

Exercícios resolvidos de *Ian Sommerville, Software Engineering – 8th Edition* indicados pelo Prof. Sérgio Guerreiro, resolvidos por Carrajola e Zélia Regina. Atualizações a azul por Victor Freire.

Exercícios Pág.18

1.1 - Fazendo referência aos custos do software indicados na secção 1.1.7, explique porque é apropriado considerar que o software é mais que programas que são executados por os usuários finais de um sistema.

Requisitos > Arquitectura > Desenvolvimento > Implementação > Testes > Implantação > Manutenção/Evolução

O que se verifica é que no processo de distribuição do software, variando consoante o tipo de aplicação, é que as fases posteriores ao desenvolvimento, a validação (integração e testes) e a evolução, têm por vezes um custo mais elevado do que a fase de desenvolvimento. Quando o software desenvolvido é integrado num sistema já existente, a fase de integração e testes é extensa e dispendiosa, atingindo cerca de 50% dos gastos totais do processo de criação do software. Igualmente dispendioso é o processo de evolução depois do software estar implementado e testado. Para uma aplicação com um longo tempo de vida, como sistemas de comando e controle que serão usados durante 10 anos ou mais, os custos de evolução provavelmente chegarão a 3 ou 4 vezes o valor gasto para o desenvolvimento desse software. Sendo assim é correcto dizer-se que o processo de criação de software inclui toda a actividade que o envolve, ou seja, a especificação, o desenvolvimento, a validação e a evolução, incluindo também toda a documentação associada a cada uma dessas fases.

1.2 – Quais são as diferenças entre o desenvolvimento de um produto de software genérico e um desenvolvimento de um produto de software personalizado.

Software genérico – Quem produz o software controla a especificação, feitos para o mercado geral.

Software à medida – Quem compra o software controla a especificação, feitos para um cliente específico.

1.7 - À parte dos desafios de heterogeneidade, entrega rápida e confiança, indique outros problemas e desafios que a engenharia de software provavelmente enfrentará no século 21.

- Performance do software – (utilização de ferramentas case cria código não otimizado e menos eficiente, novos algoritmos e linguagens mais eficazes para criação de software)
- Escalabilidade – modelos mais eficazes na escalabilidade e manutenção de projectos de software cada vez mais complexos e melhor reutilização de código. Evolução dos métodos de programação. Ex.: programação estruturada, programação orientada a objectos, ...
- Segurança
- Ergonomia do software – software cada vez mais acessível a todos os utilizadores (Ex.: Utilizadores com deficiências)
- Produção de software com linguagem natural – acelera o processo de criação de software possibilitando um nível máximo de abstracção.
- Melhores e mais fiáveis agentes inteligentes para ajuda no processo de criação de software.
- Software amigo do ambiente (performance e ética ambiental)
- Certificação dos engenheiros de software.
- Custos mais baixos na produção de software, conjunto das medidas acima indicadas

1.8 - Discuta se os engenheiros profissionais devem ser certificados do mesmo modo que médicos ou advogados

Abordagem concordante: Responsabilidade e certificação em áreas de conhecimento onde estão subjacentes riscos elevados, em vidas humanas e em prejuízos materiais, da mesma forma que as ordens regulam outras áreas (código deontológico) onde esse mesmo risco existe: medicina, direito, farmácia, engenharia, etc.

Abordagem discordante: É inviável limitar a criação de software. Custo mais elevado do software. Dificuldade a especificar qual software é de risco elevado e qual não é, por exemplo, software de uma empresa afecta os "stakeholders" mas não dependem vidas deste directamente... qual o grau de risco?, em comparação por exemplo no caso relativamente às drogas farmacêuticas existem produtos de livre utilização. Onde se enquadra o software "open source"?

2.4 - Explique por que é importante produzir uma descrição completa de uma arquitectura de sistema numa etapa inicial do processo de especificação.

Principalmente facilita o processo de gestão do projecto nos seguintes pontos:

- Viabilidade do projecto e riscos em termos financeiros, tecnológicos, de tempo e de recursos humanos e materiais
- Optimizações na gestão do plano de distribuição de recursos humanos e materiais no processo da criação do software
- Ajuda na gestão de "milestones" e plano de trabalhos (tempo, recursos e custos)
- Sendo feito ajuda a clarificar e avaliar o grau de importância de cada requisito
- Melhor documentação do projecto, vital para a continuação do trabalho em caso de mudança de recursos humanos e testes
- Melhora da qualidade do software em termos gerais
- Ajuda a especificar as condições do contrato com cliente

2.8 - Explique por que os sistemas de legado (legacy system) podem ser críticos à operação de um negócio.

