custos adicionais e sobrecarregou a fábrica, devido a necessidade da correção da operação que já estava em execução.

Para saber mais, acesse: <https://www.ibm.com/developerworks/community/blogs/academia/entry/a_import_c3_a2ncia_de_levantar_e_gerenciar_requisitos_sensor_de_chuva_de_autom_c3_b3veis23?lang=en>. Acesso em: 05 maio 2015.

Fonte: o autor.

CONTROLE DE MUDANÇAS

Assumindo que os requisitos mudam, o gerenciamento de requisitos deve, então, se ocupar de controlar essas mudanças de forma sistemática e eficiente. Sommerville (2008) afirma que 65% da manutenção de um sistema estão relacionados à implementação de novos requisitos, 18% à modificação de requisitos já existentes e 17% à correção de defeitos de um sistema. Podemos, então, com base nessas informações, considerar que a manutenção dos sistemas é uma extensão do desenvolvimento de software, com atividades associadas às de especificação, projeto, implementação e testes.

Percebemos que parte do processo de gestão de requisitos é controlar mudanças. O CMMI afirma que é fundamental que as alterações dos requisitos sejam:

- Identificadas e avaliadas.
- Avaliadas sob o ponto de vista de risco.
- Documentadas.
- Planejadas.
- Comunicadas aos grupos e indivíduos envolvidos.
- Acompanhadas até a finalização.

O processo de controle de mudanças permite que todas as mudanças sejam rastreadas e garante que nenhuma solicitação seja perdida ou deixada de lado. Sommerville (2008) recomenda que o planejamento desse processo deve começar junto com o levantamento inicial do requisito, e o gerenciamento ativo das mudanças deve começar assim que a primeira versão do Documento de Requisitos for liberada.

PLANEJAMENTO DO GERENCIAMENTO

Um requisito somente será rastreável se for possível identificar sua concepção, porque ele existe, quais os requisitos relacionados a ele e qual o seu relacionamento

com outras informações, tais como: projeto do sistema, testes, implementações e documentação do usuário. Para isso, Sommerville (2008) recomenda que, na fase de planejamento do gerenciamento do requisito, sejam considerados os seguintes aspectos:

- Identificação única para os requisitos: definição da forma de identificação do requisito. Para ser rastreável, o requisito deve ser único.
- Processo de gerenciamento de mudança: definição das atividades do processo. Seja estabelecido um conjunto de atividades que irá avaliar o impacto e o custo da mudança.
- Políticas de rastreamento: definição dos elementos de rastreamento. É necessário decidir como será o relacionamento entre os requisitos e entre os requisitos e o projeto de software, bem como decidir como serão mantidos esses relacionamentos.
- Suporte de ferramenta CASE: definição de uma ferramenta automatizada para gerenciar o processo. O gerenciamento de requisitos envolve uma grande quantidade de informações que deve ser mantida, essa ferramenta pode ser um sistema especializado em gestão de requisitos como, também, uma planilha eletrônica. Essa ferramenta dará apoio para as atividades de armazenar requisitos, gerenciar mudanças e rastreabilidade.



REFLITA

“Se você deseja reduzir a deteriorização do software, terá de melhorar o projeto de software.”

Fonte: Roger S. Pressman (2010 p. 5).

GERENCIAMENTO DE MUDANÇA DE REQUISITOS

Quando são propostas mudanças, é preciso verificar o impacto que elas provocarão sobre o requisito, nos outros requisitos relacionados e, também, no projeto de software. O processo definido no planejamento do gerenciamento de

mudanças deve ser aplicado a todas as mudanças propostas de forma consistente e controlada. Para o processo de gerenciamento de mudanças, três atividades são propostas por Sommerville (2008):

- I. **Análise do problema e especificação da mudança:** essa fase se inicia quando é detectado um problema no requisito que precisa ser remodelado, ou quando uma proposta/solicitação de mudança é apresentada. O problema ou a proposta deve ser analisado e validado, isso porque uma mudança pode ser solicitada simplesmente pela falta de entendimento do solicitante da especificação do requisito. Como resultado dessa fase, uma melhor especificação da mudança deve ser definida.
- II. **Análise de custo:** a segunda etapa do processo de gerenciamento de mudanças de requisito analisa o efeito que a mudança proposta provoca no sistema. Utilizando a rastreabilidade, o custo da mudança é estimado considerando a proporção de alterações que serão necessárias no documento de requisitos e no projeto. No final dessa etapa, decide-se pela aceitação ou não da mudança proposta.
- III. **Implementação de Mudança:** uma vez aprovada a mudança, o documento de requisitos e o projeto são alterados.

A Figura 24 representa o processo de gerenciamento de mudança de requisitos proposto.

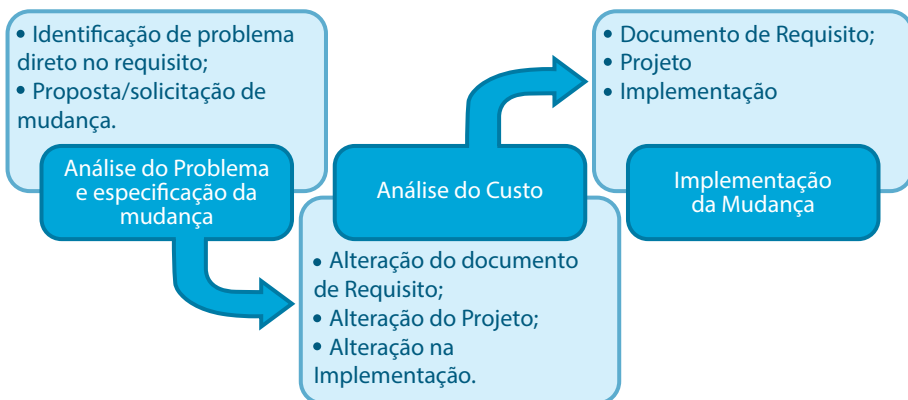


Figura 24: Processo de gerenciamento de mudança de requisitos

Fonte: adaptada de Sommerville (2008).



SAIBA MAIS

Plano de gerenciamento dos requisitos

Segundo o Guia PMBOK®, o Plano de Gerenciamento dos Requisitos documenta como eles serão analisados, documentados e gerenciados do início ao fim do projeto. Este Plano faz parte ou é um plano auxiliar do plano de gerenciamento do projeto. Trata-se de uma *baseline* que funciona como guia para todos os projetos de uma empresa/equipe, assim, todos os projetos seguirão um padrão pré-definido, garantindo um repositório único e uma fonte de pesquisa para os gerentes de projeto e desenvolvedores. O site Escritório de Projetos¹ oferece *templates* dos principais documentos utilizados para a gestão de projetos.

Visite também a página oficial do PMI e aprenda mais sobre o guia de práticas de gerenciamento de projetos: <<http://www.pmi.org/PMBOK-Guide-and-Standards.aspx>>. Acesso em: 8 jul. 2015.

Fonte: o autor.

O processo de gerenciamento de mudanças poderá incluir as seguintes atividades:

- Solicitação de mudanças: executado a partir de formulário próprio que deve ser aceito pela equipe de desenvolvimento e cliente.
- Análise do impacto e custo da mudança: estudo efetuado a partir do levantamento de complexidade, conforme informações de rastreamento, tempo de execução, influencia no prazo final, custo de implementação, custo de teste.
- Responsabilidades: levantamento das responsabilidades pela solicitação de mudança.
- Registros: ferramenta CASE para o gerenciamento do projeto.

