UNIVERSIDADE DO SUL DE SANTA CATARINA

CAMPUS DA GRANDE FLORIANÓPOLIS

CURSO DE ESPECIALIZAÇÃO EM

ENGENHARIA DE PROJETOS DE SOFTWARE

ALEXSANDRO RAFAEL DE OLIVEIRA

ESPECIFICAÇÃO DE REQUISITOS - UM ESTUDO DE CASO NA SECRETARIA DE ESTADO DA EDUCAÇÃO DE SANTA CATARINA

Florianópolis

Março/2013

ALEXSANDRO RAFAEL DE OLIVEIRA

ESPECIFICAÇÃO DE REQUISITOS - UM ESTUDO DE CASO NA SECRETARIA DE ESTADO DA EDUCAÇÃO DE SANTA CATARINA

Monografia apresentada ao Curso de pós graduação de Engenharia de Projetos de Software da Universidade do Sul de Santa Catarina, como requisito parcial para a obtenção do título de Especialista.

Orientador: Prof(a). MEng. Vera Rejane Niedersberg Schumharcher

Florianópolis

Março/2013

ALEXSANDRO RAFAEL DE OLIVEIRA

ESPECIFICAÇÃO DE REQUISITOS - UM ESTUDO DE CASO NA SECRETARIA DE ESTADO DA EDUCAÇÃO DE SANTA CATARINA

Esta Monografia foi julgada adequada à obtenção do título de Especialista em Engenharia de Projetos de Software e aprovado em sua forma final pelo Curso de Engenharia de Projetos de Software da Universidade do Sul de Santa Catarina.

Florianópolis, Março/2013

_____

Orientador MEng. Vera Rejane Niedersberg Schuhmarcher.

_____

Avaliador Dr. Aran Bey Tcholakian Morales.

RESUMO

Este trabalho apresenta como tema o estudo da engenharia de requisitos, disciplina da engenharia de software, onde está inserida a Especificação de Requisitos de Software (ERS), que é o foco da pesquisa. Um estudo de caso foi realizado na Secretaria de Estado da Educação de Santa Catarina (SED), cujo objetivo era verificar a existência de um documento de ERS, e propor um documento modelo para uso nos processos de desenvolvimento de software caso a mesma não o possuísse. Um estudo teórico em obras publicadas foi realizado para aprofundamento do tema, e posteriormente foi realizada uma pesquisa de campo, do tipo exploratória, qualitativa, tendo como instrumento de coleta a entrevista semi-estruturada. Os procedimentos executados demonstraram que a SED não possuía um documento de ERS, onde foi possível elaborar e sugerir um documento modelo de ERS adaptado às realidades da SED.

Palavras-chave: Engenharia. Software. Requisitos. ERS.

ABSTRACT

This paper presents the study of the subject as requirements engineering, software engineering discipline, where it operates the Software Requirements Specification (SRS), which is the focus of research. A case study was conducted at the Ministry of Education of Santa Catarina (SED), whose aim was to verify the existence of a document ERS and propose a document model for use in the processes of software development if it does not possess the . A theoretical study on published works was conducted to deepen the theme, and was subsequently performed a search field, type exploratory qualitative data collection instrument as having a semi-structured interview. Procedures performed showed that the SED did not have a document of ERS, where it was possible to draw up a document and suggest ERS model adapted to the realities of the SED.

Keywords: Engineering. Software. Requirements. SRS.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

A presente pesquisa tem foco na área de estudo de Engenharia de Requisitos, uma disciplina da Engenharia de Software, conforme afirmado por Paula Filho (2001), e trata-se de um estudo de caso na Secretaria de Estado da Educação de Santa Catarina (SED), sobre a especificação de requisito para desenvolvimento de software sob demanda que ocorrem nesta organização.

Não pretende-se nesta pesquisa estudar, analisar, explorar ou descrever os métodos, técnicas ou modelos conceituais disponíveis na bibliografia relativos a engenharia de software como num todo, procurando-se limitar o escopo de estudo apenas à engenharia de requisitos, no que tange especificamente a especificação de requisitos, objeto de estudo intitulado na presente monografia.

Na próxima seção, será apresentado a situação problema, justificativas, objetos e estrutura geral do trabalho.

1.1 SITUAÇÃO PROBLEMA

O tema escolhido surgiu a partir de uma experiência profissional do acadêmico na referida organização, despertando o interesse para um estudo mais aprofundado, haja vista que ao longo de 12 (doze) meses de trabalho como terceirizado em gerência de projetos de tecnologia da informação nesta organização, vivenciou grande demanda por soluções de software.

Nesta experiência observou-se deficiência de documentação técnica, tanto para produção quanto para manutenção e evolução dos sistemas de softwares legados, o que dificultava o alinhamento de comunicação entre cliente, usuários chaves e equipe técnica de desenvolvimento no atendimento às demandas, dificultando também, o repasse de tecnologia a membros da equipe de desenvolvimento quando da troca destes profissionais.

Pelo fato dos sistemas de softwares legados não possuírem documentação satisfatória, a produção de um acervo técnico documental atualizado poderia ser contra produtiva, o que também é desaconselhado na IEEE 830:1993, também não sendo o objetivo do presente estudo, contudo, garantir que os próximos sistemas de softwares sejam

concebidos ao menos com um documento de ERS, seria uma proposta interessante para proporcionar alinhamento entre usuários chaves e equipe técnica, possibilitando assim uma interpretação mais correta de requisitos e regras de negócio, bem como, servir de base para manutenção e evolução após entrega para o uso, e para difusão e repasse de tecnologia.

O acadêmico pretende estudar a questão com mais acuidade junto a organização, sugerindo um modelo de ERS mais adequado à realidade da mesma, que poderia ser facilmente adotado, implementado e replicado nos processos de especificação, de forma que não seja abandonado com facilidade, tendo a organização a opção de uso após a conclusão deste estudo.

Isto exposto, questiona-se: a Secretaria de Estado da Educação tem um modelo prescrito de ERS para a construção de seus sistemas de softwares?

1.2 JUSTIFICATIVA

Com o advento da modernização, que cada vez mais leva empresas privadas e públicas a se automatizarem, adquirindo ou criando sistemas de software por diversos motivos, seja por questões de eficiência, controle ou mesmo pela transparência, o presente trabalho pretende colaborar com a sociedade através do estudo de caso na referida empresa pública, sobre como se dá a especificação de requisitos de software frente aos demandantes desses produtos.

À SED o estudo proporcionará uma visão interna de como estão estruturados os processos que levam à especificação de requisitos de software, e isto poderá proporcionar às áreas demandantes uma visão também mais clara sobre o que está envolvido nestes processos, e à equipe de desenvolvedores uma análise crítica de seus procedimentos, que consequentemente, esta visão clarificada em torno de conceitos acadêmicos e dos processos práticos, proporcionarão reflexões para alguma melhora de qualidade no produto final.

À área de Engenharia de Software, e a própria comunidade acadêmica envolvida, o presente estudo trará como benefício a observação de como a prática de ERS é adotada na empresa pública em detrimento de modelos científicos prescritos.

Ao acadêmico, que é Bacharel em Administração e Técnico em Eletrônica, o estudo proporcionará um crescimento profissional, podendo se aprofundar na disciplina de

Engenharia de Requisitos, somando-a como competência técnica por ocasião do estudo e da prática observada.

1.3 OBJETIVOS

Nesta seção serão apresentados os objetivos propostos para este trabalho, que nortearão as atividades que serão apresentadas.

1.3.1 Objetivo geral

Pretende-se prescrever um modelo de documento especificação de requisitos de software adaptado às necessidades da Secretaria de Estado da Educação.

1.3.2 Objetivos específicos

- Explorar modelos científicos de ERS para compor sugestões a um modelo direcionado à organização.
- Identificar indivíduos chaves para realização de entrevistas;
- Descrever de forma sucinta a organização de forma genérica, seu negócio, sua estrutura física e funcional, e os sistemas de software que a organização utiliza;
- Identificar a existência e o uso de um modelo prescrito de ERS;
- Descrever o processo de desenvolvimento de software na organização;

1.4 ESTRUTURA DO TRABALHO

Este estudo está organizado em 5 capítulos.

No primeiro capítulo constam a introdução, situação problema, justificativas e os objetivos do trabalho. No capítulo 2 constam a fundamentação teórica, organizada do todo para o específico. No capítulo 3 constam os métodos, com a caracterização da empresa e da pesquisa, e os procedimentos executados. No capítulo 4 constam os resultados obtidos a partir

da aplicação dos métodos. No capítulo 5 constam as considerações finais, divididas entre conclusão com análise do autor, e recomendações para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

O marco teórico do presente estudo está basicamente delimitado pelo seu título, que trata da especificação de requisitos, e inicia-se com a exploração teórica do tema, do todo para o específico, com o intuito de nortear o presente estudo.

A área de estudo é a Engenharia de Software, na qual está inserida a Engenharia de Requisitos, que é a disciplina foco que trata sobre a especificação de requisitos de software.

2.1 CONTEXTUALIZAÇÃO GERAL

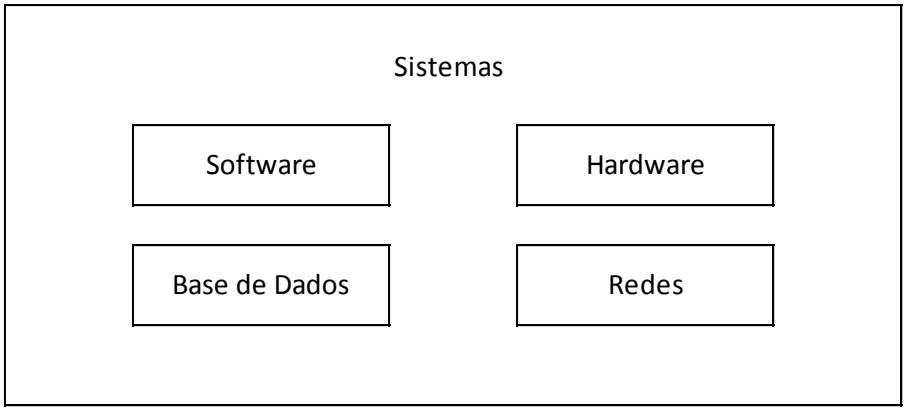
Como ponto de partida do estudo, inicia-se a abordagem com um conceito bem simples, o de sistema, que pode ser aplicado à forma mais abstrata ou concreta de sistema, onde uma definição útil é que "um sistema é uma coleção significativa de componentes inter-relacionados, que trabalham em conjunto para atingir um objetivo", conforme expressou Sommerville (2003, p. 18).

Em termos de sistema de tecnologia, destaca-se duas subseções, a de sistemas de informática e de sistemas de software, que passar a ser conceituadas:

a) Sistemas de Informática

Levando em consideração a definição genérica de sistema emprestada por Sommerville, Paula Filho (2001, p. 4) traz em sua obra algumas definições, na qual destaca que os sistemas de informática são a "disposição das partes ou dos elementos de um todo, coordenados entre si e que funcionam como estrutura organizada", e demonstra ainda o seguinte quadro explicativo conforme a Ilustração 1 – Sistemas de informática e suas partes.

Ilustração 1 – Sistemas de informática e suas partes



Fonte: Adaptado de Paula Filho (2001, p. 4)

Sommerville (2003) complementa que os sistemas não são entidades independentes, mas existem em um ambiente, e que esse ambiente afeta o funcionamento e desempenho do sistema, e que às vezes o ambiente pode ser considerado um sistema em si mesmo, mas, em geral, ele consiste em uma série de outros sistemas que interagem entre si.

b) Sistemas de software

Considerando as definições anteriores de sistemas, o software também forma um sistema, conforme Sommerville (2003, p. 5) completa:

> um sistema de software consiste, usualmente, em uma série de programas separados, arquivos de configuração que são utilizados para configurar esses programas, documentação do sistema, que descreve a estrutura desse sistema, e documentação do usuário, que explica como utilizar o sistema e, no caso dos produtos de software, sites Web para os usuários fazerem o download das informações recentes sobre o produto.

Vê-se já nesta definição de Sommerville uma abordagem sobre parte de documentação, destacando importância deste item no contexto de sistema, e que demonstra a relevância de um documento de ERS, produto da engenharia de requisitos objeto do estudo deste trabalho.

Assim sendo, passa-se a apresentar os conceitos de software, engenharia de sistemas e engenharia de software, a fim de proporcionar o estudo devido da engenharia de requisitos.

2.1.1 Conceito de software

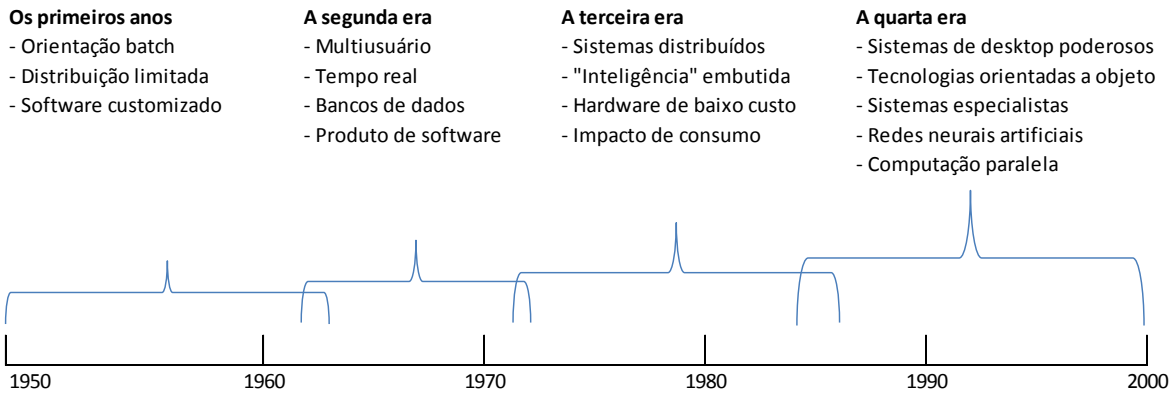
Já Paula Filho (2001, p. 5) define que o "software é a parte programável de um sistema de informática. Ele é um elemento central: realiza estruturas complexas e flexíveis que trazem funções, utilidade e valor ao sistema".

Sommerville (2003, p. 5) adiciona ainda que software não é apenas o programa "mas também toda a **documentação associada** e os dados de configuração necessários para fazer com que esses programas operem corretamente". (grifo do autor)

É oportuno mencionar, no contexto da definição do software, onde Peters e Pedricz (2001, p. V) afirmam no prefácio de seu livro, que "a cada ano são frequentes as novas versões de produtos de software existentes, bem como versões de novos produtos e tecnologias de software".

Neste sentido temporal, Pressman (1995) já descrevia a evolução do software dentro do contexto das áreas de aplicação dos sistemas baseados em computador conforme Ilustração 2 – A evolução do software.

Ilustração 2 – A evolução do software



Fonte: Adaptado de Pressman (1995, p. 5)

2.1.2 Engenharia de sistemas

Somerville (2003, p. 20) inicia sua literatura situando a compreensão de sistemas pela engenharia de sistema e de software.

> a engenharia de sistemas é a atividade de especificar, projetar, implementar, validar, implantar e manter os sistemas *como um todo*. Os engenheiros de sistemas não se ocupam apenas com o software, mas com as interações de software, hardware e sistemas com os usuários e seu ambiente. [...] Os engenheiros de software necessitam de uma compreensão sobre a engenharia de sistemas, porque os problemas de engenharia de software são, frequentemente, o resultado de decisões da engenharia de sistema.

Observa-se que Somerville remete reflexão sobre o ato de especificar, e considerando sua afirmação, cabe-se destacar a definição de Engenharia de Software, que nos aproxima mais do objeto de estudo.

2.1.3 Engenharia de software

No que diz respeito a Engenharia de Software, Paula Filho (2001) inicia a compreensão do significado a partir da combinação das definições estabelecidas no quadro da Ilustração 3 – Definições.

Ilustração 3 – Definições

Informática	Ciência que visa ao tratamento da informação através do uso de equipamentos e procedimentos da área do processamento de dados.
Processamento de Dados	Tratamento dos dados por meio de máquinas, com o fim de obter resultado da informação representada pelos dados.
Ciência	Conjunto organizado de conhecimentos relativos a um determinado objeto, especialmente os obtidos mediante a observação, a experiência dos fatos e um método próprio.
Engenharia	Arte de aplicar conhecimentos científicos e empíricos e certas habilitações específicas às criação de estruturas, dispositivos e processos que se utilizam para converter recursos naturais em formas adequadas ao atendimentos das necessidades humanas.

Fonte: Adaptado do Dicionário Aurélio Eletrônico V.2.0. apud Paula Filho (2001, p. 5)

Em resumo, Paula Filho (2001, p. 4) sintetiza.

> a Engenharia de Software não se confunde com a Ciência da Computação, e nem é uma disciplina desta, tal como a Engenharia Metalúrgica não é uma disciplina da Física dos Metais, nem a Engenharia Elétrica é uma disciplina da Física da Eletricidade. Como toda engenharia, a Engenharia de Software usa resultados da Ciência, e fornece problemas para estudo desta; mas são vocações profissionais completamente distintas, tão distintas quanto as vocações do engenheiro e do físico, do metódico e do biólogo, do político e do cientista político.