Sistema de legado: Sistema sócio-técnico desenvolvido no passado, muitas vezes com tecnologia já obsoleta. Gere habitualmente sistemas críticos para a actividade. Engloba o processo de negócio, software aplicacional, software de apoio e hardware, portanto muitas vezes a actividade de negócio não pode ser efectuada sem ele.

Ex.: Software de Secretaria Escolar

Os sistemas Legacy proporcionam serviços essenciais ao negócio mas, porque incluem processos de negócio, software aplicacional, software de apoio e sistemas de hardware, podem ser críticos no funcionamento de um negócio por ser demasiado arriscado substituí-los, visto que as políticas e procedimentos organizacionais dependem destes sistemas.

2.9 – Explique porque é que os sistemas herdados (legacy systems) podem causar dificuldades para as companhias que desejam reorganizar os seus processos de negócio.

- Custos de desenhar um novo sistema são proibitivos por causa do seu tamanho, por ser monolítico ou demasiado complexo
- O sistema requiere uma disponibilidade de 100%, não pode ser retirado de serviço, e o custo de desenhar um novo sistema com uma taxa de disponibilidade é elevado
- O sistema funciona de forma pouco clara. Essa situação pode ocorrer quando os arquitectos de sistema deixaram a organização e o sistema não foi bem documentado ou a documentação foi perdida
- Os dados do sistema são de elevado grau de confidencialidade (Ex.: Dados Militares)
- O custo de aprender a utilizar o novo software é elevado em termos de tempo e custo (Recursos Humanos)
- Riscos na migração dos dados

2.10 – Quais são os argumentos a favor e contra para considerar que a Engenharia de Sistemas é uma profissão no seu próprio direito como engenharia eléctrica ou engenharia de software.

A favor:

A Engenharia de Sistemas envolve as seguintes disciplinas: engenharia de software; engenharia electrónica; engenharia mecânica; arquitectura de interface do utilizador; arquitectura de sistemas; engenharia eléctrica; engenharia civil; engenharia de estruturas. Assim, verifica-se que Engenharia de Sistemas é uma profissão que recorre a variados conhecimentos de outras engenharias.

Competências diferentes, um engenheiro de software não tem todas as competências de um engenheiro de sistemas e vice-versa, porque apesar de ser englobado, existem aspectos técnicos especializados, senão abstractamente sem especialização só haveria engenheiros em ciências naturais.

Contra:

A engenharia de sistemas é uma actividade interdisciplinar que envolve equipas com diferentes formações técnicas, por causa do amplo conhecimento exigido para considerar todas as implicações das decisões referentes a projectos de sistemas.

Os engenheiros de sistemas não se ocupam apenas com o software, mas com as interacções de software, hardware e sistema com os utilizadores e seu ambiente.

Eles devem pensar sobre os serviços que o sistema fornece, as restrições, dentro dos quais o sistema deve ser construído e operado e as interações do sistema com o seu ambiente.

Os engenheiros de software necessitam de uma compreensão sobre a engenharia de sistemas, porque os problemas de ES, frequentemente são o resultado da engenharia de sistemas.

2.11 – Suponha que é um engenheiro relacionado com o desenvolvimento de um sistema financeiro. Durante a instalação descobre que o sistema vai reduzir um número significativo de pessoas. As pessoas envolvidas negam-lhe acesso a informação essencial para completar a instalação do sistema. Até onde deveria, como engenheiro de sistemas, ficar envolvido nisto? É responsabilidade profissional sua completar a instalação como estipula o contrato? Devia simplesmente abandonar o trabalho até que a organização tenha resolvido/classificado o problema?

Deveria manter como objectivo a conclusão do trabalho contratado, respeitando todos os âmbitos legais e éticos no seu desenvolvimento e utilizando os canais próprios e bom senso para arbitrar cada situação.