REFLITA



Na academia, um processo de desenvolvimento de software é dividido em várias disciplinas específicas, isso para facilitar a didática e a organização dos assuntos. Você consegue estabelecer as relações entre elas quando carregamos nossos conhecimentos para a aplicação prática?

Fonte: o autor.

CONSIDERAÇÕES FINAIS

Você pode considerar clichê ou, até mesmo, jargão da área, e sei que já deve ter ouvido essa afirmação inúmeras vezes durante sua formação, mas, depois de tudo que estudamos até agora, é o momento de, realmente, entender o que ela quer dizer: os requisitos estão para o desenvolvimento de software como os alicerces estão para uma construção civil. Ficou claro que requisitos devem ser levantados com critério e documentados com maestria, mais que isso, devem estar acessíveis a todos os envolvidos no processo de desenvolvimento do software.

Aprendemos que os requisitos mudam, isso é inerente à necessidade de evolução dos sistemas computacionais, para que se mantenham atualizados e úteis, por isso, o gerenciamento de requisitos deve se ocupar de controlar essas mudanças de forma sistemática e eficiente. Com isso, é possível perceber que conversar com os *stakeholders*, ouvir as suas expectativas e necessidades em relação ao sistema, e mais, mantê-los envolvidos, próximos do processo de gestão de requisitos é fundamental para o sucesso de um projeto de desenvolvimento de software.

Mudanças, se não partem de problemas encontrados durante a implementação, surgem de novos entendimentos ou novas necessidades. Em ambos os casos, são os *stakeholders* a chave para o levantamento do impacto que a mudança irá causar no projeto e para a validação da mudança no requisito. Buscando aproveitar esse recurso, as metodologias ágeis baseiam-se no princípio que os requisitos do software serão elaborados ao longo do seu desenvolvimento, e não em uma etapa específica, para os agilistas, os requisitos não precisam ter toda a informação neles desde o começo, apenas informação o suficiente. Assim, é inerente a

essa abordagem que os requisitos irão evoluir, mudar e até falhar. Dessa forma, sua elaboração contínua está diretamente integrada ao gerenciamento e controle de mudanças, apoiando diretamente o desenvolvimento.

Finalizando a unidade, foi observado que ferramentas CASE podem facilitar a alcançar os resultados esperados no processo de gerência de requisitos, auxiliando na automatização do planejamento e no registro e controle dos relacionamentos dos requisitos.

ATIVIDADES



1. O surgimento de novos requisitos ou a necessidade de alterar um já existente acontece durante o processo de desenvolvimento e, até mesmo, depois que o sistema tiver em operação, isso se chama evolução de sistemas. Sobre a gestão de controle de mudanças de requisitos, assinale a alternativa correta:
 - a. São rotinas para identificar, controlar e rastrear requisitos e modificações de requisitos em qualquer época, à medida que o projeto prossegue.
 - b. Gestão de mudanças consiste em construir um modelo técnico refinado de funções, características e restrições do software.
 - c. Para o controle de mudanças, basta negociar com os clientes os conflitos de prioridade de requisitos e identificar e analisar os riscos associados a cada requisito.
 - d. O controle de mudanças tem como prioridade avaliar os requisitos quanto à qualidade, garantindo que ambiguidades, inconsistências, omissões e erros sejam detectados e corrigidos.
2. Rastreabilidade pode ser definida como o conjunto de ligações entre os requisitos, com as fontes dos requisitos, com eles mesmos e com outros artefatos. A condição de rastreamento deve ser atribuída ao requisito no momento do seu levantamento e ela deverá refletir a facilidade de se encontrar os requisitos relacionados. Indique a alternativa que identifica os três tipos de rastreabilidade de requisitos propostos por Sommerville (2008):
 - a. Por modelo de projeto, por prototipação, por stakeholders.
 - b. Por stakeholders, por entrevistas e por modelo de projetos.
 - c. Por *stakeholders*, por requisitos dependentes e por modelos de projeto.
 - d. Por modelo de projeto, por requisitos dependentes, por questionários.
3. Analise as alternativas sobre o gerenciamento de mudanças de requisitos e indique a alternativa correta:
 - I. O problema ou a proposta deve ser analisado e validado, isso porque uma mudança pode ser solicitada simplesmente pela falta de entendimento do solicitante da especificação do requisito.
 - II. No final da análise de custo, a mudança é sempre aceita.
 - III. Uma vez aprovada a mudança, o documento de requisitos e o projeto não necessitam ser alterados.
 - a. Apenas I está correta.
 - b. Apenas II e III estão corretas.
 - c. Apenas I e III estão corretas.
 - d. I, II e III estão corretas.

ATIVIDADES



4. Analise as afirmativas referentes à gestão de mudanças de requisitos e, na sequência, assinale a alternativa correta:
- I. A rastreabilidade de requisitos ocorre, apenas, na relação entre os requisitos propriamente ditos e os artefatos ou subprodutos de desenvolvimento gerados.
 - II. O levantamento do impacto de mudança de um requisito faz com que seja necessário retornar a sua fonte. Na validação dos requisitos, a equipe deve estar sempre atenta à rastreabilidade do requisito.
 - III. O gerenciamento de mudanças nos requisitos mantém informações de rastreabilidade a serem usadas para avaliar quais outros requisitos seriam afetados por uma mudança, bem como o impacto da mudança de requisitos no projeto e na implementação do sistema.
- a. I está correta.
 - b. II e III estão corretas.
 - c. I e III estão corretas.
 - d. I, II, III estão corretas.



DESENVOLVIMENTO DE SOFTWARE REQUER PROCESSO E GESTÃO

Software, de modo genérico, é uma entidade que se encontra em quase constante estado de mudança. Essas mudanças ocorrem por necessidade de corrigir erros existentes no software e/ou de adicionar novos recursos e funcionalidades. Igualmente, os sistemas computacionais (isto é, aqueles que têm software como um de seus principais elementos) também sofrem mudanças frequentemente. Essa necessidade evolutiva do sistema de software o torna predisposto a defeitos (isto é 'não confiável'), podendo resultar em atraso na entrega e em custos acima do estimado. Concomitante com esses fatos, o crescimento em tamanho e complexidade dos sistemas de software exige que os profissionais da área raciocinem, projetem, codifiquem e se comuniquem por meio de componentes (de software) e qualquer concepção ou solução de sistema passa então para o nível arquitetural. Nesse sentido, o objetivo deste artigo é discutir e explorar o fato de que software não é construído como uma TV ou geladeira, mas desenvolvido. E, o desenvolvimento de software

requer processo (de desenvolvimento) e gestão (do projeto de desenvolvimento).

Dentro do contexto discutido acima, um aspecto importante no desenvolvimento de um sistema de software é o contínuo feedback durante o processo. A importância de ter um feedback (resposta e comentários) do cliente mais cedo no desenvolvimento implica na necessidade de fazer um protótipo. Além disso, a necessidade de melhor planejar o desenvolvimento, fazendo um balanceamento entre custos e benefícios. Isto requer uma avaliação preliminar se é viável ou não desenvolver o software desejado pelo cliente. Como consequência, você pode perceber que duas atividades importantes são planejamento e análise de riscos, que procuram exatamente responder a questão: é viável desenvolver esse software? Note que tudo isso visa minimizar custos e assegurar que o desenvolvimento irá ocorrer da forma como planejada e dentro dos prazos propostos. Agora, você poderia ainda questionar: Por que tudo isso?

Fonte: Filho (2011, online).





