Uma Taxonomia da Pesquisa na Área de Engenharia de Requisitos

Paulo Sérgio Naddeo Dias Lopes

DISSERTAÇÃO DE MESTRADO

APRESENTADA AO INSTITUTO DE MATEMÁTICA E ESTATÍSTICA DA UNIVERSIDADE DE SÃO PAULO COMO REQUISITO PARCIAL À OBTENÇÃO DO TÍTULO DE MESTRE EM CIÊNCIA DA COMPUTAÇÃO

Curso: **Mestrado em Ciência da Computação**
Área de Concentração: **Engenharia de Software**
Orientador: **Prof. Dr. Hernán Enrique Astudillo Rojas**

— São Paulo, 13 de Março de 2002 —

Uma Taxonomia da Pesquisa na Área de Engenharia de Requisitos

Paulo Sérgio Naddeo Dias Lopes

Este exemplar corresponde à redação final da dissertação devidamente corrigida, defendida por Paulo Sérgio Naddeo Dias Lopes e aprovada pela comissão julgadora.

São Paulo, 13 de Março de 2002.

Banca examinadora:

Prof. Dr. Hernán Enrique Astudillo Rojas (IME-USP)

Prof. Dr. Marcos Ribeiro Pereira Barreto (EP-USP)

Prof. Dr. Alan Mitchell Durham (IME-USP)

Agradecimentos

À minha esposa Renata e minha filha Gabrielle por sua compreensão em relação aos longos momentos de ausência. Ao amigo Jorge, por sua extrema paciência em revisões e discussões sobre o conteúdo do trabalho. A meus pais, sem cuja perseverança esta realização não seria possível. A todos que, de alguma forma, influenciaram, com suas opiniões e colocações, as argumentações aqui apresentadas.

Abstract

Requirements Engineering is in the realm of Software Engineering and is concerned with the study of requirements, their specification, documentation, validation and negotiation among all stakeholders of a system. In this work it is proposed a taxonomy of the research in Requirements Engineering. Its goal is to have it ontologically organized, in the sense that it is built up around the activities conducted during the requirements process. To achieve that, the basic concepts that set the scene for the discipline are discussed, the importance of the requirements process is highlighted, the main responsibilities during each of the activities in it are pointed out and its main participants are identified. Finally, the dimensions that affect a classification are discussed and a faceted scheme for classifying the research in Requirements Engineering is proposed.

Keywords: Requirements Engineering, requirements, requirements process, software requirements, software specification

Resumo

Engenharia de Requisitos é uma disciplina inserida no contexto da Engenharia de Software e relaciona-se ao estudo dos requisitos, sua especificação, documentação, validação e negociação entre todos os *stakeholders* de um sistema. Neste trabalho, propõe-se uma taxonomia da pesquisa em Engenharia de Requisitos, organizada ontologicamente, isto é, estruturada de acordo com as atividades conduzidas durante o processo de requisitos. Para isso, são discutidos os conceitos básicos da disciplina, são apresentados argumentos sobre a importância do processo de requisitos, são apontadas as principais responsabilidades no contexto de cada uma das atividades conduzidas e, por fim, são identificados os principais participantes do processo. Finalmente, são discutidas as dimensões que afetam uma possível classificação e apresenta-se um esquema facetado para classificar a pesquisa na área de Engenharia de Requisitos.

Palavras-chave: Engenharia de Requisitos, requisitos, processo de requisitos, requisitos de software, especificação de software

Sumário

Lista de Figuras

Lista de Tabelas

1. Introdução

Desde 1950, com o avanço da tecnologia e a construção de computadores cada vez mais poderosos, um grande desafio surgiu ao desenvolvimento de *software*. O declínio dos preços do *hardware* estava transformando aquela máquina milionária no agente da nova era: a era da informação. Apesar de todas as promessas, elas não poderiam realizar-se, a menos que *software* fosse construído para materializa-la. Assim, visando tornar possível o uso das, cada vez mais, novas e poderosas máquinas, *software* passou a absorver cada vez mais complexidade do mundo real.

Esse cenário transformou o *software* na locomotiva da nova economia [PRESSMAN 97]. *Software* está em quase todas as coisas: de fornos a celulares, de carros a máquinas de diagnóstico computadorizadas, de computadores pessoais a sistemas industriais de controle. Todos esses sistemas precisam adequar-se a um alto grau de complexidade, que tem que ser tratado e não pode ser simplificado, como ensina Brooks [BROOKS 87]. Logo *software* transformou-se na parte mais cara dos sistemas computadorizados complexos, sendo desenvolvido com baixa qualidade, com pouca previsibilidade de custo e recursos, colocando em risco o avanço da tecnologia de *hardware*.

A necessidade de se encontrar métodos que pudessem ajudar na evolução do processo de construção de *software*, culminaram com o desenvolvimento da Engenharia de Software. Essa nova e disciplinada abordagem ao processo de construção de *software* trouxe consigo métodos e técnicas que ajudaram na administração da complexidade inerente do *software*, conduzindo-o através de seus diferentes níveis de abstração de forma controlada, passível de administração e mensurável.

Contudo, apesar dos inegáveis avanços – e eles são evidentes no trabalho de Yourdon [YOURDON 78], Ross [ROSS 77a], DeMarco [DEMARCO 79], Rumbaugh [RUMBAUGH 91], Booch [BOOCH 94] e Jacobson [JACOBSON 92], para citar alguns – ainda existem pontos que permanecem obscuros, demandando atenção da comunidade acadêmica. Um desses pontos é, certamente, requisitos de *software*. Em 1979, o *Gorvernment Accounting Office* (GAO) nos USA publicou um relatório no qual afirmava que menos de 2% (dois por cento) dos investimentos feitos no desenvolvimento de *software* resultaram em *software* que atendia seus requisitos [DAVIS 93]. Na recente conferência de Engenharia de Requisitos, ICRE2000, foi conduzida uma seção de discussão com o tema "*Por que é tão difícil introduzir os resultados da pesquisa na área de Engenharia de Requisitos na prática da Engenharia de Requisitos?*". Após duas horas de intenso debate, o ponto comum entre todos os debatedores, foi de que a pesquisa ainda não está suficientemente madura para oferecer uma base sólida para a utilização das técnicas prescritas na prática. A pesquisa de Engenharia de Requisitos tem, aproximadamente, 10 anos e, historicamente, novas tecnologias saem do estado da arte para o estado da prática em aproximadamente 20 anos. Muito há ainda por fazer.

Assim, qualquer esforço no sentido de minimizar o tempo de familiarização de novos estudantes ou pesquisadores com os conceitos da área seria bem-

vindo. Uma idéia nessa direção seria criar um mecanismo que facilitasse a seleção dos textos a serem lidos. Classificá-los de acordo com critérios específicos poderia, não só auxiliar estudantes e pesquisadores, mas também profissionais da área em busca de soluções ou idéias para resolver problemas específicos, a mais rapidamente atingir seus objetivos.

Neste trabalho meu objetivo é fazer um compêndio sobre a pesquisa na área de Engenharia de Requisitos, relacionando as principais áreas de pesquisa, classificadas de acordo com o processo de condução da Engenharia de Requisitos. O trabalho está organizado da seguinte forma: na seção 2 são apresentados os objetivos deste trabalho. Na seção 3 é discutido o contexto histórico no qual a pesquisa foi conduzida. A seção 4 explica o processo de Engenharia de Requisitos, estabelecendo a terminologia que será usada, na seção 5, utilizo a terminologia apresentada para discutir a taxonomia da pesquisa de Engenharia de Requisitos até hoje. Finalmente, na seção 6, discuto as conclusões deste trabalho e a perspectiva para trabalhos futuros.

2. Objetivos e Audiência

O principal objetivo deste trabalho é classificar a pesquisa na área de Engenharia de Requisitos criando uma taxonomia. Essa taxonomia organizará o conhecimento de modo que:

- Alguém, novo ao tema, possa obter uma visão geral de suas principais sub-áreas e como elas estão relacionadas;
- Pesquisadores e estudantes, entre outros, familiarizados com os detalhes da área, possam vê-la a partir de outra perspectiva, com esperanças de que ela possa iluminar áreas obscuras da pesquisa;
- Seja ontologicamente organizada, no sentido de que deve adequar-se naturalmente à condução do processo na prática.

Para atingir esses objetivos, pretendo:

- Apresentar as origens da Engenharia de Requisitos, identificando o contexto que motivou a pesquisa na área;
- Discutir detalhadamente o processo de Engenharia de Requisitos, apresentando as atividades e as tarefas realizadas no contexto de cada uma delas, as dificuldades inerentes à sua realização e os principais problemas enfrentados;
- Apresentar, a partir da bibliografia consultada, uma visão geral dos tópicos abertos na pesquisa.

3. Contexto Histórico

Foi em 1969, na Conferência NATO *Science Committee*, que Fritz Bauer, no contexto da crise do *software*, inicialmente propôs o termo "Engenharia de Software" para identificar "*o estabelecimento e uso de sólidos princípios de engenharia para obter economicamente software confiável e que trabalhe de forma eficiente em máquinas reais*" [PRESSMAN 97].

Apesar da introdução da nova terminologia, muitas dúvidas persistiram quanto a sua adequação, ou mesmo em relação ao que deveria ser feito para conduzi-la. A definição de Bauer é, até hoje, questionada e apoiada. Ludewig argumenta que há mais diferenças que similaridades entre o que se conhece como engenharia e as atividades desenvolvidas no contexto da construção de *software*, considerando a terminologia inadequada [LUDEWIG 96]. Mary Shaw considera que a nova terminologia surgiu para chamar atenção ao problema de desenvolvimento de *software* e que a comunidade acadêmica, mais que a comunidade de desenvolvimento industrial, direcionou-se de apenas escrever programas, para a análise dos problemas de grandes sistemas distribuídos de *software* e dados [SHAW 96].

Por mais inadequada que a terminologia tenha sido, a pesquisa evoluiu na comunidade acadêmica e toda uma base de conhecimento foi produzida a partir da pesquisa na área de Engenharia de Software. Em 1970, Winston Royce propôs o modelo linear seqüencial para o processo de *software* [PRESSMAN 97]. Basicamente, o modelo dividia o processo de construção de *software* em quatro atividades: análise, projeto, codificação e teste.

Em 1975, Brooks publicou um dos marcos no campo da Engenharia de Software. "*The Mythical Man-Month*" [BROOKS 75]. Baseado em sua experiência como gerente do projeto OS/360 na IBM, Brooks discutiu, principalmente, as atividades de planejamento, dimensionamento de custos e alocação de pessoas a projetos de *software*. Brooks concluiu que o primeiro sistema produzido no contexto de um projeto é muito frágil e mal pode ser utilizado, assegurando que ele é sempre dispensado em favor da construção de outro sistema. Assim, Brooks sugeriu a adoção da abordagem de desenvolvimento através da construção de protótipos descartáveis na construção de *software*.

O trabalho em análise teve início no final da década de 1960 e início da década de 1970. Primeiramente surgiu vinculado a outro tópico importante: projeto estruturado [PRESSMAN 97]. Havia a necessidade de uma notação gráfica para representar dados e processos conduzidos para transformá-los. Esses processos seriam, no final do trabalho, mapeados ao projeto. Contudo, não foi antes de 1979 que DeMarco [DEMARCO 79] conseguiu popularizar o termo "análise estruturada", introduzido em 1977 por Ross [ROSS 77b]. A técnica consistia de um conjunto de símbolos gráficos que permitiam ao analista representar modelos de fluxo, um conjunto de heurísticas relacionadas com o uso desses símbolos, e sugeria que um *dicionário de dados* e *narrativas de processos* poderiam ser usadas como suplemento de informação aos modelos de fluxo.

Em meados da década de 1980, algumas das deficiências da análise estruturada ficaram evidentes com a tentativa de utilizá-la em aplicações destinadas a controle na área de engenharia. Em 1985, Ward e Mellor e, mais tarde – em 1987, Hatley e Pribhai introduziram extensões, transformando a técnica em um método mais robusto, que podia ser aplicado aos problemas reais de engenharia [PRESSMAN 97].

Sentindo-se desconfortável com a direção seguida pela pesquisa, novamente Brooks, em 1987, publicou uma das mais notáveis perspectivas sobre os problemas do desenvolvimento de *software* [BROOKS 87]. Brooks classificou os problemas em essenciais e acidentais. Os primeiros relacionados à natureza do *software* e os últimos relacionados com seu processo de construção. Analisando então, os últimos dez anos de pesquisa da Engenharia de Software, Brooks afirmou que todas as técnicas propostas até então visavam apenas os problemas que ele havia classificado como acidentais. Referindo-se ao que poderiam ser técnicas que auxiliariam a resolver os problemas essenciais, ele cita o refinamento dos requisitos do sistema e o uso de uma abordagem incremental e baseada em protótipo como duas questões fundamentais.

Em 1988, Barry Bohem propôs uma abordagem evolucionária e incremental ao processo de desenvolvimento de *software*, chamando-a de Modelo Espiral [BOEHM 88]. Esse novo modelo trouxe uma visão mais realista ao processo de desenvolvimento, adaptando-se a projetos grandes e pequenos e privilegiando a atividade de análise de riscos, que deveria ser conduzida com mais ou menos formalismo, independente do tipo de projeto.

Lamsweerde [LAMSWEERDE 00] aponta o trabalho de Ross [ROSS 77a], em 1977, como o embrião da área de Engenharia de Requisitos, considerando que em seu artigo, Ross explica de forma completa, não apenas o escopo da Engenharia de Requisitos, mas também introduz conceitos como objetivos, perspectivas, dados, operações, agentes e recursos como potenciais elementos fundamentais da Engenharia de Requisitos. Simultaneamente, Ross introduziu SADT [ROSS 77b], uma técnica de modelagem, pioneira na introdução de diversos conceitos que viriam a amadurecer mais tarde no trabalho de outros pesquisadores. Em particular, era uma técnica semiformal e gráfica, uma característica essencial para a comunicabilidade de modelos.

Pouco depois, em 1980, Bubenko [BUBENKO 80] introduziu uma técnica de modelagem baseada na captura de entidades e eventos. Assertivas formais podiam ser escritas para expressar os requisitos envolvendo tais entidades e eventos, em particular, restrições temporais. Segundo Lamsweerde [LAMSWEERDE 00], já se reconhecia que tais entidades e eventos seriam parte fundamental do contexto em torno do *software* a ser construído ([JACKSON 78]).

Ainda segundo Lamsweerde [LAMSWEERDE 00], outras técnicas semi-formais foram desenvolvidas no final da década de 1970, notadamente, diagramas de entidade-relacionamento para a modelagem de dados, introduzido por Chen em 1976 [CHEN 76], análise estruturada, introduzida por De Marco em 1979 [DEMARCO 79], e diagramas de mudança de estado para a modelagem da interação com o usuário, introduzido por Wasserman em 1979 [WASSERMAN 79].

Requirements Modeling Language, RML, avançou significativamente sobre os conceitos introduzidos por SADT, agregando novos e poderosos elementos de estruturação, tais como generalização, agregação e classificação [GREENSPAN 82]. Segundo Lamsweerde [LAMSWEERDE 00], nesse sentido pode-se considerá-la a precursora das técnicas de análise orientadas a objetos. Os mecanismos de estruturação introduzidos por RML aplicavam-se a três tipos de unidades conceituais: entidades, operações e restrições. Restrições eram especificadas através de linguagens de declarações formais. Nessa época foram registrados avanços nas áreas de modelagem de bancos de dados, representação do conhecimento e especificação formal baseada em estado [LAMSWEERDE 00].

Lamsweerde [LAMSWEERDE 00] prossegue afirmando que a pesquisa evoluiu com a introdução de agentes, em 1987 por Feather [FEATHER 87] e de objetivos, primeiramente caracterizados por Yue [YUE 87], em 1987. Yue argumentou que a inclusão de objetivos nos modelos de requisitos possibilitava o estabelecimento de um critério para a avaliação dos requisitos quanto a sua completitude: os requisitos estariam completos se eles fossem suficientes para estabelecer o objetivo que estavam refinando. A pesquisa na linha de objetivos avançou com os trabalhos de Mylopoulos, Chung e Nixon [MYLOPOULOS 92] em 1992, Dardene [DARDENE 93] e Nixon [NIXON 93], em 1993 e Yu [YU 97] em 1997.

A identificação e tratamento de conflitos evoluiu a partir de Robinson [ROBINSON 89], em 1989, com a utilização de objetivos para negociação de requisitos e prosseguiu com os trabalhos de Easterbrook [EASTERBROOK 94b] e Nusseibeh [NUSEIBEH 94] em 1994 com a introdução de perspectivas no tratamento de conflitos e Boehm [BOEHM 95] em 1995, com uma abordagem espiral para a negociação de conflitos através de objetivos.

Apesar da grande utilização de objetivos, identificá-los no mundo real e documentá-los pode não ser tarefa fácil. Nessa direção, evoluíram as técnicas baseadas em cenários, com os trabalhos de Benner [BENNER 93] em 1993, Potts [POTTS 94] em 1994 e Weidenhaupt [WEIDENHAUPT 98] e Jarke [JARKE 98b] em 1998, para citar alguns.

Paralelamente a todo esse trabalho, a natureza dos requisitos foi alvo de novos estudos que culminaram com a distinção entre *propriedades do domínio* e *requisitos* e entre *requisitos do sistema* e *especificação do software* [LAMSWEERDE 00]. Outras áreas às quais a pesquisa dedicou esforços foram na pesquisa de sistemas reativos, na reutilização de requisitos e na documentação de requisitos [LAMSWEERDE 00].

4. O ciclo de vida da Engenharia de Requisitos

Para melhor entender os caminhos da pesquisa na área de Engenharia de Requisitos, é preciso entender exatamente o escopo da disciplina e sua inserção no contexto da Engenharia de Software, estabelecendo o limite das atividades que devem ser desenvolvidas e o produto final a ser entregue para as próximas atividades do ciclo de vida do *software*. Assim, nesta seção, serão explicados cada um desses tópicos, visando estabelecer a terminologia que dará suporte à classificação apresentada a seguir.

4.1. *Requisitos*

Fundamental para entendermos a disciplina Engenharia de Requisitos é entendermos exatamente o que são requisitos de um sistema. Grosseiramente, podemos imaginar que requisitos são as funções que deverão ser incorporadas pelo *software*, quando inserido em seu contexto de funcionamento. Rapidamente, no entanto, essa definição inicial torna-se insuficiente para entendermos a correta dimensão do problema.

Necessidades como desempenho, integridade, disponibilidade e segurança seriam difíceis de acomodar nessa definição preliminar de requisitos. Apesar de tais qualidades transformarem-se, em algum nível de abstração, em funcionalidades do novo sistema, no nível de abstração no qual requisitos estão sendo tratados, tais qualidades não são funcionalidades do sistema em especificação.

James e Suzanne Robertson [ROBERTSON 99] definem requisitos como "*alguma coisa que o produto tenha que fazer ou uma qualidade que precise estar presente*".

Sob essa perspectiva, teríamos duas categorias de requisitos: aqueles responsáveis pela funcionalidade do sistema ("*... alguma coisa que o produto tenha que fazer...*") e aqueles responsáveis por qualidades que devam estar presentes, tais como desempenho, integridade, disponibilidade e segurança. Os primeiros são denominados requisitos funcionais e os últimos requisitos não funcionais.

Kotonya e Sommerville [KOTONYA 98] definem requisitos como a descrição de:

- uma facilidade no nível do usuário;
- uma propriedade geral do sistema;
- uma restrição específica sobre o sistema;
- a especificação de um algoritmo particular, que deva ser empregado em cálculos particulares;
- uma restrição aplicável ao processo de desenvolvimento do sistema.

A norma IEEE Std 1233 *IEEE Guide for Developing System Requirements Specifications* [IEEE 1233-1998], em sua seção de definições, em seu item 3.15, define requisitos como:

a. uma condição ou capacidade necessária para o usuário resolver um problema ou atingir um objetivo;

b. uma condição ou capacidade que precise ser atendida ou estar presente em um sistema ou componente, para satisfazer um contrato, uma norma, uma especificação ou outro documento imposto formalmente;

c. uma representação documentada de uma condição ou capacidade, tal como definidas em a ou b.

Apesar de abrangentes, tais definições não caracterizam de forma clara a origem dos requisitos. Sem caracterizar de forma clara a origem dos requisitos, sua identificação e documentação podem ficar comprometidas, gerando ambigüidades e imprecisões indesejáveis.

