# IEEE – PRÁTICAS RECOMENDADAS PARA ESPECIFICAÇÃO DE REQUISITOS DE SOFTWARE (sigla: SRS)

## 1. INTRODUÇÃO

Esta recomendação descreve abordagens recomendadas para as especificações de requisitos de software. Ela está dividida em 5 itens. O primeiro explica os objetivos desta recomendação. O segundo item lista as referências efetuadas para outros padrões. O terceiro fornece definições de termos específicos usados. O quarto item fornece informações para escrever uma boa especificação. O item 5 discute cada uma das partes essenciais de uma especificação de requisitos de software (SRS). Esta prática também contém um anexo, o qual fornece padrões de formatos alternativos.

### 1.1. OBJETIVOS

Esta é uma prática recomendada para escrever especificações de requisitos de software. Ela descreve o conteúdo e qualidades de uma boa especificação e apresenta vários exemplos de esboços de uma SRS.

Esta prática descreve é referência para especificar requisitos do software a ser desenvolvido, mas também pode ser aplicada para ajudar na seleção de produtos de software comerciais e caseiros.

Quando um software está embutido num grande sistema, como equipamentos médicos, então dúvidas além daquelas identificadas neste padrão devem ser expressas.

Esta prática descreve o processo de criar um produto e o conteúdo de um produto. O produto é uma especificação de requisitos de software. Esta prática pode ser usada para criar tanto a especificação propriamente dita ou pode ser usada como um modelo para muitos padrões específicos.

Esta prática não identifica nenhum método específico, nomenclatura ou ferramenta para preparar uma SRS.

## 2. REFERÊNCIAS

(Vide página 2 da documento original.)

## 3. DEFINIÇÕES

Em geral as definições de termos usados nesta prática estão conforme as definições fornecidas na IEEE Std 610.12.1990. As definições abaixo são termos chaves, portanto eles são usados nesta prática.

3.1. **Contrato**: um documento de compromisso legal acordado entre o cliente e o fornecedor. Incluem os requisitos técnicos e organizacionais, custo e cronograma para o produto. O contrato deve conter também informações informais, porém úteis tais como obrigações ou expectativas das partes envolvidas.

3.2. **Cliente**: A pessoa, ou pessoas, que pagam por um produto e geralmente (mas não necessariamente) decide os requisitos. No contexto desta prática, o cliente e o fornecedor devem ser membros da mesma organização.

3.3. **Fornecedor**: a pessoa, ou pessoas que produzem o produto para o cliente. No contexto deste documento, o cliente e o fornecedor devem ser membros da mesma organização.

3.4. **Usuário**: a pessoa, ou pessoas que operam ou interagem diretamente com o produto. O(s) usuário(s) e o cliente (s) geralmente não são a mesma pessoa.

## 4. CONSIDERAÇÕES PARA PRODUZIR UMA BOA SRS

Este item fornece informações adquiridas anteriormente que devem ser consideradas por escrito na SRS. Ele inclui:

a) natureza da SRS
b) ambiente da SRS
c) características de uma boa SRS
d) preparação em conjunto da SRS
e) evolução
f) protótipo
g) desenho de implantação na SRS
h) requisitos de projeto de implantação na SRS

### 4.1 Natureza da SRS

A SRS é uma especificação para um produto de software específico, programa ou conjunto de programas que realiza certas funções em um específico ambiente. Uma SRS deve ser escrita por um ou mais representantes do fornecedor, um ou mais representantes do cliente, ou por ambos. O subitem 4.4 recomenda ambos.

As questões básicas que os escritores da SRS falarão são as seguintes:

a) **Funcionalidade**. O que o software pretende fazer?

b) **Interfaces externas**. Como o software interage com as pessoas, com o hardware do sistema, outros hardwares e outros softwares?

c) **Performance**. Qual a velocidade, disponibilidade, tempo de resposta, tempo de restauração, recuperação das várias funções do software, etc?

d) **Atributos**. Qual a portabilidade, conectividade, manutenção, segurança, etc?

e) **Limites traçados estabelecidos na implementação.** Há padrões efetivados, linguagem implementada, políticas de integridade para Banco de Dados, limites de recursos, ambientes operacionais, etc?

O desenvolvedor deve evitar colocar também requisitos de projetos e desenhos na SRS.

Para conteúdos recomendados de uma SRS veja item 5.

## 4.2 Ambiente da SRS

É importante considerar o papel que a SRS faz no plano total do projeto, o qual é definido na IEEE Std 610.12.1990. O software deve conter essencialmente toda a funcionalidade de um projeto ou ele deve ser parte de um grande sistema. Neste caso tipicamente haverá uma SRS que relatará as interfaces entre o sistema e seu software e investigará a performance externa e os requisitos de funcionalidade do software. Claro que a SRS deve então concordar com isso e expandir sobre estes requisitos de sistema.

IEEE Std 1074-1991 descreve os passos do ciclo de vida de um software e as entradas aplicáveis para cada passo. Outras normas como as listadas no item 2, relatam as outras partes do ciclo de vida do software e assim complementam os requisitos de software.

Desde que a SRS tem as regras específicas no processo de desenvolvimento de software, desenvolvedores de SRS devem tomar cuidado para não ir além destas regras. Isto significa que a SRS:

a) deve definir corretamente todos os requisitos do software. Um requisito de software deve existir por causa da natureza da tarefa a ser resolvida ou por causa das características especiais do projeto.
b) Não deve descrever qualquer detalhe de desenho ou implementação. Estes devem ser descritos na fase de desenho do projeto.
c) Não deve estabelecer limites adicionais no software. Eles são especificados apropriadamente em outro documento como o plano que assegura a qualidade de um software.