LIVRO

Análise e Gestão de Requisitos de Software: Onde Nascem os Sistemas

Felipe Nery Machado

Editora: Erica

Sinopse: Apresenta os conceitos da análise e gestão de requisitos, com exemplos sobre variações e técnicas de levantamento e análise. Descreve os Modelos de Capacidade e Maturidade (CMMI), Processo Unificado da Rational (RUP), métodos ágeis, o funcionamento do SCRUM, detalha a engenharia de requisitos e a gerência de escopo de projetos. Aborda as mais diversas técnicas que podem ser utilizadas, com destaques para estudo etnográfico, entrevistas, associação entre requisitos e modelo de negócios, especificação de requisitos com casos de uso e user stories como elemento de levantamento de requisitos.



REQUISITOS NAS METODOLOGIAS ÁGEIS



Objetivos de Aprendizagem

- Oferecer, ao aluno, um panorama geral das metodologias ágeis.
- Apresentar como as metodologias ágeis tratam os requisitos de software.

Plano de Estudo

A seguir, apresentam-se os tópicos que você estudará nesta unidade:

- Requisitos nas metodologias ágeis
- Manifesto ágil
- Scrum
- Especificação de requisitos ágeis

INTRODUÇÃO

Olá, aluno(a)! Esta é a última unidade de estudos, encerraremos nossa disciplina falando sobre a especificação de requisitos ágeis. Para isso, abordaremos, rapidamente, os princípios do manifesto ágil de desenvolvimento de software e analisaremos o *framework Scrum*, e como fica a especificação de requisitos em um processo ágil.

A gerência de requisitos envolve todos os processos que cooperam para a geração de um Documento de Requisitos e para a sua manutenção durante todo ciclo de vida do projeto de desenvolvimento de software. A evolução é considerada uma das questões mais críticas no desenvolvimento de sistemas tradicionais, pois provoca mudanças, e o controle dessas é complexo.

Por outro lado, temos os métodos ágeis inseridos em ambientes dinâmicos, em que os requisitos estão em constante modificação e são construídos durante o desenvolvimento do software. Esta unidade apresenta o manifesto ágil de desenvolvimento de software, um documento divulgado em 2001, que declarou uma nova filosofia para os projetos de desenvolvimento de software.

Desse manifesto, várias metodologias e novos processos de desenvolvimento de software foram organizados para atender aos seus doze princípios. Os processos ágeis não desaprovam e não revogam a utilização de ferramentas e documentação, mas declaram que mais importantes do que isso são os valores ágeis: os indivíduos e interações entre eles, software em funcionamento, colaboração com o cliente e responder a mudanças.

Vimos que a especificação de um requisito está diretamente ligada à qualidade de software. Nos processos ágeis, temos uma visão diferente de requisitos, aqui, ele é um objetivo, no lugar de uma solicitação sobre o que deve ser feito, e sua especificação ocorre durante todo o ciclo de vida do processo de desenvolvimento do software. Vamos lá!

REQUISITOS NAS METODOLOGIAS ÁGEIS

Sistemas computacionais cada vez mais complexos e abrangentes dificultam ou, até mesmo arrisco dizer, impossibilitam a definição completa dos requisitos antes do início do projeto. A evolução da tecnologia tanto de software quanto de hardware está muito rápida, o que obriga os engenheiros de software a serem ágeis na tomada de decisões.

Os agilistas, como são chamados os profissionais que apoiam os processos ágeis, praticam desenvolvimento incremental, ou seja, pacotes de trabalhos são entregues em pequenos períodos de tempo. No processo ágil, não se faz um plano completo com tudo que se deve desenvolver antes do início da implementação, são desenvolvidos incrementos, pedaços do produto que são produzidos aos poucos e entregues constantemente.

É fácil encaixar essa abordagem quando se observa os sistemas já implantados e em fase de manutenção, em que as solicitações de mudanças passam a ser o carro chefe da equipe de desenvolvimento, mas como isso funcionaria para um novo produto de software? Sommerville (2008) explica que, no momento da elaboração do projeto, devem se estabelecer os requisitos para que os incrementos iniciais do sistema considerem as funcionalidades de alta prioridade. Com a entrega dessas funções, o cliente pode observar o requisito na prática e especificar as mudanças para serem consideradas e incorporadas nas próximas entregas.

Um estudo comparando equipes ágeis e equipes tradicionais, desenvolvido por Crispin & Gregory (2009), apresentou como resultado:

Times ágeis:

- Há um entendimento detalhado dos requisitos devido à proximidade do cliente e especialistas no negócio.
- O foco está no valor a ser entregue - equipe ágil é focada em entregar um produto de alta qualidade que forneça valor de negócio.

Times tradicionais:

- Há uma ilusão de controle, não sabem como o código é escrito, nem se os programadores executam testes.

- Entrega-se o que foi acordado nos requisitos.

Não afirmo aqui que uma metodologia é melhor que outra, a decisão pela tradicional ou ágil deve ser feita pela organização, considerando suas características (perfil organizacional e cultura) e suas necessidades.

Muitos benefícios podem ser trazidos para uma organização, já em operação nos moldes tradicionais, que decide adotar uma metodologia ágil, mas, para isso, em um primeiro momento, ela irá evidenciar as fraquezas de equipe e da própria organização, para que se possa atuar sobre elas, esse é um momento de estresse. Deve-se estar preparado para uma mudança de paradigmas e de comportamento.

MANIFESTO ÁGIL

Qual o propósito do manifesto ágil? Pressman (2010, p. 58) nos apresenta uma discussão útil:

Métodos Ágeis foram desenvolvidos em um esforço para vencer as fraquezas percebidas e reais da engenharia de software convencional. O desenvolvimento ágil pode fornecer importantes benefícios, mas ele não é aplicável a todos os projetos, produtos, pessoas e situações. Ele também não é contrário à sólida prática de engenharia de software e pode ser aplicado como uma filosofia prevalecente a todo trabalho de software.

Em 2001, um grupo de dezessete profissionais, entre eles, desenvolvedores, produtores e consultores de software, assinaram o Manifesto para o Desenvolvimento Ágil de Software. Nesse ataque construtivo à velha guarda, declararam:

Estamos descobrindo melhores modos de desenvolvimento de software fazendo-o e ajudando outros a fazê-lo. Por meio desse trabalho passamos a valorizar:

- Indivíduos e interações em vez de processos e ferramentas.
- Softwares funcionando em vez de documentação abrangente.
- Colaboração do cliente em vez de negociação de contratos.
- Respostas a modificações em vez de seguir um plano (O MANIFESTO..., online).

Pressman (2010) nos ajuda a compreender os propósitos da Aliança Ágil com seu manifesto dizendo que, se os modelos de processo devem funcionar, devem, então, ser caracterizados de uma forma mais realística, com mecanismos que mostrem tolerância com as pessoas envolvidas.

O principal foco da agilidade é o controle efetivo das mudanças, mas o manifesto também propõe uma nova filosofia de trabalho: que encoraja a comunicação, não só entre os membros da equipe, mas, principalmente, entre a equipe de desenvolvimento e os *stakeholders* envolvidos, que enfatiza a entrega rápida do produto de software e que o plano de projeto deve ser flexível. Pressman (2010) afirma que a agilidade pode ser aplicada em qualquer processo de software.



REFLITA

A agilidade é dinâmica, específica em conteúdo, agressiva no acolhimento de modificações e orientada ao crescimento.

Fonte: Pressman (2010).

