UNIDADE 4 – REQUISITOS DE *SOFTWARES*

MÓDULO 1 – O QUE SÃO REQUISITOS DE *SOFTWARE* E QUAL A SUA IMPORTÂNCIA

01

1 - IMPORTÂNCIA DOS REQUISITOS DE *SOFTWARE*

O desenvolvimento de requisitos preocupa-se com a descoberta, busca da qualidade, detalhamento, documentação, revisão e verificação dos requisitos do sistema. Essas preocupações são tratadas em detalhes por meio da divisão: elicitação, análise, especificação e verificação de requisitos.

Nesse contexto, há também a gerência de requisitos, que compreende a gerência de mudanças e o acompanhamento do desenvolvimento e da implementação dos requisitos.

A gerência de requisitos trata sobretudo das principais causas de riscos e falhas em projetos relacionadas à engenharia de requisitos, como por exemplo, desenvolvimento de interfaces erradas, desenvolvimento de requisitos supérfluos e desenvolvimentos de funções erradas.

Tudo isso leva a mudanças na aplicação, gerando atrasos, gastos desnecessários aumentando os custos do projeto.

Segundo Pressman, entender os requisitos de um problema está entre as tarefas mais difíceis e importantes enfrentadas por um analista de sistemas. Quase sempre o cliente não sabe o que é necessário e as funções que trarão benefícios; mesmo que os usuários fossem conhecedores de suas necessidades, elas mudariam ao longo do projeto. Todos que trabalham na área de sistemas há alguns anos já viveram o pesadelo de mudanças drásticas no final do projeto. São problemas desafiadores para a equipe de projetos.

Prospectar e construir sistemas é muito desafiador, e pode acabar com várias noites de sono. Mesmo assim é muito propenso que os desenvolvedores comecem a construir logo, antes de terem entendido o que realmente deve ser feito. Isso pode levar o projeto ao fracasso. A rapidez de construir algo logo para o gestor testar, pode ser um complicador se não tivermos conhecimento necessário do que o gestor precisa. Por isso essa fase de conhecimento dos requisitos é importante.

02

No aspecto de tarefas, análise do problema, documentação e técnicas para elicitação de requisitos formam a área denominada engenharia de requisitos. A engenharia de requisitos faz parte da engenharia de software e deve ser adaptada às necessidades do processo, do projeto, do produto e das pessoas que estão realizando o trabalho.

Assim, a engenharia de requisitos constrói uma ponte para o projeto e para a construção, levando todas as informações necessárias para a construção do software pelos desenvolvedores. Dentre as informações, destacam-se:

- as prioridades que orientam a ordem dos trabalhos,

- os custos,
- as funções e comportamentos que terão impactos no projeto.

A engenharia de requisitos fornece o mecanismo apropriado para entender o que o cliente necessita, analisando, avaliando a viabilidade da construção do projeto.

03

Pressman distingue a engenharia de requisitos por meio de sete tarefas, as quais permeiam toda fase de desenvolvimento do projeto:

Engenharia de requisitos						
Concepção	Levantamento	Elaboração	Negociação	Especificação	Validação	Gestão

Cada uma dessas tarefas possui várias outras tarefas.

a) Concepção – é como um projeto começa. O projeto pode começar em uma conversa informal, um telefonema, uma ideia, porém, a maioria dos projetos começa quando é identificada a necessidade do negócio ou é descoberto um novo serviço ou mercado em potencial. Geralmente é construído um plano de negócio quando temos algo novo, mas para os sistemas já construídos ou em andamento, em sua maioria, faz-se apenas uma análise de custo e risco. Aqui então estabelecemos um entendimento básico do problema, as pessoas que querem o software, a natureza da solução desejada e como será a comunicação e a colaboração entre todos os envolvidos do projeto, ou seja, equipe e gestores.

b) Levantamento – parece uma tarefa fácil, mas não é. Para entender a dificuldade é só perguntar ao cliente e aos outros usuários que utilizarão o sistema quais são os objetivos do sistema que será criado ou como o sistema atenderá às necessidades da empresa, e por fim, como o sistema deverá ser utilizado no dia a dia. Bem, percebemos que poderemos ter uma série de problemas, Christel e Kang identificaram os seguintes:

- Problemas de escopo;
- Problemas de entendimento;
- Problema de volatilidade.

Problemas de escopo
Os limites do sistema são definidos de forma precária ou os clientes especificam detalhes técnicos desnecessários que podem confundir em vez de esclarecer os objetivos do sistema;

Problemas de entendimento
Os clientes não estão completamente certos do que é preciso, têm um entendimento inadequado das capacidades e limitações de seus ambientes computacionais, não possuem entendimento completo do domínio do problema, têm problemas para transmitir as informações necessárias para o analista de sistemas, omitem informações que acreditam ser obvias;

Problema de volatilidade
Os requisitos mudam com o tempo, para ajudar a superar esses problemas devemos abordar o levantamento de requisitos de forma organizada.

c) Elaboração – as informações obtidas do cliente durante as fases de concepção e levantamento são expandidas e refinadas durante a elaboração. Essa tarefa concentra-se no desenvolvimento de um modelo de requisitos refinado que identifique os diversos aspectos da função, do comportamento e das informações do *software*. Sendo assim, a elaboração é guiada pela criação e aprimoramento de cenários de usuários que descrevem como os usuários finais irão interagir com o sistema.

d) Negociação – é comum o cliente pedir mais do que pode ser alcançado, de acordo com a quantidade de recursos limitados do negócio. Outro fator comum é clientes com necessidades conflitantes. É preciso conciliar esses conflitos por meio de um processo de negociação. Devemos solicitar aos clientes e interessados que ordenem seus requisitos, avaliem seus custos e riscos, bem como trate de seus conflitos internos, assim alguns requisitos são eliminados atingindo certo nível de satisfação por todos.

e) Especificação – no contexto dos softwares a especificação assume diferentes significados para as pessoas diferentes. Pode ser um documento por escrito, um conjunto de modelos gráficos, modelos matemáticos, conjunto de cenários de uso, protótipos ou qualquer outro tipo de documento. Essa variedade pode causar problemas no cronograma, desentendimentos e principalmente projeto fracassado.

f) Validação – os artefatos produzidos como consequência são avaliados quanto à qualidade em algum momento no desenvolvimento do projeto. A validação de requisitos examina a especificação para garantir que todos os requisitos tenham sido declarados de forma que não tenham inconsistências, omissões, que os erros detectados tenham sido corrigidos e que os artefatos estejam de acordo com os padrões estabelecidos para o processo, projeto e produto.

g) Gestão de requisitos – os requisitos sempre mudam durante o desenvolvimento do sistema, isso é comum, então é necessário que tenha uma gestão dessas mudanças nos requisitos. Sendo assim exige atividades necessárias para identificar, controlar e acompanhar as necessidades e suas mudanças a qualquer momento do projeto.

Cenários
Cada cenário de usuário é analisado sintaticamente para extrair classes de análise e outros documentos para melhor visualização e compreensão do cliente.

Avaliados
Uma ferramenta poderosa da validação de requisitos é a Revisão Técnica, que é formada pelos analistas de sistemas que trabalham com processos de Qualidade com a participação de clientes e

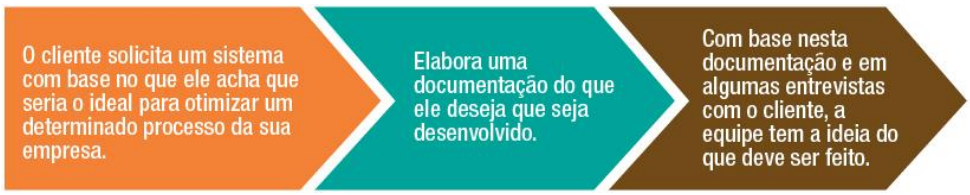
ouros interessados. Essa revisão examina a especificação de requisitos buscando erros no conteúdo ou na interpretação, inconsistências e requisitos conflitantes.

05

2 - VANTAGENS E DESVANTAGENS DE UMA BOA DOCUMENTAÇÃO

Primeiramente devemos esclarecer quais os tipos de documentos que podem ser desenvolvidos para o projeto e para produto, que é o resultante do projeto.

Segundo Rosane Marchand, analista de sistemas, para que o sistema possa ser desenvolvido no menor prazo, com menor custo operacional e maior confiabilidade, temos que definir a estratégia para cada etapa do desenvolvimento. O que acontece na prática, na maioria das empresas desenvolvedoras de produtos de *software*.



Sendo assim, com base no levantamento das regras do negócio a ser desenvolvido, deve-se elaborar uma documentação mínima, que servirá de diretriz para o desenvolvimento, homologação e implantação do sistema.

A documentação é sempre fruto de algum trabalho necessário. Esse trabalho possui um prazo estimado com base nas funcionalidades levantadas da documentação e das entrevistas. Cada funcionalidade irá demandar um esforço das equipes envolvidas, possui uma complexidade agregada e um risco, que pode ser do negócio, da implementação ou da implantação.

06

A documentação, na etapa de construção do *software* é somente a **documentação referente ao projeto**, ou seja, documentação do plano de ação para o desenvolvimento do sistema solicitado. Há também a documentação que está mais relacionada aos aspectos burocráticos do negócio, como por exemplo, contratos, cronograma, planilhas de custo e controle, plano de teste e de homologação e plano de implantação.

Bom mencionar que cada empresa segue seu próprio processo interno, portanto o que é de praxe em uma empresa não necessariamente se aplica à outra. É comum visitas técnicas, de uma empresa com outra, principalmente as governamentais, onde se pretende trocar conhecimentos e práticas nos procedimentos de desenvolvimento de *software*. As empresas buscam melhorias nos seus processos de desenvolvimento de *software*.

O objetivo de se elaborar a documentação do sistema para as equipes envolvidas no seu desenvolvimento é unicamente a de definir o que e **como deve ser feito** e **como deve funcionar.**

Vimos que o projeto documentado é fruto do trabalho da equipe do projeto, um fator muito importante é que o projeto documentado possui o histórico e as informações necessárias para se realizar uma alteração ou ajuste futura. Essa é a grande vantagem, pois passado algum tempo, caso a equipe seja desalocada ou contratada para outro projeto, a documentação apoiará os novos membros do projeto a conhecerem mais rapidamente sobre o trabalho que deverá ser feito.

Um ponto que acaba sendo uma desvantagem é quando os documentos não são revisitados, eles acabam ficando desatualizados e não ajudam quando necessário. A documentação do sistema deve servir como diretriz às equipes envolvidas, de modo a que estas mantenham o foco no que deve ser feito e como deverá funcionar.

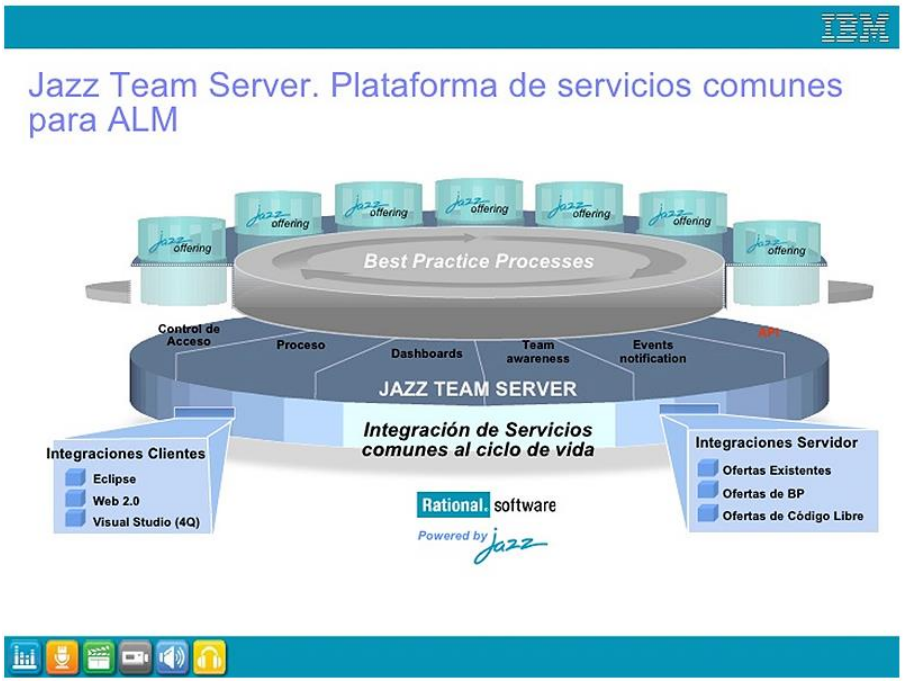
Devemos ter em mente que toda documentação é um artefato “vivo”, ou seja, mesmo depois de um produto finalizado, cada erro encontrado e reportado à empresa, alterará os documentos, tanto de projeto quanto de sistema, podendo refletir na documentação de usuário.