Michael Jackson [JACKSON 95] afirma que requisitos tratam de fenômenos do domínio de aplicação[1] e não da máquina. Máquina, em sua definição, caracteriza-se pelo conjunto formado por *hardware* e *software*, e cuja existência é fundamentada na solução de um problema real do mundo real. Descrever os requisitos de um sistema em particular, refere-se a descrever os fenômenos presentes no contexto do problema[2], bem como as relações entre eles. Não descrevemos nada sobre a máquina. A máquina, por sua vez, consegue satisfazer aos requisitos, pois compartilha alguns fenômenos com o domínio de aplicação: máquina e domínio de aplicação tem alguns eventos e/ou estados em comum.

4.2. *Engenharia de Requisitos*

Engenharia de Requisitos é um amplo campo de estudos, inserido no contexto da Engenharia de Software, e relacionado à identificação, validação e documentação das funções e restrições que precisam ser respeitadas por um *software* em sua construção e operação. Existem muitas definições na literatura.

Lamsweerde [LAMSWEERDE 00] define Engenharia de Requisitos como:

"*a identificação dos objetivos a serem atingidos pelo futuro sistema, a operacionalização*[3] *de tais objetivos em serviços e restrições, e a*

[1] Segundo Jackson [JACKSON 95], domínio de aplicação é a parte do mundo (real) na qual o cliente está interessado, com particular relevância para a solução do problema particular. Envolve qualquer coisa ou pessoa que irá interagir com a máquina ou inserir-se no contexto de seu processamento. Na seção 5.1 discutimos mais profundamente a questão do domínio de aplicação.

[2] Contexto do problema refere-se a parte do mundo (real) na qual a máquina será instalada, na qual os efeitos e benefícios de sua instalação serão percebidos e avaliados.

[3] Esta palavra não existe na língua portuguesa. Está sendo usada neste contexto para designar a transformação dos objetivos em serviços e restrições do sistema. Será utilizada no texto por ser comum no meio acadêmico.

atribuição de responsabilidades pelos requisitos resultantes a agentes humanos, dispositivos e software".

Pamela Zave [ZAVE 97a] define Engenharia de Requisitos como:

"*... o ramo da engenharia de software relacionado aos objetivos do mundo real estabelecidos para as funções e restrições aplicáveis a sistemas de software. Está também relacionada com a ligação entre esses fatores e a precisa especificação do comportamento do software e com sua evolução no tempo e através de família de produtos*".

Peter Kotonya e Ian Sommerville [KOTONYA 98] definem Engenharia de Requisitos como:

"*a forma como escolhemos denominar as atividades desenvolvidas, no contexto do ciclo de vida de software, relacionadas com a definição dos requisitos de um sistema*".

Alan Davis [DAVIS 93] define a fase de requisitos do *software* como:

"*as atividades relacionadas à produção de uma descrição completa do comportamento externo de um software a ser construído*".

É possível, através das definições apresentadas, verificarmos a existência de atividades ligadas à engenharia de requisitos que ocorrem no início do ciclo de vida do *software* ("*identificação de objetivos a serem atingidos pelo futuro sistema*"). Porém, é possível também identificar a existência de atividades que acompanham o *software* através de toda a sua vida útil ("*... com sua evolução no tempo e através de família de produtos...*"). Veremos mais adiante como são essas atividades.

4.3. *Engenharia de Requisitos no contexto do ciclo de vida de Software*

Engenharia de Software refere-se à aplicação de princípios científicos para [DAVIS 93]:

"*(1) transformar, de maneira ordenada, um problema em uma solução de software;*

(2) promover sua subseqüente manutenção, até o final de sua vida útil".

Para melhor entender as atividades conduzidas nesse contexto, é preciso estudar modelos para o ciclo de vida de *software*. Modelos de ciclo de vida de *software* são representações produzidas para auxiliar no entendimento e identificação das atividades conduzidas para a produção de *software*. O primeiro modelo de ciclo de vida proposto foi o Modelo Cascata, introduzido por Royce em 1970. Pressman [PRESSMAN 97] descreve este modelo como uma seqüência de atividades consecutivas cujo produto final é o *software* desenvolvido, tal como mostra a Figura 1.

O ciclo é composto pelas atividades de Análise, Projeto, Codificação e Teste. Como, de maneira geral, *software* é parte de um sistema maior (ou negócio), o trabalho inicia-se com a determinação dos requisitos de todos os elementos do sistema, passando depois pela alocação de um subconjunto deles ao *software*. Assim, Engenharia de Sistemas refere-se à obtenção dos requisitos e uma dose de análise e projeto em um nível mais alto de abstração, para esses macro-elementos do sistema. Engenharia da informação envolve a obtenção dos requisitos no nível do negócio.

As atividades do processo são então:

- **Análise** → relacionada à intensificação do processo de obtenção dos requisitos, focalizando-o, especificamente, no *software*. Nesta atividade são identificadas e documentadas as funções do *software*, seu comportamento e suas interfaces. Todos esses itens são, posteriormente, validados junto ao cliente.
- **Projeto** → concentra-se na transformação dos requisitos do sistema a uma representação que possa ser avaliada quanto à qualidade, antes do início da codificação.
- **Codificação** → relaciona-se à tradução do projeto a uma linguagem de programação, escolhida de acordo com outras variáveis relacionadas à condução do projeto.
- **Teste** → destina-se a verificar que o programa construído esteja de acordo com o projeto e com os requisitos levantados e documentados.

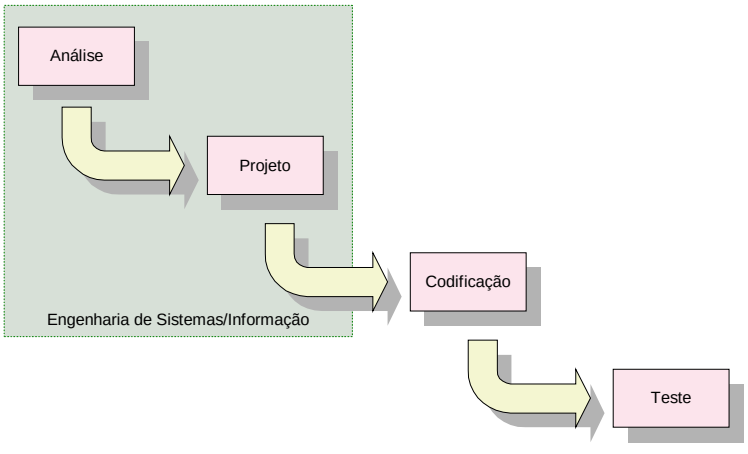

Figura 1 - O modelo em cascata para o ciclo de vida do *software*[4]

[4] Obtida de [PRESSMAN 97], página 31.

Davis [DAVIS 93] prefere uma representação diferente, que destaque a simetria existente entre as fases iniciais e finais do ciclo de vida, conforme mostra a Figura 2.

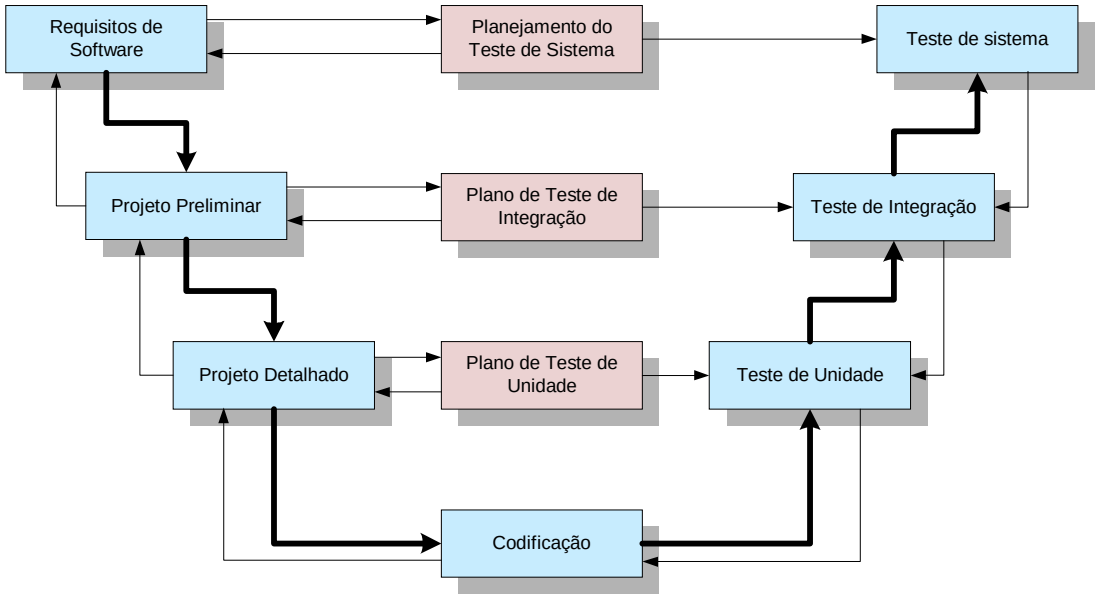

Figura 2 - Outra versão do modelo em cascata[5]

As atividades desenvolvidas são conforme abaixo:

- **Requisitos de *Software*** - inicia-se com a análise do problema de *software* que se apresenta, finalizando com a completa especificação do comportamento externo desejado do sistema (ou ainda descrição funcional, requisitos funcionais ou especificações);
- **Projeto Preliminar** - decompõe iterativamente o *software* em componentes arquiteturais até que cada subcomponente seja suficientemente pequeno para ser desenvolvido. Denomina-se cada um desses subcomponentes módulo;
- **Projeto Detalhado** - define e documenta os algoritmos de cada módulo que será convertido em código;
- **Codificação** - transforma os algoritmos definidos no projeto detalhado em uma linguagem de programação;
- **Teste de Unidade** - verifica cada um dos módulos codificados quanto à presença de defeitos[6];

[5] Extraída de [DAVIS 93], página 10.

[6] Defeito é um erro humano cometido durante a construção do software (em qualquer fase do ciclo de vida), presente em alguma de suas representações (em qualquer de seus níveis de abstração) e cuja identificação é a finalidade da atividade de teste [PFLEEGER 98].

- **Teste de Integração**- verifica o funcionamento de um conjunto de módulos interconectados. Idealmente o conjunto de módulos integrados deveria formar um componente identificado no projeto preliminar. Assim, o objetivo do teste de integração é garantir que cada componente funcione de acordo com a especificação;

- **Teste de Sistema** - verifica que o sistema, como um todo, funcione de acordo com a especificação e no *hardware* para o qual foi projetado.

Comparando-se as definições dos dois ciclos de vida vemos que Pressman chama de análise o que Davis chama de Requisitos de *Software*. O que Pressman chama de Projeto, Davis divide em duas atividades Projeto Preliminar e Projeto Detalhado. A fase de codificação é equivalente e a atividade de Teste está desmembrada na representação de Davis.

Assim, para a discussão dos tópicos a seguir, adotaremos a proposta de Pressman, definindo cada uma das atividades como a seguir (Figura 3):

- **Análise** - visa especificar o problema a ser resolvido, declarando-o e entendendo-o com o auxílio de técnicas desenvolvidas no contexto da Engenharia de Requisitos;

- **Projeto** - visa conceber e descrever uma solução para o problema;

- **Codificação** - visa construir a solução descrita;

- **Teste** - visa verificar a solução construída.

A abordagem adotada pelo modelo em cascata é bastante simplista e, de forma geral, não pode ser utilizada na construção de sistemas moderadamente complexos. Na prática, ao lidar com sistemas complexos, não podemos dar-nos ao luxo de esperar até que tenhamos todos os requisitos do sistema identificados para podermos prosseguir com sua construção. Tampouco os requisitos são estáveis a ponto de podermos supor que eles permaneçam estáticos até o final da construção do *software*.

Assim, outras abordagens para o ciclo de vida do *software* foram propostas, como o Modelo Espiral [BOEHM 88] e o *Unified Process* [KRUCHTEN 98]. A primeira prevê um modelo iterativo e incremental segundo o qual *software* é produzido por partes e através de atividades que se repetem para cada incremento a ser entregue. A segunda abordagem registra a percepção da simultaneidade das atividades ao longo do processo de desenvolvimento, caracterizando o fato de que diversas partes da equipe de desenvolvimento trabalham em uma representação diferente do *software* em cada momento, com maior ou menor intensidade.

4.4. *A Especificação de Requisitos do Software*

A cada fase do ciclo de vida do *software* produzimos um documento contendo uma representação distinta do *software* a ser construído. Cada um desses documentos representa o *software* em um nível mais baixo de abstração, isto é, incluindo mais e mais detalhes, até que, sua última representação seja o código fonte na linguagem escolhida (Figura 3).

A Engenharia de Software destaca a importância do controle da condução do processo de construção do *software*, como forma de garantir-lhe a qualidade [PRESSMAN 97]. Assim, é importante entendermos como se dá a transição entre uma atividade e outra. A definição de um documento a ser produzido ao final de cada uma das atividades auxilia, tanto na determinação das tarefas para a condução da atividade, quanto no acompanhamento do processo em si, caracterizando o final de uma fase e o início da seguinte.

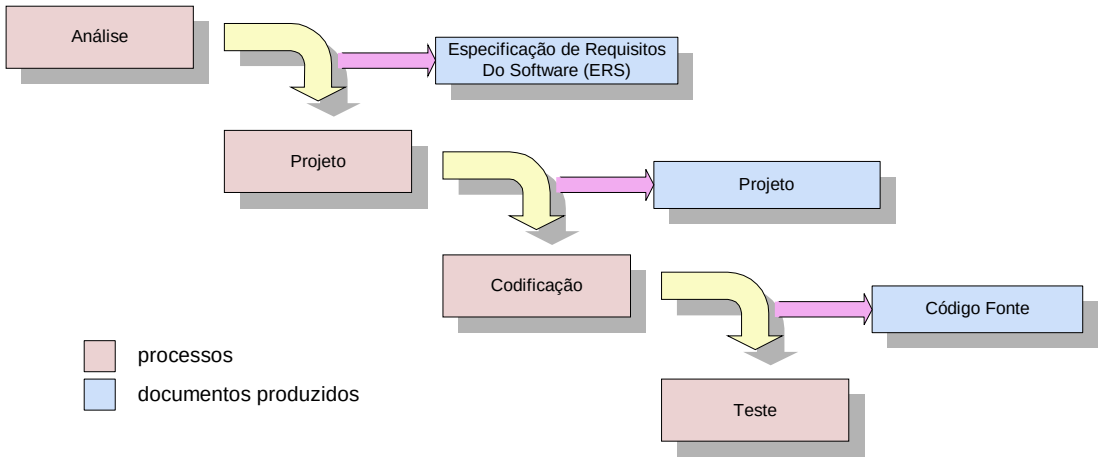

Figura 3 - O modelo em cascata simplificado[7]

Sob a perspectiva da Engenharia de Requisitos, é importante entendermos o documento fornecido pela atividade de Análise à atividade de Projeto. Diversos são os nomes utilizados para identificar esse documento. Kotnonya & Sommerville [KOTONYA 98] utilizam *Documento de Requisitos*, Davis [DAVIS 93] utiliza *Especificação de Requisitos de Software*, Robertson [ROBERTSON 99] também utiliza *Documento de Requisitos*. A norma IEEE Std-830 *IEEE Recommended Practice for Software Requirements Specifications* [IEEE 830-1998] denomina este documento Especificação de Requisitos do Software (ERS) e define um conjunto de diretrizes a serem seguidas em sua confecção.

[7] Adaptada de [PRESSMAN 97], página 31.

A princípio, a ERS é uma descrição *do quê* o *software* deve fazer, sem descrever *como* irá fazê-lo. Essa abordagem é notadamente simplista e sua aplicação depende da especificação do nível de abstração no qual o sistema está sendo tratado. *O quê*, definido para um nível, representa *como* para o nível seguinte e assim, sucessivamente, como sustentado por Davis [DAVIS 93]. De acordo com Kotonya e Sommerville [KOTONYA 98] a ERS destina-se a comunicar os requisitos a clientes, engenheiros de *software* e de sistemas e gerentes do processo de engenharia de *software*. Na verdade, essa audiência é muito maior, pois outros *stakeholders*[8] deveriam ser considerados como, por exemplo, órgãos reguladores ou núcleos organizacionais não diretamente ligados ao uso do sistema, mas com algum interesse em sua construção e/ou operação.

A ERS deve descrever:

- Serviços e funções providas pelo sistema;
- Restrições sob as quais o sistema deve operar;
- Propriedades gerais do sistema, isto é, restrições aplicáveis à suas propriedades emergentes;
- Definições de outros sistemas com os quais o sistema em desenvolvimento tenha que interagir;
- Informações sobre o domínio de aplicação do sistema e como conduzir tipos específicos de processamento (dependentes do domínio de aplicação);
- Restrições aplicáveis ao processo de desenvolvimento.

A norma IEEE Std 830 [IEEE 830-1998] define a estrutura básica do documento, especificando como deve ser organizado e fornece um modelo para sua confecção. Contudo, o conteúdo do documento depende de diversos fatores como, por exemplo, a natureza do sistema que está sendo especificado, o nível de detalhamento dos requisitos, práticas organizacionais, além de restrições orçamentárias e de tempo aplicáveis ao processo de Engenharia de Software.

Para acomodar essa natureza diversa da ERS, seu modelo padronizado tem que ser flexível, estruturando-se em torno de seções estáveis, isto é, que devem existir independente dos fatores do caso concreto e outras seções variáveis, cuja presença dependerá da natureza do caso concreto que estiver sendo tratado.

Kotonya e Sommerville [KOTONYA 98] destacam que entre as seções estáveis de um modelo padronizado de requisitos deve estar o glossário. O glossário define os termos constantes do documento de forma a normalizar a terminologia utilizada na ERS. Devido ao diferente passado intelectual dos *stakeholders*, não produzir um glossário pode causar enormes diferenças de entendimento entre eles, gerando inconsistências e imprecisões que se deseja evitar.

---

[8] Sommerville [KOTONYA 98] define *stakeholder* como pessoas ou organizações que serão afetadas pelo sistema e que têm influência, direta ou indireta em seus requisitos.

4.4.1. Qualidades de uma (ERS)

Entre as qualidades que devem estar presentes em uma ERS estão, conforme especificado pela IEEE 830 [IEEE 830-1998]: correção, não ambigüidade, completitude, consistência, classificação quanto à importância e/ou estabilidade, além de ser verificável, modificável e rastreável[9].

Abaixo são descritas, brevemente, cada uma dessas características:

- **correção** - se todo o requisito no documento é um requisito que tenha que ser implementado pelo *software*;
- **não ambigüidade** - se todo o requisito no documento tem apenas uma interpretação;
- **completitude** - se inclui todos os requisitos significantes, a definição das respostas do *software* a todos os tipos de dados de entrada em todas as situações possíveis, válidas ou inválidas, e referencia todos os termos, figuras e/ou diagramas, além de unidades de medida;
- **consistência** - se nenhum dos requisitos do documento, tomado individualmente, está em conflito com qualquer outro requisito do mesmo documento;
- **classificação** - se existirem indicações no documento quanto à importância ou estabilidade do requisito;
- **ser verificável** - se para cada um dos requisitos contidos no documento, existir um processo finito e economicamente viável através do qual uma pessoa ou máquina possa assegurar que o produto de *software* atenda ao requisito;
- **ser modificável** - se modificações puderem ser agregadas ao documento de forma fácil, completa e consistente, com relação a sua estrutura e estilo;
- **ser rastreável** - se a origem de cada um de seus requisitos é clara e a referência a cada um dos requisitos é facilitada nos documentos subseqüentes do processo ou em uma melhoria da documentação do sistema (através da numeração dos requisitos, por exemplo).

A representação dos requisitos neste estágio inicial do desenvolvimento de *software* é de grande importância, dada sua audiência bastante heterogênea.

---

[9] Novamente a palavra rastreável não existe na língua portuguesa. Está sendo utilizada aqui no sentido de que o documento permita que seus requisitos sejam passíveis de serem rastreados. Será utilizada no decorrer do texto com esse sentido.

Kotonya & Sommerville [KOTONYA 98] destacam que a linguagem natural é a única notação existente passível de interpretação por todos os potenciais leitores do documento de requisitos. Reconhecem, contudo, que requisitos expressos dessa forma são potencialmente difíceis de entender, podendo ser ambíguos, surpreendentemente obscuros e mal interpretados. Davis [DAVIS 93] destaca que uma das razões para a alta ambigüidade contida em especificações expressas em linguagem natural é o fato de que, ao nos expressarmos verbalmente, a entonação, os movimentos das mãos e a linguagem corporal ajudam a clarear a idéia que se pretende transmitir. Na linguagem escrita, contudo, essa parte da expressão verbal não existe, contribuindo para sua ambigüidade.