Exercícios Pág.91

4.2 – Explique porque é que programas que foram desenvolvidos usando desenvolvimento evolucionário tendem a ser difíceis de manter.

Requerem constantes mudanças aos requisitos, actualizações ao design, contínuos processos de desenvolvimento e testes, o que leva a custos elevados na sua manutenção. Além disso a evolução pode ser complicada de acompanhar para os gestores e utilizadores do software da organização.

4.5 - Explique porque é importante fazer a distinção do desenvolvimento dos requisitos do usuário e o desenvolvimento dos requisitos do sistema no processo de engenharia de requisitos.

Os requisitos de usuário para um sistema devem descrever os requisitos funcionais e não funcionais de modo compreensível pelos usuários do sistema que não tem conhecimentos técnicos detalhados. Eles devem especificar somente o comportamento externo do sistema. Evitando tanto quanto possível as características de sistema.

Os requisitos de sistema são as descrições mais detalhadas dos requisitos do usuário. Eles podem servir como base para um contrato destinado à implementação do sistema e portanto devem ser uma especificação completa e consistente de todo o sistema. Eles são utilizados pelos engenheiros de software como ponto de partida para o projecto do sistema.

4.6 – Descreva as principais actividades no processo de design de software e os outputs destas actividades. Usando um diagrama, mostre as relações entre os outputs destas actividades

As principais actividades no processo de desenho de software são:

1. *Projecto de arquitectura* – Os subsistemas que constituem o sistema e suas relações são identificadas e documentadas;
2. *Especificação abstracta* – Para cada subsistema, é produzida uma especificação abstracta das suas funções e das restrições dentro das quais devem operar;
3. *Projecto de interface* – Para cada subsistema, é projectada e documentada uma interface com outros subsistemas. Essa especificação de interface não pode apresentar ambiguidade, uma vez que ela permite que o subsistema seja utilizado sem conhecimentos de operação do subsistema. Os métodos de especificação formal, podem ser utilizados neste estágio;
4. *Projecto de componentes* – As funções são alocadas a diferentes componentes e as interfaces desses componentes são projectadas;
5. *Projecto de estrutura de dados* – As estruturas de dados utilizadas na implementação de sistemas são projectadas em detalhe e especificadas.
6. *Projecto de algoritmos* – Os algoritmos utilizados para proporcionar serviços são projectados detalhadamente e especificados.

Exercícios Pág.112

5.1 – Explique porque é que a intangibilidade dos sistemas de software traz problemas para a gestão de projectos de software.

Uma boa gestão do projecto de software é essencial para que os projectos de software sejam desenvolvidos dentro do prazo e do orçamento.

Os gestores de projecto de software não podem quantificar o progresso, eles dependem de outras pessoas para produzir a documentação necessária, a fim de saberem o estado de desenvolvimento. Se essa documentação é insuficiente ou inexistente o gestor não tem elementos para decidir.

5.2 – Explique porque é que os melhores programadores não fazem sempre os melhores gestores de software. Você pode ter como base útil a lista de actividades de gestão dadas na secção 5.1.

Elaboração de propostas ...descreve os objectivos do projecto e como ele será realizado...

- Planeamento e programação de projectos ...se preocupam em identificar as actividades, os marcos, e os documentos a serem produzidos em um projecto.

- Custo do projecto....custos requeridos para realizar um projecto

- Monitorização e revisões de projectos....o monitoriamento de projecto é uma actividade continua. o gestor deve manter o acompanhamento do andamento do projecto e comparar os progressos e custos reais com os que foram planeados....

- Selecção e avaliação do pessoal...o ideal é que uma equipa hábil e com experiência adequada esteja disponível...mas isso nem sempre é possível porque:

1. O orçamento pode não chegar para contratar uma equipa bem paga
2. A equipa apropriada pode não estar disponível por exemplo por estar alocada a outro projecto
3. A organização pode querer desenvolver as habilidades de seus funcionários.

- Elaboração de relatórios e apresentações....tanto para a organização do cliente como para as organizações dos fornecedores

5.3 – Explique porque é que o processo de planeamento do projecto é iterativo e porque é que um plano deve ser continuamente revisto durante um projecto de software.

