Projeto T2Ti ERP 3.0

Análise e Gerenciamento de Requisitos

A T2Ti

Apresentação

A T2Ti nasce do sonho de três colegas que trabalhavam no maior banco da américa latina.

Tudo começa em 2007 com o lançamento do curso Java Starter. Logo depois veio o Siscom Java Desktop seguido de outros treinamentos.

Desde então a Equipe T2Ti se esforça para produzir material de qualidade que possa formar profissionais para o mercado, ensinando como desenvolver sistemas de pequeno, médio e grande porte.

Um dos maiores sucessos da Equipe T2Ti foi o Projeto T2Ti ERP que reuniu milhares de profissionais num treinamento dinâmico onde o participante aprendia na prática como desenvolver um ERP desde o levantamento de requisitos. Foi através desse treinamento que centenas de desenvolvedores iniciaram seu negócio próprio e/ou entraram no mercado de trabalho.

Em 2010 a T2Ti lança sua primeira aplicação para produção, o Controle Financeiro Pessoal. O sucesso foi tanto que saiu até em matéria no site Exame, ficando entre os 10 aplicativos mais baixados da semana.

Começa então a era de desenvolvimento de sistemas para alguns clientes exclusivos, pois o foco ainda era em desenvolvimento de treinamentos. A T2Ti desenvolve sistemas para o mercado nacional e internacional.

Atualmente a T2Ti se concentra nas duas vertentes: desenvolver sistemas e produzir treinamentos.

_____

Sumário

Análise e Gerenciamento de Requisitos

Engenharia de Requisitos de Software
Introdução; Funcionalidades do Sistema; Categorias dos Requisitos; Relacionando os Requisitos; Prioridades.

Levantamento de Requisitos
Técnicas: Entrevistas, Workshops, Brainstorming.

Participantes do Processo
Introdução; Gerente de Projeto; Analista; Projetista; Arquiteto; Programador; Cliente.

Processo de Requisitos
Introdução; Requisitos Funcionais; Requisitos Não Funcionais; Requisitos de Domínio; Notações para a Especificação de Requisitos; Documento de Requisitos de Software – DRS.

Gerenciamento de Projetos | Requisitos
Introdução; Atividades de Gerenciamento; Planejamento; Programação; Diagrama de Gantt; Gerenciamento de Riscos; Aspectos Gerais.

Gerenciamento com Casos de Uso
Introdução; Usando Casos de Uso no Processo.

Introdução

As necessidades do usuário são, ou pelo menos deveriam ser, o ponto de partida para o início de um projeto. A ambiguidade e sua resolução em requisitos declarados e validados de maneira compreensível são a plataforma da qual você prosseguirá para o projeto. Dar prioridade aos requisitos permite que você desenvolva um plano de projeto significativo, adiando itens de menor prioridade para projetos posteriores.

Finalmente, compreender o escopo de seus requisitos permite que você compreenda que tipo de arquitetura terá o seu projeto.

Determinar e gerenciar requisitos, embora seja considerada uma das mais importantes atividades do processo de desenvolvimento de um projeto é, muitas vezes, tratada com pouco ou até nenhum interesse. Isso muitas vezes ocorre pelo fator tempo e pela sensação de que o processo não traz o resultado esperado.

Vamos imaginar algumas situações que requerem uma definição de requisitos:

- Uma solicitação de um cliente para desenvolver um produto ou realizar um serviço
- Uma solicitação de um amigo para realizar um favor.
- Uma solicitação de um gerente para realizar uma tarefa.

Determinar os requisitos é, em última análise, *entender exatamente o que deve ser feito e o que se espera receber como resultado*. Neste ponto ainda não se questiona como o trabalho será realizado.

Engenharia de Requisitos de Software


Como o cliente explicou


Como o lider de projeto entendeu

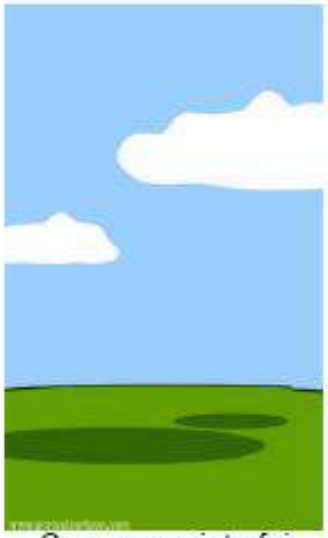

Como o analista planejou


Como o programador codificou

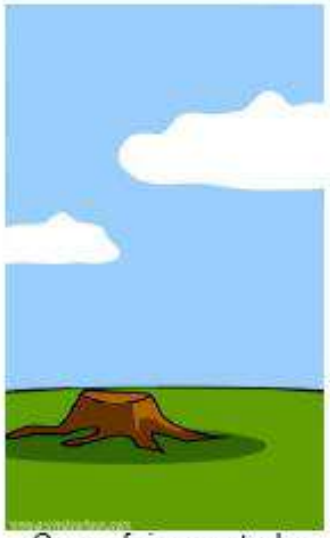

O que os beta testers receberam


Como o consultor de negocios descreveu


Valor que o cliente pagou


Como o projeto foi documentado


O que a assistencia tecnica instalou


Como foi suportado


Quando foi entregue


O que o cliente realmente necessitava

Funcionalidades do Sistema

O levantamento de requisitos faz parte de todo projeto, e as técnicas se aplicam a qualquer sistema.

Uma das coisas que se deve evitar durante o levantamento de requisitos é a ambiguidade.

A ambiguidade é um estado em que você encontra interpretações múltiplas e contraditórias em um texto ou situação.

Você cria ambiguidades quando declara os requisitos para seu sistema ou banco de dados que alguém possa interpretar de maneiras diferentes.

Será que tem algum homem corajoso o suficiente para entrar no salão mostrado na foto ao lado?

Funcionalidades do Sistema

Gause e Weinberg (1990) citam três tipos específicos de ambiguidades que são importantes no momento de levantar os requisitos:

- Requisitos ausentes: necessidades que são uma parte principal da resolução de diferenças de interpretação.
- Palavras ambíguas: palavras na descrição do requisito que são capazes de múltiplas interpretações.
- Elementos introduzidos: palavras que você "coloca na boca" dos interessados.

Imagine que você vai fazer o sistema para uma clínica médica. O dono da clínica, que é o médico, está sempre ausente e pede para você levantar os requisitos com sua secretária.

Com certeza existirão requisitos ausentes aí, pois a secretária não saberá tudo que o médico precisa para o sistema.

Além disso, a secretária não tem o mesmo conhecimento que o médico e utilizará termos que podem ser interpretados de várias maneira por você, que inclusive nem médico é.

Isso leva ao terceiro item mencionado. Com a dificuldade de passar os requisitos, você, provavelmente, completará frases da secretária em alguns momentos, "colocando palavras na boca" dela.

E agora, como resolver isso? O primeiro passo é observar e fazer as perguntas certas. É preciso explorar os requisitos. No exemplo abordado, é preciso também fazer isso com a pessoa certa: o médico.

A secretária pode até passar algumas informações introdutórias, mas você precisaria explorar os requisitos diretamente com o médico.

Funcionalidades do Sistema

E como é que se explora os requisitos de um sistema? O primeiro passo é você compreender o problema proposto e depois questionar se existe solução para o ele.

Voltemos ao nosso exemplo. Qual o problema do médico que solicitou o desenvolvimento do sistema? Digamos que nas conversas que você teve com o médico, sintetizou o problema na seguinte declaração:

"É preciso um sistema onde sejam cadastrados os pacientes e o médico consiga acompanhar todo o histórico de consultas, os medicamentos que eles estão tomando e também a parte financeira, os pagamentos efetuados pelos pacientes."

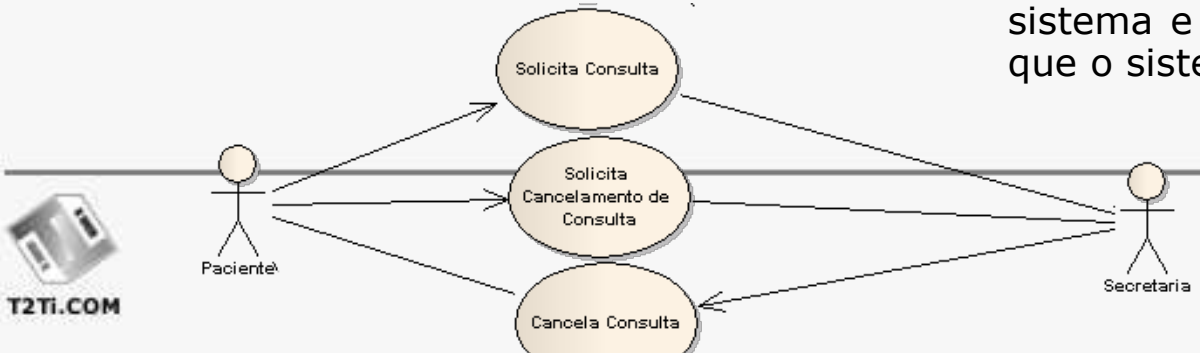
Pronto, aí está o problema dele. Tem solução?

A declaração lhe fornece o sabor do problema, mas não possui detalhes suficientes para chegar a uma solução.

É bom frisar que o consultório já existe e funciona. Ou seja, o médico já possui um sistema! Note que não é um sistema informatizado. Ele anota tudo numa ficha pré-impressa. Cada paciente tem uma ficha com o histórico das consultas, medicamentos e pagamentos. Tudo feito a mão.

Portanto, estamos começando com uma norma: uma solução existente. Com esse tipo de ponto de partida, você quer refinar a definição de seu problema utilizando o sistema existente como guia.

No processo, você identifica as limitações do sistema e talvez os problemas subjacentes de que o sistema tenta tratar.



Funcionalidades do Sistema

O passo seguinte na exploração é fazer perguntas livres de contexto, normalmente começando com os pronomes familiares *quem, o que, por que, quando, onde e como*.

Ao fazer as tais perguntas, você fatalmente chegará a algumas conclusões, que podem ser as seguintes:

- O paciente pode ser qualquer pessoa física que se sente doente e vai até o médico. Existem pacientes antigos e pacientes novos.
- Os pacientes mais antigos tem certas benesses, como atendimento preferencial e descontos.
- Alguns pacientes tem reclamado da demora do médico em encontrar suas informações, que se encontram nas fichas. Alguns chegaram a relatar que o médico passou um remédio errado. Provavelmente se confundiu com as informações da ficha ou pegou uma ficha errada.
- O conteúdo das fichas é crítico, pois contém o histórico de consultas do paciente, o histórico da medicação e também os pagamentos efetuados por eles.
- Alguns pacientes não se consultam há mais de dois anos. As informações desses pacientes não são prioritárias. As informações dos pacientes que tem vindo ao consultório no último ano devem ter prioridade de cadastro.
- O médico quer que o sistema fique pronto em dois meses. Mas ele afirma que prefere ter informações fidedignas do que o sistema funcionando no prazo curto com informações incompletas. Ele espera que você cadastre todas as informações no sistema quando o mesmo ficar pronto.
- A segurança durante o acesso ao sistema não somente é importante para a validação dos dados, ela também garante que o paciente mantenha a confidencialidade e o sigilo.

Funcionalidades do Sistema

E então, consegue achar alguma ambiguidade na lista anterior?

Vou te dar dois exemplos:

"Informações Fidedignas" e "Informações Incompletas".

É bem provável que para você fidedigno e incompleto signifique uma coisa e para o médico signifique outra.

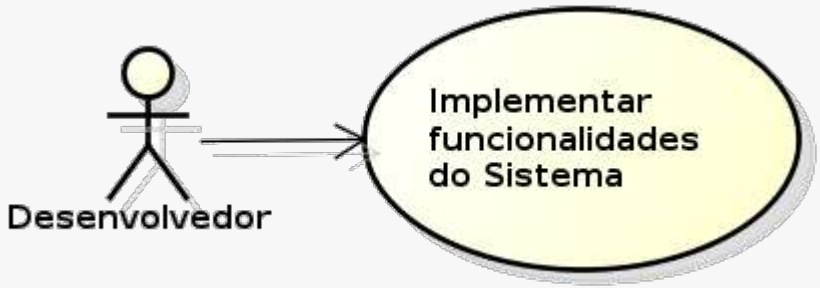
E agora?

Caso surjam termos ambíguos, substitua-os por outros termos ou explique o que os termos querem dizer dentro do contexto.

Por exemplo:

- Informações Fidedignas: são aquelas que representam uma ficha completa tal e qual a ficha escrita tomada como base. Inclusive, o sistema deve permitir que a ficha seja digitalizada e armazenada para ser utilizada como parâmetro.
- Informações Incompletas: seria aquela ficha que não coincide com a ficha escrita.

Agora ficou claro. Você não precisou alterar os termos, mas os explicou de tal maneira que não existem dúvidas sobre o requisito em questão.



Funcionalidades do Sistema

Levantar os requisitos não é o fim de seu trabalho. Nem todo requisito possui a mesma importância dos outros.