Ainda segundo Davis, uma solução para esse problema é visualizar um invólucro entorno da linguagem natural, com semântica bem definida, através do qual o significado dado às palavras é restrito a um significado específico mantido ao longo de todo o texto. Nessa direção, Leite [LEITE 93] desenvolveu o Léxico Estendido do Domínio, uma forma de estruturar e entender a terminologia utilizada no Universo de Discurso[10], definindo noções (denotação) e respostas comportamentais (conotação).

A produção da ERS, de acordo com as qualidades desejadas em linguagem natural, apresenta um problema importante: o grande volume que atingem os documentos produzidos em linguagem natural. Quanto maior a ERS, maiores são as dificuldades com sua consistência e manutenção. Neste caso, sua especificação somente através da linguagem natural é um problema ainda mais crítico. Davis [DAVIS 93] recomenda que sejam usadas técnicas formais[11] sempre que não se possa correr o risco de ter o requisito mal interpretado.

4.4.2. A representação dos requisitos

A representação dos requisitos é tema permanente de discussão e responsável por grande parte da ambigüidade da terminologia na área. Davis [DAVIS 93] afirma que um requisito, qualquer que seja a linguagem de sua especificação:

- define um objeto, uma função ou um estado;
- limita ou controla as ações associadas a um objeto, uma função ou um estado;
- define relacionamentos entre objetos, funções ou estados.

---

[10] Universo de Discurso é a terminologia utilizada pelo autor para designar o contexto definido pela engenharia de sistemas (o domínio de aplicação).

[11] Notações formais são aquelas baseadas em sintaxe e semântica de matemática formal.

Um objeto é qualquer entidade do mundo real, com interface bem definida e com relevância para a solução do problema.

Uma função, sob a perspectiva de requisitos, é uma tarefa, serviço, processo, função matemática ou atividade que é:

- executada no mundo real;
- executada pelo sistema que está sendo especificado para resolver um problema no mundo real.

Um estado, do ponto de vista de requisitos, é uma condição de algo que captura parte de sua história e é usada para determinar como ele deve se comportar em circunstâncias específicas. Este algo pode ser o sistema, um objeto ou uma função.

Segundo Zave e Jackson [ZAVE 97b] requisitos devem descrever aquilo que pode ser observado na interface entre o ambiente e o sistema, e nada sobre o sistema.

Prosseguem afirmando que todas as declarações feitas no contexto da Engenharia de Requisitos são declarações sobre o ambiente, podendo ser de dois tipos:

- ***indicativas*** - descrevendo o ambiente como ele é, na ausência do sistema ou sem considerar suas ações;
- ***optativas*** - descrevendo o ambiente tal como gostaríamos que ele fosse quando o sistema estiver conectado a ele.

Segundo essa classificação, Zave e Jackson afirmam que requisitos são declarações optativas, definindo especificações como declarações optativas implementáveis, denominando o processo que leva da declaração de requisitos à especificação como refinamento de requisitos.

Além disso, Zave e Jackson alertam para a tendência à implementação a que podem estar sujeitas especificações baseadas em estados. Ao especificarmos estados internos do sistema estamos descrevendo algo que não é observável na interface entre ele e o ambiente. Diversas abordagens utilizam técnicas baseadas em modelos, representando o estado do sistema através de estruturas matemáticas como conjuntos, relações e tuplas. Tais técnicas correm maior risco de polarizar a especificação em favor da implementação.

A polarização da especificação em favor da implementação é indesejável, pois pode restringir inadequadamente a solução, fazendo com que o modelo utilizado na especificação seja adotado no projeto, sem que outras alternativas sejam analisadas criteriosamente e que, talvez, pudessem visar de

forma mais adequada as características desejadas de um bom projeto[12].

4.5. *Um processo para a condução da Engenharia de Requisitos*

Da mesma forma que para entender as atividades contidas no contexto da Engenharia de Software produziram-se modelos de ciclo de vida, para entendermos as atividades que devem ser conduzidas para a produção da ERS, devemos delinear um processo para a condução da Engenharia de Requisitos.

Nuseibeh e Easterbrook [NUSEIBEH 00] apontam a definição de Engenharia de Requisitos fornecida por Zave como uma das mais claras. Destacam, entre os pontos que a fazem eficiente, a ênfase nos "*objetivos do mundo real*", representando tanto a importância dos *porquês* quanto dos *comos* de um sistema. Além disso, refere-se à "*especificações precisas*" oferecendo a base para as atividades de *análise*, *validação*, *definição* e *verificação*, que estarão presentes em um processo para a atividade de requisitos. Continuam, apontando a "*evolução através do tempo e através de famílias de produtos*" como a identificação da realidade de um mundo em constante mudança, no qual o reuso[13] de especificações parciais é muito importante, tal como ocorre em outros ramos da engenharia.

Essas observações sugerem, como as próprias definições, a existência de atividades no contexto da Engenharia de Requisitos desenvolvidas no início do ciclo de vida do *software*, bem como atividades desenvolvidas ao longo do ciclo de vida, acompanhando as atividades de análise, projeto, codificação, teste e, posteriormente, manutenção[14].

Macaulay [MACAULAY 96] destaca que Engenharia de Requisitos refere-se à análise de um problema utilizando uma perspectiva futura, é uma atividade iterativa, uma vez que informações que afetam os requisitos podem ser identificadas em fases avançadas do processo de desenvolvimento e portanto, caracteriza-se por sua natureza dinâmica. Macaulay prossegue destacando que um processo para a criação da ERS deve ser sistemático, interativo, repetível e controlável. Sua definição envolve:

- a identificação de etapas a serem desenvolvidas para a elaboração de um processo sistemático de forma que possa ser repetido;

---

[12] A representação dos requisitos tem papel fundamental no processo de requisitos. Não há consenso entre os pesquisadores a respeito do tema, e essa falta de convergência afeta a solução dos problemas encontrados no contexto do processo de requisitos. A modelagem de requisitos é ainda um problema em aberto na pesquisa, como veremos mais adiante.

[13] Palavra utilizada tecnicamente para denotar reutilização.

[14] Nome dado às atividades de atualização do *software* (manutenção evolutiva) e de correção de erros identificados (manutenção corretiva) após sua entrega.

- a identificação dos participantes do processo de forma a estabelecer medidas de conhecimento do problema para a determinação do fim do processo;

- a definição de uma forma adequada de representação para os documentos produzidos nas fases intermediárias e final do processo;

- o estabelecimento de medidas para a exatidão das informações obtidas através das atividades desenvolvidas no processo.

Ainda segundo Macaulay, uma proposta genérica de processo divide as atividades em duas grandes etapas: levantamento de informações e produção da ERS. A primeira fase caracteriza-se pela incerteza e pela expansão do conhecimento sobre o domínio do problema. A segunda pela organização do conhecimento, sua análise a partir de diferentes perspectivas e a eliminação de inconsistências. Estabelecidas essas características, Macaulay [MACAULAY 96] estabelece um processo básico para a atividade de Engenharia de Requisitos, contendo as seguintes atividades: conceituação, análise do problema, escolha de alternativas e viabilidade, análise, modelagem e produção do documento de requisitos. Cada uma dessas atividades é descrita, brevemente, abaixo:

- **Conceituação** - destina-se à identificação de uma necessidade de automação;
- **Análise do problema** - desenvolve o entendimento da natureza do problema;
- **Escolha de alternativas e viabilidade** - desenvolve a análise de custo-benefício das soluções alternativas propostas;
- **Análise e modelagem** - aprofunda o estudo da alternativa escolhida através da modelagem do espaço da solução;
- **Produção do documento de requisitos** - relaciona-se com as ações de produzir a ERS na forma prevista para o processo.

Esse modelo básico, contudo, não caracteriza a natureza dinâmica dos requisitos, que implica um processo iterativo e incremental, tal como a abordagem do modelo espiral para o processo de *software*.

Kotonya e Sommerville [KOTONYA 98] propõem um modelo de processo semelhante que, inclui a realimentação necessária à caracterização da natureza dinâmica dos requisitos, como mostra a Figura 4.

Nesse modelo, cada uma das atividades desenvolvidas é como se segue:

- **Elicitação** - consiste na identificação dos requisitos a partir de consulta aos *stakeholders*, da análise de

documentos, da análise de informações do domínio e/ou de estudos de mercado;

- **Análise e Negociação** - consiste na análise detalhada dos requisitos com a negociação entre os diferentes *stakeholders* visando decidir quais os requisitos que serão aceitos;

- **Documentação** - os requisitos aprovados na atividade de análise e negociação devem ser registrados em um documento apropriado, em nível de detalhamento adequado;

- **Validação** - depois de documentados, os requisitos devem sofrer cuidadosa verificação de consistência e completitude.

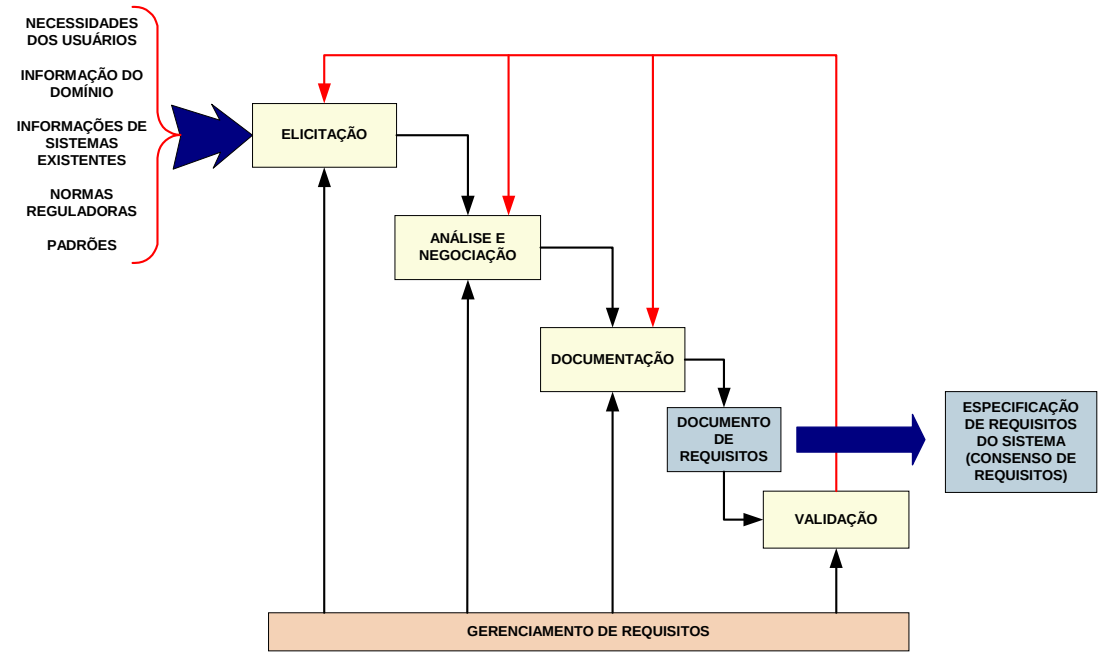

Figura 4 - Um modelo para o processo de Engenharia de Requisitos[15]

O procedimento de validação pode encontrar problemas de várias naturezas no documento de requisitos. Cada um desses problemas deve ser remetido de volta à atividade responsável por sua execução. Assim, problemas como falta de informações devem ser remetidos de volta à atividade de elicitação. Ambigüidades ou alternativas mal resolvidas devem ser remetidas de volta à análise e negociação, assim como problemas com a documentação (redação, diagramas, referências, etc.) devem ser remetidos à atividade de documentação. Esse retorno à atividades anteriores implementa um ciclo de realimentação que garante a natureza iterativa do processo de requisitos.

[15] Adaptada de [KOTONYA 98], página 32.

Por outro lado, a natureza dinâmica dos requisitos requer que, na presença de alguma mudança em relação à Especificação de Requisitos do Software aprovada inicialmente (o consenso dos requisitos), seja conduzida uma ampla análise das alterações propostas, visando identificar e documentar adequadamente quais são essas alterações.

No modelo de processo proposto por Kotonya e Sommerville, essa atividade é o Gerenciamento de Requisitos, que tem início no contexto da Engenharia de Requisitos e se propaga por todo o processo de *software*, analisando o impacto das alterações nos requisitos em todas as demais representações do *software*, chegando, finalmente, ao programa propriamente dito com a implementação das alterações.[16]

O estudo isolado das atividades da engenharia de requisitos pode nos levar a perder algumas características importantes de todo o processo de *software*. Swartout e Balzer [SWARTOUT 82] caracterizaram a natureza intimamente ligada da implementação e da especificação[17]. Assim, é inevitável que ao longo da implementação sejam necessários ajustes no Projeto (documento produzido pela atividade de Projeto e base para o desenvolvimento da atividade de codificação no processo de *software*) e na Especificação de Requisitos do Software (base para o desenvolvimento da atividade de Projeto). É através do Gerenciamento de Requisitos que se dá essa realimentação do processo de requisitos com informações das atividades subseqüentes do processo de *software*.

Adotaremos esse modelo de ciclo de vida para o processo de requisitos em na argumentação das seções que se seguem. Para isso, vamos entender um pouco melhor o trabalho desenvolvido em cada uma das atividades do modelo.

4.5.1. Os participantes do processo

Parte do entendimento das atividades desenvolvidas no processo de Engenharia de Requisitos envolve a identificação de seus participantes, caracterizando seus papéis e suas responsabilidades no contexto do processo.

Uma das características do processo de Engenharia de Requisitos é a diversidade de interesse das pessoas envolvidas. Existem pessoas interessadas no sistema como uma solução para um

---

[16] A figura representativa do processo de requisitos caracteriza um modelo de processo orientado a atividades. Na Engenharia de Software, tanto quanto em outras disciplinas de engenharia que tratam de processos produtivos, os modelos que mais se aproximam da realidade são aqueles orientados a processos, que presumem a simultaneidade das atividades ao logo de todo o ciclo de vida, como o UP, na Engenharia de Software [KRUCHTEN 98]. Tais modelos de processo são, no entanto, mais complexos e refletem um melhor entendimento da disciplina. Engenharia de Requisitos está ainda em sua infância. Assim, devemos entender as atividades do modelo de processo apresentado mais como um agrupamento das tarefas a serem desenvolvidas, do que como atividades de um macro processo seqüencial. Dessa forma facilitamos o entendimento do processo como um todo.

[17] Segundo os autores, especificação refere-se à descrição detalhada de um sistema em uma linguagem formal, em um alto nível de abstração e de maneira independente de implementação. Implementação refere-se à tradução dessa especificação a uma linguagem de programação, incluindo todas as decisões necessárias quanto às estruturas que irão compor o futuro programa.

problema ou apoio na condução de uma atividade do mundo real, pessoas interessadas na solução de problemas relacionados ao desenvolvimento de tais sistemas, pessoas ligadas a órgãos reguladores interessadas em como o uso desses sistemas afeta o ambiente no qual serão instalados, estabelecendo regras e/ou restrições que deverão ser obedecidas pelos sistemas desenvolvidos naquele domínio específico de regulamentação. Essas pessoas são denominadas, genericamente, *stakeholders* [KOTONYA 98].

Kotonya e Sommerville [KOTONYA 98] classificam em cinco os principais papéis existentes no processo de requisitos, conforme Tabela 1. Apesar de os papéis descritos estarem definidos, a princípio, para um projeto no qual esteja sendo construído um protótipo, sua estrutura básica serviria, de forma genérica, para qualquer projeto.

Dada a diversidade de interesses sobre o sistema, seria útil se conseguíssemos melhor classificar a natureza do interesse de cada *stakeholder* sobre os requisitos, de forma a poder melhor entender a natureza da influência que irão exercer sobre os requisitos. Nessa direção, Kotonya e Sommerville [KOTONYA 96], em seu método VORD - *Viewpoint Oriented Requirements Definition*, afirmam que o processo de entendimento do sistema em análise, seu ambiente, seus requisitos e restrições baseiam-se fortemente nas "autoridades do sistema". "Autoridades do sistema" são pessoas ou documentos com interesse ou conhecimento de especialista no domínio de aplicação. Incluem, entre outros, usuários finais, engenheiros de requisitos, engenheiros de sistemas e documentação de sistemas existentes. A partir dessa percepção, definem uma técnica para orientar na identificação das fontes de requisitos, classificando-as em pontos de vista diretos e pontos de vista indiretos. Pontos de vista diretos representam os *stakeholders* que interagem diretamente com o sistema. Pontos de vista indiretos representam os *stakeholders* que não interagem diretamente com o sistema, mas influenciam seus requisitos.

Tais definições fundamentam a criação da infra-estrutura, baseada nos conceitos de generalização/especialização, a partir da qual cada um dos pontos de vista passa a ser representado como classe ou subclasse de classes abstratas definidas na infra-estrutura prescrita. Essa infra-estrutura base pode ser então particularizada para o caso específico em análise, permitindo que sejam identificados todos os *stakeholders* que exercem algum tipo de influência sobre o sistema em estudo.

4.5.2. Elicitação de Requisitos

Designada por esse nome por conta do termo em inglês *elicit*, significando obter informações ou reações, esta fase do processo

relaciona-se à obtenção dos requisitos do sistema[18]. Assim, analistas e engenheiros de *software* trabalham com clientes e usuários finais para descobrir o problema a ser resolvido, os serviços do sistema, o desempenho necessário, restrições de *hardware* e outras informações [KOTONYA 98].

Papel	Descrição
Especialista do domínio	Responsável por prover informações sobre o domínio de aplicação e do problema específico a ser resolvido naquele domínio
Usuário final	Responsável pelo uso do sistema após a entrega
Engenheiro de requisitos	Responsável por identificar e especificar os requisitos do sistema
Engenheiro de *software*	Responsável por desenvolver o protótipo do sistema[19]
Gerente de Projeto	Responsável pelo planejamento do projeto

Tabela 1 - Os papéis no processo de Engenharia de Requisitos

Kotonya e Sommerville [KOTONYA 98] argumentam a existência de quatro dimensões na atividade de elicitação de requisitos:

- **Entendimento do domínio da aplicação** - na qual se dá o entendimento geral da área na qual o sistema será aplicado;
- **Entendimento do problema** - na qual se dá o entendimento dos detalhes do problema específico a ser resolvido com o auxílio do sistema a ser desenvolvido;
- **Entendimento do negócio** - na qual se dá o entendimento da contribuição do sistema para que sejam atingidos os objetivos gerais da organização;
- **Entendimento das necessidades e das restrições dos *stakeholders*** - na qual dá-se o entendimento detalhado:
    - das necessidades de apoio a serem providas pelo sistema à realização do trabalho e aos interesses de cada um dos *stakeholders*;
    - dos processos de trabalho a serem apoiados pelo sistema e;

[18] A palavra elicitar não existe na língua portuguesa. Contudo, devido à ampla utilização do termo na comunidade especializada, estarei utilizando-o neste texto.

[19] Este papel refere-se a todos os participantes do processo de software que efetivamente trabalham na construção do sistema, uma vez que nem sempre o projeto adotará uma abordagem por construção de protótipo. Neste contexto, protótipo refere-se à primeira versão do sistema.

- do papel de eventuais sistemas existentes na execução e condução dos processos de trabalho.

Davis [DAVIS 93] denomina Análise do Problema a atividade relacionada ao aprendizado sobre o problema a ser resolvido, ao entendimento das necessidades dos usuários, à correta identificação dos usuários e ao entendimento de todas as restrições aplicáveis à solução. Análise do Problema pode ser entendida como a determinação do espaço do produto, isto é, o conjunto de todas as possíveis soluções de *software*.

Nenhuma dessas definições fornece, contudo, a real dimensão da dificuldade na condução da atividade. Tal dificuldade surge da natureza menos técnica e mais social da atividade de engenharia de requisitos, como bem destacam Goguen e Linde [GOGUEN 93]. Segundo sua pesquisa, poucos esforços foram conduzidos no sentido de utilizar mais profundamente os conhecimentos desenvolvidos na área das ciências sociais no contexto da Engenharia de Requisitos. Menos ainda são os esforços baseados nas técnicas consideradas mais promissoras, particularmente etnometodologia e sócio-linguística.

Etnometodologia, ao contrário das técnicas convencionais que impõem a ordem do analista sobre o mundo social, considera que a ordem social é obtida pelos membros da sociedade em suas atividades do dia a dia. Dessa forma, a ordem social é construída pela ação dos participantes ao invés de ser uma categoria preexistente através da qual as ações dos participantes são definidas. Sócio-linguística deu origem a abordagens lingüísticas ao problema de requisitos concentrando-se, principalmente, na estrutura interna de certas formas de discurso.