Sommerville (2003, p. 5) complementa a abordagem, e mais uma vez reforça a preocupação à especificação:

> engenharia de software é uma disciplina da engenharia, segundo a qual os engenheiros de software utilizam métodos e teorias da ciência da computação, e aplicam tudo isso de modo eficaz em relação aos custos, a fim de solucionar problemas difíceis, [...] e se ocupam de todos os aspectos da produção, desde os estágios iniciais de especificação, até a manutenção, depois que o produto entrou em operação.

Pressman (1995, p. 31) por sua vez já afirmava que a "engenharia de software é um rebento da engenharia de sistema e de hardware, e que ela abrange três elementos fundamentais: métodos, ferramentas e procedimentos", como explica.

> Os *métodos* de engenharia de software proporcionam os detalhes de "como fazer" para construir o software. Os métodos envolvem um amplo conjunto de tarefas que incluem: planejamento e estimativa de projeto, análise de requisitos de software e de sistemas, projeto da estrutura de dados, arquitetura de programas e algoritmo de processamento, codificação, teste e manutenção. [...]
> As *ferramentas* de engenharia de software proporcionam um apoio automatizado ou semi-automatizado aos métodos, e que existem ferramentas que sustentam cada um dos métodos anotados acima, que são as ferramentas CASE – *Computer-Aided Software Engineering*.
> Os *procedimentos* da engenharia de software constituem o elo de ligação que mantém juntos os métodos e as ferramentas e possibilita o desenvolvimento racional e oportuno do software de computador. Os procedimentos definem a sequência em que os métodos serão aplicados [...]. (PRESSMAN, 1995, p. 31)

Peters e Pedricz (2001, p. 2), completam que "o conjunto total de atividades necessárias para transformar os requisitos de um usuário em software é denominado um processo de engenharia de software".

Estas definições sugerem uma abordagem ao processo de engenharia de software, do qual trata a próxima seção.

2.1.3.1. Processos de engenharia de software

Paula Filho (2001, p. 7) menciona que "o desenvolvimento de software é feito dentro de um projeto, e um projeto representa a execução de um processo". Completa ainda Idib., p. 17, que "não se deve confundir um processo com o respectivo produto ou com a execução do processo através de um projeto".

Segundo Sommerville (2003, p. 7), um "processo de software é um conjunto de atividades e resultados associados que geram um produto de software, [...] e menciona ainda que há quatro processos fundamentais comuns a um processo de software":

> *Especificação do software* A funcionalidade do software e as restrições em sua operação devem ser definidas.
> *Desenvolvimento de software* O software deve ser produzido de modo que atenda as suas especificações
> *Validação do software* O software tem de ser validado para garantir que ele faz o que o cliente deseja,
> *Evolução do software* O software deve evoluir para atender às necessidades mutáveis do cliente

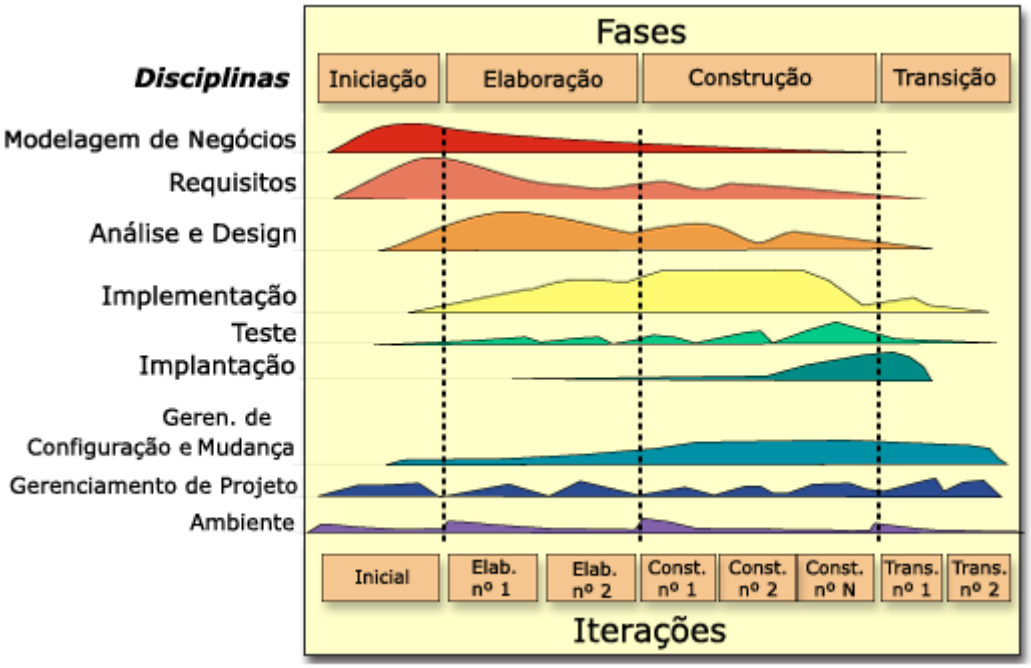
Para Peters e Pedricz (2001, p. 29), o "processo de software é uma sequência de etapas com feedback que resultam na produção e na evolução do software".

De forma um pouco diferente a Sommerville, Paula Filho, op. cit., p. 17, menciona que "em Engenharia de Software, processos podem ser definidos para atividades como desenvolvimento, manutenção, aquisição e contratação de software, mas completa que o ponto de partida para a arquitetura de um processo é a escolha de um modelo de ciclo de vida".

Neste sentido, em relação ao ciclo de vida, e em se tratando de processos de software, apesar de não ser o foco do presente estudo, existem dois grandes modelos de processo de desenvolvimento de software, conforme Pressman (2006) apud Souza (2006, p.19) declaram: os modelos prescritivos de processo, ou tradicionais como o cascata, espiral e RUP, entre outros, e os modelos ágeis (Scrum, XP, Crystal, etc).

No que tange a modelos prescritos, que têm forte ênfase em documentação (e pelo fato deste estudo se ater ao documento de ERS), destacar-se-á ao longo da fundamentação alguns aspectos do *Rational Unified Process* (RUP), que é um modelo de processo proprietário da Rational Software Corporation, que introduz um modelo de processo de desenvolvimento de software que tem duas dimensões: as fases e as disciplina, sendo que as fases são executadas no tempo ao longo do ciclo de vida do processo de desenvolvimento (iniciação, elaboração, construção e transição); e disciplinas que representam atividades lógicas executadas ao longo deste tempo em fluxos de trabalho (modelagem de negócios, requisitos, análise e design, implementação, testes, implantação, gerenciamento de configuração e mudança, gerenciamento de projeto e ambiente). A Ilustração 4 – Fases e disciplinas do RUP sintetiza essa abordagem.

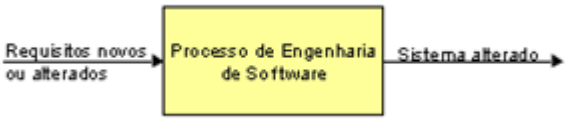
Ilustração 4 – Fases e disciplinas do RUP



Fonte: Adaptado do RUP (RATIONAL, 2002)

O RUP (RATIONAL, 2002) destaca ainda que o processo de engenharia de software é o processo de desenvolvimento de um sistema a partir dos requisitos, sejam eles novos ou alterados, como resumo a Ilustração 5 – Processo resumido do RUP.

Ilustração 5 – Processo resumido do RUP



Fonte: Adaptado do RUP (RATIONAL, 2002)

Cabe portanto, conceituar na próxima seção o ciclo de vida de software mencionado nas passagens citadas.

2.1.3.2. Ciclo de vida de software

Peters e Pedricz (2001, p. 36) relatam que "um ciclo de vida de software (CVS) é o período de tempo que se inicia com um conceito para um produto de software e acaba sempre que o software deixa de estar disponível para utilização".

Paula Filho (2001, p. 6) menciona que a "Engenharia de Software se preocupa com o software como produto [...], e como todo produto industrial, o software tem um ciclo de vida, dividido em fases", conforme descreveu o autor:

> - ele é concebido a parir da percepção de uma necessidade
> - é desenvolvido, transformando-se em um conjunto de itens entregue a um cliente
> - entra em operação, sedo usado dentro de algum processo de negócio, e sujeito a atividades de manutenção, quando necessário
> - é retirado de operação, ao final da vida útil.

O mesmo autor, Paula Filho, apresenta ainda um modelo, a Ilustração 6 – Esquema simplificado do ciclo de vida do software, que destaca as fases do clico de vida com suas divisões e subdivisões.

Ilustração 6 – Esquema simplificado do ciclo de vida do software

<table>
<tr><td rowspan="10">Ciclo de Vida</td><td colspan="4">Percepção da necessidade</td></tr>
<tr><td rowspan="8">Desenvolvimento</td><td colspan="3">Concepção</td></tr>
<tr><td colspan="3">Elaboração</td></tr>
<tr><td rowspan="5">Construção</td><td colspan="2">Desenho inicial</td></tr>
<tr><td rowspan="3">Liberação</td><td>Desenho detalhado</td></tr>
<tr><td>Codificação</td></tr>
<tr><td>Teste de unidade</td></tr>
<tr><td colspan="2">Teste alfa</td></tr>
<tr><td colspan="3">Transição</td></tr>
<tr><td colspan="4">Operação</td></tr>
<tr><td colspan="4">Retirada</td></tr>
</table>

Fonte: Adaptado de Paula Filho (2001, p. 7)

Peters e Pedricz (2001, p. 37) relatam ainda que "um ciclo de vida de software torna-se específico como resultado da escolha de um determinado modelo de ciclo de vida e das atividades de mapeamento de projeto para aquele modelo".

É importante destacar, apesar de não ser objeto do presente estudo, a existência da NBR ISO/IEC 12207 (ABNT, 2008) que introduz padrões de Processos de Ciclo de Vida de *Software*, como fonte de pesquisa mais aprofundada.

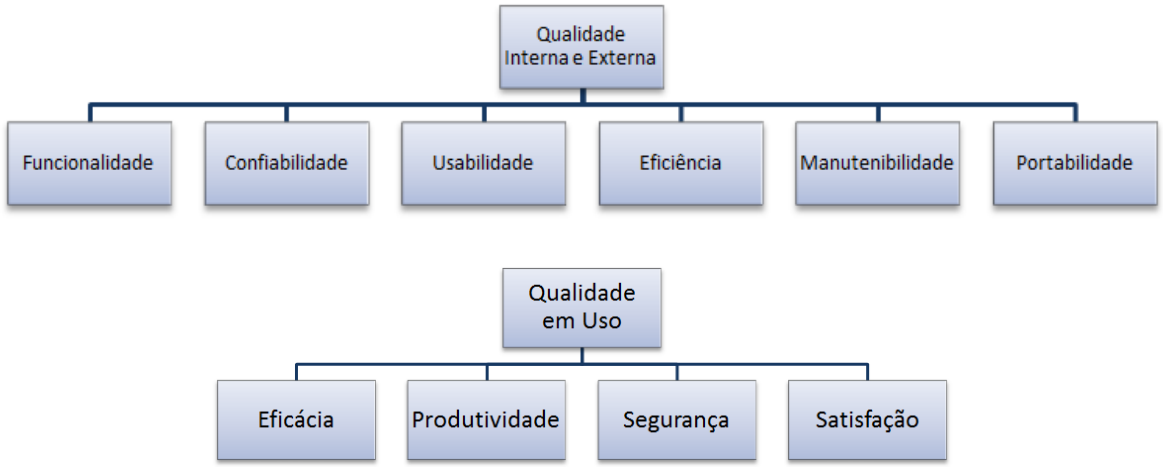
Abordada brevemente a questão de ciclo de vida, como este estudo concentra-se na especificação de requisitos, na próxima seção será abordada de forma destacada a qualidade software, que estão atreladas a questões de requisitos, processos e ciclo de vida.

2.1.3.3. Qualidade em software

Tomar-se-á como referência nesta seção a NBR ISO/IEC 9126-1 (ABNT, 2003), que padroniza a qualidade para o produto de software.

Segundo a NBR ISO/IEC 9126-1 (ABNT, 2003), o padrão de qualidade para o produto de software é composto de duas partes: a) qualidade interna e externa, e b) qualidade de uso, como demonstrada na Ilustração 7 – Dimensões da qualidade, sendo a primeira parte dividida em 6 (seis) características que se dividem em subcaracterísticas, e a segunda parte se divide em 4 (quatro) características, normatizando assim um modelo de qualidade e sua métricas.

Ilustração 7 – Dimensões da qualidade



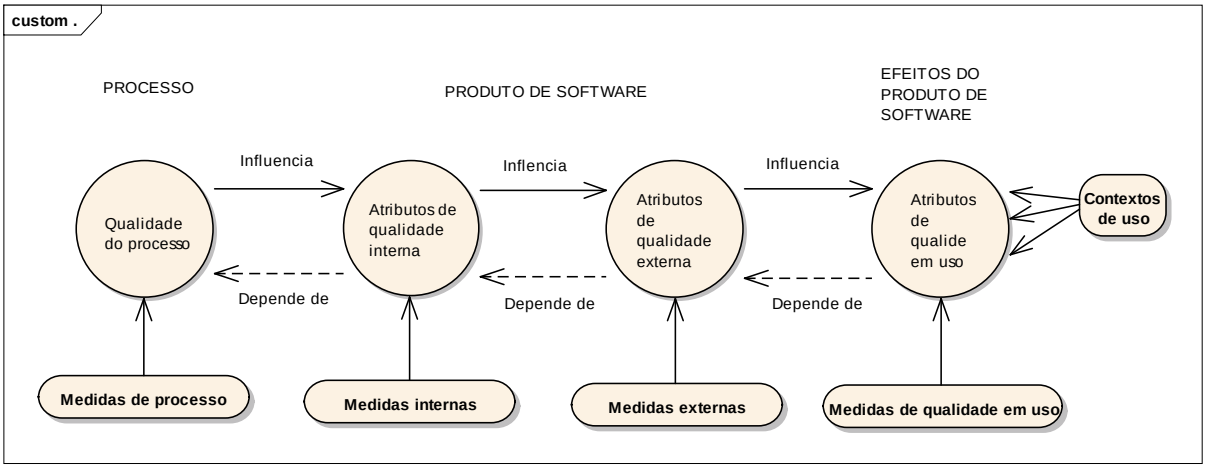
Fonte: Adaptado de NBR ISO/IEC 9126-1 (ABNT, 2003)

As características e subcaracterísticas mencionadas serão exploradas numa abordagem complementar à ERS constante da seção 2.2.1.6.

Tal ilustração sugere, conforme NBR ISO/IEC 9126-1 (ABNT, 2003), que atributos internos adequados são pré-requisitos para atingir o comportamento externo requerido, e o comportamento externo adequado, é um pré requisito para obter-se qualidade em uso.

A mesma norma menciona influências e dependências dentro do ciclo de vida do software, como demonstrado na Ilustração 8 – Qualidade no ciclo de vida,

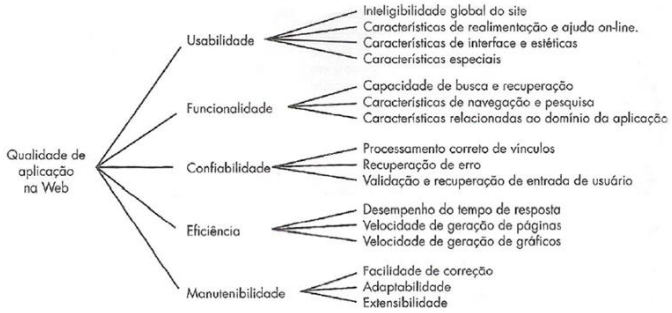
Ilustração 8 – Qualidade no ciclo de vida



Fonte: Adaptado de NBR ISO/IEC 9126-1 (ABNT, 2003)

De forma um pouco diferente, destacando aplicações Web, Olsima, Pressman (2006) apud Souza (2006, p. 36), destaca 5 característica de qualidade que são coincidentes à NBR ISO/IEC 9126-1 (ABNT, 2003), conforme demonstra a Ilustração 9 - Qualidade de aplicação web.