A tarefa seguinte é priorizar os requisitos compreendendo-os, relacionando-os uns aos outros e categorizando-os. É um processo iterativo que frequentemente exige uma análise mais profunda por meio de casos de uso. Isso se você estiver trabalhando com um UML.

Existem maneiras ágeis de desenvolver software, onde não cabe a utilização da UML. Veremos cada um dos casos para que o leitor possa escolher o melhor dentro da sua realidade.

Até você chegar a um nível razoável de detalhes, pode haver dificuldade para perceber quais são as verdadeiras prioridades e que coisas dependem das outras.

Quando você chegar ao projeto, geralmente achará que existem dependências tecnológicas que também deve levar em consideração, forçando-o a revisar novamente suas prioridades.

Porém, quando você possuir seus requisitos básicos, pode iniciar a criação de uma estrutura para o desenvolvimento do sistema e do banco de dados.

O fato de você possuir uma lista do que as pessoas esperam que seu sistema faça, não significa que você compreende os requisitos. A lógica do erro afirma que todo sistema resiste aos esforços dos seres humanos em mudá-lo.

Quanto mais você fizer, e quanto mais você aprender com seus erros, melhor você ficará na arte de levantar requisitos

Funcionalidades do Sistema | Categorias dos Requisitos

No nível mais fundamental, você pode categorizar requisitos em:

1) Objetivos Operacionais.
2) Propriedades de Objetos.
3) Regras.
4) Preferências.

1) Objetivos Operacionais

Expressam o propósito existente por trás do sistema e a abordagem que você pretende assumir para alcançar esse propósito. Metas, funções, comportamentos, operações são todos sinônimos de objetivos operacionais.

É o tipo mais importante de requisito, porque ele expressa o significado por trás do sistema, o porquê do que você está fazendo.

2) Propriedades de Objetos

Quando você começar a desenvolver os requisitos de dados para o sistema, começará a ver os objetos (entidades, tabelas, classes, e assim por diante) que existirão no sistema. Quando você pensar neles, começará a ver suas propriedades. Propriedades de objetos são a segunda categoria de requisito, e provavelmente a mais importante para o projetista do banco de dados. As propriedades podem ser colunas de tabelas, atributos de objetos ou membros de dados de classe.

Nem todas as propriedades de objetos estão diretamente relacionadas aos elementos do banco de dados. Em sua maioria, boas propriedades de objetos são muito específicas: o sistema deve estar disponível 24 horas por dia; a resposta do sistema deve levar dois segundos ou menos para qualquer informação determinada, etc. Cuidado com as ambiguidades.

Funcionalidades do Sistema | Categorias dos Requisitos

3) Regras

São requisitos condicionais sobre propriedades de objetos. Para o valor de uma propriedade ser aceitável, ele deve satisfazer a condição. A maioria de restrições de integridade referencial se enquadra nessa categoria de requisito.

Por exemplo, para um paciente ser cadastrado no sistema, será necessário um identificador único, um documento, e tal documento deve ser verificado pelo sistema. A decisão é usar o CPF, que deve ser validado antes de ser persistido. Consegue enxergar os problemas que vão surgir ao implementar tal requisito no sistema?

As regras de regulação de tempo também se enquadram nesta categoria. Por exemplo, você pode possuir um objetivo operacional de que o sistema responda uma solicitação de informações dentro de dois segundos.

Isso passa a ser uma regra: todas as informações do banco de dados devem possuir características de acesso que permitam recuperação em dois segundos ou menos.

Temos uma terceira categoria: a regra de qualidade. Você pode especificar tolerâncias de erro que se tornem parte de seus requisitos. Esses tipos de regras estabelecem tolerâncias ou limites além dos quais os dados não podem passar.

Pense um pouco antes de criar uma regra. As regras têm a tendência a se tornarem leis, particularmente em um projeto de software com prazo de entrega. Esteja razoavelmente seguro de que sua regra é necessária antes de impô-la a seu projeto. Você pode ter que conviver com ela por muito tempo.

Voltando à regra do CPF, imagine um estrangeiro ou uma criança sendo consultada pelo nosso médico.

Funcionalidades do Sistema | Categorias dos Requisitos

4) Preferências

Uma preferência é uma condição sobre um objeto que expressa um estado preferido.

É importante saber a diferença entre regra e preferência. Uma regra diz 'deverá', 'irá' ou 'poderá'. Normalmente, as preferências assumem a forma do 'máximo possível', 'ótimo', 'em equilíbrio' ou algo similar. É uma questão de integridade versus uma questão de desejo (assim como tantos outros aspectos da vida).

É possível utilizar valores iniciais ou valores padrão - valores que o banco de dados atribuirá se você não especificar nada - para representar preferências.

Obrigar a digitação do CPF para todos os pacientes é uma regra, não uma preferência.

E o que seria uma preferência? Por exemplo, o médico prefere que o paciente pague sempre em dinheiro e fornece um desconto de 10%.

Então ele deseja que, ao lançar um recebimento, o sistema já informe os dados em dinheiro e calcule o valor do desconto. Dessa forma, o paciente saberá que está ganhando um desconto e o médico receberá o dinheiro na hora, de acordo com sua preferência. Assim ele deseja. Pode ser que o paciente decida pagar no cartão. Nesse caso, o médico deverá alterar a forma padrão do recebimento, aquele que é a sua preferência.

Percebeu a diferença entre regra e preferência? Pense em outros exemplos.

Funcionalidades do Sistema | Relacionando os Requisitos

O próximo passo na compreensão de requisitos é tratá-los como um sistema, relacionando-os uns aos outros. A combinação sinérgica de requisitos geralmente pode mudar a natureza da sua abordagem do projeto.

Com requisitos de dados, você normalmente expressa relacionamentos diretamente por meio de modelos de dados que contenham os objetos e seus relacionamentos.

Por exemplo, você pode ter uma tabela onde guarda os tipos de remédio, mas em algum momento os remédios armazenados terão que se relacionar com os pacientes, criando assim o histórico de medicamentos do paciente.

Isso também ocorre com os pagamentos. Você pode ter uma tabela que armazene os tipos de pagamento: dinheiro, cartão, etc. Em algum momento você vai relacionar os pagamentos aos pacientes, criando o histórico financeiro.

É importante compreender como os requisitos se aglomeram. As conexões que você pode estabelecer entre os requisitos normalmente lhe dirão como a arquitetura de seu sistema pode organizar seus subsistemas. Se em determinado conjunto os requisitos se relacionam fortemente uns com os outros, mas são relativamente desconexos de outros requisitos, você possui um candidato a um subsistema. Por outro lado, se você sentir intuitivamente que vários requisitos pertencem uns aos outros, mas não encontrar nenhum relacionamento entre eles, pense mais. Você deixou escapar alguma coisa.

Por exemplo, armazenar o paciente é o requisito central do nosso sistema de estudo. Os históricos de medicamentos e financeiro fazem parte desse requisito principal. Saber qual paciente mais frequenta o consultório para ganhar um brinde seria um subsistema, com implementação separada, embora exista relação com o requisito central.

Funcionalidades do Sistema | Prioridades

Os requisitos assumem muitos aspectos. Alguns são vitais para o projeto, outros, são menos importantes.

Você precisa treinar muito para distinguir entre requisitos sem importância e aqueles que realmente farão o sistema funcionar de acordo com o esperado por quem o demandou. Um aspecto disso é a relatividade das prioridades. Elas dependem da natureza das pessoas envolvidas no projeto.

Essa configuração de crenças e atitudes pode mudar radicalmente de projeto para projeto. De muitas formas, você deve ver as prioridades como completamente novas em cada projeto. Cada um dos diferentes tipos de requisito possui um diferente conjunto de prioridades.

Devem-se priorizar os objetivos operacionais por sua contribuição à missão básica do sistema.

Se o alcance da missão depende totalmente de um determinado objetivo, este é crítico. Se você puder pular esse objetivo e ainda assim atingir a missão, ele será marginal. Se o objetivo não possui relação com a missão, ele é um requisito ruim.

Exemplos:

1) Objetivo crítico: cadastrar o paciente.
2) Objetivo marginal: cadastrar os pais do paciente na sua ficha.
3) Requisito ruim: cadastrar as preferências do paciente em relação a filmes.

Devem-se priorizar as propriedades de objetos em categorias similares: obrigatória, desejável e ignorável. Uma propriedade 'obrigatória' é uma propriedade sem a qual o sistema não realizará sua missão (1). Uma propriedade 'desejável' é algo bom de ter (2). Uma propriedade 'ignorável' é algo que talvez seja útil, mas provavelmente não vale o custo (3).

Levantamento de Requisitos

Técnicas | Entrevistas

A entrevista é um instrumento fundamental para o levantamento de requisitos.

A entrevista se baseia em uma técnica com seus procedimentos ou regras empíricas com os quais não só se amplia e se verifica como também, ao mesmo tempo, se aplica o conhecimento científico.

A técnica é o ponto de interação entre a ciência e as necessidades práticas. É assim que a entrevista alcança a aplicação de conhecimentos científicos e, ao mesmo tempo, obtém ou possibilita levar a vida diária do ser humano ao nível do conhecimento e da elaboração científica. E tudo isso em um processo ininterrupto de interação.

A entrevista é um instrumento muito difundido e devemos delimitar o seu alcance em função de suas regras ou indicações práticas de sua execução.

A entrevista pode ser de dois tipos fundamentais: aberta e fechada.

- Aberta: o entrevistador tem uma liberdade para as perguntas ou para suas intervenções, permitindo-se toda a flexibilidade necessária em cada caso particular.
- Fechada: as perguntas já estão previstas, assim como a ordem e a maneira de formulá-las. O entrevistador não pode alterar nenhuma dessas disposições.

Levantamento de Requisitos

Técnicas | Entrevistas

A entrevista fechada é, na realidade, um questionário que passa a ter uma relação estreita com a entrevista, na medida em que uma manipulação de certos princípios e regras facilita e possibilita a aplicação do questionário.

No entanto, a entrevista aberta não se caracteriza essencialmente pela liberdade de inserir perguntas, pois há uma flexibilidade suficiente para permitir, na medida do possível, que o entrevistado configure o campo da entrevista segundo sua estrutura psicológica particular. Dito de outra forma: a entrevista pode se configurar o máximo possível dependendo da personalidade do entrevistado.

Podemos considerar a entrevista de um outro ponto de vista, classificando também de duas maneiras, de acordo com o número de participantes.

Assim sendo, teríamos a entrevista individual e a entrevista grupal. A individual seria apenas entre o entrevistador e o entrevistado. A grupal envolveria um ou mais entrevistadores e/ou entrevistados.

No entanto, em todos os casos, a entrevista é sempre um fenômeno grupal, já que mesmo com a participação de um só entrevistado sua relação com o entrevistador deve ser considerada.

Levantamento de Requisitos

Técnicas | Entrevistas

Um ponto fundamental numa entrevista é que o entrevistador desperte interesse e participação, que "motive" o entrevistado.

O certo é que não há possibilidade de uma entrevista correta e frutífera se não se incluir a investigação. Uma utilização correta da entrevista integra na mesma pessoa e no mesmo ato o profissional e o pesquisador.

A chave fundamental da entrevista está na investigação que se realiza durante o seu transcurso. As observações são sempre registradas em função das hipóteses que o observador vai emitindo.

A forma de observar bem é ir formulando hipóteses enquanto se observa, e durante a entrevista verificar e retificar as hipóteses no momento mesmo em que ocorrem em função das observações subsequentes, que por sua vez se enriquecem com as hipóteses prévias.

Quanto ao tipo de comunicação que se estabelece na relação do entrevistador com o entrevistado, deve-se sempre levar em conta a personalidade do entrevistado.

O entrevistador deve observar como o entrevistado se comporta.

Interessam particularmente os momentos de mudança na comunicação e as situações e temas ante os quais ocorrem, assim como as inibições, interceptações e bloqueios.

O tipo de comunicação não é importante apenas por oferecer dados de observação direta que, inclusive, podem ser registrados, mas porque é o fenômeno-chave de toda a relação interpessoal, que, por sua vez, pode ser manipulado pelo entrevistador e, assim, graduar ou orientar a entrevista.

De uma boa entrevista sairão excelentes requisitos.