07

3 - FERRAMENTAS QUE GERENCIAM DOCUMENTOS

Planejar um projeto não é tarefa simples. Planejar um projeto sem ter ferramentas adequadas, torna a tarefa ainda mais complexa. É muito comum encontrarmos gerentes de projeto batalhando com ferramentas antiquadas, tentando entender o que está na tela, finalmente abrindo um sorriso quando tem um plano finalizado. E também é comum ver este sorriso desaparecer ao saber que o recurso pretendido para as principais tarefas está sobrecarregado e não poderá atender à demanda, ou que as tarefas predecessoras, que alguém informara estarem finalizadas, na verdade não estão ou que seus documentos estão todos desatualizados.

Ferramenta como a plataforma Jazz (RTC, RQM, RRC) da IBM é um ambiente "vivo", onde as informações atualizam-se e formalizam-se em tempo real, a tarefa de planejamento ganha novos contornos. Diferentes visualizações são permitidas, não perdendo algumas características que muitas ferramentas de mercado trazem. Podemos listar com facilidade todas as tarefas e atividades agrupadas em fases, com quaisquer informações sobre seus atributos.



Também podemos ter uma visão por recursos, indicando as tarefas atribuídas a cada pessoa e o tempo planejado para execução das mesmas. Caso o colaborador não tenha disponibilidade para atender a demanda, um simples drag-and-drop muda o responsável pela tarefa. Tudo em tempo real - se um funcionário planeja e aponta férias na ferramenta, por exemplo, a disponibilidade é automaticamente refletida! A funcionalidade de drag-and-drop pode ser utilizada em diversos movimentos para alterar o planejamento.

08

Atualmente as empresas investem em ferramentas para gerar documentação, controlar as alterações e gerenciar todas as alterações que forem sendo realizadas. Para manter toda a documentação, são necessários mecanismos que facilitem o controle das alterações nos documentos.

A IBM fornece algumas ferramentas que fazem esse armazenamento de informações em diversos níveis e fases do projeto, ajudando no gerenciamento de todo o projeto. Abaixo elencamos algumas das ferramentas importantes:

- **RTC - Rational Team Concert** - O IBM® Rational Team Concert™ é uma solução de gerenciamento de ciclo de vida do *software* que permite a colaboração contextual em tempo real para equipes distribuídas. Com base na plataforma do IBM Rational® Jazz™, o Rational Team Concert fornece uma configuração do processo, orientação e estrutura de execução que podem suportar seu ambiente inteiro de entrega de *software*.

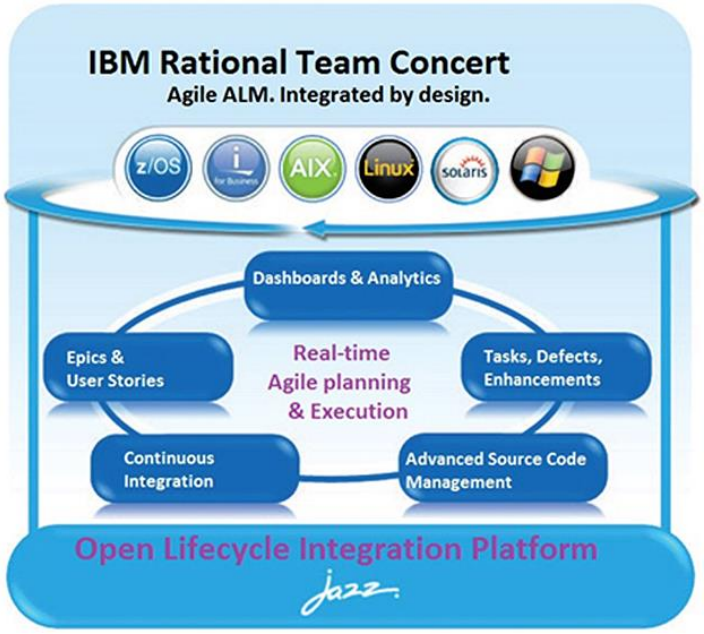
Essa solução oferece *software* de servidor sem custos e modelos com preço flexível. Oferece também licenciamento baseado em função e suporte de diversas plataformas em uma única liberação, permitindo a implementação de componentes individuais agora e no futuro.

Além disso, o Rational Team Concert fornece recursos de gerenciamento de mudanças colaborativo. Essas capacidades estão disponíveis separadamente e podem ser integradas como sistemas de controle de fonte popular.

09

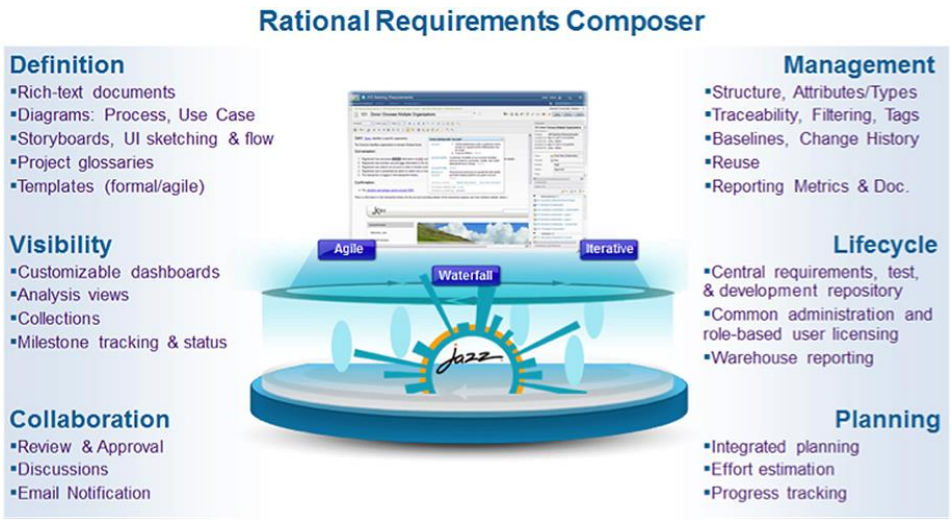
O Rational Team Concert ajuda as equipes a colaborarem com a entrega de *software* mais rápida. Além disso:

- Aprimora a colaboração da equipe com recursos integrados incluindo o item de trabalho, construção e gerenciamento de configuração de*software*.
- Fornece alta visibilidade em atividades do projeto e progresso da equipe com recursos de painéis multinível e relatórios.
- Facilita o planejamento e a execução de projetos ágeis e formais com ferramentas e modelos de planejamento. Os processos consistentes ajudam a melhorar a qualidade do *software*
- Ajuda a melhorar a produtividade com controle de fonte avançado para equipes distribuídas geograficamente.



10

- **RRC - Rational Requirements Composer** – o *software* capacita as equipes a definir, gerenciar e emitir relatórios sobre requisitos em um projeto de desenvolvimento de ciclo de vida. Esse aplicativo baseado na web suporta metodologias de desenvolvimento iterativo, em cascata e de escalada ágil usando processos de requisitos leves. Agora é possível reduzir o retrabalho e os custos, encurtar o tempo de entrada no mercado e melhorar os resultados dos negócios.



O Rational Requirements Composer ajuda a trabalhar com:

- Ciclo de vida de informações sobre requisitos;
- Equipe ampliada;
- Ciclo de vida do projeto.

Ciclo de vida de informações sobre requisitos
Implementando uma abordagem ampla e flexível que permite às equipes colaborar, esclarecer e chegar rapidamente a um consenso sobre os requisitos à medida que desenvolvem soluções movidas a negócios.

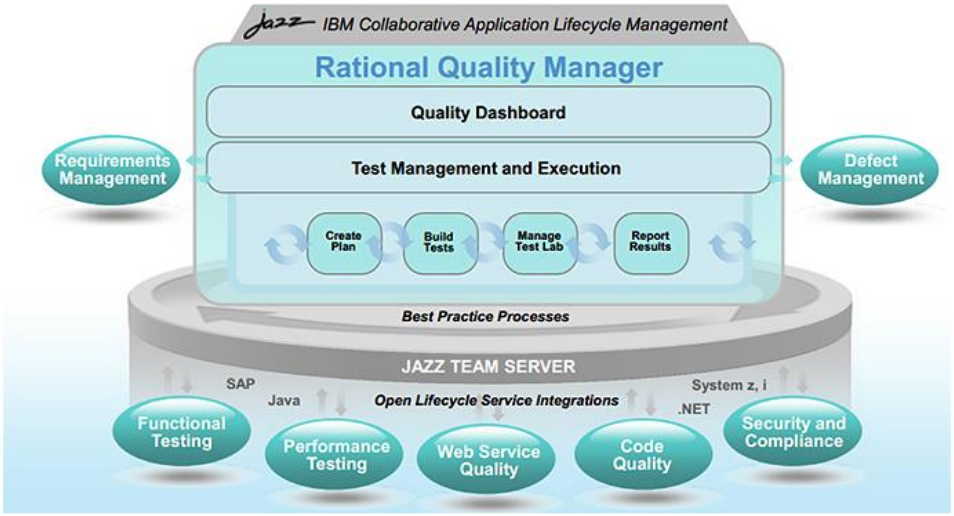
Equipe ampliada
Unificando vários interessados em todo o mundo, incluindo o patrocinador do cliente, os usuários, o marketing, departamento jurídico e conformidade, finanças, educação, operações e os próprios membros da equipe do projeto.

Ciclo de vida do projeto
Alinhando requisitos, desenvolvimento, gerenciamento de mudanças e gerenciamento de qualidade em uma solução abrangente do gerenciamento do ciclo de vida colaborativo IBM.

11

- **RQM - Rational Quality Management** – O Rational Quality Manager, construído sobre a plataforma Jazz, é uma solução de gerenciamento de qualidade colaborativa e baseada na web que oferece amplo planejamento de teste e gerenciamento de ativos de teste de requisitos a defeitos. O IBM® Rational® Quality Manager é um hub colaborativo para *software* orientado a negócios e qualidade de sistemas entre virtualmente qualquer plataforma e tipo de teste.

Este *software* ajuda as equipes a compartilhar informações sem problemas, usar automação para acelerar planejamentos de projeto e relatar em métricas para obter decisões de liberação informadas.



O Rational Quality Manager ajuda as equipes de controle de qualidade a:

• Colaborar

Compartilhar informações e atualizações de status do projeto sem problemas para que os membros da equipe possam sincronizar o trabalho em equipe por todo o ciclo de vida.

• Automatizar

Reduzir atividades de intensa mão de obra para acelerar planejamentos do projeto.

• Controlar

Entender e relatar em métricas do projeto para permitir decisões de liberação precisas, confiáveis e oportunas.

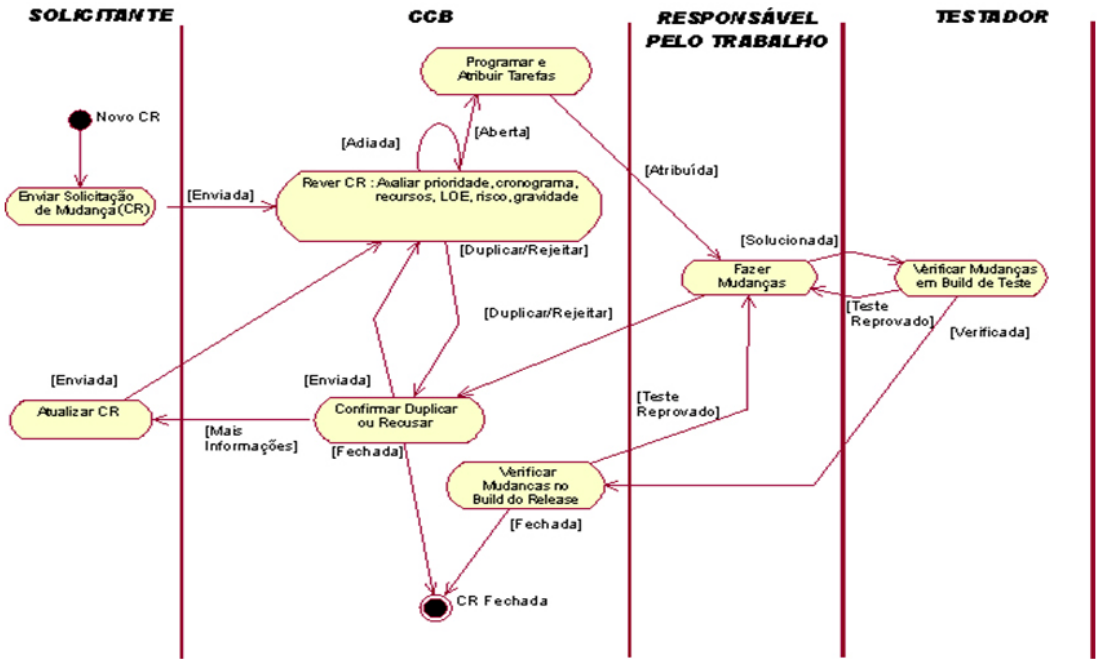
11

4 - LIDANDO COM ALTERAÇÕES NO SISTEMA

A imagem a seguir apresenta um exemplo de uma solicitação de alteração de um sistema. É gerada uma abertura de demanda, que irá desencadear diversas tarefas para realizar as alterações necessárias.