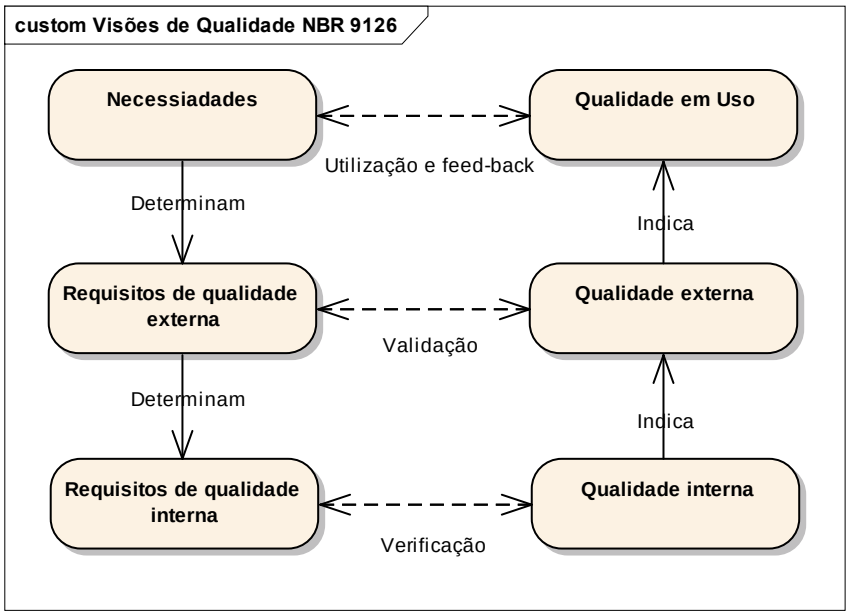
Ilustração 9 – Qualidade de aplicação web



Fonte: Adaptado de Olsina, Pressman (2006) apud Souza (2006, p. 36)

A NBR ISO/IEC 9126-1 (ABNT, 2003) apresenta ainda a Ilustração 10 – Visões de qualidade do software, demonstrando como a qualidade (interna, externa e em uso) é indicada e determinada através de um processo de utilização, feed-back, validação e verificação sobre as necessidades e os requisitos.

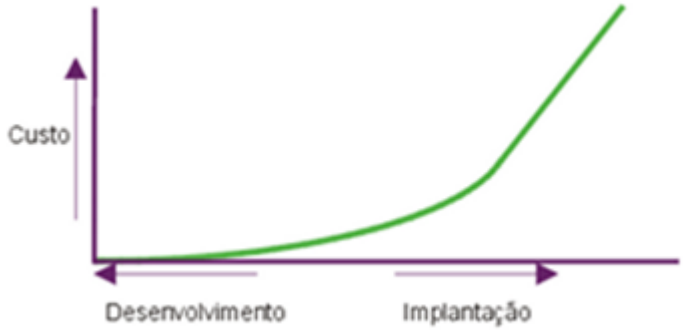
Ilustração 10 – Visões de qualidade do software



Fonte: Adaptado de NBR ISO/IEC 9126-1 (ABNT, 2003)

O RUP (RATIONAL, 2002) incrementa uma abordagem ao gerenciamento da qualidade, que deve ser contínuamente monitorada e controlada, pois afirma que “o custo de localização e solução de problemas de software ficam de 10 a 1.000 vezes mais caros, se realizados após a implantação, e que por isso, o gerenciamento da qualidade durante o ciclo de vida do projeto torna-se essencial para atingir os objetivos certos nos momentos certos.” A Ilustração 11 – Qualidade continuada, demosntra graficamente a expressão citada.

Ilustração 11 – Qualidade continuada



Fonte: Rational (2002)

Vale destacar também um registro ao CMMI-DEV e MPS.Br, que são modelos de referência para melhoria de qualidade de software através de processos, que referenciam diversos normas e padrões de qualidade em seus conteúdos, mas que não são escopo deste trabalho, mas que vale a pena a observação destas abordagens, que podem merecer um estudo à parte quanto ao tema for especificamente qualidade em software.

Abordado o tema de qualidade e alguns de seus relacionamentos, e encerrando a subseção que trata da Engenharia de Software e da seção de contextualização geral, destina-se no próximo item uma seção exclusiva para tratar sobre a Engenharia de Requisito, que é o foco deste estudo conforme já mencionado no início do capítulo.

2.2 ENGENHARIA DE REQUISITOS

Paula Filho (2001) explica que engenharia de requisitos é uma disciplina da disciplina da Engenharia de Software, e afirma que o “conjunto de técnicas empregadas para

levantar, detalhar, documentar e validar os requisitos de um produto forma a Engenharia de Requisitos".

Peters e Pedricz (2001, p. 101) completam que dentro de "um processo de software, a engenharia de requisitos é a primeira atividade importante após a conclusão de um relatório de necessidades resultante de um processo de pré-desenvolvimento".

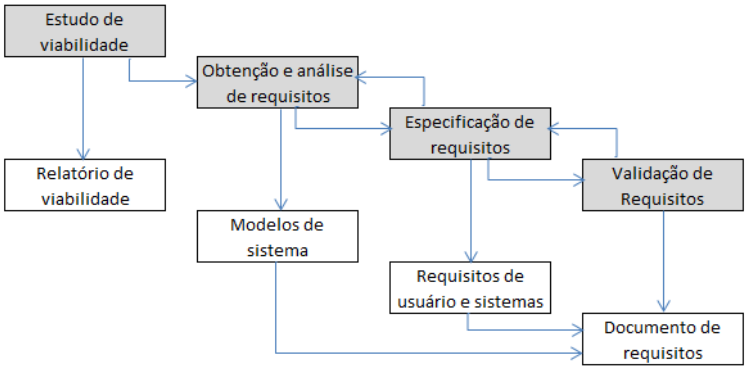
Os mesmos autores, Peters e Pedricz (Op. Cit. p. 101) informam ainda que a "engenharia de requisitos é definida em função de suas atividades principais: entendimento dos problemas (descritos num relatório de necessidades), determinação de soluções e especificação da solução [...]".

Sommerville (2003, p. 103) define que a "engenharia de requisitos é um processo que envolve todas as atividades exigidas para criar e manter o documento de requisitos de sistema, e que existem quatro atividades genéricas neste processo, o estudo de viabilidade, a obtenção e análise de requisitos, especificação e produção da documentação de requisitos, e finalmente a validação de requisitos."

2.2.1 Processo de engenharia de requisitos

Conforme Sommerville (2003) afirmou anteriormente, a engenharia de requisitos é um processo, e está dividida segundo este autor, em quatro partes, que estão conectadas conforme a Ilustração 12 – Processo de engenharia de requisitos.

Ilustração 12 – Processo de engenharia de requisitos



Fonte: Adaptado de Sommerville (2003, p. 103)

Segundo o RUP (RATIONAL, 2002), processos podem ser assim definidos:

> Um processo é um conjunto de passos parcialmente ordenados com a intenção de atingir uma meta. Em engenharia de software, a meta é criar um software ou aperfeiçoar um existente; em engenharia de processos, a meta é desenvolver ou aperfeiçoar um processo. No RUP, eles são organizados em um conjunto de disciplinas para posteriormente definirem os fluxos de trabalho e outros elementos do processo.

O RUP (RATIONAL, 2002) apresenta seu esquema de processo conforme a Ilustração 13 – Processo de engenharia de software no RUP.

Ilustração 13 – Processo de engenharia de requisitos

Fonte: Adaptado do RUP (RATIONAL, 2002)

Destaca-se na abordagem do RUP (RATIONAL, 2002), dois processos em especial, que é a modelagem de negócio pois precede os requisitos, e o gerenciamento de configuração e mudança, que controla as alteração que impactam no produto e nos requisitos, merecendo um detalhamento mais adiante nesta seção.

Já Leite (1988) apud Silva (2006, p. 27), defende que “o processo de definição de requisitos pode ser definido, resumidamente, por três atividades: elicitação, modelagem e análise”.

Peters e Pedricz (2001) mencionam que os principais produtos de um processo de requisitos satisfatórios são os seguintes:

> **Especificação de requisitos de software (ERS) completa**. Uma descrição de um sistema (suas funções, seu comportamento, seu desempenho, suas interfaces internas e externa e seus atributos de qualidade).
> **Plano de garantia de qualidade**. Uma indicação da portabilidade, eficiência, confiabilidade, critérios de validação e verificação, custos e critérios de aceitação a serem seguidos pelas equipes de projeto.
> (PERTERS E PEDRICZ, 2001, p. 102)

Tomando as divisões apresentadas por Sommerville (2003), combinado com alguns aspectos abordados no RUP (RATIONAL, 2002) e autores citados, a seção seguinte foi organizada de forma a classificar o processo envolvido na engenharia de requisitos.

2.2.1.1. Estudo de viabilidade

Sommerville (2003, p. 103) menciona que "para todo sistema novo, um processo de engenharia de requisitos deve iniciar pelo estudo de viabilidade, que é uma descrição geral do sistema e como ele será utilizado. O resultado deste estudo deve ser um relatório de viabilidade que aponta se é viável ou não o desenvolvimento do sistema, diante de vários aspectos que deverão ser considerados".

2.2.1.2. Modelagem de negócio

O RUP (RATIONAL, 2002) considera que a etapa inicial do processo de desenvolvimento é a modelagem de negócio, da qual derivarão os requisitos.

A modelagem de negócios segundo ainda segundo o RUP (RATIONAL, 2002) serve para comprender a definição e o escopo do problema, onde são identificados os envolvidos e suas principais solicitações, tendo como artefatos organogramas, diagramas e casos de uso.

Os casos de uso são amplamente manipulados no RUP, e utiliza-se da linguagem de modelagem unificada, UML, que tem expressão como notação padronizada para desenho de atividades, tarefas, eventos, fluxos e comportamentos conhecidos dos analistas, desenvolvedores, e em alguns momentos de até fácil interpretação pelos clientes se minimamente orientados.

Outra técnica para modelar pode ser executada a partir do BPMN - *Business Process Modeling Notation (*modelagem de processos de negócio), lançado pela organização BPMI (Business Process Management Initiative), que tem parceria com a OMG (Object Management Group), e tem como finalidade representar diagramas de processos de negócio, bem como ser de fácil compreensão por todas as pessoas do negócio, desde analistas do negócio até os técnicos responsáveis pela sua automatização", conforme relatou Fernandes (2010).

A BPMN encontra-se embutida na maioria das ferramentas que apoiam o desenvolvimento de software, conforme abordou White (2009) apud Fernandes (2010, p. 16), e um suas notações encontra-se em síntese na Ilustração 14 – Síntese dos elementos gráficos do BPMN, sendo complementados pelo no Anexo I – Notações BPMN 2.0.

Ilustração 14 – Síntese dos elementos gráficos do BPMN

Elemento	Descrição	Notação
Evento	Um "Evento" é representado por um círculo e é algo que acontece durante um processo do negócio. Há três tipos de Eventos: Início (*Start)*, Intermediário (*Intermediate)*, e Fim (*End),* ilustrados a direita, respectivamente.	
Atividade	Uma "Atividade" é representada por um retângulo de canto arredondado. Um Sub-Processo é distinguido por uma pequena cruz no centro inferior da figura.	
Decisão	A "Decisão" é representada pela forma de losango e usada para controlar a divergência e a convergência do fluxo. Assim, determinará decisões tradicionais, como juntar ou dividir trajetos. Os marcadores internos indicarão o tipo de controle de comportamento.	
Fluxo de Seqüência	Um "Fluxo de Seqüência" é representado por uma seta com uma linha contínua e é usado para mostrar a ordem (a seqüência) com que as atividades serão executadas em um processo.	
Fluxo da Mensagem	Um "Fluxo da Mensagem" é representado por uma linha tracejada e usado para mostrar o fluxo das mensagens entre dois participantes.	
Associação	Uma "Associação" é representada por uma linha pontilhada é usada para associar dados, texto, e outros artefatos com os objetos do fluxo.	
Pool	O "*Pool"* representa um participante em um processo. Ele representa também um recipiente que separa um conjunto de atividades de outros "*Pools".*	
Subdivisão	Representa uma "Subdivisão" dentro de um "*Pool"* e se estenderá no comprimento inteiro do "*Pool"*, verticalmente ou horizontalmente. São usadas para organizar e categorizar as atividades.	
Objetos de dados	Os "Objetos de dados" são mecanismos para mostrar como os dados são requeridos ou produzidos por atividades. São conectados às atividades com as "Associações".	
Grupo	Um "Grupo" é representado por um retângulo de canto arredondado extraído com uma linha tracejada. Agrupar pode ser usado para finalidades da documentação ou da análise, mas não afeta o Fluxo de Seqüência.	
Anotação	As anotações são mecanismos para que um modelador forneça a informação do texto adicional para o leitor de um diagrama de BPMN.	

Fonte: Adaptado BPMN (2005) apud Fernandes (2010, p. 39)

O RUP (RATIONAL, 2002) reforça ainda que a modelagem de negócio tem como finalidade entender os problemas atuais e as possibilidades de melhoria, bem como assegurar que as partes envolvidas tenham a mesma compreensão do negócio, e esta visão se integra a abordagem apresentada por Sommerville.

2.2.1.3. Levantamento e análise de requisitos

Sommerville (2003, p. 104), já defende e define que:

> o processo posterior ao estudo de viabilidade, é o levantamento e análise de requisitos, que consiste no trabalho conjunto e sistêmico dos membros da equipe técnica de desenvolvimento com o cliente e usuários finais do sistema, para descobrir mais informações sobre o domínio da aplicação, que serviços o sistemas deve fornecer, o desempenho exigido do sistema, as restrições de hardware e assim por diante.

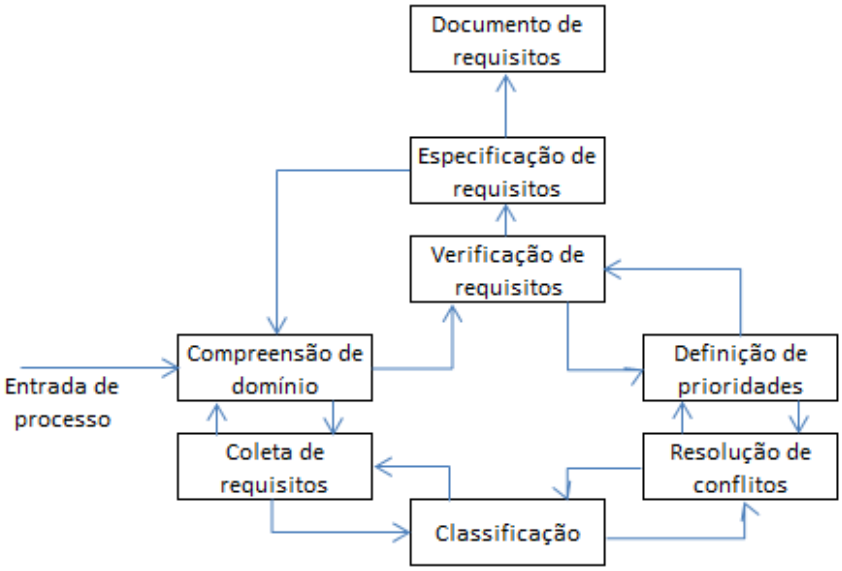
Paula Filho (2001, p. 37) complementa Sommerville afirmando que "enquanto o Levantamento de Requisitos focaliza a visão que o cliente e usuário têm dos requisitos de um produto, a Análise dos Requisitos focaliza a visão dos desenvolvedores".

Paula Filho (2001, p. 53) frisa que os "requisitos devem ser levantados pela equipe do projeto, em conjunto com representante do cliente, usuários chaves e outros especialistas da área de aplicação".

Sommerville (2003, p. 106) apresenta um modelo genérico do processo em questão na Ilustração 15 – Processo de levantamento e análise de requisitos:

Sommerville, Op. Cit. P.106, informa ainda que existem algumas "técnicas para realizar o levantamento e análise de requisitos, mas que não existe uma abordagem perfeita e universalmente aplicável, e que normalmente é preciso utilizar várias dessas abordagens para desenvolver uma compreensão completa dos requisitos, e destaca algumas, como o levantamento orientado a pontos de vista, cenários, etnografia, análise estruturada e prototipação".

Ilustração 15 – Processo de levantamento e análise de requisitos



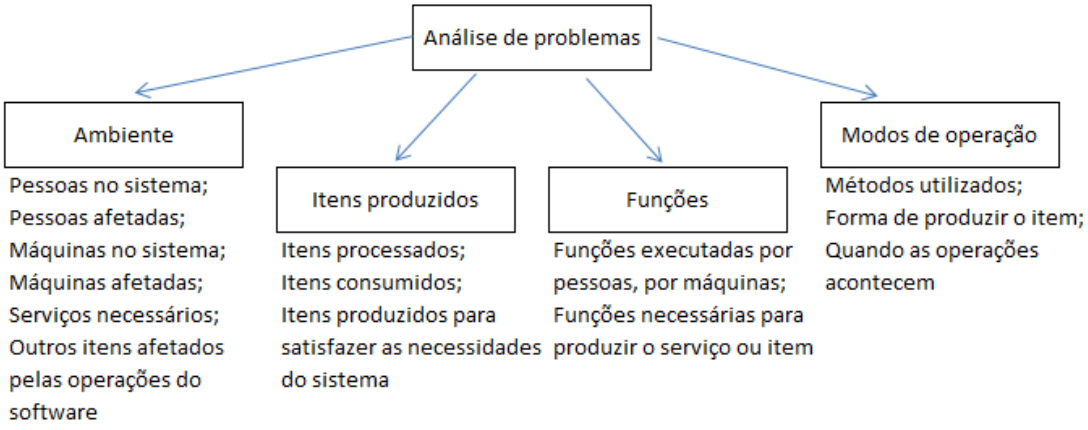
Fonte: Adaptado de Sommerville (2003, p. 106)

Paula Filho (2001) destaca novamente sobre a distinção entre o levantamento e a análise de requisitos informando que:

> - o Levantamento de requisitos visa a captura das necessidade dos usuário em relação ao produto, expressas na linguagem desses usuários;
> - a Análise de Requisitos confecciona um modelo conceitual do produto, que é usado para validade os requisitos levantados e para planejar o desenvolvimento posterior. (PAULA FILHO, 2001, p. 37)

Peters e Pedricz (2001, p. 103) tratam a "análise de requisito como a análise de problema, e que a análise de problema define o contexto para possíveis soluções de software para um problema. Ela fornece um ponto de partida necessário para o desenvolvimento das especificações de requisitos de software, e que a especificação de requisitos de software tem como objetivo principal descrever os objetos, as funções e os estados relacionados a um problema", e apresenta a Ilustração 16 – Componentes da análise de problema, para demostrar sua narrativa.

