Especificação de requisitos

Baseado em Arndt von Staa

Especificação

- Objetivo dessa aula
  - Apresentar especificações de requisitos enfatizando o seu uso ao especificar módulos e funções
  - Estabelecer uma forma de incorporar especificações de boa qualidade ao código dos módulos.

Sumário

- Retrabalho inútil
- Por que especificar?
- O que é uma especificação?
- Evolução do nível de abstração de especificações
- Classes de especificações
- Exemplos
- Critérios de qualidade de especificações
- Especificações no código

Motivação

- Uma das principais causas do excessivo tempo gasto (custo) ao desenvolver software é o retrabalho inútil
  - em [Boehm, B.W.; Basili, V.R.; "Software Defect Reduction Top 10 List"; *IEEE Computer* 34(1); Los Alamitos, CA: IEEE Computer Society; 2001; pags 135-137] é mencionado que cerca de 40% a 50% do esforço de desenvolvimento é gasto em retrabalho inútil

O que é retrabalho inútil?

- Exemplos:
  - desenvolver algo e descobrir que não era isso que se queria ou que se precisava
    - falta especificação
  - desenvolver algo e descobrir que está com de defeitos, alguns deles de difícil diagnóstico e depuração
    - falta de disciplina
    - falta de conhecimento de como raciocinar sobre programas, módulos e código
  - trabalhar sem foco
    - falta de método de trabalho
  - perfeccionismo patológico
    - melhorar, melhorar e melhorar mais ainda algo que já está satisfatório
  - **...**

Como eliminar causas de retrabalho inútil?

- Vamos nos fixar na causa de retrabalho inútil:
  - desenvolver algo e descobrir que não era isso que se queria ou que se precisava
- O que fazer para reduzir ou evitar de vez esse risco?
  - produzir uma boa especificação do que se deseja que seja feito
    - tem autores que afirmam que cerca de 70% dos defeitos encontrados em software após a entrega devem-se a especificações erradas ou ausentes
  - produzir uma arquitetura – organização da solução – adequada ao problema a resolver
    - modelar o problema a resolver ➔ modelagem conceitual
    - modelar a solução ➔ modelagem física
  - especificar detalhes da implementação
    - especificação de funções, assertivas

Por que especificar?

- É óbvio que antes de começar a escrever código precisa-se saber o que este código deve fazer
  - requisitos funcionais
- Também é óbvio que, existindo condicionantes ou restrições, estas devem ser explicitadas antes
  - requisitos não funcionais
    - usualmente chamados de requisitos de qualidade
      - desempenho
      - capacidade
      - segurança
      - ...
  - requisitos inversos (o que o software não deve fazer)
    - exemplos
      - o módulo não deve emitir mensagens
      - o módulo jamais deverá cancelar a execução

Um exemplo simplório

- Escreva a função raiz quadrada
  - dá para começar sem nenhuma informação a mais?
- Veja algumas perguntas que podem surgir
  - raiz quadrada de que?
    - inteiros, reais, complexos?
  - qual é a precisão requerida?
    - float, double, múltipla?
    - quantos algarismos significativos?
  - existe alguma exigência de desempenho?
    - tempo de resposta versus precisão
  - deve-se verificar se o argumento fornecido é válido?
    - x >= 0. ?
    - como responder se não for válido?
      - condição de retorno, cancelamento, exceção (C++, Java ...)

Problema da especificação

- Nem sempre sabemos exatamente o que desejamos
  - precisamos ver para saber se está bom ou não
    - IKWIWWISI - *I'll know what I want when I see it*
    - IKIWISI - *I'll Know It When I See It*
  - exemplo: interface humano computador
  - exemplo: abrangência da solução
    - ao por em uso o programa o usuário descobre que o programa poderia fazer mais e/ou fazê-lo de forma diferente
  - exemplo: produto inovador
- Soluções para reduzir o conseqüente retrabalho inútil
  - desenvolvimento incremental
    - pequenos incrementos servem para avaliar se a solução está no caminho certo
  - pequenos experimentos ou protótipos

O que é uma especificação?

