Instituto de Matemática e Estatística – Universidade de São Paulo

Engenharia de requisitos: Um estudo comparativo de dois métodos utilizados na indústria

Autor: Mario Marques Junior
Orientador: Flávio Soares Corrêa da Silva

2005

Introdução

Antes de assistir uma aula de Engenharia de Software, achava que desenvolver um sistema, embora sendo algo que eu desejasse, era uma tarefa muito além das minhas capacidades. Já ouvi histórias de pessoas que, tendo o mesmo nível de conhecimento que eu, batiam no peito e falavam "eu consigo, deixe comigo", mas eu não pensava assim. Antes, achava minhas habilidades em programação tão elementares que nunca conseguiria fazer um programa que fosse útil tanto para mim como para outras pessoas, tendo alguma funcionalidade a mais que ilustrar algum conceito teórico que me permitisse ser aprovado em alguma disciplina da graduação.

Depois do contato com essa disciplina, e dentro dela mais especificamente a Engenharia de Requisitos, pude divisar uma forma de tornar essa necessidade pessoal uma realidade. Seguir seus ensinamentos permite visualizar o sistema todo, em detalhes, *antes* do desenvolvimento; fazendo com que pessoas como eu fiquem muito mais seguras dos passos a seguir para o sucesso do projeto, e previnindo que pessoas como essas do comentário acima se frustrem com um projeto que nunca chega ao fim e se torna mais difícil quanto mais se avança nele.

Esse texto é direcionado para estudantes de graduação em Ciências da Computação que precisam de um ponto de partida para estudar essa disciplina; e analistas ou desenvolvedores que estão nas empresas enfrentando dificuldades com seus projetos, e reconhecem a urgência em adotar práticas que reduzam seus problemas relativos levantamento e manutenção de requisitos. Seria interessante que professores da graduação pudessem usar esse texto como estopim para discussões a respeito do tema, mas é imprescindível para os três grupos de leitores acima mencionados que tomem conhecimento do conteúdo dos dois livros estudados antes de colocar qualquer coisa em prática.

A monografia foi organizada em duas partes; a primeira contendo uma pequena introdução à Engenharia de Requisitos, um resumo das características dos livros comparados, e por fim a comparação, com uma conclusão sintetizando a discussão. Os analistas e desenvolvedores devem achar esse trecho mais útil para suas necessidades. A segunda parte é um tutorial, listando as boas práticas de engenharia de requisitos, e referenciando as duas obras comparadas para maior aprofundamento nos tópicos mencionados. Espero que você aprecie essa leitura!

Parte I – Estudo Comparativo

A idéia geral da Engenharia de Requisitos

Bem vindo ao mundo da engenharia de requisitos. Essa disciplina, antes de mais nada, serve para suprir a necessidade de desenvolvedores de software, que lutam para manter seus projetos dentro do prazo e com o custo controlado. É sabido que a maior parte dos gastos em re-trabalho e correção de sistemas se originam em problemas na fase de desenvolvimento de requisitos. Ou seja, são "defeitos" de software que já estavam lá no início do projeto, mesmo antes de qualquer linha de código estar escrita; mas são descobertos bem depois, quando muito tempo e dinheiro já foi gasto no desenvolvimento de um sistema que está, a bem da verdade, errado, incoerente com os objetivos de quem o requisitou.

Antes de entrar na comparação entre as duas teorias, precisamos entender os fundamentos do processo de desenvolvimento de software. Esse processo pode ser entendido como uma série de fases subseqüentes e iterativas. Começamos pela fase de **Desenvolvimento de requisitos**, cujo produto principal é a especificação de requisitos. Com essa especificação em mãos, podemos fazer uma **Análise de sistemas** baseado nos requisitos, estimando custo, risco, organizando a equipe para trabalhar no projeto, e assim vai. A próxima fase é o **Design do produto**, na qual os analistas delimitam, baseado nos requisitos e na análise, as restrições de desenvolvimento do produto; bem como a estrutura de classes, as interfaces, etc. Depois disso vem a **Construção** e o **Uso do produto**. Como disse anteriormente, essas fases são iterativas, ou seja, após o produto ser entregue para uso, novos requisitos são descobertos. Essas fases também podem se sobrepõem, e todas as fases retro-alimentam a fase de desenvolvimento de requisitos.