- Nova informação fica disponível
- O objectivo do software (negócio) pode alterar-se
- O projecto pode sofrer atrasos
- Pode haver recursos que deixam de estar disponíveis
- Custos mais elevados que o previsto

5.4 – Resumidamente explique o propósito/objectivo de cada uma das secções num plano de projecto de software.

O plano do projecto define os recursos disponíveis para o projecto, a estrutura analítica do trabalho e uma programação (calendário) para realizar o trabalho.

Contudo, a maioria dos planos deve incluir as seguintes secções:

Introdução - descreve com brevidade os objectivos do projecto e define as restrições (orçamento , prazo) que afectam a gestão do projecto.

Organização do projecto – descreve o modo como a equipe é organizada, as pessoas envolvidas e os seus papéis na equipe.

Análise de riscos – descreve possíveis riscos do projecto, a probabilidade de surgir esses riscos e as estratégias propostas para a redução deles.

Requisitos necessários de hardware e de software – descreve o hardware e o software de apoio exigidos para realizar o desenvolvimento. Se o hardware tiver de ser comprado deverão ser incluídos os prazos de entrega e as estimativas de preço.

Divisão do trabalho – descreve a divisão do trabalho em actividades e identifica os marcos (milestones) e os produtos a serem entregues com cada actividade (deliverables).

Calendarização de projecto – descreve as dependências entre actividades, o tempo estimado requerido para cada marco e a alocação de pessoas nas actividades.

Mecanismos de monitoramento e de elaboração de relatórios – Descreve os relatórios de gestão que devem ser produzidos, quando eles devem ser produzidos e quais os mecanismos de monitorização que são utilizados.

5.5 – Qual é a diferença fundamental entre um “milestone” e um “deliverable”?

- **Milestone**

Um “milestone” é interno e só entregue ao gestor do projecto.

“Milestone” é para o gestor acompanhar o projecto.

- **Deliverable**

Um “deliverable” é um “milestone” de uma fase importante no desenvolvimento do projecto que é entregue ao cliente.

“Deliverable” é para o cliente acompanhar o projecto.

5.6 – A figura 5.15 mostra um conjunto de actividades, durações e dependências. Desenha uma rede de actividades e um gráfico de barras que mostre a programação do projecto.

5.7 – A figura 5.5 assinala a duração das tarefas para as actividades do projecto de software. Suponha que há uma contrariedade imprevista em lugar de requerer 10 dias, a tarefa T5 requer 40 dias. Fazer uma revisão da rede de actividades resultante, destacando o novo caminho crítico. Desenhe um novo gráfico de barras que mostre como se poderia reorganizar o projecto.

5.9 – Para além dos riscos que se mostram na figura 5.11, indique outros seis possíveis riscos nos projectos de software.

No livro:

- **Tecnologia**
Riscos que derivam do software ou hardware usados no desenvolvimento do sistema.
Ex.: Defeitos no hardware, insuficiências do software, recursos indisponíveis.
- **Pessoas**
Riscos associados às pessoas na equipa de desenvolvimento.
Ex.: Doenças, Inexistência de especialistas disponíveis.
- **Organizacionais**
Riscos que derivam do ambiente da organização onde o software está a ser desenvolvido.
Ex.: Reestruturações, falta de fundos.
- **Ferramentas**
Riscos que derivam de ferramentas CASE e de outro software usado no desenvolvimento do software.
Ex.: Código gerado é ineficiente, impossibilidade de integração das ferramentas.
- **Requisitos**
Riscos que derivam de mudanças dos requisitos por parte do cliente e no processo de gestão dessa mudança de requisitos.
Ex.: As modificações inseridas pelo cliente geram um trabalho avultado, os clientes não compreendem o impacto das mudanças de requisitos, o cliente não deu os requisitos certos.

- **Estimativas**
Riscos que derivam das estimativas feitas pela gestão quanto às características do sistema e dos recursos necessários para o construir.
Ex.: Estimativas erradas quanto ao tempo de desenvolvimento, à extensão de defeitos, ao tamanho do software.