- Especificação é um documento ou fragmento que:
  - Determina o que deve ser feito, sem dizer como fazê-lo
  - Determina por quê (ou para quê) deve ser feito
  - Determina os resultados (artefatos) a serem entregues
  - Estabelece os critérios de aceitação dos artefatos resultado
    - a qualidade desejada
      - requisitos não funcionais
      - requisitos inversos
    - teste a ser satisfeito

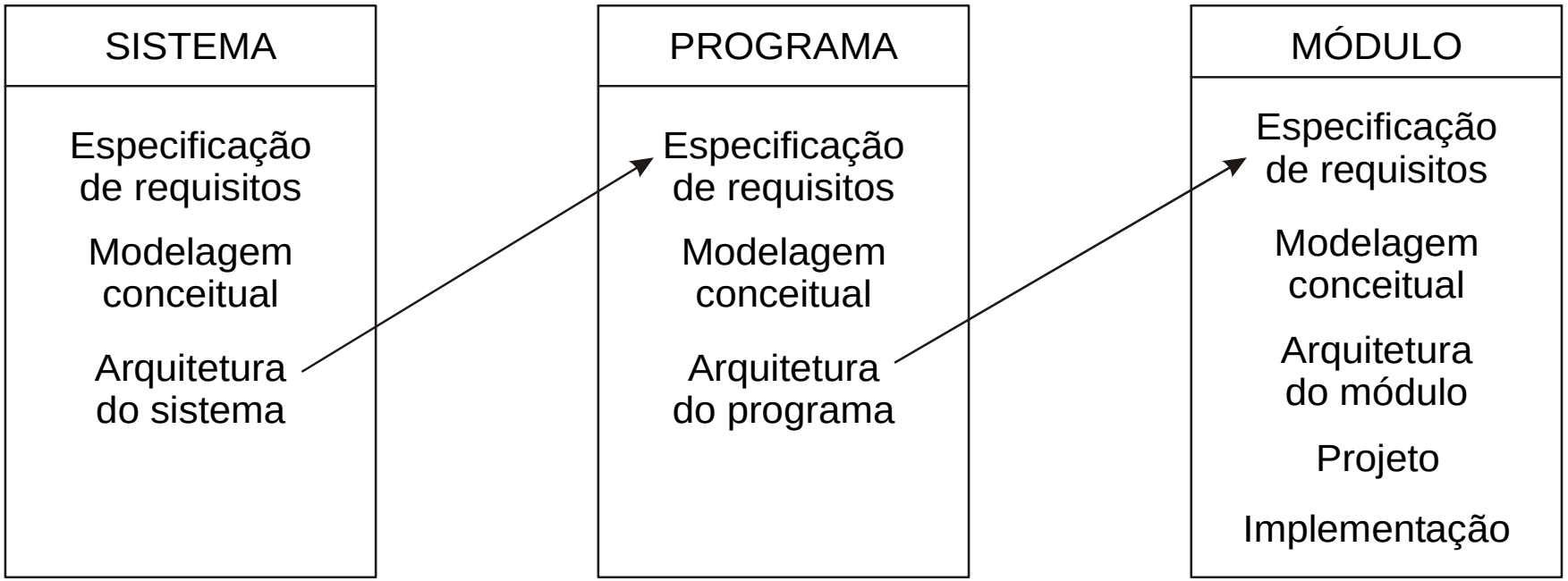
O que é uma especificação?

- Especificações de um artefato são veículos de comunicação entre as diversas pessoas interessadas (*stakeholders*) neste artefato
  - desenvolvedores do artefato
  - mantenedores do artefato
  - desenvolvedores cliente que pretendem localizar e utilizar artefatos servidores já existentes (reuso)
    - componentes
    - bibliotecas
  - outros artefatos
  - usuários que utilizarão um programa contendo o artefato
  - as organizações adquirentes de bibliotecas de artefatos
  - os gerentes de desenvolvimento e manutenção

O que é uma especificação?