Levantamento de Requisitos

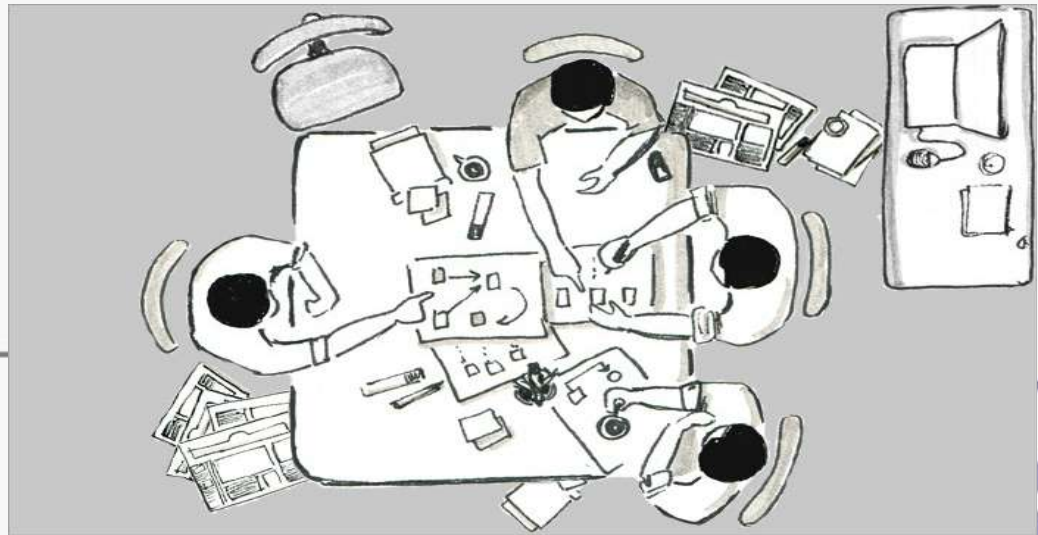
Técnicas | Workshops

Conduzir um workshop de requisitos implica em reunir todos os envolvidos durante um período intensivo, concentrado. Um Analista de Sistemas atua como um facilitador da reunião. Todos os participantes deverão contribuir ativamente e os resultados da sessão deverão ser disponibilizados imediatamente para eles.

O workshop de requisitos fornece um framework para aplicar as outras técnicas de identificação como, por exemplo, brainstorming, encenação, interpretação de papéis e revisão dos requisitos existentes. Essas técnicas poderão ser usadas isoladamente ou combinadas. Todas poderão ser combinadas ao método de caso de uso. Por exemplo, você poderá criar uma ou algumas encenações para cada caso de uso previsto para o sistema. Você pode usar a interpretação de papéis como uma maneira de compreender como os atores usarão o sistema e para ajudá-lo a definir os casos de uso.

O facilitador de um workshop de requisitos precisará estar preparado para as seguintes dificuldades:

- É possível que os envolvidos saibam o que desejam, mas não consigam expressar isso.
- É possível que os envolvidos não saibam o que desejam.
- Os envolvidos poderão achar que sabem o que desejam até que você lhes forneça a alternativa correta.
- Os analistas poderão achar que compreendem os problemas dos usuários melhor do que eles próprios.



Levantamento de Requisitos

Técnicas | Workshops

Algumas pessoas não gostam de participar desse tipo de dinâmica. Se você perguntar para as pessoas se elas preferem uma entrevista ou um workshop, muitas delas dirão que preferem a entrevista. O facilitador precisará divulgar o workshop para os envolvidos que deverão estar presentes e estabelecer o grupo que participará dele.

Os futuros participantes deverão receber material de estudo de "aquecimento" antes do workshop. São funções do facilitador:

- Dar a todos a oportunidade de falar.
- Manter a sessão sob controle.
- Reunir informações para Atributos de Requisitos aplicáveis.
- Registrar as descobertas.
- Resumir a sessão e elaborar conclusões

Após o workshop de requisitos, o facilitador (juntamente com os colegas analistas de sistemas) precisará dedicar um tempo à síntese das descobertas e ao resumo das informações em um formato apresentável.

Levantamento de Requisitos

Técnicas | Brainstorming

Segundo a wikipédia, o brainstorming (literalmente: "tempestade cerebral" em inglês) ou tempestade de ideias é uma atividade desenvolvida para explorar a potencialidade criativa de um indivíduo ou de um grupo - criatividade em equipe - colocando-a a serviço de objetivos pré-determinados.

A técnica de brainstorming tem várias aplicações, mas é frequentemente usada em:

- Desenvolvimento de novos produtos - obter ideias para novos produtos e efetuar melhoramentos aos produtos existentes.
- Publicidade - desenvolver ideias para campanhas de publicidade.
- Resolução de problemas - consequências, soluções alternativas, análise de impacto, avaliação.
- Gestão de processos - encontrar formas de melhorar os processos comerciais e de produção.
- Gestão de projetos - identificar objetivos dos clientes, riscos, entregas, pacotes de trabalho, recursos, tarefas e responsabilidades.
- Formação de equipes - geração de partilha e discussão de ideias enquanto se estimulam os participantes a raciocinar e a criar: criatividade em equipe.

Levantamento de Requisitos

Técnicas | Brainstorming

Podemos dividir o brainstorming em três partes:

1) Encontrar os fatos.
   - Definição do problema.
   - Preparação.
2) Geração da ideia.
3) Encontrar a solução.

Inicialmente, define-se o problema. Poderá ser necessário subdividir o problema em várias partes. A técnica de brainstorming funciona para problemas que têm muitas soluções possíveis tal como o levantamento de requisitos para um sistema. Depois é necessário colher toda a informação que pode relacionar-se com o problema. Após isso, parte-se para a geração de ideias por brainstorming. Finalmente, para encontrar a solução, deve-se avaliar e selecionar as melhores ideias.

Existem dois princípios para o brainstorming:

1) Atraso do julgamento. A maioria das más ideias são inicialmente boas ideias. Atrasando ou adiando o julgamento, é dada a hipótese de se gerarem muitas ideias antes de se decidir por uma. Quando geramos ideias, é necessário ignorar as considerações à importância da ideia, à sua usabilidade, à sua praticabilidade. Neste patamar, todas as ideias são iguais.

2) Criatividade em quantidade e qualidade. O segundo princípio é relativo à quantidade e qualidade da criatividade. Quanto mais ideias forem geradas, será mais provável encontrar uma boa ideia. Uma ideia pode levar a uma outra. Ideias más podem levar a boas ideias. O brainstorming dá-nos a hipótese de pôr as ideias que passam pela cabeça no papel, de maneira a conseguir obter as melhores delas.

Levantamento de Requisitos

Técnicas | Brainstorming

Existem quatro regras principais para o brainstorming:

1) Críticas são rejeitadas: Esta é provavelmente a regra mais importante. A não ser que a avaliação seja evitada, o princípio do julgamento não pode operar.
2) Criatividade é bem-vinda: Esta regra é utilizada para encorajar os participantes a sugerir qualquer ideia que lhe venha à mente, sem preconceitos e sem medo. As ideias mais desejáveis são aquelas que inicialmente parecem ser sem domínio e muito longe do que poderá ser uma solução.
3) Quantidade é necessária: Quanto mais ideias forem geradas, mais hipóteses há de encontrar uma boa ideia.
4) Combinação e aperfeiçoamento são necessários: O objetivo desta regra é encorajar a geração de ideias adicionais para a construção e reconstrução sobre as ideias dos outros.

Frequentemente a tempestade cerebral permite um desbloqueio, um aquecimento para a conversação entre o analista e o cliente.

O funcionamento durante o levantamento de requisitos pode ser assim: dado um tema, o cliente expressa oralmente, em uma palavra ou em frases bem curtas, tudo o que lhe vem à cabeça, sugerido por aquele tema, sem se preocupar em censurar essas ideias. O analista vai anotando tudo e a seguir faz a seleção das ideias, segundo algum critério prévio, seja agrupando-as por alguma semelhança, seja eliminando as que não podem ser postas em prática, etc.

Participantes do Processo

Introdução

O desenvolvimento de software é uma tarefa altamente cooperativa. Tecnologias complexas demandam especialistas em áreas específicas. Uma equipe de desenvolvimento de sistemas de software pode envolver vários especialistas, como por exemplo, profissionais de informática para fornecer o conhecimento técnico necessário ao desenvolvimento do sistema de software e especialistas do domínio para o qual o sistema de software deve ser desenvolvido.

Não é incomum que um mesmo profissional "incorpore" diversos papéis. Isso ocorre muitas vezes no caso de um profissional autônomo. Ele passa a ser o analista, o programador e até mesmo o vendedor.

Mas também é comum em empresas grandes o acúmulo de papéis por parte de alguns profissionais.

Uma equipe de desenvolvimento de software típica consiste de um gerente, analistas, projetistas, programadores, clientes e grupos de avaliação de qualidade. Vamos analisar alguns desses papéis.

No entanto, é importante notar que a descrição dos participantes do processo tem mais um fim didático. Na prática, a mesma pessoa desempenha diferentes funções e, por outro lado, uma mesma função é normalmente desempenhada por várias pessoas.

Participantes do Processo

Gerente de Projeto

É o profissional responsável pela gerência ou coordenação das atividades necessárias à construção do sistema. Esse profissional também é responsável por fazer o orçamento do projeto de desenvolvimento, como por exemplo, estimar o tempo necessário para o desenvolvimento do sistema, o cronograma de execução das atividades, a mão de obra especializada, os recursos de hardware e software etc.

Ele deve acompanhar as atividades realizadas durante o desenvolvimento do sistema. É sua função verificar se os diversos recursos alocados estão sendo gastos na taxa esperada e, caso contrário, tomar providências para adequação dos gastos. Questões tais como identificar se o sistema é factível, escalonar a equipe de desenvolvimento e definir qual o processo de desenvolvimento a ser utilizado também devem ser cuidadosamente estudadas pelo gerente do projeto.

Participantes do Processo

Analista de Sistemas

É o profissional que deve ter conhecimento do domínio do negócio. Esse profissional deve entender os problemas do domínio do negócio para que possa definir os requisitos do sistema a ser desenvolvido.

Analistas devem estar aptos a se comunicar com especialistas do domínio para obter conhecimento acerca dos problemas e das necessidades envolvidas da organização empresarial.

O analista não precisa ser um especialista do domínio. Contudo, ele deve ter suficiente domínio do vocabulário da área de conhecimento na qual o sistema será implantado para que, ao se comunicar com o especialista de domínio, este não precise ser interrompido a todo o momento para explicar conceitos básicos da área.

Tipicamente, o analista de sistemas é o profissional responsável por entender as necessidades dos clientes em relação ao sistema a ser desenvolvido e repassar esse entendimento aos demais desenvolvedores do sistema. Neste sentido, o analista de sistemas representa uma ponte de comunicação entre dois times: a dos profissionais de computação e a dos profissionais do negócio.

Para realizar suas funções, o analista de sistemas deve entender não só do domínio do negócio da organização, mas também ter conhecimento dos aspectos de alto nível de computação. Neste sentido, o analista de sistemas funciona como um tradutor, que mapeia informações entre duas "linguagens" diferentes: a dos especialistas de domínio e a dos profissionais técnicos da equipe de desenvolvimento.

Participantes do Processo

Analista de Sistemas

Com a experiência adquirida através da participação no desenvolvimento de diversos projetos, alguns analistas se tomam gerentes de projetos.

Na verdade, as possibilidades de evolução na carreira de um analista são excelentes. Isso se deve ao fato de que, durante a fase de levantamento de requisitos de um sistema, o analista se torna quase um especialista no domínio do negócio da organização.

Para essa organização, é bastante interessante ter em seu quadro um profissional que entenda ao mesmo tempo de técnicas de desenvolvimento de sistemas e do processo de negócio da empresa.

Dessa forma, não é rara a situação em que uma organização oferece um contrato de trabalho ao analista de sistemas ao final do desenvolvimento.

Uma característica importante que um analista de sistemas deve ter é a capacidade de comunicação, tanto escrita quanto falada, pois ele é um agente facilitador na comunicação entre os clientes e a equipe técnica. Muitas vezes, as capacidades de se comunicar agilmente e de ter um bom relacionamento interpessoal são mais importantes para o analista do que o conhecimento tecnológico.

Uma outra característica necessária a um analista é a ética profissional. Muitas vezes, o analista de sistemas está em contato com informações sigilosas e estratégicas dentro da organização na qual está trabalhando. Os analistas de sistemas têm acesso a informações como preços de custo de produtos, margens de lucro aplicadas, algoritmos proprietários etc. Certamente, pode ser desastroso para a organização se informações de caráter confidencial como essas caírem em mãos erradas.

Participantes do Processo

Projetista

O projetista de sistemas é o componente da equipe de desenvolvimento cujas funções são:

1) Avaliar as alternativas de solução (da definição) do problema resultante da análise.
2) Gerar a especificação de uma solução computacional detalhada.

A tarefa do projetista de sistemas é muitas vezes chamada de projeto físico. Na prática, existem diversos tipos de projetistas.

Pode-se falar em projetistas de interface (especializados nos padrões de uma interface gráfica, como o Windows ou o Mac OS), projetista de redes (especializados no projeto de redes de comunicação), projetista de bancos de dados (especializados no projeto de bancos de dados) e assim por diante. O ponto comum a todos esses tipos é que eles trabalham nos modelos resultantes da análise para adicionar os aspectos tecnológicos a tais modelos.

Participantes do Processo

Arquiteto de Software

Um profissional encontrado principalmente em grandes equipes reunidas para desenvolver sistemas complexos é o arquiteto de software. O objetivo desse profissional é elaborar a arquitetura do sistema como um todo.

É ele quem toma decisões sobre quais são os subsistemas que compõem o sistema como um todo e quais são as interfaces entre esses subsistemas.

Além de tomar decisões globais, o arquiteto também deve ser capaz de tomar decisões técnicas detalhadas (por exemplo, decisões que têm influência sobre o desempenho do sistema).

Esse profissional também trabalha em conjunto com o gerente de projeto para priorizar e organizar o plano de projeto.