O processo de desenvolvimento de requisitos, por sua vez, é composto das seguintes fases: **Levantamento, Análise de requisitos, Especificação** e **Validação**. Também é um processo iterativo, uma vez que enquanto você atua no desenvolvimento de uma das fases, surgem questões e correções que remetem às fases anteriores. No levantamento de requisitos, você usa técnicas de relacionamento com o cliente e usuários para delimitar o escopo do sistema, descobrir funcionalidades, e questões relativas ao ambiente de trabalho e infra-estrutura, por exemplo. Na análise, você modela os requisitos de forma a verificar a factibilidade de implementação deles, suas prioridades, criar um dicionário do sistema, entre outros. Esses requisitos devem ser escritos, na especificação, de forma a organizá-los para serem constantemente lidos, modificados, removidos, comparados e classificados. Por fim, na validação, além de revisar a especificação dos requisitos, você deve testá-los para verificar se eles são coerentes entre si e aderem à especificação inicial do escopo do sistema.

As duas teorias

Compreendendo como se dá processo de desenvolvimento de software, vamos agora às idéias dos dois autores. Vamos começar por [Rob99]. Ele é conhecido por propor um método, chamado Volere, e dois modelos de documento: um para documentar cada requisito individualmente, utilizando uma ficha em papel; e outro para organizá-los em conjunto, na especificação de requisitos. O modelo de especificação é voltado para tipos de requisitos. Ou seja, cada seção da especificação compreende um tipo de requisito, e a tarefa central do método é preencher todas as seções com os requisitos pertinentes; sem os quais, segundo os autores, você não tem garantias de desenvolvimento posterior. O método tem passos bem determinados desde o princípio; ele determina os produtos esperados em cada etapa do processo, necessários para a etapa seguinte. Entende-se que você vai usá-lo como um todo, adaptando algumas partes para seu projeto ou equipe.

Em contra partida, em [Wie03], você não encontrará um método. Antes ele adota uma abordagem voltada para boas práticas de Engenharia de Software. A idéia é que, incorporando boas práticas gradualmente em seu ambiente de desenvolvimento, você consegue sistematizar um método personalizado para seus projetos ou sua organização (você pode ver mais sobre boas práticas no tutorial – Parte II). Outra característica é adotar modelos mais variados, de acordo com os padrões mais adotados na indústria. Assim, por exemplo, você pode resolver não organizar os requisitos pelos seus tipos, mas pelos casos de uso ou *features*. Nessa abordagem fica a cargo do analista quais técnicas ele vai incorporar, muito embora as motivações para se adotar cada uma das práticas sejam bem altas.

Rigor versus Flexibilidade

Comparando os dois livros, nota-se que a essência é a mesma, mas a forma com que os autores abordam o tema faz com que cada um sirva para um propósito um pouco diferente. Em Rob99 você tem pouca flexibilidade; em Wie03, total liberdade. O método Volere encontrado em Rob99 é mais rigoroso, espera que você tenha completado toda a especificação para depois começar o desenvolvimento. Isso força o analista a avaliar todos os tipos de requisitos. Esse rigor também implica um maior controle da especificação. Além disso, ele é auto-contido, ou seja, depende somente das técnicas do próprio método (embora ele recomende o uso de ferramentas para organizar os requisitos de projetos grandes). Isso faz com que ele seja mais fácil de aprender, é só seguir o passo a passo; embora tanto Rob99 como Wie03 exijam uma dedicação razoável de tempo para o desenvolvimento de requisitos.