A forte influência das questões sociais acaba por introduzir problemas nos requisitos levantados. Tais problemas precisam ser identificados para que possam ser tratados. Entre os problemas comuns enfrentados na atividade de elicitação Kotonya e Sommerville [KOTONYA 98] citam:

- A forma dispersa como são encontrados os requisitos (em livros, manuais, conhecimento de pessoas específicas, etc.);

- A terminologia específica do domínio da aplicação que precisa ser entendida para garantir o entendimento do problema no contexto do domínio da aplicação;

- A tarefa de auxiliar no levantamento de requisitos é, via de regra, secundária no contexto de trabalho dos *stakeholders* constituindo uma barreira à execução do trabalho de requisitos, culminando com o não

envolvimento dos *stakeholders* no processo de requisitos[20];

- Questões organizacionais e fatores políticos que exercem grande influência sobre os requisitos. Tais fatores nem sempre são identificados pelos usuários finais e podem passar desapercebidos pelo engenheiro de requisitos.

Além desses problemas, a possibilidade de automação altera a perspectiva dos *stakeholders* sobre o próprio trabalho, fazendo com que não tenham uma correta percepção sobre os requisitos do sistema [KRUCHTEN 98].

4.5.3. Análise e Negociação de Requisitos

As atividades conduzidas no contexto de Análise e Negociação de requisitos destinam-se a resolver problemas relacionados aos requisitos que envolvam diferentes percepções do mesmo problema ou necessidade por mais de um *stakeholder*, bem como os problemas gerados pela representação do conhecimento obtido de mais de um *stakeholder*.

Conflitos

Easterbrook [EASTERBROOK 93a] explica que devido a diferentes fontes e tipos de conhecimento a serem registrados na especificação, sempre haverá diferenças de opinião. A Engenharia de Requisitos pode ser vista como o processo de integração da expressão de diversos objetivos e restrições das pessoas envolvidas em uma especificação única e consistente. Assim, todo o processo de requisitos pode ser tratado como uma forma de negociação entre os *stakeholders* e os analistas (engenheiros de requisitos), no qual todos compartilham seus conhecimentos através da apresentação de sugestões e críticas às sugestões apresentadas. Nesse processo, é inevitável o surgimento de conflitos.

Easterbrook et al [EASTERBROOK 93b] definem conflito como a interferência das atividades, necessidades ou objetivos de um *stakeholder* com as de outro. Robbins [ROBBINS 89] explica que conflitos, tal como definidos acima, são comuns em interações de grupo.

Diversas podem ser as fontes de conflitos: as diferentes percepções de cada participante sobre o domínio da aplicação (conflitos entre as restrições declaradas, conflitos entre as necessidades identificadas, conflitos no uso de recursos, discrepâncias na avaliação de prioridades), os diferentes

[20] É claro que tais condições dependem do tipo de sistema que está sendo desenvolvido, o que se tornará uma das dimensões da taxonomia. Essas condições são particularmente importantes no caso de Sistemas de Informação, como veremos mais adiante.

objetivos de um projeto (conflitos entre componentes de uma solução proposta), entre outras.

Podemos enumerar três áreas que concentram as principais causas de conflitos:

- a dispersão do conhecimento sobre o domínio;
- instabilidade e inconsistência dos requisitos[21];
- deficiências na comunicação e na coordenação entre os participantes [EASTERBROOK 93a].

Inconsistências

O conhecimento elicitado dos *stakeholders* precisa ser representado. A complexidade intrínseca do *software* faz com que seja necessário representar tal conhecimento através de diversas notações distintas [BROOKS 87], mesmo nessa fase inicial do processo (poderíamos estar utilizando linguagem natural e termos que especificar um algoritmo particular e utilizar para isso diagrama de fluxos). Tais representações têm que ser mantidas consistentes entre si. Quando isso não acontece, dizemos que há uma inconsistência na especificação.

Nuseibeh e Easterbrook [NUSEIBEH 99] definem inconsistência como "*qualquer situação na qual duas descrições não obedecem a alguma relação que deveria ser verificada entre elas*". Inconsistências surgem da elicitação de requisitos de múltiplos *stakeholders* e de suas representações nas especificações parciais do processo. Surgem também com a evolução dos sistemas e a inclusão de novos requisitos à especificação existente. Inconsistências que ocorrem nos estágios iniciais do processo de *software* são mais:

- freqüentes;
- toleráveis;
- importantes sob a ótica gerencial, uma vez que seu impacto nos estágios seguintes do processo de desenvolvimento pode ser crítico.

Easterbrook et al [EASTERBROOK 94a], afirmam que manter a consistência entre os diversos documentos produzidos por diferentes equipes trabalhando nas atividade de engenharia de requisitos pode inviabilizar a realização do trabalho. Assim, é preciso reconhecer que, muitas vezes é desejável tolerar e, até mesmo, encorajar o surgimento de inconsistências. Dessa forma estaremos maximizando a liberdade do projeto e evitando a

[21] Dizemos que dois requisitos são inconsistentes entre si quando existe um conflito entre eles.

tomada prematura de decisões, caracterizando assim gerenciamento de inconsistências como uma atividade mais importante que a manutenção de consistência.

Nuseibeh e Easterbrook [NUSEIBEH 99] definem uma infra-estrutura para a atividade de gerenciamento de inconsistências, dividindo-a em quatro fases:

- **monitorar inconsistências** - onde um conjunto de descrições é monitorado quanto à consistência com base em um conjunto de regras, visando identificar as inconsistências, tão logo seja possível;

- **diagnosticar inconsistências** - uma vez identificada uma inconsistência, diagnosticá-la consiste em: localizá-la, identificar sua causa e classificá-la, para que seja possível escolher uma estratégia para administrá-la;

- **administrar inconsistências -** consiste em adotar uma estratégia para acompanhar a inconsistência identificada até sua resolução. As estratégias vão desde ignorar as inconsistências até seu melhoramento, sem necessidade de resolução imediata (através de mais elicitação, por exemplo);

- **medir inconsistências e analisar impacto e risco** - a escolha da estratégia dependerá da existência de medidas de progresso e impacto. Isso torna as inconsistências mensuráveis, permitindo sua análise, avaliação, verificação e validação.

Negociação de Requisitos

No centro das atividades conduzidas para a resolução de conflitos e inconsistências está a negociação de requisitos. Negociação é o processo através do qual os *stakeholders* interagem visando a obtenção de comprometimento mútuo **Erro! A origem da referência não foi encontrada.**. É uma atividade difícil, pois um entendimento global de todos os objetivos, soluções e da interação entre eles, pode ser extremamente complexo. Assim como no desenvolvimento de *software*, diferentes contextos de negociação exigem diferentes métodos de negociação.

Diversas técnicas foram desenvolvidas para auxiliar na negociação de requisitos. Robinson [ROBINSON 89], em 1989, introduz o conceito de objetivos do domínio na integração de múltiplas perspectivas. Em seu trabalho, Robinson considera perspectivas como linhas de especificação paralelas que deverão ser integradas em algum ponto no tempo. Nesse contexto, uma árvore de objetivos do domínio é construída e utilizada para resolução de conflitos na integração das especificações.

Mais tarde Robinson [ROBINSON 98] estabelece uma infra-estrutura para modelar o processo de negociação. De acordo com ele, o processo de negociação é constituído de três dimensões:

- perspectivas de negociação, que classificam os participantes do processo em papéis que permitem estabelecer o comportamento esperado do participante no processo de negociação;

- processos de negociação, que caracterizam as fases através das quais a negociação é conduzida. As fases iniciais desse processo concentram-se nos requisitos dos participantes e as fases finais definem como os requisitos podem ser satisfeitos.

- produtos da negociação, constituídos dos itens sobre os quais os participantes trabalham ao longo do processo de negociação e cuja evolução durante o processo de negociação deve ser rastreada. Entre os produtos da negociação, Robinson cita modelos de agentes, alternativas, conflitos, tratativas e seus inter-relacionamentos.

Em 1995, Boehm [BOEHM 95] introduz uma variante do modelo espiral, o Processo Espiral Win-Win, endereçando especificamente o problema da negociação que ocorre no contexto do desenvolvimento de *software*. Nesse processo, o envolvimento explícito e contínuo de cada um dos *stakeholders* é valorizado através das diversas fases do processo, com vistas a estabelecer um conjunto explícito de objetivos para o desenvolvimento e definição colaborativos do *software*, identificando atividades de colaboração no contexto do modelo de ciclo de vida do *software*.

Como em qualquer atividade de engenharia, as técnicas desenvolvidas têm que se adaptar a um cenário de recursos cada vez mais escassos. Assim, torna-se muito importante para auxiliar no processo de negociação, que se tenha um mecanismo para classificar os requisitos durante o processo. Nessa direção, Karlsson [KARLSSON 97] introduziu uma técnica baseada no AHP (*Analytic Hierarchy Process*), que compara os requisitos dois a dois de acordo com seu valor e custo, visando estabelecer uma relação de prioridade entre todos os requisitos do sistema. Segundo Karlsson, a comparação em pares introduz muita redundância e diminui percepções subjetivas, minimizando erros de avaliação comuns em técnicas que utilizam critérios absolutos.

4.5.4. Documentação de Requisitos

Uma vez identificados e negociados, os requisitos devem ser documentados para que possam servir de base para o restante do processo de desenvolvimento. Esta é uma área do processo

de requisitos onde há muita controvérsia e ambigüidade na literatura.

Kotonya & Sommerville [KOTONYA 98] afirmam que a linguagem natural é a única capaz de ser compreendida por todos os participantes do processo de requisitos e, dessa forma, deve ser utilizada para representá-los. Contudo, reconhecem que o foco principal é o entendimento e a acuidade dos requisitos especificados, de forma que sempre que necessário, pode-se incluir à documentação dos requisitos diagramas, gráficos, tabelas entre outros que facilitem o entendimento e a elucidação das informações registradas no documento de requisitos.

Davis [DAVIS 93] destaca que, apesar da vantagem da linguagem natural, que pode ser entendida por todos os participantes do processo de requisitos, ela apresenta um alto grau de ambigüidade, o que favorece o aparecimento de inconsistências. Segundo ele, isso acontece, pois, quando nos comunicamos verbalmente, parte dos elementos utilizados em nossa expressão oral na linguagem natural - a expressão corporal - é perdida. Dessa forma, conseguimos registrar apenas parte das informações que expressamos.

Diversas iniciativas tentaram contornar esse problema. Leite [LEITE 93] introduziu o LEL - Léxico Estendido de Linguagem. Seu objetivo era restringir e definir precisamente o vocabulário utilizado nas especificações de requisitos. Fuchs e Schwitter [FUCHS 96] propuseram uma linguagem de especificação denominada ACE – *Attempto Controlled English*, na qual padrões de escrita restringem a gramática e o vocabulário, permitindo a documentação de requisitos de forma mais previsível e menos ambígua. ACE é um subconjunto do Inglês para escrever especificações de requisitos. Especificações escritas nessa linguagem dão a impressão de serem informais. São, contudo, passíveis de execução computacional.

Entre os muitos problemas que enfrentamos na documentação de requisitos, certamente, administrar o grande volume de informações gerado pelo processo de requisitos é um dos principais. O uso de uma notação gráfica certamente diminuiria o tamanho do modelo, tornando mais fácil a atividade de administrar a documentação dos requisitos. Isto faz com que se utilize notações tradicionalmente empregadas na atividade de projeto para modelar requisitos. Apesar de ser fato a necessidade de se utilizar alguma notação para podermos administrar o problema de requisitos, utilizar notações de projeto pode polarizar a solução em favor da implementação [ZAVE 97b].

A intensificação do uso das técnicas de orientação a objetos e a introdução da UML para modelar artefatos de *software* baseados em objetos, em 1997, vieram a agravar ainda mais a situação. UML foi introduzida sem um processo que orientasse seu uso como uma linguagem de representação de requisitos e, dessa

forma, pode, mais facilmente, polarizar a solução em favor da implementação. Os métodos de desenvolvimento de sistemas orientados a objetos utilizam-se de um artifício para aproximar as atividades do processo de *software*: utilizar o mesmo formalismo entre as estruturas de cada atividade [LOPES 02]. Contudo, é importante que, em cada atividade, se tenha a noção exata da semântica de cada elemento utilizado. De outra forma aumentamos muito os riscos de polarizar a solução em favor da implementação. É fato, contudo, que tais notações são empregadas na modelagem de requisitos.

Outras linguagens, no entanto, estão sendo desenvolvidas para tal. É o caso, por exemplo, de KAOS, uma linguagem baseada em objetos, agentes e seus objetivos, desenvolvida exclusivamente para modelar requisitos [DARDENE 93]. Ainda na linha de objetivos, porém estendendo o conceito para o contexto organizacional e para o caso particular de sistemas de informação[22], Yu [YU 97] desenvolveu a técnica denominada i* (i estrela).

i* é utilizada nas fases iniciais do processo de requisitos para raciocinar sobre ambientes organizacionais e seus sistemas de informação. Modela, basicamente, a relação entre vários atores do contexto organizacional (Modelo de Dependência Estratégica) e a descrição de interesses e responsabilidades de cada um deles, caracterizando como os objetivos e interesses de cada um deles podem ser atendidos através das várias configurações de sistema e ambiente (Modelo de Raciocínio Estratégico).

Outros conceitos foram desenvolvidos para a documentação de requisitos não funcionais, como é o caso de *Softgoals*, utilizados para caracterizar objetivos que podem ser parcialmente atendidos [MYLOPOULOS 99]. Objetivos exercem influência sobre outros objetivos. Essa influência pode ser favorável a seu atendimento (influência positiva) ou contrária a seu atendimento (influência negativa). Um *softgoal* é considerado atendido se existem mais influências positivas que negativas sobre ele.

Esse conceito é fundamental para tratar requisitos não funcionais cujo atendimento é fortemente baseado em soluções de compromisso. Por exemplo, mais desempenho aumenta o consumo de memória e pode requerer a construção de algoritmos especiais, o que será mais caro. Assim, a decisão sobre qual o nível de desempenho adequado depende do consumo de memória tolerável, do custo que se está disposto a pagar e do desempenho mínimo necessário para não comprometer a aplicação em desenvolvimento.

---

[22] Sistemas de informação são uma categoria particular de sistemas que utilizam um banco de dados para manter um espelho das informações do mundo real e apoiar processos de negócio. Veremos mais adiante, em detalhes, algumas categorias de sistemas.

4.5.5. Validação de Requisitos

Esta é a última atividade do processo de requisitos, ocorrendo ao final da elaboração da Especificação de Requisitos do Software. Seu objetivo é verificar o documento quanto à sua consistência, completitude e precisão.

De acordo com Kotonya e Sommerville [KOTONYA 98], deve-se fazer uma distinção entre análise de requisitos e validação de requisitos. Apesar de ambas as atividades terem muito em comum - ambas envolvem analisar os requisitos, julgar se são uma descrição apropriada das necessidades dos *stakeholders* e verificar os requisitos com relação a presença de problemas - existem diferenças importantes que justificam o tratamento de ambas em separado.

Análise de requisitos lida com requisitos em seu estado primitivo, tal como elicitados dos *stakeholders*. Tais requisitos são, em geral, informais, descritos de forma incompleta e não estruturada e, freqüentemente, representados através de uma mistura de notações. O objetivo durante a análise de requisitos é garantir que os requisitos atendam às necessidades dos usuários e não aos detalhes da descrição de requisitos. Kotonya e Sommerville [KOTONYA 98] descrevem a atividade com uma frase: *"Requirements analysis should, mostly be concerned with answering the question: have we got the right requirements? "*

Por outro lado, validação de requisitos relaciona-se com a verificação da versão final do documento. Nesta versão já foram removidas todas as inconsistências e omissões conhecidas e o documento já deve obedecer a padrões de qualidade. O principal foco da atividade de validação de requisitos é a forma através da qual os requisitos são descritos. Kotonya e Sommerville [KOTONYA 98] caracterizam validação de requisitos como a atividade na qual devemos responder a seguinte questão: *"have we got the requirements right?"*

A principal técnica utilizada na validação de requisitos é basicamente aquela utilizada em outras atividades do processo de *software*: revisões. Além dessa técnica básica, são também utilizadas validação de protótipo, validação de modelos e teste.

Revisões

Revisões consistem de reuniões estruturadas nas quais um grupo de pessoas, prévia e cuidadosamente escolhidas, após lerem e analisarem o documento de requisitos, reúnem-se para discutir os problemas encontrados e chegar a uma solução de consenso em relação às ações a serem adotadas para corrigi-los.

Como uma reunião formal, deve ser nomeado um presidente que irá conduzir os trabalhos e um relator, para produzir uma ata na qual constem os problemas identificados e as ações pelas quais se decidiu na reunião. Neste ponto, as revisões de requisitos diferem das revisões de outros documentos do processo de

software. Quando se revisa programas, por exemplo, é comum que ao se encontrar um erro, o código seja devolvido ao autor para correção. Diferente do que ocorre com código fonte, problemas nos requisitos envolvem discussão e soluções de compromisso e, dessa forma, as ações a serem adotadas são discutidas na reunião de revisão e registradas na ata.

Devido ao grande volume de trabalho envolvido com essas revisões, elas são lentas. Kotonya e Sommerville [KOTONYA 98] estimam que, aproximadamente, de 20 a 40 requisitos poderiam ser revisados por hora, dependendo de seu tamanho. Assim, seria necessário um esforço de, aproximadamente, 50 homens-hora para revisar um documento com 400 requisitos, considerando-se uma equipe de revisão com quatro pessoas.

Para auxiliar nessa tarefa trabalhosa, podemos adotar algumas técnicas que simplificam a condução dos trabalhos. A revisão de conformidade com os padrões pode ser feita como uma atividade de pré-revisão. Neste caso problemas como erros de grafia e não conformidade com os padrões da empresa poderiam ser tratados em separado, talvez até de forma automática.

A escolha das pessoas que irão compor a equipe de revisão também deve ser cuidadosa. Se possível a equipe deverá conter um usuário final, um representante do cliente, um ou mais especialistas do domínio, engenheiros de *software* (que serão responsáveis pelo projeto e implementação do sistema) e engenheiros de requisitos. Pessoas com diferentes bases de conhecimento trazem mais experiência à reunião de revisão e tornam mais provável a identificação de problemas nos requisitos. Além disso, ao discutir os problemas identificados com profissionais de outras especializações, os *stakeholders* tendem a entender melhor as razões que levaram à mudanças nos requisitos por eles propostos.

Outra técnica que facilita a condução das revisões é a utilização de listas de verificação. Essas listas funcionam bem na revisão de código. Principalmente porque programadores cometem os mesmos tipos de erros através dos programas que constroem. Apesar de o mesmo ser válido no caso dos engenheiros de requisitos, as similaridades dos erros são, em geral, ocultadas pela diversidade dos domínios de aplicação. Assim, de forma geral, ao invés de prescreverem verificações individuais para cada requisito, os itens das listas de verificação para requisitos estão mais ligados à qualidade geral do documento e ao relacionamento entre requisitos.

Validação por Protótipo

Se durante a atividade de elicitação foi adotada uma abordagem por construção de protótipo, validar os requisitos documentados através do protótipo pode ser uma boa idéia. Por outro lado, se um protótipo não estiver disponível, construí-lo apenas para a validação dos requisitos não será economicamente atraente. Um

protótipo para validação deverá ser mais completo que o protótipo de elicitacão, cujo objetivo principal era dar uma idéia aos *stakeholders* do sistema que se estava imaginando.

Um protótipo para validação dos requisitos deverá incluir um número razoável de facilidades de forma a possibilitar o uso prático do sistema. Caso contrário os *stakeholders* não poderão utilizar o sistema de uma forma natural, invalidando o esforço de validação.

Alguns cuidados são necessários ao se adotar uma abordagem de validação por protótipo. Um cuidado importante é escolher adequadamente as pessoas que irão testar o protótipo. Elas deverão ser usuários experientes e abertos a inovações de forma a poder analisar criticamente a proposta do novo sistema. Deverão ser desenvolvidos cenários de uso do sistema para que as pessoas sigam o roteiro e, de fato, exercitem as funcionalidades que se deseja testar. Caso contrário as pessoas poderão dispersar-se em torno das novas funcionalidades e acabar por não testar o sistema tal como necessário. Ao serem encontrados, os problemas deverão ser documentados de forma sistemática, se possível, através de um modelo fornecido antecipadamente para que todas as informações relevantes sejam registradas.

Outro cuidado importante é disponibilizar uma versão preliminar do manual do usuário. Os *stakeholders* estarão lidando com uma versão experimental do sistema. É de se esperar que problemas aconteçam e, portanto, que o manual explique como fazer para reiniciar ou finalizar o sistema de uma forma segura.

