Capítulo 7

Definição e Especificação de Requisitos

No Capítulo 4 dissemos que a engenharia de requisitos de um sistema é composta pelas atividades de análise, definição e especificação de requisitos, e no Capítulo 5 apresentamos uma metodologia sistemática para a análise de requisitos. Neste Capítulo veremos como podemos desenvolver a definição e a especificação de requisitos. Para isto cobriremos os seguintes tópicos:

- Ilustrar um método baseado em formulários para se descrever com precisão a definição dos requisitos de um sistema;
- Descrever alguns modos de se elaborar uma especificação de requisitos;
- Explicar a importância dos requisitos não funcionais que restringem o sistema desenvolvido e o processo de desenvolvimento;
- Descrever os diferentes tipos de requisitos não funcionais e como estes requisitos podem ser descritos.

Pois vamos aos métodos...

7.1 Definição dos requisitos

A definição de requisitos é uma descrição abstrata dos serviços que o sistema deve oferecer e das restrições sob as quais ele deve operar (requisitos funcionais e não funcionais). Esta definição deve apenas tratar do comportamento externo do sistema e não deve tratar das características do projeto, ou seja, a definição dos requisitos não deve ser definida usando um modelo de implementação.

A definição de requisitos tem por função modelar o sistema a ser desenvolvido e apresentar este modelo aos usuários, portanto este documento deve ser escrito utilizando-se modelos bastante inteligíveis como a linguagem natural e diagramas de fácil compreensão.

Recurso do grid. Para auxiliar no posicionamento das entidades em um diagrama, o usuário pode ativar o recurso do grid tanto em centímetros como em polegadas, através de uma opção do painel de controle. Inicialmente o grid está desativado. O grid pode ser ativado e desativado a qualquer momento durante a sessão de edição a unidade de medida pode ser alternada entre centímetros e polegadas também a qualquer momento. O grid deverá adaptar-se a a escala (zoom) que o usuário especifica para seu diagrama, contudo quando o zoom diminui, o número de linhas do grid deve diminuir para não poluir a tela com muitas linhas.

Figura 7.1: Definição de requisitos para o grid de um editor de diagramas.

À princípio a definição dos requisitos funcionais deve ser completa (todos os serviços do sistema devem ser descritos) e consistente (os requisitos não podem ter definições contraditórias), entretanto isto é praticamente impossível dada a inerente complexidade dos sistemas de software e por isto as inconsistências e a falta da descrição de alguns serviços podem não ser trivialmente identificáveis à princípio e só irão emergir com o tempo de processo. Outros problemas que identificamos na definição de requisitos são:

- **Falta de clareza.** É muito difícil utilizar a linguagem natural de forma precisa e inambígua sem tornar o documento muito extenso e de difícil leitura.
- **Confusão entre requisitos.** Requisitos funcionais e não funcionais, objetivos do sistema e informações de projeto não são facilmente distinguíveis.
- **Mistura de requisitos.** Vários de requisitos podem vir a ser descritos juntos como um único requisito.
- **Mistura de objetivos do documento.** É comum encontrarmos equipes de desenvolvimento que preferem misturar a definição e a especificação de requisitos em um único documento, o que também traz muita confusão.

O exemplo da Figura 7.1 é um requisito que faz parte da definição de requisitos de um editor de diagramas (que pode fazer parte do sistema descrito na Figura). Você pode pensar na definição de requisitos, à princípio, como uma lista parágrafos de texto como este da figura, cada um descrevendo um dos serviços oferecidos pelo sistema.

Vejamos os problemas com a definição deste requisito. A primeira sentença traz três tipos de requisitos.

1. Um requisito funcional: necessidade do editor de oferecer um grid. A sentença também traz uma justificativa (motivação) para este requisito.
2. Um requisito não funcional: informações detalhadas sobre as unidades de medida do grid.

3. Outro requisito não funcional: A definição de como e quando o grid é ativado ou desativado.

O requisito também traz algumas informações sobre a inicialização do grid dizendo que este é inicialmente desativado, mas não especifica que este inicialmente deve utilizar. Também está escrito que o usuário pode escolher a unidade de comprimento mas não diz se o usuário pode definir o espaçamento entre as linhas do grid.