- Uma especificação é composta por vários itens de especificação
- Cada item de especificação deve definir ou descrever um aspecto bem delimitado, exemplos
  - uma característica de uma função, exemplos
    - a função calcula a raiz quadrada de x
    - a precisão do resultado é de 5 dígitos decimais ou melhor
  - uma característica de um parâmetro de uma função
    - x deve ser maior ou igual a zero
    - se x < 0, o resultado será imprevisível
  - um diagrama ilustrando ou modelando uma estrutura de dados
  - um texto descrevendo um dos itens de um diagrama

Especificações evoluem em grau de detalhe



Terminologia

- Requisitos: são funções, condições, atributos, propriedades e características a serem disponibilizadas ou satisfeitas pelo artefato
- Hipóteses: são funções, condições, atributos, propriedades ou características assumidas como satisfeitas externamente à equipe de desenvolvimento
  - o usuário logado tem direitos de ativar esta função
    - pressupõe que os direitos já foram controlados antes de efetuar a chamada
  - o volume de memória disponível será sempre suficiente para executar o programa
    - mesmo assim sempre verifique se **`malloc( )`** retornou **`NULL`**
- Restrições: são condições que restringem a liberdade de escolha de alternativas de construção do artefato sendo especificado
  - o programa deverá ser redigido em C
  - o módulo deverá utilizar a biblioteca xpto.lib

Exemplos de requisitos não funcionais

- São exemplos de requisitos não funcionais:
  - o tempo de resposta deverá ser menor do que 10 ms
  - o módulo deverá ser portátil
    - melhor é enumerar os sistemas operacionais explicitamente
  - os dados obtidos através de janelas de diálogos devem ser válidos, independentemente da seqüência com que foram fornecidos ou editados.
  - todo espaço de dados temporário deve ser desalocado ao terminar uma função, independentemente da forma do término
  - nenhuma função desse módulo deverá cancelar a execução do programa ou *thread*
    - isso é um exemplo de requisito inverso

Exemplos de hipóteses

- o volume de memória disponível será sempre suficiente para executar o programa
  - mesmo assim sempre verifique se **`malloc( )`** retornou **`NULL`**
- os recursos X e Y estão bloqueados para uso exclusivo por esta função
- o usuário logado tem direitos de uso que permitem ativar esta função
  - pressupõe que os direitos já foram controlados antes de efetuar a chamada

Exemplos de restrições

- O programa deverá ser redigido em C
- O módulo deverá utilizar a biblioteca xpto.lib

Classes de especificação

- Especificação externa
  - destina-se a
    - desenvolvedores cliente
    - desenvolvedores do correspondente módulo servidor
      - desenvolvedores: desenvolvem e mantém o módulo
    - controladores da qualidade do módulo
    - produtores de documentação para o usuário (pessoa)
      - especificação da interface humana
      - especificação da sintaxe dos arquivos usados
  - informa como corretamente utilizar um módulo ao desenvolver um módulo cliente dele
  - define a interface a ser respeitada pela implementação
  - estará disponível no módulo de definição

Classes de especificação

- Especificação interna
  - destina-se a
    - desenvolvedores: programadores que irão implementar ou manter o correspondente módulo servidor
    - controladores da qualidade do módulo
  - define como implementar e manter o módulo
    - complementa a especificação externa
    - pode ser vazia (freqüentemente é vazia)
  - estará disponível no módulo de implementação

Classes de especificação

- Especificação de uso
  - destina-se a
    - usuários pessoas
    - produtores da documentação para o usuário
    - desenvolvedores
    - controladores da qualidade
  - define como interagir com programas que contenham este módulo
    - complementa as outras especificações
    - pode ser vazia (quase sempre é vazia)
  - estará disponível no módulo de definição

Exemplo de especificação de programa 1/2

- Preciso de um programa para registrar o esforço realizado por pessoas
- Ao desenvolver um projeto, diversas pessoas realizam atividades
  - Atividades podem ser particionadas em tarefas
- Quero saber
  - quanto tempo cada pessoa trabalhou por dia
  - em quais atividades cada pessoa atuou
  - em quais tarefas cada pessoa atuou
  - quanto tempo cada pessoa gastou por atividade e por tarefa
  - quanto tempo foi gasto nas atividades e tarefas de um projeto
- Tarefas podem ser classificadas de diversas maneiras, por exemplo:
  - especificação
  - projeto
  - implementação
  - depuração
  - controle da qualidade
  - estudo
- Quero saber o percentual de tempo gasto por natureza de tarefa