Portanto, escrever apropriadamente uma SRS limita-se em ordenar os desenhos válidos, mas não especificar qualquer desenho particular.

## 4.3 Características de uma boa SRS

Uma SRS deve ser:

a) correta
b) não ambígua
c) completa
d) consistente
e) superior pela importância e/ou estabilidade
f) verificável
g) modificável
h) rastreável

### 4.3.1 Correta

Uma SRS é correta se, e somente se, cada requisito expresso nela for encontrado no software também.

Não há ferramenta ou procedimento que consiga medir a correção. Uma SRS deve ser comparada com qualquer especificação superior aplicável, como uma especificação de requisitos de sistema, com outra documentação de projeto e com outras normas aplicáveis, para assegurar sua

concordância. Alternadamente o cliente ou usuário pode determinar se a SRS reflete corretamente as necessidades atuais. A reconstituição torna este procedimento mais fácil e menos susceptível a erros.

### 4.3.2 Não obscura

Uma SRS não é obscura se, e somente se cada requisito declarado nela tiver somente uma interpretação. No mínimo isto requer que cada característica do produto final a ser descrito use um único termo. Nos casos onde o termo usado pode ter múltiplo significado, o termo deve ser incluído em um glossário onde seu significado é obtido mais especificamente.

Uma SRS é uma parte importante do processo de requisitos do ciclo de vida de um software e é usado em desenho, implementação, monitoração de projeto, verificação e validação e em treinamento como descrito na IEEE Std 1074-1991. A SRS deve ser clara para aqueles que a criam e para aqueles que a usam. De qualquer modo, estes grupos freqüentemente não têm a mesma experiência e portanto não tendem a descrever os requisitos de software da mesma maneira. Representações que aperfeiçoam as especificações de requisitos pelo desenvolvedor devem ser contra-produtivas à medida que eles diminuem o entendimento para o usuário e vice-versa.

Os subitens a seguir recomendam como evitar ambigüidades.

#### 4.3.2.1 Armadilha da linguagem natural

Requisitos são freqüentemente escritos numa linguagem natural (por. Ex. Inglês). A linguagem natural é inerentemente ambígua. A linguagem natural da SRS deve ser revisada por uma parte independente para identificar o uso de ambigüidades da linguagem, assim pode ser corrigido.

#### 4.3.2.2 Linguagens de especificação de requisitos

Uma forma de evitar a ambigüidade inerente da linguagem natural é escrever a SRS numa linguagem particular de especificação de requisitos. Esta linguagem detecta automaticamente muitos erros léxicos, sintáticos e semânticos.

Uma desvantagem no uso de tais linguagens é o tempo requerido para aprende-las. Também muitos usuários não técnicos acham-nas incompreensíveis. Além disso, estas linguagens tendem a ser melhores para expressar certos tipos de requisitos e direcionar certos tipos de sistemas. Assim, elas devem influenciar os requisitos de maneira sutil.

#### 4.3.2.3 Ferramentas de representação

Em geral, métodos e requisitos e linguagens e as ferramentas que os suportam dividem-se em 3 categorias gerais: objeto, processo e comportamental. Abordagens orientadas a objeto organizam os requisitos em termos de objetos do mundo real, seus atributos e os serviços realizados por estes objetos. Abordagens baseadas em processos organizam estes requisitos em hierarquias de funções que comunicam via fluxo de dados. Abordagens comportamentais descrevem o comportamento externo de um

sistema em termos de alguma noção abstrata (como cálculo de atributos), funções matemáticas ou calculadoras.

O nível que todas as ferramentas e métodos tornam-se úteis na preparação de um SRS depende do tamanho e complexidade do programa. Nenhum esforço é feito aqui para descrever ou endossar alguma ferramenta particular.

O uso de qualquer uma destas abordagens é melhor para manter as descrições da linguagem natural.

Desta maneira, clientes não familiarizados com as notações podem ainda entender a SRS.

### 4.3.3 Completa

Uma SRS é completa se e somente se, ela incluir os seguintes elementos:

a) Todos os requisitos significantes, se relacionados com funcionalidade, performance, limites de design, atributos ou interfaces externas. Em particular qualquer requisito externo referente a uma especificação de sistema deve ser reconhecido e tratado.
b) Definição das respostas do software para todas as classes realizáveis dos dados de entrada em todas as classes realizáveis de situações. Note que é importante especificar as repostas para os valores de entrada válidos e inválidos.
c) Labels completos e referências de todas as figuras, tabelas e diagramas na SRS e definição de todos os termos e unidades de moeda.

#### 4.3.3.1 Uso da TBD ( to be determined) (para ser determinado)

Qualquer SRS que usa a frase “para ser determinado” (TBD), não é uma SRS completa. O TBD é , de qualquer modo, ocasionalmente necessário e deve ser acompanhado por:

a) uma descrição das condições que causam o TBD (por exemplo, porque uma resposta não é conhecida), só assim a situação pode ser resolvida.
b) uma descrição do que deve ser feito para eliminar o TBD, quem é o responsável por sua eliminação e quando ele deve ser eliminado.

