

Engenharia de Requisitos e Processos de *Software*



Cruzeiro do Sul Virtual
Educação a distância

Material Teórico



Ferramentas e Exemplos de Engenharia de Requisitos

Responsável pelo Conteúdo:

Prof. Me. Artur Marques

Revisão Textual:

Prof.^a Dr.^a Selma Aparecida Cesarin

UNIDADE

Ferramentas e Exemplos de Engenharia de Requisitos



- Requisitos Ágeis – Histórias dos Usuários;
- *Planning Game*;
- Principais Ferramentas para Gestão de Requisitos;
- Método Tradicional – Cascata.



OBJETIVO DE APRENDIZADO

- Familiarizar o aluno com as práticas, as ferramentas e as representações de artefatos gráficos para requisitos de sistemas.



Orientações de estudo

Para que o conteúdo desta Disciplina seja bem aproveitado e haja maior aplicabilidade na sua formação acadêmica e atuação profissional, siga algumas recomendações básicas:

Determine um horário fixo para estudar.

Mantenha o foco! Evite se distrair com as redes sociais.

Procure manter contato com seus colegas e tutores para trocar ideias! Isso amplia a aprendizagem.

Seja original! Nunca plágie trabalhos.

Aproveite as indicações de Material Complementar.

Conserve seu material e local de estudos sempre organizados.

Não se esqueça de se alimentar e de se manter hidratado.

Assim:

- ✓ Organize seus estudos de maneira que passem a fazer parte da sua rotina. Por exemplo, você poderá determinar um dia e horário fixos como seu “momento do estudo”;
- ✓ Procure se alimentar e se hidratar quando for estudar; lembre-se de que uma alimentação saudável pode proporcionar melhor aproveitamento do estudo;
- ✓ No material de cada Unidade, há leituras indicadas e, entre elas, artigos científicos, livros, vídeos e sites para aprofundar os conhecimentos adquiridos ao longo da Unidade. Além disso, você também encontrará sugestões de conteúdo extra no item **Material Complementar**, que ampliarão sua interpretação e auxiliarão no pleno entendimento dos temas abordados;
- ✓ Após o contato com o conteúdo proposto, participe dos debates mediados em fóruns de discussão, pois irão auxiliar a verificar o quanto você absorveu de conhecimento, além de propiciar o contato com seus colegas e tutores, o que se apresenta como rico espaço de troca de ideias e de aprendizagem.

Requisitos Ágeis – Histórias dos Usuários

Os requisitos ágeis, ou seja, as histórias dos usuários nas metodologias ágeis em geral e, notadamente, no processo XP, é a unidade fundamental das atividades dos times de desenvolvimento.

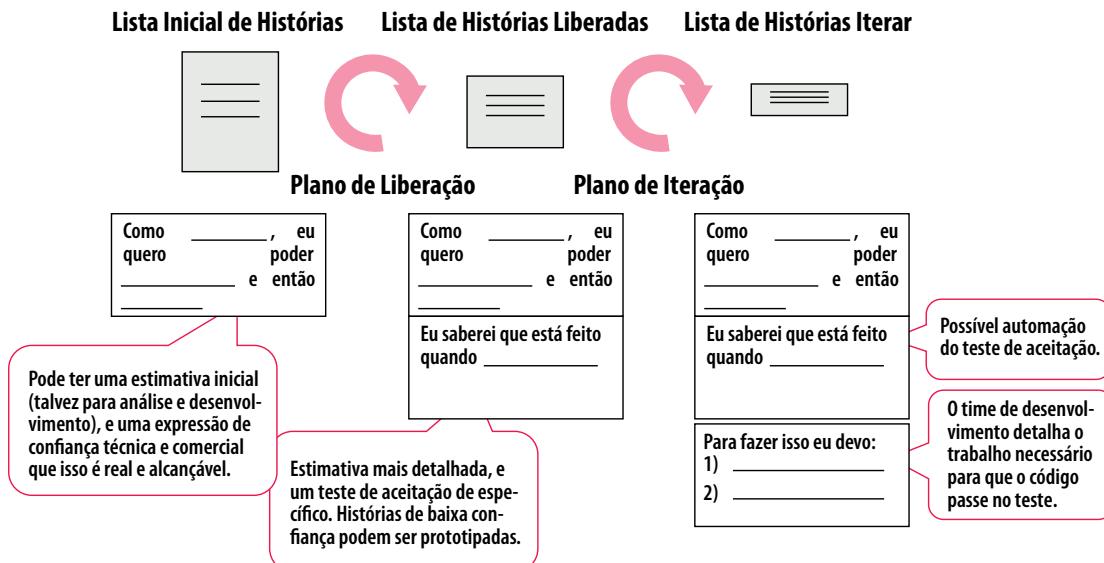


Figura 1 – Histórias e seu fluxo pelo XP, por Naresh Jain – *Agile FAQs*

As histórias de usuários são pequenas, muito menores do que outros artefatos de requisitos, como casos de uso ou cenários de uso. É importante reconhecer que todas as declarações feitas num *Index Card* (Cartão de Índice ou Cartão de Histórias do Usuário) representam uma única história de usuário.