Adicionais:

- **Acidentes**
Riscos que derivam de um evento acidental, interno ou externo.
Ex.: Perda de trabalho acidentalmente, incêndios, roubos.
- **Comunicação**
Riscos que derivam de transmissão de informação não clara, errada ou em falta entre a equipa de desenvolvimento.
Ex.: Equipa de várias nacionalidades, canais de comunicação deficientes, equipas de em locais diferentes.
- **Motivação**
Riscos que derivam da motivação da equipa ou do cliente para a realização do projecto
Ex.: Indisponibilidade para reuniões, local de trabalho desconfortável, salários baixos.
- **Expectativas**
Riscos que derivam de uma expectativa frustrada dos utilizadores do software.
Ex.: Interface não "user friendly", tarefas dificultadas no software, ergonomia.
- **Conflitos**
Riscos que derivam de conflitos entre os membros da equipa e/ou com o cliente.
Ex.:
- **Segurança**
Riscos que derivam de falhas de segurança no sistema.
Ex.: Intrusões, vírus.

5.11 – Seu chefe solicitou que entregue um software num tempo que só pode ser possível cumprir perguntando à equipa do projecto se deseja trabalhar horas extras não pagas. Todos os membros da equipa têm filhos pequenos. Comente se deveria aceitar esta exigência do seu chefe ou se você deveria persuadir a equipa para dar o seu tempo à organização mais que as suas famílias. Que factores poderiam ser significativos na decisão?

- **Importância na data de entrega do software**
Perda do contracto se o software não for entregue ou um simples "milestone".
- **Consequências para os trabalhadores**
Perda de emprego, prémios.
- **Motivação**
Nível de motivação da equipa para terminar o projecto.

5.12 – Como programador, se lhe oferecessem uma promoção como gestor de projecto, mas você sente que pode ter uma contribuição melhor num papel técnico do que num papel administrativo. Comente se deveria aceitar esta promoção.

Factores de decisão:

- **Competências**
- **Desafio**
- **Vantagens para a organização**
- **Ordenado**

Exercícios Pág.141

6.1 – Identifique resumidamente e descreva 4 tipos de requisitos que se podem definir para um sistema informático.

- **Funcionais**
Declarações de funções que o sistema deve fornecer, como o sistema deve reagir a entradas específicas e como se deve comportar em determinadas situações. Em alguns casos os requisitos funcionais podem também explicitamente dizer o que o sistema não deve fazer.
Ex.: Consulta de dados de histórico, diferentes graus de privilégios de administrador.
- **Não funcionais**
Restrições sobre os serviços ou as funções oferecidas pelo sistema. Entre eles destacam-se restrições de tempo, processo de desenvolvimento (produto, organizacionais, externos).

Ex.: Tempo de resposta de acesso aos dados inferior a 5 segundos, disponibilidade a 100%.

- **Requisitos dos utilizadores**

Declarações em linguagem natural e também em diagramas sobre as funções que o sistema deve fornecer e as restrições sob as quais deve operar.

Ex.: Interface simplificada com uma curva de aprendizagem rápida, linguagem do programa em português e inglês.

- **Requisitos do sistema**

Descrições mais detalhadas dos requisitos de usuário. Eles podem servir de base a um contrato destinado à implementação do sistema e, portanto deve ser uma especificação completa e consistente do sistema.

Ex.: Sistemas redundantes com tolerância a falhas em RAID 5, utilização de código aberto grátis PHP e MySQL em ambiente Linux.

6.2 – Discuta o problema de usar linguagem natural para definir os requisitos do usuário e do sistema e mostre, usando pequenos exemplos, porque é que a estruturação da linguagem natural em formulários pode ajudar a evitar algumas destas dificuldades.

Tipos de Notações para especificação requisitos:

- **Linguagem natural estruturada**

Definição de formulários standard ou templates para exprimir a especificação dos requisitos.

- **Linguagens de descrição de design**

Utiliza uma espécie de linguagem de programação mas com conceitos abstractos. Não é muito utilizada, a não ser para especificar interfaces.

- **Notações gráficas**

Uma linguagem gráfica com anotações em texto. É utilizada para definir requisitos funcionais para o sistema. São exemplos comuns as descrições use-case e diagramas de sequência.

- **Especificações matemáticas**

Notações baseadas em conceitos matemáticos como máquinas de estado finitas. Esta especificação reduz os argumentos com o cliente sobre a funcionalidade de sistema. Dificulta no entanto a compreensão por parte do cliente, podendo este recusar-se a aceitar essa especificação como contracto do sistema.