### 4.3.4 Consistente

Uma SRS é internamente consistente se, e somente se, nenhum subconjunto dos requisitos individuais descritos tiver em conflito. Há 3 tipos de conflitos mais comuns numa SRS.

a) As características de objetos do mundo real especificadas, podem conflitar. Por exemplo:
    1) O formato dos relatórios de saída, podem ser descritos em um dos requisitos, como tabular, mas em outro como textual.
    2) Um requisito pode dizer que todas as luzes serão verdes enquanto outros dizem que todas as luzes serão azuis.

b) Há conflitos lógicos ou temporais entre duas ações especificadas. Por exemplo:
    1) Um requisito pode especificar que o programa fará a adição de duas entradas e outro especifica que o programa fará a multiplicação delas.
    2) Um requisito pode dizer que “A” deve seguir “B” enquanto outro diz que “A” e “B” devem ocorrer simultaneamente.
c) Dois ou mais requisitos podem descrever o mesmo objeto, mas usar diferentes termos para este objeto. Por exemplo, uma solicitação para o usuário entrar pode ser chamada “prompt” em um requisito e “cue” (sinal de entrada) em outro. O uso de uma terminologia e definições padrões promovem consistência.

### 4.3.5 Alcançar um lugar de destaque pela importância e/ou estabilidade.

Uma SRS ocupa um lugar de destaque pela importância e/ou estabilidade se em cada requisito tiver um identificador para indicar também a importância ou estabilidade daquele requisito.

Tipicamente, todos os requisitos que relatam um produto de software não são igualmente importantes. Alguns requisitos podem ser essenciais, especialmente as aplicações para vida-critica, enquanto outros podem ser desejáveis.

Cada requisito na SRS deve ser identificado para tornar estas diferenças claras e explícitas. Identificar os requisitos a seguir pode ajudar:

a) ter clientes que tomam cuidado com cada requisito, que clarifica qualquer dúvida que eles tenham.
b) Ter desenvolvedores que tomam corretas decisões de design e dedicam níveis de esforços apropriados para diferentes partes do produto de software.

#### 4.3.2.1 Grau de Estabilidade

Um método de identificar os requisitos utiliza a dimensão de estabilidade. Estabilidade pode ser expressa em termos de número de expectativas de mudanças para qualquer requisito, baseado na experiência ou no conhecimento de eventos prestes a surgir, que afetam a organização, funções e pessoas representadas pelo sistema.

#### 4.3.2.2. Grau de Necessidade

Uma outra maneira para destacar os requisitos é distinguir classes de requisitos como essencial, condicional e opcional:

a)Essencial. Indica que o software não será aceitável a menos que estes requisitos forem providos de um processo de concordância.

b)Condicional. Implica que estes são requisitos que realçariam o produto de software, mas não o tornariam inaceitável se eles estiverem ausentes.

c)Opcional. Indica as classes de funções que podem ou não valer a pena serem observadas. Isto dá ao desenvolvedor a oportunidade de propor alguma coisa que exceda a SRS.

### 4.3.6 Verificável

Uma SRS é verificável se, e somente se, cada requisito expresso nela for verificável. Um requisito é verificável se e somente se, existir algum processo de custo-efetivo com o qual a pessoa ou máquina pode checar que o produto de software encontra o requisito. Em geral qualquer requisito ambíguo não é verificável.

Requisitos não verificáveis incluem confirmações como “funciona bem”, “boa interface humana” e “geralmente acontecerá”. Estes requisitos não podem ser verificados porque é impossível definir os termos bom, bem ou geralmente. A afirmação que “o programa nunca entrará num loop infinito” é não verificável porque o teste desta qualidade é teoricamente impossível.

Um exemplo de uma sentença verificável é “A saída o programa será produzida dentro de 20 segundos do evento X 60% do tempo; e será produzida dentro de 30 segundos do evento X 100% do tempo”.

Esta afirmação pode ser verificada porque ela usa concretamente termos e quantidades mensuráveis.

Se um método não pode ser idealizado para determinar se um software encontra um requisito particular, então o requisito deve ser removido ou revisado.

### 4.3.7 Modificável

Uma SRS é modificável se, e somente se, sua estrutura e estilo permitem qualquer mudança, onde os requisitos podem ser feito facilmente, completamente e consistentemente, enquanto mantêm a estrutura e estilo. Modificabilidade geralmente requer que uma SRS:

a) tenha uma organização coerente, fácil de usar, contendo uma tabela de conteúdos, um índice e uma explícita referência cruzada.
b) Não seja redundante; isto é, o mesmo requisito não pode aparecer em mais do que um lugar na SRS.
c) Expresse cada requisito separadamente, ao invés do que mesclado com outros requisitos.

Redundância por si só não é um erro, mas ela pode facilmente levar ao erro. Redundância pode ocasionalmente ajudar a fazer uma SRS mais legível, mas um problema pode ocorrer quando o documento redundante é atualizado. Por exemplo, um requisito pode ser alterado em somente um dos lugares onde ele aparece. Uma SRS então se torna inconsistente. Quando a redundância for necessária, a SRS deve incluir explicitamente uma referência cruzada para fazer sua modificação.

### 4.3.8 Rastreável (reconstituível)

Uma SRS é rastreável se a origem de cada um dos seus requisitos for clara e se facilitar a referência de cada requisito no desenvolvimento futuro ou para enriquecer a documentação. Dois tipos de rastreamento são recomendados:

a) rastreamento para trás (isto é para estágios prévios de desenvolvimento). Isto depende de cada requisito explicitamente referenciando sua fonte em seus primeiros documentos.

b) Rastreamento para frente (isto é, para todos os documentos gerados pela SRS). Isto depende de cada requisito na SRS ter um único nome ou número de referência.

O rastreamento para frente é especialmente importante quando o produto do software entra na faze de operação e manutenção. Quando o código e o desenho de documentos são modificados, é essencial ter a possibilidade de acertar o conjunto completo de requisitos que devem ser afetados por aquelas modificações.