Wiegers, em Wie03, permite a construção do sistema mesmo com a especificação incompleta, desde que isso não comprometa o desenrolar do processo (ou seja, deve-se evitar re-trabalho, a qualquer custo). Essa abordagem ajuda a desenvolver sistemas em vários releases, por exemplo. Como as etapas também são flexíveis, você pode melhorar seu processo de engenharia de requisitos gradualmente, sem o

trauma de implantar um método totalmente diferente do que sua equipe adota. Ele mesmo admite que certas boas práticas podem não compensar ou nem mesmo serem pertinentes para o seu tipo de projeto. Wie03 é mais atual que Rob99, mencionando tópicos voltados para técnicas de modelagem e desenvolvimento contemporâneas, como UML e *eXtreme Programming*. Ele ainda se preocupa em aderir aos padrões de Engenharia de Software adotados pela indústria, como o IEEE; e às certificações SW-CMM e CMMI-SW/SE.

Para finalizar, gostaria de chamar a atenção a dois tópicos que são comuns aos dois autores e que são muito importantes. Primeiro, o tratamento aos usuários. Ambos concordam que o envolvimento dos usuários no desenvolvimento dos requisitos é essencial, a ponto de comprometer todo o projeto no caso de negligência. Wie03 dedica capítulos sobre como os usuários devem ser considerados, como melhorar o envolvimento deles, etc. O outro tópico é sobre as características dos requisitos. Todo requisito deve ser documentado e ter dados para rastreabilidade; como identificação única, origem (quem pediu o requisito), e consequências (como foi modelado e implementado); além de critérios de aceitabilidade do requisito (se ele foi implementado a contento). Tudo para tornar a análise dos requisitos funcional dentro do processo, ou seja, permitir a manutenção e teste de requisitos.

Conclusão

Adotar um processo de desenvolvimento de requisitos certamente é uma tarefa necessária, tanto quanto melhorá-lo; esses cuidados só geram benefícios. Na dúvida entre qual livro escolher, leve em conta a experiência de sua equipe. Rob99 é indicado para grupos que, ou não têm idéia do que seja um requisito, ou não têm nenhum processo formal para desenvolvimento. Já para equipes que têm um processo, e desejam melhorá-lo, Rob99 não é recomendado, pois é mais difícil implantar um método como o Volere entre pessoas viciadas em suas "más práticas" de desenvolvimento. Wie03 tem mais vantagens que seu concorrente, por ser flexível e atual, mas é necessário sensibilidade para saber qual caminho tomar dentro das possibilidades que ele coloca. É recomendado para equipes mais maduras dentro do processo de desenvolvimento de requisitos.

Parte II – Tutorial

Como usar esse tutorial?

Esse tutorial é uma compilação das boas práticas de engenharia de requisitos. Ele oferece uma base das técnicas de desenvolvimento mais usadas na indústria; algumas delas testadas e comprovadas por especialistas e pesquisadores. São práticas que usualmente foram adotadas nos produtos de sucesso, e que fizeram falta nos que fracassaram.

Elas estão agrupadas em áreas: **Conhecimento**, **Desenvolvimento de Requisitos**, **Gerenciamento de Requisitos** e **Gerenciamento de Projeto**. O Desenvolvimento de Requisitos por sua vez se divide em **Levantamento**, **Análise**, **Especificação** e **Validação** de requisitos.

Técnicas relacionadas com o conhecimento têm relação com treinamento de pessoal sobre requisitos e sobre o domínio da aplicação. Já as práticas agrupadas sob o levantamento de requisitos ajudam a desvendar, juntamente com o cliente e os usuários, os requisitos do sistema. A análise de requisitos consiste de técnicas de modelagem de requisitos, para abstrair a visão dos mesmos e permitir maior comunicabilidade. Práticas ligadas à especificação remetem a técnicas de documentação efetivas, enquanto a validação é o processo formal pelo qual o analista decide que um requisito está coerente ou não com as necessidades dos usuários. Por fim, gerenciamento de requisitos e de projeto são práticas ligadas ao controle de mudanças no desenrolar do projeto.