Mas, como escrevemos histórias de usuários?

- Os estudantes podem comprar passes de estacionamento mensais *on-line*;
- Os passes de estacionamento podem ser pagos por meio de cartões de crédito;
- Os passes de estacionamento podem ser pagos via *PayPal*;
- Professores podem inserir as notas dos alunos;
- Os estudantes podem obter o cronograma atual do seminário;
- Os alunos podem solicitar transcrições oficiais;
- Os alunos só podem se inscrever em seminários para os quais tenham pré-requisitos;
- Transcrições estarão disponíveis *on-line* por meio de um navegador padrão.

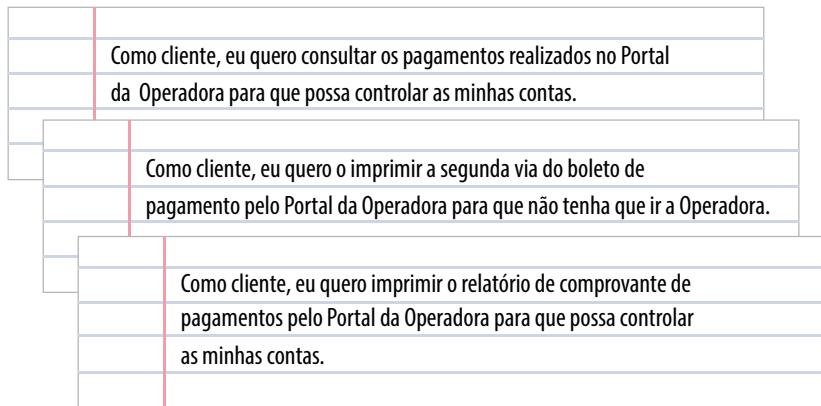


Figura 2 – Exemplo de História Inicial do Usuário formal

Há situações importantes para as partes interessadas ou a área de negócios escreverem histórias de usuários:

- **“As partes interessadas escrevem histórias de usuários:** Um conceito importante é que os envolvidos no projeto escrevem as histórias do usuário, não os desenvolvedores. As histórias de usuários são simples o suficiente para que as pessoas aprendam a escrevê-las em poucos minutos, por isso faz sentido que os especialistas do domínio (as partes interessadas) as escrevam;
- **Use a ferramenta mais simples:** As histórias de usuários geralmente são escritas em fichas de índice. Os cartões de índice são muito fáceis de trabalhar e, portanto, são uma técnica de modelagem inclusiva;
- **Lembre-se dos requisitos não funcionais:** As histórias podem ser usadas para descrever uma ampla variedade de tipos de requisitos;
- **Indique o tamanho estimado:** Inclua uma estimativa para o esforço de implementar a história do usuário. Uma maneira de estimar é atribuir pontos de história do usuário (*planning poker*) a cada cartão, uma indicação relativa de quanto esforço levará um par de programadores para implementar a história. A equipe sabe então que, se atualmente, leva em média 2,5 horas por ponto da carta; portanto, a história do usuário fica muito mais fácil de estimar. Então $2,5h \times$ uma carta com 5 pontos teremos uma atividade cuja duração seria de aproximadamente 12,5h;
- **Indique a prioridade:** Os requisitos, incluindo os defeitos identificados como parte de suas atividades de testes paralelos independentes ou por suas operações e esforços de suporte, são priorizados pelos envolvidos no projeto e adicionado à pilha no local apropriado. Você pode facilmente manter uma pilha de requisitos priorizados movendo as cartas na pilha conforme apropriado. O cartão de história do usuário inclui uma indicação da prioridade. Podemos usar uma escala de um a dez, sendo 1 a prioridade mais alta e 10 a mais baixa;
- **Inclua um identificador exclusivo.** O cartão também deverá possuir um identificador exclusivo para a história do usuário. A única razão para isso é manter a rastreabilidade entre a história do usuário e outros artefatos, em particular testes de aceitação” (AMBLER, 2009, p. 3).