### 4.4. Preparação conjunta da SRS

O processo de desenvolvimento de software deve começar com o acordo entre fornecedor e cliente sobre o que o software completo deve fazer. Este acordo, em forma de SRS, deve ser preparado em conjunto. Isto é importante porque normalmente nem o cliente, nem o fornecedor está qualificado para escrever uma boa SRS sozinho.

a) clientes normalmente não entendem o processo de projeto e desenvolvimento de software o suficiente para escrever uma útil SRS.
b) Fornecedores geralmente não entendem o problema do cliente e a área de alcance o suficiente para especificar requisitos para um sistema satisfatório.

Portanto, o cliente e o desenvolvedor devem trabalhar juntos para a produção de uma SRS bem elaborada e completamente entendida.

Uma situação especial existe quando um sistema e seus softwares estão sendo definidos concomitantemente. Então a funcionalidade, interfaces, performance e outros atributos e contratempos do software não são pré-definidos, mas ao contrário, são definidos em conjunto e sujeitos a negociação e mudança. Isto torna mais difícil o processo, mas não menos importante, para encontrar características expressas no item 4.3. Em particular, a SRS que não está de acordo com os requisitos de sua especificação de software principal está incorreta.

Esta prática não discute apenas o estilo, a linguagem usada, ou técnicas de boa escrita. Mas é muito importante que uma SRS seja bem escrita. Livros de técnicas gerais de escrita podem ser usados como guia.

### 4.5 Expansão de uma SRS

Uma SRS necessita expandir-se com o progresso do desenvolvimento de um produto de software. Isto pode ser impossível de especificar alguns detalhes, no momento que o projeto é iniciado. Por exemplo, pode ser impossível definir todos os formatos de tela para um programa interativo durante a fase de requisitos. Mudanças adicionais podem acontecer quando deficiências, falhas e imprecisões são descobertas na SRS.

As duas maiores considerações neste processo são o seguinte:

a) requisitos devem ser especificados minuciosamente e completamente conforme são conhecidos, mesmo se revisões para expansão puderem ser pressupostas como inevitáveis. O fato de eles estarem incompletos deve ser anotados.
b) Um processo de mudança formal deve ser iniciado para identificar, controlar, localizar e relatar mudanças projetadas.

Mudanças aprovadas nos requisitos devem ser incorporadas na SRS de tal maneira que:

1) forneça uma completa e precisa pista verificada das mudanças
2) permita uma revisão da parte atual e substituída da SRS.

## 4.6 Protótipo

Protótipo é usado freqüentemente durante a parte de requisitos de um projeto. Muitas ferramentas existem que permitem um protótipo, exibindo algumas características do sistema, podendo ser criado rápida e facilmente. VEja também ASTM Std 1340-90.

Protótipos são usados por 3 razões:

a) o cliente torna-se mais amigável ao ver o protótipo e contra atacá-lo do que ler uma SRS e contra atacá-la. Além de que o protótipo permite um rápido feedback.
b) O protótipo mostra aspectos antecipados do comportamento de um sistema. Além de produzir não somente respostas mas também novas questões. Isso ajuda a concluir a SRS.
c) Uma SRS baseada no protótipo tende a estar sujeita a menos mudanças durante o desenvolvimento e assim reduz o tempo de desenvolvimento.

Um protótipo deve ser usado como uma maneira de descobrir requisitos de software. Algumas características tais como telas e formatos de relatórios podem ser determinados imediatamente do protótipo. Outros requisitos podem ser concluídos executando experimentalmente um protótipo.

## 4.7. Design de Implantação na SRS

Um requisito especifica uma função externamente visível ou um atributo de um sistema. Um design descreve um sub-componente particular de um sistema e/ou suas interfaces com outros sub-componentes. O desenvolvedor da SRS deve distinguir claramente entre identificar os limites do design requeridos e projetar um design específico. Note que cada requisito na SRS limita designes alternativos. Isto não significa, contudo, que cada requisito é um design.

A SRS deve especificar quais funções são transformadas em quais dados para produzir quais resultados, para quem, de que lugar. Uma SRS deve focar no serviço a ser transformado. Uma SRS não deve normalmente especificar itens de design conforme segue:

a) particionar o software dentro dos módulos
b) alocar funções para os módulos
c) descrever o fluxo de informações ou controle entre os módulos
d) escolher estrutura de dados

### 4.7.1. Requisitos de design necessários

Em casos especiais alguns requisitos podem severamente se restringir ao desenho. Por exemplo, requisitos de segurança podem refletir diretamente no desenho para:

a) manter certas funções em módulos separados
b) permitir somente comunicação limitada entre algumas áreas do programa
c) checar a integridade de dados para variáveis críticas.

Exemplo de desenhos válidos são requisitos físicos, performance de requisitos, normas de desenvolvimentos de software e normas de precisão e qualidade de software.

Portanto, os requisitos devem ser expressos de um ponto de vista puramente externo. Quando se usam modelos para ilustrar os requisitos, lembre que o modelo só pode indicar o comportamento externo e não especificar um desenho.

### 4.8. Requisito de Projeto de Implantação na SRS

A SRS deve visar um produto de software, não o processo de produzir o software.

Requisitos de projeto representam um entendimento entre cliente e fornecedor sobre questões contratuais pertencentes à produção de software e deste modo não deve ser incluído na SRS. Estes normalmente incluem itens tais como:

a) custo
b) cronograma de entrega
c) descrição de procedimentos
d) métodos de desenvolvimentos de software
e) controle de qualidade
f) critério de validação e verificação
g) procedimentos de aprovação

Requisitos de projeto são especificados em outros documentos, tipicamente no plano de desenvolvimento de software, no plano de controle de qualidade do software ou no plano de trabalho.

## 5. AS PARTES DE UMA SRS

Este item discute cada uma das partes essenciais de uma SRS. Estas partes estão dispostas na figura 1, num esboço que pode servir de exemplo para escrever uma SRS.