Exemplo de especificação de programa 2/2

- Surgem perguntas:
  - o que mesmo é uma atividade? E uma tarefa?
  - quem registra as naturezas das tarefas?
  - podem existir diversos projetos?
  - existem atividades não vinculadas a um projeto?
  - atividades e tarefas podem perdurar por vários dias?
  - várias pessoas podem atuar em uma mesma atividade?
  - várias pessoas podem atuar em uma mesma tarefa?
  - atividades estão relacionadas com artefatos?
  - atividades estão relacionadas com registros de problemas?
  - . . .

Exemplo de historietas (*extreme programming*) 1/2

- O registro de esforço é realizado através de uma folha de tempo. Cada folha de tempo informa o desenvolvedor, o dia, as observações relevantes relativas ao conjunto de eventos do dia e a relação das <tarefas, atividades> realizadas no decorrer deste dia por este desenvolvedor.
- Cada desenvolvedor registra o tempo consumido por tarefa realizada no decorrer de um determinado dia, informando hora inicial, hora final ou duração, natureza da tarefa, nome da atividade na qual se encaixa a tarefa e observações relativas à tarefa. Caso o desenvolvedor informe a hora inicial e a final, a duração será calculada. Caso informe a hora inicial e a duração, a hora final será calculada.
  - pergunta: tarefa não tem nome?

Exemplo de historietas 2/2

- Quando determinado usuário ativa o editor de folha de tempo, será buscada a folha de tempo correspondente ao dia atual conforme o relógio do computador. Caso já existam 1 ou mais tarefas registradas relativas ao dia atual, a folha é inicializada com esta lista de tarefas. Caso contrário a folha será deixada em branco.
- Ao clicar sobre um campo hora inicial ou hora final, se este estiver vazio, será inserida a hora corrente conforme o relógio do computador. Se não estiver vazio, será aberto um "pop up menu" perguntando se deve editar o campo. Caso o campo seja editado, deve ser verificada a validade: hora final é sempre pelo menos 1 minuto maior do que a hora inicial, hora é sempre menor ou igual ao valor dia/hora conforme registrado no relógio do computador.

Outro exemplo

- Todos os *strings* são mantidos em uma tabela residente em memória
- Cada *string* está associado a um identificador: **`idString`**.
- A função **`ObterTamanhoString( idString )`** retorna o tamanho do *string* **`idString`** contido na tabela, ou **`TBS_CondRetNaoString`** caso não exista.
- A função **`ObterString( idString )`** retorna o ponteiro para o *string* terminado em zero identificado por **`idString`**, ou **`NULL`** caso não exista
- As funções **`ObterTamanhoString(idString)`** e **`ObterString(idString)`** assumem que a tabela esteja. Caso contrário, os resultados serão imprevisíveis.

- Note:
  - os nomes usados são próximos ou iguais aos nomes usados no código fonte
  - preocupação com o significado e não com a implementação
    - qual é o tipo de idString?
      - este tipo de informação deve ser acrescentado mais tarde
  - texto simples e afirmativo (“é” ou “não é”)
  - de maneira geral, cada frase estabelece uma propriedade

Exemplo de especificação ruim

```
tpCondRet EmpilharValor( int   dimValor ,
                         void* pValor    )
/*   dimValor   - dimensão do valor */
/*   pValor     - ponteiro para o valor */
```

- Faltou dizer como é realizado o empilhamento
  - empilhar uma cópia do valor apontado
  - ou empilhar o ponteiro
- Faltou dizer o que acontecerá se ocorrer alguma falha durante o processamento
  - exemplos: capacidade esgotada, pValor nulo
- Faltou dizer qual é a unidade da dimensão: bytes, int, **...**