Em algumas das práticas descritas, são encontradas referências às duas bibliografias, para um estudo mais profundo. Essas referências são rotuladas de acordo com o livro que a contém. Assim para [Wie03] essas referências tem os rótulos marcados com WI-1, WI-2 e assim por diante, enquanto para [Rob99] são marcados com RR-1, RR-2. Note que nem todas as práticas têm referências associadas, o que não quer dizer que o livro não aborda a prática. Isso acontece quando a referência encontrada não é suficientemente relevante; mas seguindo à risca a proposta de cada obra, você estará de uma forma ou de outra alcançando tal prática.

Para saber como colocar um processo de desenvolvimento de requisitos em funcionamento, consulte o Apêndice A de Rob99. Para ter idéias de como começar com as novas práticas em seu processo de desenvolvimento, Wie03 dá passos práticos no final de cada capítulo. Além disso consulte a tabela 3-2 na página 58 do mesmo livro.

Convenções

Aqui são definidos alguns termos usados no tutorial. O engenheiro de requisitos é chamado **analista de requisitos**, ou simplesmente analista. Ele é responsável por levantar e organizar todas as informações sobre requisitos do projeto e escrever a especificação, além de comunicar os requisitos para os outros participantes do projeto.

O **cliente** é a pessoa ou empresa que está pagando pelo desenvolvimento do sistema, e espera retorno em valor do produto desenvolvido. Seus requisitos são relacionados com o negócio e geralmente são colocados em um documento de visão e escopo do sistema.

Os **usuários** são as pessoas que irão usar o sistema para produzir o retorno esperado pelo cliente. São eles que provém os requisitos de funcionalidade do produto, que formarão a principal parte da especificação dos requisitos. Esses requisitos devem estar em conformidade com as necessidades do cliente, contidos no documento de visão e escopo. O analista deve ainda explorar requisitos de qualidade do sistema com os usuários.

A despeito de a atividade de requisitos ser também de desenvolvimento; os **desenvolvedores**, aqui, representam a equipe técnica. Eles oferecem suporte ao projeto verificando a factibilidade de requisitos de funcionalidade e qualidade, e colocando os requisitos dentro da perspectiva de suas respectivas limitações tecnológicas. Além disso, são eles que transformarão a especificação em um produto que pode ser usado pelos usuários.

Conhecimento

Treine analistas de requisitos

O analista precisa ser treinado para adquirir as qualidades necessárias de paciência, organização, habilidade interpessoal, de comunicação, e conhecimentos do domínio da aplicação. Precisa também conhecer diversas técnicas de Engenharia de Requisitos.

- **WI-1** Você pode encontrar mais sobre esse tópico no capítulo 4 de Wie03

Eduque usuários representativos e gerentes sobre requisitos

Convença os usuários e gerentes de desenvolvimento e de cliente do valor da Engenharia de Requisitos, e da importância de todas atividades envolvidas. Fale sobre o risco da negligência.

Treine desenvolvedores nos conceitos do domínio da aplicação

Promova um seminário a respeito das atividades do negócio do cliente, terminologia, etc.; para evitar desentendimentos e re-trabalho, e dar um conhecimento básico do domínio da aplicação aos desenvolvedores. Tenha um usuário representativo acompanhando cada desenvolvedor para maiores esclarecimentos.

Crie um glossário do projeto

Para evitar desentendimentos, um glossário com termos especializados do domínio da aplicação é necessário. Adicione a esse glossário tudo que possa causar conflito entre nomes.

- **WI-2** Saiba como usar o glossário em Wie03, Capítulo 10, página 180
- **RR-1** Veja exemplos desse tópico em Rob99, Apêndice B, página 364

Levantamento de requisitos

Defina um processo de desenvolvimento de requisitos

Sistematize e documente os passos que sua equipe faz para levantar, analizar, especificar, e validar requisitos. Torne esses passos um senso comum em seu ambiente de desenvolvimento. Isso ajuda a planejar as tarefas, prazos, e recursos necessários.