Podemos, ainda, ter histórias de usuários muito grandes que chamamos de **epopeias** e grupos de histórias que chamamos de **temas**.

Vejamos como são estruturadas:

- **Epopeias:** são grandes histórias de usuário, normalmente, aquelas que são grandes demais para serem implementadas em uma única iteração e, portanto, precisam ser desagregadas em histórias de usuário menores em algum momento. Normalmente, têm prioridade mais baixa. Não faz sentido desagregar um épico de baixa prioridade porque você estaria investindo tempo em algo que você nunca poderá abordar, a menos que uma parte do épico tenha alta prioridade e precise ser descartada;
- **Tema:** é uma coleção de histórias de usuários relacionadas. Por exemplo, para um sistema de registro universitário, pode haver temas em torno dos alunos, gerenciamento de cursos, geração de transcrição, administração de notas, processamento financeiro. São usados para organizar histórias em lançamentos ou organizá-las para que várias subequipes possam trabalhar nelas.

As histórias de usuários é o alfa e o ômega dos requisitos ágeis, e os desenvolvedores costumam escrever testes baseados nelas.

As histórias são noções básicas perfeitas para testes, porque são breves e caracterizam os recursos mais importantes do produto final. Os testes são geralmente escritos antes de começar a criação do código do produto. Essa abordagem do desenvolvimento visa a economizar tempo e a atender aos termos do projeto.

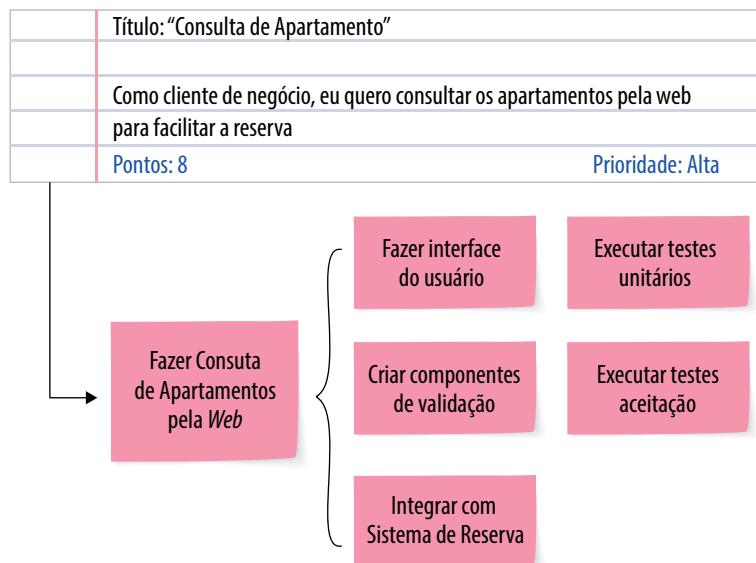


Figura 3 – Exemplo de Cartão de Índice (cartão de história de usuário)

As Histórias do Usuário são descrições de funcionalidades que têm valor para o cliente, não é uma lista de tarefas. O ato de dividir uma história em tarefas ajuda a torná-la compreensiva aos desenvolvedores.

No exemplo ilustrado na Figura 3, o cliente precisa que os usuários que navegam pelo seu site possam consultar e reservar os apartamentos. Para desenvolver esta funcionalidade, a equipe de desenvolvimento decidiu dividir a história nas seguintes

tarefas: ‘Fazer interface do usuário’, ‘Criar Componentes de validação’ e ‘Integrar com Sistema de Reserva’. E para garantir que estas tarefas irão atender a descrição da estória, criaram testes unitários e de aceitação, baseados no que o *Product Owner* escreveu no verso do cartão. A história tem valor agregado para o cliente porque representa a funcionalidade em questão. Por outro lado, as tarefas representam apenas uma parte do todo e não tem o mesmo valor.

Planning Game

O Jogo de Planejamento permite-nos encontrar rapidamente um plano aproximado e depois refiná-lo à medida que o projeto avança. Poderíamos dizer que o Jogo do Planejamento é uma reunião, é um ponto vital de interação entre cliente e desenvolvedor.

A reunião acontece com a equipe trabalhando em uma pilha de cartões de índice que contêm as histórias do usuário. Os requisitos do usuário são escritos em fichas de índice e são tratados durante o Jogo de Planejamento.

Cartões de índice são uma ferramenta altamente eficaz. A utilidade simples dos cartões conecta clientes e programadores para atingir seu objetivo comum. O uso de cartões de índice não é obrigatório no Jogo de Planejamento, e você pode descobrir que outras ferramentas, como aplicativos da Web, podem ser eficazes.