As alterações irão gerar atualizações nos documentos de requisitos, a equipe de projetos deve executar as tarefas de programação sem esquecer-se de atualizar os documentos. Caso essa atualização não seja realizada, com o passar do tempo, ninguém lembra mais das mudanças que foram realizadas.

Qualquer técnico novo que entrar na equipe terá acesso a documentos desatualizados, o que não ajuda muito, podendo até atrapalhar as próximas manutenções e até incorrendo no risco de pagar por serviços que já foram realizados.



12

RESUMO

Os requisitos de um *software* são tão importantes como um mapa para o viajante. A equipe deve entender as necessidades do cliente, sendo uma das tarefas mais difíceis para o Analista de Sistemas. Os requisitos são fundamentais para o sucesso do projeto, e para que eles existam, é necessário que sejam documentados e controlados. Por esse motivo temos o gerenciamento de requisitos. No gerenciamento dos requisitos podemos dividir em grandes tarefas (Concepção, Levantamento, elaboração, negociação, especificação, validação e gestão). Quando se fala em requisitos estamos também falando na documentação do projeto. Somente se sabe que a documentação foi bem feita quando alguma alteração no aplicativo é realizada. As vantagens de uma documentação atualizada são enormes para que a demanda não atrase e seja orçada corretamente. Por outro lado, ter documento somente para ter é prejuízo para o projeto e só leva ao atraso. Algumas empresas utilizam ferramentas para gerenciar toda a documentação do projeto, pois entendem ser primordial para o projeto e para o futuro sistema. Outro ponto importante de ter a documentação atualizada do projeto e do sistema é no tratamento da ambientação dos novos membros na equipe. Fica mais ágil e menos custoso o aprendizado quando se tem a realidade documentada.

UNIDADE 4 – REQUISITOS DE *SOFTWARES*

MÓDULO 2 – PRIMEIROS PASSOS PARA O DETALHAMENTO DOS REQUISITOS

01

1 - DETALHAMENTO DE REQUISITOS PARA PEQUENOS PROJETOS, SEGUNDO RUP 7

Vimos que a engenharia de requisitos possui várias tarefas que percorrem todo o projeto durante o seu ciclo de vida. Uma das primeiras tarefas para o levantamento de requisitos é a modelagem de requisitos, que leva a especificação dos requisitos e a representação do projeto para o*software* que será construído. Pressman utiliza os termos modelo de requisitos e modelo de análise para a mesma coisa, que são aspectos do problema a ser resolvido, sendo que a análise é uma ação que ocorre à medida que os requisitos são construídos.

Em 1979, Tom DeMarco, descreveu em seu livro sobre métodos de modelagem de requisitos o seguinte texto:

> *"Revendo reconhecidos problemas e falhas anteriores da fase de análise, sugiro que precisamos realizar os seguintes acréscimos ao nosso conjunto de metas. Os produtos da análise devem apresentar grande facilidade em sua manutenção. Isso se aplica particularmente ao Documento Alvo (especificação de requisitos de software). Problemas de tamanho devem ser tratados por meio de método efetivo de fracionamento. A especificação escrita como se fossem romances vitorianos está acabada. Elementos gráficos têm de ser usados sempre que possível. Temos de diferenciar as considerações lógicas (essenciais) das físicas (implementação)... No mínimo, precisamos... Algo que nos ajude a subdividir nossos requisitos e documentar essa subdivisão antes da especificação... Alguns meios de acompanhar e avaliar interfaces... Novas ferramentas para descrever lógica e política, algo melhor do que um texto narrativo."*

O autor apresenta uma visão do passado que ainda permanece nos dias de hoje. Vários documentos foram criados para detalhar melhor a representação dos requisitos para um projeto de *software*. Tudo depende da metodologia que se irá aplicar. As metodologias devem ser referências para os técnicos criarem seus processos de desenvolvimento. As empresas devem utilizar apenas o que é necessário e não se esquecer da simplicidade para escrever os requisitos. Requisitos complexos geram dificuldade na hora da manutenção do *software* e pode gerar a desatualização, em alguns casos os requisitos devem ser reconstruídos.

02

Analisaremos agora o conjunto de passos e de documentos de requisitos que os métodos Processo Unificado e Ágil trabalham no seu contexto, segundo o RUP. São eles:

Na sequência, apresentaremos uma breve descrição das tarefas elencadas acima. Visitando o RUP você terá mais detalhes técnicos e orientações sobre o assunto e sobre as etapas.

03

a) Captar um vocabulário comum

A finalidade dessa tarefa é definir um vocabulário comum que possa ser utilizado em todas as descrições textuais do sistema, especialmente os requisitos de *software*.

Na disciplina Requisitos, você deve definir um vocabulário comum que utilize os termos e as expressões mais frequentes do domínio de problemas. Em seguida, você deve utilizar esse vocabulário consistentemente em todas as descrições do sistema e de seus requisitos. Dessa forma, você mantém a consistência das descrições textuais e evita mal-entendidos entre os membros do projeto sobre o uso e o significado dos termos. Procure documentar o vocabulário em um glossário.

Para encontrar os termos comuns do domínio de problemas, considere os termos utilizados nos requisitos e o conhecimento geral da equipe de desenvolvimento sobre o sistema a ser criado. Concentre-se nos termos que descrevem os seguintes conceitos:

- Os **objetos de negócio**, que representam conceitos utilizados no trabalho diário da organização ou no ambiente operacional esperado do sistema. Em muitos casos, uma lista de conceitos desse tipo já existe.
- Os **objetos do mundo real**, que o sistema precisa identificar. Esses objetos ocorrem naturalmente e podem ser, por exemplo: carro, cachorro, garrafa, aeronave, passageiro, reserva ou fatura.

Exemplo: em um Sistema para Administração de Depósito, a comunicação inclui, entre outros, os itens contidos no depósito, os possíveis locais de armazenamento para esses itens e os eventos que o sistema precisa identificar. Um evento natural de um Sistema para Administração de Depósito, por exemplo, é a liberação de mercadorias no depósito. Para cada entrega, o sistema deve "memorizar" a data de

entrega, quem recebeu as mercadorias, quais mercadorias foram entregues e a quantidade de itens de cada tipo.

Glossário
Normalmente, cada termo descrito é um substantivo acompanhado de uma definição. Os termos devem estar no singular, por exemplo, "ordem" e "tarefa", não "ordens" e "tarefas". Todas as partes interessadas devem concordar com as definições dos termos.

Evento
Nesse contexto, significa um ponto no tempo ou um incidente cronológico que o sistema precisa identificar, como uma reunião ou a ocorrência de um erro.

b) Desenvolver a visão

Esta tarefa descreve como desenvolver a visão geral para o sistema, incluindo o problema a ser resolvido, os investidores chave, o escopo/limite do sistema, os recursos-chave do sistema e quaisquer restrições.

A finalidade dessa tarefa é:

- Estabelecer um acordo sobre quais problemas precisam ser resolvidos.
- Identificar investidores do sistema.
- Definir os limites do sistema.
- Descrever os principais recursos do sistema.

Ao desenvolver o documento de Visão, lembre-se do seguinte:

- Os usuários potenciais (ou atuais) do sistema irão mapear as funções dos atores humanos do sistema sendo desenvolvido.
- Geralmente, é bastante eficaz utilizar agentes para definir e descrever os limites do sistema (Consulte Tarefa: Localizar Agentes e Casos de Uso).
- As restrições reunidas nessa tarefa serão a entrada inicial para as restrições de design definidas nas Especificações Suplementares.

Etapas

- Conquistar Contrato sobre o Problema sendo Resolvido
- Identificar Investidores
- Definir os Limites do Sistema
- Identificar Restrições a serem Impostas no Sistema
- Formular a Instrução do Problema
- Definir os Recursos do Sistema

- Avaliar Seus Resultados

05

c) Desenvolver Especificações Suplementares

Essa tarefa captura os requisitos que não se aplicam a casos de uso específicos.

Em todas as atividades de requisitos, com base nos pedidos do envolvido que foram reunidos, os requisitos que não forem aplicáveis a Casos de Uso específicos serão capturados na Especificação Suplementar. Requisitos funcionais e não funcionais podem ser capturados na Especificação Suplementar.

Ao executar esta tarefa, é importante certificar-se de que todos os requisitos estejam especificados no nível de detalhe necessário, para que sejam distribuídos aos designers, testadores e escritores de documentação. Se os requisitos forem rastreados ou, caso contrário, formalmente gerenciados, certifique-se de que cada requisito esteja claramente identificado e etiquetado.

Etapas

- Capturar os Requisitos Funcionais que não São Específicos de Caso de Uso
- Capturar Qualidades do Sistema
- Capturar Restrições
- Capturar Requisitos de Conformidade
- Capturar Requisitos de Documentação

Os requisitos que estendem casos de uso (possivelmente requisitos de sistema inteiro) tendem a conduzir o desenvolvimento da arquitetura do sistema. De fato, em alguns projetos, tais requisitos podem ser significativamente mais importantes que seus equivalentes mais específicos de domínio (ou específicos de caso de uso). Por exemplo, se você estiver desenvolvendo sistemas médicos de suporte à vida, então requisito de confiabilidade serão críticos.

Desafortunadamente, tais requisitos são frequentemente difíceis de reunir e, portanto, são muitas vezes negligenciados. Essa tarefa descreve a abordagem geral para reunir esses requisitos.

06

d) Detalhar os Requisitos de *Software*

Essa tarefa descreve como detalhar os requisitos de software do sistema.

A finalidade dessa tarefa é coletar, detalhar e organizar o conjunto (pacote) de produtos de trabalho que descrevem completamente os requisitos de *software* do sistema. Detalhar os requisitos

de *software* envolve detalhamento dos casos de uso e da especificação suplementar para os requisitos que devem ser entregues no release atual.

Etapas

- Detalhar os Requisitos de *Software*
- Gerar Relatórios de Suporte
- Empacotar os Requisitos para Revisão

O Protótipo da Interface do Usuário pode ser uma origem excelente de requisitos detalhados que podem ter sido pedidos durante a retirada dos requisitos iniciais. Se o usuário aceitar o protótipo, é importante documentar explicitamente qualquer requisito detalhado que precise implementar o protótipo.

07

e) Detalhar um Caso de Uso

Esta tarefa envolve detalhar, refinar um caso de uso específico.

O escopo dessa tarefa é:

- Descrever um ou mais dos fluxos de eventos do caso de uso em detalhes suficientes para permitir que o desenvolvimento do software seja iniciado nele.
- Detalhar o caso de uso para a compreensão e satisfação do representante agente ou cliente.

Etapas

- Revisar e Refinar os Cenários
- Detalhar o Fluxo de Eventos
- Estruturar o Fluxo de Eventos
- Ilustrar Relacionamentos com Agentes e Outros Casos de Uso
- Descrever os Requisitos Especiais
- Definir o(s) Protocolo(s) de Comunicação
- Descrever Condição Prévia
- Descrever Condições Posteriores
- Descrever os Pontos de Extensão
- Avaliar Seus Resultados

08

f) Estruturar o Modelo de Caso de Uso

Esta tarefa envolve a estruturação do modelo de casos de uso, de modo a tornar os requisitos mais fáceis de compreender e de manter.

Isso inclui apurar a semelhança entre casos de usos e agentes e identificar comportamentos excepcionais e opcionais.

Etapas

- Identificar os Requisitos Comuns
- Estabelecer Relacionamentos de Inclusão entre Casos de Uso
- Estabelecer Ralações de Extensão entre Casos de Uso
- Estabelecer Generalizações entre Casos de Uso
- Estabelecer Generalizações entre Agentes
- Organize o Conteúdo do Modelo de Casos de Uso em Pacotes
- Avaliar Seus Resultados

09

g) Identificar pedidos dos fornecedores

Esta tarefa descreve como obter pedidos dos investidores do que eles gostariam que o sistema fornecesse.

A finalidade dessa tarefa é:

- Entender quem são os investidores do projeto.
- Coletar pedidos de quais necessidades o sistema deve preencher.
- Priorizar pedidos dos envolvidos.

Pedidos dos Investidores são obtidos e reunidos ativamente a partir de origens existentes para obter uma "lista de desejos" do que os diferentes investidores do projeto (clientes, usuários, patrocinadores do produto) esperam ou desejam que o sistema inclua, juntamente com informações sobre como cada pedido foi considerado pelo projeto.

Etapas

- Determinar as Origens de Requisitos
- Reunir Informações
- Conduzir Workshops de Requisitos
- Avaliar Seus Resultados

10

h) Localizar Agentes e Casos de Uso

Essa tarefa é onde os agentes e casos de uso são identificados para suportar os requisitos que estão sendo implementados. Identificar os agentes e os casos de uso explicitamente define o escopo do sistema.