Tabela de conteúdos:

1. Introdução
   1.1 Objetivo Geral
   1.2 Objetivo Específico (escopo)
   1.3 Definições, siglas e abreviações
   1.4 Referências
   1.5 Visão Geral

2. Descrição Global
   2.1 Aspecto Geral do Produto
   2,2 Funções do Produto
   2.3 Características do usuário
   2.4 Limites
   2,5 Suposições e Dependências

3. Requisitos Específicos (Veja 5.3.1 – 5.3.8 para explicações de possíveis requisitos específicos. Veja também anexo A para diferentes organizações desta seção de SRS)

Apêndices

Índice

Figura 1. Esboço de um Protótipo de SRS

Mesmo que uma SRS não seguir este esboço ou usar nomes dados aqui para suas partes, uma boa SRS deve incluir todas as informações discutidas aqui.

## 5.1 Introdução (seção 1 da SRS)

A introdução da SRS deve fornecer uma visão geral da SRS inteira. Ela deve conter as seguintes subseções:

a) Objetivo Geral
b) Objetivo Específico (escopo)
c) Definições, siglas e abreviações
d) Referências
e) Visão Geral

### 5.1.1. Objetivo Geral (1.1 de uma SRS)

Esta subseção deve conter o seguinte:

a) delinear o objetivo da SRS
b) especificar o público alvo da SRS

### 5.1.2. Objetivo Específico (1.2 da SRS)

Esta subseção deve:

a) identificar o produto do software a ser produzido pelo nome (por exemplo: Geração de Relatório, etc...)
b) explicar o que o produto de software fará e se necessário o que não fará
c) descrever a aplicação do software a ser especificado, incluindo benefícios relevantes, objetivos e metas.
d) Ser consistente com sentenças idênticas nas especificações de alto nível (por exemplo, a especificação de requisitos de sistema), se elas existirem.

### 5.1.3. Definição, siglas e abreviações (1.3 da SRS)

Esta subseção deve fornecer as definições de todos os termos, siglas e abreviações necessárias para interpretar apropriadamente a SRS. Esta informação deve ser fornecida por referência para uma ou mais apêndices na SRS ou por referência para outros documentos.

### 5.1.4. Referências

Esta subseção deve:

a) fornecer uma lista completa de todos os documentos referenciados na SRS
b) identificar cada documento por título, nº do relatório (se aplicável), data e gráfica.
c) Especificar as origens das referências

Esta informação pode ser fornecida de preferência por um apêndice ou por outro documento.

#### 5.1.5. Visão Geral (1.5 da SRS)

Esta subseção deve:

a) descrever o que a base da SRS contém
b) explicar como a SRS está organizada

### 5.2. Descrição Global (seção 2 da SRS)

Esta seção da SRS deve descrever os fatores gerais que afetam o produto e seus requisitos. Esta seção não refere a requisitos específicos. Ao contrário, ela fornece uma base para estes requisitos, que são definidos em detalhe na seção 3, e torna-os mais fáceis de entender.

Esta seção normalmente consiste de seis subseções, conforme segue:

a) Aspecto Geral do produto
b) Funções do produto
c) Características do usuário
d) Limites
e) Suposições e Dependências
f) Conjunto de Requisitos

#### 5.2.1. Aspecto Geral do Produto

Esta subseção da SRS deve colocar o produto em perspectiva com outros produtos relacionados. Se o produto é independente e possui conteúdo próprio, ele deve ser então dito aqui. Se a SRS define um produto que é um componente de um grande sistema, como freqüentemente ocorre, então esta subseção deve relatar os requisitos do grande sistema para a funcionalidade do software e deve identificar as interfaces entre o sistema e o software.

Um diagrama de bloco mostrando os principais componentes do grande sistema, interconexões e interfaces externas podem ser úteis.

Esta subseção deve também descrever como o software opera dentro de vários limites. Por exemplo, poderia incluir:

a) interfaces do sistema
b) interfaces do usuário
c) interfaces do hardware
d) interfaces do software
e) interfaces de comunicação
f) limites de memória

g) operações
h) requisitos de adaptação do local

#### 5.2.1.1. Interfaces do Sistema

Deve listar cada interface do sistema e identificar a funcionalidade do software para aperfeiçoar os requisitos do sistema e a descrição da interface que se une ao sistema.

#### 5.2.1.2. Interfaces do Usuário

Deve especificar o seguinte:

a) As características lógicas de cada interface entre o produto de software e seus usuários. Isto inclui as características de configuração (isto é, formato de telas, layouts de página ou janela, conteúdo de relatórios ou menus, ou disponibilidade de chaves de função programada) necessárias para completar os requisitos de software.
b) Todos os aspectos de otimizar a interface com a pessoa que deve usar o sistema. Isto pode simplesmente abranger uma lista de faça e não faça, quando o sistema aparecer para o usuário. Um exemplo pode ser um requisito para a opção de pequenas e grandes mensagens de erros. Como todos os outros, estes requisitos devem ser verificáveis, por ex., “um digitador grua 4 pode fazer a função X em Z minutos depois de uma hora de treinamento”, ao invés de “um digitador pode fazer a função X.” (Isto deve também ser especificado nos atributos do sistema sob a seção chamada “Fácil de usar”.)

#### 5.2.1.3. Interfaces de hardware

Deve especificar as características lógicas de cada interface entre o produto de software e os componentes de hardware do sistema. Isto inclui características de configuração (nº de portas, conjunto de instruções, etc). Deve também cobrir assuntos como quais dispositivos serão suportados, como eles serão suportados e protocolos. Por exemplo, suporte para terminal deve especificar suporte de tela inteira ao invés de linha a linha.