Quaisquer que sejam as ferramentas escolhidas, há uma clara separação de responsabilidades durante o Jogo de Planejamento.

Vejamos quem são os atores em XP:

Tabela 1 – Responsabilidades durante o jogo de planejamento

O negócio	Técnico
Definir o escopo do lançamento	Estimar quanto tempo cada história do usuário levará
Definir a ordem de entrega (quais histórias são feitas primeiro)	Comunicar os impactos técnicos da implementação de requisitos
Definir datas e horários de lançamento	Dividir as histórias do usuário em tarefas e aloque o trabalho

Fonte: Adaptado de informit.com

Você pode realizar as estimativas em cima desses cartões de histórias utilizando o *Planning Poker*, muito útil para as equipes ágeis. O planejamento acontece com frequência e é feito levando-se em consideração a expectativa de mudança. Mesmo com o nosso plano aproximado no jogo, temos uma imagem mais precisa do que a maioria, porque o cliente e a equipe de desenvolvimento trabalharam juntos para criá-lo.

O planejamento aborda duas questões-chave no desenvolvimento de software: prever o que será realizado até a data de vencimento e determinar o que fazer a seguir.

A ênfase está em direcionar o projeto. Duas etapas trabalham com esses conceitos, conforme Lacey:

- **O Planejamento de Liberação** é uma prática em que o Cliente apresenta os recursos desejados aos programadores e os programadores estimam sua dificuldade. Com as estimativas de custos em mãos e com o conhecimento da importância dos recursos, o Cliente estabelece um plano para o projeto. Os planos de lançamento inicial são necessariamente imprecisos: nem as prioridades nem as estimativas são verdadeiramente sólidas e, até que a equipe comece a trabalhar, não saberemos com que rapidez elas serão. Até mesmo o primeiro plano de lançamento é preciso o suficiente para a tomada de decisões, no entanto, e as equipes revisam o plano de lançamento regularmente.
- **Planejamento de Iteração** é a prática pela qual a equipe recebe orientação a cada duas semanas. As equipes de XP criam softwares em “iterações” de duas semanas, entregando softwares úteis ao final de cada iteração. Durante o planejamento de iteração, o cliente apresenta os recursos desejados para as próximas duas semanas. Os programadores dividem-nas em tarefas e estimam seu custo (em um nível mais refinado de detalhes do que no Planejamento da Liberação). Com base na quantidade de trabalho realizado na iteração anterior, a equipe se inscreve para o que será realizado na iteração atual. (LACEY, 2018)

O jogo de planejamento XP tem dois participantes no processo de planejamento: Negócios e Desenvolvimento. Isso pode ajudar a remover parte do calor emocional inútil da discussão. Não há como um simples conjunto de regras existir para lembrar a todos como eles gostariam de atuar, e eles fornecem uma referência comum quando as coisas não estão indo bem (BECK, 1999, p.38).

- **Desenvolvedores:** são as pessoas que irão realizar a implementação do que foi definido no cartão;
- **Negócio (cliente):** são as pessoas que vão dizer aos desenvolvedores o que eles querem que seja implementado. O cliente decide a prioridade do trabalho a ser feito e o que está dentro e fora do escopo. O cliente Agile garante a máxima satisfação do cliente do produto e o valor das organizações, garantindo quais são as histórias nas quais as equipes estão trabalhando, são priorizadas para gerar o máximo valor.

Segundo Beck, existem três fases do jogo a saber:

1. **Exploração:** Esta é uma maneira de tentar descobrir coisas novas que o sistema é capaz de fazer. O propósito da fase de exploração é dar aos jogadores uma apreciação do que todo o sistema deve eventualmente fazer. Exploração tem três movimentos:
 - **Escreva uma história:** Negócios escreve uma história descrevendo algo que o sistema precisa fazer. As histórias são escritas em cartões de índice, com um nome e um parágrafo curto descrevendo o propósito da história;