Podemos ficar horas falando mal deste requisito pois são inúmeras os detalhes esquecidos em função do uso da linguagem natural, o que prova os deficiências em completeza e coerência do definição de requisitos que citamos.

Mas como preparar uma boa definição de requisitos? Para fazermos um bom trabalho podemos começar sistematizando o modelo a ser utilizado. Podemos começar inventando um formato padrão para os requisitos e descrevem todos os requisitos neste formato e assim diminuimos a possibilidade de ambiguidades e facilitamos a verificação de referências cruzadas.

A Figura 7.2 dá um exemplo de como podemos organizar o documento de definição dos requisitos. Neste exemplo os serviços e requisitos são numerados de forma hierárquica (como os capítulos e seções de um livro), no caso, a Figura mostra o serviço 2.6, *Recurso do Grid*, que é definido pelos requisitos 2.6.1 e 2.6.2. Para cada requisito é feita uma descrição de um único tópico que procura ser o mais desacoplado possível de outros requisitos, seguido por uma motivação que justifica o porque daquele requisito.

A anotação da motivação para cada requisito é interessante pois no futuro podemos precisar alterar um determinado requisito e podemos não nos lembrarmos o motivo daquele requisito ter sido definido de uma determinada forma e então ficarmos sem saber se podemos fazer a alteração.

O uso de notações especiais de texto como o uso de negrito, itálico ou fontes coloridas também pode fazer parte do padrão estabelecido para a definição de requisitos e podem ajudar a deixar o documento mais inteligível. No exemplo, foi utilizado negrito para destacar o requisito principal do serviço *Grid*.

Outro exemplo mais detalhado de definição de requisito neste mesmo formato é apresentada na Figura 7.3. Neste exemplo um requisito inclui uma lista de ações do usuário, mas não descreve detalhes de implementação.

Você pode notar que o texto da Figura 7.3 não descreve como o cursor é movimentado. Seria utilizando-se um mouse? Pode ser, mas se for de outro modo não descaracteriza o sistema sendo analisado e por isto não é comentado.

7.2 Especificação de requisitos

A especificação de requisitos traz informações adicionais a definição de requisitos podendo ser desenvolvida com uma linguagem mais estruturada e acompanhada de modelos do sistema (que vimos no capítulo anterior).

ë comum encontrarmos projetos onde a especificação de requisitos é feita utilizando-se linguagem natural. Entretanto, para este tipo de documentação a linguagem natural é

2.6 Recurso do Grid.

2.6.1 **O editor deve oferecer um recurso de grid composto por uma matriz de linhas horizontais e verticais assentadas no fundo da janela de edição.** Este deve ser um grid passivo e o alinhamento das entidades é de responsabilidade do usuário.
Motivação: Um grid ajuda o usuário a criar diagramas mais alinhados com as entidades devidamente espaçadas. Entretanto um grid ativo, onde as entidades são movidas automaticamente (*snap-on*) para as linhas do grid seja útil, o posicionamento neste caso é impreciso. O usuário é a melhor pessoa para decidir onde as entidades devem ser posicionadas.

2.6.2 Quando a escala (zoom) do diagrama (veja 2.1) é diminuída a norma (espaçamento em unidades de medida) entre as linhas deve aumentar.
Motivação: Se a norma não é aumentada, o fundo da janela ficará com muitas linhas do grid.

Figura 7.2: Definição de requisitos sistematizada para o grid de um editor de diagramas.

3.5.1 Adicionando Nodos ao Diagrama.

3.5.1.1 **O editor deve oferecer um recurso no qual os usuários podem adicionar nodos para um tipo específico de diagrama.** Nodos são *marcados como selecionados* (veja 3.4) quando são adicionados ao projeto.

3.5.1.2 A sequência de ações para se adicionar um nodo são:

(1) O usuário seleciona o tipo de nodo a ser adicionado.
(2) O usuário move o cursor para a posição aproximada do nodo no diagrama e indica que o símbolo do nodo deve ser adicionado naquele ponto.
(3) O símbolo é arrastado a sua posição final.