#### 5.2.1.4. Interfaces de Software

Deve especificar o uso de outros produtos de software requeridos (por exemplo, um sistema de gerenciamento de dados, um sistema operacional, ou um pacote matemático), e interfaces com outros sistemas aplicativos (por exemplo, o link entre um sistema de contas a receber e um sistema de contabilidade geral). Para cada produto de software requerido deve ser fornecido o seguinte:

a) nome
b) mnemônico
c) nº de especificação
d) nº de versão
e) fonte

Para cada interface, deve ser fornecido o seguinte:

a) Discussão dos objetivos do interfaceamento do software como relatadas para este produto de software.
b) Definição da interface em termos de conteúdo e formato da mensagem. Não é necessário detalhar qualquer interface bem documentada, mas, uma referência para o documento definindo a interface, é requerido.

#### 5.2.1.5. Interfaces de Comunicação

Deve especificar as várias interfaces para comunicação como protocolos de redes locais, etc.

#### 5.2.1.6. Limites de Memória

Deve especificar quaisquer características e limites aplicáveis na memória primária e secundária.

#### 5.2.1.7. Operações

Deve especificar operações normais e especiais requisitadas pelo usuário assim como:

a) Os vários modos de operação na organização do usuário; por exemplo, operações iniciadas pelo usuário.
b) Períodos de operação interativa e período de operações não atendidas.
c) Funções de suporte de processamento de dados
d) Operações de backup e recuperação

Nota: Esta seção muitas vezes é especificada como parte da seção de interfaces do usuário.

#### 5.2.1.8. Requisitos de adaptação do local

Deve:

a) Definir os requisitos para qualquer seqüência de inicialização ou de dados que são específicas para um dado local, missão ou modo operacional, por exemplo, valores de rede, limites de segurança, etc.
b) Especificar o local ou características da referida missão que deve ser modificada para adaptar o software para uma particular instalação.

### 5.2.2. Funções do Produto

Esta subseção da SRS deve fornecer um sumário das principais funções que o software realizará. Por exemplo, uma SRS para um programa de contabilidade deve usar esta parte para falar sobre a manutenção da conta

cliente, confirmação do cliente e preparação da fatura sem mencionar a vasta quantidade de detalhes que cada uma destas funções requerem.

Algumas vezes o sumário da função que é necessário para esta parte, pode ser tomada diretamente da seção de especificação de alto nível (se ela existe), que aloca funções específicas para o produto de software. Preste atenção na clareza:

a) As funções devem ser organizadas através de uma lista de funções compreensíveis para o cliente ou para qualquer um, além da leitura do documento num primeiro momento.
b) Métodos textuais ou gráficos podem ser usados para mostrar as diferentes funções e seus relacionamentos. Assim como um diagrama não é usado para mostrar um desenho de um produto mas simplesmente mostra os relacionamentos lógicos entre variáveis.

### 5.2.3. Características do Usuário (2.3 da SRS)

Esta subseção da SRS deve descrever as características gerais dos usuários do produto incluindo o nível educacional, experiência e habilidade técnica. Não deve ser usado para falar sobre requisitos específicos mas deve fornecer justificativas, (pois requisitos específicos são certamente especificados mais tarde na seção 3 da SRS).

### 5.2.4. Limites (2.4 da SRS)

Esta subseção da SRS deve fornecer uma descrição geral de qualquer outro item que limitará as opções do desenvolvedor. Estas incluem:

a) política reguladora
b) limitações de hardware (por exemplo, requisitos de sincronização de sinal)
c) interfaces para outras aplicações
d) operação paralela
e) funções de auditoria
f) funções de controle
g) requisitos de linguagem de mais alta ordem
h) protocolos de transmissão de sinal (por exemplo: XON-XOFF, ACK-NACK)
i) requisitos de segurança
j) consistência da aplicação
k) considerações de segurança e estabilidade

### 5.2.5. Suposições e Dependências (2.5 da SRS)

Esta subseção da SRS deve listar cada um dos fatores que afetam os requisitos expressos na SRS. Estes fatores não são os limites do software, mas são quaisquer mudanças que podem afetar os requisitos na SRS. Por exemplo, uma suposição pode ser que um sistema operacional específico estará disponível no hardware designado para o produto de software. Se, de fato, o sistema operacional não estiver disponível, a SRS deve então ter que mudar, coerentemente.

### 5.2.6. Particionamento de requisitos

Esta subseção da SRS deve identificar os requisitos que podem ser adiados até as versões futuras do sistema.

## 5.3. Requisitos Específicos (seção 3 da SRS)

Esta seção da SRS deve conter todos os requisitos de software num nível de detalhe suficiente para que os desenvolvedores estejam aptos para satisfazer estes requisitos no desenvolvimento do sistema, e testarem para que o sistema satisfaça estes requisitos. Em toda a parte desta seção, cada requisito expresso deve ser compreendido externamente pelos usuários, operadores ou outros sistemas externos. Estes requisitos devem incluir no mínimo uma descrição de cada input (estímulo) dentro do sistema, cada output (resposta) do sistema e todas as funções realizadas pelo sistema em resposta a uma input ou em suporte a uma output. Como isto é freqüentemente a maior e a mais importante parte da SRS, os seguintes princípios devem ser observados:

a) requisitos específicos devem ser expressos em conformidade com todas as características descritas no item 4.3. desta recomendação.

b) requisitos específicos devem ter sido referenciados em documentos anteriores a este relato.

c) Todos os requisitos devem ser unicamente identificáveis

d) Atenção cuidadosa deve ser dada para organizar os requisitos para uma melhor interpretação.