Os 12 princípios do desenvolvimento ágil de software:

1. Nossa maior prioridade é satisfazer o cliente por meio da entrega contínua e adiantada de software com valor agregado.
2. Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.
3. Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo.
4. Pessoas de negócio e desenvolvedores devem trabalhar, diariamente, em conjunto por todo o projeto.
5. Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.

6. O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é por meio de conversa face a face.
7. Software funcionando é a medida primária de progresso.
8. Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.
9. Contínua atenção à excelência técnica e bom design aumentam a agilidade.
10. Simplicidade - a arte de maximizar a quantidade de trabalho não realizada - é essencial.
11. As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis.
12. Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e, então, refina e ajusta seu comportamento de acordo.

SAIBA MAIS



Os modelos ágeis estão cada vez mais presentes no processo de desenvolvimento de softwares, o manifesto ágil é uma declaração de princípios que fundamentam o desenvolvimento ágil de software. Como vimos, possui princípios claros. Saiba mais sobre os membros que formam a Aliança Ágil e como se desenvolveu a concepção do Manifesto para o Desenvolvimento Ágil de Software, acessando o link disponível em: <<http://www.drdoobs.com/open-source/the-agile-manifesto/184414755?queryText=the+agile+manifesto>>. Acesso em: 05 maio 2015.

Fonte: o autor.

Existem, no mercado, várias metodologias ágeis, as mais conhecidas e aplicadas são: *Extreme Programming* (XP), *Scrum*, *Lean Development*, *Feature-Driven Development* (FDD), *Kanban*, RUP e *OpenUP*. Para os nossos estudos, utilizaremos o framework *Scrum*.

SCRUM

A gestão de projetos é uma área chave no desenvolvimento de software. Os processos de desenvolvimento de software que subsidiam a gestão de projetos podem dar mais garantias de que o produto a ser desenvolvido será entregue dentro do prazo, do custo e com qualidade. Por isso, no momento de se definir um processo de software, questões como essa devem ser consideradas. O *Scrum* é um processo empírico que oferece uma estrutura para gestão de projeto. A Figura 25 apresenta o processo do *Scrum*:

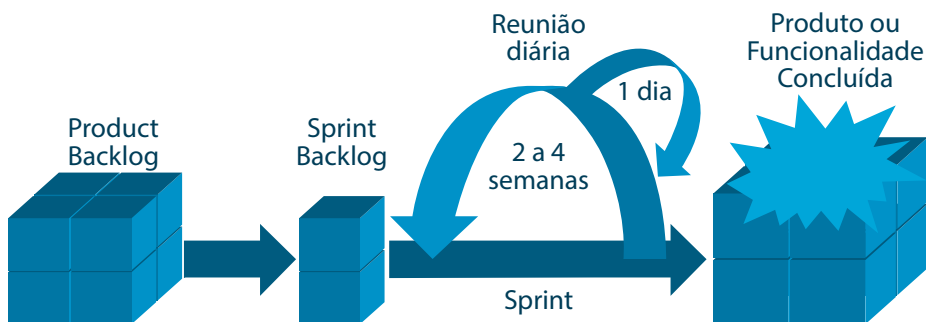


Figura 25: Processo do Scrum

Fonte: Vieira (2014, online).

O *Scrum* estipula papéis bem definidos e diversas etapas que devem ser cumpridas em prazos estipulados, visando entregar o produto de forma rápida e que, ao mesmo tempo, atenda às expectativas do cliente. O *Product Owner* representa os *stakeholders* e o negócio, o *Team* formado por poucas pessoas (recomenda-se sete, no máximo, e nunca menos de três) e multidisciplinares é uma das principais características do *Scrum*. O *ScrumMaster* atua como um líder de equipe, coordenando a equipe para que as metas sejam alcançadas.

Metodologias ágeis de desenvolvimento de software são iterativas, ou seja, o trabalho é dividido em iterações, que, no *Scrum*, são chamadas de *Sprints* e, geralmente, duram de 2 a 4 semanas.

A seguir, veremos as etapas do processo:

- O *Sprint* representa um tempo definido dentro do qual um conjunto de atividades deve ser executado. No início de cada *Sprint*, faz-se um *Sprint Planning Meeting* (reunião de planejamento), em que o *Product Owner*

(geralmente, o representante do *stakeholder* - cliente) prioriza todos os itens do *Product Backlog* para que a equipe selecione as funcionalidades que será capaz de implementar durante o *Sprint* que se inicia.

- *Product Sprint Backlog* (Backlog do Produto). As funcionalidades a serem implementadas no projeto são mantidas em uma lista que é conhecida como *Product Backlog*. As funcionalidades alocadas em um *Sprint* são transferidas do *Product Backlog* para o *Sprint Backlog*.
- *Kanban* (Quadro de Trabalho). O time mantém um quadro de trabalho, em que organiza as atividades dos itens de *Backlog da Sprint*, classificando-as, geralmente, com os seguintes status: a fazer, em andamento, Em testes e concluído. A Figura 26 demonstra um quadro de atividades:

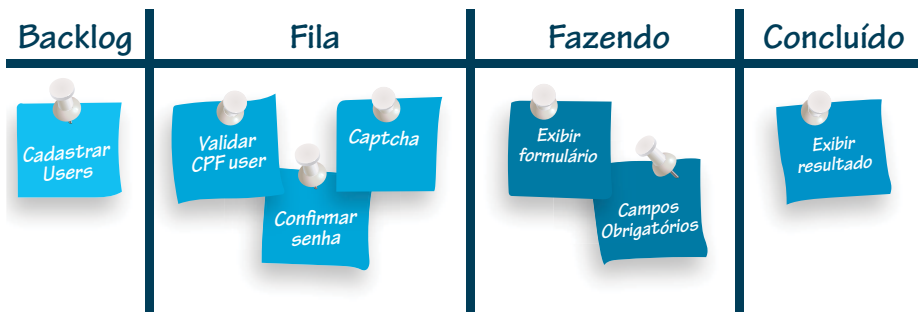


Figura 26: Modelo de organização do quadro de atividades no Scrum

Fonte: Metodologias ágeis (online).

- *Daily Scrum*: com o objetivo de manter a equipe focada e dirimir impedimentos mais rapidamente, diariamente, a equipe faz uma breve reunião de, no máximo, 15 minutos com todos os participantes em pé, chamada *Daily Scrum*. Na *Daily*, cada integrante diz o que fez no dia anterior, o que pretende fazer no dia que se inicia e se existe algum impedimento que está atrapalhando o seu trabalho.
- *Sprint Review Meeting*: reunião para apresentação do que foi implementado durante o *Sprint* e para a realização de uma *Sprint Retrospective* para identificar o que funcionou bem e o que pode ser melhorado para o próximo *Sprint*. Essa mesma reunião pode ser utilizada para o planejamento do próximo *Sprint*.
- *Burn Down Chart*. O *Burndown* é um simples gráfico, com dois eixos X e Y, em que o eixo X representa o número de tarefas existentes no *Sprint* e

o eixo Y os dias que representam o tamanho do *Sprint*. A Figura 27 mostra um gráfico *BurnDown*.