- **WI-3** Consulte o Capítulo 3, página 59, para um processo geral, e o Capítulo 22, para melhorar seu processo corrente
- **RR-2** Se preferir usar o método Volere, ele se encontra no Apêndice A

Escreva um documento de visão e escopo

O documento de visão e escopo contém os requisitos de negócio, dados pelo cliente. Ele estabelece os objetivos do produto, e define o que está dentro ou fora do projeto.

- **WI-4** Aprenda a escrever esse documento no Capítulo 5, usando o modelo da pagina 82
- **RR-3** Rob99 insere esse tópico na própria especificação. Veja Apendice B, página 366

Identifique as classes de usuários e suas características

Evite ignorar uma classe de usuário, pois cada classe pode ter necessidades diferentes, e afetar de forma diversa o design do produto.

- **WI-5** Para uma discussão mais profunda, veja o Capítulo 6
- **RR-4** Para saber como identificar os usuários e suas características, consulte Apêndice B, página 358

Selecione um usuário representativo de cada classe de usuário

Procure usuários que podem dizer com precisão as necessidades da classe a qual ele pertence. Esses usuários tem participação constante no projeto e poder de decisão sobre requisitos em nível de usuário.

- **WI-6** Wie99 enfatiza essa prática, e cita as características desse usuário especial. Consulte Capítulo 6

Estabeleça grupos de foco de usuários típicos

Agrupe alguns usuários de versões passadas ou produtos semelhantes, para coletar funcionalidades e características de qualidade. Esse grupo não necessariamente deve ter autoridade para fazer decisões sobre os requisitos.

Identifique casos de uso, através dos usuários representativos

Explore com os usuários as tarefas que eles precisam realizar com o software. Esses são os casos de uso. Documente esses casos de uso em um modelo padrão, pois elees servirão para derivar requisitos de funcionalidade.

- **WI-7** No Capítulo 8 você encontrará a abordagem de casos de uso
- **RR-5** Casos de uso é o tema do Capítulo 4

Identifique eventos e respostas de sistema

Liste os eventos externos (sinais de hardware, dados recebidos, e interações de usuários), e as respectivas respostas esperadas para cada evento.

- **WI-8** Tabelas de eventos e respostas são mencionados no final do Capítulo 8
- **RR-6** Eventos e respostas são usados para construir os casos de uso, no Capítulo 4

Realize *workshops* de levantamento de requisitos

Essas reuniões colocam usuários e analistas em torno dos requisitos e são uma forma poderosa de explorá-los e fazer o rascunho da especificação do projeto.

- **WI-9** Wie03 fala sobre essa técnica com a finalidade de ouvir a voz do usuário, no Capítulo 7
- **RR-7** Rob99 usa essa técnica desde o princípio do processo, no Apêndice A, páginas 282-288 e 299

Observe usuários fazendo seu trabalho

Use essa técnica para desenhar diagramas de fluxo de dados, que ajudam a identificar requisitos relativos às tarefas dos usuários.

- **WI-10** Essa técnica é mencionada no Capítulo 6, página 97
- **RR-8** Também chamada de apreendizagem, é encontrada na página 87 do Capítulo 5 e no Apêndice A, página 294

Examine relatórios de problemas do sistema atual para idéias

Esses relatórios, e pedidos de melhoramento do sistema, são fontes de idéias que certamente devem ser considerados para um projeto de sistema substituto, ou uma nova versão.

- **RR-9** O capítulo 5, página 83, descreve como aproveitar a situação atual do sistema para novos desenvolvimentos. Veja também o Apêndice A, página 294 e 297

Reuse requisitos entre projetos

Requisitos de projetos semelhantes podem ser adaptados, mas isso requer negociação com os usuários para flexibilização dos requisitos.