Dificuldades:

- **Ambiguidade da linguagem natural**
A interpretação da linguagem natural depende de quem a lê ou a escreve, o que pode levar a mal entendidos no significado dos requisitos.
- **Excesso de flexibilidade**
Um mesmo requisito pode ser identificado de diversas maneiras diferentes, levando a confusão se os requisitos são os mesmos ou diferentes.
- **Dificuldade a modular**
Sendo a linguagem difícil de modular, é difícil relacionar os requisitos entre si de forma a verificar as consequências de uma mudança.

Ex.:

- Dicionário de requisitos, em todos os sítios na documentação e no programa determinado requisito é referenciado da mesma forma.
- Template ou formulário para preenchimento de cada requisito (Descrição da função, inputs/output, origem/destino, pré/pós condições).

6.3 - Descubra ambiguidades ou omissões na seguinte afirmação de requisitos de uma parte de um sistema de emissão de bilhetes.

Um sistema automático de emissão de bilhetes vende bilhetes de comboio. Os usuários seleccionam o seu destino e introduzem um cartão de crédito e um número de identificação pessoal. O bilhete de comboio é emitido e a conta deles de cartão de crédito é cobrada. Quando o usuário pressiona o botão de início, é mostrado um menu que mostra os possíveis destinos, junto com uma mensagem para o usuário que lhe indica para seleccionar um destino. Uma vez que se seleccionou um destino, pede aos usuários que introduzam o cartão de crédito. A sua validade é verificada e é pedido ao usuário para introduzir um identificador pessoal. Quando a transacção de crédito for validada, o bilhete é emitido.

Ambiguidades

- **Só cartões de crédito ou também de débito? Bancário ou interno?**
- **Nº de identificação pessoal = identificador pessoal?**
- **(Conta de cartão de crédito cobrada = transacção de crédito validada?)**
- **O que faz 1º? Depois da transacção é que emite o bilhete, não antes!**

Omissões

- **Tipos de bilhetes?**
- **Tipos de comboios?**
- **Quais os destinos?**
- **Número de identificação pessoal de quê? Do cartão?**
- **Ecrã inicial (Início) aparece por defeito é a escolha de destino? Onde fica o botão de início?**
- **Validar antes de inserir um identificador pessoal?**
- **Não descreve quando e como é solicitado o código pessoal ao utilizador.**
- **Não descreve como o sistema deve reagir a um cartão ou código pessoal não válido.**
- **Não define como é devolvido o cartão ao utilizador.**

6.4 – Volte a escrever a descrição anterior utilizando a aproximação estruturada descrita neste capítulo. Resolva de forma apropriada as ambiguidades identificadas.

Pretende-se o desenvolvimento de um sistema automático de venda de bilhetes de comboio.

Para iniciar a utilização do sistema o utilizador deverá premir o botão início, activando o menu de destinos possíveis associado a uma mensagem que lhe indicará que deve seleccionar o destino pretendido.

Após a selecção do destino pretendido, o sistema pede ao utilizador para inserir o cartão de crédito na ranhura existente para o efeito mediante a apresentação de uma mensagem no ecrã. De seguida, o sistema solicitará a introdução do código pessoal mediante a apresentação de uma mensagem. Após a introdução do código pessoal pelo utilizador, recorrendo ao teclado existente no dispositivo, o sistema comprovará a validade do cartão. Se o cartão ou código pessoal não for válido, o sistema apresentará a mensagem de "Cartão não válido" e expelirá o cartão. Se o cartão e código pessoal estiverem correctos, o sistema iniciará a transacção. Terminada a transacção, o sistema devolverá o cartão ao utilizador apresentando a seguinte mensagem: "Retire o cartão por favor".

Após o utilizador retirar o cartão do sistema, será impresso o bilhete e expedido pela respectiva ranhura, apresentando uma mensagem de convite ao utilizador para retirar o bilhete.

Retirado o bilhete, termina a operação e o sistema volta ao estado de início, aguardando que novo utilizador pressione no respectivo botão.

6.7 – Descreva 4 tipos de requerimentos não funcionais que podem existir em um sistema. De exemplos de cada um destes tipos de requerimentos.

- **Requisitos de produtos**

Requisitos que especificam o comportamento do produto.

Ex.: 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.

- **Requisitos organizacionais**

procedentes de políticas e procedimentos nas organizações do cliente e do desenvolvedor.