Participantes do Processo

Programador

É o responsável pela implementação do sistema. Normalmente há vários programadores em uma equipe de desenvolvimento. Um programador pode ser proficiente em uma ou mais linguagens de programação, além de ter conhecimento sobre bancos de dados e poder ler os modelos resultantes do trabalho do projetista.

A maioria das equipes de desenvolvimento possui analistas que realizam alguma programação, e programadores que realizam alguma análise. No entanto, o analista de sistemas está envolvido em todas as etapas do desenvolvimento, diferentemente do programador, que participa unicamente das fases finais (implementação e testes). Outra diferença é que analistas de sistemas devem entender tanto de tecnologia de informação quanto do processo de negócio, já os programadores tendem a se preocupar somente com os aspectos tecnológicos do desenvolvimento.

Não é raro que um programador seja promovido a analista de sistema. Essa é uma prática comum nas empresas de desenvolvimento de software e é baseada na falsa lógica de que bons programadores serão bons analistas de sistemas. A verdade é que não se pode ter certeza de que um bom programador será um bom analista. Além disso, um programador não tão talentoso pode se tornar um ótimo analista. De fato, uma crescente percentagem de analistas sendo formados tem uma formação anterior diferente de computação. Por outro lado, um analista de sistemas deve ter algum conhecimento de programação para que produza especificações técnicas de um processo de negócio para serem passadas a um programador.

Participantes do Processo

Cliente

Outro componente da equipe de desenvolvimento é o cliente. O cliente é o indivíduo, ou grupo de indivíduos, para o qual o sistema é construído. Podem-se distinguir dois tipos de clientes

- Cliente Usuário: é o indivíduo que efetivamente utilizará o sistema. Normalmente é um especialista no domínio do negócio. É com ele que o analista de sistemas interage para levantar os requisitos do sistema. É o usuário final.
- Cliente Contratante: é o indivíduo que solicita o desenvolvimento do sistema. É ele quem encomenda e patrocina os custos de desenvolvimento e manutenção.

Em pequenas organizações, eles são a mesma pessoa. Em grandes organizações, o cliente proprietário faz parte da gerência e é responsável pela tomadas de decisões, enquanto o cliente usuário é o responsável pela realização de processos operacionais.

Há casos em que o produto de software não é encomendado por um cliente. Em vez disso, o produto é desenvolvido para posteriormente ser comercializado. Esses produtos de software são normalmente direcionados para o mercado de massa. Exemplos: processadores de texto, editores gráficos, jogos eletrônicos etc. Nesses casos, a equipe de desenvolvimento trabalha com o pessoal de marketing como se estes fossem os clientes reais.

Além disso, algumas empresas possuem seu próprio setor de tecnologia para desenvolver os sistemas internos. Nesses casos, o cliente passa a ser um funcionário de outro departamento que solicita a aplicação, normalmente chamado de gestor.

O envolvimento do usuário final no desenvolvimento de um sistema de software é de fundamental importância.

Processo de Requisitos

Introdução

Os problemas que os engenheiros de software têm para solucionar são, muitas vezes, imensamente complexos. Compreender a natureza dos problemas pode ser muito difícil, especialmente se o sistema for novo.

Consequentemente, é difícil estabelecer com exatidão o que o sistema deve fazer. As descrições das funções e das restrições são os requisitos para o sistema e o processo de descobrir, analisar, documentar e verificar essas funções e restrições é chamado de engenharia de requisitos.

O termo requisito não é utilizado de modo consistente. Em alguns casos, um requisito é visto como uma declaração abstrata de uma função que o sistema deve fornecer ou de uma restrição do sistema. Outras vezes, ele é uma definição detalhada, matematicamente formal, de uma função do sistema.

Imagine uma grande empresa que precise desenvolver um ERP através de uma licitação. Ela precisa definir suas necessidades de maneira suficientemente abstrata para que uma solução não seja predefinida.

Os requisitos devem ser redigidos de modo que os prospectivos fornecedores possam apresentar propostas, oferecendo, talvez, diferentes maneiras de atender às necessidades organizacionais do cliente.

Uma vez estabelecido um contrato, o fornecedor prepara uma definição de sistema para o cliente, com mais detalhes, de modo que o cliente compreenda e possa validar o que o software fará.

Esses dois documentos (o previamente feito pelo cliente e o que foi feito pelo fornecedor) podem ser chamados de documentos de requisitos do sistema.

Processo de Requisitos

Introdução

Observe que já citamos dois documentos. No exemplo mencionado, citamos uma grande empresa que deseja fazer uma licitação. Mas isso poderia ocorrer com qualquer outra empresa que, previamente, declarasse de forma abstrata seus requisitos. Esse é o primeiro documento.

Daí o fornecedor contratado analisa os tais requisitos e redige os seus próprios requisitos de forma mais detalhada. Esse é o segundo documento.

Alguns acadêmicos chamam o primeiro documento de Requisitos do Usuário e o segundo documento de Requisitos de Sistema. Além desses dois documentos, um terceiro pode aparecer: Especificação de Projeto de Software. Trata-se de uma descrição mais detalhada para associar a engenharia de requisitos e as atividades de projeto.

1) Requisitos do Usuário são declarações, em linguagem natural e também em diagramas, sobre as funções que o sistema deve fornecer e as restrições sob as quais deve operar.
2) Requisitos de Sistema estabelecem detalhadamente as funções e as restrições de sistema. O documento de requisitos de sistema, algumas vezes chamado de especificação funcional, deve ser preciso. Ele pode servir como um contrato entre o comprador do sistema e o desenvolvedor do software.
3) Especificação de Projeto de Software é uma descrição do projeto de software, que é uma base para o projeto e a implementação mais detalhados.

Processo de Requisitos

Requisitos Funcionais

Os requisitos de sistema de software são, frequentemente, classificados como funcionais ou não-funcionais ou como requisitos de domínio.

Os requisitos funcionais são declarações de funções que o sistema deve fornecer, como o sistema deve reagir a entradas específicas e como deve se comportar em determinadas situações. Em alguns casos, os requisitos funcionais podem também explicitamente declarar o que o sistema não deve fazer.

Tais requisitos dependem do tipo de software que está sendo desenvolvido e dos usuários de software que se espera verificar. Quando expressos como requisitos de usuário, eles são normalmente descritos de um modo bastante geral. Quando expressos como requisitos de sistema, eles descrevem a função de sistema detalhadamente, suas entradas e saídas, exceções, etc.

Os requisitos funcionais podem ser expressos de diversas maneiras. Seguem alguns requisitos funcionais de um sistema de biblioteca de uma universidade (KOTONYA, 1998) para que os estudantes e a faculdade possam pedir livros e documentos de outras bibliotecas:

- O usuário deverá ser capaz de buscar todo o conjunto inicial de banco de dados ou selecionar um subconjunto a partir dele.
- O sistema fornecerá telas apropriadas para o usuário ler documentos no repositório de documentos.
- Cada pedido será alocado a um único identificador (ID_ORDER), que o usuário poderá copiar para a área de armazenagem permanente da conta.

Processo de Requisitos

Requisitos Funcionais

Vários problemas de engenharia de software se originam da imprecisão na especificação de requisitos. Normalmente um desenvolvedor de sistemas interpreta um requisito ambíguo para simplificar sua implementação. Muitas vezes, contudo, isso não é o que o cliente quer. Novos requisitos devem ser estabelecidos e é necessário realizar mudanças no sistema (um novo ciclo). Naturalmente, isso atrasa a entrega do sistema e aumenta os custos.

Observe o segundo requisito visto para o sistema da biblioteca. Ele se refere a 'telas apropriadas' fornecidas pelo sistema. O sistema de biblioteca pode fornecer documentos em uma série de formatos, e a intenção desse requisito é que as telas para todos esses formatos estejam disponíveis.

Mas ele está redigido de forma ambígua, pois não deixa claro que as telas para cada formato de documento devem ser disponibilizadas.

Um desenvolvedor sob pressão pode simplesmente fornecer uma tela de texto e argumentar que os requisitos foram cumpridos.

Em tese, a especificação de requisitos funcionais de um sistema deve ser completa e consistente. Quando digo 'completa', significa que todas as funções requeridas pelo usuário devem estar definidas. A consistência significa que os requisitos não devem ter definições contraditórias. Na prática, para sistemas complexos e grandes, é quase impossível atingir a consistência e a completeza dos requisitos. Isso ocorre por conta da complexidade inerente ao sistema e porque diferentes pontos de vista apresentam necessidades inconsistentes.

À medida que os problemas são descobertos durante as revisões ou em fases posteriores do ciclo de vida, os problemas no documento de requisitos devem ser corrigidos.

Requisitos Não Funcionais

São restrições sobre os serviços ou as funções oferecidas pelo sistema. Exemplos: restrições de tempo, restrições sobre o processo de desenvolvimento, padrões, entre outros. São aqueles que não dizem respeito diretamente às funções específicas fornecidas pelo sistema. Eles podem estar relacionados a propriedades de sistema emergentes, como confiabilidade, tempo de resposta e espaço em disco.

Muitos requisitos não funcionais dizem respeito ao sistema como um todo, e não a características individuais do sistema. Isso significa que eles são, frequentemente, mais importantes do que os requisitos funcionais individuais. Enquanto a falha em cumprir com um requisito funcional individual pode degradar o sistema, a falha em cumprir um requisito não funcional de sistema pode tomar todo o sistema inútil. Por exemplo, se um sistema de aviação não atender a seus requisitos de confiabilidade, ele não será atestado como seguro para operação.

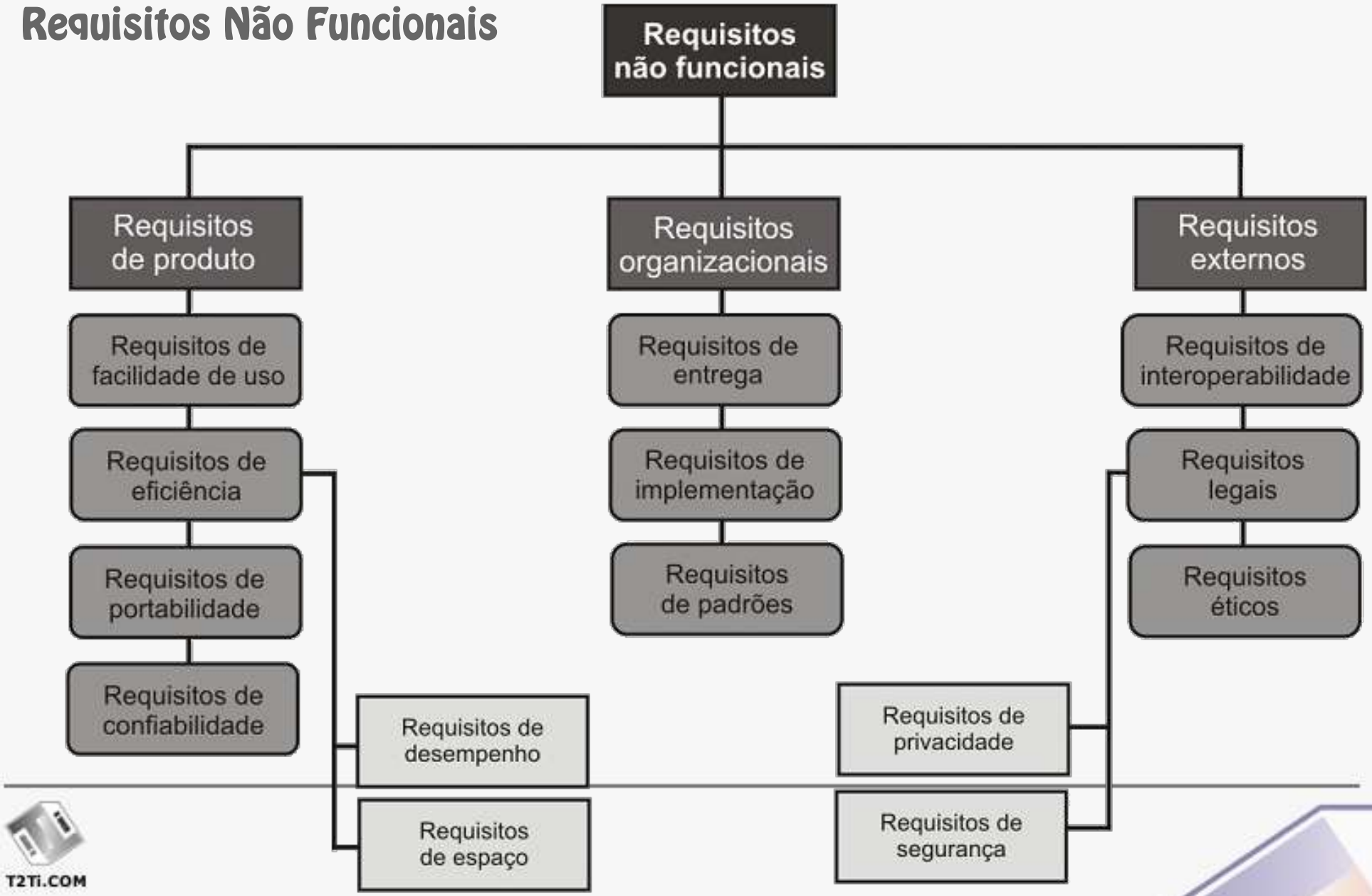
Os requisitos não funcionais nem sempre dizem respeito ao sistema de software a ser desenvolvido. Alguns requisitos não funcionais podem restringir o processo que pode ser utilizado para desenvolver o sistema.