- **RR-10** Essa técnica é descrita no Capítulo 12

Análise de requisitos

Desenhe um diagrama de contexto

O diagrama de contexto é um modelo de análise que serve para definr os limites e interfaces entre o sistema e as entidades externas (pessoas, software e hardware).

- **WI-11** Você encontra esse tópico no final do Capítulo 5
- **RR-11** O contexto do trabalho e seu respectivo diagrama é mostrado no Capítulo 3, páginas 44-45

Crie protótipos de interface de usuário e protótipos técnicos

As dúvidas sobre certos requisitos podem ser eliminados com a presença de um protótipo: é mais fácil ver como os requisitos devem funcionar em conjunto.

- **WI-12** O Capítulo 13 trata desse tópico
- **RR-12** Esse tópico está no Capítulo 11 desse livro, além do Apêndice A, páginas 316-321

Analise a factibilidade dos requisitos

Essa análise se preocupa com o custo, performance; riscos de conflito com outros requisitos, dependências, impossibilidades técnicas; entre outros.

- **RR-13** Encontrado no Capítulo 10, página 191, e no Apêndice A, página 314

Priorize os requisitos

Prioridades relativas aos requisitos, casos de uso e *features* ajudam a decidir em que versão do sistema um conjunto de funcionalidades estará. Também ajudam a gerenciar mudanças, tanto de requisitos quanto de necessidades dos usuários ou objetivos do cliente.

- **WI-13** Esse tópico é tratado no Capítulo 14
- **RR-14** Está no Apêndice A, página 311

Modele os requisitos

Usando modelos gráficos de análise dos requisitos, você os verifica em um nível maior de abstração; ajudando a encontrar erros, inconsistências, requisitos que estão faltanto ou sobrando.

- **WI-14** Várias formas de modelagem estão especificadas no Capítulo 11

Crie um dicionário de dados

Definindo todos os itens de dados e estruturas associadas ao sistema você evita inconsistência no uso desses dados e na comunicação entre usuários e desenvolvedores.

- **WI-15** Encontre esse tópico discutido no final do Capítulo 10

Faça alocação de requisitos para sub-sistemas

Para lidar com a complexidade de um sistema grande, divida-o em vários componentes e levante requisitos para esses sub-sistemas.

- **WI-16** Uma maior explicação está no Capítulo 17

Aplique *Quality Function Deployment*

Essa técnica serve para relacionar qualidades e funcionalidades do produto com a satisfação dos usuários e do cliente. Isso ajuda a separar requisitos essenciais e não essenciais.

- **RR-15** Uma prática relacionada está no Apêndice A, página 311

Especificação de requisitos

Adote um modelo de especificação de requisitos

Defina um modelo padrão para documentar os requisitos dos seus projetos de forma consistente. Projetos diferentes em tamanho e complexidade precisam de modelos diferentes.

- **WI-17** Um modelo geral é mostrado no Capítulo 10
- **RR-16** O Apêndice B apresenta o modelo padrão do método Volere, também explicado no Capítulo 8

Identifique a fonte dos requisitos

Requisitos podem ter sua origem em casos de uso, regras do negócio, ou pedidos de usuário, entre outros. Lembrar a fonte dos requisitos ajuda no gerenciamento em caso de mudanças.

- **WI-18** A fonte é um atributo necessário, explicado no Capítulo 18
- **RR-17** O Apêndice B mostra um cartão para levantamento de requistos que contém essa informação

Rotule univocamente cada requisito

Com essa identificação, você pode gerenciar mudanças mais facilmente, além de ter um maior controle sobre o rastro do requisito. Deve ser fácil adicionar, remover e modificar requisitos com essa rotulação.

- **WI-19** Encontre esse tópico no Capítulo 10
- **RR-18** O cartão mostrado no Apêndice B pede identificação dos requisitos

Escreva as regras de negócio

As regras de negócio são políticas corporativas, regulamentos governamentais, e algoritmos computacionais. Elas precisam um documento especial, fora da especificação dos requisitos, pois geralmente regras de negócio estão além do escopo de um projeto específico.