Ilustração 16 – Componentes da análise de problemas



Fonte: Adaptado de Peters e Pedricz (2001, p. 103)

2.2.1.4. Validação de requisitos

Sommerville (2003, p. 115) define que a "validação de requisitos se ocupa de mostrar que os requisitos realmente definem o sistema que o cliente deseja, e que embora ela seja parecida com as atividades do processo de análise de requisitos, ele difere pois sua função é elaborar um documento final de requisitos".

Adicionalmente a abordagem tomada de Sommerville (2003), incrementasse dois processos paralelos que devem ser observados, o gerenciamento de requisitos e a rastreabilidade, como segue apresentado nas próximas seções.

2.2.1.5. Gerenciamento de requisitos

Sommerville (2003, p. 118) define que "o gerenciamento de requisitos é o processo de compreender e controlar as mudanças nos requisitos de sistema, e é realizado em conjunto com outros processos da engenharia de requisitos", conforme abordado no parágrafo anterior.

O RUP (RATIONAL, 2002) aborda que o gerenciamento de requisitos é um modelo sistemático para encontrar, documentar, organizar e rastrear os requisitos variáveis de

um sistema, bem como estabelecer um acordo entre cliente e a equipe do projeto na manutenção desses requisitos variáveis.

Paula Filho (2001, p. 9) adiciona que “uma boa engenharia de requisito reduz a instabilidade dos requisitos, ajudando a obter os requisitos corretos em um estágio anterior ao desenvolvimento, contudo, a engenharia de requisitos está sujeita a limitações humanas, e, mesmo que o levantamento seja perfeito, podem ocorrer alterações nos requisitos por causas externas aos projetos”.

Completa ainda Paula Filho (2001) que a gestão de requisitos é disciplina da engenharia de software, e procura manter sob controle o conjunto de requisitos de um produto, mesmo diante de algumas alterações inevitáveis.

2.2.1.6. Rastreabilidade de requisitos

O RUP (RATIONAL, 2002) declara que a rastreabilidade é a capacidade de rastrear um elemento do projeto a outros elementos correlatos, especialmente aqueles relacionados a requisitos.

Paula Filho (2001, p. 58) infere que uma especificação de requisitos é rastreável se permitir a fácil determinação dos antecedentes e consequências de todos os requisitos, e que dois tipos de rastreabilidade devem ser observadas:

> - **Rastreabilidade para trás** – deve ser possível localizar a origem de cada requisito, Deve-se sempre saber porque existe cada requisito, e quem ou o que o originou. Isso é importante para avaliar o impacto da mudança daquele requisito, e dirimir dúvidas de interpretação.
> - **Rastreabilidade para frente** – deve ser possível localizar quais os resultados do desenvolvimento que serão afetados para cada requisito. Isso é importante para garantir que os itens de análise, desenho, código e testes abranjam todos os requisito, e para localizar os itens que serão afetados por uma mudança de requisitos.

Peters e Pedricz (2001, p. 151) corroboram com o mesmo pensamento, informando que “além de rastrear os requisitos para os recursos de software planejados, também é necessário que os requisitos facilitem o rastreamento de conexões entre componentes e especificação de software, e que o uso de uma numeração completa e cuidadosa é fundamental para o rastreamento ascendente e descendente”.

Por fim, Peters e Pedricz (2001) completam que a "*rastreabilidade* é um atributo para uma ERS de boa qualidade incluída pelo padrão IEEE 830."

2.2.1.7. Especificação de Requisitos de Software - ERS

Nesta seção serão abordados conceitos relativos especificamente à ERS e o documento de requisitos.

a) Conceituando requisitos de software

Sommerville (2003, p. 82) informa que o "termo *requisito* não é utilizado pela indústria do software de modo consistente, e que em alguns casos um requisito é visto como uma declaração abstrata, de alto nível, de uma função que o sistema deve fazer ou de uma restrição do sistema, e num outro extremo, o requisito é uma definição detalhada, matematicamente formal de uma função do sistema".

Peters e Pedricz (2001, p. 102) conectam a definição de requisito de software à qualidade do produto resultante através de um documento de requisito de software:

> Um requisito de software é uma descrição dos principais recursos de um produto de software, seu fluxo de informações, comportamento e atributos. Em suma, um requisito de software fornece uma estrutura básica para o desenvolvimento de um produto de software. O grau de compreensibilidade, precisão e rigor da descrição fornecida por **um documento de requisito de software tende a ser diretamente proporcional ao grau de qualidade do produto resultante**. (PETERS e PEDRICZ, 2001, p.102, grifo do autor)

Na seção 2.2.1.6.2, que trata do documento de ERS – Especificação de Requisitos de Software, serão abordados em detalhes as características dos requisitos.

b) Classificações de requisitos

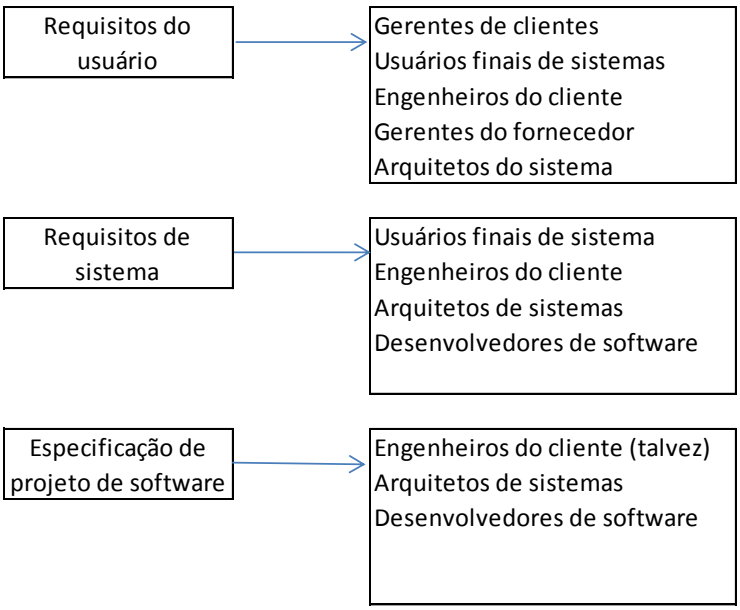
Paula Filho (2001, p. 7) define que em “tratando-se de software, costuma-se dividir as suas características entre funcionais e não funcionais, sendo que a características funcionais representam os comportamentos que um programa ou sistema deve apresentar diante de certas ações de seus usuários, enquanto as características não funcionais quantificam determinados aspectos do comportamento”.

Segundo Kotoya (1998) apud Silva (2006), os requisitos são sentenças que indicam necessidades dos interessados, sendo que os requisitos funcionais representam a funcionalidade do sistema, e os requisitos não funcionais restringem os requisitos funcionais.

Sommerville (2003, p. 82) distingue os “requisitos entre requisitos de usuários, requisitos de sistema e especificação de projeto de software, que representa diferentes níveis de abstração para cada tipo de interessado no produto de software”, conforme demonstra na Ilustração 17 - Leitores dos diferentes tipos de especificação, e classifica-os ainda entre funcionais, não funcionais ou como requisitos de domínio, sendo:

> Requisitos funcionais: São declarações de funções que o sistema deve fornecer, como o sistema deve reagir a entradas específicas e como deve se comportar em determinadas situações. Em alguns casos os requisitos funcionais podem também explicitamente declarar o que o sistema não deve fazer.
> Requisitos não funcionais: São restrições sobre os serviços ou as funções oferecidas pelo sistema. Entre eles destaca-se restrições de tempo, restrições sobre processo de desenvolvimento, padrões, entre outros.
> Requisitos de domínio: São requisitos que se originam do domínio de aplicação do sistema e que refletem características desde domínio. Podem ser requisitos funcionais e não funcionais.
> (SOMERVILLE, 2003, p. 83)

Ilustração 17 – Leitores dos diferentes tipos de especificação



Fonte: Adaptado de Sommerville (2003, p. 83)

Já Peters e Pedricz (2001) através de um modelo de *feed-back* ao processo de requisitos, descreve que o processo de análise de requisitos auxilia a identificar os principais recursos subdivididos em:

> - *Funcionais (ações principais)*: Uma descrição funcional identifica as atividades do sistema,
> - C*omportamental (atividades de controle)*: Uma descrição comportamental descreve a sequencia e a possível sobreposição das funções do sistema em uma hierarquia de atividades de controle.
> - *Não comportamental (atributos)*: Uma descrição não-comportamental do software inclui planejamento de engenharia humana e de garantia da qualidade.
> (PETERS e PEDRICZ, 2001, p. 102)

Já o RUP (RATIONAL, 2002) recomenda que os requisitos funcionais sejam organizados em casos de uso, ao invés de uma lista com marcadores, devendo ilustrar como uma pessoa usa o sistema, o que enfatiza a importância do requisito na visão do usuário, e que o uso de notações UML (Linguagem Unificada de Modelagem) enfatizam o comportamento esperado do sistema.

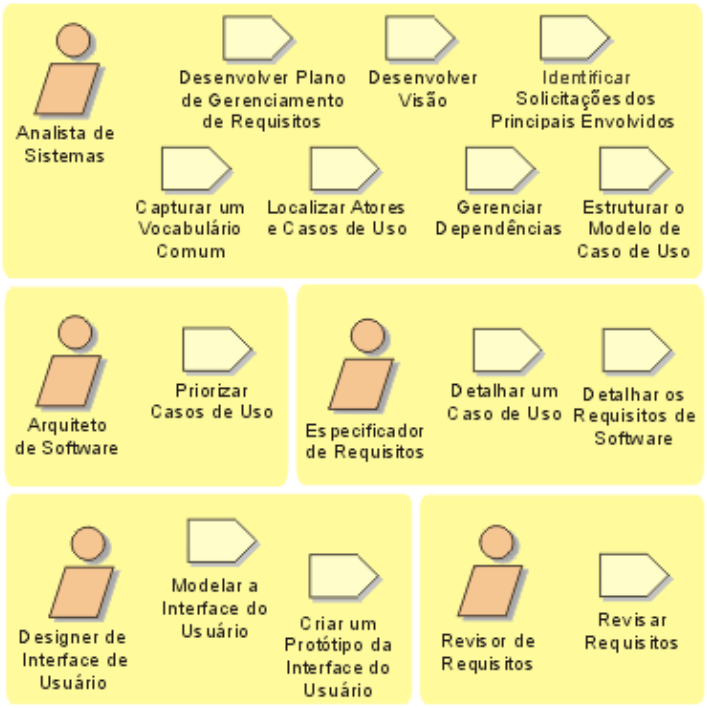
Destaca ainda o RUP (RATIONAL, 2002) que uma especificação complementar e os casos de uso captam todos os requisitos de software (funcionais e não funcionais) necessários a ERS completa.

Os autores, Peters e Pedricz (2001, p. 103), classificam ainda que dentro da análise de requisitos (ou análise de problemas), a especificação de requisitos tem como objetivo descrever os objetos, funções e os estados relacionados ao problema.

2.2.1.7.1. Documento de ERS

O RUP (RATIONAL, 2002) atribui a função da ERS a um especificador de requisitos, que tem como atribuição detalhar Casos de Uso e os Requisitos de Software, conforme destaca a Ilustração 18 – Especificador de Requisitos.

Ilustração 18 – Especificador de Requisitos



Fonte: Adaptado do RUP (RATIONAL, 2002)

Em termos de documento, uma prática recomendada para elaboração de uma ERS é orientada pela IEEE Std. 830 (1998), a qual é referenciada em diversas obras, como será visto ao longo desta seção.

Um modelo de estrutura de um documento sugerido pela IEEE 830 é apresentado na Ilustração 19 – Exemplo da estrutura de ERS segundo IEEE 830.

Ilustração 19 – Exemplo de estrutura da ERS segundo IEE 830

Índice analítico
1. Introdução
    1.1 Objetivo
    1.2 Escopo
    1.3 Definições, acrônimos e abreviaturas
    1.4 Referências
    1.5 Visão geral
2. Descrição geral
    2.1 Perspectivas do produto
    2.2 Funções do produto
    2.3 Características do usuário
    2,4 Restrições
    2.5 Suposições e dependências
3. Requisitos específicos

Apêndices

Índice

Fonte: Adaptado de IEEE Std. 830 (1998)

A IEEE Std. 830 (1998) sugere ainda 8 (oito) variações de modelos para o item 3 que trata dos requisitos específicos, variando na sua forma por organização, modo, classe de usuário, objeto, recurso, estímulo e pela hierarquia funcional ou em várias organizações.

Segundo Peters e Pedricz (2001), existem várias abordagens para escrever uma ERS, que num dos capítulos de seu livro, seu modelo de ERS é derivada do Padrão IEEE 830 com adição da rastreabilidade de requisitos, baseada no padrão do Departamento de Defesa do Estados Unidos, como sintetiza a Ilustração 20 – Exemplo da estrutura de ERS segundo Peter e Pedricz.

Ilustração 20 – Exemplo de estrutura da ERS segundo Peter e Pedricz

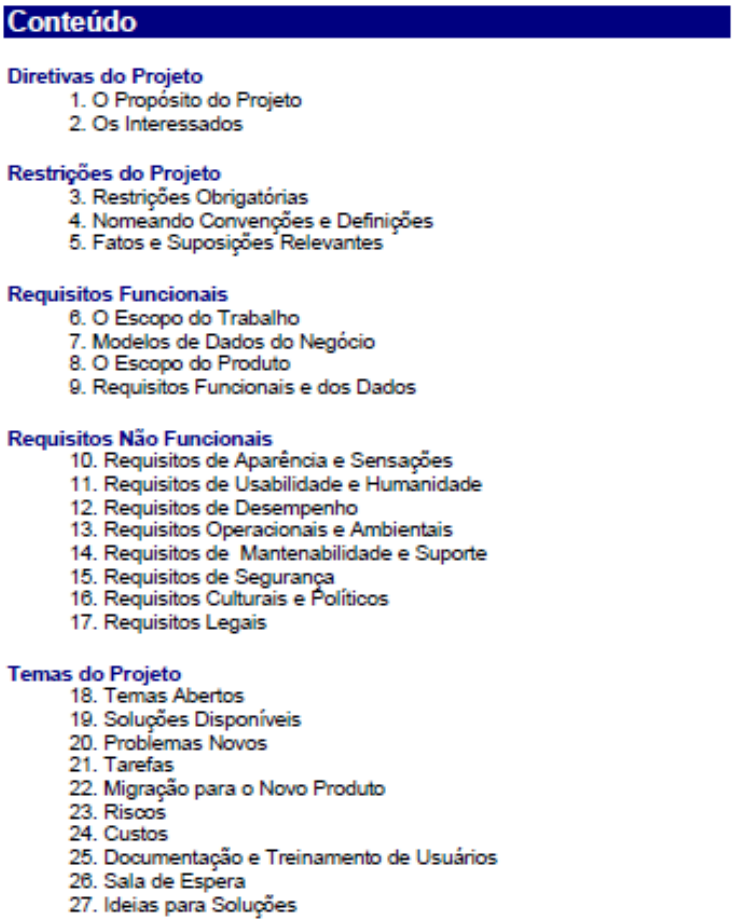
Índice Analítico
1. Introdução
   1.1 Objetivo
   1.2 Escopo
   1.3 Definições
   1.4 Referências
   1.5 Visão geral
2. Descrição global
   2.1 Perspectivas do produto
   2.2 Funções do produto
   2.3 Características do usuário
   2.4 Restrições
   2.5 Hipóteses e dependências
3. Requisitos específicos
   (interfaces externas,
   requisitos de processo e dados,
   requisitos de desempenho e qualidade,
   requisito de banco de dados lógico,
   restrições de projeto,
   atributos de sistema de software,
   organização de requisitos específicos)
4. Rastreabilidade dos requisitos
Apêndices
Índice remissivo

Fonte: Adaptado de Peters e Pedricz (2001, p. 104)

Sommerville (2003, p. 98) também faz uma abordagem ao padrão IEEE 830, e coloca que "embora esse padrão não seja o ideal, ele contém uma grande quantidade de boas orientações sobre como escrever os requisitos e como evitar problemas. Critica ainda que é muito geral para ser um padrão organizacional, e **orienta que o modelo IEEE deva ser adaptado às necessiadades de uma organização em particular**." (grifo do autor)

A Volere (2009), uma associação especializada em recurso relativos a requisitos, produziu um manual prático para ERS, recomendando uma estrutura ampla que pode ser adaptada para cada aplicação. Uma estrutura básica produzida pela Volere é apresentada na Ilustração – 21 – Exemplo de estrutura de ERS da Volere.