São exemplos de requisitos de processo uma especificação dos padrões de qualidade, que deve ser utilizada no processo, uma especificação de que o projeto deve ser produzido com um conjunto especificado de ferramentas CASE e uma descrição de processo a ser seguido.

Os requisitos não funcionais surgem conforme a necessidade dos usuários, em razão de restrições de orçamento, de políticas organizacionais, pela necessidade de interoperabilidade com outros sistemas de software ou hardware ou devido a fatores externos, como demandas legais (legislação).

Processo de Requisitos

Requisitos Não Funcionais



Requisitos Não Funcionais

Vamos observar agora a descrição dos requisitos não funcionais vistos no diagrama da página anterior:

1) Requisitos de Produtos: são os requisitos que especificam o comportamento do produto. Entre os exemplos estão os requisitos de desempenho sobre com que rapidez o sistema deve operar e quanta memória ele requer, os requisitos de confiabilidade, que estabelecem a taxa aceitável de falhas, os requisitos de portabilidade e os requisitos de facilidade de uso.

2) Requisitos Organizacionais: são procedentes de políticas e procedimentos nas organizações do cliente e do desenvolvedor. Exemplos: os padrões de processo que devem ser utilizados, os requisitos de implementação, como a linguagem de programação ou o método de projeto utilizado, e os requisitos de fornecimento (quando o produto e seus documentos devem ser entregues).

3) Requisitos Externos: abrange todos os requisitos procedentes de fatores externos ao sistema e a seu processo de desenvolvimento. Dentre eles destacam-se os requisitos de interoperabilidade, que definem como o sistema interage com sistemas em outras organizações, os requisitos legais, que devem ser seguidos para assegurar que o sistema opera de acordo com a lei, e os requisitos éticos, que são definidos para garantir que o sistema será aceitável para seus usuários e o público em geral.

É importante que os requisitos não funcionais sejam expressos quantitativamente, utilizando-se métricas que possam ser objetivamente testadas.

As medições podem ser feitas durante o teste de sistema, para determinar se o sistema cumpre com esses requisitos ou não. A próxima página contém uma tabela com possíveis métricas.

Processo de Requisitos

Requisitos Não Funcionais

Propriedade	Métrica
Velocidade	Transações processadas/segundo Tempo de resposta ao usuário/evento Tempo de refresh da tela
Tamanho	K bytes Número de chips de RAM
Facilidade de Uso	Tempo de Treinamento Número de frames de ajuda
Confiabilidade	Tempo médio para falhar Probabilidade de indisponibilidade Taxa de ocorrência de falhas Disponibilidade
Robustez	Tempo de reinício depois de uma falha Porcentagem de eventos que causam falhas Probabilidade de que dados sejam corrompidos por falhas
Portabilidade	Porcentagem de declarações dependentes de sistemas-alvo Número de sistemas-alvo

Processo de Requisitos

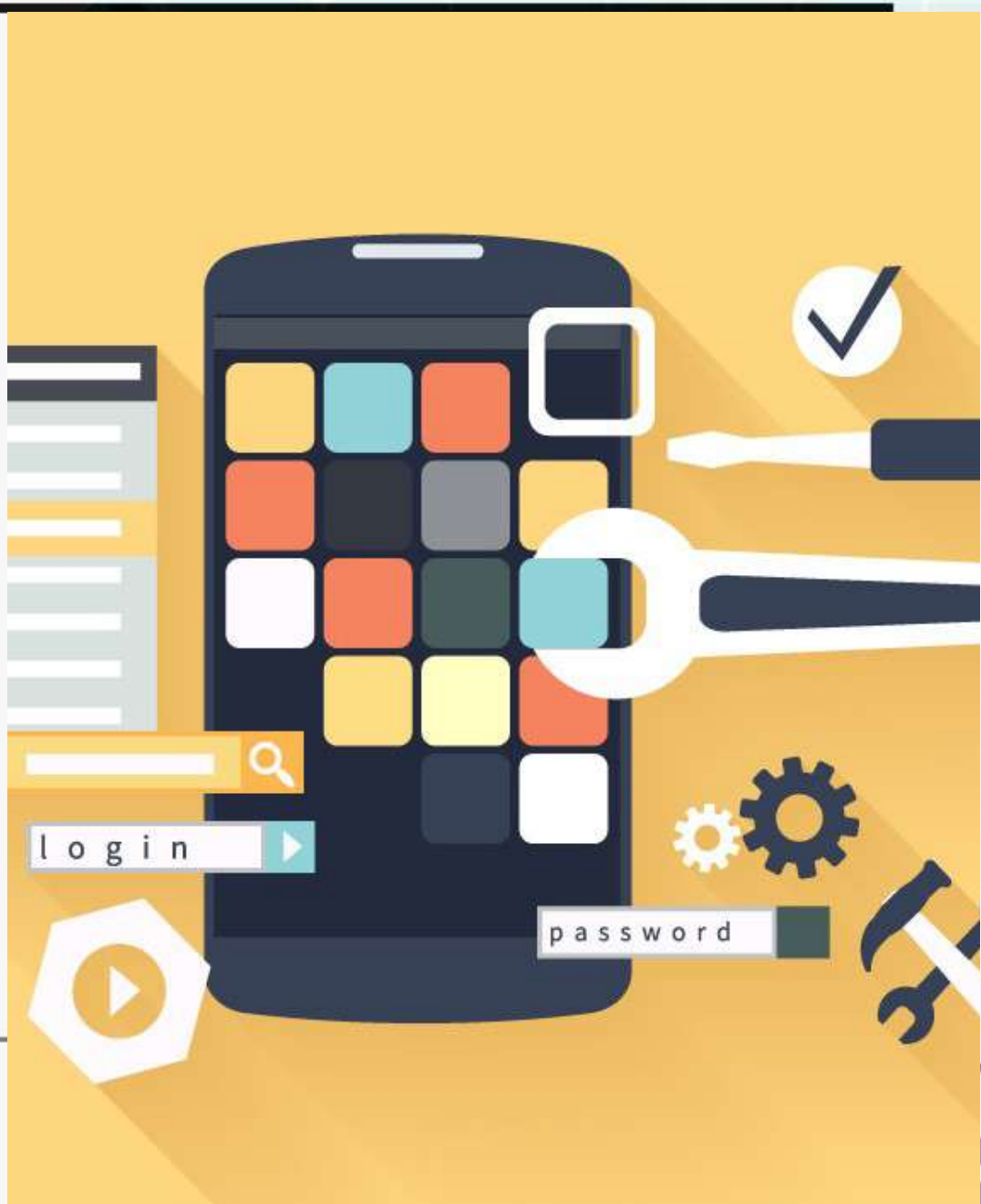
Requisitos Não Funcionais

O ideal é que os requisitos funcionais e não funcionais sejam diferenciados em um documento de requisitos. Na prática, isso é difícil.

Se os requisitos não funcionais forem definidos separadamente dos requisitos funcionais, algumas vezes será difícil ver o relacionamento entre eles. Se eles forem definidos com os requisitos funcionais, poderá ser difícil separar considerações funcionais e não funcionais e identificar os requisitos que correspondem ao sistema como um todo.

É preciso encontrar um equilíbrio adequado e isso depende do tipo de sistema que está sendo especificado.

Tenha em mente que os requisitos claramente relacionados às propriedades emergentes do sistema devem ser explicitamente destacados.



Processo de Requisitos

Requisitos de Domínio

Os requisitos de domínio são derivados do domínio da aplicação do sistema, em vez de serem obtidos a partir das necessidades específicas dos usuários do sistema. Eles podem ser novos requisitos funcionais em si, podem restringir os requisitos funcionais existentes ou estabelecer como devem ser realizados cálculos específicos.

Os requisitos de domínio são importantes porque, muitas vezes, refletem fundamentos do domínio da aplicação. Se esses requisitos não forem satisfeitos, poderá ser impossível fazer o sistema operar satisfatoriamente.

Para ilustrar, vamos pensar novamente naquele sistema para a clínica médica. Observe os seguintes requisitos.

1) Deve haver uma interface-padrão com o usuário para o bancos de dados: XYZ.
2. Em razão das restrições referentes a venda de alguns medicamentos, alguns registros devem ser impedidos de serem armazenados.

O primeiro desses requisitos é uma restrição sobre uma exigência funcional de sistema. Ele especifica que a interface com o usuário para o banco de dados deve ser implementada de acordo com um padrão específico. O segundo requisito foi introduzido devido às regras da agência reguladora de medicamentos.

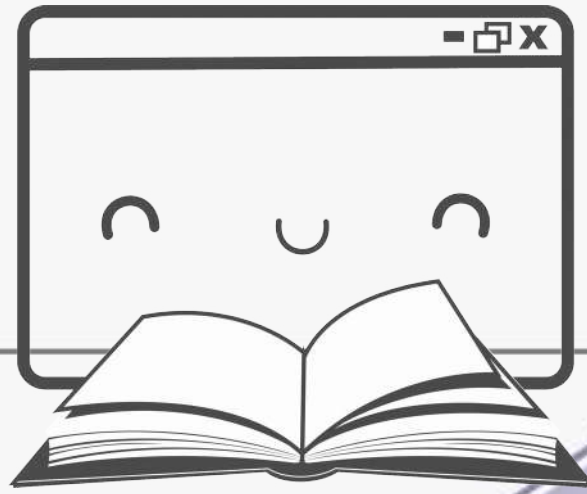
E no caso de um requisito de domínio que especifica um cálculo? Podemos pensar no cálculo de juros para um boleto atrasado.

JURO = VALOR * (TAXA/30) * DIA_ATUAL.

Notações para a Especificação de Requisitos

As especificações de requisitos escritas em linguagem natural estão sujeitas a divergências. Muitas vezes, elas somente são descobertas em fases posteriores do processo de software e, então, a solução para elas pode ser muito dispendiosa. Para solucionar esse problema, foram criadas especificações. DAVIS (1990) resume e compara algumas dessas diferentes abordagens da especificação de requisitos.

- Linguagem natural estruturada: essa abordagem depende da definição de formulários-padrão ou *templates* para expressar a especificação de requisitos. Exemplo: o documento descritivo de casos de uso.
- Linguagem de Descrição de Projeto: essa abordagem utiliza uma linguagem como uma linguagem de programação, mas com recursos mais abstratos para especificar os requisitos pela definição de um modelo operacional do sistema.
- Notações Gráficas: uma linguagem gráfica, complementada com anotações de texto, é utilizada para definir os requisitos funcionais do sistema. Exemplo: UML.
- Especificações Matemáticas: são notações com base em conceitos matemáticos, como uma máquina de estados finitos e conjuntos. Essas especificações não ambíguas reduzem as discussões entre cliente e fornecedor sobre a funcionalidade do sistema. No entanto, a maioria dos clientes não compreende as especificações formais e reluta em aceitá-las no momento de uma contratação de sistema.



Documento de Requisitos de Software – DRS

O documento de requisitos de software (DRS) é a declaração oficial do que é exigido dos desenvolvedores de sistema. Em inglês é comum o uso da sigla SRS (*software requirements specification*).

Ele deve incluir os requisitos de usuário para um sistema e uma especificação detalhada dos requisitos de sistema. Em alguns casos, os requisitos de usuário e de sistema podem ser integrados em uma única descrição. Em outros casos, os requisitos de usuário são definidos em uma introdução à especificação dos requisitos de sistema.

Se houver um grande número de requisitos, os requisitos detalhados de sistema poderão ser apresentados como documentos separados.

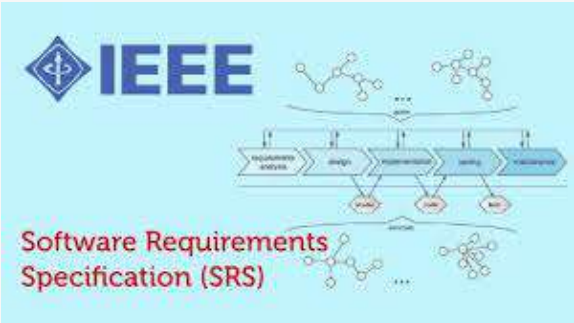
Vários usuários terão acesso a esse artefato. KOTONYA (1998) menciona os possíveis usuários e como eles utilizam o documento.

- Clientes/Gestores do Sistema: especificam os requisitos e os leem para verificar se eles atendem às suas necessidades. Especificam as mudanças nos requisitos.
- Gerentes: utilizam o documento de requisitos para planejar um pedido de proposta para o sistema e para planejar o processo de desenvolvimento do sistema.
- Engenheiros de Sistema: utilizam os requisitos para compreender que sistema deve ser desenvolvido.
- Engenheiros de Teste: utilizam os requisitos para desenvolver testes de validação para o sistema.
- Engenheiros de Manutenção: utilizam os requisitos para ajuda a compreender o sistema e as relações entre suas partes.