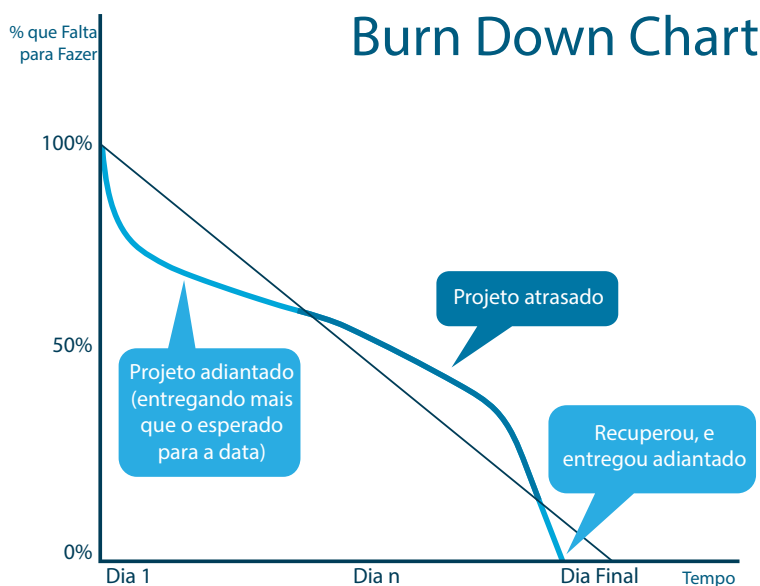


Figura 27: Exemplo de gráfico *BurnDown* para o acompanhamento das tarefas do Sprint

Fonte: Metodologias ágeis (online).



SAIBA MAIS

A escolha de uma metodologia de desenvolvimento é uma das etapas no início do desenvolvimento de um novo software, assim como a escolha da linguagem de programação e o banco de dados que será utilizado. Saber identificar a melhor metodologia a ser utilizada para o software em questão repercute, sem dúvida, em benefícios ao longo de todo o processo de fabricação de um software. Várias questões devem ser analisadas e levantadas no momento da escolha, como: a quantidade de integrantes da equipe de desenvolvimento, o tamanho do software que será desenvolvido, o conhecimento da metodologia por parte dos integrantes e o prazo de entrega.

Aprofunde seus conhecimentos estudando a implantação de um modelo ágil, o Scrum, no desenvolvimento de um software.

Fonte: Zenaro (2012, online).

Alguns estudos apontam fragilidades no Scrum, como a dificuldade em se capturar requisitos, gerenciá-los e de tratar as dependências. Uma maneira de superar essas fragilidades é utilizar o Scrum em conjunto com alguma ferramenta CASE para gestão de projetos, que possibilite o registro e a rastreabilidade dos *user stories*.

ESPECIFICAÇÃO DE REQUISITOS ÁGEIS

Em processos ágeis, a especificação dos requisitos se dá por meio dos *user stories*. Como estudamos na unidade II, trata-se de uma técnica de especificação de requisitos.

O objetivo do *user stories* é garantir à equipe capacidade de responder rapidamente às necessidades do usuário. O *user stories* cria menos sobrecarga de documentação e mostra, de forma rápida, a evolução das necessidades do cliente ou a descoberta de novos requisitos no decorrer do desenvolvimento.

O levantamento e gerenciamento dos requisitos nos processos ágeis acontece durante todo o ciclo de vida de desenvolvimento, e conta com a participação efetiva do *Stakeholder*, promovendo esclarecimentos imediatos no caso de dúvidas.

Relembrando o estudado na unidade II, os *user stories* são comumente escritos em cartões utilizados. Segundo Longo e Silva (2014), o cartão é a parte visível dos *user stories*, mas não é a mais importante. Ali, fica registrado o texto de um requisito, mas os seus detalhes são discutidos nas reuniões entre a equipe e o *stakeholder* para, em seguida, o requisito, aqui chamado de história, ser registrado e verificado por meio da confirmação. Você pode rever a forma recomendada de escrever um *user stories* registrada na unidade II.

Longo & Silva (2014) dizem que uma das principais diferenças das histórias de usuário para as declarações tradicionais de requisitos está na comunicação verbal. A linguagem escrita é, muitas vezes, imprecisa e não garante que um cliente e/ou desenvolvedor irá interpretar uma declaração da mesma forma. A garantia da compreensão está na conversa com o cliente. Algumas

vantagens dos *user stories* foram levantadas por Longo e Silva (2014) e são relacionadas a seguir:

- Podem ser utilizadas prontamente no planejamento do projeto.
- São escritas de modo a permitir que estimativas de complexidade, custo e prazo sejam especificadas a partir delas.
- São implementadas por completo em determinada iteração de um projeto ágil.
- Incentivam a equipe a adiar detalhes de coleta. Uma história inicial pode ser escrita e, quando necessários mais detalhes, podem ser substituídas com histórias mais detalhadas.
- Iniciar a codificação muito mais cedo. Uma equipe pode muito rapidamente, escrever algumas histórias para dar-lhes uma sensação geral do sistema.
- Facilita a priorização dos requisitos. Se considerar milhares ou dezenas de milhares de declarações do tipo “o sistema deve...” em uma especificação de requisitos de software (e as relações entre eles), pode-se encontrar dificuldades em priorizá-los.

É importante registrar que um *user story* não é um requisito. Segundo Longo & Silva (2014), enquanto os requisitos sugerem o que deve ser feito, *user stories* focam nos objetivos, e isso torna a visão do produto completamente diferente.

A diferença final entre as histórias de usuário e o padrão IEEE 830 de requisitos, como afirma Longo & Silva (2014), é em relação ao custo do requisito que não é visível até que todos os requisitos sejam escritos. O cenário típico é que um ou mais analistas passam dois ou três meses escrevendo um documento de requisitos. Esse documento é, então, entregue aos programadores, que detectam que o projeto levará 24 meses, ao invés dos seis meses previstos.

A especificação de um requisito em um *user stories* deve seguir as mesmas

SAIBA MAIS



As exigências do mercado mudaram e, com isso, a engenharia de requisitos teve que se adequar a elas. Os métodos ágeis alteraram processos da engenharia de requisitos tradicional e os adaptaram para seus interesses. Tradicionalmente, engenharia de requisitos é fortemente baseada em documentação para compartilhar conhecimento, enquanto nas metodologias ágeis são focados em colaboração entre desenvolvedores e clientes para garantir objetivos similares.

Leia mais em: Desenvolvimento Ágil: análise sobre requisitos tradicionais, disponível em: <<http://www.devmedia.com.br/desenvolvimento-agil-analise-sobre-requisitos-tradicionais/30202#ixzz3Zlm1vbYk>>. Acesso em: 8 jul. 2015.

Fonte: o autor.

recomendações de especificação das metodologias tradicionais. Para auxiliar essa atividade, uma vez que o próprio cliente pode ser designado para descrever os *user stories*, a IBM desenvolveu uma técnica chamada INVEST: Independente, Negociável, Valiosa, Estimável, Pequena (Small) e Testável.

- Independente: um *user story* deve ser único, tudo o que necessita ser escrito deve estar contido ali, não havendo a necessidade de outros itens para complementar o seu significado.
- Negociável: o *user story* nunca deve ser fechado, dessa forma, na necessidade de acrescentar ou retirar alguma informação, o seu autor pode intervir sem maiores problemas.
- Valiosa: não colocar informações desnecessárias para não poluir o cartão. Ao realizar essa seleção, *user stories* se tornam mais eficazes.
- Estimável: descrever o item de forma a possibilitar estimativas de complexidade, a fim de determinar o tempo necessário para seu desenvolvimento. Só dessa forma é que a equipe poderá decidir quantos *user stories* serão assumidos para a implementação no tempo determinado para a iteração.
- *Small* (pequeno): a descrição no *user story* deve ser breve.

- Testável: os itens que compõem aquele *user story* devem sempre possibilitar testes.