Antes de examinar as maneiras específicas de organizar os requisitos é útil entender os vários itens que abrangem os requisitos conforme descritos nos seguintes subitens:

### 5.3.1. Interfaces externas

Deve ser uma descrição detalhada de todas as inputs e outputs do software do sistema. Deve complementar as descrições das interfaces no item 5.2 e não deve repetir informação.

Deve incluir os conteúdos e formatos conforme segue:

a) nome do item
b) descrição dos objetivos
c) fonte da input ou destino da output
d) classificação válida, média e/ou tolerada
e) unidades de medida
f) sincronização/cronometragem
g) relacionamentos com outras inputs/outputs
h) organização/formato de telas
i) organização/formato de janelas
j) formato de dados
k) formato de comandos
l) mensagens finais

### 5.3.2. Funções

Requisitos funcionais devem definir as ações fundamentais que devem acontecer no software na aceitação e processamento das entradas e no processamento e geração de saídas. Estas são geralmente listadas como comandos obrigatórios (Shall) começando com Shall do Sistema.

Elas incluem:

a) checar a validade das entradas
b) seqüência exata das operações
c) respostas para situações anormais, incluindo:
    1) overflow
    2) facilidades de comunicação
    3) erros manuais e recuperação
d) efeitos de parâmetros
e) relacionamento de outputs para input, incluindo:
    1) seqüências de input/output
    2) fórmulas para conversão de input para output

Isto deve ser apropriado para dividir os requisitos funcionais dentro de subfunções ou subprocessos. Isto não implica que o software projetado também será dividido desta maneira.

### 5.3.3. Performance de Requisitos

Esta subseção deve especificar os requisitos numéricos estáticos e dinâmicos existentes no software ou uma interação humana com o software como um todo. Requisitos numéricos estáticos devem incluir:

a) o nº de terminais
b) o nº de usuários simultâneos
c) quantidade e tipo de informação a ser manuseada

Requisitos numéricos estáticos são identificados numa seção separada.

Requisitos numéricos dinâmicos devem incluir por exemplo, os números de transações e tarefas e a quantidade de dados a serem processados dentro de um certo período de tempo em condições normais e de pico de trabalho.

Todos este requisitos devem ser expressos em termos mensuráveis. Por exemplo:

95% das transações serão processadas em menos de 1 segundo, ao invés de

Um operador não deve esperar para a transação ser completada.

Nota: limites numéricos aplicados numa função específica são normalmente especificados como parte de um subparágrafo da descrição do processamento desta função.

### 5.3.4. Requisitos do Banco de Dados Lógico

Deve especificar os requisitos lógicos para qualquer informação que pertencerá ao banco de dados. Deve incluir:

a) tipos de informação usada pelas várias funções
b) freqüência de uso

c) capacidade de acesso
d) entidade de dados e seus relacionamentos
e) limites de integridade
f) requisito de retenção de dados

### 5.3.5. Limites do desenvolvimento

Deve especificar limites do desenvolvimento que podem ser impostos por outros padrões, limitação de software, etc.

#### 5.4.5.1. Regras de Concordância

Esta subseção deve especificar os requisitos derivados de normas ou regulamentos padrões existentes. Eles devem incluir:

a) formato de relatório
b) denominação de dados
c) procedimentos de cálculos
d) registros de verificação

Por exemplo, este item pode especificar os requisitos para o software para registros de atividades de processamento. Tais requisitos são necessários para algumas aplicações para encontrar normas financeiras e regulamentais. Um requisito de registro de verificação, por exemplo, exprime que todas as mudanças no Banco de Dados da Folha de Pagamento devem ser registradas em um arquivo constituído dos valores de antes e depois do processamento.

### 5.3.6. Atributos dos programas do sistema

Há um número de atributos de software que pode servir como requisito. É importante que os atributos requeridos sejam especificados, assim que o processamento deles seja objetivamente verificado.

Os seguintes itens fornecem uma lista parcial de exemplos:

#### 5.3.6.1. Confiabilidade

Deve especificar os fatores requeridos para estabelecer a confiabilidade do software do sistema no momento da entrega.

#### 5.3.6.2. Disponibilidade

Deve especificar os fatores requeridos para garantir o nível de disponibilidade para todo o sistema, como o ponto de checagem (checkpoint), recuperação (recovery) e recomeço (restart).

#### 5.3.6.3. Segurança

Deve especificar os fatores que protegem o software de um acidental ou malicioso acesso, uso, modificação, destruição ou descoberta.

Requisitos específicos nesta área pode incluir a necessidade de:

a) utilizar certas técnicas de criptografia
b) manter logs específicos ou histórico do conjunto de dados

c) atribuir certas funções para diferentes módulos
d) comunicação restrita entre algumas áreas do programa
e) checar a integridade de dados para variáveis críticas

#### 5.3.6.4. Capacidade de Manutenção

Deve especificar atributos de software que relatam os casos de manutenção do software por ele mesmo. Pode ser algum requisito para determinada modularidade, interface, complexidade, etc.