Processo de Requisitos

Documento de Requisitos de Software – DRS

Uma série de diferentes organizações de grande porte, como o Departamento de Defesa dos Estados Unidos e o IEEE, definiram padrões para os documentos de requisitos. O padrão mais amplamente conhecido é o padrão IEEE/ANSI 830-1993 (IEEE, 1993). Esse padrão do IEEE sugere a seguinte estrutura para os documentos de requisitos:



1 - Introdução
1.1 - Propósito do documento de requisito
1.2 - Escopo do produto
1.3 - Definições, acrônimos e abreviações
1.4 - Referências
1.5 - Visão geral do restante do documento

2 - Descrição geral
2.1 - Perspectiva do produto
2.2 - Funções do produto
2.3 - Características do usuário
2.4 - Restrições gerais
2.5 - Suposições e dependências

3 - Requisitos específicos que abrangem os requisitos funcionais, não funcionais e de interface.

Essa é a parte mais substancial do documento. Devido à ampla variabilidade na prática organizacional, não é apropriado definir uma estrutura-padrão para essa seção. Os requisitos podem documentar interfaces externas, descrever funcionalidade e desempenho de sistema, especificar requisitos lógicos de banco de dados, restrições de projeto, etc.

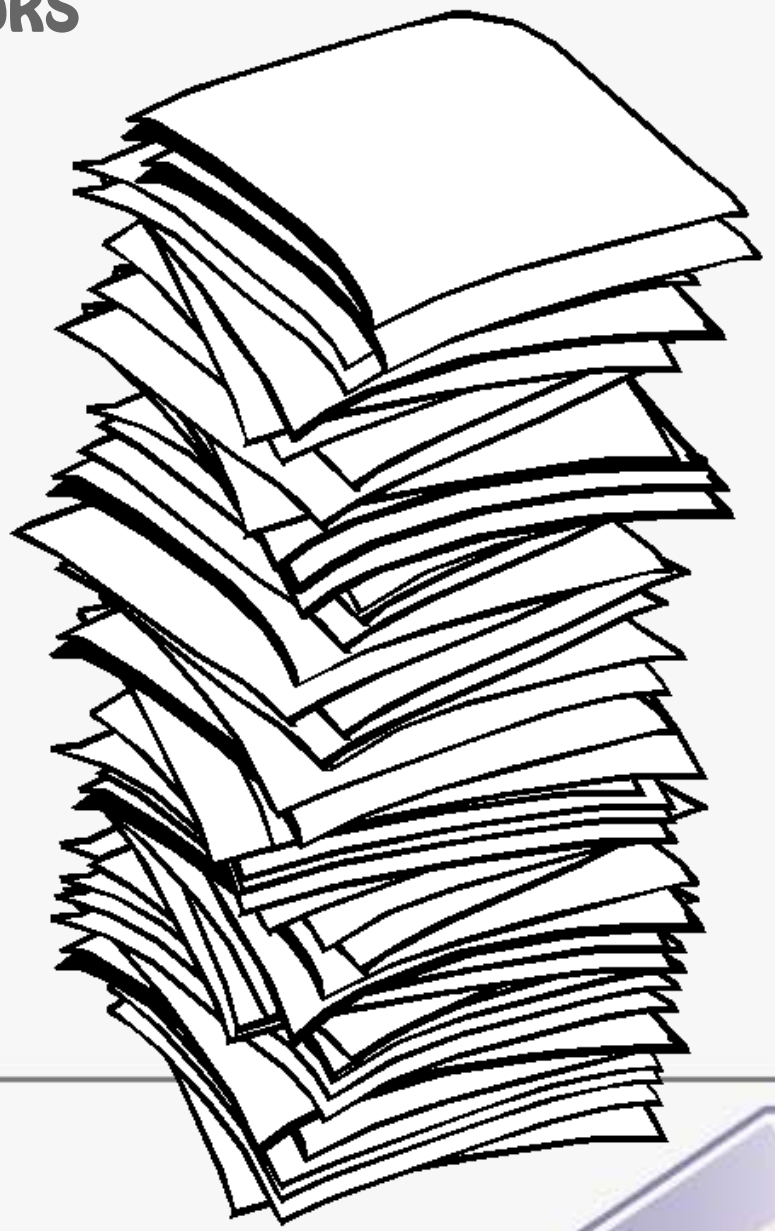
4 - Apêndices
5 - Índice

Documento de Requisitos de Software – DRS

Observe que o padrão IEEE é nada mais do que isso: um padrão. Ou seja, não é obrigatório você criar o seu DRS com as mesmas seções e na mesma estrutura desse padrão. Você pode cortar algumas seções ou incluir novas.

As informações incluídas em um documento de requisitos devem depender do tipo de software que está sendo desenvolvido e da abordagem de desenvolvimento que é utilizada.

Quando o software é parte de um grande projeto de engenharia de sistemas, que inclui sistemas de hardware e software que interagem entre si, frequentemente, é essencial definir os requisitos com um refinado nível de detalhes. Para documentos extensos, é particularmente importante incluir uma tabela abrangente de conteúdo e um índice de documentos, de modo que os leitores possam encontrar as informações de que precisam.



Gerenciamento de Projetos

Introdução

A necessidade de gerenciamento é uma importante distinção entre o desenvolvimento profissional de software e a programação em nível amador.

Necessitamos do gerenciamento de projetos de software porque a engenharia de software profissional está sempre sujeita às restrições de orçamento e de prazo. Essas restrições são estabelecidas pela organização que desenvolve o software.

O trabalho do gerente de projeto de software é garantir que o projeto de software cumpra essas restrições e entregue um produto de software que contribua para as metas da empresa.

Os gerentes de software são responsáveis por planejar e programar o desenvolvimento do projeto. Eles supervisionam o trabalho para assegurar que ele seja realizado em conformidade com os padrões requeridos e monitoram o progresso para verificar se o desenvolvimento está dentro do prazo e do orçamento.

O bom gerenciamento não pode garantir o sucesso do projeto. Contudo, o mau gerenciamento geralmente resulta no fracasso do projeto. O software é entregue com atraso, custa mais do que originalmente foi estimado e apresenta falha no cumprimento de seus requisitos.

Introdução

Os gerentes de software fazem o mesmo tipo de trabalho que outros gerentes de projeto de engenharia. No entanto, a engenharia de software é diferente de outros tipos de engenharia, o que podem tomar o gerenciamento de software particularmente difícil. Algumas dessas diferenças:

1) O produto é incorpóreo. O gerente do projeto de construção de um prédio pode ver o produto sendo desenvolvido. Se há um atraso na programação, o efeito no produto é visível. Partes da estrutura estão obviamente inacabadas. O software é intangível, não pode ser visto ou tocado.

Os gerentes de projeto de software não podem ver o progresso, eles dependem de outras pessoas para produzir a documentação necessária, a fim de examinar o progresso.

2) Não há um processo padrão. Não temos uma compreensão clara das relações entre o processo de software e os tipos de produto. Nas disciplinas de engenharia com longo histórico, o processo é experimentado e testado. Nossa compreensão do processo de software se desenvolveu significativamente nos últimos anos. No entanto, ainda não podemos prever com certeza quando um processo de software específico poderá causar problemas de desenvolvimento.

3) Cada novo projeto é único. Os projetos de software são normalmente diferentes de projetos anteriores. Os gerentes têm ampla experiência anterior, que pode ser utilizada para reduzir a incerteza nos planos futuros. No entanto, é mais difícil prever problemas. Além disso, as rápidas mudanças tecnológicas em computadores e nas comunicações desatualizam as experiências prévias. As lições aprendidas com essas experiências podem não ser transferíveis para novos projetos.

Atividades de Gerenciamento

É muito difícil fornecer dar uma descrição do trabalho-padrão de um gerente de software. O trabalho varia muito, dependendo da organização e do produto a ser desenvolvido. No entanto, a maioria dos gerentes assume a responsabilidade, em algum estágio, por algumas das seguintes atividades ou por todas elas:

- Elaboração de propostas.
- Planejamento e programação de projetos custo do projeto.
- Monitoramento e revisões de projetos.
- Seleção e avaliação de pessoal.
- Elaboração de relatórios e apresentações.

O primeiro estágio de um projeto de software pode envolver a elaboração de uma proposta para executar o projeto. A proposta descreve os objetivos do projeto e como ele será realizado. Em geral, inclui também estimativas de custo e programação.

A elaboração da proposta é uma tarefa crucial, uma vez que a existência de muitas empresas de software depende de um número suficiente de propostas aceitas e contratos firmados.

Para empresas pequenas, a elaboração de propostas pode se tornar um desafio.

Imagine uma empresa formada pelo proprietário e outros cinco profissionais: analistas e programadores. Eles já tem um sistema rodando no mercado e estão atarefados na manutenção e melhoria desse sistema. Então eles recebem um e-mail ou uma ligação de um prospectivo cliente para fazer um novo sistema. O tal cliente solicita uma proposta. É bem capaz de o proprietário e um de seus colaboradores se envolverem com essa proposta durante uma semana ou mais. No entanto, não é garantido que o prospectivo cliente vá contratar a empresa para desenvolver o sistema.

Atividades de Gerenciamento

Daí surge um dilema: deve-se cobrar para fazer uma proposta? A ideia parece surreal e é bem provável que o prospectivo cliente diga não a essa cobrança e procure outro fornecedor. Numa empresa pequena, como a citada no exemplo, pode ser um custo muito alto tirar alguns colaboradores para elaborar propostas que talvez não surtam efeito.

O planejamento de projeto se ocupa de identificar as atividades, os marcos e os documentos a serem produzidos em um projeto. Um plano deve, então, ser traçado para guiar o desenvolvimento em direção aos objetivos do projeto. A estimativa de custos é uma atividade que se ocupa de estimar os recursos requeridos para realizar o projeto.

O monitoramento de projeto é uma atividade contínua. O gerente deve manter o acompanhamento do andamento do projeto e comparar os progressos e custos reais com os que foram planejados.

Embora a maioria das organizações tenha mecanismos formais para o monitoramento, um gerente hábil pode formar um quadro mais nítido do que está acontecendo mediante discussões informais com a equipe de projetos.

O monitoramento informal pode prever problemas em potencial no projeto, no sentido de que pode revelar dificuldades à medida que elas ocorrem. Por exemplo, discussões diárias com a equipe de projetos podem revelar um problema específico ao encontrar uma falha de software.

Durante um projeto, é normal ocorrer uma série de revisões formais pelo gerenciamento de projetos. Essas revisões se ocupam de examinar o progresso geral, o desenvolvimento técnico do projeto, considerando o status do projeto em relação aos objetivos da organização que autoriza a operação do software.

Planejamento

O gerenciamento eficaz de um projeto de software depende de um planejamento acurado do andamento do mesmo. O gerente de projeto deve prever os problemas que podem surgir e preparar soluções experimentais para esses problemas. Um plano traçado no início do projeto deve ser utilizado como guia. O plano inicial deve ser o melhor possível, em face das informações disponíveis. Ele deve evoluir à medida que o projeto seja desenvolvido e melhores informações se tornem disponíveis.

Em algumas organizações, o plano de projeto é um documento único que incorpora todos os diferentes tipos de plano apresentados anteriormente. Em outros casos, o plano de projeto se ocupa exclusivamente do processo de desenvolvimento. São incluídas referências a outros planos, mas os planos são separados entre si. Uma boa estrutura de plano de projeto teria as seguintes seções:

1) Introdução: descreve brevemente os objetivos do projeto e define as restrições.
2. Organização de Projeto: descreve o modo como a equipe de desenvolvimento é organizada.
3. Análise de Riscos: descreve possíveis riscos de projeto, a probabilidade de surgir esses riscos e as estratégias propostas para a redução deles.
4. Requisitos de Hardware e Software: descreve o hardware e o software de apoio exigidos para realizar o desenvolvimento.
5. Estrutura Analítica: descreve a divisão do trabalho em atividades e identifica os marcos e os produtos a serem entregues com cada atividade.
6. Programação de Projeto: descreve as dependências entre atividades, o tempo estimado requerido para atingir cada marco e a alocação de pessoas nas atividades.
7. Monitoramento e de Elaboração de Relatórios: descreve os relatórios de gerenciamento que devem ser produzidos.

Planejamento

Os gerentes de projeto revisam as suposições sobre o projeto, à medida que mais informações se tomam disponíveis, refazendo a programação do mesmo.

Se o projeto estiver atrasado, talvez tenham de renegociar com o cliente as restrições do projeto e os produtos a serem entregues.

Se essa renegociação não for bem sucedida e a programação não puder ser cumprida, poderá ser considerada uma revisão técnica do projeto.

O objetivo dessa revisão é encontrar uma abordagem de desenvolvimento alternativa, que se limite às restrições do projeto e cumpra a programação.

Assim como um plano de projeto, os gerentes podem também ter de traçar outros tipos de planos.