A finalidade dessa tarefa é:

- Definir o escopo do sistema - o que será tratado pelo sistema e o que será tratado fora dele.
- Definir quem e o que interagirá com o sistema.
- Delinear a funcionalidade do sistema.

Há uma técnica muito bem sucedida que pode ser utilizada para localizar os agentes e os casos de uso para manter um Workshop de Caso de Uso.

Etapas

- Localizar Agentes
- Localizar Casos de Uso
- Descrever Como Agentes e Casos de Uso Interagem
- Empacotar Casos de Uso e Agentes
- Apresentar o Modelo de Casos de Uso em Diagramas
- Desenvolver uma Pesquisa de Opinião do Modelo de Casos de Uso
- Avaliar Seus Resultados

11

i) Priorizar Casos de Uso

Essa tarefa tem o objetivo de definir a prioridade dos casos de uso, de forma a decidir a sequência de desenvolvimento de cada caso. Isso quer dizer que os casos de uso arquiteturalmente significativos são identificados e priorizados.

A finalidade dessa tarefa é:

- Definir a entrada para a seleção do conjunto de cenários e casos de uso que serão analisados na iteração atual.
- Definir o conjunto de cenários e casos de uso que representam alguma funcionalidade central significativa.
- Definir o conjunto de cenários e casos de uso que possuem cobertura arquitetural substancial (que exercita vários elementos de arquitetura) ou que enfatizam ou ilustram um ponto específico, delicado, da arquitetura.

Alguns dos fatores utilizados para definir a prioridade dos casos de uso podem ser capturados como atributos. As prioridades de casos de uso resultantes podem, da mesma forma, ser capturadas como atributos dos requisitos, para que possam ser gerenciadas com eficiência.

Etapas

- Priorizar Casos de Uso e Cenários
- Documentar a Visualização de Casos de Uso
- Avaliar Seus Resultados

12

j) Revisar Requisitos

Esta tarefa descreve como revisar os produtos de trabalho dos requisitos.

O objetivo dessa tarefa é garantir formalmente que os resultados das tarefas dos requisitos estejam em conformidade com a visão que o cliente tem do sistema.

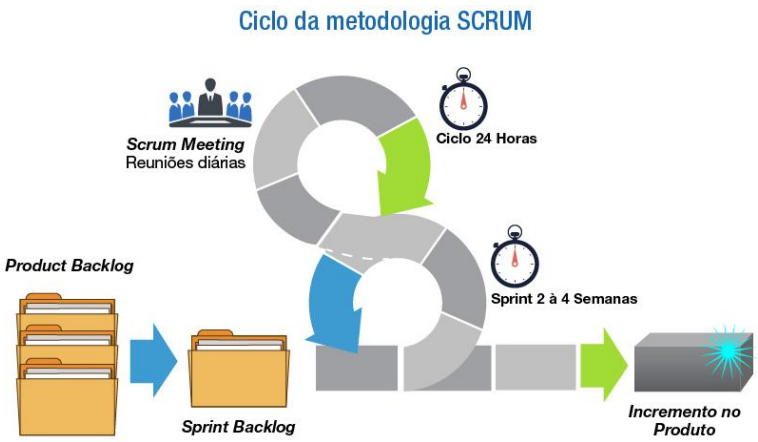
Etapas

- Recomendações Gerais
- Reuniões de Revisão Recomendadas
- Preparar Registro de Revisão e Documentar Defeitos

13

2 - Método Ágil – Passos e documentos de requisitos para projetos, segundo SCRUM

Para os métodos ágeis a quantidade de documentos e tarefas é muito menor. No caso do SCRUM veremos mais adiante Product Backlog e Sprint Backlog.



Antes de desenvolver as histórias, é importante realizar uma estimativa de complexidade (planning poker) para entregar uma história, não temos lá muitos detalhes sobre o que de fato ela faz, tampouco como a construiremos. Essa estimativa possibilita escolher que histórias irão fazer parte da Sprint.

Nesse momento, quando essa história entra numa sprint, é que começa o trabalho de levantamento de requisitos, porém, ele é específico para aquela história ou para aquela(s) funcionalidade(s). Não é um exercício longo e extenso para todo o projeto, como realizado em outras metodologias. Caso a equipe não tenha ainda o entendimento suficiente para construir, podemos utilizar as mesmas técnicas para a elicitação de requisitos já vistas em módulos passados, porém, o foco é somente a história em evidência.

14

Muitas pessoas na TI acham que o método ágil significa desprezar toda a técnica e a documentação.

Na verdade o que a metodologia propõe é um constante exercício de valor, priorização com requisitos: o importante é o que deve ser feito primeiro para retornar o valor agregado para o cliente. Se o que retorna mais valor para o cliente é o requisito com maior incerteza, devemos começar a construí-lo primeiro, para rapidamente ajudarmos o cliente a materializar a ideia da sua necessidade. Ao mesmo tempo teremos o conforto necessário para continuar a construção do *software*.

Com isso, temos dentro do horizonte de uma sprint um produto funcionando, ou parte dele, e uma materialização dos requisitos, dadas as restrições de prazos e recursos. Para a próxima sprint as alterações sempre são bem-vindas. Os requisitos são sempre revistos, são aprimorados quando necessários e o *feedback* é sempre bem vindo, utilizado para construir um melhor produto. Dessa forma, o SCRUM aproveita o aprendizado durante o projeto.

O Scrum trata dos requisitos de software através do Product Backlog e Sprint Backlog. Deve-se deixar bem claro que o método Scrum não possui definido em sua implantação um Gerenciamento de

Requisitos, como acontece com o RUP, os requisitos são apenas identificados, tratados e tornam-se funcionalidades do sistema em desenvolvimento. Mas são todos documentados.

As histórias (User Stories), artefatos que caracterizam algumas das metodologias ágeis, nesse caso o Scrum, são responsáveis por coletar os requisitos através do cliente, ou seja, é o próprio cliente quem cria as histórias. Depois elas são analisadas e tornam-se funcionalidades que são atribuídas ao Product Backlog, onde são analisadas e posteriormente ao Sprint Backlog para o início de uma Sprint.

15

3 - USE CASE X USER STORY

Existe muita discussão sobre igualdades e diferenças entre Casos de Uso e User Story, ou seja, Caso de Uso do RUP e as histórias do Scrum.

Casos de Uso são descrições não técnicas e não tecnológicas daquilo que o usuário pode esperar do sistema. A principal finalidade do Caso de Uso é capturar o comportamento requerido do sistema a partir da perspectiva do usuário final na busca de atingir uma ou mais metas desejadas, ou seja, são descrições da interação do usuário com o sistema com as ações que o usuário toma e o sistema responde até que o objetivo seja alcançado.

Os Casos de Uso são utilizados para várias funções diferentes e com várias finalidades.

Um caso de uso (*user case*) agrupa todas as sequências específicas de ações e interações entre atores e o sistema em forma de cenários, regras de negócio, requisitos de interface, requisitos de desempenho, etc. Um cliente por sua vez, não tem a capacidade de agrupar todas essas informações, ele só está interessado no produto final.

A **história (*user story*)** é uma descrição simples e curta daquilo que o gestor precisa, ou seja, que a aplicação terá que fazer, sendo aconselhável que o cliente escreva a história, ou pelo menos esteja presente quando for criada pelo analista de sistemas.

Histórias podem ser particionadas em outras “mais simples” buscando aumentar a velocidade ou separar o trabalho de implementação por alguma razão.

A uma história é uma menor quantidade de informações necessárias para que o cliente defina um caminho através de um sistema. Um caso de uso é uma descrição completa de uma sequência de interações; ele não é normalmente um passo ou atividade individual em um processo. Pode haver mais de um caso de uso para uma funcionalidade.

16

Para qualquer metodologia usada existe a necessidade de um levantamento de requisitos. Se esse levantamento é documentado ou não, é uma escolha à parte. Vimos nos módulos anteriores a fase de levantamento de requisitos e seus métodos de elicitação. Importante que exista uma lista de requisitos para qualquer metodologia que será utilizada.

	Use Case	User Story
Definição	São descrições não técnicas da iteração do usuário com o sistema, as ações que o usuário toma e o sistema responde até que o objetivo seja alcançado.	São descrições não técnicas, simples e curtas daquilo que o sistema terá que fazer.
Qual a sua utilização	Servem como um protótipo para que todos saibam o que esperar do sistema.	Servem para que a equipe possa estimar o que tem que fazer e quanto esforço será necessário.
Quem cria o artefato	Cliente ou Analista	Cliente
Tempo de vida dos requisitos	Os requisitos que dão origem ao use case são mantidos até o final do projeto.	Os requisitos utilizados numa user story são excluídos do projeto depois do teste de aceitação.
Como tratam os requisitos	Use Cases são exemplos funcionais dos requisitos e são tarefas "macro" do sistema.	User Stories são levantadas diretamente dos requisitos, são tarefas tão "micro" quanto necessário.

Comparação do caso de uso (use case) com a história (user story)

Portanto, pode-se, sim, usar as duas, mas a equipe de desenvolvimento deve escolher como usá-las. Uma mistura onde user stories são utilizadas para incrementar e desenvolver e os casos de uso, simples descritivos, utilizados para a documentação, ajudando na rastreabilidade dos requisitos. Pode-se utilizar modelos gráficos, diagramas de caso de uso, de atividades e classes, para uma comunicação eficiente com o cliente e obter um *feedback* mais rápido, melhor e recebendo como resposta User Stories.

17

4 - Críticas sobre a documentação em Métodos Ágeis

Muitas pessoas criticam Métodos Ágeis afirmando que não existe documentação, que não utilizam por esse motivo. Essas pessoas possivelmente nunca utilizaram.

Os profissionais que utilizam, sendo os mais experientes até os novatos em método ágil, sabem que essa história de ágil não documentar é um equívoco. O Scrum também realiza a documentação das tarefas realizadas, mas de uma maneira diferente, uma maneira mais ágil. O caso é que muitas pessoas acham que documentação se limita àquela papelada toda gerada manualmente ou por ferramentas. A documentação de um programa pode ser mais simples, basta ter criatividade para isso.

Uma diferença entre as metodologias ágeis e as metodologias tradicionais, é que enquanto as tradicionais buscam documentar tudo (ou quase tudo) o que acontece, ou o que é feito, para manter uma organização, ou para futuros programadores que vão entrar na equipe, as metodologias ágeis documentam apenas até agregar valor.

Provavelmente nós, Analistas de Sistemas, iremos nos confrontar com diversos métodos de desenvolvimento com suas diversas documentações. A melhor metodologia é aquela que você sabe utilizar, a melhor documentação é aquela atualizada, em que você entenda o que o sistema faz.

18

RESUMO

Projetos grandes e pequenos possuem várias tarefas ao longo de sua execução. As tarefas que compõem o detalhamento de requisitos, para pequenos projetos, percorrem todo o projeto durante o seu ciclo de vida. O RUP 7 apresenta o detalhamento das tarefas para os projetos grandes e pequenos.

Já o método ágil tem a finalidade de executar projetos em tempo menor que os projetos tradicionais. O SCRUM, que é um método ágil, trata dos requisitos de maneira rápida, enquanto o método estruturado tem toda uma estrutura de documentação. Esses documentos permeiam todo o ciclo de vida do projeto.

Percebe-se que as duas metodologias apresentadas têm maneiras diferentes de tratar os requisitos, o RUP utiliza-se de Use Case (Caso de Uso) enquanto o SCRUM usa User Story (História). O Scrum trata dos requisitos de maneira rápida, enquanto o método estruturado tem toda uma estrutura de documentação.

A documentação dos Métodos Ágeis é muito criticada por muitos e elogiada por outros. Ela preocupa-se apenas com as informações essenciais para o projeto.

Mesmo com as diferenças, ainda existem pontos em comum nos artefatos dos métodos, que podem torná-los complementares se usados da maneira correta. O importante é sempre documentar os requisitos, independente do método a ser utilizado.

UNIDADE 4 – REQUISITOS DE *SOFTWARES*

MÓDULO 3 – TIPOS DE REQUISITOS DE *SOFTWARE*

01

1 - REQUISITOS: NECESSIDADES E CARACTERÍSTICAS

Vimos anteriormente o detalhamento sobre a disciplina de requisitos, que representa o ponto de partida para toda a definição do sistema, sendo fator decisivo no desenvolvimento do projeto. Estudamos também que a engenharia de requisitos determina de que forma e em quais condições o sistema deve funcionar para resolver um problema do usuário, assim como a capacidade que deva alcançar e as restrições que o sistema poderá ter.

Sabemos que o processo de produção de *software* está amarrado à definição clara de qual produto construir. E esta definição baseia-se no conhecimento do problema e na viabilização de oportunidade de negócio com o uso de tecnologia da informação.

Contudo, um dos grandes problemas enfrentados pelos analistas de requisitos é provocado simplesmente por deixarem de questionar dois aspectos ao usuário:

- sua expectativa em relação ao sistema e
- seus parâmetros de desempenho.