Motivação: O usuário é a melhor pessoa para decidir onde o nodo deve ser posicionado no diagrama. Esta abordagem dá ao usuário o controle direto sobre a seleção e o posicionamento de nodos.

Figura 7.3: Definição de requisitos para a criação de um nodo.

inadequada por alguns motivos que vemos abaixo, além daqueles já citados na definição de requisitos:

- Diferentes leitores podem fazer interpretações diferentes do mesmo texto. Esta é uma deficiência que advem da ambiguidade inerente na linguagem natural;
- A especificação de requisitos em linguagem natural é mais do que flexível pois é possível se dizer a mesma coisa de diversas maneiras. O leitor pode vir a identificar dois requisitos distintos quando se está falando à respeito de apenas um.
- A linguagem natural não permite que requisitos sejam particionados eficientemente. Para se descobrir os efeitos de uma mudança na especificação é necessário revisar a especificação inteira.

Por estes motivos uma especificação de requisitos em linguagem natural é difícil de ser compreendida e portanto é interessante encontrarmos ferramentas alternativas para diminuir a ambiguidade da especificação. Vejamos algumas:

- **Linguagem natural estruturada.** Esta abordagem baseia-se na elaboração de formulários padrão detalhados. Esta técnica é uma extensão da definição de requisitos baseada em formulário que acabamos de estudar.
- **Linguagens de descrição de projetos.** Baseia-se na utilização de uma linguagem similar a uma linguagem de programação mas com operações mais abstratas para a definição do modelo operacional do sistema.
- **Linguagens de especificação de requisitos.** Existem algumas linguagens desenvolvidas especialmente para a definição de requisitos como a PSL/PSA e a RSL, mas que não são comumente utilizadas em projetos do mundo real.
- **Notações gráficas.** Também existem algumas notações gráficas para a especificação de requisitos como a SADT, mas que costumam apresentar uma modelagem muito complexa sendo em geral restrita aos especialistas.
- **Especificação matemática.** Baseia-se em, quando possível, descrever o sistema utilizando-se uma notação de matemática, baseada em formulas algébricas. É totalmente inambígua mas em geral não reflete toda a complexidade de um sistema de grande porte.

Um documento de especificação de requisitos em geral é carregado de referências cruzadas, já que um sistema é sempre composto por módulos que interagem, portanto é necessário utilizemos convenções que permitam que referências sejam resolvidas rapidamente pelo leitor do documento[1]. Algumas técnicas que ajudam a estruturar o uso de referências cruzadas em um documento de especificação de requisitos são:

[1] Chamamos este problema de **restreabilidade de requisitos** (requirement taceability)

- Numerar todos os requisitos com um identificador único;
- Identificar o requisito ao qual se está referenciando através de seu número;
- Elaborar uma matriz ou ma tabela de referências cruzadas mostrando quais requisitos se relacionam.

7.2.1 Especificação com linguagem natural estruturada

Um forma de descrever a especificação de requisitos é utilizando a linguagem natural de forma estruturada, ou seja, utilizado um formulário bem definido que impeça que, mesmo utilizando uma linguagem mais flexível, tenhamos pouca ambiguidade. Especificações em linguagem natural podem ser facilmente compreendidos mas muito extensos e tediosos de serem escritos e lidos.

Para garantirmos a baixa ambiguidade da especificação precisamos elaborar um formulário padrão que deverá conter as seguintes informações:

- A descrição da função da entidade sendo especificada;
- A descrição das entradas de dados e sua origem (de onde vem);
- A descrição das saídas de dados e seu destino (para onde vão);
- A identificação das outras entidades requeridas por aquela entidade;
- Caso estejamos descrevendo um processo, a descrição das condições iniciais e finais do processamento;
- Uma descrição dos efeitos colaterais da operação.

A Figura 7.4 mostra como exemplo a especificação funcional do requisito *Adicionar um Nodo* utilizando a linguagem natural estruturada. Esta especificação é bem menos ambígua do que a definição mostrada na Figura 7.3 mas ainda apresenta alguns problemas que poderiam ser resolvidos se utilizássemos uma linguagem mais algorítmica. É que vemos à seguir.