Ex.: Os padrões de processo que devem ser utilizados, os requisitos de implementação, como a linguagem de programação ou o método de projecto utilizado, e os requisitos de fornecimento, que especificam quando o produto e seus documentos devem ser entregues.

- **Requisitos externos**

Abrange todos os requisitos procedentes de factores externos ao sistema e a seu processo de desenvolvimento.

Ex.: Os requisitos de interoperabilidade, que definem como o sistema interage com sistemas noutras organizações, os requisitos legais, que devem ser seguidos para assegurar que o sistema opera de acordo com a lei, e os requisitos éticos (os requisitos éticos são definidos num sistema para garantir que este será aceitável para os seus utilizadores e o público em geral).

6.10 – Você aceitou um trabalho com um utilizador de software que contratou anteriormente o seu antigo empregador para desenvolver um sistema. Você descobre que a interpretação de sua empresa dos requisitos é diferente da interpretação feita pelo seu antigo empregador. Discuta o que deve fazer nesta situação. Você sabe que os custos para o seu actual empregador vão aumentar se as ambiguidades não forem resolvidas. Tem também uma responsabilidade de confidencialidade para com seu empregador anterior.

Tentava resolver as ambiguidades sem revelar informação confidencial.

Exercícios Pág.167

7.1 – Sugira quem podem ser os “stakeholders” num sistema de registo de estudantes numa universidade. Explique por que é quase sempre inevitável que os requisitos de diferentes “stakeholders” sejam, de alguma maneira, conflituosos.

- Os “stakeholders” não sabem o que querem do sistema a não ser num carácter geral. Podem ter dificuldades a verbalizar o que querem ou fazem exigências irrealistas pois não têm noção dos custos.
- Cada “stakeholder” expressa requisitos baseados no seu conhecimento específico do negócio, tendo estes que ser integrados no todo.
- “Stakeholders” diferentes têm requisitos diferentes que expressam de maneira diferente.
- Factores políticos influenciam o design do sistema, por exemplo, gestores que requerem algo específico para aumentar a sua influência na organização.
- O ambiente onde o sistema se integra é dinâmico, os requisitos podem variar durante a sua avaliação e entrada de novos “stakeholders” que não foram consultados.

7.6 – Discuta o exemplo de um tipo de sistema em que os factores sociais e políticos podem influenciar fortemente os requisitos do sistema. Explique porque é que esses factores são importantes no seu exemplo.

Mudanças legislativas originam requisitos novos.

Ex: mudança do IVA

Escutas autorizadas na Internet.

Projectos internacionais – questões linguísticas, dicionários e adaptar a cada país.

Políticas da empresa – barrar sites, acessos, ver produções dos empregados.

7.7 – Quem deve estar envolvido em uma revisão de requisitos? Desenhe um modelo de processo mostrando como uma revisão de requisitos pode ser organizada.

Os requisitos devem ser analisados sistematicamente pela equipa de revisores.

É um processo manual, que envolve muitos leitores, tanto do pessoal do cliente como do fornecedor, que verifica o documento de requisitos a fim de detectar anomalias ou omissões. (pode e deve envolver todos os stakeholders do sistema)

Como alternativa, esse processo pode ser organizado em maior escala, com muitos participantes envolvidos na verificação de diferentes partes do documento.

As revisões de requisitos podem ser formais ou informais.

As informais simplesmente envolvem os fornecedores/desenvolvedores que discutem os requisitos com tantos stakeholders qt possível .

Na formal a equipe de desenvolvimento deve conduzir o cliente pelos requisitos do sistema. Explicando as implicações de cada um.

7.8 – Porque é que as matrizes de facilidade de rastreio se tornam difíceis de gerir quando existem muitos requisitos de sistema? Projecte um mecanismo de estruturação de requisitos, com base em pontos de vista, que possa ajudar a reduzir a escala desse problema.

Existem muitas relações entre requisitos e outros requisitos e entre os requisitos e o design

Quando são propostas modificações é preciso verificar o impacto dessas mudanças sobre outros requisitos e o design do projecto.

A facilidade de rastreio é uma propriedade geral de uma especificação de requisito e reflecte a facilidade de se encontrar requisitos relacionados.