Os requisitos ágeis são testados a cada iteração dentro do processo ágil de desenvolvimento. Testes de aceitação são criados a partir de critérios de aceitação. Segundo Longo & Silva (2014), os critérios de aceitação definem os limites de *user stories* e são usados para confirmar quando uma história é concluída e se está funcionando conforme o esperado.

Os critérios devem ser escritos em linguagem simples, assim como o *user story*. Quando a equipe de desenvolvimento concluir o *user story*, deverá demonstrar a funcionalidade para o cliente, mostrando como cada critério solicitado foi satisfeito. Um *user story* não é considerado completo até que seu teste de aceitação tenha sido aprovado.

CONSIDERAÇÕES FINAIS

Enfim, chegamos ao final de nossa disciplina, nesta última unidade, vimos que a adoção dos processos na engenharia de requisitos é fundamental para sucesso de qualquer tipo de projeto, os processos tradicionais exigem mais tempo nas etapas iniciais de planejamento e levantamento de requisitos. Os métodos ágeis foram propostos para solucionar a burocracia imposta pelos modelos tradicionais, mas não extinguem por completo os documentos a serem gerados na fase de planejamento e elicitação dos requisitos, a proposta é reduzi-los e torná-los mais práticos, a fim de diminuir o esforço empregado nessas etapas e impulsionar a equipe de desenvolvimento para a implementação em menos tempo.

A entrega de valor é objetivo nos métodos ágeis e se inicia na definição dos requisitos. Os stakeholders participam ativamente durante todo o ciclo de vida do projeto e os requisitos são implementados conforme a prioridade estipulada por eles.

A documentação do processo tradicional não é produzida da mesma maneira na metodologia ágil, ela apresenta os requisitos de maneira executável e implementável, priorizando uma documentação enxuta e eficaz.

O engenheiro de requisitos na metodologia ágil pode assumir o papel de *product owner* e tem como objetivo escrever somente o suficiente. Foi apresentada uma técnica para garantir que a descrição registrada no *user stories* seja curta, clara e objetiva. O que foi especificado no *user stories* deve ser uma unidade de desenvolvimento única e passível de teste, porque um *user stories* somente é considerado finalizado depois de ter seu teste validado.

Finalmente, percebemos que a engenharia de requisitos deve se fazer presente nos processos ágeis, para garantir aos requisitos a capacidade de relacionamento e de rastreamento.

ATIVIDADES



1. O termo *Sprint* aplica-se ao modelo ágil do processo de engenharia de software denominado:
 - a. XP.
 - b. DAS.
 - c. DSDM.
 - d. Scrum.
 - e. Crystal.

2. Um dos principais conceitos do Scrum é a implantação de um controle descentralizado, capaz de lidar de forma mais eficiente em ambientes pouco previsíveis. Quais são os papéis definidos pelo framework para gerenciar essas questões?
 - a. Product Owner, Product Backlog e Planning Meeting.
 - b. Product Owner, Scrum Master e Team.
 - c. Product Owner, Scrum Team e Scrum Master.
 - d. Sprint, Scrum Master e Planning Meeting

3. Sobre os user stories, é correto afirmar que:
 - a. O user story tem a intensão real de fornecer à equipe uma capacidade para responder rapidamente o que o usuário quer e precisa.
 - b. O user story cria sobrecarga de documentação e mostra a evolução das necessidades do cliente.
 - c. Com os user stories, o levantamento e gerenciamento dos requisitos nos processos ágeis acontecem durante toda a fase de elicitação de requisitos.
 - d. Os user stories são comumente escritos em planilhas utilizadas para detalhar em uma linguagem técnica os requisitos.



A HISTÓRIA DA TAHINI-TAHINI: MELHORIA DE PROCESSOS DE SOFTWARE COM MÉTODOS ÁGEIS E MODELO MPS

A discussão sobre a possibilidade ou não da utilização de métodos ágeis em conjunto com modelos de maturidades em processos de software é frequente e atual. Alguns consideram que as exigências dos modelos de maturidade não podem ser implementadas em organizações com desenvolvimento ágil. Outros consideram que a implementação destes modelos irá ferir a agilidade do desenvolvimento. Esta incompatibilidade é discutida, portanto, por defensores do uso de métodos ágeis e por defensores dos modelos de maturidade.

Neste contexto se situa o livro “A História da Tahini-Tahini: Melhoria de Processos de Software com Métodos Ágeis e Modelo MPS” de Jorge Boria, Viviana Rubinstein e Andrés Rubinstein.

O livro teve origem em uma chamada realizada em dezembro de 2011 pela Secretaria de Política de Informática – SEPIN, do Ministério de Ciência, Tecnologia e Inovação – MCTI, responsável pela condução do Programa Brasileiro de Qualidade e Produtividade em Software – PBQP Software, para o Ciclo 2012 do Programa “Série de Livros do PBQP Software”. Entre vários concorrentes foi o texto selecionado para publicação.

É um livro técnico, mas fascinante e de leitura muito agradável. Por vezes nos leva a rir, tamanho o bom humor dos autores ao tratar o tema. Certamente será um caso de

sucesso, nesta excelente iniciativa do MCTI.

Nelson Franco, gerente de Qualidade da Softex, espera que esta publicação “motive as empresas a investirem na melhoria contínua de seus processos de software e de sua competitividade tanto no mercado brasileiro como no exterior”. Kival Weber, coordenador executivo do programa MPS.BR, lembra que o modelo MPS-SW (Software) está sendo implementado desde 2004 no Brasil e, de março de 2010 a março de 2014, na Colômbia, Peru e México no âmbito do Projeto BID/FUMIN ‘Rede Latino Americana da Indústria de Software (RELAIS)’. “Este ano, nossa meta é superar a marca das 500 avaliações MPS, dando também sequência ao processo de internacionalização do modelo MPS”, complementa. Coordenado pela Softex, o programa mobilizador MPS.BR – Melhoria de Processo do Software Brasileiro já totalizou 491 Avaliações MPS-SW (Software) em empresas de todas as regiões do país, muitas das quais utilizam métodos ágeis, em um intervalo de tempo justo e a um custo módico. Desse total, 70% são empresas de micro, pequeno e médio porte e 30% são grandes organizações públicas e privadas. A iniciativa conta com investimentos das empresas e apoio do Ministério da Ciência, Tecnologia e Inovação (MCTI), da Financiadora de Estudos e Projetos (FINEP), do Banco Interamericano de Desenvolvimento (BID/FUMIN) e do SEBRAE.

Fonte: Modelo MPS (2013, online).





LIVRO

Metodologias Ágeis: Engenharia de Software Sob Medida

Paulo Cesar de Macedo, José Henrique Teixeira de Carvalho Shrocco

Editora: Erica

Sinopse: Tendo em vista o crescente interesse pelo uso das metodologias ágeis, esta obra apresenta, de maneira comparativa e completa, as características, aplicações e exemplos de seus principais paradigmas, tais como Iconix, SCRUM, XP, FDD, DSDM, ASD e família de metodologias Crystal. Contempla também um capítulo sobre uma nova proposta de metodologia ágil, desenvolvida para ser utilizada em projetos acadêmicos. Destinada a estudantes e pesquisadores da área de computação e de gestão de projetos, mostra-se alinhada com as novas tendências mundiais, necessárias para atender aos requisitos da atual realidade de mercado de projetos de software.



CONCLUSÃO

Chegamos ao final de mais uma etapa! Juntamente com esta unidade, encerramos a disciplina de Engenharia de Requisitos. Espero que você tenha aprendido um pouco mais com os temas abordados neste material.