O foco se reduz ao que ele *quer que o sistema faça* e isso pode levar à frustação do usuário quando se depara com o resultado final do desenvolvimento.É fundamental, portanto, identificar as **necessidades** do usuário e as **características** do negócio, de modo que se tenha um detalhamento de requisitos que leve a uma maior eficácia do sistema.

02

Abaixo os conceitos de Necessidade e Características, segundo RUP:

Necessidades são reflexões sobre os problemas do negócio, sejam de pessoas ou operacionais, que devem ser estruturadas para justificar o desenvolvimento de um novo sistema.

É comum os *stakeholders* terem dificuldade de enumerar suas necessidades reais. Em geral, limitam-se a fazer comentários como:

"Preciso aumentar a produtividade neste setor ou não terei promoção este ano.";

"Devemos identificar nossos melhores clientes."

Características são os serviços observáveis por meio dos quais o sistema satisfaz uma ou mais necessidades dos interessados.

A característica, portanto, é uma função do sistema, está no espaço da solução, uma vez que descreve como o sistema deve satisfazer as necessidades. Exemplos de características:

"O sistema deve ser ágil para poder mostrar, sem constrangimentos ou demora, os relatórios aos clientes."

"A compra de produtos deve ser completamente automatizada, desde a consulta até o pagamento."

03

2 - REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS

Vale lembrar que requisito é uma **condição** ou capacidade com a qual o sistema deve estar de acordo, ou seja, sua exigência deve ser satisfeita. Se a condição é produzir algo, dizemos que o requisito é **funcional**. Se a condição é caracterizar algo (seja propriedade, atributo, comportamento, restrição etc.), dizemos que o requisito é **não funcional**.

Requisitos Funcionais são requisitos que especificam como o sistema interage com o contexto na sua

volta, ou seja, são as funcionalidades que o projeto terá.

Segundo Sommerville requisitos funcionais são declarações de serviços que o sistema deve prover, descrevendo o que **o sistema deve fazer**. As especificações dos requisitos funcionais devem determinar o que se espera que o *software* faça, sem preocupar em como ele irá fazer.

Assim, requisito funcional representa o funcionamento perceptível do sistema pelos usuários, por exemplo: telas, relatórios, informações, fluxo de negócio etc.

Requisitos Não Funcionais: são requisitos que expressam atributos de qualidade para a solução e estão ligados diretamente aos parâmetros de funcionamento do sistema.

Segundo Sommerville, requisitos não funcionais descrevem restrições sobre os serviços ou funções oferecidos pelo sistema. Assim, os requisitos não funcionais são muito importantes para o projeto, pois definirão a arquitetura necessária para que o sistema funcione adequadamente.

Sommerville destaca que os requisitos não funcionais têm origem nas necessidades dos usuários, em restrições de orçamento, em políticas organizacionais, em necessidades de interoperabilidade com outros sistemas de *software* ou *hardware* ou em fatores externos, como regulamentos e legislações. Assim, os requisitos não funcionais podem ser classificados quanto à sua origem, conforme veremos a seguir.

04

Existem diversas **classificações de requisitos não funcionais**. Sommerville classifica-os em:

• Requisitos de produto

Especificam o comportamento do produto (sistema). Referem-se a atributos de qualidade que o sistema deve apresentar, tais como confiabilidade, usabilidade, eficiência, portabilidade, manutenibilidade e segurança.

• Requisitos organizacionais

São derivados de metas, políticas e procedimentos das organizações do cliente e do desenvolvedor. Incluem requisitos de processo (padrões de processo e modelos de documentos que devem ser usados), requisitos de implementação (linguagem de programação a ser adotada), restrições de entrega (tempo para chegar ao mercado - *time to market*, restrições de cronograma etc.), restrições orçamentárias (custo, custo-benefício) etc.

• Requisitos externos

Referem-se a todos os requisitos derivados de fatores externos ao sistema e seu processo de desenvolvimento. Podem incluir requisitos de interoperabilidade com sistemas de outras organizações, requisitos legais (tais como requisitos de privacidade) e requisitos éticos.

No que se refere aos requisitos não funcionais de produto, podemos relacionar as propriedades emergentes do sistema como um todo, ou seja, propriedades que não podem ser atribuídas a uma parte específica do sistema, mas que, ao contrário, só aparecem após a integração de seus componentes, tal como **confiabilidade**. Porém, algumas vezes, essas características podem estar associadas a uma função específica ou a um conjunto de funções. Por exemplo, uma função pode ter restrições de desempenho, enquanto outras funções não apresentam tal restrição.

Vale destacar que há outras classificações de requisitos, embora a classificação em requisitos funcionais e não funcionais seja a mais amplamente aceita.

05

Os requisitos devem ser redigidos de modo a serem passíveis de entendimento pelos diversos interessados do projeto e do sistema. Os clientes, usuários finais e desenvolvedores são todos interessados nos requisitos, mas com expectativas diferentes.

Enquanto desenvolvedores e usuários finais têm interesse em detalhes técnicos, clientes requerem descrições mais abstratas.

Assim, Sommerville sugere dois **níveis de descrição de requisitos**:

Requisitos de Usuário	Requisitos de Sistema
São declarações em linguagem natural, acompanhadas de diagramas intuitivos de quais serviços são esperados do sistema e das restrições sob as quais ele deve operar. Devem estar em um nível de abstração mais alto, de modo que sejam compreensíveis pelos usuários do sistema que não possuem conhecimento técnico.	Definem detalhadamente as funções, serviços e restrições do sistema. São versões expandidas dos requisitos de usuário, usadas pelos desenvolvedores para projetar, implementar e testar o sistema. Os requisitos de sistema são mais detalhados. Como as especificações em linguagem natural são insuficientes e para especificá-los, notações mais especializadas devem ser utilizadas, como desenhos e gráficos.

06

No ciclo de vida de um projeto existem níveis de descrição de requisitos, tudo depende do propósito e do momento do projeto. O levantamento preliminar dos requisitos (requisitos de usuário) é elaborado nos estágios iniciais do projeto, servindo de base para um entendimento entre clientes, gestores e desenvolvedores sobre o que o sistema irá realizar.

Esses requisitos são, normalmente, usados como base para a contratação e o planejamento do projeto.

Requisitos de sistema, por sua vez, devem ser elaborados de acordo com a necessidade dos requisitos funcionais, onde se captura detalhes importantes, técnicas, para o bom funcionamento do sistema e testes complementares.

Não se deve esquecer que **requisitos de sistema são derivados dos requisitos de usuário**. Não se pode pensar somente nos requisitos de usuário. Em vários casos é comum esquecer os requisitos de sistema, quando isso acontece, muitas vezes o sistema não roda, ou seja, não faz o que deveria fazer, que é ajudar o cliente. Nesse caso, gasta-se muito dinheiro corrigindo e ajustando o sistema para que ele tenha os requisitos de sistema necessários para funcionar adequadamente como solicitado pelo cliente.

07

Os **requisitos não funcionais** possuem atributos, que são importantes para determinar o tamanho e a complexidade do sistema que está sendo construído.

Listamos abaixo alguns desses atributos importantes:

- Aspectos de desempenho;
- Interfaces com o usuário;
- Confiabilidade;
- Segurança;
- Manutenibilidade;
- Portabilidade.

Todos os itens destacados acima são importantes e críticos, pois eventuais erros na elicitação desses requisitos constituem os mais caros e difíceis de corrigir, uma vez que um sistema tenha sido implementado.

Aspectos de desempenho
O quão rápido o sistema deverá ser, um bom exemplo é o sistema de loterias, que é acessado nas lotéricas para cadastrar os jogos de milhões de pessoas, o sistema deverá aguentar esses acessos e com certa rapidez de processamento.

Interfaces com o usuário
A facilidade de uso do sistema por parte do usuário depende de sua interface, devendo ser mais simples ou mais complexa, dependendo do público que irá usar o aplicativo.

Confiabilidade
É a probabilidade de a aplicação não causar falhas durante um determinado período, sob condições específicas, um bom exemplo são os sistemas que atuam em robôs cirúrgicos.

Segurança
Dependendo do aplicativo, ele deverá criptografar todas as mensagens de comunicação que trafegam na rede.

Manutenibilidade
É a capacidade de realizar as modificações e evoluções no aplicativo.

Portabilidade
O aplicativo deve ser usado em mais de uma plataforma: PC e Mac.

08

Documentos importantes

Uma vez que requisitos de usuário e de sistema têm propósitos e público alvo diferentes, é útil descrevê-los em documentos diferentes. Pfleeger, em seu livro de Engenharia de Software, sugere que dois tipos de documentos de requisitos sejam elaborados:

Documento de Definição de Requisitos, ou Documento de Requisitos	Documento de Especificação de Requisitos
Deve ser escrito de maneira que o cliente possa entender, na forma de uma listagem do que o cliente espera que o sistema proposto faça. É um consenso entre o cliente e o desenvolvedor sobre o que o cliente quer.	Redefine os requisitos de usuário em termos mais técnicos, apropriados para o desenvolvimento de software, sendo produzido por analistas de requisitos.

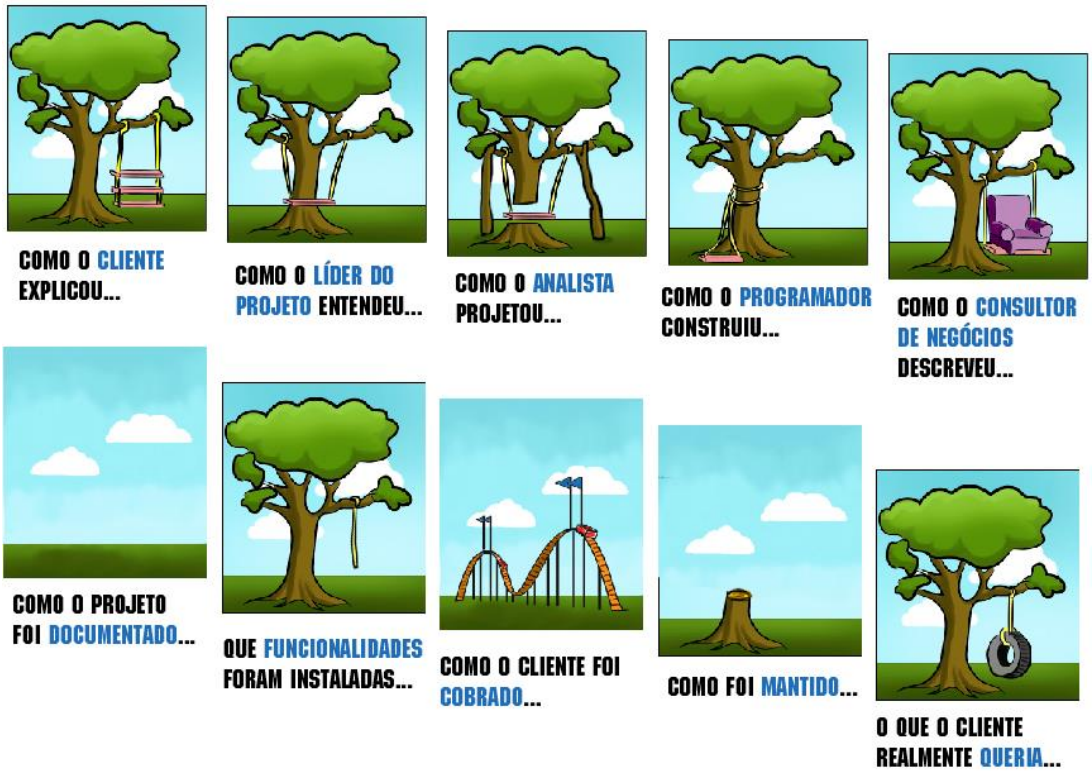
Esses documentos apoiam o programador a implementar as funcionalidades do software e um complementa o outro. Conforme a metodologia utilizada, esses documentos podem variar de nome, mas permanecem com a mesma essência.

09

Exemplo de requisitos Funcionais e Não Funcionais

A imagem abaixo é famosa e já apareceu, inclusive em outro momento da nossa disciplina. Ela representa um exemplo simples de uma solicitação de usuário, que possui requisito funcional e não funcional.

O cliente solicitou um balanço com corda na árvore para balançar uma criança. A funcionalidade ou requisito funcional seria balançar uma criança. No caso dos requisitos não funcionais, poderíamos citar como exemplos o peso máximo que a corda poderia aguentar, a segurança do balanço para que a criança não se machuque, a altura, a qualidade do balanço e a usabilidade.



10

3 - RASTREABILIDADE NOS PROJETOS E SISTEMAS

Na literatura existem vários textos, metodologias que detalham a importância da rastreabilidade dos requisitos. Vimos anteriormente que a rastreabilidade dos requisitos é muito importante, pois permite que a equipe saiba exatamente o que será afetado em uma alteração solicitada pelo gestor, facilitando identificar o custo e prazo de toda a mudança.