- **WI-20** O Capítulo 9 explica como tratar as regras de negócio
- **RR-19** O Apêndice B, páginas 381-383; e o Capítulo 7, páginas 129-132; falam sobre requisitos de ordem política, cultural ou legal

Especifique atributos de qualidade

Atributos de qualidade incluem performance, eficiência, confiabilidade, usabilidade, entre outros. Eles estão além dos requisitos de funcionalidade, mas devem ser explorados e colocados na especificação dos requisitos.

- **WI-21** Atributos de qualidade é o tema do Capítulo 12
- **RR-20** O Capítulo 7 trata dos requisitos de qualidade

Validação de requisitos

- **WI-22** Todos os tópicos de validação são encontrados no Capítulo 15
- **RR-21** A inspeção, teste e validação dos requisitos é o tema do Cápitulo 10

Inspecione documentos de requisitos

Inspeção formal da especificação de requisitos e documentos relacionados é uma das práticas de mais alto valor na engenharia de requisitos. Monte grupos representando as diversas perpectivas no seu projeto para examinar os documentos em busca de erros.

Teste os requisitos

Derive casos para teste de requisitos. Verifique com o cliente e usuários se os passos do teste correspondem ao comportamento esperado do sistema. Garanta que todos os requisitos têm um teste relacionado, mantendo um rastro entre os requisitos e testes.

Defina critério de aceitação

Pergunte aos usuários como eles vão determinar se o produto atende a suas necessidades e se serve para uso. Faça testes sobre os requistos para ver se eles alcançam esse critério definido pelo usuário.

- **RR-22** Rob99 dedica o Capítulo 9 inteiro sobre esse tópico

Gerenciamento de requisitos

- **WI-23** O gerenciamento de mudanças é encontrado principalmente no Capítulo 19; incluindo o processo, a análise de impacto, e medição da volatilidade dos requisitos

Defina um processo de controle de mudanças

Estabeleça em processo para proposta, análise e resolução de mudanças. Gerencie todas as mudanças usando esse processo.

- **RR-23** Uma menção a mudanças nos requistos é feita no Capítulo 14, páginas 269-271

Estabeleça uma comissão de controle de mudanças

Crie um pequeno grupo de pessoas do seu projeto para avaliar propostas de mudança de requisitos, e decidir pela aceitação ou rejeição; além de colocar prioridades na implementação dos requisitos.

Faça uma análise de impacto das mudanças

Analisar o impacto das mudanças ajuda a gerenciá-las de forma mais consciente. Avalie cada proposta de mudança para determinar o seu efeito no projeto. Identifique as tarefas necessárias para implementar a mudança e estime o esforço necessário para tal.

Faça um *baseline* e controle versões de documentos de requisitos

Um *baseline* é um conjunto de requisitos que foram acordados para implementação em uma certa versão do sistema. Mudanças nesse conjunto somente devem ser feitas usando o processo de controle de mudanças já definido. Dê um identificador único para cada versão da especificação dos requisitos.

- **WI-24** Esse tópico é encontrado no Capítulo 18

Mantenha um histórico das mudanças dos requisitos

Escreva a data das mudanças na especificação dos requisitos, as mudanças feitas, por quem e porque as mudanças foram feitas.

- **RR-24** O cartão usado no modelo de especificação, encontrado no Apêndice B, inclui essa informaçao

Rastreie o estado de cada requisito

O estado de um requisito pode ser proposto, aprovado, implementado ou verificado, por exemplo. Guardar esse atributo permite monitoramento do número de requisitos em cada situação.

- **WI-25** Veja como isso funciona no Capítulo 18

Meça volatilidade dos requisitos

Escreva o número de requisitos na *baseline*, e o número de mudanças propostas e aprovadas por semana. Uma alta taxa de mudanças sugere um trabalho mais profundo no estudo do domínio da aplicação, na definição do escopo, ou no levantamento de requisitos.