Critérios de qualidade de especificações

- Especificações devem ser de boa qualidade
  - quanto melhor a qualidade, menor será o retrabalho inútil
  - é estimado que 70% dos defeitos de um programa são causados por especificações inexistentes ou erradas
- Uma das técnicas de controle da qualidade de especificações é a leitura com intenção de encontrar defeitos
  - uma forma simples de fazer isso é verificar se a especificação satisfaz um conjunto de critérios
  - existem formas mais eficazes e bem mais complexas
    - desenvolvimento por pares de desenvolvedores

Critérios de qualidade de especificações

- Ausência de redundância
  - Redundância ocorre quando um mesmo item é especificado em vários lugares (duplicação, repetição)
  - Exemplo: descrever a interface com o usuário no cabeçalho do módulo e também na função que implementa esta interface
  - Exemplo: especificar a parte externada tanto no módulo de definição como no módulo de implementação
- Verificabilidade
  - deve ser possível determinar objetivamente a satisfação de cada item da especificação
- Testabilidade
  - é uma forma de verificabilidade em que a satisfação dos itens de especificação é verificada através de casos de teste

Critérios de qualidade de especificações

- Concisão
  - a especificação é redigida com poucas palavras
- Compreensibilidade
  - a especificação deve ser compreensível pelos diversos leitores, em particular por leigos em computação
- Não-ambigüidade (inequívoco)
  - diferentes leitores entendem o item exatamente da mesma maneira
- Exatidão
  - a especificação está em conformidade observável com o mundo real
- Prioridade
  - está claro o que é efetivamente requerido e o que é apenas desejado

Critérios de qualidade de especificações

- Explicitude (explícito + -(t)ude)
  - todos os itens da especificação estão explicitamente definidos e documentados
- Necessidade
  - a especificação contém somente os itens cuja presença possa ser justificada
- Suficiência
  - a especificação não omite aspectos relevantes
  - exemplo: a especificação de uma função relaciona todos os parâmetros e todas as variáveis globais e respectivos tipos
- Consistência
  - a especificação não contém contradições internas, ou com outros documentos

Critérios de qualidade de especificações

- Nivelamento
  - a especificação está no nível de abstração do artefato sendo especificado. Exemplos
    - para um usar um programa não interessa saber quais as estruturas de dados utilizadas
    - para a arquitetura do programa, são relevantes os modelos conceituais das estruturas de dados
    - para um módulo de definição é relevante o modelo conceitual relativo a esse módulo – especificação externa do módulo
    - para o módulo de implementação, são relevantes os modelos físicos – especificação interna do módulo
    - uma função assume e respeita a especificação interna

Critérios de qualidade de especificações

- Portatilidade (portátil + -(i)dade)
  - cada item da especificação deve ser o mais independente possível da tecnologia e da plataforma utilizada
- Viabilidade
  - cada item da especificação pode ser implementado pela equipe disponível
    - é teoricamente possível implementar com o desempenho desejado
      - computabilidade
      - análise de algoritmos
    - a equipe tem competência para implementar

Especificações no código

- Especificações podem ser incluídas como comentários no código
- Princípio básico para comentários
  - qualquer comentário deve adicionar informação não evidente a partir da leitura do código
  - Exemplo: se o nome de uma variável ou função explicita o seu significado, não adicione comentários
    - tpArvore * pArvore ; /* ponteiro para arvore */
      - comentário redundante, evite tais comentários !!!
    - ... pArvore ; /* ponteiro para arvore genealogica */ OK
      - mas por que não: ... pArvoreGenealogica ; ???

Como proceder?

- Antes de codificar um módulo, escreva a sua especificação externa no módulo de definição
  - depois, se necessário, redija a especificação interna no módulo de implementação
  - focalize nas propriedades do módulo como um todo
- Para cada função declarada no módulo de definição e para a qual não se consegue escrever uma assinatura suficientemente abrangente, escreva a sua especificação externa
- Redija casos de teste iniciais
  - scripts de teste são uma forma de especificar!
- Verifique se tudo está OK usando a lista de critérios de avaliação da qualidade de especificações
- Implemente algumas funções e teste
  - continue até concluir
- Reteste tudo, agora com casos de teste suficientemente rigorosos

FIM