A rastreabilidade pode ser definida como a capacidade de se acompanhar a vida de um requisito em ambas as direções do processo de *software* e durante todo o ciclo de vida.

Atualmente entende-se que mudanças dos requisitos ao longo do desenvolvimento do aplicativo é comum, mas que quanto mais cedo identificar essas mudanças, menos custo teremos, pois alteração de um requisito pode implicar mudanças em vários requisitos. Se tivermos um mecanismo, uma ferramenta para identificação e gerenciamento dessas mudanças, provavelmente teremos menos problemas.

Imaginemos um projeto de um carro em que já exista o desenho e até o protótipo construído. Se o gestor resolver alterar as dimensões do porta-malas, o que vai acontecer? Visivelmente temos em nossa

mente as alterações a serem feitas. A princípio será necessário alterar a porta do porta-malas, depois verificar as adaptações na estrutura do carro, o amortecedor do porta-malas, o tamanho do vidro do porta-malas, dentre outras mudanças.

E em um projeto de sistema, como seria? Como controlar as mudanças em um sistema?

A **Gerência de Requisitos** responde essas perguntas. A princípio é necessário que todo projeto possua algum tipo de rastreabilidade que possa haver segurança na hora da mudança. Algo que informe o que será afetado com uma simples alteração. A Gerência de Requisitos trata de um aspecto fundamental e crítico em qualquer processo de *software*, pois procura estabelecer uma visão comum entre o cliente e a equipe do projeto em relação aos requisitos.

11

As atividades ligadas ao gerenciamento de requisitos incluem o **controle de mudanças**, que é o registro e análise das solicitações e alterações dos requisitos já definidos. Indispensável à tarefa de gerenciar os requisitos, o rastreamento dos requisitos é vital para o bom andamento do projeto e do sistema. Vimos que projeto é algo novo, algo que está em processo de construção. O sistema é algo já construído, pronto, portanto, infere-se que todos os requisitos sejam conhecidos, os funcionais e não funcionais.

O rastreamento de requisitos é utilizado para prover relacionamento entre os requisitos, arquitetura e implementação final do projeto, viabilizando eventuais alterações em projetos e em qualquer sistema. A rastreabilidade na literatura tem sido identificada como um fator de**qualidade**. Muitos analistas acreditam que realizar a rastreabilidade é uma das tarefas mais importantes para o desenvolvimento do projeto e para a manutenção de um sistema.

A rastreabilidade estendida para outras informações do projeto, além dos requisitos, pode ser uma **técnica de relacionamento** entre outras disciplinas do projeto. Para os sistemas, a rastreabilidade possibilita uma análise de impacto nas solicitações de mudanças no sistema.

A **análise de impacto** é realizada para verificar todos os itens que serão afetados e ocorre quando o cliente ou o gestor do sistema solicita uma alteração no sistema.

Muitas empresas utilizam um **artefato** para registrar essa análise do impacto na mudança. Esse artefato contém:

- Uma **introdução**, descrevendo o objetivo do documento;
- A **origem** da mudança ou desvio;
- Uma **descrição** detalhada da mudança ou desvio.

12

Uma análise de impacto avalia:

- O impacto no escopo do produto;,

- O impacto no escopo do projeto, com referências ao acordo vigente e o acordo proposto, com o impacto nas estimativas e com a evolução no escopo do projeto.

Além disso, a análise de impacto gera:

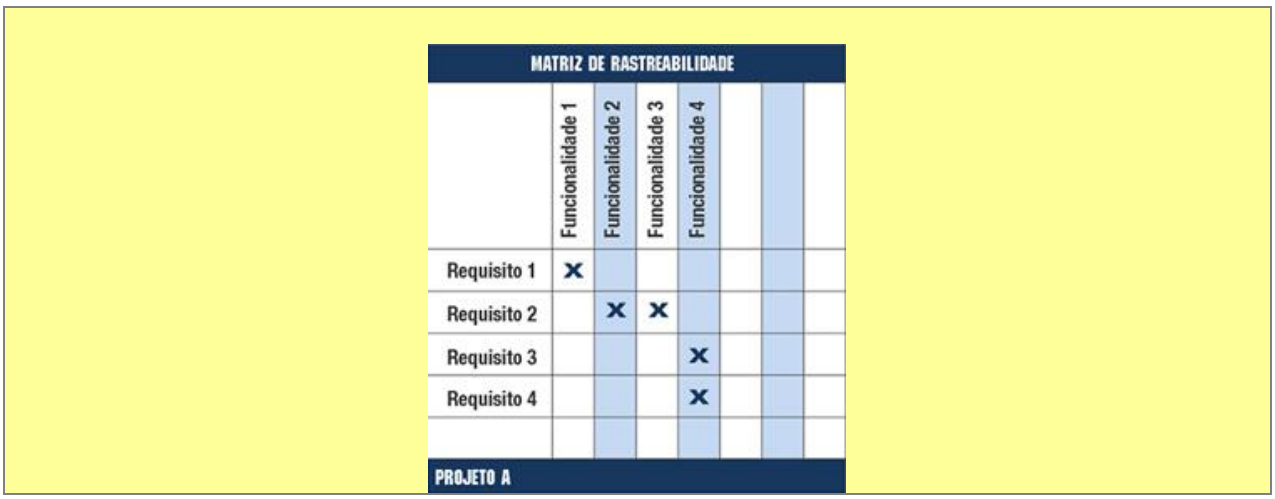
- Um parecer técnico;
- Identificação dos responsáveis pela mudança ou desvio;
- Um comparativo entre a visão do cronograma vigente e o cronograma previsto, no caso de projetos.

E o principal para uma análise de impacto, é a **identificação dos requisitos** que serão afetados, para o que pode ser utilizada uma tabela como esta:

Identificação dos <*Requisitos ou Casos de Uso*> afetados pela mudança				
Código	**Descrição**	**Situação**		
		Incluído	**Alterado**	**Excluído**
TOTAIS APURADOS NA DEMANDA				
		Total de itens:		

No quadro acima devem ser anotados os requisitos ou casos de uso que serão incluídos, alterados ou excluídos. Quando necessário, podem ser identificados outros produtos de *software* afetados pela mudança, tais como: Modelo de Dados, Especificações Técnicas, Casos de Testes, e outros. Para um preenchimento rápido e correto é necessário que a equipe tenha a matriz de rastreabilidade preenchida.

Matriz de rastreabilidade
A matriz de rastreabilidade de requisitos é destinada a apoiar as atividades do ciclo de vida do projeto, criando automaticamente elos de rastreabilidade entre os requisitos. A matriz de rastreabilidade é uma tabela que liga os requisitos às suas origens e os rastreia durante o projeto em desenvolvimento. Segue exemplo abaixo:



13

4 - FERRAMENTAS PARA RASTREABILIDADE

Algumas ferramentas de mercado auxiliam os projetos e sistemas na rastreabilidade, análise de impacto, planejamento e no gerenciamento. Vejamos algumas.

a) Plataforma Jazz (RTC, RRC, RQM)

Segundo a IBM o Rational Team Concert (RTC) é uma ferramenta de colaboração de equipe que é construída em uma plataforma escalável e extensível. O Rational Team Concert usa o aplicativo Change and Configuration Management (CCM) para fornecer recursos que integram tarefas de projeto de desenvolvimento, incluindo planejamento de iteração, definição de processo, gerenciamento de mudanças, rastreamento de defeitos, controle de fonte, automação da compilação e relatório.

O RTC torna fácil trocar informações diretamente no contexto de seu trabalho. Se um pedido de aprimoramento é alterado, você e os outros membros são notificados da alteração automaticamente. Você pode referenciar a alteração em sessões de bate-papo e vincular com artefatos. Os acionistas de negócios também podem ser automaticamente informados sobre o status das alterações de tarefas que os interessam.

Diversas visualizações possibilitam que você compartilhe informações com a equipe. Você pode rastrear atividades da equipe, apresentar informações em mais detalhes ou configurar quais informações estão visíveis a qualquer momento.

Muitos aspectos do ciclo de vida de desenvolvimento do *software* estão integrados, incluindo planejamento agile, definição de processo, controle de origem, acompanhamento de defeito, gerenciamento de compilação e relatórios. Cada um destes aspectos é integrado em um único ambiente. Você pode rastrear e gerenciar os relacionamentos entre artefatos, promover processos de desenvolvimento eficientes e reunir informações do projeto de maneira automática e reservada.

14

b) Gerenciamento de Mudanças

O principal recurso do gerenciamento de mudanças são **itens de trabalho**, que controlam e coordenam tarefas, incluindo histórias, defeitos, itens de plano e tarefas ordinárias. Os itens de trabalho e o processo de fluxo de trabalho que seguem podem ser customizados para se adequar ao seu projeto. Também é possível integrar itens de trabalho com outros sistemas de gerenciamento de mudanças, para obter um diagrama de fluxo de trabalho interativo de itens de trabalho em uso.

c) Planejamento

O recurso de planejamento fornece ferramentas para ajudar no planejamento, rastreamento e balanceamento de carga de trabalho de liberações e iterações para projetos inteiros, para equipes que estão nesses projetos e para desenvolvedores individuais. Os planos são acessíveis a todos da equipe e podem alterar durante o curso da iteração para refletir a posição e direção da equipe. Para obter um diagrama de fluxo de trabalho interativo do planejamento em uso.

d) Gerenciamento de configuração de *software*

O sistema de controle de origem integrado é baseado no componente e construído na plataforma Jazz. Ele possui forte suporte a desenvolvimento paralelo e ágil e equipes distribuídas geograficamente. Ele se integra completamente ao controle de defeitos, construções e automação centralizada em processo.

e) Relatório

O componente Jazz Team Reports fornece um reconhecimento de ações, comportamentos e progresso de uma equipe ou de um projeto. A visualização de dados sobre o processo de desenvolvimento de *software* pode tornar determinadas tendências mais acessíveis, quando elas podem então ser ocultadas ou obscurecidas. Disponibilizando essas informações em uma visão rápida, os relatórios podem permitir a tomada de decisão efetiva.

f) Painéis

Um painel é um componente da Web destinado a fornecer informações sobre o status do projeto em uma visão rápida. Ele fornece a opção de*drill down* para obter informações mais completas. Eles também representam o ponto de integração para os dados fornecidos por todos os componentes do Jazz.

15

RESUMO

Requisitos Funcionais e Não funcionais estão presentes em todos os sistemas desenvolvidos e projetos em desenvolvimento. As necessidades do usuário e comportamentos que o sistema deve possuir fazem parte desses tipos de requisitos. Os requisitos funcionais são todas as funcionalidades que o sistema deverá ter, como as consultas e cadastros. Já os requisitos não funcionais prezam pela sustentabilidade do aplicativo e outros aspectos, como segurança, acessos etc. Espera-se dos requisitos não funcionais

que eles possuam Qualidade, Privacidade, Confiabilidade, Éticos, Segurança, Eficiência, Acessibilidade, Eficiência, Usabilidade, Performance, Entrega, Escalabilidade, Suporte e Manutenção, Legais, Portabilidade.

A rastreabilidade é um fator importante para o bom gerenciamento dos requisitos, sendo que o ideal é que se mantenha todo o projeto com rastreabilidade. O uso de ferramentas é importante para facilitar essa rastreabilidade e todo o planejamento e gerenciamento do projeto. São exemplos de ferramentas que permitem rastrear os requisitos: RTC, RRC e RQM.

UNIDADE 4 – REQUISITOS DE *SOFTWARES*

MÓDULO 4 – DETALHAMENTO DOS REQUISITOS DE *SOFTWARE*

01

1 - DETALHAMENTO DOS REQUISITOS DE SOFTWARE: DOCUMENTO DE VISÃO

Trataremos agora do detalhamento inicial dos requisitos, os quais serão transformados em códigos e funções gerando um software utilizável pelo cliente.

O momento inicial de um projeto no processo unificado é consolidado em um documento que elenca quase tudo o que será realizado no projeto, o **documento de visão**. Dizemos “quase tudo” porque pode haver mudanças ao longo do projeto, o que saberemos apenas no futuro.

O dono do produto ou analista de negócios trabalha com as partes interessadas para desenvolver o documento de visão e artefatos relacionados, os quais determinam a finalidade e as características do produto.

A IBM, detentora do RUP, destaca que o **documento de visão** define o escopo de alto nível e o propósito de um programa, produto ou projeto.

Uma instrução clara do problema, solução proposta e os recursos de alto nível do produto ajudam a estabelecer expectativas e a reduzir consideravelmente os riscos. Esta ajuda é proporcionada por meio da divulgação do conteúdo do documento de visão a todos aqueles que estejam integrados no sistema.

Esse documento deve proporcionar aos participantes do projeto uma compreensão clara e abrangente sobre o sistema que se pretende desenvolver. O artefato vai responder à equipe perguntas como: “POR QUE esse projeto existe?” “O QUE é esse projeto?”, “QUEM está envolvido nesse projeto?”. Essas perguntas estão reunidas num formato apropriado, nos itens que se seguem.