Ilustração 21 – Exemplo de estrutura da ERS segundo Volere



Fonte: Volere (2009, p. 2)

Outro modelo prescrito de ERS é fornecido pelo RUP Ilustração – 22 – Exemplo de estrutura da ERS do RUP.

Sommerville (2003, p. 95), menciona que “a especificação de requisitos de software, é a declaração oficial do que é exigido dos desenvolvedores do sistema, e que devido ao fato dele atender a diversos tipos de usuários, conforme o número de requisitos, os requisitos de usuário e de sistema devem estar estruturados de tal forma que permita sua melhor compreensão, integrado num único documento, ou em documentos separados.”

Ilustração 22 – Exemplo de estrutura da ERS segundo RUP

Fonte: Adaptado do RUP (RATIONAL, 2002)

Por fim, Paula Filho (2001) reforça, a exemplo do que descreve a IEEE 830:1993, que este documento deve ser elaborado por **membros da equipe de desenvolvimento, com a participação obrigatória de um ou mais usuários chaves** do produto, sendo este usuário chave, aquele indicado pelo cliente, com capacidade e experiência no domínio do problema, atuando como fornecedor de requisitos. (grifo do autor)

2.2.1.7.2. Características de uma ERS

Segundo Paula Filho (2001, p. 56), “para servir de base a um produto de boa qualidade, a própria Especificação de Requisitos deve satisfazer uma série de características de qualidade”, a saber:

> Correta – Todo requisito presente realmente é um requisito do produto a ser construído.
> Precisa – Todo requisito presente possui apenas uma única interpretação, aceita tanto pelos desenvolvedores quanto pelos usuários chaves.
> Completa – Reflete todas decisões de especificação que foram tomadas.
> Consistente – Não há conflitos entre nenhum dos subconjuntos de requisitos presentes.
> Priorizada – Cada requisito é classificado de acordo com a sua importância, estabilidade e complexidade.
> Verificável – Todos requisitos são verificáveis
> Modificável – Sua estrutura e estilo permitem a mudança de qualquer requisito, de forma fácil, completa e consistente.
> Rastreável – Permite a fácil determinação dos antecedentes e consequência de todos os requisitos.

Tais características citadas por Paula Filho (2001) são oriundas e recomentadas da IEEE 830:1993.

O RUP (RATIONAL, 2002) destaca que a ERS usando um modelo preciso e não ambíguo, ajuda a garantir que todos os envolvidos possam entender e concordar com os requisitos.

Heninger (1980) apud Sommerville (2003, p. 95) em outras palavras já defendia que um documento de requisitos de software deveria satisfazer:

> - Deveria satisfazer somente o comportamento externo do sistema
> - Deveria especificar as restrições à implemenação
> - Deveria ser fácil de ser modificado
> - Deveria registrar a estratégia sobre o ciclo de vida do sistema
> - Deveria caracterizar respostas aceitáveis para eventos indesejáveis

Paula Filho (2001, p. 54) completa que uma "Especificação de Requisitos de Software é o documento oficial de descrição de requisitos de um projeto de software, e pode se referir a um produto indivisível de software, ou a um conjunto de componentes de software que forma um produto, e deve incluir as seguintes características também":

> - Funcionalidade: O que o software deverá fazer
> - Interfaces externas: Como o software interage com as pessoas, com o hardware do sistema, com outros sistemas e com outros produtos?
> - Desempenho: Quais velocidades de processamento, o tempo de resposta e outros parâmetros de desempenho requeridos pela natureza da aplicação?
> - Outros atributos: Quais as considerações sobre portabilidade, manutenibilidade e confiabilidade que devem ser observadas?
> - Restrições impostas pela aplicação: Existem padrões e outros limites a serem obedecidos, como linguagem de implementação, ambientes de operação, limites de recursos, etc.

A IEEE Std. 830 (1998) recomenda porém, que uma ERS deve evitar abordar questões de design e projeto.

Em relação a atributos, na próxima seção é aprofundado as considerações elencadas pelos autores citados.

2.2.1.7.3. Atributos de qualidade que influenciam uma ERS

A seção 2.1.3.3 delineou uma abordagem sobre qualidade de software, que nesta seção categorizam-se os atributos de qualidade, expressos num modelo de qualidade de software fornecido pela NBR ISO/IEC 9126-1 (ABNT, 2003), que podem nortear uma ERS, conforme mencionou Paula Filho, Op. Cit. p. 54, os quais se apresenta:

a) Atributos de qualidade interna e externa:

I. Funcionalidade: Capacidade do software em prover funções que atendam às necessidades explícitas e implícitas, e está subdividido em:

i. Adequação: Capacidade de prover funções para tarefas e objetivos do usuário especificados;
ii. Acurácia: Capacidade de software de prover, com grau de precisão necessário, resultados conforme esperados;
iii. Interoperabilidade: Capacidade de interagir com um ou mais sistemas especificados;
iv. Segurança de acesso: Capacidade de proteger informações e dados contra pessoas ou sistemas não autorizados, e permitir acesso às pessoas ou sistemas autorizados;
v. Conformidade relacionada à funcionalidade: Capacidade de estar de acordo com normas, convenções ou regulamentações previstas em leis relacionados à funcionalidade.

II. Confiabilidade: Capacidade do software de manter um nível de desempenho especificado, e está subdividido em:

i. Maturidade: Capacidade de evitar falhas decorrentes de defeitos no software;
ii. Tolerância a falhas: Capacidade de manter um nível de desempenho especificado em casos de defeitos ou de violação;
iii. Recuperabilidade: Capacidade de restabelecer um nível de desempenho em casos de defeitos ou de violação;
iv. Conformidade relacionada à confiabilidade: Capacidade de estar de acordo com normas, convenções ou regulamentações relacionadas à confiabilidade.

III. Usabilidade: Capacidade do software de ser compreendido, aprendido, operado e atraente ao usuário, e está subdividido em:

i. Inteligibilidade: Capacidade de possibilitar ao usuário compreender se o software é apropriado e como ele pode ser usado para tarefas e condições de uso;
ii. Apreensibilidade: Capacidade de possibilitar ao usuário aprender sua aplicação;
iii. Operacionabilidade: Capacidade de possibilitar ao usuário operá-lo e controlá-lo;
iv. Atratividade: Capacidade de ser atraente ao usuário;
v. Conformidade relacionada à usabilidade: Capacidade de estar de acordo com normas, convenções, guias de estilo ou regulamentações relacionadas à usabilidade.

IV. Eficiência: Capacidade do software de apresentar desempenho apropriado, relativo à quantidade de recursos usados, e está subdividido em:

i. Comportamento em relação ao tempo: Capacidade de fornecer tempos de resposta e de processamento, e taxas de transferências apropriadas;
ii. Utilização de recursos: Capacidade de usar tipos e quantidades apropriados de recursos;
iii. Conformidade relacionada à eficiência: Capacidade de estar de acordo com normas e convenções relacionadas à eficiência;

V. **<u>Manutenibilidade</u>: Capacidade de ser modificado. As modificações podem incluir correções, melhoras ou adaptações devido a mudanças no ambiente e nos seus requisitos ou especificações funcionais, e está segmentado em:**

i. <u>Modificabilidade</u>: Capacidade de permitir que uma modificação especificada seja implementada;
ii. <u>Estabilidade</u>: Capacidade de evitar efeitos inesperados decorrentes de modificações no software;
iii. <u>Testabilidade</u>: Capacidade de permitir que o software, quando modificado, seja validado;
iv. <u>Conformidade relacionada a manutenibilidade</u>: Capacidade de estar de acordo com normas ou convenções relacionadas à manutenibilidade.

VI. **<u>Portabilidade</u>: Capacidade do software de ser transferido de um ambiente para outro, e está subdividido em:**

i. <u>Adaptabilidade</u>: Capacidade de ser adaptado para diferentes ambientes, sem necessidade de aplicação de outras ações ou meios além daqueles fornecidos para essa finalidade pelo software considerado;
ii. <u>Capacidade para ser instalado</u>: Capacidade para ser instalado em um ambiente especificado;
iii. <u>Coexistência</u>: Capacidade de coexistir com outros produtos de software independentes, em uma ambiente comum, compartilhando recursos comuns;
iv. <u>Capacidade para substituir</u>: Capacidade de ser usado a outro produto de software, com o mesmo propósito e no mesmo ambiente;
v. <u>Conformidade relacionada à portabilidade</u>: Capacidade de estar de acordo com normas ou convenções relacionadas à portabilidade.

b) <u>Atributos de qualidade em uso</u>:

I. **<u>Qualidade em uso</u>: Capacidade do software de permitir que usuários atinjam metas especificadas com eficácia, produtividade, segurança e satisfação, e está subdividido em:**

i. Eficácia: Capacidade de permitir que usuários atinjam metas com acurácia e completitude;
ii. Produtividade: Capacidade de permitir que seus usuários empreguem quantidade apropriada de recursos em relação à eficácia obtida;
iii. Segurança: Capacidade de apresentar níveis aceitáveis de riscos de danos a pessoas, negócios, software, propriedades ou ao ambiente;
iv. Satisfação: Capacidade de satisfazer usuários, em um contexto de uso especificado;

Por fim, encerra-se este capítulo considerando que os principais temas norteadores foram abordados, e que apesar de vasta conteúdo bibliográfico disponível para consulta, os autores consultados e citados foram suficientes para delimitação e destaque do objeto de estudo.

Na próxima seção é iniciado o capítulo que aborda os métodos utilizados para a pesquisa.

3 MÉTODOS

Neste capítulo serão apresentados os métodos utilizados na pesquisa.

3.1 CARACTERIZAÇÃO DA PESQUISA

A pesquisa é do tipo exploratória, de caráter meramente qualitativo, e trata-se de um estudo bibliográfico e estudo de caso, pois visa a pesquisa de material publicado, e o estudo da organização em questão, com seus processos e práticas relacionados ao tema de estudo, a Engenharia de Requisitos.

3.2 EQUIPAMENTOS E MATERIAIS UTILIZADOS

Para realização da pesquisa foram utilizados como recursos físicos, bloco de anotações, caneta, computador com acesso à internet e impressora.

3.3 INSTRUMENTO DE COLETA DE INFORMAÇÕES

Como instrumento de coleta de informações foi utilizado a técnica de entrevista semi-estruturada, com o objetivo de conduzir o entrevistado pelos temas de interesse, mas ainda sendo-lhe permitido liberdade para se expressar. De acordo com D´Oliveira, Lima e Luna (1996) apud Silva (2008, p. 35).

> a entrevista guiada permite ao entrevistador utilizar um ‘guia’ de temas a ser explorado durante o transcurso da entrevista [...] o pesquisador conhece previamente os aspectos que deseja pesquisar e, com base neles, formula alguns pontos a tratar na entrevista. O entrevistado tem a liberdade de expressar-se como ele quiser guiado pelo entrevistador.

As informações coletadas foram anotadas à mão livre, para posterior transcrição no presente trabalho de forma estruturada.

3.4 PROCEDIMENTOS

A pesquisa foi realizada em 2 (duas) etapas, sendo a primeira constituída de uma pesquisa bibliográfica, e a segunda etapa composta pelo estudo de campo na organização.

A pesquisa bibliográfica foi realizada em material disponível na biblioteca da Unisul e em sites. Diversos conteúdos destes foram coletados, e os selecionados formaram a Fundamentação Teórica do Capítulo 2 do presente estudo.

A pesquisa de campo se deu nas dependências da SED, onde foi possível realizar entrevista apenas com o Gerente de Tecnologia e Governança Eletrônica da Secretaria de Estado da Educação – GETIG, pois não foi permitida abordagem direta à equipe de desenvolvimento e demais usuários chaves na SED, sendo estes representados pelo GETIG, que tem conhecimento de todo processo prático.

O GETIG tem formação acadêmica na área de tecnologia da informação, possuindo diversos cursos técnicos, e está lotado na SED a mais de 20 (vinte) anos, tendo trabalhado em áreas operacionais, e recentemente assumiu a gerência do setor de tecnologia.

A entrevista foi realizada em Janeiro de 2013, na sala da GETIG, e durou cerca de 4 (quatro) horas, sendo norteada basicamente por 3 (três) questões guias, que estão abaixo relacionadas, e vinculam-se aos objetivos do presente estudo:

a) 1ª. abordagem:

Inicialmente foi solicitado ao entrevistado que descrevesse de forma geral a organização, o negócio, estrutura física e funcional SED, e os sistemas de software que são utilizados pela organização.

b) 2ª. abordagem:

Posteriormente foi questionado ao entrevistado a hipótese de pesquisa formulada no presente estudo, e que ajuda também a responder parte dos objetivos específicos do

trabalho: A Secretaria de Estado da Educação tem um modelo prescrito de especificação de requisitos de software (ERS) para a construção de seus sistemas de softwares?

c) 3ª. abordagem:

Em seguida, foi solicitado ao entrevistado que descrevesse o processo de desenvolvimento de software na SED, desde o levantamento de necessidades até a entrega do produto final de software para uso.

As respostas e dinâmica da entrevista foram anotadas, e transcritas de forma estruturada no presente estudo, e são apresentadas no próximo capítulo como resultados obtidos.

4 RESULTADOS OBTIDOS

Neste capítulo são apresentados transcritos os resultados da aplicação da pesquisa.

4.1 SOBRE A ORGANIZAÇÃO

A Secretaria de Estado da Educação (SED) é um órgão do poder executivo estadual, instituído pela Constituição Estadual do Estado de Santa Catarina de 1989.

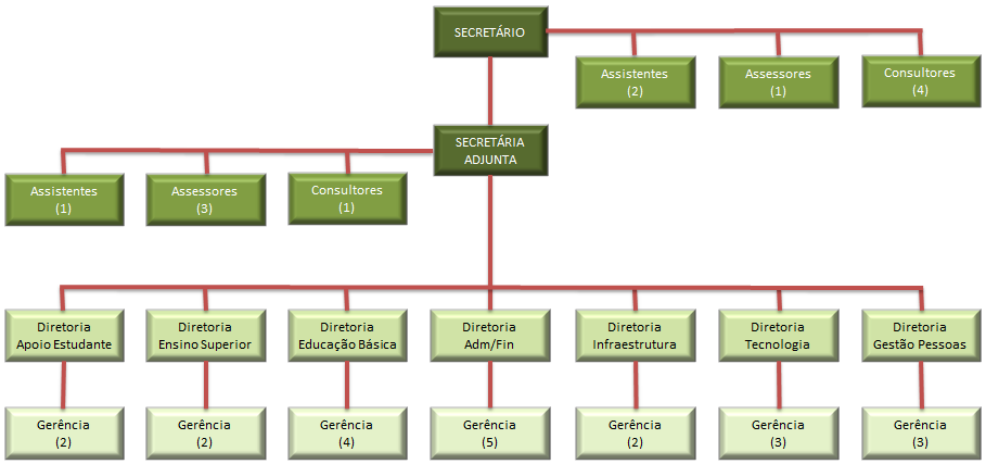
A SED é mantida com recursos do Governo Estadual e Federal, possuindo orçamento anual de em torno de R$ 3 Bilhões por ano.

Sua sede está localizada em uma unidade administrativa na região central da capital do Estado de Santa Catarina.

Desde sua fundação até o presente ano, 2013, a SED já foi comandada por 16 (dezesseis) Secretários de Estado.

Ao longo dos anos passou por diversas reformas administrativas, efetuados conforme as políticas de governos vigentes à época, sendo que atualmente conta com 43 (quarenta e três) colaboradores que ocupam cargo de confiança, regulamentados pela Lei Complementar no. 534 de 20 de abril de 2008, assim representados na Ilustração 23 – Organograma da SED.

Ilustração 23 – Organograma da SED



Fonte: Adaptado pelo Autor

Conta ainda com 36 (trinta e seis) unidades descentralizadas, representadas pelas Gerências Regionais de Educação (GEREDs), estabelecidas fisicamente nas Secretarias de Desenvolvimento Regional (SDRs) no Estado que dão apoio articulado à SED, e possui 1.280 unidades escolares distribuídas pelos municípios de Santa Catarina.

Possui um corpo funcional de aproximadamente 5.000 servidores efetivos e comissionados distribuídos nas escolas, que representam a estrutura administrativa da SED como num todo em conjunto com o prédio sede, e aproximadamente 38.000 professores efetivos e temporários que representam a estrutura operacional nas escolas, e um contingente anual de quase 800.000 alunos/ano, do 1º. ano fundamental ao 3º. ano do ensino médio.