Validação de Modelos

Como visto anteriormente, a documentação de requisitos pode conter modelos desenvolvidos em diversas notações. Segundo Kotonya e Sommerville [KOTONYA 98] três são os objetivos desta atividade:

- Demonstrar que cada um dos modelos é consistente de forma a conter toda a informação necessária e a não haver conflitos entre suas diferentes partes;

- Demonstrar, na existência de diversos modelos, que eles são consistentes externa e internamente. Isto inclui verificar nomenclatura através dos modelos, consistência das referências múltiplas, entre outras;

- Demonstrar que os modelos refletem precisamente os requisitos reais dos *stakeholders*. Esta é a parte mais difícil da validação de modelos, uma vez que consiste na produção de argumentos capazes de convencer que o modelo realmente representa os requisitos.

Teste de requisitos

Uma das qualidades de um requisito bem elaborado é que ele possa ser testado. Uma boa maneira de identificar problemas nos requisitos, tais como falta de completitude ou ambigüidade, é tentar propor casos de teste para um requisito. É preciso ter em mente que o objetivo de se propor casos de teste nesta fase é validar o requisito e não o sistema.

Ao testarmos requisitos, tal como na atividade de testes, devemos registrar cuidadosamente o resultado dos testes. Assim, convém fornecer um modelo a quem for testar os requisitos para que o resultado do teste seja registrado a contento.

4.5.6. Gerenciamento de Requisitos

Requisitos possuem uma natureza volátil. Diversos fatores contribuem para sua instabilidade ao longo do tempo. Mudanças externas no ambiente (mudanças de legislação, mudanças no mercado, mudança no posicionamento estratégico da empresa), erros incorridos no processo de requisitos, entre outros. Todos esses fatores fazem com que seja necessário alterar os requisitos. Tais alterações precisam ser conduzidas de forma ordenada para que não se perca controle sobre o prazo e o custo do desenvolvimento. Denominamos a atividade de administrar os requisitos ao longo do tempo de gerenciamento de requisitos.

Os benefícios desta atividade são percebidos no médio prazo sendo que são necessários investimentos no curto prazo. Assim, a atividade é, muitas vezes, tida como um fardo desnecessário à condução do processo. Contudo, sua não implementação faz com que as economias de curto prazo sejam logo suplantadas pelas despesas no longo prazo, verificadas com superação de custo e prazo nos projetos conduzidos.

Rastreabilidade

No centro da atividade de gerenciamento de requisitos está a rastreabilidade. Rastreabilidade [JARKE 98a] é definida como a habilidade de se acompanhar a vida de um requisito em ambas as direções do processo de *software* e durante todo o seu ciclo de vida.

Segundo Jarke [JARKE 98a] existem quatro tipos de rastreabilidade:

- **a partir dos requisitos** (*forward from requirements*) - que vincula a responsabilidade por cada um dos requisitos aos componentes do sistema. Através deste tipo de rastreabilidade seremos capazes de vincular, por exemplo, cada uma das estruturas do projeto aos requisitos que a originaram;

- **de volta aos requisitos** (*backward to requirements*) - que verifica a aderência do sistema desenvolvido a seus requisitos. Deve-se evitar projetar funcionalidades para as quais não existem requisitos, uma prática conhecida como *gold-plating.* Assim, através deste tipo de rastreabilidade seríamos capazes de, por exemplo, identificar quais requisitos são suportados por determinado trecho de código;

- **em direção a novos requisitos** (*forward to requirements*) - que acompanha as mudanças das necessidades dos usuários e nas condições técnicas assumidas, avaliando seu impacto na relevância dos requisitos definidos. Analisando novas necessidades seremos capazes de formular novos requisitos e identificar quais requisitos sofrem impacto com a inclusão dos novos requisitos formulados;

- **de volta a partir dos requisitos** (*backward from requirements*) - que associa os requisitos às estruturas que contribuíram para originá-los, cruciais na validação dos requisitos, especialmente em cenários altamente políticos. Através destes registros poderemos dizer o objetivo e o *stakeholder* responsáveis pela existência de determinado requisito na especificação de requisitos do sistema[23].

Segundo Pinheiro [PINHEIRO 00], o processo de desenvolvimento de *software* deixa rastros cuja identificação e acompanhamento é função da rastreabilidade. Assim, ele define rastreabilidade de requisitos como a habilidade de definir, capturar e acompanhar os rastros deixados pelos requisitos nos outros elementos do ambiente de desenvolvimento de *software* e os rastros deixados por esses elementos nos requisitos.

A dificuldade envolvida com a rastreabilidade está no grande volume de informações gerado. As informações do processo de requisitos devem ser catalogadas e associadas aos outros elementos de forma que possam ser referenciadas através dos diversos itens de informação registrados. É um trabalho extenso que, sem o auxílio de ferramentas de automação adequadas, muito provavelmente, não poderá ser feito.

Ramesh [RAMESH 98] conduziu uma ampla pesquisa sobre a influência e impacto do ambiente organizacional e técnico sobre a adoção e implementação da rastreabilidade nas organizações. O resultado de sua pesquisa não chega a ser surpreendente e reflete as afirmações acima. Ramesh divide as organizações em dois grupos: organizações que empregam um nível básico de

[23] Devemos notar aqui que a interdependência entre os requisitos não se encaixa diretamente em nenhuma das categorias propostas e é um elemento de controle essencial na implementação da rastreabilidade.

rastreabilidade e organizações que utilizam técnicas mais sofisticadas.

Integrantes do primeiro grupo encaram rastreabilidade como uma exigência dos patrocinadores do projeto, utilizam esquemas de rastreabilidade simples e registram basicamente inter-relacionamentos entre requisitos e entre os componentes do sistema. Não capturam nem registram a razão por trás de cada requisito.

Já os componentes do segundo grupo encaram rastreabilidade como uma tarefa fundamental na melhoria do processo de *software*. Assim, empregam esquemas de controle mais ricos, com captura e registro precisos das razões dos elementos rastreados. Dessa forma, conseguem um melhor uso das informações relativas à rastreabilidade.

Rastreabilidade, tal como o controle do processo de *software*, é uma questão de cultura. A equipe técnica precisa empenhar-se na atividade e vislumbrar benefícios pessoais para auxiliar na implementação da rastreabilidade. Como fator cultural, precisa ser desenvolvido através de toda a equipe da organização e, portanto, sua implementação é cara e vagarosa.

5. A Taxonomia Proposta

Engenharia de Requisitos é um amplo e multidisciplinar campo de pesquisa, envolvendo ramos das ciências sociais e das ciências exatas, uma vez que se relaciona com a tradução de observações informais à linguagens matemáticas de especificação formal [ZAVE 97a].

Segundo Zave [ZAVE 97a] a heterogeneidade dos tópicos geralmente discutidos no âmbito da Engenharia de Requisitos faz com que a proposição de uma classificação seja uma tarefa complexa. A saída natural para essa dificuldade é a classificação através de diversas dimensões ortogonais que, quanto mais numerosas, mais aumentam a precisão da classificação. Penalizam, contudo, o entendimento tornando-a complexa demais para utilização. Nas seções que se seguem é apresentada uma discussão das possíveis dimensões de uma classificação e a seguir, as dimensões utilizadas na taxonomia do presente trabalho são discutidas em maiores detalhes.

5.1. *As dimensões de uma classificação*

Diversos são os problemas analisados no contexto da pesquisa de requisitos. Problemas relacionados à comunicação, seja ela escrita ou verbal, questões relacionadas à organização de grande quantidade de informação, problemas relacionados à coordenação do trabalho de múltiplos profissionais com interesses distintos sobre o mesmo tema, problemas relacionados à integração desses diferentes interesses em um documento consistente e capaz de apoiar o desenvolvimento do *software*, problemas relacionados à administração da evolução dos sistemas e do impacto dessa evolução nos requisitos, entre tantos outros.

Entre as possíveis dimensões para uma classificação da Engenharia de Requisitos, discutiremos nas seções seguintes cinco delas:

- **o problema original** - como apontado por Pohl [POHL 94] representa um dos três problemas principais da Engenharia de Requisitos: especificação, representação e consenso. Outros problemas que enfrentamos na Engenharia de Requisitos são oriundos da abordagem que utilizamos para resolver os problemas originais;

- **o domínio de aplicação** - representa uma divisão importante das técnicas aplicáveis, uma vez que certos tipos de aplicação exigem maior ou menor rigor em uma ou outra dimensão da Engenharia de Requisitos, tais como definidas por Pohl [POHL 94];

- **a natureza do problema** - um possível critério para a análise e escolha das técnicas mais adequadas à condução da Engenharia de Requisitos é a natureza do problema. Agrupar os problemas de acordo com a dificuldade específica que pretendem vencer pode ser muito útil na classificação da pesquisa;

- **a natureza da solução** - analisar as dificuldades envolvidas na construção da solução pode indicar, indiretamente, as técnicas mais adequadas a uma ou outra situação;

- **a atividade do ciclo de vida** - para entendermos como a atividade de Engenharia de Requisitos é conduzida na prática, é fundamental conhecermos em que momento no processo de requisitos nos deparamos com determinado problema, ou utilizamos uma técnica ou conceito.

Nas seções a seguir discutiremos em mais detalhes cada uma dessas possíveis dimensões.

5.1.1. O problema original

Pohl [POHL 94] sustenta que um primeiro passo no sentido de se chegar ao cerne da Engenharia de Requisitos é distinguir entre dois tipos de problema:

- problemas originais da engenharia de requisitos;

- problemas originados pelas abordagens utilizadas na solução dos problemas originais.

Analisando o processo de requisitos a partir deste ponto de vista, Pohl examina as **entradas iniciais** do processo e a **saída desejada**, concluindo serem três os principais problemas, ou dimensões, originais da engenharia de requisitos

- **especificação** – que lida com o grau de entendimento dos requisitos em um ponto no tempo;

- **representação** – que trata da forma utilizada para expressar o conhecimento adquirido sobre o sistema[24];

- **consenso** – que lida com o grau de concordância que se atingiu a respeito de uma especificação.

Segundo a dimensão da **especificação** o objetivo do processo de requisitos é levar de um entendimento vago e esparso do sistema, em sua fase inicial, a um entendimento completo e detalhado no final do processo. A dimensão de **representação** objetiva expressar o conhecimento adquirido sobre o sistema durante o processo de requisitos e leva de representações informais, característica de *stakeholders* menos especializados, à representações formais, necessárias à construção do *software* para o sistema. Segundo a dimensão de **consenso,** o processo

[24] A palavra sistema é utilizada de forma muito ambígua na literatura da área. Sistema caracteriza todo um contexto do mundo real no qual estão envolvidos *software*, *hardware* e pessoas. A palavra sistema utilizada neste contexto refere-se a esse sentido amplo, uma vez que trata da representação de requisitos, uma característica mais do mundo real do que do mundo da máquina (*hardware* e *software* utilizados para executar um programa de computador que irá interagir com o mundo real).

de requisitos visa levar os *stakeholders* de sua percepção individual sobre os requisitos a uma visão única e que reflita o acordo a que chegaram durante o processo de requisitos. Pohl denomina essa percepção de visão comum.

Essas três dimensões principais não são independentes, uma vez que o avanço em uma das direções produz reflexo em outra (ou outras). O avanço na direção de uma especificação formal é um exemplo disso. Melhoramos a dimensão da **representação**, uma vez que conhecimento informal está sendo traduzido a uma representação formal. Contudo, durante essa formalização, pode-se detectar contradições na representação formal. A solução dessas contradições exige que se estabeleça um processo de comunicação, o que demandará uma melhoria na dimensão de **consenso**. Como efeito colateral, pode-se identificar um novo requisito. A integração do novo requisito bem como o consenso obtido com a negociação da contradição leva a uma melhora da dimensão de **especificação**.

Além da estrutura identificada com as três dimensões, outros fatores relacionados ao ambiente no qual acontece a Engenharia de Requisitos afetam o processo, positiva e negativamente. Pohl cita cinco fatores principais relacionados ao ambiente:

- **métodos** - o processo é influenciado pelos métodos utilizados em seu contexto, uma vez que concentram-se em diferentes aspectos (a utilização de análise estruturada deverá produzir uma especificação formal totalmente diferente daquela obtida através do uso de análise orientada a objetos);

- **ferramentas**- a especificação final depende das ferramentas utilizadas durante o processo. Se uma ferramenta de análise baseada em representações formais for utilizada, inconsistências poderão ser identificadas que, de outro modo, estariam presentes na especificação final;

- **aspectos sociais** - o ambiente social da equipe de engenharia de requisitos afeta sua eficiência no trabalho. Um ambiente no qual as pessoas se entendem e se relacionam melhor produzirá um resultado muito melhor;

- **competência** - pessoas têm competências e qualificações distintas. Se pessoas com as corretas qualificações e melhor competência conduzirem o processo de requisitos, o resultado final é freqüentemente melhor;

- **restrições econômicas** - limitam os recursos (pessoas, ferramentas, tempo, etc.) que podem ser usados durante o processo. Nem sempre podemos afirmar que com mais recursos um melhor resultado poderia ser obtido. Contudo, se os recursos estiverem abaixo de determinado limite, o resultado certamente será de menor qualidade.

A Tabela 2 resume os problemas apresentados por Pohl.

Categoria de problemas	Fatores que os influenciam
Problemas originais	Especificação
	Representação
	Consenso
Problemas originados pelas abordagens adotadas	Métodos
	Ferramentas
	Aspectos sociais
	Competência
	Restrições econômicas

Tabela 2 - Problemas da Engenharia de Requisitos segundo Pohl

A classificação de Pohl analisa o processo de requisitos a partir de uma macro visão. Nesse nível de abstração, não se fez distinção entre diversos fatores, aplicáveis a situações específicas que surgem da especialização do processo quando utilizado em situações particulares. Vejamos então os fatores que podem influenciar e exigir que o processo seja especializado para resolver problemas específicos. Tais fatores, certamente, influenciarão as dimensões de uma classificação da pesquisa de Engenharia de Requisitos.

5.1.2. Domínio de aplicação

Segundo Zave [ZAVE 97a], um dos fatores que influenciam a condução do processo de requisitos é o domínio de aplicação. Domínio de aplicação [JACKSON 95] é uma classe de aplicações utilizada para resolver os problemas específicos de uma área de aplicação. Refere-se à área geral na qual o sistema será utilizado. A norma IEEE Std 1362 [IEEE 1362-1998] define domínio do problema como sendo "*um conjunto de problemas similares que ocorrem em um ambiente e levam a soluções comuns*".

Jackson [JACKSON 95] faz distinção entre dois termos: **contexto do problema** e **domínio de aplicação**. **Contexto do problema** é a parte do mundo na qual a máquina[25] será instalada, seus benefícios poderão ser identificados e avaliados. **Domínio de aplicação** é a parte do mundo na qual o cliente está interessado e que é relevante para a solução de um problema particular. Estão incluídas aí pessoas, produtos da companhia, outros sistemas baseados em computador, edifícios, conceitos intangíveis – tais como imagens gráficas, quadros de

[25] Máquina [JACKSON 95] é um conjunto de *hardware* e *software* que irá interagir com o mundo real para modificá-lo de forma que passe a se comportar da forma como desejado.

horário, tabelas de salários – ou qualquer outra coisa que irá interagir com a máquina ou que componha o cenário de seu processamento[26].

Para poder interagir com o mundo real, a máquina deve estar conectada a ele, isto é, a máquina e o mundo real devem compartilhar um conjunto de fenômenos (*shared phenomena*) – informação, eventos, dados entre outros conceitos que existem no mundo real e que devem ser percebidos pela máquina.

Vejamos alguns exemplos. Um sistema de contas a pagar existe em um domínio de aplicação onde estão presentes fornecedores, faturas, pagamentos, vencimentos, fluxo de caixa, e etc. O principal problema envolvido é fazer com que os fornecedores sejam pagos de acordo com suas faturas, com o pedido ao qual se referem e segundo uma política de fluxo de caixa adequada.

Neste caso, fenômenos compartilhados entre a máquina e o mundo real são, entre outros, identificação dos pedidos, dados dos fornecedores, vínculos entre os pedidos e os fornecedores, identificação das faturas, vínculo entre as faturas e os pedidos, datas de vencimento e emissão dos documentos, a política (ou os parâmetros para a condução da política) de fluxo de caixa, entre outros. Para compartilhar tais fenômenos com o mundo real, a máquina irá depender, principalmente, de operadores humanos que irão fornecer as informações através de digitação de dados em telas do sistema.

Um sistema de controle de tráfego aéreo em um aeroporto deve monitorar e apresentar aos controladores humanos, decolagens e aterrissagens, posição dos aviões (para mantê-los suficientemente afastados), tempo de espera na fila (quer de decolagem quer de aterrissagem) respeitando em seus cálculos normas de segurança estabelecidas por órgãos reguladores.

No conjunto de fenômenos compartilhados entre o mundo real e a máquina, neste caso, estão: a identificação dos aviões, sua posição, a identificação dos controladores de vôo, a identificação dos vôos, o vínculo entre os aviões e os vôos, o vínculo entre os controladores e os vôos controlados por ele, dados sobre cada um dos vôos, as normas de segurança a serem respeitadas nos cálculos de posicionamento (ou parâmetros que permitam conduzi-la), etc. Essas informações deverão chegar ao sistema de várias formas. Parte delas, dado o nível crítico da atividade, deverão ser inseridas no sistema através de interação direta com

[26] Devemos observar aqui que a definição de Jackson apresenta desvios em relação à forma como o termo é geralmente utilizado na literatura da área. Apesar de não se encontrar a definição explícita na bibliografia de Engenharia de Requisitos ou Engenharia de Software, o termo é utilizado no sentido de designar uma classe de aplicações utilizada para resolver os problemas específicos daquela categoria industrial ou de negócios. Refere-se à área geral na qual o sistema será utilizado. A definição de Jackson, por sua vez, aproxima-se mais do conceito introduzido por Shlaer & Mellor, no qual o desenvolvimento de sistemas deve lidar com diferentes temas ou focos de interesse, denominados domínios.

outros subsistemas como, por exemplo, o radar, que deverá fornecer a posição dos aviões no ar.

No caso do sistema de controle de tráfego aéreo a natureza dessa interação é fortemente restrita no tempo de seu processamento, uma vez que o sistema deverá efetuar os cálculos baseando-se sempre na informação mais recente possível. Um erro nesse processamento pode fazer com que os aviões se aproximem perigosamente, colocando em risco vidas humanas.

É fácil ver que o segundo sistema estará sujeito a restrições especificadas em limites muito inferiores que o primeiro. Essa diferença faz com que a abordagem da engenharia de requisitos para o primeiro sistema seja diferente da abordagem do segundo caso[27]. Assim, o domínio de aplicação irá exercer influência sobre como as atividades serão conduzidas em cada uma das dimensões apresentadas por Pohl, influenciando a qualidade da especificação no final do processo. Mais ainda, os fatores identificados como problemas não originais terão maior influência sobre o resultado do processo para determinados domínios de aplicação do que para outros.

Do ponto de vista do desenvolvimento de sistemas, o interesse na classificação de domínios de aplicação está na possibilidade de se identificar dificuldades e/ou soluções comuns que possam servir de base para o novo desenvolvimento[28]. Através de diferentes domínios poderemos enfrentar dificuldades semelhantes. Por exemplo, sistemas de automação de processos industriais sofrem severas restrições de tempo, uma vez que tem que atuar nas máquinas de acordo com os limites de tempo do processo de fabricação do produto sendo controlado. O desenvolvimento desses sistemas assemelha-se bastante com o desenvolvimento de sistemas destinados a controlar equipamentos médicos, onde as restrições de tempo também são críticas.

Assim, as semelhanças e diferenças relativas às aplicações estão mais no problema a ser resolvido do que no ramo de utilização do sistema a ser desenvolvido.

5.1.3. Natureza do problema

Para um dado domínio de aplicação, é preciso entender exatamente o problema a ser resolvido. Jackson [JACKSON 95] afirma que o problema específico a ser resolvido situa-se no

---

[27] As técnicas aplicadas aqui deverão garantir a correção dos requisitos especificados. Elas são diferentes daquelas aplicadas a outros tipos de sistema, para os quais tal exigência imponha custos muito elevados.

[28] Observe aqui que estamos tratando o assunto do ponto de vista da construção de especificações. Do ponto de vista global de desenvolvimento de software, existem outros interesses, como, por exemplo, reuso de componentes.

contexto do problema. Porém, é preciso identificar exatamente o que se pretende resolver, delineando-se o problema em si. Ainda segundo Jackson [JACKSON 95], somente através de um perfeito entendimento do problema a ser resolvido é que poderemos escolher os métodos adequados a resolvê-lo.