Existem 3 tipos de informações sobre a facilidade de rastreio , que podem ser mantidas.

- 1- Informações sobre a facilidade de rastreio da origem vinculam os requisitos aos stakeholders que propuseram esses requisitos.
- 2- Informações sobre a facilidade de rastreio de requisitos vinculam requisitos dependentes dentro de seu respectivo documento.
- 3- Informações sobre a facilidade de rastreio de design vinculam os requisitos aos módulos de design em que esses requisitos são implementados.

As informações sobre a facilidade de rastreio são, frequentemente representadas com o uso de matrizes de facilidade de rastreio. Estas relacionam os requisitos aos stakeholders , os requisitos entre si ou aos módulos de design .

As matrizes de facilidade de rastreio podem ser utilizadas quando um pequeno numero de requisitos precisa de ser gerido, mas elas tornam-se muito difíceis de serem manuseadas e são de manutenção dispendiosa em grandes sistemas com muitos requisitos.

Para esses sistemas temos de obter as informações da facilidade de rastreio em base de dados de requisitos , em que cada requisito é explicita/ vinculado a requisitos relacionados. O impacto das mudanças pode ser avaliado pelo uso dos recursos de visualização das bases de dados. Como alternativa, pode ser possível gerar as matrizes de facilidade de rastreio automática/.

A gestão de requisitos precisa de algum apoio automatizado e as ferramentas CASE devem ser usadas em :

- 1- armazenaneto de requisitos
- 2- gestão de mudanças
- 3- gestão de facilidade de rastreio

mecanismo de estruturação de requisitos, com base em pontos de vista

mesmo para um sistema relativa/ simples existem muitos pontos de vista diferentes que devem ser considerados. Os diferentes pontos de vista a respeito de um problema vem o problema de modos diferentes. Contudo, suas perspectivas não são inteiramente independentes, tem alguma duplicidade, e apresentam requisitos comuns.

A abordagem orientada aos pontos de vista e usada para estruturar e organizar o processo de levantamento de requisitos.

Os estágios do VORD (viewpoint -oriented requirements definition)

- 1- Identificação dos pontos de vista- descobrir os pontos de vista e respectivos serviços
- 2- Estruturação de pontos de vista- agrupar pontos de vista relacionados segundo uma hierarquia
- 3- Documentação de ponto de vista – refinar a descrição
- 4- Maneamentos de sistema de ponto de vista – envolve identificar objectos em um design orientado a objectos, utilizando informações de serviço que estão encapsuladas nos pontos de vista

Exercícios Pág.214

9.2 – Num sistema de bomba de insulina, o usuário tem de modificar a agulha e a provisão de insulina regularmente e também pode modificar a dose única máxima e o máximo diário que pode ser administrado. Sugira três erros de usuários que poderiam ocorrer e propor exigências de segurança que evitariam esses erros que resultam em um acidente

9.3- Um sistema de software seguro e crítico para o tratamento de doentes com cancro tem dois componentes principais:

A máquina de radioterapia que administra doses controladas de radiação aos sítios do tumor. Esta máquina é controlada por um sistema de software embebido.

Uma base de dados de tratamentos que inclui os detalhes do tratamento dado a cada paciente. Os requerimentos do tratamento são introduzidos nesta base de dados e automaticamente se descarregam na máquina de radioterapia.

Identifique três contingências que podem surgir neste sistema. Para cada contingência sugira um requerimento defensivo que reduza a probabilidade de que estas contingências provoquem um acidente. Explique porque é que a defesa sugerida por você é provável que reduza o risco associado a contingência.

Anexo:

Software development process
Activities and steps
Requirements Architecture Design Implementation Testing Deployment
Models
Agile Cleanroom Iterative RAD RUP Spiral Waterfall XP Scrum
Supporting disciplines
Configuration management Documentation Software quality assurance (SQA) Project management User experience design

Some software development methods:

[Waterfall model](#)

[Spiral model](#)

[Model driven development](#)

[User experience](#)

[Top-down and bottom-up design](#)

[Chaos model](#)

[Evolutionary prototyping](#)

[Prototyping](#)

[ICONIX Process](#) (UML-based object modeling with use cases)

[Unified Process](#)

[V-model](#)

[Extreme Programming](#)

[Software Development Rhythms](#)

[Incremental funding methodology](#)