- **Estimar uma história:** Desenvolvimento estima quanto tempo a história levará para implementar. Se o desenvolvimento não puder estimar a história, ela pode pedir aos negócios que esclareçam ou dividam a história;
 - **Uma história é contada:** Se Desenvolvimento não puder estimar uma história completa ou se os negócios perceberem que parte de uma história é mais importante do que o restante, os negócios podem transformar uma história em duas ou mais histórias.
2. **Compromisso:** Será tomada a decisão quanto aos passos a serem seguidos na realização dos requisitos. O propósito da fase de compromisso é que as empresas escolham o escopo e a data da próxima versão e que o desenvolvimento se comprometa com a entrega. Possui 4 movimentos a saber:
- **Classificar por Valor:** Negócios classifica as histórias em 3 pilhas:
 1. Aquelas sem as quais o sistema não funcionará;
 2. Aquelas que são menos essenciais, mas fornecem valor comercial significativo;
 3. Aquelas que seria bom ter.
 - **Classificar por risco:** Desenvolvedores classificam as histórias 3 pilhas:
 1. Aquelas que podem estimar com precisão;
 2. Aquelas que podem ser razoavelmente estimadas;
 3. Aquelas que não podem estimar.
 - **Definir velocidade:** Desenvolvedores informam a rapidez com que a equipe pode programar em tempo de engenharia ideal por mês;
 - **Escolher escopo:** A empresa escolhe o conjunto de cartões na liberação, seja definindo uma data para a engenharia estar completa e escolhendo cartões com base em sua estimativa e velocidade do projeto, ou escolhendo os cartões e calculando a data.
3. **Steer (direção):** Esta é a orientação dos desenvolvedores para o requisito do projeto. O propósito da fase de direção é atualização o plano com base no que é aprendido pelo desenvolvimento e pelos negócios. A fase de direção tem quatro movimentos:
1. **Iteração:** no início de cada iteração (de 1 até 3 semanas), Negócios selecionam uma iteração das histórias mais valiosas a serem implementadas. As histórias da primeira iteração devem resultar em um sistema que é executado de ponta a ponta;
 2. **Recuperação:** Se o desenvolvimento perceber que ele superestimou sua velocidade, ele poderá solicitar a Negócios que é o conjunto de histórias mais valioso a ser retido no release atual com base na nova velocidade e nas estimativas;
 3. **Nova história:** Se os negócios perceberem que precisa de uma nova história durante o meio do desenvolvimento de um lançamento, ele poderá escrever a história. O desenvolvimento estima a história, depois o negócio remove as histórias com a estimativa equivalente do plano restante e insere a nova história;

4. **Reestimativa:** Se o desenvolvimento achar que o plano não fornece mais um mapa preciso do que fazer, ele pode reestimar todas as histórias restantes e definir a velocidade novamente. (BECK, 1999)

Fowler (2012) afirmou que “Valores sem práticas são difíceis de aplicar e podem ser aplicados de muitas maneiras. É difícil saber por onde começar e práticas sem valores são como atividades sem propósito”.

O jogo de planejamento de maneira sumária se resume a:

- Relacionar os itens de trabalho que talvez precisem ser feitos (histórias de usuários que carregam implicitamente requisitos funcionais, não funcionais e regras de negócios aglutinadas para que formem uma *sprint* que tem propósito de entregar uma funcionalidade por si só ou que seja dependente de mais *sprints* e também quando estiverem concluídas gerarão um *release* que será liberado em produção, com entregas valiosas ao cliente);
- Estimar os itens;
- Definir um orçamento para o ciclo de planejamento;
- Concordar com o trabalho que precisa ser feito dentro do orçamento. Ao negociar, não altere as estimativas ou o orçamento.

Principais Ferramentas para Gestão de Requisitos

Muitas vezes, eliciar os requisitos de um sistema complexo e de natureza dinâmica gera desafios sérios para controlar o ciclo de vida dos requisitos. Por isso, as ferramentas nos auxiliam na organização, afinal, num sistema, muitas vezes centenas de requisitos devem ser cumpridos.

Aqui vão algumas ferramentas que vão auxiliar você nessa tarefa:

- **Helix RM:** ajuda as equipes a capturar, decompor e priorizar os requisitos, identificar o status de cada requisito dentro do processo de aprovação, realizar revisões de requisitos, manter-se atualizado com as alterações e colaborar com todas as partes interessadas. A ferramenta também permite que as empresas reutilizem os requisitos de outros projetos para reduzir o tempo de validação, o retrabalho e garantir a consistência entre os projetos. Os requisitos podem ser vinculados a outros requisitos, casos de teste e resultados ou código-fonte. O Helix RM oferece uma matriz de rastreabilidade para ajudar as equipes a identificar melhor as causas raiz. A análise de impacto para executar cenários hipotéticos, rastreamento de bugs e gerenciamento de tarefas pode ser concluída ao lado do Jira. Toda a colaboração com as partes interessadas pode ser feita em tempo real para que todos estejam atualizados sobre as mudanças conforme elas acontecem. Esta ferramenta não é gratuita;