- Plano de Qualidade: descreve os procedimentos para teste de qualidade que serão em um projeto.
- Plano de Validação: descreve a abordagem, os recursos e o método utilizados para a validação do sistema.
- Plano de Gerenciamento de Configuração: descreve os procedimentos de gerenciamento e as estruturas a serem utilizadas.
- Plano de Manutenção: prevê os requisitos de manutenção dos sistema, os custos de manutenção e o esforço necessário.
- Plano de Desenvolvimento em Equipe: descreve como as habilidades e a experiência dos membros da equipe de projeto serão desenvolvidas.

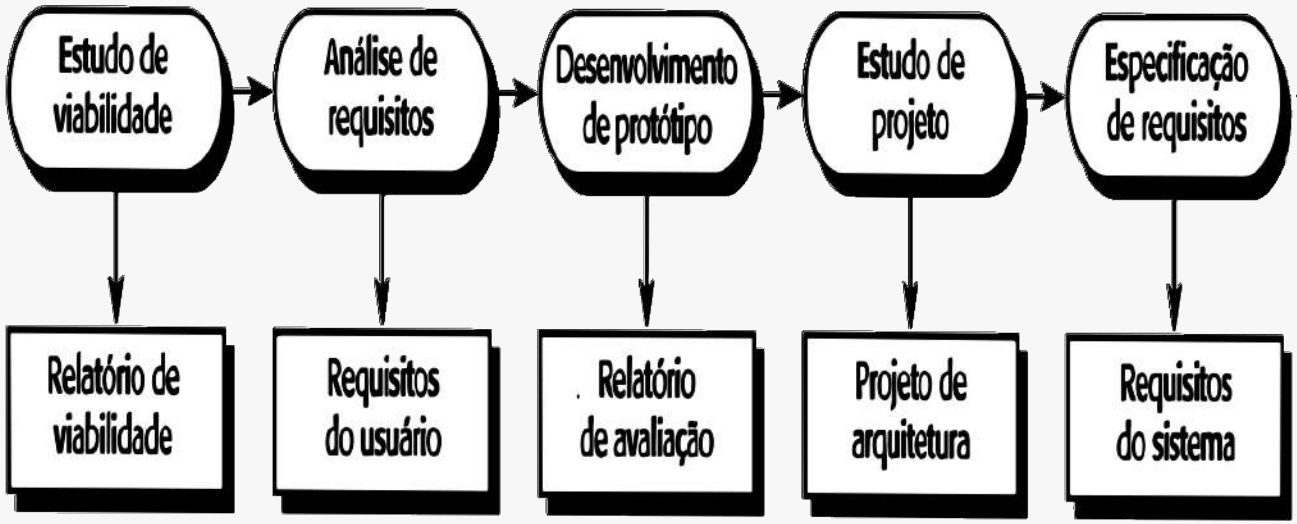
Planejamento

Os gerentes precisam de informações. Como o software é intangível, essas informações somente podem ser fornecidas como documentos que descrevem o estado do software que está sendo desenvolvido. Sem essas informações, é impossível julgar o progresso e as estimativas de custos e as programações não podem ser atualizadas.

Uma forma de realizar o planejamento é através de tarefas e marcos. Um marco é o ponto final de uma atividade de processo de software. A cada marco deve haver uma saída formal, como um relatório, que possa ser apresentado para a gerência.

Os marcos podem ser resultados internos do projeto, que são utilizados pelo gerente de projeto na verificação do andamento do projeto, sem serem entregues ao cliente. Normalmente, os produtos a serem entregues são marcos, mas os marcos não precisam ser produtos a serem entregues.

Para estabelecer marcos, o processo de software deve ser dividido em atividades básicas com saídas associadas.



Programação

A programação de projeto é uma tarefa particularmente necessária para os engenheiros de projeto. Os gerentes estimam o tempo e os recursos exigidos para completar as atividades e os organizam em uma sequência coerente.

As atividades de projeto devem normalmente durar pelo menos uma semana. Subdivisões menores significam que um tempo desproporcional deve ser dedicado para a estimativa e a revisão do diagrama.

Também é útil estabelecer um tempo máximo de duração para qualquer atividade, que vai de oito a dez semanas. Se uma atividade demorar mais tempo do que isso, ela deverá ser subdividida para a programação e o planejamento do projeto.

Ao estimar as programações, os gerentes não devem presumir que todos os estágios do projeto estarão livres de problemas. Imprevistos acontecem: pessoas doentes, problemas de hardware, etc.

Os gerentes devem estimar os recursos necessários para completar cada tarefa, além dos prazos de execução. O recurso principal é o esforço humano requerido. Outros recursos podem ser o espaço em disco exigido em um computador, o tempo de uso necessário de um hardware especializado, como um simulador, e o orçamento de viagens para o pessoal da equipe de projetos.

Uma boa regra prática geral é fazer a estimativa como se nada fosse acontecer de errado e, então, aumentar a estimativa, a fim de poder resolver possíveis problemas.

Por exemplo, se tudo ocorresse sem problemas e o prazo do sistema fosse orçado em 2 meses, aumente esse prazo em 50%, ou seja, mais 1 mês para lidar com os imprevistos.

Gerenciamento de Projetos

Programação

A programação de projeto é normalmente representada como um conjunto de diagramas, mostrando a estrutura analítica do trabalho, dependências das atividades e alocação de pessoal.

Observe a tabela abaixo.

Tarefa	Duração (dias)	Dependências
T1	8	
T2	15	
T3	15	T1 (M1)
T4	10	
T5	10	T2, T4 (M2)
T6	5	T1, T2 (M3)
T7	20	T1 (M1)
T8	25	T4 (M5)
T9	15	T3, T6 (M4)
T10	15	T5, T7 (M7)
T11	7	T9 (M6)
T12	10	T11 (M8)

Na tabela é possível identificar as tarefas, duração em dias de cada uma e suas dependências. Por exemplo, a tarefa T3 depende da tarefa T1, ou seja, essa tarefa só pode ser iniciada depois da conclusão da outra.

Veja também que a tarefa T3 tem uma dependência e logo depois dela chega-se no primeiro marco (M1).

Através da análise dessa tabela é possível ter uma noção de como o projeto vai se desenrolar, de sua programação.

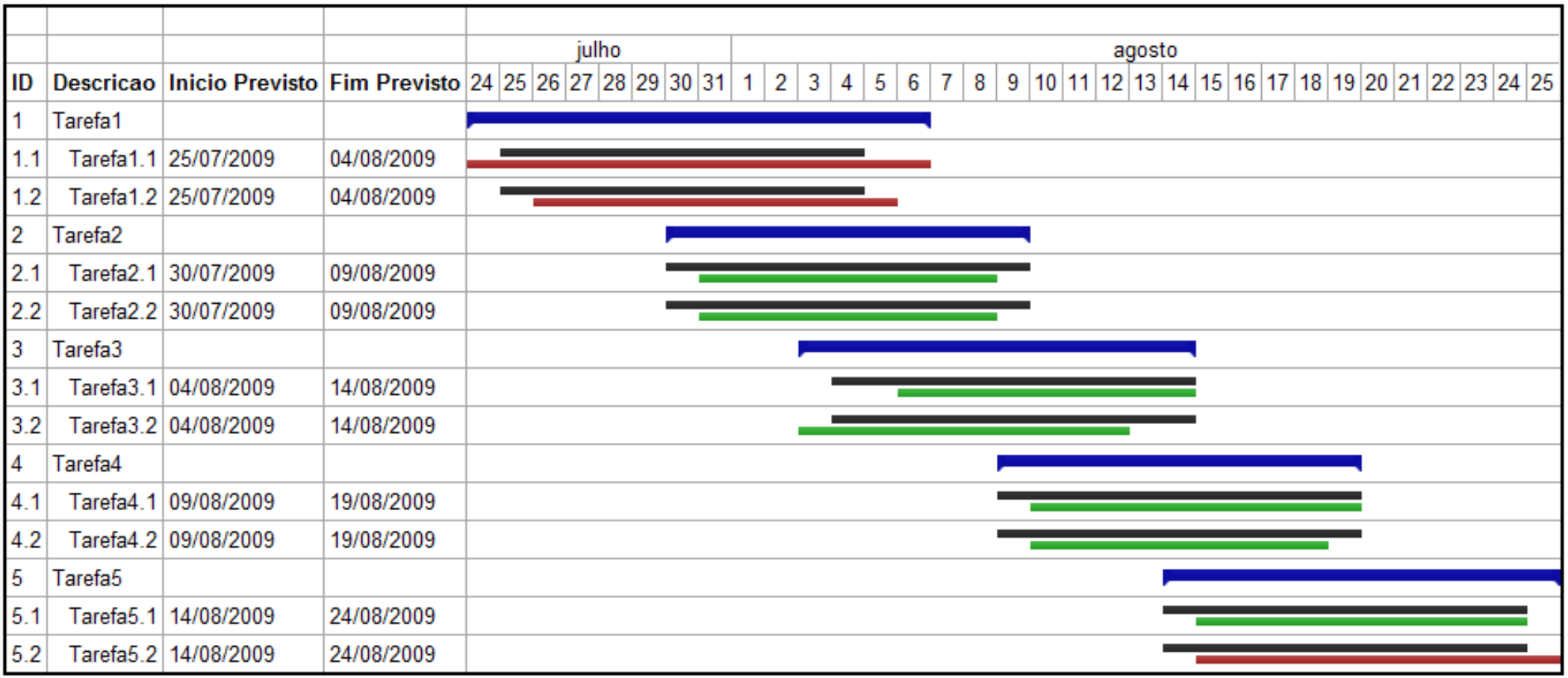
Mas ela não permite uma análise visual acurada. É preciso transformar essa tabela num diagrama que mostre de forma visual como as coisas vão se desenrolar, incluindo o calendário do projeto, início e fim das atividades e a amplitude de atraso possível. É possível fazer isso através de um diagrama de gantt ou diagrama de barras.

Gerenciamento de Projetos

Programação – Diagrama de Gantt

Observe a descrição das tarefas, as datas e a legenda que explica o que significam as cores utilizadas no diagrama.

ID	Descricao	Inicio Previsto	Fim Previsto
1	Tarefa1		
1.1	Tarefa1.1	25/07/2009	04/08/2009
1.2	Tarefa1.2	25/07/2009	04/08/2009
2	Tarefa2		
2.1	Tarefa2.1	30/07/2009	09/08/2009
2.2	Tarefa2.2	30/07/2009	09/08/2009
3	Tarefa3		
3.1	Tarefa3.1	04/08/2009	14/08/2009
3.2	Tarefa3.2	04/08/2009	14/08/2009
4	Tarefa4		
4.1	Tarefa4.1	09/08/2009	19/08/2009
4.2	Tarefa4.2	09/08/2009	19/08/2009
5	Tarefa5		
5.1	Tarefa5.1	14/08/2009	24/08/2009
5.2	Tarefa5.2	14/08/2009	24/08/2009



Gerenciamento de Riscos

Uma importante tarefa de um gerente de projeto é prever os riscos que podem afetar a programação do projeto ou a qualidade do software em desenvolvimento e tomar as medidas necessárias para evitar esses riscos.

Podemos pensar no risco como uma probabilidade de que alguma circunstância adversa realmente venha a ocorrer. Os riscos podem ameaçar o projeto, o software que está sendo desenvolvido ou a organização.

Podemos dividir os riscos em categorias:

1) Relacionados ao Projeto: são os riscos que afetam a programação ou os recursos do projeto.
2) Relacionados ao Produto: são os riscos que afetam a qualidade ou o desempenho do software que está em desenvolvimento.
3) Riscos para os Negócios: são os riscos que afetam a organização que está desenvolvendo ou adquirindo o software.

O gerenciamento de riscos é particularmente importante para projetos de software, devido às incertezas inerentes que a maioria dos projetos enfrenta.

Os riscos podem surgir como decorrência de requisitos mal definidos, de dificuldades em estimar o prazo e os recursos necessários para o desenvolvimento de software, da dependência de habilidades individuais e de mudanças nos requisitos, em razão de modificações nas necessidades do cliente.

O gerente de projeto deve prever riscos, compreender o impacto desses riscos no projeto, no produto e nos negócios e tomar providências para evitar esses riscos.

Planos de contingência podem ser traçados para que, se os riscos realmente ocorrerem, seja possível uma ação imediata que vise à recuperação.

Gerenciamento de Projetos

Gerenciamento de Riscos

Os tipos de riscos que podem afetar um projeto dependem do projeto e do ambiente organizacional em que o software está sendo desenvolvido.