Tem como atribuição de competências a Lei Complementar no. 284 de 28 de fevereiro de 2005 e alterações posteriores, que é citada a íntegra de seu art. 64, com destaque especial aos inciso em destaque:

> Art. 64. À Secretaria de Estado da Educação, Ciência e Tecnologia, compete:
> **I - formular, coordenar, controlar e avaliar a execução das políticas educacionais da educação básica, profissional e superior em Santa Catarina, observando as normas regulamentares de ensino emanadas do Conselho Estadual de Educação de Santa Catarina;**
> [...]
> **III - coordenar a elaboração e acompanhar a execução de programas de educação superior para o desenvolvimento regional;**
> **IV - definir e implementar a política de tecnologias educacionais;**
> [...]
> **VI - fomentar a utilização de metodologias e técnicas estatísticas do banco de dados da educação, objetivando a divulgação das informações aos gestores escolares;**
> **VII - coordenar, acompanhar e participar da elaboração e execução de um programa de pesquisa na rede pública do Estado;**
> **VIII - formular e implementar a Proposta Curricular de Santa Catarina;**
> [...]
> **XI - sistematizar e emitir relatórios periódicos de acompanhamento e controle de alunos, escolas, profissionais do magistério, de construção e reforma de prédios escolares e aplicação de recursos financeiros destinados à educação;**
> **XII - coordenar as ações da educação de modo a garantir a unidade da rede, tanto nos aspectos pedagógicos quanto administrativos;**
> **XIII - apoiar, assessorar e supervisionar as Secretarias de Estado de** [...]
> **XVII - estimular a articulação entre as instituições de pesquisa, as universidades e os setores produtivos e o seu intercâmbio com instituições de pesquisa de outros estados brasileiros e do exterior;**
> [...]
> **XIX - normatizar, supervisionar, orientar, controlar e formular políticas de gestão de pessoal do magistério público estadual articuladamente com o órgão central do Sistema de Gestão de Recursos Humanos; e**
> **XX - promover a formação, treinamento e aperfeiçoamento de recursos humanos para garantir a unidade da proposta curricular no Estado de Santa Catarina, articuladamente com o órgão central do Sistema de Gestão de Recursos Humanos.**
> (Lei 284 de 28 de fevereiro de 2005, grifo nosso)

Desta Lei, dos 20 (vinte) incisos relacionados, os 12 (doze) incisos destacados em negrito são apoiados ou suportados por sistemas de software, desenvolvidos pela SED, para que a mesma possa atingir seus objetivos gerais por meio da gestão da informação.

4.1.1 Sobre a área de tecnologia da SED

No que tange a Tecnologia da Informação (TI), a SED possui uma Diretoria na qual está ligada a Gerência de Tecnologia da Informação e Governança Eletrônica (GETIGE), que é responsável por todo ambiente de informática e telecomunicações, em hardware e software.

A GETIGE conta com 36 (trinta e seis) Núcleos de Tecnologia (NTEs) sediados nas SDRs como braço de apoio para questões regionalizadas ligadas a TI.

A GETIGE atende as demandas de TI do prédio central e unidades escolares através de técnicos próprios, empresas e profissionais terceirizados, e também através dos NTEs, conforme o tipo de ocorrência.

Os atendimentos estão divididos entre; a) software e sistema; b) hardware, rede interna e ativos de rede; c) servidores; d) email e internet; e) rede de governo; f) telefonia.

Em termos de equipamentos, tem um parque instalado de mais de 20.000 (vinte mil) computadores e 3.000 (três mil) impressoras, além de milhares de ativos de rede como hub, switch, modem, acesses point, roteadores, servidores.

Para o ano de 2013 o parque será ampliado com a adição de aproximadamente 2.000 (duas mim) lousas digitais e 12.000 (doze mil) *tablets* através do programa Pacto pela Educação, um projeto do Governo Estadual financiado por recursos federais, que pretende melhorar a qualidade do ensino no Estado através do melhoramento do aparato tecnológico.

Na próxima seção dedica-se um espaço para focar a questão de sistemas e software, para alinhar o foco de estudo da presente pesquisa.

4.1.2 Sobre os sistemas de softwares da SED

Em termos de sistemas de software, a SED é usuário dos seguintes sistemas corporativos adotados pelo governo de uma forma geral: SIGRH, FRH, SIGEF, SGPE, LIC, DOE, SME, PAE, **GETEC, SIGESC, PORTAL**, sendo os 3 (três) últimos em destaque de propriedade da SED. (grifo do autor)

Todos os sistemas são acessados via internet através de uma rede corporativa do Estado, que é mantida pelo CIASC, obedecendo regras e políticas de segurança de acesso e uso. O CIASC é uma empresa pública de economia mista mantida pelo Governo do Estado, e lida com questões estratégicas do governo em informática, conforme art. 100 e 108 da Lei Complementar 284.

Os 3 (três) sistemas de propriedade da SED foram desenvolvidos internamente através de empresas fornecedoras de mão de obra de tecnologia, como o próprio CIASC de forma contínua, e outras empresas privadas através de demandas específicas.

Estes sistemas estão instalados em servidores da SED dentro das dependências do CIASC, configurados em ambiente de homologação e produção, sendo que o ambiente de desenvolvimento está configurado nos servidores da SED.

A equipe de produção, operação e suporte de sistemas chegou a ser composta por 27 (vinte e sete) profissionais no ano de 2010 alocados fisicamente na SED, entre técnicos do CIASC e de empresas privadas, onde a produção de software foi acelerada e atingiu bons níveis de qualidade, contudo, desde o final de 2011 esse número de profissionais vem diminuindo por ocasião de reformulação de contratos com as empresas fornecedoras, e a SED conta atualmente com apenas 4 (quatro) técnicos do CIASC atuando como desenvolvedores em suas instalações.

O principal sistema da SED atualmente é o SIGESC, um sistema *Web* que foi concebido em pouco mais de 1 ano, que entrou em uso no final de 2010 e início de 2011, em vem substituindo um sistema legado chamado SERIE EDU, o qual foi concebido, aprimorado e mantido durante 18 (dezoito) anos na SED, e que vem sendo descontinuado face sua desatualização e limitações tecnológicas, sobre tudo por ser desenvolvido em plataforma *desktop,* não permitindo a integração das escolas, e por apresentar problemas de concepção de projeto em termos de desenho, desempenho, segurança e confiabilidade, haja vista possui quase 2 (duas) década de idade, e por ter evoluído sem planejamento de longo prazo.

O SERIE EDU foi desenvolvido em Genexus V.7, operando com banco de dados local do Access, e o SIGESC foi desenvolvido em Genexus V.10, ferramenta de alta produção, com geração de código em .NET, rodando sobre banco de dados SQL Server 2005.

O SIGESC é um software administrativo que gerencia dados das unidades escolares, alunos e professores, sendo responsável por um dos processos de negócio mais críticos da SED, que é a geração de dados para folha de pagamento dos professores, que é formada a partir de várias regras de negócio do processo acadêmico e funcional, como carga horária, especialidade curricular, matriz curricular, gratificações, licenças, faltas, entre outras variadas regras.

É no SIGESC que as principais demandas são incorporadas, pois ele foi concebido para ser um sistema único da SED, que centralizada e compartilha toda inteligência de negócios, crescendo em módulos, como Acadêmico, Biblioteca, Merenda Escolar, Transporte Escolar, ACT, Art 171, entre outros, e diversas são as demandas rotineira que fazem este sistema evoluir, e tornar-se mais completo, servido de base única de informações.

4.2 SOBRE A HIPÓTESE DE PESQUISA

Em relação a hipótese de pesquisa, (A Secretaria de Estado da Educação tem um modelo prescrito de especificação de requisitos de software (ERS) para a construção de seus sistemas de softwares?), a resposta do entrevistado foi negativa.

Explicou o entrevistado que não há um modelo de documento prescrito e padronizado na SED, que descreva os requisitos de cada sistema já concebido até o presente momento.

Complementou que existem apenas anotações realizadas pelos analistas quando do levantamento de necessidades, de forma não estruturada, e documentos adicionais que são fornecidos pelos usuários chaves, como manuais, editais, leis, decretos, normas e instruções normativas e portarias.

4.3 SOBRE O PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

A partir da negativa do usuário quanto hipótese de pesquisa, a não existência de um modelo de documento de ERS, foi solicitado ao mesmo que então descrevesse o processo de desenvolvimento de software na SED desde a sua concepção até a entrega do produto ao usuário chave, com o objetivo de reforçar a hipótese pesquisada, e como forma também de avaliar o processo para sugerir um modelo adequado à realidade da organização, que é um dos objetivos deste estudo.

Nesta etapa da entrevista o entrevistado relatou que o processo está subdividido em 2 (dois) momentos distintos e consecutivos, sendo um destinado a avaliação de necessidades e viabilidade pela alta gerência da SED, e outro em que, se aprovado, passa para o processo seguinte em nível mais operacional, que trata da construção do software em si.

O primeiro processo é representado pela Ilustração 24 – Processo de avaliação de necessidades e viabilidade, e é assim descrito:

a) Processo 1:

   a. Os usuários de uma determinada área identificam ou sentem a necessidade de serem atendidos por uma solução de software para atender determinada demanda;

   b. Levam sua necessidade à Diretoria de área respectiva;

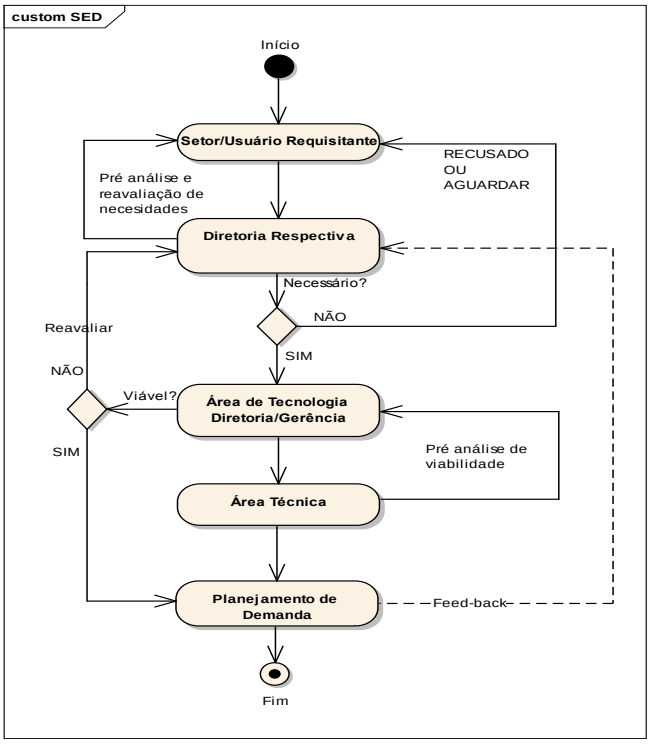
   c. Esta Diretoria avalia em conjunto com a área demandante se realmente é uma necessidade real, e se não está fora do foco das diretrizes do setor em relação a prioridades, e estando de acordo, leva à diante para discussão com a área de tecnologia, ou já descarta a demanda dentro de sua área sem levar adiante, ou aguardando para rediscuti-la em outro momento com sua equipe;

   d. A Diretoria demandante leva ao conhecimento da área de tecnologia, que por sua vez procede uma análise técnica junto aos desenvolvedores, e dá um parecer sobre a viabilidade de atender ou não a demanda requisitada dentro das premissas apresentadas, podendo

aceitar ou não a demanda dentro dos recursos disponíveis, ou ainda prioriza mediante ordem da alta administração da SED;

e. Estando aprovado tecnicamente a realização, a demanda vai para área técnica para ao menos estar preparada para absorver a demanda, e poder se planejar para atendê-la, de forma imediata ou programada. Esta demanda é comunicada normalmente de forma verbal, para cientificar os envolvidos.

Ilustração 24 – Processo de avaliação de necessidades e viabilidade



Fonte: Elaborado pelo autor

Somente após a última etapa no processo supra é que se iniciará em algum momento a interação da área técnica com o usuário demandante, ou usuários chaves, conforme Ilustração 25 – Processo de construção de software, que é assim descrito:.

b) Processo 2:

a. A área técnica, através de analistas ou desenvolvedores, executam junto com o usuário chave um processo de levantamento de necessidades. O

usuário chave é chamado a municiar a equipe de informações verbais e formais a respeito da demanda, onde o analista atua como um investigador e interprete das necessidades para implementar a solução a nível de software. O Analista tenta capturar as informações através de esboços manuscritos e através de documentação complementar fornecida pelo usuário chave, para então passar para fase de desenvolvimento;

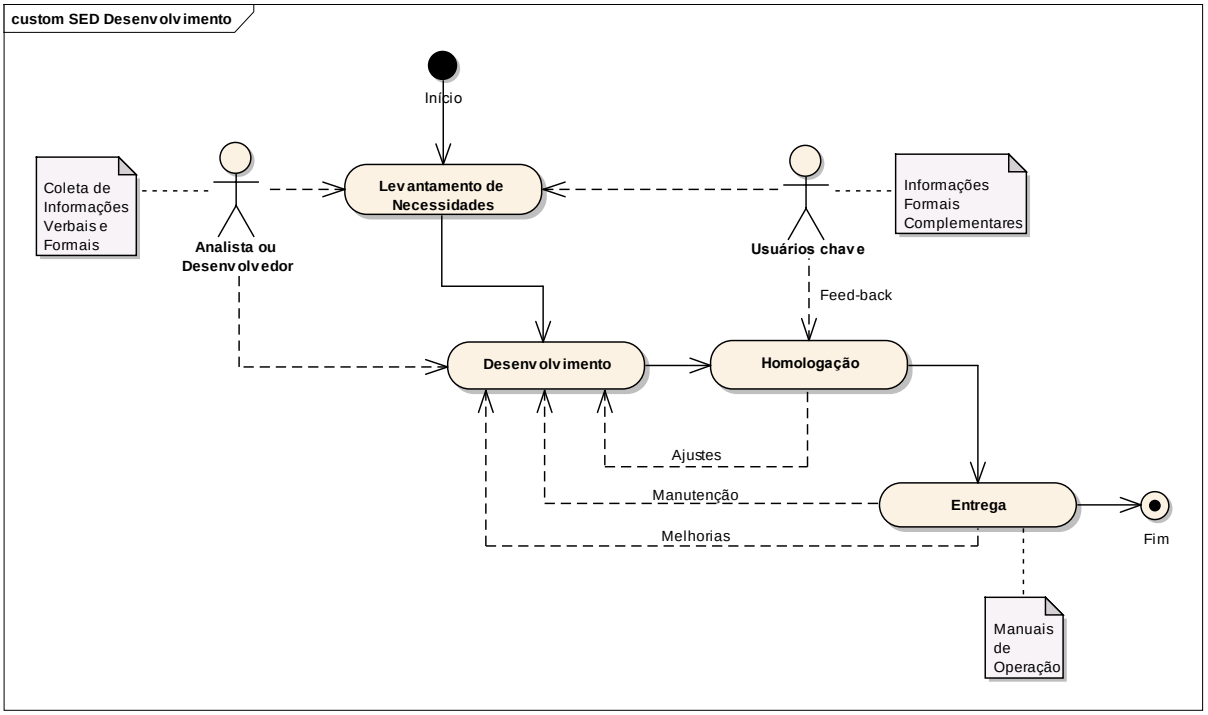
b. Na etapa de desenvolvimento é muito provável que o desenvolver seja o mesmo que ocupou a função de analista, o que de certa forma facilita o processo de criação e implementação. Contudo, se a demanda for repassada a outros da equipe, de forma individual ou coletiva conforme o tamanho da demanda, são realizadas reuniões para difusão da informações coletadas. Caso existam dúvidas relativas a etapa de levantamento, o usuário chave é convocado para esclarecê-las a qualquer tempo do desenvolvimento. Não são gerados nenhum documento formal de projeto, de modelagem, ou de especificação de requisitos para construção do software, sendo reconhecidos e compartilhados os acordos verbais e documentos obtidos do usuário chave;

c. Quando algum entregável é passível de ser homologado pelo usuário chave, após uma validação pela equipe de desenvolvimento, o usuário chave é convocado para fazer a sua crítica. Na hipótese de haver correções, o feed-back do usuário chave demandará ajustes no desenvolvimento num processo retroalimentado até se obter uma aprovação da homologação;

d. Após aprovação na etapa de homologação, e havendo um conjunto mínimo operacional de entrega viável, é preparado manuais para capacitação de usuários, onde é realizada a entrega formal do produto, ou partes deste.

e. Após entrega o produto passa por processo de melhoria ou de correção sempre que necessário e possível, fechando o ciclo do processo de desenvolvimento do software, mantendo-se assim durante todo período de evolução e manutenção, até que seja retirado de uso.

Ilustração 25 – Processo de construção de software



Fonte: Elaborado pelo autor

Menciona o entrevistado durante a entrevista que não há nenhuma técnica, método, ou modelo formalmente constituído e seguido pela SED no desenvolvimento de software, e os processos vão sendo executados e desenvolvidos conforme necessidade, demanda e prioridade.

Destaca que a equipe atual é composta exclusivamente por desenvolvedores, e que não existe na equipe a função exclusiva de analista de negócios ou sistemas, bem como, não existe na equipe a função de testadores, coordenadores de trabalho, gerentes de projetos ou gerentes de processo.

A GETIG por sua vez representa apenas a função de mediação, acompanhamento e fiscalização, sem muita influência na capacidade e disponibilidade dos serviços de software prestados pela equipe de desenvolvedores, uma vez que estes não pertencem ao quadro funcional da SED, e sim de outra(s) organizações.

Em função do nível de informalidade existente na construção dos softwares, e pela falta de definição de papéis, técnicas e métodos, aproveitou-se questionar o entrevistado sobre prazos e confiabilidade nas entregas (em relação ao que foi pedido pelo usuário e efetivamente entregue pela equipe).