- » **Integrações:** O *Helix RM* integra-se ao *Atlassian Jira*, *Atlassian Bamboo*, Produtos Microsoft, Slack, Adobe, ActiveState Komodo, Apache Maven, Amazon Lumberyard, Autodesk 3ds Max, Autodesk Maya, Crytek CRYENGINE, DBmaestro TeamWork, Eclipse, ElectricFlow, GitHub, Go2Group ConnectALL, IBM Rational DOORS, IBM Aspera, JetBrains, Puppet, Prometheus, Thoughtworks e Unity.
- **Jira:** é uma das ferramentas mais reconhecidas usadas pelas equipes para gerenciamento de ciclo de vida de aplicativos (ALM) e gerenciamento de requisitos. O Jira ajuda as equipes a identificar e mapear os requisitos de negócios, colaborar com as partes interessadas, garantir que todas as tarefas se conectem diretamente a qualquer requisito de captura e fornecer às partes interessadas entregas de alta qualidade. O Jira facilita a visualização de como os requisitos de negócios e os problemas existentes estão vinculados, rastreia as tarefas do projeto relacionadas aos requisitos de negócios e determina como os requisitos diferem de especificações padrão. O Jira também permite que as partes interessadas criem e visualizem uma estrutura de hierarquia de requisitos. O Jira não é uma ferramenta gratuita;
- » **Integrações:** O Jira possui uma extensa lista de integrações, incluindo gerenciamento de projetos, ferramentas administrativas, utilitários, blueprints, gráficos e diagramas, CRM, painéis, codificação, e-mail, gerenciamento de documentos, suporte técnico, segurança, testes e muito mais. O site da Atlassian oferece uma lista completa das atuais integrações do Jira.
- **Orcanos:** A ferramenta de *Application Lifecycle Management (ALM)* e gerenciamento de requisitos (RM) da Orcanos é uma solução flexível e poderosa que fornece um único repositório para gerenciamento de requisitos em empresas de pequeno a grande porte. É compatível com dispositivos médicos e oferece rastreabilidade de ponta a ponta, análise de impacto, painéis em tempo real, alertas e notificações. A Orcanos oferece uma ferramenta de rastreabilidade de requisitos que facilita o rastreamento de cobertura e rastreabilidade de requisitos de sistema, hardware ou software, casos de teste, riscos e muito mais. As partes interessadas podem obter uma visão hierárquica de seus requisitos e colaborar através de e-mail, mensagens instantâneas, alertas ou notificações. A Orcanos também fornece um gerador de documentos Microsoft Word que suporta relatórios incorporados e modelos personalizados. Orcanos, não é uma ferramenta gratuita;
- » **Integrações:** Entre em contato com a Orcanos para obter uma lista de todas as integrações de aplicativos.
- **ReQtest:** A ferramenta de gerenciamento de requisitos ReQtest coloca o foco no gerenciamento de projetos de ponta a ponta com foco na experiência do usuário. O ReQtest oferece um módulo de gerenciamento de testes, rastreamento avançado de bugs e um dashboard intuitivo para rastrear e gerenciar tarefas. A ferramenta simplifica o processo de identificação e gerenciamento da interface comercial, de marketing, funcional, não funcional, de usuário ou

outros requisitos. Os requisitos de negócios são armazenados em uma estrutura semelhante a uma árvore, na qual as equipes podem ver, planejar e gerenciar tudo com mais eficiência. *ReQtest* não é uma ferramenta gratuita;

» **Integrações:** *ReQtest* integra-se ao Jira; integrações personalizáveis também são oferecidas via API.

• **Visure Requirements:** *Requirements* oferece uma ferramenta que fornece suporte essencial para o processo de requisitos de negócios de ponta a ponta. A ferramenta visa padronizar e aplicar processos claramente definidos e formalizar uma estrutura de especificação de requisitos comum. Ele gerencia as alterações durante todo o ciclo de vida do aplicativo e captura, analisa, valida, rastreia e permite a reutilização de requisitos. As partes interessadas podem criar famílias de produtos que compartilham um conjunto de requisitos, requisitos de padrões ou recursos comuns, juntamente com testes ou casos de uso. Como uma ferramenta multiusuário com recursos de controle de versão, várias partes interessadas podem fazer alterações no mesmo requisito, simultaneamente, sem nenhum problema. *Visure* não é uma ferramenta gratuita;

» **Integrações:** integrações com ferramentas de terceiros incluem *Doors*, *HP ALM*, *RiskCAT*, *Jira* e *Enterprise Architect* (MOIRA, 2019).