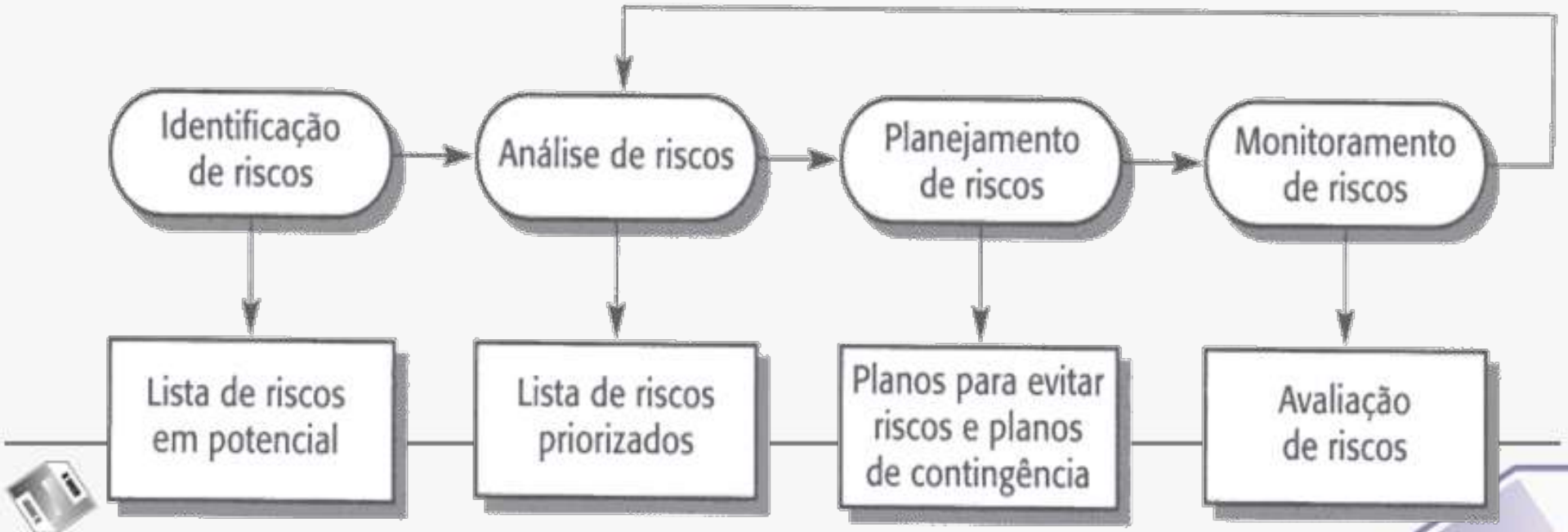
No entanto, muitos riscos são considerados universais e alguns deles são descritos na tabela ao lado.

Risco	Tipo de risco	Descrição
Rotatividade de pessoal	Projeto	O pessoal experiente deixará o projeto antes do término.
Mudança de gerenciamento	Projeto	Haverá uma mudança no gerenciamento organizacional, com a definição de prioridades diferentes.
Indisponibilidade de hardware	Projeto	O hardware essencial ao projeto não será entregue dentro do prazo.
Alteração nos requisitos	Projeto e produto	Haverá maior número de mudanças nos requisitos do que o previsto.
Atrasos na especificação	Projeto e produto	As especificações de interfaces essenciais não estavam disponíveis dentro dos prazos.
Tamanho subestimado	Projeto e produto	O tamanho do sistema foi subestimado.
Baixo desempenho de ferramentas CASE	Produto	As ferramentas CASE que apóiam o projeto não apresentam desempenho conforme o previsto.
Mudanças na tecnologia	Negócios	A tecnologia básica sobre a qual o sistema está sendo construído foi superada por nova tecnologia.
Concorrência com o produto	Negócios	Um produto concorrente foi lançado no mercado, antes que o sistema fosse concluído.

Gerenciamento de Riscos

Como se dá o processo de gerenciamento dos riscos?

1) Identificação de Riscos: são identificados os possíveis riscos de projeto, produto e negócios.
2) Análise de Riscos: são avaliadas as possibilidades e as consequências da ocorrência desses riscos.
3) Planejamento de Riscos: são traçados planos para enfrentar os riscos, seja evitando-os seja minimizando seus efeitos sobre o projeto.
4) Monitoramento de Riscos: o risco é constantemente avaliado e os planos para a diminuição de riscos são revisados, à medida que mais informações sobre eles se tomam disponíveis.



Gerenciamento de Requisitos

Aspectos Gerais

O gerenciamento de requisitos é um modelo sistemático para identificar, organizar e documentar os requisitos do sistema, e estabelecer e manter acordo entre o cliente e a equipe do projeto nos requisitos variáveis do sistema.

Os principais itens para o gerenciamento eficiente de requisitos incluem manter uma declaração clara dos requisitos, juntamente com atributos aplicáveis para cada tipo de requisito e rastreabilidade.

Rastreabilidade é a capacidade de rastrear um elemento do projeto a outros elementos correlatos, especialmente aqueles relacionados a requisitos.

Os elementos do projeto envolvidos em rastreabilidade são chamados de itens de rastreabilidade. Os itens típicos de rastreabilidade incluem diferentes tipos de requisitos.

A coleta de requisitos pode parecer uma tarefa precisa. Nos projetos reais, entretanto, você encontrará dificuldades. Motivos:

- Nem sempre os requisitos são óbvios e podem vir de várias fontes.
- Nem sempre é fácil expressar os requisitos claramente em palavras.
- Existem diversos tipos de requisitos em diferentes níveis de detalhe.
- O número de requisitos poderá impossibilitar a gerência se não for controlado.
- Os requisitos estão relacionados uns com os outros.
- Os requisitos têm propriedades exclusivas ou valores de propriedade.
- Há várias partes interessadas, o que significa que os requisitos precisam ser gerenciados por grupos de pessoas de diferentes funções.
- Os requisitos são alterados.

Aspectos Gerais

Que habilidades é preciso ter para gerenciar essas dificuldades?

Analisar devidamente o problema é uma delas. A análise do problema é feita para compreender os problemas e as necessidades iniciais dos envolvidos, e propor soluções de alto nível.

É necessário ter uma noção básica sobre as necessidades dos envolvidos. Os requisitos vêm de várias fontes, como clientes, parceiros, usuários finais e peritos do domínio. Você precisa saber o melhor modo de determinar quais devem ser as fontes, como obter acesso a essas fontes e qual a melhor forma de levantar as informações delas.

É preciso definir o sistema. Isso significa traduzir e organizar as necessidades dos envolvidos em descrições significativas do sistema a ser construído.

É necessário gerenciar o escopo do projeto. É preciso priorizar os requisitos, com base em retorno dado por todos os envolvidos, e gerenciar o seu escopo. Você também precisa ter controle das origens dos requisitos, da aparência dos produtos liberados pelo projeto e do processo de desenvolvimento propriamente dito.

A definição do sistema deve passar por um refinamento. A definição detalhada do sistema precisa ser apresentada de maneira que os envolvidos possam entendê-la, concordar com ela e sair dela.

Por fim, é preciso gerenciar os requisitos variáveis. Você precisa certificar-se de compor uma estrutura de requisitos que seja adaptável a mudanças, e de usar vínculos de rastreabilidade para representar as dependências entre os requisitos e outros artefatos do ciclo de vida do desenvolvimento.

Gerenciamento de Requisitos com Casos de Uso

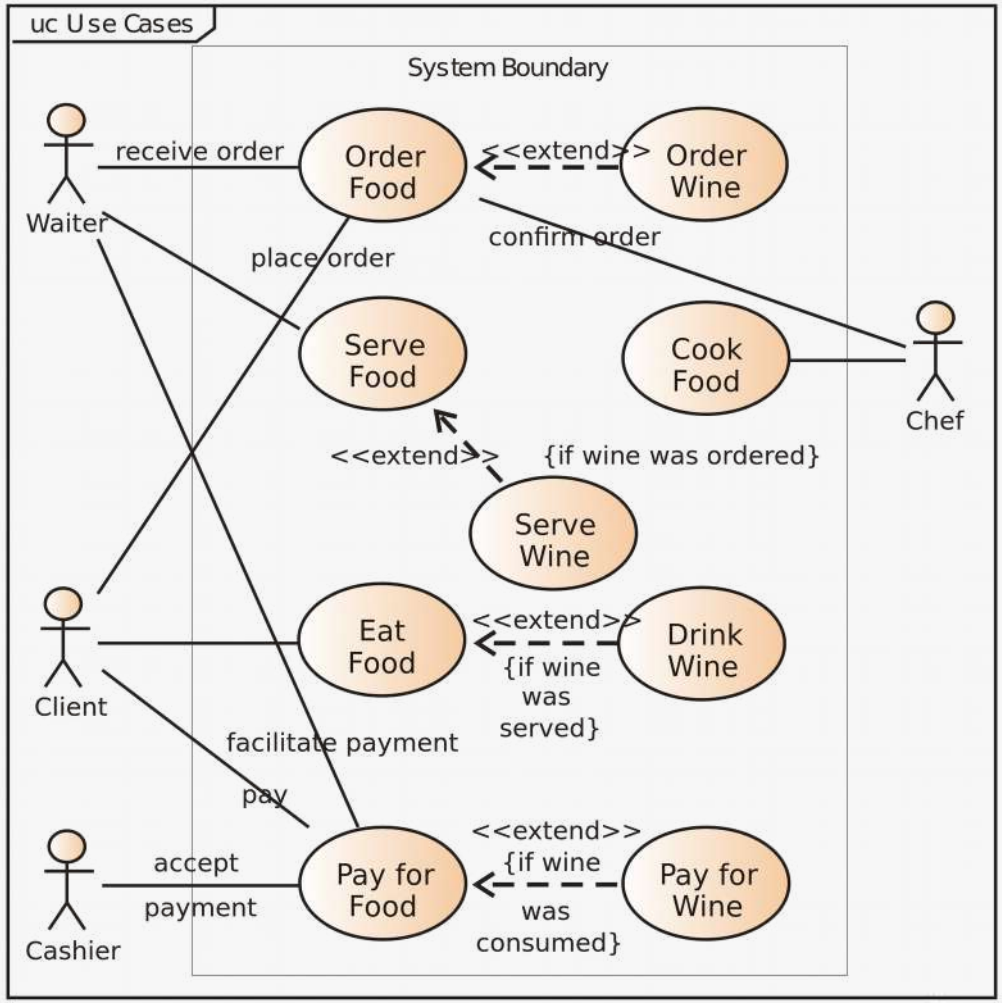
Introdução

O modelo de casos de uso captura os requisitos funcionais e força o desenvolvedor a pensar em como os agentes externos interagem como o sistema.

No entanto, esse modelo corresponde somente aos requisitos funcionais. Outros tipos de requisitos (desempenho, interface, segurança, regras do negócio etc.) que fazem parte do documento de requisitos de um sistema não são considerados pelo modelo de casos de uso.

Considerando que o processo de desenvolvimento incremental é utilizado, a identificação da maioria dos atores e casos de uso é feita pelos analistas na fase de concepção.

Veremos sobre processos de software e sobre UML mais profundamente em livros posteriores.



Usando Casos de Uso no Processo

A descrição dos casos de uso considerados mais críticos começa na fase de concepção, que termina com 10% a 20% do modelo de casos de uso completo.

Na fase de elaboração, a construção do modelo continua de tal forma que, ao seu término, 80% do modelo de casos de uso esteja construído. Na fase de construção, casos de uso formam uma base natural através da qual podem-se realizar as iterações do desenvolvimento. Um grupo de casos é alocado a cada interação.

O processo continua até que todos os casos de uso tenham sido desenvolvidos e o sistema esteja completamente construído. Esse tipo de desenvolvimento é também chamado de desenvolvimento dirigido a casos de uso.

Casos de uso que mencionam detalhes de interface gráfica são indesejáveis durante a análise.

O mais adequado é utilizar casos de uso essenciais e, posteriormente, na etapa de projeto, transformá-los em casos de uso reais adicionando mais detalhes.

Na fase de análise, descrições de casos de uso devem capturar os requisitos funcionais do sistema e ignorar aspectos de projeto, como a interface gráfica com o usuário.

O modelo de casos de uso é uma ferramenta fundamental para o gerente de um projeto no planejamento e controle de um processo de desenvolvimento incremental e iterativo. Ao final de cada iteração, o gerente pode avaliar a produtividade na realização das tarefas. Essa avaliação serve como massa de dados para que esse profissional realize a alocação das tarefas e recursos para as próximas interações.

Referências

ABREU, M. C. O Professor Universitário em Sala de Aula. São Paulo, MG Associados, 1990

BERRY, D. M.; LAWRENCE, B. Requirements engineering. 1. ed.

BEZERRA, E. Princípios de Análise e Projeto de Sistemas com UML. Rio de Janeiro, Campus, 2002.

FIORINI, S. T. & STAA, A. & BAPTISTA, R. M. Engenharia de Software com CMM. Rio de Janeiro, Brasport, 1998.

GAUSE, D. C.; WEINBERG, G. M. Are your lights on? 1. ed. USA: Dorset House Publishing Co. Inc., 1990.

MULLER, R. J. Projeto de Banco de Dados. São Paulo, Berkeley, 2002.

REZENDE, D. A. Engenharia de Software e Sistemas de Informação. Rio de Janeiro, Brasport, 2002.

SOMMERVILLE, I. Engenharia de Software. São Paulo, 2011.

SEBESTA, R. Conceitos de linguagens de programação. 9. ed. Porto Alegre: Bookman, 2011.

MACLENNAN, Bruce J. Principles of Programming Languages: Design, Evaluation and Implementation (em inglês). 3ª ed. Oxford: Oxford University Press, 1999.

BAL, Henri E.; Grune, Dick. Programming Language Essentials (em inglês). Wokingham: Addison-Wesley, 1994.

Wikipedia, Consultas em Janeiro/2016.