Tentamos estabelecer elementos práticos versus teóricos que contribuam com a sua formação. A engenharia de software exige que tenha um entendimento global de todos os seus processos, o que, didaticamente, não é possível e, por isso, seus processos são divididos em disciplinas específicas. No decorrer do seu curso, faça um esforço para compreender o funcionamento dos processos na prática, a fim de estabelecer um paralelo.

Definimos, durante nossos estudos, que engenharia de requisito pode ser definida como um conjunto das atividades envolvidas no levantamento, documentação e manutenção de um conjunto de requisitos para um sistema baseado em computador e que o processo de engenharia de requisitos apresenta dois tipos básicos de atividades: primeiro, a análise do problema, em que são aplicadas as técnicas “brainstorming”, entrevistas, etnografia, a fim de identificar as funções e as possíveis restrições sobre a solução do problema; segundo, a especificação do produto, escrever, negociar, validar e preparar o Documento de Requisitos que representa o comportamento esperado do produto.

Finalizamos falando sobre as metodologias ágeis de desenvolvimento de software, manifesto de profissionais da área contra prazos e a complexidade dos métodos tradicionais da engenharia de software. Nessa metodologia, a engenharia de requisitos fica em plano de fundo, uma vez que os requisitos são desenvolvidos à medida que o software é desenvolvido.

Esperamos ter alcançado o objetivo inicial, que era mostrar a importância da engenharia de requisitos. Desejamos que a utilização do que foi aqui apresentado te garanta sucesso e realização profissional. Se, de alguma forma, pudemos ajudar, estamos à disposição! Muito obrigada pela atenção! Até uma próxima!



REFERÊNCIAS

CALEGARO, B. Aula 9. Universidade Federal de Santa Catarina. Centro de Tecnologia. Disponível em: <<http://www-usr.inf.ufsm.br/~calegaro/ELC1069/Aula%2009.pdf>>. Acesso em: 8 jul. 2015.

CÉSAR, A. C. F. Qualidade de documento de requisitos, visando alguns padrões, normas e modelo. Monografia – Curso de pós-graduação em Ciência da Computação, Centro de Informática, Universidade Federal de Pernambuco, 2008. Disponível em: <www.cin.ufpe.br/.../monografica_ER_ANA_CRISTINA_MESTRADO.doc>. Acesso em: 25 abr. 2015.

CORAZZIN, E. Escopo do Projeto X Escopo do Produto. **Auctus Qualidade e Gestão.** Disponível em: <<http://www.auctus.com.br/escopo-do-projeto-x-escopo-do-produto/>>. Acesso em: 30 jun. 2015.

CRISPIN, L.; GREGORY, J. Agile Testing: A practical guide for testers and agile teams. Addison Wesley, 2009.

DAVID, L. Técnicas para reuniões em JAD (Joint Application Design). Disponível em: <<http://engenhariadesoftware.info/downloads/JAD.ppt>>. Acesso em: 03 set. 2012.

DEVMEDIA. Introdução a Requisito de Software. Disponível em: <<http://www.devmedia.com.br/introducao-a-requisitos-de-software/29580>>. Acesso em: 23 mar. 2015.

DICIONÁRIO do Aurélio. Significado de Processo. Disponível em: <<http://www.dicionarioaurelio.com/processo>>. Acesso em: 30 jun. 2015.

ENGENHARIA de requisitos: requisitos funcionais e não funcionais. Concurso de TI: dicas e materiais para concursos. Disponível em: <<http://concursosdeti.net/engenharia-de-requisitos-requisitos-funcionais-e-nao-funcionais/>>. Acesso em: 06 jul. 2015.

FAGUNDES, R. M. Engenharia de requisitos: Do perfil do analista de requisitos ao desenvolvimento de requisitos com UML® e RUP. Salvador: E-book, 2011. Disponível em: <https://books.google.com.br/books?id=i2pIBQAAQBAJ&pg=PA23&lpg=PA23&dq=na+pratica+como+documentar+requisitos&source=bl&ots=hV_Na4Fthg&sig=_wp2TErMZ-1FCqVa9_HrVDph22c&hl=p-t-BR&sa=X&ei=QfEhVd2xFsK0sAS1_4CwCQ&ved=0CE4Q6AEwCQ#v=onepage&q=na%20pratica%20como%20documentar%20requisitos&f=false>. Acesso em: 21 mar. 2015.

FARIA, C. Desdobramento da função qualidade (QFD). **Infoescola.** Disponível em: <http://www.infoescola.com/administracao/_desdobramento-da-funcao-qualidade-qfd/>. Acesso em: 02 jul. 2015.

FILHO, A. M. da S. Desenvolvimento de Software requer Processo e Gestão. **Revista Espaço Acadêmico**, n. 23, ago. 2011. Disponível em: <<http://www.periodicos.uem.br/ojs/index.php/EspacoAcademico/article/viewFile/14312/7593>>. Acesso em: 01 maio 2015.



REFERÊNCIAS

FILHO, A. M. da S. Documento de requisitos. **Engenharia de Software Magazine**, a. 1, 10. ed. Disponível em: <http://academico.ifrr.edu.br/uploads/MATERIAIS_AULAS/7243-DOCUMENTO_DE_REQUISITOS.pdf>. Acesso em: 22 jun. 2015.

FOURNIER, R. **Guia prático para o desenvolvimento e manutenção de sistemas estruturados**. São Paulo: Makron Books, 1994.

GALEOTE, S. Levantamento e análise de requisitos: uma visão pragmática sobre ferramentas. **Qualidade de Software**. Disponível em: <[http://www.galeote.com.br/blog/2012/05/levantamento-e-anlise-de-requisitos-uma-viso-pragmtica-sobre-fer-rametas/](http://www.galeote.com.br/blog/2012/05/levantamento-e-analise-de-requisitos-uma-viso-pragmtica-sobre-fer-rametas/)>. Acesso em: 6 jul. 2015.

GENVIGIR, E. C. **Um modelo para rastreabilidade de Requisitos de software baseado em Generalização de elos e atributos**. 2001. 202f. Tese (Doutorado) – Pós-Graduação em Computação Aplicada, INPE, São José dos Campos. 2009. Disponível em: <<http://mtc-m18.sid.inpe.br/col/sid.inpe.br/mtc-m18@80/2009/03.02.14.17/doc/publicacao.pdf?languagebutton=pt-BR>>. Acesso em: 01 maio. 2015.

GUIA PMBOK® - **Project Management Body of Knowledge** – Experiência e Conhecimento em Gerenciamento de Projetos. Disponível em <<http://pmkb.com.br/artigo/analise-e-classificacao-dos-stakeholders-para-gestao-de-projetos/>>. Acesso em: 25 mar. 2015.

IBM Rational Software. Disponível em: <<http://www.ibm.com/software/rational>>. Acesso em: 01 maio 2015.

IEEE Standard Glossary of Software Engineering Terminology. **IEEE Std 610 12-1990**, december, 1990, p. 67.

INSTITUTE of Electrical and Electronics Engineers. Recommended practice for software requirements specifications. **Revision 830**. New York: IEEE: 1998.

KAREL, R. Menos coleta de requisitos, mais exploração de dados. **Informatica**. Disponível em: <<http://international.informatica.com/br/potential-at-work/information-leaders/less-requirements-gathering-more-data-exploration.aspx>>. Acesso em: 22 jun. 2015.

KARLSSON, J.; RYAN, K. A **Cost-Value Approach for Prioritizing Requirements**, IEEE Software, Sept./Oct. 1997, p. 67-74.