7.2.2 Especificação com uma linguagem de descrição de algoritmos

Uma forma outra de especificar os requisitos de um sistema é utilizando uma linguagem de descrição de algoritmos que pode ser uma linguagem hipotética ou um a linguagem de programação como C, Java ou Pascal.

Seguindo a tradição do curso de Engenharia de Computação da FURG vamos utilizar a LDA (Linguagem de Especificação de Algoritmos) que é apresentada na apostila da disciplina de Estruturas de Dados [ROD01]. Esta é uma linguagem similar a linguagem Pascal, porém com identificadores em português e que possui uma estrutura de construção de tipos abstratos de dados (TAD), similar aos pacotes de Ada.

Função: Adicionar um Nodo

Descrição: Adiciona um nodo em um diagrama existente. O usuário seleciona o tipo de nodo e a sua posição. Quando adicionado ao diagrama, o nodo torna-se a seleção atual. O usuário escolhe a posição do nodo movendo o cursor para a área aonde o nodo foi adicionado.

Entrada: Tipo do nodo. Posição do nodo. Identificador do diagrama.

Origem: Tipo e Posição do nodo são fornecidas pelo usuário. Identificador do diagrama é fornecido pelo banco de dados de diagramas.

Saída: Identificador do diagrama.

Destino: Banco de dados de diagramas. O banco de dados é compromissado quando a operação é completada.

Requer: Que o identificador do diagrama deve ser um ponteiro para o grafo do diagrama.

Pré condições: O diagrama está aberto e é mostrado na tela do usuário.

Pós condições: O diagrama permanece imutável senão pela adição do novo nodo.

Efeitos colaterais: Nenhum.

Figura 7.4: Especificação de requisitos utilizando um formulário padrão.

Em um projeto do mundo real você pode escolher a linguagem que preferir para a especificação de requisitos. É sempre interessante escolher uma linguagem já familiar a aquelas pessoas que irão ler as especificações, que seja uniforme e concisa e que ofereça estruturas de tipos abstratos de dados ou de orientação a objetos. Linguagens como Ada que conduzem a uma programação mais estruturada são as melhores para esta atividade, já linguagens como C ou C++, embora muito eficientes e muito utilizadas na implementação de sistemas do mundo real, possuem uma sintaxe muito rebuscada e não são muito aconselháveis. Uma linguagem como Python é interessante quando deseja-se fazer uma especificação bastante abstrata. Já linguagens exóticas e minimalistas como Perl não devem ser utilizadas jamais na especificação de requisitos!

Na especificação de requisitos não vamos utilizar a linguagem para escrever um programa compilável, mas sim para demonstrar como um determinado sistema, ou uma parte de um sistema funciona. Assim podemos descrever a especificação utilizando algoritmos bem abstratos.

A melhor maneira de se utilizar a linguagem de LDA para especificar requisitos é acompanhando estes algoritmos de outros modelos. O uso de uma linguagem de descrição de algoritmos é mais indicado em duas situações:

- Quando a operação é especificada na forma de uma sequência simples de ações e a ordem destas ações e importante.
- Quando é necessário descrever as interfaces de hardware e software, especialmente

quando alguns subsistemas que já existem e precisam ser integrados ao novo sistema.

O uso da (ou *de uma*) LDA, para aquelas pessoas que já conhecem esta linguagem, torna a especificação de requisitos muito menos ambígua e fácil de se compreender. Se a linguagem utilizada é similar a linguagem com a qual o projeto será implementado, existe uma transição mais natural entre a especificação de requisitos, o projeto do sistema e a implementação. Mas existem algumas desvantagens no uso da LDA:

- A linguagem não ter todos os recursos necessários para descrever o domínio da aplicação de uma forma compreensível. Em especial, os clientes do sistema, mesmo os clientes especialistas, podem não estar familiarizado com linguagens de programação.
- A especificação utilizando a LDA pode ser vista como um projeto abstrato e não como uma especificação de requisitos e decisões de projeto podem ser tomadas cedo demais.

O Algoritmo 2 mostra como a LDA pode ser utilizada em uma situação onde a especificação de uma sequência de ações é importante. O exemplo do algoritmo descreve um sistema de auto-atendimento bancário que é um exemplo típico onde o uso de formulários seria ineficiente para descrever todo aquele processo de validação do cartão, validação da senha, escolha de um serviço, e assim por diante.