Jackson [JACKSON 95] define *problem frames* como a generalização de uma classe de problemas. Se muitos outros problemas se encaixarem no mesmo *problem frame* do problema que se estiver tratando, é possível que as mesmas técnicas utilizadas para resolver os problemas anteriores da mesma categoria (*problem frame*) sejam também adequados para resolver o problema atual.

Um *problem frame* é composto de **partes principais** (*principal parts*) e de uma **tarefa de solução** (*solution task*) e consiste da caracterização de uma classe de problemas através da percepção de seus elementos essenciais. Alguns problemas reais talvez não possam ser simplificados dessa forma mas, certamente, poderão ser representados através da utilização de múltiplos *problem frames*, ao que Jacskon [JACKSON 95] chama de *multi-frame problems*.

Assim, um conjunto de *problem frames* pode ser utilizado para caracterizar os métodos disponíveis e que poderão ser utilizados no desenvolvimento de *software* em qualquer atividade do ciclo de vida, em particular na engenharia de requisitos.

5.1.4. Natureza da solução

Apesar de o domínio de aplicação, tal como definido neste texto, agrupar classes de aplicações, não é raro encontrarmos soluções, em diferentes domínios de aplicação, que compartilham das mesmas características construtivas. Assim, a dificuldade encontrada na construção de um sistema de controle de um processo industrial pode ser a mesma encontrada na construção de um sistema de controle de um equipamento médico.

Davis [DAVIS 93] classifica as aplicações em dois grandes grupos, de acordo com:

- o tipo de projeto necessário;
- o tipo de *software* necessário.

Segundo os tipos de projeto, os domínios podem exigir:

- **projetos normais** - lidam com problemas conhecidos e para os quais as soluções são também conhecidas. Nesse caso, existem soluções padrão além de métodos e tecnologias disponíveis para serem utilizadas;

- **projetos revolucionários** - lidam com problemas que ainda não foram resolvidos e/ou tentativas anteriores de resolvê-los falharam. Tais projetos não estão, de fato, no escopo da engenharia.

Segundo o tipo de *software* necessário, Davis [DAVIS 93] classifica os sistemas em quatro categorias diferentes:

- quanto à disponibilidade dos dados e do processamento:
    - **estático** - quando os dados estão disponíveis antes do início do processamento;
    - **dinâmico** - quando os dados continuam a chegar durante o processamento e irão afetar seu resultado;
- quanto ao número de tarefas desenvolvidas simultaneamente (do ponto de vista do usuário):
    - **seqüencial** - quando realiza apenas uma tarefa por vez;
    - **paralelo** - quando múltiplas tarefas são executadas simultaneamente;
- quanto ao aspecto externo mais difícil de especificar:
    - **dados** - a parte mais difícil da especificação do sistema é a definição, descrição, organização e formato dos dados que cruzam a fronteira entre o ambiente e o sistema;
    - **controle** - a parte mais difícil da especificação é a definição e a descrição de como o ambiente irá controlar o sistema e/ou de como o sistema irá controlar o ambiente. Este tipo de aplicação tende a ser muito restritiva em relação ao tempo;
    - **algoritmo** - o aspecto mais complicado é a especificação da função (ou funções) de transformação aplicável à entrada para produzir a saída desejada;

Muitas aplicações apresentam mais de um desses atributos, podendo ser classificadas, quanto à complexidade de especificação, como híbridas.

- quanto a previsibilidade da saída para um mesmo conjunto de dados de entrada;
    - **determinístico** - quando o sistema deve prover sempre a mesma saída para um mesmo

conjunto de dados de entrada. Seus estímulos e respostas são perfeitamente conhecidos e definíveis;

- **não-determinístico** - quando as respostas do sistema não são bem entendidas. Tais sistemas são menos previsíveis. Em geral, tomam decisões com base no significado do conjunto de dados obtidos de fontes variadas, podendo haver mais de uma resposta certa. Esses sistemas tentam imitar o comportamento humano.

5.1.5. Atividade do ciclo de vida

Na seção 4.5 discutimos extensamente o ciclo de vida da Engenharia de Requisitos, apontando suas atividades e as tarefas realizadas no contexto de cada uma delas; delineando um modelo para o ciclo de vida da Engenharia de Requisitos. O modelo de ciclo de vida apresentado é bastante simples, caracterizando as principais atividades desenvolvidas, a realimentação que evidencia sua natureza iterativa, além de apontar as principais entradas e saídas.

Para podermos caracterizar a taxonomia como ontológica, precisamos alocar cada um dos itens da literatura de acordo com sua aplicabilidade nas tarefas conduzidas no contexto do ciclo de vida da Engenharia de Requisitos. Esta dimensão da classificação é importante, pois raramente conseguimos encontrar na indústria processos totalmente aderentes ao modelo teórico. Dessa forma, a partir de um modelo teórico simples e bem caracterizado - com suas atividades e tarefas bem definidas - é possível utilizar a taxonomia como base para a seleção de textos que possibilitem o entendimento e a possível melhoria na condução da tarefa que mais afeta o processo específico utilizado, independente de o ciclo de vida ser o mesmo ou não.

Além disso, o trabalho de aprofundamento do conhecimento em determinado tema é um processo lento e ineficiente. A princípio, não se tem o conhecimento necessário para a seleção dos textos a serem lidos, fazendo com que parte do esforço empenhado seja desperdiçado com a leitura e pesquisa de textos que se mostrarão inúteis em sua contribuição para atingir o objetivo.

A utilização da taxonomia pode ajudar na seleção de textos relacionados ao tema que se pretende estudar, permitindo ao leitor que, ao se deparar em um texto escolhido com uma referência que considere para leitura, possa verificar se a obra,

de fato, trata do tema desejado através de sua pesquisa na taxonomia[29].

5.2. *Algumas classificações propostas*

Encontramos na literatura algumas classificações da Engenharia de Requisitos. Goguen [GOGUEN 94], ao apresentar uma classificação para os métodos utilizados na Engenharia de Requisitos, argumenta que cada um deles possui uma teoria de organização, de forma não articulada, que é, implicitamente, uma teoria sociológica. Dessa forma, uma boa classificação das teorias sociológicas seria uma boa base para uma classificação dos métodos aplicados à engenharia de requisitos.

Assim, a partir do trabalho do filósofo francês Jean-François Lyotard, Goguen propõe a classificação da Figura 5. Segundo essa classificação, os métodos são divididos em dois grandes grupos de teorias de sociedade: *modern*, segundo a qual as sociedades baseiam-se em uma meta-narrativa (ou "*grand unifying story*") para legitimar seu argumento de universalidade, e *post-modern*, que defende que as sociedades são baseadas em "jogos de linguagem local" e não podem ser unificadas ou divididas em partes de forma ordenada.

Teorias *modern* podem ser de dois tipos: *unitary*, que assumem a existência de uma única realidade objetiva e *pluralistic*, que permite a existência de múltiplos grupos na sociedade. Teorias *unitary* podem ser ainda subdivididas em *hard*, que defende ser a organização da sociedade um sistema racional, e *soft* na qual um sistema pode servir a múltiplos objetivos.

Teorias *pluralistic* podem ser subdivididas em *divisive* e *cooperative*. Teorias *divisive* se subdividem em *dual*, que se baseiam no conflito marxista; Engenharia de Requisitos precisa tomar partido, e *critical*, que busca alternativas às condições sociais existentes. Teorias *cooperative* subdividem-se em *democratic*, que procuram envolver todos os pontos de vista de forma democrática e *network* que adotam abordagens evolucionárias.

Claramente a principal dimensão do trabalho de Goguen é a dimensão social, sendo a forma como os participantes do processo se inter-relacionam o fator mais importante da classificação.

Outro trabalho importante de classificação da pesquisa na área de Engenharia de Requisitos é o de Zave [ZAVE 97a] (Tabela 3 - A classificação de Zave). Em seu trabalho, Zave opta por limitar o número de dimensões da classificação em apenas duas, orientadas basicamente

---

[29] É claro que, para que isso fosse possível, a taxonomia deveria classificar grande parte das obras produzidas na área e eventualmente até mesmo aquelas que, apesar de não tratarem de tema diretamente ligado à área, iluminem algum tema relevante para a área. Nitidamente não conseguiremos produzir aqui um trabalho dessa dimensão. Contudo, a classificação produzida é tal que, sua extensão é factível e não muito trabalhosa. Dessa forma ela poderia ser mantida e crescer ao longo do tempo. Veremos na seção 5.3 que isso será possível.

à pesquisa. Zave classifica a pesquisa em duas categorias: Problemas e Contribuições para Soluções.

Na categoria Problemas, estão identificadas tarefas que precisam ser conduzidas no contexto da Engenharia de Requisitos, caracterizando de forma bastante explícita tarefas, problemas reconhecidos e soluções. Na categoria Contribuição para Soluções são destacadas as formas como a pesquisa pode contribuir para a solução de problemas e assume que, na condição de engenheiros de *software*, podemos nos esforçar por entender os fatores sociais, porém, podemos influenciar apenas as práticas técnicas.

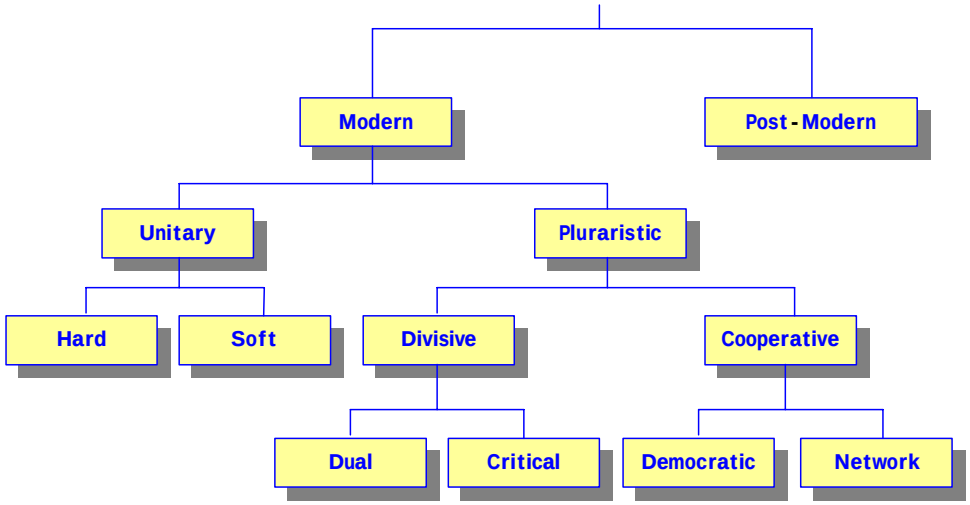

Figura 5 - Classificação dos Métodos para Engenharia de Requisitos[30]

Apesar da simplicidade obtida com a manutenção de apenas duas dimensões na taxonomia, alguns problemas são inevitáveis. Quando algum artigo expande seu escopo através de muitas categorias em uma dimensão, ele pode estar também focado em outra dimensão da mesma classificação, embora menos enfaticamente. É o caso, por exemplo, de trabalhos que propõem um método para a condução da atividade da engenharia de requisitos.

Outro problema, segundo a própria autora, está na dimensão do domínio de aplicação. Em um nível de abstração mais elevado, o processo de requisitos independe do domínio de aplicação; os problemas a serem resolvidos são semelhantes e as macro-atividades podem ser consideradas as mesmas. Contudo, quando se aprofunda nos detalhes de cada uma das macro-atividades do processo, o domínio de aplicação faz bastante diferença, como discutido na seção 5.1.

[30] Extraída de [GOGUEN 94], página 13.

Easterbrook e Nusseibeh [EASTERBROOK 02] produziram uma bibliografia comentada de Engenharia de Requisitos. Apesar de não ser uma classificação propriamente dita, seu trabalho traz implícita uma classificação da pesquisa, cujo objetivo principal é fornecer uma orientação de leitura aos alunos da disciplina Engenharia de Requisitos na Universidade de Toronto. A principal dimensão que se pode intuir do trabalho é a profundidade da abordagem do tema, dividindo-o em *textos genéricos* e *tópicos específicos.*

No contexto de *tópicos específicos*, os textos estão organizados de acordo com a condução das atividades ao longo do ciclo de vida de Engenharia de Requisitos sendo, portanto, a segunda dimensão da classificação o processo de requisitos. No contexto de *textos genéricos*, a classificação está orientada à finalidade do texto, sendo divididos em dirigidos a estudantes e dirigidos à aplicação prática das técnicas. A classificação completa é conforme Tabela 5.

Problemas	**Investigação de objetivos, funções e restrições de um sistema de *software***	Vencendo barreiras de comunicação
		Estratégias para converter objetivos vagos em propriedades específicas ou comportamento
		Entendendo prioridades e graus de satisfação
		Estratégias para alocar requisitos entre os sistemas e os vários agentes do ambiente
		Investigação de objetivos, funções e restrições de um sistema de *software*
		Estimando custos, riscos e cronogramas
		Garantindo completitude
	Especificação do comportamento do sistema de *software*	Integrando múltiplos pontos de vista e representações
		Avaliando estratégias alternativas para satisfazer os requisitos
		Obtendo especificações completas, consistentes e não ambíguas
		Verificando que o sistema especificado irá satisfazer os requisitos
		Obtendo especificações adequadas às atividades de projeto e implementação
	Gerenciamento da evolução de sistemas e de famílias de sistemas	Reutilizando a engenharia de requisitos nas fases evolucionárias
		Reutilizando engenharia de requisitos para o desenvolvimento de sistemas similares
		Reconstruindo requisitos
Contribuição para Soluções	Relatório sobre o estado da prática	
	Proposta de solução orientada a processos	
	Proposta de solução orientada a produtos	
	Caso de Estudo aplicando uma solução proposta a um exemplo substancial	
	Avaliação ou comparação de soluções propostas	
	Proposta de solução orientada a medidas	

Tabela 3 - A classificação de Zave

A Tabela 4, abaixo, ilustra as dimensões através das quais foi classificada a pesquisa na área de Engenharia de Requisitos por cada um dos autores mencionados.

Autor	Dimensões da classificação
Goguen	• Teorias sociológicas
Zave	• Problemas e soluções
Easterbrook & Nusseibeh	• Profundidade da abordagem do tema • Processo de requisitos

Tabela 4 - As dimensões das classificações apresentadas

5.3. *Princípios de classificação*

Ao classificar um conjunto de coisas, estamos agrupando essas coisas de acordo com algum critério pré-estabelecido. Os grupos obtidos com a classificação, classes, compartilham ao menos uma característica que os membros de outra classe não compartilham. Assim, o resultado de uma classificação é uma rede ou estrutura de relacionamentos.

De acordo com Prieto-Díaz [PRIETO-DÍAZ 87], existem dois tipos de relacionamentos que uma classificação deve expressar: hierárquico e sintático. Relacionamentos **hierárquicos** baseiam-se no princípio de subordinação ou inclusão, enquanto que relações **sintáticas** são criadas para relacionar dois ou mais elementos de diferentes hierarquias. Ainda segundo ele, esquemas de classificação típicos são estritamente hierárquicos. Relações sintáticas são posteriormente apresentadas através de classes compostas. Por exemplo, a "respiração dos pássaros" relaciona o termo "respiração", da classe de processos, com o termo "pássaros", da classe "aves"[31].

Classificações podem ser organizadas de duas formas: através de enumeração ou através de facetas. Classificações enumerativas (ou decimais) prescrevem um universo de conhecimento subdividido em classes sucessivamente menores que incluem todas as possíveis classes compostas (relações sintáticas). Essas classes são então organizadas de forma a apresentar suas relações hierárquicas.

Classificações facetadas sintetizam a declaração de conteúdo dos elementos a serem classificados que são analisadas segundo suas classes elementares. Essas classes são, então, relacionadas no esquema. Seus relacionamentos genéricos são os únicos apresentados. Quando é necessário apresentar uma classe composta utiliza-se uma combinação de suas classes elementares. Esse processo é denominado síntese.

[31] Poderíamos aqui relacionar "pássaros" a diversas classes da classificação animal bastando, para isso, selecionar diferentes níveis da hierarquia da classificação (por exemplo, vertebrados).

Textos Genéricos	Orientados a estudantes	
	Orientados à prática	
	Coletânea de textos para leitura	
Tópicos específicos na Engenharia de Requisitos	**Material genérico e introdutório**	Contexto para a Engenharia de Requisitos – Engenharia de Software
		Contexto para a Engenharia de Requisitos – Engenharia de Sistemas
		Abordagens à Engenharia de Requisitos
	Base (leituras selecionadas em disciplinas correlatas)	Lingüística
		Lógica e Métodos Formais
		Usabilidade e IHC (Interação Homem-computador)
		Trabalho colaborativo apoiado por computador
		Arquitetura de *Software*
		Ciências Sociais
		Filosofia
	Elicitando Requisitos	Elicitação – geral
		Soft Systems
		Abordagens Etnográficas
		Abordagens de Aquisição de Conhecimento
		Prototipação
		Elicitação orientada à objetivos
		Cenários e Casos de uso
	Modelando e Analisando Requisitos	Modelando organizações e empresas
		Modelagem de domínio
		Análise e Modelagem Orientada a Objetos
		Análise e Modelagem Estruturadas
		Análise e Modelagem Formais
		Modelando requisitos não funcionais
		Análise de Segurança (safety analysis)
		Análise de Segurança do Sistema (security analysis)
		Análise de Confiabilidade
		Análise de Usabilidade
		Animando/Executando Modelos de Requisitos
		Análise Baseada em Conhecimento
		Reuso de Modelos de Requisitos
	Comunicando Requisitos	Especificações em linguagem natural
		Especificações em linguagens baseadas em lógica
		Padrões e Modelos para Especificações
	Negociação de Requisitos (*agreeing requirements*)	Validação de Requisitos
		Negociação e Resolução de Conflitos
		Comparando e Priorizando Requisitos
	Evoluindo Requisitos	Gerenciando mudanças
		Gerenciando Inconsistências
	Processos Integrados de Requisitos	Modelos de Processos de Requisitos
		Engenharia de Métodos e Meta-modelagem
		Processos de Engenharia de Requisitos baseados em perspectivas
		Ambientes e Ferramentas de Engenharia de Requisitos

Tabela 5 – A bibliografia comentada de Easterbrook e Nuseibeh

Os grupos de classes elementares que compõem o esquema de classificação são as facetas. Os elementos ou classes que constituem uma faceta são denominados termos.Na Tabela 6 vemos um exemplo de uma classificação facetada.

Prieto-Díaz [PRIETO-DÍAZ 87] explica que ambos os esquemas podem representar o mesmo número de classes. Contudo, no esquema hierárquico, classes com mais de um componente elementar são imediatamente incluídos na classificação, enquanto que para o esquema facetado tem-se que sintetizar elementos de múltiplas classes. Ordenamento sistemático em uma classificação facetada consiste em dispor as facetas em ordem de citação, de acordo com sua relevância aos usuários da classificação. Termos nas facetas são ordenados por seu inter-relacionamento (proximidade conceitual).

(Faceta de processo)
- Fisiologia
  - Respiração
  - Reprodução

(Faceta do habitat animal)
- Animais aquáticos
- Animais terrestres

(Faceta taxonômica da zoologia)
- Invertebrados
  - Insetos
- Vertebrados
  - Répteis

Tabela 6 - Um exemplo de classificação facetada

Uma versão hierárquica da mesma classificação seria conforme Tabela 7:

Fisiologia
- Respiração
- Reprodução

Animais aquáticos
- Fisiologia de animais aquáticos
  - Respiração de animais aquáticos
  - Reprodução de animais aquáticos

Animais terrestres
- Fisiologia de animais terrestres
  - Respiração de animais terrestres
  - Reprodução de animais terrestres

Invertebrados
- Fisiologia de invertebrados
- *etc.*

Tabela 7 - Um exemplo de classificação hierárquica

Existem diferentes critérios para o ordenamento dos termos, porém, por convenção, tal ordenamento consiste de uma relação na qual a proximidade conceitual entre dois termos quaisquer da relação é indicada pelo número de termos que existem entre eles. Assim, para classificar um elemento dentro de um esquema facetado, o termo mais importante da descrição da classificação é o termo selecionado da faceta mais importante para o usuário. Vejamos um exemplo.

A partir do esquema da Tabela 8, vamos classificar o título "Necessidades nutricionais dos Ursos-pardos".