02

2 - ESTRUTURA DO DOCUMENTO DE VISÃO - INTRODUÇÃO

O documento de visão pode até assustar um pouco, visto que contém inúmeras seções. Entretanto, os templates do RUP servem, única e exclusivamente, como apoio, podendo ser copiados em sua forma original, mas não significa que devam ser contempladas todas as seções.

Veremos, a seguir, a estrutura do documento de visão com descrição e particularidades necessárias para termos entendimento de como será o futuro sistema. Cada tópico pode ou não ser utilizado, tudo depende do produto que se deseja construir.

a) Introdução - a introdução fornece uma visão geral de todo o documento de visão. Ela inclui o propósito, escopo, definições, acrônimos, abreviações, referências e visão geral de todo o documento.

- **Propósito** - determina o propósito deste documento de visão.
- **Escopo** - descreve brevemente o escopo deste documento de visão, incluindo a quais programas, projetos, aplicativos e processos de negócios o documento está associado. Inclui qualquer outra coisa que este documento afete ou influencie.
- Definições, acrônimos e abreviações;
- Referências;
- Visão geral.

Definições, acrônimos e abreviações
Define todos os termos, acrônimos e abreviações necessários para interpretar a visão corretamente. Essas informações podem ser fornecidas por referência ao glossário do projeto, que pode ser desenvolvido online no repositório de requisitos do projeto.

Referências
Lista todos os documentos aos quais o documento de visão faz referência. Identifique cada documento por título, número de relatório (se aplicável), data e organização de publicação. Especifique as origens a partir das quais os leitores podem obter as referências; as origens estão disponíveis de maneira ideal no repositório de requisitos ou em outros repositórios online. Essas informações podem ser fornecidas por referência para um apêndice ou para outro documento.

Visão geral
Descreve o conteúdo do documento de visão e explica como o documento é organizado.

03

b) Oportunidade de negócios - descreve resumidamente a oportunidade de negócios que é tratada por este projeto.

c) Instrução do problema - sintetiza o problema que este projeto pretende resolver. Use as instruções a seguir como um modelo, fornecendo detalhes do projeto para substituir os elementos entre parênteses:

"O problema de (descreva o problema) afeta (as partes interessadas afetadas pelo problema). O impacto do problema é (qual é o impacto do problema). Uma solução bem sucedida incluiria (lista alguns principais benefícios de uma solução bem sucedida)."

d) Instrução de posição do produto - descreve uma instrução geral abreviada no nível mais alto, a posição exclusiva que o produto pretende preencher no mercado de trabalho. Use as instruções seguintes como um modelo, fornecendo detalhes do projeto para substituir os elementos entre parênteses:

"Para o (cliente alvo) quem (instrução da necessidade ou oportunidade). O (nome do produto) é uma (categoria do produto) que (instrução do principal benefício, isto é, o motivo convincente para comprar). De outro modo (principal alternativa competitiva), nosso produto (instrução da principal diferenciação)."

Uma instrução de posição do produto comunica o intento do aplicativo e a importância do projeto para todas as partes interessadas.

04

3 - DESCRIÇÕES DA PARTE INTERESSADA E DO USUÁRIO

Para fornecer produtos e serviços para atender às necessidades dos *stakeholders*, você deve identificar e envolver todas as partes interessadas como parte do processo de definição dos requisitos. É importante também identificar os **usuários** do sistema e assegurar que estejam representados adequadamente de modo que suas necessidades sejam descritas e contempladas no documento.

Esta seção fornece um perfil das partes interessadas e usuários que estão envolvidos no projeto, bem como a descrição dos principais problemas que as partes interessadas e os usuários esperam que a solução proposta deva atender. Contudo, esta seção não descreve solicitações ou requisitos específicos, esses itens estão contemplados em um artefato separado.

a) Demográficos de mercado – neste item, você deve resumir os principais demográficos de mercado que motivam as decisões sobre o produto. Descreva e posicione os segmentos do mercado alvo. Faça uma estimativa do tamanho e do crescimento do mercado utilizando o número de usuários potenciais ou a quantia em dinheiro que seus clientes gastam tentando atender às necessidades que seu produto ou aprimoramento poderia suprir. Reveja as principais tendências e tecnologias do segmento de mercado.

Em suma, este item deve responder as seguintes perguntas estratégicas:

- Qual é a reputação de sua organização nesses mercados?
- Qual você gostaria que fosse?
- Como esse produto ou serviço suporta suas metas?

05

b) Resumo da parte interessada – liste aqui todas as partes interessadas identificadas. Lembre-se de que nem todos os interessados são usuários finais. Para cada tipo de parte interessada, forneça as informações a seguir:

- Nome.
- Representa;
- Função.

c) Resumo do usuário - enumere todos os tipos de usuários identificados. Para cada tipo de usuário, forneça as seguintes informações:

- Nome;
- Descrição;
- Parte interessada.

d) Ambiente do usuário – detalhe aqui o ambiente de trabalho do usuário alvo. Seguem algumas sugestões:

- Quantas pessoas estão envolvidas na conclusão da tarefa? Está sendo alterado?
- Quanto tempo leva um loop de tarefa? Quanto tempo os usuários gastam em cada atividade? Está sendo alterado?
- Quais restrições de ambiente exclusivas afetam o projeto? Por exemplo, os usuários requerem dispositivos remotos, trabalham externamente ou trabalham durante as viagens?
- Quais plataformas de sistema estão em uso atualmente? Existem plataformas futuras planejadas?
- Que outros aplicativos estão em uso? Seu aplicativo precisa se integrar a eles?

Nesta seção, você pode incluir extrações do modelo de negócio para descrever a tarefa e os trabalhadores envolvidos.

Nome
Nome do tipo da parte interessada.

Representa
Escreva brevemente que pessoas, equipes ou organizações esse tipo de parte interessada representa.

Função
Descreva brevemente a função que esse tipo de parte interessada desempenha no esforço de desenvolvimento.

Nome
Nome do tipo de usuário.

Descrição Descreva de forma breve o relacionamento desse tipo de usuário com o sistema que está em desenvolvimento.

Parte interessada Identifique que tipo de parte interessada representa esse tipo de usuário.

06

e) Perfis das partes interessadas – deve ser descrito neste item cada parte interessada no projeto. Lembre-se de que os tipos de partes interessadas podem ser usuários, departamentos estratégicos, departamentos jurídicos ou de conformidade, desenvolvedores técnicos, equipes de operação, entre outros. Um perfil completo abrange os tópicos abaixo, portanto, é necessário preencher uma tabela para cada tipo de parte interessada.

Representante	Determine quem representa a parte interessada para o projeto (essa informação será opcional se estiver documentada em algum outro lugar). Insira os nomes dos representantes.
Descrição	Descreva de forma breve o tipo de parte interessada.
Tipo	Qualifique o conhecimento da parte interessada, como "usuário avançado", "especialista em negócios", ou "usuário informal". Essa designação pode sugerir a experiência técnica e o grau de sofisticação.
Responsabilidades	Liste as principais responsabilidades da parte interessada no sistema em desenvolvimento; liste seus interesses como uma parte interessada.
Critérios de Sucesso	Determine como a parte interessada define o sucesso. Como a parte interessada é recompensada?
Envolvimento	Descreva de que forma a parte interessada está envolvida no projeto. Onde for possível, relate o envolvimento nas funções do processo (por exemplo, uma parte interessada pode ser um revisor de requisitos).
Entregas	Identifique as entregas adicionais que a parte interessada requer. Esses itens podem ser entregas do projeto ou saída a partir do sistema em desenvolvimento.
Comentários ou Problemas	Identifique os problemas que interferem no processo ou resultado e quaisquer outras informações relevantes.

07

f) Perfis do usuário - descreve cada usuário do sistema aqui, preenchendo a seguinte tabela para cada tipo de usuário. Lembre-se que os tipos de usuário podem ser especialistas e novatos; por exemplo, um especialista pode precisar de uma ferramenta sofisticada e flexível com suporte para várias plataformas, enquanto um novato pode precisar de uma ferramenta que seja fácil de usar. Um perfil completo abrange esses tópicos para cada tipo de usuário:

- Representante;
- Descrição;
- Tipo;
- Responsabilidades;
- Critérios de Sucesso;
- Envolvimento;
- Entregas;
- Comentários ou Problemas.

Representante
Indica quem representa o usuário para o projeto. (Essa informação será opcional se estiver documentada em algum outro lugar.) Esse representante, geralmente refere-se à parte interessada que representa o conjunto de usuários; por exemplo, Parte Interessada: Parte Interessada1.

Descrição
Descreve brevemente o tipo de usuário.

Tipo
Qualifica o conhecimento do usuário, como "usuário avançado" ou "usuário informal." Essa designação pode sugerir a experiência técnica e o grau de sofisticação.

Responsabilidades
Lista as principais responsabilidades do usuário com respeito ao sistema; por exemplo, determina quem captura os detalhes do cliente, produz relatórios e coordena trabalho etc.

Critérios de Sucesso
Determina como o usuário define o sucesso. Como o usuário é recompensado?

Envolvimento
Descreve como o usuário está envolvido no projeto. Onde for possível, relate o envolvimento nas

funções do processo; por exemplo, uma parte interessada pode ser um revisor de requisitos.

Entregas
Identifica as entregas que o usuário produz e para quem.

Comentários ou Problemas
Determina os problemas que interferem com o sucesso e quaisquer outras informações relevantes. Descreve as tendências que tornam a tarefa do usuário mais fácil ou mais difícil.

08

g) Principais necessidades da parte interessada ou do usuário – devem ser listados nesta seção os principais problemas com soluções existentes observadas pela parte interessada. Para cada problema, procure responder as seguintes questões:

- Quais são as causas desse problema?
- Como o problema é resolvido atualmente?
- Que soluções são esperadas pela parte interessada?

Procure compreender a importância relativa que a parte interessada atribui à solução de cada problema. Técnicas de classificação e votação podem indicar os problemas que devem ser resolvidos versus os problemas que as partes interessadas gostariam que fossem tratados. Use a tabela abaixo para capturar as necessidades da parte interessada.

Necessidade	Prioridade	Interesses	Solução Atual	Solução Proposta

h) Alternativas e concorrência - identifique alternativas que a parte interessada considera disponíveis. Essas alternativas podem incluir a compra do produto de um concorrente, a criação de uma solução própria ou apenas a manutenção do *status quo*. Liste todas as opções competitivas disponíveis que existem ou que podem se tornar disponíveis. Inclua os principais pontos fortes e fracos de cada concorrente conforme observados pela parte interessada.

09

4 - Visão Geral do Produto

Esta seção fornece uma visualização de alto nível dos recursos e capacidades do produto, da interface com outros aplicativos e configurações do sistema. Os aspectos a seguir devem ser contemplados.

a) Perspectiva do produto – o objetivo deste item é colocar o produto em perspectiva com outros produtos relacionados, bem como com o ambiente do usuário. Se o produto for independente, declare isso aqui. Se o produto for um componente de um sistema maior, relacione como esses sistemas interagem e identifique as interfaces relevantes entre os sistemas. Uma maneira fácil de exibir os principais componentes do sistema maior, suas interconexões e interfaces externas é utilizando um diagrama de casos de uso ou diagrama de bloco.

b) Resumo das capacidades - resuma os principais benefícios e recursos que o produto proverá. Por exemplo, um documento de Visão para um sistema de suporte ao cliente pode usar essa parte para tratar da documentação do problema, do roteamento e do relatório de status, sem descrever em detalhes o que essas funções requerem. Organize as funções de modo que a lista seja compreensível para o cliente ou para qualquer outra pessoa que leia o documento pela primeira vez. Uma tabela listando os principais benefícios e seus recursos de suporte pode ser suficiente, como o exemplo a seguir.

Benefício para o cliente	Recursos de Suporte
A nova equipe de suporte pode aprender rapidamente como usar o produto.	A base de conhecimento ajuda a equipe de suporte a identificar rapidamente as correções e soluções alternativas conhecidas.
A satisfação do cliente é melhorada porque não há falhas.	Os problemas são exclusivamentes detalhados em itens, classificados e controlados em todo o processo de resolução. A notificação automática ocorre para quaisquer problemas anteriores.
O gerenciamento pode identificar as áreas com problema e calibrar a carga de trabalho da equipe.	Os relatórios de tendências e distribuição permitem a revisão de alto nível do status do problema.
Equipes de suporte distribuídas podem trabalhar juntas para resolver problemas.	Com um servidor de replicação, as informações do banco de dados podem ser compartilhadas em toda a empresa.
Os clientes podem se ajudar, reduzindo os custos de suporte e melhorando o tempo de resposta.	A base de conhecimento pode ficar disponível na Internet. A base de conhecimento inclui recursos de pesquisa de hipertexto e um mecanismo de consulta gráfica.