O uso de mnemônicos significativos neste processo é bastante importante. A especificação também não deve ficar sobrecarregada de detalhes que não faça diferença para a compreensão do sistema, como a preocupação com a tipagem e com variáveis auxiliares.

Especificação de interfaces

Sistemas em geral precisam comunicar-se com outros sistema já implementados respeitando protocolos de comunicação previamente definidos. Nestes casos é necessário especificar estas interfaces logo no começo do processo de desenvolvimento do software, e a especificação de requisitos é um bom local/momento para isto. Existem três tipos de interfaces que precisam ser definidas:

1. Interfaces procedurais onde subsistemas já existentes oferecem um conjunto de serviços que são acessados por uma interface de chamas ao sistema. Bons exemplos são a comunicação com drivers de periféricos de hardware – como servidores de impressoras, de modems, etc – ou artefatos de software que oferecem serviços específicos, ao sistema – como servidores de bancos de dados, o próprio sistema operacional, com suas *system calls*, entre outros. O Algoritmo 3 mostra um exemplo da descrição de uma interface procedural com um servidor de impressão. Observando esta definição é possível compreender quais operações são oferecidas por este servidor e daí começar a planejar como será a interação entre este sistema e o sistema que se quer desenvolver.

Algorithm 2 Descrição em LDA da operação de um sistema de auto-atendimento bancário.

```
proc Auto-atendimento;
    var
        senha : numero_senha;
        nro_conta : numero_conta;
        balanço : quantia;
        serviço : serviço_disponível;
        cartão_válido, senha_válida: lógico;
    início
        repita
            cartão_válido := pega_catão (nro_conta, senha);
            se (cartão_válido) então início
                senha_valida := valida_senha (senha);
                se (senha_valida) então início
                    balanço := lê_conta (nro_conta);
                    serviço := lê_serviço;
                    enquanto há_serviço faça início
                        faça_serviço_selecionado;
                        serviço := lê_serviço;
                    fim;
                    retorne_cartão;
                fim;
            fim;
        até desligar_o_sistema;
    fim;
```

Algorithm 3 Descrição em LDA de um sistema de impressão.

```
TAD SERVIDOR_DE_IMPRESSÃO def
    proc inicialize (P : IMPRESSORA);
    proc imprima (P : IMPRESSORA; F : ARQUIVO);
    proc exiba_fila_de_impressão (P : IMPRESSORA);
    proc cancela_impressão (P : IMPRESSORA; N : COD_IMPRESSÃO);
    proc troca_impressora (P1, P2 : IMPRESSORA; N : COD_IMPRESSÃO);
fim SERVIDOR_DE_IMPRESSÃO;
```

2. Estruturas de dados que devem ser passadas de um subsistema para outro. Neste caso é necessário que haja uma descrição mais forte da estrutura de dados em si. Suponhamos um exemplo em que dois sistemas (um já existente e outro que se está desenvolvendo) se comunicam através de trocas de mensagens. Este é um caso onde precisamos conhecer em detalhes a estrutura de dados que o sistema já existente utiliza. O Algoritmo 4 mostra com poderíamos especificar esta estrutura de dados, observemos que só depois de conhecer em detalhes a estrutura de dados utilizada para trocar mensagens é que podemos especificar que procedimentos podemos especificar com esta estrutura e quais as especificações não funcionais inerentes dela.

Algorithm 4 Representação de um formato intercambiável.

```
tipo MENSAGEM = reg (
        remetente : COD_SISTEMA;
        destinatário : COD_SISTEMA;
        hora_de_envio : DATA_E_HORA;
        comprimento : COMPR_MENSAGEM;
        terminador : CARACTERE;
        mensagem : TEXTO );
tipo COD_SISTEMA = variação entre 20000 e 30000;
tipo TIPO_ANO = variação entre 1980 e 2080;
tipo DATA_E_HORA = reg (
        segundos : NATURAL;
        ano : TIPO_ANO);
tipo COMPR_MENSAGEM = variação entre 0 e 10000;
tipo TEXTO = vet (COMPR_MENSAGEM) de CARACTERE;
```

3. Representações de dados que foram estabelecidas para um subsistema existente. Muito comum quando o desenvolvimento é feito à partir de componentes COTS. Se vamos utilizar um componente pronto é necessário compreender qual sua interface, pelos mesmos motivos do tipo anterior.