(Faceta de processo)
- Fisiologia
  - Respiração
  - Reprodução
  - Nutrição

(Faceta do habitat animal)
- Animais aquáticos
- Animais terrestres

(Faceta taxonômica da zoologia)
- Invertebrados
  - Insetos
- Vertebrados
  - Répteis
  - Mamíferos
    - Ursos
      - Ursos-pardos

Tabela 8 - Um esquema de classificação facetado

A ordem de citação do esquema dado é: *processo* → *habitat* → *taxonomia zoológica.* Primeiramente devemos verificar se há algum termo no título listado na faceta mais relevante: *processo*. A palavra *nutricionais* encontra-se referenciada como *nutrição* no esquema de classificação. Uma busca nas demais facetas apontará *Ursos-pardos* na faceta *taxonomia zoológica*. A classificação será, portanto, *nutrição/ursos-pardos*. Se, por outro lado, em nosso esquema de classificação a faceta *taxonomia zoológica* aparecesse antes que a faceta de *processo*, nossa classificação seria *ursos-pardos/nutrição*. Essa ordem reflete o grau de importância escolhido pelo usuário. Em um esquema enumerativo, a menos que uma categoria "nutrição de ursos" estivesse disponível, teríamos que decidir qual o termo mais importante: *nutrição* ou *ursos* para podermos classificar o item.

Esquemas facetados são mais fáceis de expandir. Para incluirmos *nutrição de ursos-pardos*, tudo que tivemos que fazer foi acrescentar ao esquema da Tabela 6 o termo *nutrição* (na faceta de processos) e os termos *ursos* e *ursos-pardos* (na faceta de taxonomia zoológica). Em um esquema hierárquico, o item entraria como uma classe composta na classe *vertebrados*, na classe *vertebrados terrestres*, na classe *animais terrestres* com referências cruzadas às outras duas.

Ainda segundo Prieto-Díaz [PRIETO-DÍAZ 87], esquemas facetados são mais flexíveis, mais precisos e mais adequados à classificação de coleções de muitos elementos e com crescimento permanente. Em biblioteconomia, esquemas facetados são construídos a partir da escolha de uma amostra da coleção a ser classificada, um processo denominado *literary-warrant*. Termos são escolhidos dos textos selecionados, agrupados e, então, as facetas são definidas a partir dos grupos. As facetas são então classificadas em ordem de citação e os termos em cada faceta ordenados arbitrariamente, de acordo com as necessidades dos usuários.

Por exemplo, os termos na faceta "planetas do sistema solar" (nome dos planetas) podem ser ordenados por seu tamanho, distância do Sol ou em ordem alfabética, dependendo do que for mais relevante ao usuário. Um esquema construído a partir de *literary-warrant* gera esquemas específicos e personalizados. Contudo, novos termos podem ser facilmente adicionados ao esquema de classificação.

5.4. *As dimensões utilizadas na taxonomia deste trabalho*

A taxonomia deste trabalho será apresentada na forma de um esquema de classificação facetada, no qual as facetas serão definidas em função de dimensões específicas escolhidas com base na argumentação apresentada e na bibliografia coletada durante o trabalho. A facilidade de extensão do esquema permitirá que a taxonomia seja aproveitada e adaptada a interesses específicos.

Para classificar a pesquisa na área de Engenharia de Requisitos é necessário definir claramente o que está sendo considerado como pesquisa. A princípio, as idéias obtidas com a investigação científica em algum campo do conhecimento são a verdadeira produção da pesquisa. Essas idéias são então desenvolvidas e, algumas delas acabam por encontrar um caminho que possibilite sua aplicação[32] – prática ou teórica – que justifique sua ampla divulgação e discussão no meio acadêmico e, posteriormente (se for o caso), no meio industrial.

Em geral, idéias obtidas com uma linha de investigação científica são associadas a modelos e/ou idéias anteriores na produção de um artigo que, submetido a algum veículo de divulgação técnica, junta-se a outros artigos e transforma-se em uma publicação. Essas publicações revestem as idéias oriundas da pesquisa de um caráter mais sólido, uma vez que o crivo editorial aumenta-lhe a confiabilidade.

Assim, para classificar a pesquisa na área de Engenharia de Requisitos, consideraremos que ela está identificada na produção literária resultante da publicação de artigos nos diversos veículos reconhecidos, nacional e internacionalmente, pela comunidade acadêmica e industrial.

---

[32] Devemos observar que não está sendo considerado o tempo necessário para a maturação da idéia, que empiricamente, tem-se verificado estar em torno de 20 (vinte) anos.

Tipo de publicação

Considerando-se que um dos objetivos do presente trabalho é orientar iniciantes na área a entender suas subáreas e como se relacionam, é necessário que se estabeleça, inicialmente, uma faceta da taxonomia destinada a classificar os tipos de artigos que podemos encontrar quando a literatura da área é analisada.

Assim, a partir da leitura das publicações encontramos diversos tipos de arquivos. Para compor a faceta destinada a classificar os tipos de artigos, proponho 6 (seis) classes distintas:

- **Conceituação** - estabelecem e formalizam os conceitos a serem usados através da pesquisa, por exemplo, definindo o que são requisitos, um modelo para o processo de requisitos ou ainda quem são os participantes desse processo;

- **Método-procedimento** - propõem uma técnica (método ou processo[33]) para a condução de determinada atividade no contexto da Engenharia de Requisitos;

- **Estudo de caso** - relatam os resultados do estudo da aplicação de um método, conceito ou técnica em um caso particular, controlado e de especial interesse para a pesquisa;

- **Investigação-solução** - divulgam resultados intermediários de uma investigação científica cuja publicação pode atrair mais pesquisadores para a mesma linha de pesquisa;

- **Compêndio** - revisam, destacam, classificam, agrupam ou resumem um conjunto de técnicas aplicáveis a determinado problema no contexto da área de pesquisa. Envolvem a comparação das técnicas delimitando seu campo de aplicação;

- **Posicionamento** - revisam a produção científica passada e/ou fazem projeções em relação aos caminhos da pesquisa futura. São muito úteis na escolha de um caminho para o início de uma pesquisa e para identificar os temas mais bem cotados pela comunidade acadêmica ou industrial.

Problema original

Ao definir as dimensões da Engenharia de Requisitos, Pohl [POHL 94] faz distinção entre os problemas originais da área e os problemas originados pelas abordagens adotadas. Os problemas originais constituem um conjunto simples e eficiente para analisar a questão de requisitos de forma independente dos demais fatores que influenciam a

---

[33] Método é uma técnica para a realização de uma tarefa. De maneira geral, envolve a condução de passos seqüenciais. Processo é um conjunto de atividades que devem ser desenvolvidas para atingir um objetivo. A principal distinção entre ambos é a ordem de grandeza. Um processo, em geral, envolve um conjunto de atividades para as quais podem existir métodos aplicáveis.

condução da atividade, tais como o domínio de aplicação ou o modelo de processo utilizado. Conhecer o problema original do qual um artigo trata antes de lê-lo ajudaria na seleção de um título desconhecido.

Assim, o problema original de que trata o artigo é a segunda faceta da taxonomia proposta e suas classes serão como se segue:

- **Especificação** - lida com o grau de entendimento dos requisitos em um ponto no tempo;
- **Representação** - trata da forma utilizada para expressar o conhecimento adquirido sobre o sistema;
- **Consenso** - lida com o grau de concordância que se atingiu a respeito de uma especificação.

Atividade do ciclo de vida

Um dos objetivos deste trabalho é fazer com que a taxonomia seja organizada de modo a refletir naturalmente a condução do trabalho de requisitos, tal como executado na prática. Dessa forma, uma faceta da classificação proposta será a **atividade do ciclo de vida** na qual se enquadra o texto classificado.

A atividade do ciclo de vida dá ao leitor da taxonomia uma perspectiva sobre o problema de que trata o item classificado, no sentido das dificuldades a serem vencidas. Assim, alguém conduzindo trabalhos com requisitos poderia, facilmente, encontrar a atividade que vem desenvolvendo no modelo de processo proposto, identificar seu problema específico e selecionar um artigo que trate do assunto, facilitando o acesso de profissionais da área à literatura acadêmica.

A terceira faceta da classificação proposta será, então, a atividade do ciclo de vida, definido conforme seção 4.5. Em seu contexto teremos, portanto, as seguintes classes:

- **Elicitação** - consiste na identificação de requisitos a partir da consulta aos *stakeholders*, da análise de documentos, da análise de informações do domínio e/ou de estudos de mercado;
- **análise e negociação** - consiste na análise detalhada dos requisitos com a negociação entre os diferentes *stakeholders* visando decidir quais os requisitos que serão aceitos;
- **documentação** - os requisitos aprovados na atividade de análise e negociação devem ser registrados em um documento apropriado, em nível de detalhamento adequado;
- **validação** - depois de documentados, os requisitos devem sofrer cuidadosa verificação de consistência e completitude;
- **gerenciamento** - é constituída das atividades relacionadas à manutenção de consistência e ao controle de alterações dos requisitos.

Domínio de aplicação

Um grande divisor de águas das técnicas aplicáveis na condução da Engenharia de Requisitos é o domínio de aplicação. Kotonya e Sommerville [KOTONYA 98] classificam os sistemas desenvolvidos em três tipos distintos:

- **Sistemas de informação** - lidam com a manutenção de grande quantidade de informação em uma base de dados, como é o caso dos Sistemas Integrados de Gestão Empresarial (ERP). Os principais problemas do processo de requisitos para sistemas dessa área estão na captura de requisitos e na obtenção de consenso sobre eles, em um ambiente com um grande número de *stakeholders*;

- **Sistemas embutidos** - lidam com o acionamento de dispositivos e o controle de *hardware* específico em condições de restrições extremas de tempo e, por vezes, de segurança. É o caso de sistemas que acionam telefones celulares. Os principais problemas do processo de requisitos para sistemas deste tipo estão relacionados a sua especificação;

- **Sistemas de comando e controle** - são uma combinação das duas categorias anteriores utilizando uma base de dados para o comando e/ou controle de dispositivos e para a tomada de decisões. É caso, por exemplo, de sistemas inteligentes que controlam subestações de energia elétrica. Novamente, neste caso, a maior dificuldade do processo é a especificação.

O domínio de aplicação é uma excelente perspectiva sobre a pesquisa. Técnicas utilizadas para sistemas de informação tendem a ser menos formais (no sentido matemático), utilizando abordagens que exigem menor esforço matemático. Isso faz sentido uma vez que os principais problemas do processo são sociais. Já no caso das duas outras categorias de sistemas, resolver o principal problema do processo de requisitos envolve adotar uma abordagem com maior formalismo matemático.

Neste ponto há grande divergência na comunidade acadêmica. Há pesquisadores que acreditam não haver outro caminho possível para a solução dos problemas da Engenharia de Requisitos que não o do formalismo matemático [PARNAS 00]. Há, por outro lado, aqueles que defendem a utilização de técnicas menos formais [GLINZ 00a].

Em verdade, tais discussões não levam em conta um fator determinante na condução de qualquer processo técnico na prática: o custo de seu desenvolvimento. A utilização de técnicas formais aumenta sobremaneira a precisão da especificação, diminuindo-lhe a ambigüidade e facilitando a detecção automática de conflitos. Por outro lado requer maior qualificação da equipe de desenvolvimento (tanto dos engenheiros de requisitos – que irão construir a especificação – quanto dos engenheiros de *software* – que deverão lê-la), dificulta a participação dos *stakeholders* no processo (não entenderão a

documentação produzida e precisarão de um tradutor), faz com que o processo seja mais lento e, portanto, mais caro (pessoas mais qualificadas sendo alocadas ao projeto por mais tempo).

Assim, tais relações de custo devem ser verificadas antes de se escolher a técnica mais adequada para utilização em um processo de requisitos. Dessa forma, a quarta faceta de nossa classificação será o **domínio de aplicação**. Dentro dela incluiremos duas classes: **sistemas de informação**, que atendem a classificação de Kotonya & Sommerville [KOTONYA 98], e **outros sistemas**, englobando todos os sistemas que não se qualifiquem como sistemas de informação (sistemas embutidos e de comando e controle segundo a classificação de Kotonya & Sommerville [KOTONYA 98]).

A Tabela 9 descreve o esquema de classificação facetado a ser utilizado na taxonomia proposta.

(Tipo de publicação)
- Conceituação
- Método-procedimento
- Estudo de caso
- Investigação-solução
- Compêndio
- Posicionamento

(Problema original)
- Especificação
- Representação
- Consenso

(Atividade do ciclo de vida)
- Elicitação
- Análise e negociação
- Documentação
- Validação
- Gerenciamento

(Domínio de Aplicação)
- Sistemas de Informação
- Outros Sistemas

Tabela 9 – O Esquema de classificação facetada proposto

5.5. *A utilização da Taxonomia na classificação de alguns trabalhos*

Para aplicar o esquema de classificação proposto aos itens bibliográficos abaixo foram adotados alguns critérios. Idealmente seria conveniente que as classes de cada faceta fossem mutuamente exclusivas, isto é, a presença de determinado artigo em uma classe excluiria sua possível classificação como elemento de outra classe da mesma faceta.

No entanto, o que ocorre é que a distribuição de cada item classificado através das classes de uma mesma faceta é função subjetiva do julgamento do classificador. Assim, existe a possibilidade de que o classificador julgue ser diferente o aspecto mais importante de um

determinado texto e classifique-o em outra classe da mesma faceta. Nos casos abaixo, o item foi classificado na classe que caracterizou seu aspecto mais importante, segundo a minha própria avaliação.

A classificação dos itens abaixo é apresentada em uma tabela na qual constam:

- A referência bibliográfica completa do item classificado;
- Um breve resumo do texto;
- A classificação segundo o esquema proposto.

Como forma de caracterizar o aspecto mais importante do texto classificado, o resumo apresentado foi construído visando destacar as características do artigo que levaram seu enquadramento nas classes apresentadas logo a seguir.

Caso alguma faceta não esteja presente na classificação, isto significa que apresentá-la não agregaria informação adicional ao item bibliográfico (por exemplo, o item tem igual importância em quaisquer classes da faceta).

A princípio consideramos no item 5.4 que a produção da pesquisa seria identificada por artigos publicados em veículos de renome industrial ou acadêmico. No entanto, algumas exceções foram necessárias. Alguns temas mais maduros na pesquisa foram assunto de livros. Essas obras, apesar de não representarem a linha de frente da pesquisa, encontraram seu lugar na classificação, pois representam temas fundamentais para o entendimento e compreensão da área em estudo (Engenharia de Requisitos).

A Tabela 10 apresenta, a título de exemplo da aplicação da taxonomia, algumas referências (uma classificação mais completa de grande parte dos itens da bibliografia deste texto é apresentada no Apêndice 1), classificadas em núcleo estrutural como na Figura 6. Cada uma das células contém informações como indicado.

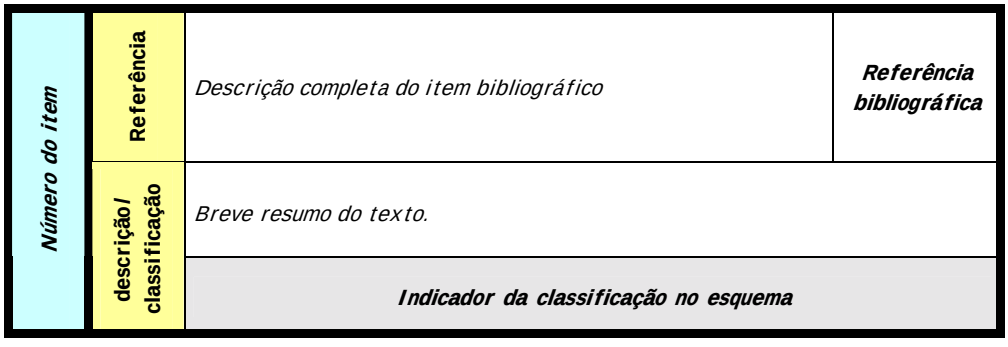

Figura 6 - O Esquema para representação da Taxonomia

1	referência	Structured Design E. N. Yourdon & L. L. Constantine Yourdon Press – 1978	**[YOURDON 78]**
	Descrição / classificação	Método para a elaboração de projeto de *software* englobando uma seqüência de passos para decomposição de um sistema em funções e a representação dos elementos dessa decomposição em uma notação específica.	
		Método-procedimento / representação / documentação / sistemas de informação	
2	referência	Requirements Engineering in the Year 00: A Research Perspective Axel van Lamsweerde Proc of the 22nd International Conference on Software Engineering June 4 - 11, 2000, Limerick Ireland	**[LAMSWEERDE 00]**
	Descrição / classificação	Uma análise da pesquisa na área de Engenharia de Requisitos até o ano de 2000, focalizando a evolução das técnicas e abordagens e os problemas ainda em aberto e que deveriam ser foco da pesquisa nos próximos anos.	
		Posicionamento	
3	referência	Four Dark Corners of Requirements Engineering Pamela Zave & Michael Jackson ACM Transactions on Software Engineering and Methodology, 6(1): 1-30	**[ZAVE 97b]**
	Descrição / classificação	Texto apontando as principais deficiências conceituais que causam ambigüidade no entendimento das necessidades da Engenharia de Requisitos. Cria um modelo conceitual para o entendimento dos requisitos separando a parte interna da máquina (*hardware* + *software*) do mundo externo no qual ela irá operar.	
		Conceituação / especificação	
4	referência	Techniques for Requirements Elicitation Joseph A. Goguen e Charlotte Linde Proceedings of The International Symposium on Requirements Engineering, 1993	**[GOGUEN 93]**
	Descrição / classificação	Análise das técnicas aplicáveis à elicitação de requisitos, enfatizando, particularmente, a forma como cada uma delas lida com as questões sociais da atividade.	
		Compêndio / consenso / elicitação	

5	**referência**	Attempto Controlled English (ACE) Norbert E. Fuchs & Rolf Schwitter CLAW 96, The First International Workshop on Controlled Language Applications Katholieke Universiteit Leuven – 26-27/March/1996	**[FUCHS 96]**
	Descrição / classificação	Proposta de redução do vocabulário da língua inglesa visando utilizá-la como linguagem formal na especificação de sistemas.	
		Investigação-solução / especificação / validação	
6	**referência**	Factors Influencing Requirements Traceability Practice Balasubramaniam Ramesh Comunications of the ACM 41(12), Dezembro/1998	**[RAMESH 98]**
	Descrição / classificação	Estudo de caso sobre a aplicação da rastreabilidade em um conjunto de empresas de desenvolvimento de *software*. O autor conduz uma análise sobre o efeito dos fatores ambientais, organizacionais e técnicos que influenciam a condução da rastreabilidade no processo de *software*.	
		Estudo de caso / gerenciamento	

Tabela 10 - A taxonomia proposta

6. Problemas em aberto na área de Engenharia de Requisitos

Easterbrook & Nusseibeh [NUSEIBEH 00] afirmam que os anos 1990 foram tempos de grandes avanços na área de Engenharia de Requisitos. Durante esse período, Easterbrook & Nusseibeh identificam três novas idéias que desafiaram e reviraram as visões ortodoxas da Engenharia de Requisitos:

- ***Contexto social*** - a idéia de que modelagem e análise não podem ser conduzidas adequadamente de forma isolada do contexto social no qual qualquer novo sistema deverá operar. Essa percepção enfatizou o uso de técnicas contextualizadas, tais como etnometodologia e observação de participantes;

- ***Modelar o ambiente*** - a noção de que Engenharia de Requisitos não deveria concentrar-se na especificação da funcionalidade de um novo sistema, mas, ao invés disso, deveria concentrar-se na modelagem de propriedades indicativas e optativas do ambiente. Somente através da descrição do ambiente e do que o sistema precisa fazer naquele ambiente, podemos capturar o propósito do sistema e analisar se determinado projeto irá atender aquele propósito;

- ***Viver com inconsistências*** - a idéia que a tentativa de construir modelos de requisitos completos e consistentes é fútil e que Engenharia de Requisitos deve analisar e resolver requisitos conflitantes, apoiar a negociação de *stakeholders* e raciocinar sobre modelos que contêm inconsistências.