O entrevistado respondeu que praticamente todos os projetos atrasam em relação ao prazo acordado, tendo possivelmente, conforme seu sentimento, um índice de atrasos em torno de 90%, e que os recursos e funcionalidades nem sempre atendem aos requisitos iniciais dos usuários, o que demanda um processo de homologação mais demorado, ou o produto é entregue com deficiências já conhecidas para serem introduzidas e melhoradas posteriormente, durante o uso, uma vez que os prazos de entrega já se encontrariam atrasados.

4.4 SOBRE O DOCUMENTO DE ERS

Como não foi constatado o uso de um documento padrão de ERS, impossibilitando até uma análise para incrementos de melhoria a partir de um modelo existente na SED, apresenta-se no Apêndice II, um modelo de documento de ERS que se idealiza como sendo o mais adequado à sua realidade da SED.

O modelo de ERS apresentado no apêndice tem o propósito de ser simples, de fácil implementação e replicação, para que seja facilitado o seu uso contínuo, e foi elaborado a partir da IEEE 830:1998, levando em consideração as abordagens apresentadas na Fundamentação Teórica deste estudo, em especial alguns aspectos do RUP e Volere.

Faz parte do modelo ERS instruções de preenchimento, e o mesmo deve ser confeccionado de forma dinâmica, podendo crescer em quantidade e volume, ou ser suprimido em partes não aplicáveis conforme a aplicação prática.

Seria uma boa prática na constituição de um documento de ERS, identificá-lo numerando-o, dantando e versionando-o, procedendo sempre que possível, referências cruzadas com outras ERS, para completude da documentação.

Este modelo servirá de guia para a equipe de desenvolvimento e usuário chave, podendo servir como um passo-a-passo de itens que devem ser levados em consideração na construção de um software.

Servirá também como artefato de alinhamento de comunicação, que não cause interpretações errôneas sobre o que deve ser entregue, pois vários interessado lerão a mesma informação, proporcionando um processo adicional de verificação e crítica pelas partes interessadas, que não existente no processo atual da SED.

4.5 VALIDAÇÕES

A transcrição da entrevista realizada junto a instituição, demonstrada pelos capítulo 3 que trata dos métodos, e por este capítulo 4 que trata dos resultados, bem como o artefato de ERS produzido constante do Apêndice II, foram submetidos à avaliação do representante da instituição no fechamento do presente estudo para validação, na forma e teor que neste trabalho se encontram.

Os documentos supra mencionados foram devidamente validados pela instituição, e sua comprovação encontra-se no Apêndice I – Validação da organização.

5 CONSIDERAÇÕES FINAIS

Neste capítulo são apresentadas as conclusões e recomendações finais.

5.1 CONCLUSÃO

O presente trabalho atendeu a todos os objetivos propostos, desde o objeto geral ao específico, e respondeu também à hipótese de pesquisa.

Constatou-se que a SED não utiliza nenhum documento de ERS em seu processo, e como se viu na Fundamentação Teórica, tal artefato está muito atrelado à qualidade do software, pois é constatado mediante a prática de um processo, onde um processo de qualidade leva a um produto de qualidade, uma vez que utiliza métodos para obtenção de resultados. Tal deficiência, quanto ao não uso de um documento de ERS, prenuncia em parte a não prática de um processo, que é constatada posteriormente em entrevista, pois não há de fato um processo formal estabelecido, e sim, informal, que se molda conforme demanda e necessidade.

Apesar de não ser escopo do presente estudo, constatou-se na prática também a inexistência de métodos de construção de software, prescritivos ou ágeis, também influenciados por um processo informal.

Outro fato que se percebe, e talvez seja a causa raiz de certas deficiências, é que o cliente não é uma fábrica de software como as que se vê no mercado, e pelo seu negócio chave ser a Educação, a organização não está estruturada para gerir processos profissionais de engenharia de software, tanto em desenvolvimento quanto para aquisição deste serviço de terceiros.

Tal desencontro entre problemas de execução e gestão de serviços de desenvolvimento de software, talvez se deva também a questões ligadas a licitações e contratos inerentes a órgão públicos, pois podem permitir, por vícios tecnológicos, contratações a baixo de níveis de qualidade aceitáveis, ou ainda permitir subdimensionamento de recursos e demandas, geralmente originárias de termos de referências e especificações técnicas deficientes e mal escritas, inviabilizando a execução e fiscalização eficiente e eficaz

de contratos por ambas as partes, pois as métricas e níveis de acordo de serviços não ficam claros, dando margem a contratos problemáticos.

Analogamente falando, em termos práticos, não se constrói um prédio sem planejamento, métricas, planos, métodos e técnicas bem definidos, tão pouco, o pedreiro cumpre papel de mestre de obra ou engenheiro, projetando e coordenando atividades ao mesmo em que às executa, e isso é constatado na organização, quando vê-se que sua equipe de software é pequena, sem estrutura de apoio e organizacional, e os desafios demandados pela SED são grandes para o grupo que lá se encontra atualmente.

Conclui-se que pode-se chegar com o passar do tempo a produtos de software entregáveis, contudo, com qualidade duvidosa, prazos de entregas comprometidos e custos elevados de desenvolvimento e manutenção.

Acredita-se que o uso de um documento de ERS na SED como regra padrão, é uma boa e simples prática que ajuda a melhorar de alguma forma a qualidade do software, porém, para o estudo de caso em questão, ele não passará de uma ação isolada, ainda que positiva, pois a organização deverá tomar outras medidas profissionalizantes para melhorar seus processos e métodos, a fim de se obter melhores prazos, custos e qualidade.

5.2 RECOMENDAÇÕES

Para trabalhos futuros, recomenda-se estudos na organização para implementação de métodos e processos voltados a construção de software a partir de modelos de mercado, o que melhorará o processo, introduzindo controles de qualidade, e provavelmente melhorará os prazos de entrega, reduzirá custos e elevará a qualidade dos produtos entregues, consequentemente reduzindo as reclamações de cliente e melhorando a imagem da TI da organização para com os usuários, que perceberam mais valor nos serviços entregues.

REFERÊNCIAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 6023:** Informação e documentação – referências e elaboração. Rio de Janeiro, 2002. Disponível em: < www.habitus.ifcs.ufrj.br/pdf/abntnbr6023.pdf>. Acesso em: 20 janeiro 2013.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 12207:** Tecnologia de informação – Processos de ciclo de vida de *software*. Rio de Janeiro, 1998. Disponível em: <ftp://ftp.unilins.edu.br/caldas/Engenharia de Software/Normas_de_Qualidade/NBR 12207 - Tecnologia de informacao - Processos de ciclo de vida de software.v2.pdf>. Acesso em: 11 fevereiro 2013.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 9126:1:** Engenharia de software – Qualidade de Produto. Rio de Janeiro, 2003. Disponível em: <http://portal2.tcu.gov.br/portal/page/portal/TCU/comunidades/biblioteca_tcu/ABNT NBR ISOIEC 9126-1.pdf >. Acesso em 10 janeiro de 2013.

FERNANDES, Dantes Guilherme Figueiredo. **Um estudo para automatizar com a linguagem BPEL o processo de levantamento de infomações de uma empresa desenvolvedora de software.** 2010. 137 f. Monografia (Graduação em Sistemas de Informação) – Universidade do Sul de Santa Catarina, Palhoça, 2010.

IEEE Std. 830-1998; **IEEE Recommended Pratice for Software Requirements Specifications**. Institute of Electrical and Electronics Engineers. 1998, 37p. ISBN 0-7381-0332-2. Disponível em: <http://www.math.uaa.alaska.edu/~afkjm/cs401/IEEE830.pdf>. Acesso em: 12 janeiro 2013.

PAULA FILHO, Wilson de Pádua. **Engenharia de software***:* fundamentos, métodos e padrões. Rio de Janeiro: LTC, 2001.

PETERS, James F.; PEDRICZ, Witold. **Engenharia de Software**. tradução de Ana Patrícia Garcia. Rio de Janeiro: Campus, 2001.

PRESSMAN, Roger S. **Engenharia de Software**. São Paulo: Pearson Makron Books, 1995.

SANTA CATARINA. Constituição (1989). **Constituição do Estado de Santa Catarina**. Florianópolis, SC: Alesc, 1989. Disponível em: < http://www.alesc.sc.gov.br/portal/legislacao /docs/constituicaoEstadual/CESC%202012%20-%2063%20e%2064%20emds.pdf>. Acesso em: 10 fevereiro 2013.

SANTA CATARINA. **Lei Complementar nr. 289, de 28 de fevereiro de 2015**. Estabelece modelo de gestão para a Administração Pública Estadual e dispõe sobre a estrutura organizacional do Poder Executivo. Disponível em: <200.192.66.20/alesc/docs/2005/ 284_2005_lei_ complementar.doc>. Acesso em: 11 de fevereiro 2013.

SANTA CATARINA. **Lei Complementar nr. 534, de 20 de abril de 2011**. Dispõe sobre o modelo de gestão e a estrutura organizacional da Administração Pública Estadual e estabelece outras providências. Disponível em: <200.192.66.20/alesc/docs/2011/534_2011_lei_ complementar.doc>. Acesso em: 11 de fevereiro 2013.

SILVA, Lyrene Fernandes. **Uma estratégia orientada a aspectos para modelagem de requisitos**. 2006. 222 f. Tese (Programa de Pós-Graduação em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro. 2006. Disponível em: <www-di.inf.puc-rio.br/~julio/lyrene-tese.pdf>. Acesso em: 25 fevereiro 2013.

SOMMERVILLE, Ian. **Engenharia de requisitos de software**. Tradução André Maurício de Andrada Ribeiro. 6ª ed. São Paulo: Addison Wesley, 2003

SOUZA, Graziela Goedert de. **Estudo de engenharia de software e métodos para web, com aplicação no desenvolvimento de um protótipo.** 2009. 126 f. Monografia (Graduação em Ciência da Computação) – Universidade do Sul de Santa Catarina, Palhoça, 2009.

RATIONAL SOFTWARE CORPORATION. **Rational Unified Process**: RUP 2002.05.00 Português. Disponível em: <http://www.wthreex.com/rup/ portugues/index.htm>. Acesso em: 28 fevereiro 2013.

UNIVERSIDADE DO SUL DE SANTA CATARINA. Pró-Reitoria Acadêmica. Programa de Bibliotecas. **Trabalhos acadêmicos na Unisul**: apresentação gráfica para tcc, monografia, dissertação e tese. 3. ed. rev. e ampl. Palhoça: Ed. Unisul, 2010.

VOLERE. **Modelo de especificação de requisitos**. 14. ed. 2009. Disponível em: <http://www.volere.com.uk/>. Acesso em 20 de fevereiro 2013.

APÊNDICES

APÊNDICE I – Validação da organização

UNIVERSIDADE DO SUL DE SANTA CATARINA
Av. José Acácio Moreira, 787 - Bairro Dehon - Cx. Postal 370
88704-900 - Tubarão - SC
Fone: (48) 621-3000
Campus Trajano - Florianópolis

VALIDAÇÃO DA ENTREVISTA
E ARTEFATO DE ERS

Florianópolis/SC, 26 de Março de 2013.

Pela presente, o pesquisador infra citado, do curso de pós graduação de Especialização em Projetos de Engenharia de Software, vem por meio desta solicitar a vossa senhoria a validação do relatório submetido (partes abaixo especificadas e já apresentadas) relativo a monografia que esta sendo elaborada a partir de estudo de caso nesta instituição.

Constam em anexo para validação:

- Capítulos dos métodos e resultados obtidos
- Artefato de ERS – Especificação de Requisitos de Software

Pesquisador
Alexsandro Rafael de Oliveira

Validação da Instituição Pesquisada
Secretaria de Estado da Educação de Santa Catarina
Francisco Reis Von Hertwig
Gerente de Tecnologia da Informação
e Governança Eletrônica
Matrícula 238.355-1-01

APÊNDICE II – Documento Modelo de ERS para SED

(em dezessete páginas adicionais)

ERS
ESPECIFICAÇÃO DE REQUISITOS DE SOFTWARE

IDENTIFICAÇÃO RESUMO

Função a ser desenvolvida em:

☐ Sistema existente

Nome: _____
Módulo: _____

☐ Sistema não existente

☐ Independente
Denominação: _____

☐ Incorporar em sistema

Nome: _____

Nr Rev:	Data:

Controle Revisão

Existentes	1	2	3	4	5	6	7	8	9	10
Nr Rev										
Data										
Existentes	11	12	13	14	15	16	17	18	19	20
Nr Rev										
Data										

SUMÁRIO

1. INTRODUÇÃO

a. Descrever nesta seção uma apresentação geral, superficial, e norteadora da leitura completa deste documento.

1.1. Propósitos

a. Descrever nesta seção de forma sucinta as motivações que levam a desenvolver a funcionalidade/software/sistema. Orientar-se por situações problemas, justificativas e benefícios. Isto estabelece uma relação entre necessidades e metas.

1.2. Escopo

a. Descrever nesta seção de forma sucinta, como resumo, "o quê" pretende-se desenvolver, e se necessário "o quê não" se pretende desenvolver.

1.3. Definições, acrônimos e abreviaturas

a. Descrever nesta seção as principais definições, acrônimos e abreviaturas utilizados ao longo do documento, de forma a não haver erros de interpretações.

1.4. Referências

a. Descrever aqui uma relação de referências externas existentes que devem, ou podem ser consultadas para completar o entendimento do referido escopo.
b. Estruturar com denominação adequada, com titulação, numeração, data, fontes, etc.

1.5. Visão geral

a. Descrever nesta seção como está estruturado o presente documento, de forma a propiciar ao futuro leitor uma rápida visão sobre o que consta neste documento;

2. DESCRIÇÃO GERAL

a. Descrever nesta seção é descrita a parte conceitual do produto de software que será desenvolvido. É o entendimento técnico da situação problema que deve ser resolvida através do software.

2.1. Perspectivas do produto

a. Descrever nesta seção a relação que produto a ser desenvolvido tem para com outros sistemas. Se é único e independente, se é parte de outro sistema, se interagem com outros sistemas, etc.
b. Exemplos de perspectivas: de sistema, de usuário, de hardware, de software, de comunicação, etc.
c. Pode ser utilizado um diagrama de blocos, interconexões, interfaces para compreensão destas perspectivas.

2.2. Funções do produto

a. Descrever nesta seção um resumo das principais funções que o produto de software irá executar;
b. Descrever de forma lógica, podendo utilizar diagramas para auxiliar na interpretação;

2.3. Características do usuário

a. Descrever nesta seção qual é o publico que será atendido, seu nível educacional, experiência e conhecimento técnico, volume de usuários, onde estão localizados, etc. Isto servirá para validação e alinhamento posterior dos requisitos;
b. Pode-se utilizar diagramas e tabelas para auxiliar no estabelecimento de perfis de público alvo.

2.4. Restrições

a. Descrever nesta seção as restrições nas quais o produto que será desenvolvido não tem o propósito de atender, como, tipos de equipamentos ou sistemas operacionais nos quais não "rodará", quais são as fronteira e limites exatos para se buscar soluções dentro destes limites;
b. Descrever limitações impostas por tipo de tecnologia ou conhecimento para justificar o não uso de outros meios relativos;
c. Descrever limites impostos por usuários e cliente, como políticas, legislações, etc;
d. Descrever restrições de orçamento e de prazo, como por exemplo, data limite em função de algum evento físico externo.

2.5. Suposições e dependências

a. Descrever nesta seção situações que devem ser consultadas e esclarecidas, ou que ainda causam dependência na execução do desenvolvimento, em qualquer uma de suas fases, servindo como premissas para monitoramento e controle;
b. Descrever também os recursos necessários que deverão ser disponibilizados para execução do desenvolvimento, desde mão de obra a licenças de sistema operacional ou de desenvolvimento;
c. Descrever também premissas que o cliente deve cumprir para não impactar no produto final, como servidores, equipamentos, licenças para ambiente de produção.
d. Descrever consultas adicionais a leis e decisões políticas que podem influenciar no produto que será entregue.
e. Este item não caracteriza restrições, mas situações que podem impactar na entrega ou conclusão do desenvolvimento. Podem ser consideradas como acordos de projeto.

2.6. Modelagem de negócio

a. Descrever nesta seção como é o fluxo do processo de negócio que o software a ser desenvolvido deverá desempenhar;
b. Se necessário, desenhar o antes, e o depois do processo de negócio;
c. Se necessário, desenhar o fluxo de interação com outros sistemas;
d. Utilizar diagrama de fluxo preferencialmente;
e. Como referência, a fim de auxiliar no desenho, no Anexo III constam notações de modelagem de processo de negócio (BPMN – *Notation Modeling Process Business*). Contudo, não há pretensão de ensinar o uso do BPMN, devendo-se buscar literatura e instruções adicionais para correta compreensão e uso.