Método Tradicional – Cascata

Durante o estágio de requisitos, os desenvolvedores anotam todos os requisitos possíveis de um sistema em um documento de requisitos. O documento define o que o sistema deve fazer, mas não necessariamente como ele funcionará. Os desenvolvedores basearão todo o desenvolvimento futuro do *software* no documento de requisitos.

Na próxima etapa da análise, os desenvolvedores usam o documento de requisitos para examinar e detalhar o *design* lógico ou teórico do sistema, sem levar em consideração suas tecnologias de *hardware* ou *software*.

Como sabemos, isso é muito útil, porque os desenvolvedores podem detectar erros de *design* durante os estágios de análise e *design*, o que os ajuda a evitar a geração de código com defeito durante o estágio de codificação. O projeto definiu metas claramente, para que os desenvolvedores possam trabalhar em direção a objetivos concretos e medir facilmente seu progresso.

No processo em cascata, todos os requisitos do sistema a serem desenvolvidos são capturados nessa fase de análise com base nas necessidades e nos problemas e estão em um documento chamado DRS ou Documento de Especificação de Requisitos.

Os requisitos devem ser claros e também de maneira detalhada. Esses requisitos são estabelecidos pela equipe, que inclui especialistas no assunto, usuários de negócios e analistas de negócios.

O analista entende o domínio do problema, corrige a captura e documenta os requisitos no documento de especificação de requisitos. O analista usa o método de perguntas e respostas para esclarecer as dúvidas e também para a atualização do documento de requisitos.

Os usuários e os analistas se valem, ainda, de reuniões presenciais, ou chamadas de vídeo e áudio, além, é claro, de se valerem, também, de questionários, folhas de entrevista, ferramentas de modelagem, quadro branco para capturar as entradas nos requisitos.

Primeiro, após a captura, são analisados os requisitos e se escrevem as especificações, depois que um documento de requisitos é concluído, ele é revisado pelos analistas e pelos usuários, para garantir que os requisitos esperados sejam capturados, e sejam claros e compreendidos com facilidade.

Após essa revisão, o documento de requisitos é assinado para ser usado nas outras fases do modelo Cascata.

Os requisitos são considerados uma linha de base para processos de controle de alterações para gerenciar a modificação feita. Escrever boas especificações de requisitos do Sistema é essencial para o sucesso de qualquer Projeto de Software.

Geralmente, a elaboração de especificações técnicas para o software ocorre após uma primeira discussão entre a equipe de desenvolvimento e o proprietário do produto, isso em qualquer tipo de processo, mas no Cascata é muito necessário.

As especificações servem como referência para a estimativa de custo e tempo. Como o documento de requisitos do sistema visa a descrever fielmente o software a ser desenvolvido, isso torna o processo de estimativa muito mais fácil e muito mais preciso. Lembre-se de que, no processo cascata, que não permite alteração, uma vez definido o escopo, é fundamental a precisão.

O documento de especificação de requisitos DRS também é utilizado como documento de especificação funcional DFD ou documento de requisitos do produto DRP, sendo que o DER descreve todas as funcionalidades e explica como a funcionalidade estará dentro de um determinado sistema como parte de um sistema maior ou como um sistema independente.

Na sequência, são apresentadas as perguntas chave para você responder durante o desenvolvimento do software:

- O que o aplicativo ou software deve fazer? Isso ajuda a identificar as principais funcionalidades e o objetivo principal do software;
- Como o software deve se comportar? Serve para entender como o software interage com o ambiente em que é implantado. Ele também define a especificação de hardware e define como o software interage com os usuários finais. O software a ser descrito pode ser um sistema inteiro ou, às vezes, faz parte de um sistema mais extenso. É, então, essencial definir como essa parte interage com um sistema maior e como os dois sistemas se comunicam. A especificação

de requisitos do sistema de CRM (Relacionamento com o Cliente) ou ERP (Gestão e Planejamento de Recursos da Empresa) são exemplos em que é essencial entender como o *software* deve se comportar;

- Quais são os requisitos em termos de desempenho? Fornecerá, por exemplo, informações sobre o tempo de resposta aceitável, com que rapidez deve responder e com que rapidez deve lidar com os problemas quando ocorrerem. Sim, às vezes, eles são requisitos não funcionais;
- Existem requisitos ou restrições que devem ser levados em consideração ou respeitados? Ele tem como objetivo determinar as restrições a serem levadas em consideração durante o *design*, o desenvolvimento e a implantação do Sistema.