Use uma ferramenta de gerenciamento de requisitos

O uso de uma ferramenta automatiza várias práticas descritas nesse tutorial. Você pode definir atributos dos requisitos, rastrear seu estado e ligar diversos objetos da especificação, como um requisito e sua implementação.

- **WI-26** O Capítulo 21 traz uma discussão dos benefícios de usar ferramentas de software, e uma lista de ferramentas recomendadas

Crie uma matriz de rastreabilidade de requisitos

Faça uma tabela que liga cada requisito com o design e código que o implementa e os testes que o validam. Também ligue requisitos aos casos de uso e regras de negócio do qual eles foram derivados. Faça isso durante o desenvolvimento do projeto, não no fim.

- **WI-27** O Capítulo 20 ilustra essa técnica
- **RR-25** Uma discussão similar é encontrada no Capítulo 14, páginas 265-269

Gerenciamento de projeto

Selecione um ciclo de vida de desenvolvimento apropriado

Dependendo do nível de certeza ou segurança a respeito dos requisitos, pode ser necessário um ciclo de vida de estrutura modificável, para permitir pequenos incrementos baseados no conhecimento disponível dos requisitos.

Faça planos de projeto baseados nos requisitos

Desenvolva planos e prazos para seu projeto de forma iterativa, conforme o escopo e os requisitos forem ficando detalhados e claros. Com o avanço do desenvolvimento de requisitos, as estimativas ficarão cada vez mais precisas.

- **WI-28** O começo do Capítulo 17 aborda esse tópico

Renegocie compromissos quando os requisitos mudarem

Conforme novos requisitos forem incorporados no projeto, avalie se os prazos e a qualidade ainda podem ser mantidos, dentro dos recursos disponíveis. Renegocie novos compromissos de prazo e qualidade com o cliente; e caso isso falhe, alerte sobre prováveis conseqüencias no projeto.

Documente e gerencie riscos relativos aos requisitos

Identifique e documente riscos relacionados com os requisitos. Crie estratégias para diminuí-los e impedí-los de se tornarem realidade em seu projeto.

- **WI-29** O tópico é tratado no Capítulo 23 do livro
- **RR-26** O Capítulo 13, páginas 247-249, comenta sobre essa prática

Rastreie o esforço gasto com a engenharia dos requisitos

Registre o esforço gasto por sua equipe com o desenvolvimento e gerenciamento de requisitos. Com isso é possível avaliar se o projeto está correndo como o esperado e melhor estimar o esforço em projetos futuros.

- **WI-30** Encontrado no Capítulo 18 do livro
- **RR-27** Em Rob99, está no Capítulo 13, páginas 249-250

Revise lições aprendidas a respeito de requisitos em outros projetos

Faça uma retrospectiva (ou *postmortem*) do seu projeto. O estudo posterior dessas lições sobre requisitos e práticas irão certamente ajudar em projetos futuros.

- **RR-28** O Capítulo 14, páginas 271-275, ilustra como usar essa prática

Fim

Espero que esse tutorial tenha servido para o propósito para o qual foi criado, ilustrando os principais conceitos por trás das boas práticas de Engenharia de Requisitos. Sinta-se na obrigação de consultar as obras mencionadas na bibliografia, se o seu desejo é colocar esses conceitos em prática. A Engenharia de Requisitos também não se encerra nesses livros, então procure sempre novas referências. Principalmente, divirta-se, pois essa disciplina é uma fonte inesgotável de aprendizado.

Bibliografia

[Rob99] S. Robertson, J. Robertson. *Mastering the Requirements Process.* Harlow, England: Addison-Wesley, 1999.

[Wie03] K. E. Wiegers, *Software requirements, 2nd edition.* Redmond, WA: Microsoft Press, 2003.