2.6.1. Casos de Uso

a. Descrever nesta seção os Casos de Uso que compõem o escopo do software, que são os momentos e funções de interação do software em relação a pessoas e outros sistemas. Isto auxiliará a definir o escopo e fronteiras do software que será desenvolvido.
b. Utilizar exemplos de referência constante do Anexo IV para esquematizar a interpretação em nível de desenho, Contudo, não há pretensão de ensinar o uso dos Casos de Uso, devendo-se buscar literatura e instruções adicionais para correta compreensão e uso.
c. A fim de complementar, além de um desenho de interação, uma descrição do caso de uso complementa o entendimento;
d. Dê um número de identificação único ao caso de uso, exemplo: CSU.0001, isto facilitará a organização, rastreabilidade, vinculação e gerência de requisitos.

CSU.ID	Descrição

3. ESPECIFICAÇÕES DE REQUISITOS

a. Descrever nesta seção como serão apresentados os requisitos, sua tipificação, classificação e agrupamentos.
b. As seções anteriores servem como fonte de requisitos para desdobramento das seções anteriores para cobertura do domínio de problema.

3.1. Requisitos

a. Descrever nesta seção:
    a. Ressalvas:
        i. Deve descrever os requisitos do produto de software, e não o processo de construção;
        ii. Não deve-se incluir na ERS questões como custo, prazo, repasse de tecnologia, métodos de desenvolvimento, garantia da qualidade, critérios de validação e verificação, e procedimentos de aceitação. Todos estes itens devem constar de outros documentos, e não de uma ERS;
        iii. Não deve-se preocupar-se com questões estruturais internas de construção software, devendo os requisitos ser declarados de um ponto de vista puramente externo.
    b. Recomendações ao escrever uma ERS:
        i. Correta – Todo requisito presente realmente é um requisito do produto a ser construído;
        ii. Precisa – Todo requisito presente possui apenas uma única interpretação, aceita tanto pelos desenvolvedores quanto pelos usuários chaves;
        iii. Completa – Reflete todas decisões de especificação que foram tomadas;
        iv. Consistente – Não há conflitos entre nenhum dos subconjuntos de requisitos presentes;
        v. Priorizada – Cada requisito é classificado de acordo com a sua importância, estabilidade e complexidade;
        vi. Verificável – Todos requisitos são verificáveis;
        vii. Modificável – Sua estrutura e estilo permitem a mudança de qualquer requisito, de forma fácil, completa e consistente;
        viii. Rastreável – Permite a fácil determinação dos antecedentes e consequência de todos os requisitos.
    c. Descrever, procurando separar, os requisitos entre funcionais e não funcionais;
    d. Ao preencher as listas a seguir (RF e RNF), descrever os requisitos coletados do cliente, de forma organizada, estruturada, identificados por numerador único, por exemplo: RF.0001. (Requisito Funcional), e RNF.0001 (Requisito Não funcional);
        i. Destacar o que é estímulo (entrada), o que é resposta (saída), e quais funções, ações e comportamentos são desejados.
        ii. Descrever num nível de detalhes que usuários chaves possam validar o texto, e que equipe desenvolvimento possa construir em cima deste texto;
        iii. No campo rastreabilidade, procurar relacionar, sempre que possível, a que ID está ligado. Podem ser vários relacionamentos de rastreabilidade. Isso auxiliará na gestão de mudança para melhorias, alterações e manutenções.
        iv. No campo tipo, é fornecido no Anexo I uma lista de tipificação do requisito. Posteriormente, pode-se gerar uma lista de requisitos por tipificação. Um requisito pode não ter uma tipificação, ou se não houver tipificação declarada na lista do Anexo I, ela pode ser incrementada. Da mesma forma, um requisito pode se enquadrar em mais de uma tipificação, o que não o enquadra como ambíguo.
    e. Para os RNF, também chamados de específicos ou especiais, cujo o cliente não tem esclarecimento técnico para tias considerações, é fornecida uma lista orientativa baseada em atributos de qualidade, que está localizada no Anexo I desta ERS.
        i. Além da do Anexo I, fornece-se aqui algumas dimensões que podem ser consideradas, como: estilo, aparência, sensações, linguagem, condições físicas e ambientais, integrações com outros sistemas, metáforas do público alvo, condições de liberação, instalação e treinamento, internacionalização, cultura, política, legislações.

3.2. Lista de Requisitos Funcionais - RF

RF.ID.	Descrição (regras, comportamentos, ações, eventos, respostas)	Rastreabilidade	
		CSU.ID	RNF.ID

3.3. Lista de Requisitos Não Funcionais – RNF

RNF.ID.	Descrição	Rastreabilidade	
		CSU.ID	RF.ID

3.4. Gerenciamento de Requisitos

a. Os requisitos supracitados devem estar completos, como se fossem uma fotografia do que foi levantado no momento. Contudo, é fato que alguns requisitos mudam, por isso, dever-se-á aplicar um método de controle, acompanhamento e gerência de requisitos.

b. Revisões formais deste plano devem ser realizadas, identificadas e numeradas, criando uma trilha de auditoria das modificações. Isto servirá para evidenciar o esforço de desenvolvimento, e as mudanças de acordo ou necessidades, além de fazer circular formalmente as mudanças às partes envolvidas.

GESTÃO DE REQUISITOS						
Controle de Alteração		Classe		Controle da ERS		
ID Mudança	ID dos Requisitos	Introduzidos	Modificados	Data	Revisão	GERENCIAMENTO DE MUDANÇAS

4. CONSIDERAÇÕES FINAIS

4.1. Temas abertos

a. Descrever nesta seção temas pertinentes que foram discutidos e não se obteve uma conclusão, recomendando estudos e discussões posteriores, e validando formalmente questões que foram debatidas mas não foram definidas.

4.2. Soluções disponíveis

a. Descrever nesta seção soluções que podem ser consultadas, referendas, copiadas legalmente, reutilizadas.

4.3. Novos problemas

b. Descrever nesta seção novos problemas que podem ser encontrados ou enfrentados a partir da implementação desta ERS, como alerta a riscos, capacitação, aquisições, etc.

4.4. Planos complementares

a. Descrever nesta seção, documentos externos que devem ser referenciados para consulta complementar, a existência de planos de projetos, plano de gerenciamento de projeto, plano de desenvolvimento, plano de custos, plano de riscos, plano de comunicações, plano de transição, plano de entrega e capacitação.

4.5. Atefatos adicionais

Em caso de adição de documentos complementares, estes devem constar no Anexo ou Apêndice da seguinte forma:

Anexos:..... Documentos coletados, não produzidos a partir desta ERS;
Apêndices:..... Documentos produzidos a partir desta ERS.

Sugere-se que, se utilizado algum tipo de protótipo de software, de tela ou relatório, o mesmo faça parte deste item, ou mesmo referenciado. Protótipos ajudam o cliente a validar os requisitos escritos.

5. DIREITOS AUTORIAIS

Este modelo serve de *Tamplate* orientativo à confecção de uma ERS.

Em caso de não preenchimento de informações, não deixar em branco para não causar um aspecto de "esquecimento de preenchimento", corrigindo tal desvio com notações do tipo: "NA – Não Aplicável".

Tal documento pode ser adaptado e customizado, sugerindo apenas que se atualize a versão do mesmo em seu cabeçalho, devendo sempre, em qualquer versão do mesmo, manter citação de o crédito ao autor:

"modelo criado por Alexsandro Rafael de Oliveira"

ANEXO I – LISTA DE REFERÊNCIA DE TIPIFICAÇÃO DE REQUISITOS

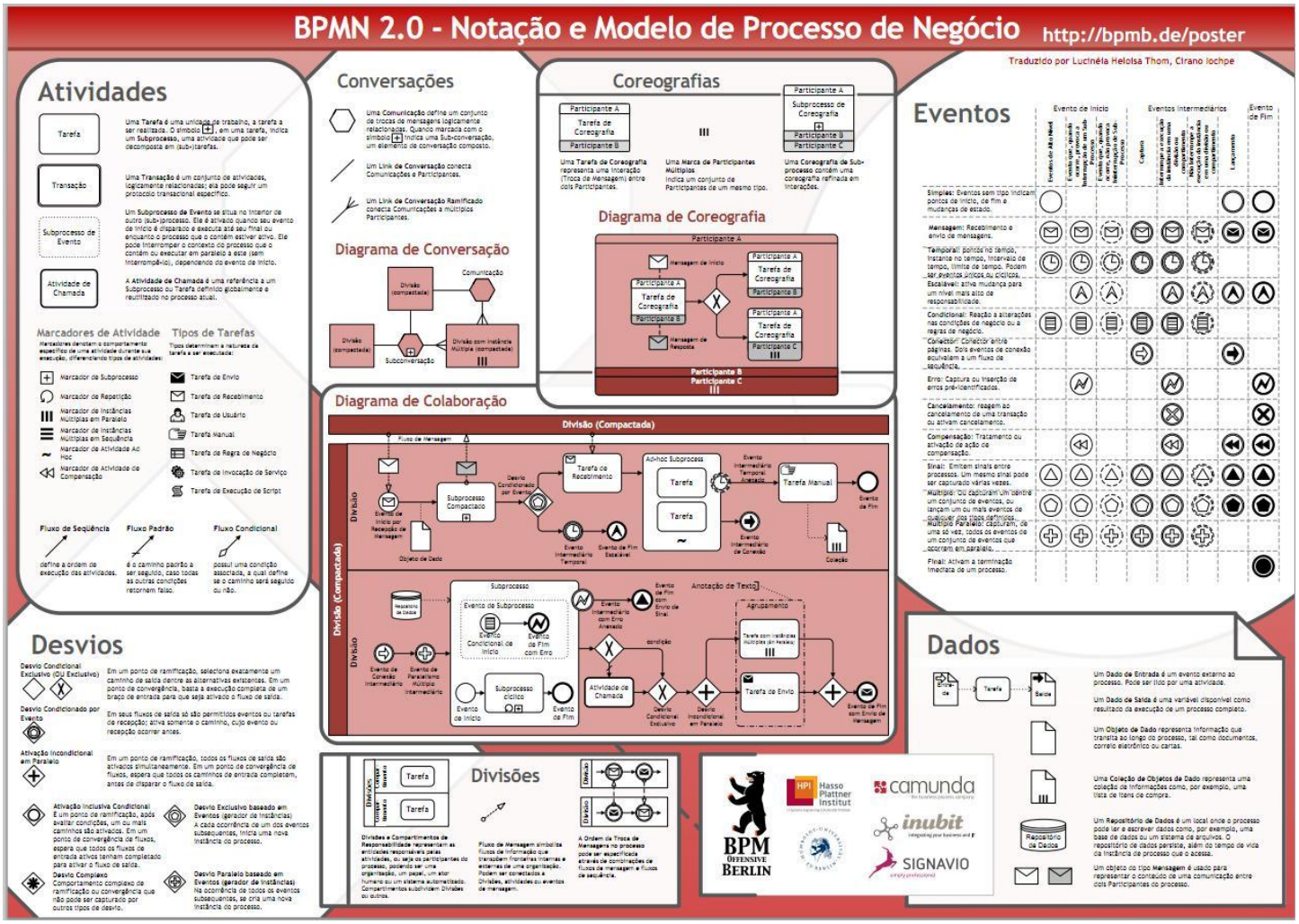
Tipificação de requisitos	
TP.ID	Descrição
1	Requisitos de processo e dados
2	Requisitos de desempenho e qualidade
3	Requisitos de banco de dados
4	Requisitos de projeto
5	Requisitos de interfaces externas
6	Requisitos de atributos de sistema

ANEXO II – LISTA DE REFERÊNCIA DE ATRIBUTOS DE QUALIDADE

ATRIBUTOS DE QUALIDADE				
1	Funcionalidade:	Capacidade do software em prover funções que atendam às necessidades explícitas e implícitas.	Considerado SIM	NÃO
1.1	Adequação:	Capacidade de prover funções para tarefas e objetivos do usuário especificados.		
1.2	Acurácia:	Capacidade de software de prover, com grau de precisão necessário, resultados conforme esperados.		
1.3	Interoperabilidade:	Capacidade de interagir com um ou mais sistemas especificados.		
1.4	Segurança de acesso:	Capacidade de proteger informações e dados contra pessoas ou sistemas não autorizados, e permitir acesso às pessoas ou sistemas autorizados.		
1.5	Conformidade relacionada à funcionalidade:	Capacidade de estar de acordo com normas, convenções ou regulamentações previstas em leis relacionados à funcionalidade.		
2	Confiabilidade:	Capacidade do software de manter um nível de desempenho especificado.	Considerado SIM	NÃO
2.1	Maturidade:	Capacidade de evitar falhas decorrentes de defeitos no software.		
2.2	Tolerância a falhas:	Capacidade de manter um nível de desempenho especificado em casos de defeitos ou de violação.		
2.3	Recuperabilidade:	Capacidade de restabelecer um nível de desempenho em casos de defeitos ou de violação.		
2.4	Conformidade relacionada à confiabilidade:	Capacidade de estar de acordo com normas, convenções ou regulamentações relacionadas à confiabilidade.		
3	Usabilidade:	Capacidade do software de ser compreendido, aprendido, operado e atraente ao usuário.	Considerado SIM	NÃO
3.1	Inteligibilidade:	Capacidade de possibilitar ao usuário compreender se o software é apropriado e como ele pode ser usado para tarefas e condições de uso.		
3.2	Apreensibilidade:	Capacidade de possibilitar ao usuário aprender sua aplicação.		
3.3	Operacionabilidade:	Capacidade de possibilitar ao usuário operá-lo e controlá-lo.		
3.4	Atratividade:	Capacidade de ser atraente ao usuário.		
3.5	Conformidade relacionada à usabilidade:	Capacidade de estar de acordo com normas, convenções, guias de estilo ou regulamentações relacionadas à usabilidade.		
4	Eficiência:	Capacidade do software de apresentar desempenho apropriado, relativo à quantidade de recursos usados.	Considerado SIM	NÃO
4.1	Comportamento em relação ao tempo:	Capacidade de fornecer tempos de resposta e de processamento, e taxas de transferências apropriadas.		
4.2	Utilização de recursos:	Capacidade de usar tipos e quantidades apropriados de recursos.		

4.3	Conformidade relacionada à eficiência:	Capacidade de estar de acordo com normas e convenções relacionadas à eficiência.		
5	Manutenibilidade:	Capacidade de ser modificado. As modificações podem incluir correções, melhoras ou adaptações devido a mudanças no ambiente e nos seus requisitos ou especificações funcionais.	Considerado SIM	NÃO
5.1	Modificabilidade:	Capacidade de permitir que uma modificação especificada seja implementada.		
5.2	Estabilidade:	Capacidade de evitar efeitos inesperados decorrentes de modificações no software.		
5.3	Testabilidade:	Capacidade de permitir que o software, quando modificado, seja validado.		
5.4	Conformidade relacionada a manutenibilidade:	Capacidade de estar de acordo com normas ou convenções relacionadas à manutenibilidade.		
6	Portabilidade:	Capacidade do software de ser transferido de um ambiente para outro.	Considerado SIM	NÃO
6.1	Adaptabilidade:	Capacidade de ser adaptado para diferentes ambientes, sem necessidade de aplicação de outras ações ou meios além daqueles fornecidos para essa finalidade pelo software considerado.		
6.2	Capacidade para ser instalado:	Capacidade para ser instalado em um ambiente especificado.		
6.3	Coexistência:	Capacidade de coexistir com outros produtos de software independentes, em uma ambiente comum, compartilhando recursos comuns.		
6.4	Capacidade para substituir:	Capacidade de ser usado a outro produto de software, com o mesmo propósito e no mesmo ambiente.		
6.5	Conformidade relacionada à portabilidade:	Capacidade de estar de acordo com normas ou convenções relacionadas à portabilidade		
7	Qualidade em uso:	Capacidade do software de permitir que usuários atinjam metas especificadas com eficácia, produtividade, segurança e satisfação.	Considerado SIM	NÃO
7.1	Eficácia:	Capacidade de permitir que usuários atinjam metas com acurácia e completitude.		
7.2	Produtividade:	Capacidade de permitir que seus usuários empreguem quantidade apropriada de recursos em relação à eficácia obtida.		
7.3	Segurança:	Capacidade de apresentar níveis aceitáveis de riscos de danos a pessoas, negócios, software, propriedades ou ao ambiente.		
7.4	Satisfação:	Capacidade de satisfazer usuários, em um contexto de uso especificado.		

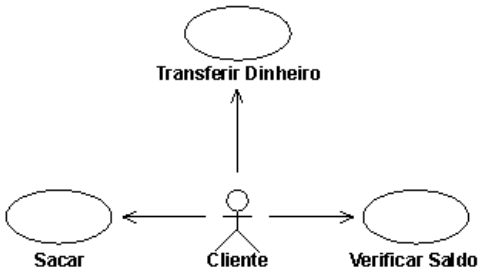
ANEXO III – NOTAÇÕES BPMN



Fonte: www.bpmb.de/images/BPMN2_0_Poster_PT.pdf

ANEXO IV – EXEMPLOS DE CASO DE USO

Ilustração 1 - Exemplo de Caso de Uso



Em um caixa eletrônico, o cliente pode, por exemplo, retirar dinheiro de uma conta, transferir dinheiro para uma conta ou verificar o saldo de uma conta. Essas funções correspondem a fluxos que você pode representar com casos de uso.

Fonte: Rational Software Corporation (RUP)

APÊNDICE

- TÍTULO -

ANEXO I – Notações BPMN 2.0