O DRS, em sua construção, varia de empresa e de abordagem, portanto não existe um padrão, mas ao menos esses componentes são úteis em sua composição:

- **Introdução:** é importante que a equipe de desenvolvimento e os proprietários do produto definam e escrevam essa parte juntos. No final, podemos incluir uma breve visão geral do documento para dar aos leitores uma ideia do que eles podem esperar do documento. Não esqueça de incluir os termos técnicos;
- **Descrição Geral:** é importante explicar as diferentes funcionalidades do aplicativo. Nesta parte, as interfaces de *hardware* e de usuário são definidas. Em que dispositivo os usuários finais esperam acessar o aplicativo;
- **Requisitos específicos:** os requisitos são detalhados para facilitar o *design* do produto e validar o *software* de acordo com os requisitos. Aqui, é importante descrever todas as entradas que o *software* manipula e todas as saídas para definir melhor a interação com outros sistemas e facilitar a integração. As funcionalidades enumeradas na seção anterior serão detalhadas aqui;
- **Referências:** é importante incluir informações sobre o conteúdo, para que seja mais fácil encontrá-las quando necessário.

Segundo Osotskyi, são consideradas más práticas num DRE (Documento de Requisitos do Sistema):

- **Dicionário incompleto:** o DER pode incluir jargões que somente pessoas familiarizadas com a empresa podem entender. Uma especificação de requisitos visa proporcionar a todos os envolvidos no desenvolvimento do *software* uma melhor compreensão do que o *software* faz, etc. é importante que todos compreendam todos os termos usados no documento.
- **Misturando conceitos:** pode ser tentador lançar todas as informações que temos no mesmo local, mas isso leva a uma documentação ruim.
- **Inclua instruções de desenvolvimento:** é importante separar os requisitos de *software* da implementação técnica. Os proprietários do produto conhecem melhor suas necessidades e a equipe de desenvolvimento sabe como desenvolver o *software* que atende a essas necessidades.

- **Ação passiva:** Por exemplo: os relatórios são gerados clicando em um determinado botão. É importante saber que esperamos a geração de relatórios a partir do *software*, mas também é importante saber quem clicará no botão para gerar o relatório.
- **Documentação ambígua e incompleta:** algumas vezes algumas linhas nos requisitos podem levar a várias interpretações. Além disso, para cada funcionalidade ou situação descrita é importante que o documento não apresente aspectos ainda não determinados. (OSETSKYI, 2018, p. 4)



Veja um exemplo claro de um documento de requisitos, acessando: DA SILVA, F. R. **Documento de Requisitos do Sistema** – Módulo de Avaliação Acadêmica – Versão 0.1. 2016. Disponível em: <http://bit.ly/2FyJ6Ux>

Material Complementar

Indicações para saber mais sobre os assuntos abordados nesta Unidade:

▶ Vídeos

Histórias de Usuário – Fatto Consultoria e Sistemas

<https://youtu.be/sEtiCJfXTE8>

3 formas de Quebrar Histórias de Usuário – Rodrigo Vieira

<https://youtu.be/0AVR9Zc5Del>

O que é requisito – Fatto Consultoria e Sistemas. 2015

<https://youtu.be/oo06hyLuFNU>

Gerenciamento de requisitos sem mistério

<https://youtu.be/jajQyzOpLaE>

Referências

AMBLER, S. W. **Histórias de usuários: uma introdução ágil.** 2009. Disponível em: <<http://www.agilemodeling.com/artifacts/userStory.htm>>. Acesso em: 30 ago. 2019.

BECK, K. **Extreme Programming Explained: Embracing Change.** 1st edition. Boston: Addison-Wesley, 1999.

FOWLER, M. **O Modelo de fluência Ágil.** 2012. Disponível em: <<https://martinfowler.com/articles/agileFluency.html>>. Acesso em: 30 ago. 2019.

LACEY, M. **Planning Game 2018.** Mitch Lacey & Associates, INC. EUA. Disponível em: <<https://www.mitchlacey.com/intro-to-agile/extreme-programming/planning-game>>. Acesso em: 30 ago. 2019.

MOIRA, A. **5 principais ferramentas de gerenciamento de requisitos.** 2019. Disponível em: <<https://cio.com.br/5-principais-ferramentas-de-gerenciamento-de-requisitos/>>. Acesso em: 30 ago. 2019.

OSETSKYI, V. **Como escrever a especificação de requisitos do sistema para desenvolvimento de software.** 2018. Disponível em: <<https://dzone.com/articles/how-to-write-the-system-requirements-specification>>. Acesso em: 30 ago. 2019.



Cruzeiro do Sul
Educacional