Exemplo de Benefícios e Recursos

10

c) Suposições e dependências – neste item devem ser listados os fatores que afetam os recursos declarados no documento de Visão. Liste as premissas que, caso alteradas, poderão modificar o documento de visão. Por exemplo, uma premissa pode declarar que um sistema operacional específico fique disponível para o hardware designado para o produto de *software*. Se o sistema operacional não estiver disponível, será necessário alterar o documento de visão.

d) Custo e precificação – Para produtos vendidos a clientes externos e para muitos aplicativos internos, os problemas de custo e preço podem impactar diretamente a definição e a implementação do aplicativo. Portanto, nesta seção devem ser registradas todas as restrições de custo e preço que sejam

relevantes. Por exemplo, custos de distribuição (número de CDs e controle do CD) ou outras restrições de custo de mercadorias vendidas (manuais, embalagem) podem constituir material importante para o sucesso dos projetos, ou irrelevantes, dependendo da natureza do aplicativo.

e) Licenciamento e instalação - os problemas de licenciamento e instalação também podem impactar diretamente o esforço de desenvolvimento. Por exemplo, a necessidade de suportar a serialização, a segurança da senha ou a licença de rede criarão requisitos adicionais ao sistema que devem ser considerados no processo de desenvolvimento. Os requisitos de instalação também podem afetar a codificação ou criar a demanda de um *software* de instalação à parte.

11

5 - RECURSOS DO PRODUTO E RESTRIÇÕES

a) Recursos do produto

Nesta seção devem ser listados e descritos de forma resumida os recursos do produto. Recursos são as capacidades de alto nível do sistema, necessárias para fornecer benefícios aos usuários. Cada recurso é um serviço solicitado que, em geral, requer uma série de entradas para alcançar o resultado desejado. Por exemplo, um recurso de um sistema de rastreamento de problemas pode ser a capacidade de fornecer relatórios de tendências. À medida que o modelo de casos de uso toma forma, atualize a descrição para fazer referência aos casos de uso.

Como o documento de visão é revisado por uma variedade de equipes envolvidas, mantenha o nível de detalhes geral o suficiente para que todos possam entender. Porém, detalhes suficientes devem estar disponíveis para fornecer à equipe as informações necessárias para criar um modelo de caso de uso ou outros documentos de *design*.

Para gerenciar efetivamente a complexidade do aplicativo, é recomendado, para todo novo sistema ou uma mudança incremental, que se faça uma listagem de recursos a um nível alto o suficiente para que possa resultar em aproximadamente 25 a 99 recursos. Esses recursos fornecem a base para a definição do produto, gerenciamento de escopo e gerenciamento do projeto. Cada recurso será expandido em maiores detalhes no modelo de casos de uso.

Em toda esta seção, torne cada recurso observável para usuários, operadores ou outros sistemas externos. Para cada recurso, inclua uma descrição de funcionalidade e problemas de usabilidade que devem ser tratados. As seguintes diretrizes se aplicam:

- Evite *design*. Mantenha as descrições do recurso em um nível geral. Mantenha o foco nos recursos necessários e no *porquê* (não como) eles devem ser implementados.

- Indique cada recurso como requisito de um tipo de recurso específico para fácil referência e rastreamento.

12

b) Restrições

É importante observar todo tipo de restrição, sobretudo as restrições de design, as restrições externas, como requisitos operacionais ou regulamentares ou outras dependências.

c) Faixas de qualidade

Defina as faixas de qualidade para desempenho, robustez, tolerância a falhas, usabilidade e características similares que não são descritas ou não são perceptíveis no conjunto de recursos.

d) Precedência e prioridade

Defina a prioridade dos diferentes recursos do sistema.

13

6 - OUTROS REQUISITOS DO PRODUTO

Em um alto nível, liste os padrões aplicáveis, os requisitos de hardware ou plataforma, os requisitos de desempenho e os requisitos ambientais.

a) Padrões aplicáveis - liste todos os padrões com os quais o produto deve estar em conformidade. A lista pode incluir padrões dos seguintes tipos:

- Padrões jurídicos e regulamentares (FDA, UCC);
- Padrões de comunicações (TCP/IP, ISDN);
- Padrões de conformidade da plataforma (Windows, UNIX etc.);
- Padrões de qualidade e segurança (UL, ISO, CMM).

b) Requisitos do sistema - defina os requisitos do sistema necessários para suportar o aplicativo. Estes incluem os sistemas operacionais de host suportados e as plataformas de rede, configurações, memória, periféricos e *software* de parceiros.

c) Requisitos de desempenho – nesta seção devem ser detalhados os requisitos de desempenho. Os problemas de desempenho podem incluir itens como fatores de carregamento de usuário, largura de banda ou capacidade de comunicação, rendimento, exatidão, confiabilidade ou tempos de resposta sob uma variedade de condições de carregamento.

d) Requisitos ambientais - detalhe os requisitos ambientais conforme necessário. Para sistemas baseados em *hardware*, os problemas ambientais podem incluir temperatura, choque elétrico, umidade e radiação, entre outros. Para aplicativos de *software*, os fatores ambientais podem incluir condições de uso, ambiente do usuário, disponibilidade de recursos, problemas de manutenção, manipulação de erros e recuperação de erros.

14

7 - REQUISITOS DE DOCUMENTAÇÃO E ATRIBUTOS DO RECURSO

Nesta seção deve ser descrita toda documentação a ser desenvolvida para suportar a implementação bem sucedida do aplicativo.

a) Notas sobre a liberação, arquivo Leia-me - as notas sobre a liberação ou um arquivo Leia-me podem incluir uma seção "O que Há de Novo", uma discussão sobre problemas de compatibilidade com releases anteriores, e alertas de instalação e atualização. O documento pode também conter ou vincular correções na liberação e quaisquer problemas ou soluções alternativas conhecidos.

b) Ajuda *on-line* - muitos aplicativos fornecem um sistema de ajuda *on-line* para ajudar o usuário. A natureza desses sistemas é exclusiva para desenvolvimento de aplicativo, uma vez que eles combinam aspectos de programação (hyperlinks e outros) com aspectos de composição técnica (organização, apresentação). Muitas equipes consideram que o desenvolvimento do sistema de ajuda*on-line* é um projeto dentro de um projeto, que se beneficia do gerenciamento de escopo e planejamento no início do projeto.

c) Guias de instalação - um documento que inclui instruções de instalação, configuração e de atualização é importante para oferta de solução completa.

d) Rótulo e embalagem - uma aparência consistente começa com a embalagem do produto e se aplica aos menus de instalação, telas iniciais, sistemas de ajuda, caixas de diálogo de GUI e assim por diante. Esta seção define as necessidades e tipos de rótulos ou etiquetas a serem incorporados ao código. Os exemplos incluem observações sobre *copyright* e avisos de patentes, logotipos corporativos, ícones padronizados e outros elementos gráficos.

15

Apêndice 1 : Atributos do Recurso

Forneça aos recursos atributos que podem ser usados para avaliar, controlar, rastrear, priorizar e gerenciar os itens de produtos propostos para implementação. Todos os tipos de requisitos e atributos devem estar esboçados no Plano de Gerenciamento de Requisitos, contudo, nesta seção você pode listar e descrever brevemente os atributos para os recursos que foram escolhidos. As subseções a seguir representam um conjunto de atributos de recursos sugeridos.

a) Status - as equipes configuram o status do recurso após negociação e revisão pela equipe de gerenciamento do projeto. O status controla o andamento do projeto. A tabela a seguir fornece um exemplo de valores para o atributo de status.

Status	Descrição
Proposta	Descreve os recursos que estão em discussão, mas não foram revistos e aceitos pelo "canal oficial". O canal oficial pode ser um grupo de trabalho que consiste em representantes da equipe do projeto, gerenciamento do produto e comunidade do usuário ou do cliente.

Aprovado	Capacidades que são consideradas úteis e factíveis e que foram aprovadas para implementação pelo canal oficial.
Incorporado	Os recursos que foram incorporados na linha de base do produto.

Exemplos de Valores do Status

b) Benefício - o grupo de *marketing*, o gerente de produto ou o analista de negócios configura os benefícios do recurso. Os requisitos não são criados igualmente. A classificação de requisitos por seu benefício relativo para o usuário final abre um diálogo com clientes, analistas e membros da equipe de desenvolvimento. O benefício deve ser utilizado no gerenciamento do escopo e na determinação da prioridade de desenvolvimento. A tabela a seguir fornece um exemplo de valores de atributos de prioridade do benefício.

Prioridade	Descrição
Crítico	Recursos essenciais. A falha na implementação de um recurso crítico significa que o sistema não atenderá às necessidades do cliente. Todos os recursos críticos devem ser implementados na liberação ou a programação falhará.
Importante	Recursos importantes para a eficácia e eficiência do sistema para a maioria dos aplicativos. As funções não podem ser facilmente fornecidas de alguma outra maneira. Omitir um recurso importante pode afetar a satisfação do cliente ou do usuário, ou até mesmo a receita. No entanto, a liberação não será atrasada porque um recurso importante não foi incluído.
Útil	Os recursos que são úteis em menos aplicativos típicos, são usados menos frequentemente, ou podem corresponder às soluções alternativas razoalmente eficientes. Nenhuma receita significativa ou impacto na satisfação do cliente poderá ser esperada se tal item não for incluído em uma liberação.

Exemplos de Prioridades do Benefício

c) Esforço – o esforço necessário para implementar os recursos é definido pela equipe de desenvolvimento. Alguns recursos requerem mais tempo e recursos do que outros. A estimativa de tempo por equipe ou pessoa, linhas de código necessárias ou pontos de funções, por exemplo, ajuda a calibrar a complexidade e definir expectativas do que pode ou não pode ser realizado em um determinado período de tempo. Use a estimativa para gerenciar o escopo e determinar a prioridade de desenvolvimento.

d) Risco - a equipe de desenvolvimento estabelece os níveis de risco, com base na probabilidade do projeto experimentar eventos indesejados, como saturações de custos, atrasos na programação ou até cancelamento. A maioria dos gerentes de projetos considera categorizar os riscos como alto, médio e suficiente, embora graduações mais refinadas sejam possíveis. Em geral, o risco pode ser avaliado indiretamente medindo-se a incerteza (faixa ou intervalo) da estimativa de planejamento da equipe de projetos.

e) Estabilidade – é definida pelo analista e pela a equipe de desenvolvimento, com base na probabilidade do recurso ser alterado ou no entendimento da equipe acerca de possível alteração no recurso. A estabilidade é usada para ajudar a estabelecer prioridades de desenvolvimento e determinar os itens que deverão ser extraídos.

f) Liberação de destino - As equipes registram a versão anterior (release) do produto na qual o recurso aparece primeiro. Este campo deve ser usado para alocar recursos de um documento de visão para um release de linha de base específico. Quando combinado com o campo Status, sua equipe pode propor, registrar e discutir vários recursos da liberação sem comprometer o desenvolvimento. Somente os recursos cujo status definido como "incorporado" e cuja liberação de destino é definida serão implementados. Com o gerenciamento de escopo, o número da versão da liberação de destino poderá ser aumentado de forma que o item permaneça no documento de visão, mas seja programado para uma liberação (release) posterior.

g) Designado para - Na maioria dos projetos, os recursos serão designados para “equipes de recursos”, que são responsáveis por uma extração maior, compondo os requisitos e a implementação do *software*. O processo ajuda todos da equipe do projeto a entender melhor as responsabilidades.

h) Motivo - esse campo de texto é usado para controlar (rastrear) a origem do recurso solicitado. Os requisitos existem por motivos específicos. Esse campo registra uma explicação ou uma referência a uma explicação. Por exemplo, a referência pode ser a página e número da linha de uma especificação de requisito do produto, ou um marcador de minuto em um vídeo de entrevista de um cliente importante.

16

Depois de criar o documento de visão, temos, portanto, o registro de metas de negócio e necessidades das partes interessadas, uma instrução do problema, uma instrução de solução e requisitos de recurso de alto nível. Temos, também, o rastreamento de recursos para solicitações das partes interessadas.

Os requisitos de recurso de alto nível e artefatos relacionados no documento de visão fornecem um ponto de início para elaborar e priorizar um conjunto de requisitos aprovados para a liberação.

É necessário fazer a revisão do documento de visão e requisitos de recurso. Depois disso, o próximo passo será trabalhar com as partes interessadas para definir, organizar e priorizar requisitos.

17

RESUMO

O documento de Visão, segundo a metodologia RUP, está inserido no contexto da área de Requisitos e é no início do ciclo de vida de um projeto de *software*. É a referência, o caminho é a chave para se implantar um processo de requisitos efetivo e consequentemente o projeto. O nome do documento já diz tudo, é uma visão de como será o projeto e todos os envolvidos saberão o que está tratando e qual o caminho a seguir, ou seja, objetivo único para um projeto.

Sem o documento de Visão, o projeto pode tomar qualquer direção, sendo para o sucesso ou para o fracasso.