KOTONYA, G.; SOMMERVILLE, I. **Requirements Engineering: Process and Techniques**. John Wiley and Sons: 1998.

LONGO, H. E. R.; SILVA, M. P. da. A Utilização de Histórias de Usuários no Levantamento de Requisitos Ágeis. **Int. J. Knowl. Eng. Manag**, Florianópolis, v.3, n.6, p. 1-30, jul/nov, 2014.

MEDEIROS, H. Princípios da Engenharia de Software. **Devmedia**. Disponível em <<http://www.devmedia.com.br/principios-da-engenharia-de-software/29630>>. Acesso em: 29 mar. 2015.



REFERÊNCIAS

METODOLOGIAS ágeis – SCRUM. **BRQ**. Disponível em: <<http://www.brq.com/metodologias-ageis/>>. Acesso em: 06 jul. 2015.

MICHAELIS. Dicionário online. Disponível em: <<http://michaelis.uol.com.br/>>. Acesso em: 8 jul. 2015.

MODELO MPS e Métodos ágeis é tema de livro. **Acate**. Disponível em: <<https://www.acate.com.br/node/45474>>. Acesso em: 24 jun. 2015.

O MANIFESTO ÁGIL. **Agile Alliance**. Disponível em: <<http://www.agilealliance.org/pt/o-manifesto/>>. Acesso em: 8 jul. 2015.

OLIVEIRA, C. Utilização de checklist para validação de requisitos de software. **iMaster.com**, 2014. Disponível em: <<http://imasters.com.br/desenvolvimento/software/utilizacao-de-checklist-para-validacao-de-requisitos-de-software/>>. Acesso em: 01 maio. 2015.

PMKB. **Análise e Classificação dos stakeholders para gestão de projetos**. Disponível em: <<http://pmkb.com.br/artigo/analise-e-classificacao-dos-stakeholders-para-gestao-de-projetos/>>. Acesso em: 30 jun. 2015.

PMSURVEY.ORG. **A global initiative of PMI chapters**. Disponível em: <<http://pmsurvey.org/>>. Acesso em: 8 jul. 2015.

PRESMAN, R. S. **Engenharia de Software**. 6. ed. Porto Alegre: McGrawHill, 2010.

PRIBERAM, Dicionário da Língua Portuguesa. Eliciar. Disponível em: <<http://www.priberam.pt/dlpo/eliciar>>. Acesso em: 05 maio. 2015.

PRIMO, G. User Storie – o que são? Como usar? **Blog ScrumHalf**. Disponível em: <<http://blog.myscrumhalf.com/2011/10/user-stories-o-que-sao-como-usar/>>. Acesso em: 02 jul. 2015.

RAMIRES, J. J. da C. V. **Negociação de requisitos no processo de desenvolvimento de software**. Dissertação (Mestrado em informática) – Faculdade de Ciências, Universidade de Lisboa, 2004. Disponível em: <<http://www.di.fc.ul.pt/~paa/reports/T13.pdf>>. Acesso em: 28 abr. 2015.

RIBEIRO, F.G.; SOUZA, R. L. da S. A importância da engenharia de requisitos para o ciclo de desenvolvimento de software de tempo real. In: **MOSTRA CIENTÍFICA DO CENTRO DE ENSINO SUPERIOR DE CATALÃO**, 10., 2012, Catalão. **Resumo...** Catalão, 2012. Disponível em: <http://www.cesuc.br/_xmostracientifica/artigos/artigo_5.pdf>. Acesso em: 05 maio 2015.

SAYÃO, M.; BREITMAN, K. K. **Gerência de Requisitos**. Faculdade de Informática da PUC-RS e DI/PUC-Rio. Disponível em: <http://www-di.inf.puc-rio.br/~karin/TELECOM/index_files/gerencia_req.pdf>. Acesso em: 07 maio. 2015.

SAYÃO, S.; STAA, A. v.; LEITE, J. C. S. do P. **Qualidade em requisito**. Monografia em Ciência da Computação – Departamento de Informática – Pontifícia Universidade



REFERÊNCIAS

Católica, Rio de Janeiro. Disponível em: <http://www.dbd.puc-rio.br/depto_informatica/03_47_sayao.pdf>. Acesso em: 8 jul. 2015.

SIGNIFICADOS.com.br. **Significado de Brainstorming**. Disponível em: <<http://www.significados.com.br/brainstorming/>>. Acesso em: 06 jul. 2015.

SILVA, M. A. A importância do levantamento de requisitos no sucesso dos projetos de software. **Linha de Código**. Disponível em: <<http://www.linhadecodigo.com.br/artigo/1685/a-importancia-do-levantamento-de-requisitos-no-sucesso-dos-projetos-de-software.aspx#ixzz3ZlgykkYG>>. Acesso em: 02 maio 2015.

SILVA, S. C. da L. e. **Priorização e negociação de requisitos**. Monografia – Pós-graduação em ciências da computação, Universidade Federal de Pernambuco, 2008. Disponível em: <http://www.cin.ufpe.br/~in1020/arquivos/monografias/2008-1/Monografica_SCLS.pdf>. Acesso em: 28 abr. 2015.

SILVA, S. F. B. **Engenharia de Requisitos**: Uma análise das técnicas de levantamento de requisitos. Monografia – Curso de Ciência da Computação, Universidade FUMEC, Belo Horizonte. 2012. Disponível em: <http://www.ricardoterra.com.br/publications_files/students/2012_fumec_silva.pdf>. Acesso em: 01 maio. 2015.

SOMMERVILLE, I. **Engenharia de Software**. São Paulo: Pearson Prentice hall, 2008.

SOTILLE, M. A. **Gerenciamento do escopo em projetos**. Rio de Janeiro: Editora FGV, 2014.

SOUZA, V. E. S. **Exercícios Pizzaria** – Análise de Objetivos. UFES. Disponível em: <<http://www.inf.ufes.br/~vitorsouza/wp-content/uploads/academia-br-requisitos-exercicio-pizzaria-01-resolucao.pdf>>. Acesso em: 01 jul. 2015.

STAKEHOLDERS: da identificação ao Plano de Gerenciamento de Comunicações. **InovaGP**. Disponível em: <<http://www.inovagp.com/2012/03/stakeholders-da-identificacao-ao-plano-de-gerenciamento-de-comunicacoes/>>. Acesso em: 23 jun. 2015.

STANDISH Group. **Chao Report**. 2014. Disponível em: <<http://www.projectsmart.co.uk/docs/chaos-report.pdf>>. Acesso em: 04 maio 2015.

VIEIRA, D. Scrum: A Metodologia Ágil Explicada de forma Definitiva. **MindMaster Educação Profissional**. Disponível em: <<http://www.mindmaster.com.br/scrum/>>. Acesso em: 04 jul. 2015.

ZENARO, F. dos S. A utilização do Scrum em um sistema web: um estudo de caso. **T.I.S.** Disponível em: <revistatis.dc.ufscar.br/index.php/revista/article/download/16/20>. 05 maio 2015.



UNIDADE I

1. B.
2. D.
3. B.
4. D.
5. A.

UNIDADE II

1. C.
2. D.
3. A.
4. C.
5. A.
6. C.
7. B.

UNIDADE III

1. A.
2. D.
3. D.
4. D.
5. A.
6. C.
7. C.
8. A.



GABARITO

UNIDADE IV

1. A.
2. C.
3. A.
4. B.

UNIDADE V

1. D.
2. B.
3. A.