#### 5.3.6.5. Portabilidade

Deve especificar atributos de software que relatam o caso de transportar o software para outra máquina e/ou sistemas operacionais. Isto deve incluir:

a) porcentagem de componentes com código host-dependente
b) porcentagem de código que é host-dependente
c) uso de uma comprovada linguagem portável
d) uso de um compilador próprio ou configurador de linguagem
e) uso de um sistema operacional próprio

### 5.3.7. Organizando os requisitos específicos

Para qualquer sistema simples, os requisitos detalhados tendem a ser extensos. Por esta razão, é recomendado que considerações cuidadosas devem ser dadas para organizá-los de uma forma ótima e compreensível. Não há uma organização ótima para todos os sistemas. A seção 3 da SRS mostra as diferentes organizações de requisitos para diferentes classes de sistemas.. Algumas dessas organizações são descritas nos subitens a seguir:

#### 5.3.7.1. Modo do sistema

Alguns sistemas se comportam bem diferente dependendo do modo de operação. Por exemplo, um sistema de controle pode ter diferentes conjunto de funções dependendo do seu modo: teste, normal ou emergência. Quando organizar esta seção por modo, use o esboço mostrado no anexo A1 ou A2. A escolha depende se as interfaces e performances são dependentes no modo.

#### 5.3.7.2. Classe de usuários

Alguns sistemas fornecem diferentes conjuntos de funções para diferentes classes de usuários. Por exemplo, um sistema de controle de elevador apresenta diferentes capacidades de passageiros, trabalhadores de manutenção e bombeiros. Quando organizar esta seção por classe de usuário, use o esboço mostrado no anexo A3.

#### 5.3.7.3. Objetos

Objetos são entidades do mundo real que tem uma contra-partida dentro do sistema. Por exemplo, num sistema de monitoramento de um

paciente, os objetos incluem o paciente, sensores, enfermeiros, salas, médicos, remédios, etc. Associado com cada objeto há um conjunto de atributos (daquele objeto) e funções (realizadas por aquele objeto). Estas funções são também chamadas de serviços, métodos e processos. Quando organizar esta seção por objeto, use o esboço mostrado no anexo A4. Note que conjuntos de objetos devem possuir atributos e serviços. Estes são agrupados juntos, conforme as classes.

#### 5.3.7.4. Particularidades

Uma particularidade é um serviço desejado externamente pelo sistema que pode requerer uma seqüência de inputs para efetuar os resultados desejados. Por exemplo, num sistema de telefonia, particularidades incluem chamadas locais, chamadas protegidas e chamadas de conferência. Cada particularidade é geralmente descrita na seqüência de pares de estímulo-resposta. Quando organizar esta seção por particularidades, use o esboço mostrado no anexo A5.

#### 5.3.7.5. Estímulos

Alguns sistemas podem ser melhores organizados pela descrição de suas funções em termos de estímulos. Por exemplo, as funções de um sistema de aviação de aterrisagem automática devem estar organizadas em seções por perda de força, rajada de vento, súbita mudança no motor, velocidade vertical excessiva, etc. quando organizar esta seção por estímulos, use o esboço mostrado no anexo A6.

#### 5.3.7.6. Respostas

Alguns sistemas podem estar organizados pela descrição de todas as funções suportadas pela geração de respostas. Por exemplo, as funções de um sistema de pessoal podem estar organizadas dentro de seções correspondentes para todas as funções associadas com a geração de contra-cheques, lista de empregados, etc. Use o anexo A6 (em lugar de estímulo use resposta).

#### 5.3.7.7. Hierarquia Funcional

Quando nenhum dos esquemas de organização acima for útil, a total funcionalidade pode ser organizada dentro de uma hierarquia de funções organizadas por entradas comuns, saídas comuns, ou acesso de dados internos comuns. Diagrama de fluxo de dados e dicionários de dados podem ser usados para mostrar os relacionamentos entre quantidade de funções e dados. Quando organizar esta seção por hierarquia funcional, use o esboço do anexo A7.

### 5.3.8. Comentários adicionais

Quando uma nova SRS é contemplada, mais do que uma técnica organizacional mostrada no item 5.3.7.7 pode ser apropriada. Em tais casos, organize os requisitos específicos para múltiplas hierarquias feitas sob medida para necessidades específicas do sistema que está sendo especificado. Por

exemplo, veja o anexo A8 para uma organização combinando classe de usuários e particularidades. Qualquer requisito adicional pode ser colocado em separado no final da SRS.

Há muitas notações, métodos e ferramentas de suportes automatizadas disponíveis para apoiar a documentação de requisitos. Para a maior parte, a utilidade delas é uma função da organização. Por exemplo, quando organizar por modo, máquinas de condições limitadas, e gráficos podem ser úteis. Quando organizar por objeto, análise orientada a objetos pode ajudar, quando organizar por particularidades, seqüência de estímulo-resposta pode ajudar; quando organizar por hierarquia funcional, diagrama de fluxo de dados e dicionários de dados podem ajudar.

Em qualquer esboço mostrado de A1 a A8, as seções chamadas “Requisitos funcionais 1” descrevem em linguagem nativa (isto é em Inglês), em pseudocódigo, ou numa linguagem de definição de sistemas, as quatro subseções chamadas: Introdução, Entradas, Processamento, Saídas.

## 5.4. Informações de Suporte

A informação de suporte torna a SRS mais fácil de ser usada. Inclui:

a) Tabela de conteúdos
b) Índice
c) Apêndice

### 5.4.1. Tabela de Conteúdo e Índice

A tabela de conteúdo e o índice são muito importantes e seguem as práticas gerais.

### 5.4.2. Apêndices

Os apêndices nem sempre são considerados parte da especificação de requisitos e nem sempre são necessários. Eles podem incluir:

a) exemplos de formato de I/O, descrições de estudos de análise de custo, ou resultados de verificação do usuário.
b) Informações de suporte ou de coleta de dados anteriores que podem ajudar os leitores da SRS.
c) Uma descrição dos problemas a serem resolvidos pelo software.
d) Pacotes de instruções especiais para o código e o meio para encontrar segurança, exportar, armazenar dados iniciais ou outros requisitos.

Quando os apêndices são incluídos, a SRS deve explicitar se eles são ou não considerados parte dos requisitos.