Em resumo: os tipos de requisitos que necessitam de uma especificação de interfaces são sistemas que utilizam serviços de outros sistemas (1), sistemas que comunicam-se com outros sistemas (2) e sistemas que utilizam componentes prontos em sua construção (3).

7.3 Requisitos não funcionais

A especificação de requisitos não funcionais também cumpre um papel importante na engenharia de requisitos de um sistema de software, pois descrevem as propriedades e as restrições do sistema. Em alguns sistemas os requisitos não funcionais são mais importantes do que os funcionais. Imagine por exemplo se um sistema embarcado de uma aeronave não é testado de acordo com um conjunto vasto e preciso de restrições.

Clientes impõem requisitos não funcionais por dois motivos:

- requisitos não funcionais
    - requisitos do produto
        - requisitos de usabilidade
        - requisitos de eficiência
            - requisitos de performance
            - requisitos de espaço
        - requisitos de robustez
        - requisitos de portabilidade
    - requisitos do processo
        - requisitos (prazos) de entrega do produto
        - requisitos de implementação
        - requisitos de padronização
    - requisitos externos
        - requisitos de interoperabilidade
        - requisitos éticos
        - requisitos legais
            - requisitos de privacidade
            - requisitos de segurança

Figura 7.5: Tipos de requisitos não funcionais.

- **Qualidade do sistema.** Em geral um bom processo de desenvolvimento leva a um bom produto.
- **Manutenibilidade do sistema.** Os requisitos do sistemas podem ser impostos de forma que os métodos de desenvolvimento utilizados para projetar e implementar o sistema sejam compatíveis com aqueles utilizados pelos mantenedores do sistema.

A especificação de requisitos não funcionais geralmente não foge muito da descrição em linguagem natural, embora você possa utilizar outras técnicas caso considere mais conveniente. Entretanto é interessante estabelecer uma classificação destes requisitos. [SOM96] estabelece uma classificação dos requisitos não funcionais em três grupos. A Figura 7.5 mostra uma listagem hierárquica de tipos de requisitos não funcionais de um sistema.

- **Requisitos do produto.** São requisitos que resultam das necessidade de que o sistema se comporte de uma determinada maneira em particular. Por exemplo temos a especificação de tempos de respostas para operações, quanta memória o sistema deve utilizar no máximo, requisitos de robustez e tolerância a falhas, requisitos de portabilidade e usabilidade, etc. Este tipo de requisito pode ser derivado diretamente da especificação que o usuário faz do sistema.
- **Requisitos organizacionais.** são conseqüências da política e dos procedimentos organizacionais do ambiente em que se encontra o sistema. Como exemplo temos

padrões de processo, linguagem de programação e metodologia de projeto, formato e tipos de documentos gerados, etc. Requisitos organizacionais pode derivar tanto do cliente quanto do desenvolvedor.

- **Requisitos externos.** Cobre todo os requisitos que emergem de fatores externos ao sistema e seu desenvolvimento. Como exemplo temos requisitos de interoperabilidade que define a interface entre sistemas de outras operações, requisitos legais e éticos, etc.

Conclusões

A distinção feita neste capítulo entre definição e especificação de requisitos dentro do contexto da disciplina de Projetos de Sistemas de Software, ou seja, dentro do estudo da Engenharia de Software, exerce uma função mais didática do que prática. Isto significa que em um processo de desenvolvimento de software do mundo real estas duas técnicas de modelagem de um sistema podem não representar a melhor forma de modelar um sistema dentro das restrições de prazo, custo e qualidade desejados. Entretanto o que deve prevalecer neste capítulo é que na atividade profissional de informática, é comum a necessidade de desecrever um sistema para diferentes pessoas com diferentes graus de conhecimento e emvolvimento com o processo, e a descrição do sistema para cada tipo de pessoa possui suas implicações específicas que você deverá observar, independente do modelo que irá utilizar.