Para os próximos anos, Easterbrook & Nusseibeh [NUSEIBEH 00] apontam seis desafios para a Engenharia de Requisitos:

1. ***Modelar o ambiente*** - desenvolver novas técnicas para modelar e analisar as propriedades do ambiente, ao invés do comportamento do *software*. Tais técnicas devem levar em conta a necessidade de tratar modelos inconsistentes, incompletos e em permanente evolução. Espera-se que, com isso, que a Engenharia de Requisitos possa evoluir em áreas ainda obscuras como, por exemplo, a especificação das necessidades de recursos que um componente exige de seu ambiente, o que poderia facilitar a adaptação de componentes em outros ambientes de *software* ou *hardware* e a adaptação de produtos em família de produtos;

2. ***Transição entre elicitação e análise*** - preencher a lacuna existente entre as técnicas de elicitação contextuais e técnicas mais formais de especificação e análise. Abordagens contextuais, como etnografia, oferecem a possibilidade de um rico entendimento do contexto organizacional de um novo sistema de *software*, porém, não possuem correspondência direta com as técnicas de modelagem formal das propriedades, atuais e desejadas, do domínio do problema. Segundo Easterbrook & Nusseibeh isso deve incluir a incorporação de uma grande variedade de mídias, tais como vídeo e áudio, nas técnicas de modelagem de comportamento;

3. ***Requisitos não funcionais*** - criar modelos mais ricos para capturar e analisar requisitos não funcionais;

4. ***Arquitetura*** - melhor entender o impacto das escolhas de arquitetura na priorização e evolução dos requisitos. Os esforços na área de arquitetura de *software* concentraram-se em sua representação e na análise de seu comportamento. É necessário estudar como seria possível analisar o impacto da escolha de uma arquitetura particular na habilidade de satisfazer os requisitos atuais e futuros, ou ainda variações nos requisitos através de famílias de produtos;

5. ***Reuso de modelos de requisitos*** - propor modelos de referência para requisitos através de diversos domínios de aplicação, de forma a reduzir o esforço para a produção de modelos de requisitos. Isso permitiria que muitos projetos fossem simplificados e também facilitaria a seleção de COTS (*commercial off-the-shelf software*);

6. ***Treinamento*** - treinamento multidisciplinar para profissionais de requisitos. De forma geral tratamos do profissional que aplica técnicas de requisitos para elicitar, analisar e especificar requisitos como Engenheiro de Requisitos. Embora tais atividades sejam de importância crucial no contexto de desenvolvimento de *software*, a maior parte das empresas não possui profissional com qualificações para realizar essas tarefas. O Engenheiro de Requisitos deverá ter habilidades para interagir com uma diversidade de *stakeholders*, incluindo pessoas de perfil não técnico, bem como habilidades técnicas que o permitam interagir com analistas e programadores.

Lamsweerde [LAMSWEERDE 00] aponta ainda algumas áreas onde a pesquisa deve evoluir:

a. Aproximar a pesquisa de requisitos com a pesquisa de arquitetura de *software*;

b. Melhor entender a sistemática derivação de parâmetros a partir dos requisitos. Isso facilitaria a configuração e implantação de pacotes complexos personalizados para utilização específica, como o caso dos Sistemas de ERP;

c. Desenvolver abordagens que apóiem Engenharia de Requisitos em pequena escala, envolvendo usuários finais como os únicos *stakeholders*. Isso facilitaria a personalização do *software* às necessidades específicas do número crescente de usuários de *software* surgido com o avanço da internet;

d. Preencher a lacuna entre a pesquisa na Engenharia de Requisitos e a especificação formal. Enquanto Engenharia de Requisitos oferece abstrações mais ricas, especificação formal oferece meios para uma análise mais detalhada e precisa;

e. Melhorar a expressividade dos modelos de domínio e de requisitos, possibilitando-lhes capturar mais conhecimento sobre os múltiplos aspectos, responsabilidades e atividades envolvidas no processo de engenharia de requisitos (melhor balanço entre expressividade e precisão dos modelos);

f. Modelar os agentes em uma área particular de interesse. Técnicas tradicionais de Engenharia de Requisitos dividem o mundo em dois componentes: o *software* e seu ambiente. Na maioria dos casos, contudo, múltiplos componentes de *software*, humanos e físicos tem que cooperar. É preciso muito trabalho para apoiar a análise desses ambientes complexos durante a atividade de Engenharia de Requisitos, em particular a atribuição de responsabilidades;

g. Modelos que permitam analisar a situação atual e as possíveis alterações nos requisitos. Apesar de ser um tópico fundamental na Engenharia de Requisitos, pouca atenção foi dedicada a este assunto nos últimos anos;

h. A construção de ferramentas para apoiar o processo de requisitos e a manutenção de consistência nos complexos e extensos documentos de requisitos produzidos, na maioria dos casos, em linguagem natural. Tal ferramenta deveria auxiliar na geração do documento, na manutenção da estrutura do modelo de requisitos, possibilitar a extração de suas partes e manter elos de rastreabilidade ao material de elicitação e, futuramente, ao material de desenvolvimento.

Da análise das áreas para futura pesquisa na Engenharia de Requisitos, podemos distinguir três principais áreas de concentração:

- **Modelagem** - modelar requisitos é ainda tarefa difícil. Além de não existir consenso sobre as técnicas aplicáveis, tal como descrito na seção 4.4.2, a maior parte das técnicas utilizadas têm origem no projeto de software. Dessa forma estão mais voltadas aos problemas e estruturas internas do sistema (da máquina) do que às características do ambiente propriamente dito. É preciso formalizar adequadamente os fenômenos que devem ser observados no ambiente para que seja possível criar um mecanismo de modelá-los. Dessa forma, poderemos melhor representar as exigências de um componente em relação a seu ambiente, requisitos não funcionais, a transição entre elicitação e análise, entre outros;

- **Aproximar o estudo de Arquitetura do estudo de Engenharia de Requisitos** - essa aproximação é necessária para que possamos construir *software* com maior manutenibilidade, melhor reutilizar requisitos através de família de produtos e aumentar a reutilização de requisitos através de diferentes projetos;

- **Investimento no desenvolvimento de recursos de apoio ao processo** - melhorar os recursos utilizados no processo de requisitos é fundamental para melhorarmos a condução da atividade na prática. Melhor especialização dos profissionais envolvidos com requisitos, a construção de ferramentas de apoio ao enorme volume

de informações que devem ser administradas entre outras, são iniciativas fundamentais para a próxima década da pesquisa.

A Tabela 11 resume em qual das categorias se enquadra cada um dos pontos levantados por Easterbrook & Nusseibeh e Lamsweerde.

Categoria	Easterbrook & Nusseibeh	Lamsweerde
Modelagem	1, 2, 3	c, d, e, f, g
Aproximação de ER e Arquitetura	4, 5	a, b
Investimento em recursos de apoio ao processo	6	h

Tabela 11 – Um resumo dos principais tópicos em aberto na ER

7. Conclusões e Trabalhos Futuros

Neste trabalho, na seção 1 introduzimos o contexto que faz da Engenharia de Requisitos uma área de extrema relevância ao desenvolvimento de *software* de forma geral. Na seção 4 apresentamos os principais conceitos relacionados à disciplina, o processo utilizado em sua condução e discutimos extensamente os problemas relacionados a cada uma de suas atividades. Na seção 5 analisamos os elementos que poderiam ser utilizados em uma classificação da pesquisa, apresentamos algumas classificações encontradas na literatura, discorremos sobre questões teóricas relacionadas a classificações em geral e, por fim, na seção 5.4, discutimos os elementos que iriam compor as facetas da taxonomia proposta. Como forma de validar o esquema e entender sua aplicabilidade, utilizamo-lo na classificação das referências pertinentes deste trabalho, tendo produzido a classificação da Tabela 10.

A classificação apresentada possibilita uma visão geral das principais áreas da Engenharia de Requisitos já no esquema de classificação, em sua faceta *Atividade do ciclo de vida*. Através de sua simples leitura é possível obter uma macro-visão de quais são as atividades conduzidas no contexto da Engenharia de Requisitos. Além disso, ela mesma dá à taxonomia um caráter ontológico, uma vez que permite entender as atividades da forma como são conduzidas na prática.

A visão de Pohl [POHL 94] caracteriza de forma bastante útil os problemas que enfrentamos na Engenharia de Requisitos. Conhecer qual dos problemas identificados por Pohl é referenciado por um trabalho específico ou por um conjunto de trabalhos, proporciona uma visão mais abstrata sobre os problemas tratados no contexto da disciplina, permitindo a um pesquisador ou estudante da área entender as implicações de um trabalho em relação a outros através de uma perspectiva de mais alto nível.

Quando um estudante inicia-se no estudo de um novo campo, uma das maiores dificuldades que ele enfrenta é entender os elementos básicos que o compõe, de forma a racionalizar o tempo de seus estudos. A taxonomia proposta possibilita que um estudante que deseje aprofundar-se na área de Engenharia de Requisitos possa localizar mais facilmente artigos que se relacionem ao tema específico que esteja estudando. Além disso, o resumo apresentado na Tabela 10 orienta a escolha final de um item específico em uma bibliografia classificada, possibilitando uma melhor análise de interesse sobre o trabalho.

Assim, consideramos que os objetivos apresentados na seção 2 tenham sido inteiramente cumpridos.

Uma possível utilização da taxonomia proposta seria sua adoção em dois níveis: pelos veículos de publicação e pelos mecanismos de busca em bases de conhecimento. Ao publicar um artigo, o veículo de publicação poderia incluir uma referência mais completa, como a célula mostrada na Figura 6. As informações poderiam ser preenchidas pelo próprio autor, que apontaria mais claramente as nuanças mais importantes de seu trabalho. Por outro lado, os mecanismos de busca em bases de conhecimento poderiam permitir

que os elementos da taxonomia fossem utilizados como parâmetros de busca. Dessa forma, alguém pesquisando na base a partir das facetas e classes da taxonomia, poderia achar referências mais adequadas mais rapidamente. Além disso, o esquema facetado utilizado na classificação é extensível, possibilitando a acomodação de outras perspectivas ou classes em suas facetas, o que permitiria que o grande trabalho a ser desenvolvido inicialmente (para classificar os trabalhos já existentes) fosse aproveitado através do tempo.

Essa utilização assume que, tanto o autor do texto quanto o pesquisador que tem interesse em consultar uma base de conhecimento, conheçam tanto a taxonomia quanto os critérios utilizados em sua confecção. Sem esse conhecimento talvez não seja possível obter o benefício descrito (é preciso saber a que se refere, por exemplo, a classe *análise e negociação* da faceta *Atividade do ciclo de vida*).

Outra possibilidade de uso, embora de mesma natureza, é menos ambiciosa e, portanto, provavelmente mais factível. A classificação poderia ser utilizada através dos alunos de um departamento da universidade, destinado a estudar Engenharia de Requisitos. A cada trabalho lido por um dos alunos, uma entrada na classificação seria produzida. Dessa forma, uma base de conhecimento estaria sendo produzida e iria beneficiar os alunos que viessem a pesquisar sobre o tema no futuro. No atual contexto da educação brasileira, onde a avaliação dos cursos em nível de pós-graduação leva em conta o tempo para a conclusão do trabalho, uma ajuda dessa natureza seria muito útil, ainda que não fosse automatizada.

8. Bibliografia[34]

[BENNER 93] BENNEER, K. M. et al. *Utilizing Scenarios in the Software Development Process*. Information Systems Development Process, North-Holland: Elsevier Science Publisher, 1993, p. 117-134.

[BOEHM 88] BOHEM, B. *An Spiral Model for Software Development and Enhancement*, Computer, v. 21, n. 5, p. 61-72, mai. 1988.

[BOEHM 95] BOEHM, B. et al. *Software Requirements Negotiation and Renegotiation Aids: A Theory-W Based Spiral Approach*. In: International Conference on Software Engineering (ICSE-17), 17., 1995, Seattle, USA. Proceedings... IEEE Computer Society Press, p. 243-253.

[BOOCH 94] BOOCH, G. *Object Oriented Analysis and Design with Applications*. Addison Wesley, 1994.

[BROOKS 75] BROOKS, F. P. *The Mythical Man-Month*. Addison Wesley, 1975.

[BROOKS 87] BROOKS, F. *No Silver Bullet – essence and accidents of software engineering*. Computer, v. 20, n. 4, p. 1069-1076, abr. 1987.

[BUBENKO 80] BUBENKO, J. *Information Modeling in the Context of System Development*. In: IFIP Congress 1980, North Holland. Proceedings... p. 395-411.

[CHEN 76] CHEN, P. *The Entity Relationship Model – Towards a Unified View of Data*. ACM Transactions on Database Systems, v. 1, n. 1, p. 9-36, 1976.

[COLEMAN 94] COLEMAN, D. et al. *Object-Oriented Development: The Fusion Method*, Prentice Hall, 1994.

[CYSNEIROS 01] CYSNEIROS, L. M.; LEITE, J. C. S. P; NETO, J. M. *A Framework for Integrating Non-Functional Requirements into Conceptual Models*. Requirements Engineering Journal, v. 6, n. 2, p. 97-115, 2001.

[34] Organizada de acordo com as normas NBR-6023:2000 – Informação e Documentação – Referências – Elaboração e NBR 10520:1992 – Apresentação de citações em documentos – Procedimento.

[CYSNEIROS 99] CYSNEIROS, L. M.; LEITE, J. C. S. P. *Integrating Non-Functional Requirements into Data Model*. In: 4th International Symposium on Requirements Engineering (RE'99), 1999, Limerick, Ireland. Proceedings... IEEE Computer Society Press, p. 162-171.

[DARDENE 93] DARDENE, A.; LAMSWEERDE, A.; FICKAS, S. *Goal - directed Requirements Acquisition*. Science of Computer Programming, v. 20, p. 3-50, 1993.

[DAVIS 93] DAVIS, A. M. *Software Requirements - Objects, Functions and States*. PTR-Prentice Hall, 1993.

[DEMARCO 79] DEMARCO, T *Structured Analysis and System Specification*. Prentice Hall, 1979.

[EASTERBROOK 93a] EASTERBROOK, S. *Negotiation and the Role of the Requirements Specification*. In: Social Dimensions of Systems Engineering: People, processes, policies and software development. London: Ellis Horwood, 1993, p.144-164.

[EASTERBROOK 93b] EASTERBROOK, S. M. et al. *A Survey of Empirical Studies of Conflict.* In: CSCW:-Co-operation or Conflict? London: Springer-Verlag, 1993, p. 1-68.

[EASTERBROOK 94a] EASTERBROOK, S. M. et al. *Co-ordinating Conflicting ViewPoints by Managing Inconsistency*. In: Workshop on Conflict Management in Design - International Conference on Artificial Intelligence in Design, aug. 1994, Lausanne, Switzerland. Proceedings... Disponível em <http://www.cs.toronto.edu/~sme/papers/index.html>. Acesso em: 07 abr. 2002.

[EASTERBROOK 94b] EASTERBROOK, S. M. *Resolving Requirements Conflicts with Computer-Supported Negotiation*. In: Requirements Engineering Social and Technical Issues, Academic Press, 1994, p. 41-65.

[EASTERBROOK 02] EASTERBROOK, S. M.; NUSEIBEH, B. *Requirements Engineering Annotated Bibliography.* Disponível em **<**http://www.cs.toronto.edu/~sme/CSC2106S/readings.pdf**>**. Acessado em 07 abr. 2002.

[FEATHER 87] FEATHER, M. *Language Support for the Specification and Developmento of Composite Systems*. ACM Transactions on Programming Languages and Systems, v. 9, n. 2, p. 198-234, 1987.

[FUCHS 96] FUCHS, N. E.; SCHWITTER, R. *Attempto Controlled English (ACE)*. In: The First International Workshop on Controlled Language Applications (CLAW 96), 1996, Würzburg, Germany. Proceedings... p. 211-218.

[GLINZ 00a] GLINZ, M. *A Lightweight Approach to Consistency of Scenarios and Class Models*. In: 4th International Conference on Requirements Engineering (ICRE 2000), 2000, Schaumburg, IL, USA. Proceedings... IEEE Computer Society Press, p. 49-58.

[GLINZ 00b] GLINZ, M. *Problems and Deficiencies of UML as a Requirements Specification Language*. In: 10th International Workshop on Software Specification and Design, 2000, San Diego, CA, USA. Proceedings... IEEE Computer Society Press, p. 11-22.

[GOGUEN 93] GOGUEN, J. A.; LINDE, C. *Techniques for Requirements Elicitation*. In: International Symposium on Requirements Engineering, 1., 1993, San Diego, Ca. Proceedings... IEEE Computer Society Press, p. 152-164.

[GOGUEN 94] GOGUEN, J. A. *Requirements Engineering as the Reconciliation of Technical and Social Issues*. In: Requirements Engineering Social and Technical Issues, Academic Press, 1, 1994, pp.165-199.

[GREENSPAN 82] GREENSPAN, S. J.; MYLOPOULOS, J.; BORGIDA, A. *Capturing More World Knowledge in the Requirements Specification*. In: 6th International Conference on Software Engineering (ICSE-6), 1982, Tokyo, Japan. Proceedings... IEEE Computer Society Press, p. 225-235.

[IEEE 830-1998] IEEE-SA STANDARDS BOARDS *IEEE Std 830-1998: IEEE Recommended Practice For Software Requirements Specifications*, jun. 1998.

[IEEE 1233-1998] IEEE-SA STANDARDS BOARD *IEEE Std 1233-1998: IEEE Guide for Developing System Requirements Specifications*, dec. 1998.

[IEEE 1362-1998] IEEE STANDARDS BOARD *IEEE Std 1362–1998: IEEE Guide for Information Technology System Definition – Concept of Operations (ConOps) Document*, mar. 1998.

[JACKSON 78] JACKSON, M. *Information Systems: Modeling, Sequencing and Transformation*. In: 3rd International Conference on Software Engineering (ICSE-3), 1978, Munich. Proceedings... IEEE Computer Society Press, p. 72-81.

[JACOBSON 92] JACOBSON, I.; CHRISTERSON, M.; JONSSON, P.; ÖVERGAARD, G. *Object-Oriented Software Engineering - A Use Case Driven Approach*. Addison Wesley, 1992.

[JACKSON 95] JACKSON, M. *Software Requirements and Specifications*. Addison Wesley, 1995.

[JARKE 98a] JARKE, M. *Requirements Tracing*. Communications of the ACM, v. 41, n. 12, dec. 1998.

[JARKE 98b] JARKE, M.; KURKI-SUONIO *Special Issue on Scenario Management*. IEEE Transactions on Software Engineering, v. 24, n. 12, 1998.

[KARLSSON 97] KARLSSON, J.; RYAN, K. *A Cost-Value Approach for Prioritizing Requirements*. IEEE Software, v. 14, n. 5, p. 67-74, Oct. 1997.

[KOTONYA 96] KOTONYA, G.; SOMMERVILLE, I. *Requirements Engineering With Viewpoints*. Software Engineering Journal, v. 11, n 1, p. 5-11, jan. 1996.

[KOTONYA 98] KOTONYA, P.; SOMMERVILLE, I. Requirements Engineering: Processes and Techniques. Viley 1998.

[KRUCHTEN 98] KRUCHTEN, P. *The Rational Unified Process: An Introduction.* Addison Wesley, 1998.

[LAMSWEERDE 00] LAMSWEERDE, A. *Requirements Engineering in the Year 00: A Research Perspective*. In: International Conference on Software Engineering, 22., jun. 2000, Limerick, Ireland. Proceedings... ACM, 2000, p. 5-19.

[LEITE 91] LEITE, J. C. S. P.; FREEMAN, P. *Requirements Validation Through Viewpoint Resolution*. IEEE Transactions on Software Engineering, v. 17, n. 12, p. 1253-1269, dec. 1991.

[LEITE 93] LEITE J.C.S.P.; FRANCO, A.P.M. *A Strategy for Conceptual Model Acquisition*. In: International Symposium on Requirements Engineering, 1., 1993, SanDiego, Ca. Proceedings... IEEE Computer Society Press, p. 243-246.

[LOPES 02] LOPES, P. S. N. D.; BARRETO, M. R. P. *Requirements Modeling with UML Discussed*. A ser publicada como BT da Escola Politécnica da USP. Disponível em <http://www.ime.usp.br/~pnaddeo>. Acessado em 07 abr. 2002.

[LUDEWIG 96] LUDEWIG, J. *Software Engineering – Why it did not work*. In: Dagstuhl Seminar 9635 on History of Software Engineering, ago. 1996, p. 25-27. Disponível em: <http://www.dagstuhl.de/DATA/Seminars/96/>.

[MACAULAY 96] MACAULAY, L. *Requirements Engineering*. Springer, 1996.

[MYLOPOULOS 92] MYLOPOULOS, J.; CHUNG, L.; NIXON, B. *Representing and Using Nonfunctional Requirements: A Process-Oriented Approach*. IEEE Transactions on Software Engineering, v. 18, n. 6, p. 483-497, 1992.

[MYLOPOULOS 99] MYLOPOULOS, J.; CHUNG, L.; YU, E. *From Object-Oriented to Goal-Oriented Requirement Analysis*. Communications of the ACM, v. 42, n. 1, jan. 1999.

[NETO 00] NETO, J. M. S.; LEITE, J. C. S. P.; CYSNEIROS, L. M. *Non-functional Requirements For Object-Oriented Modelling*. In: III Workshop de Engenharia de Requisitos (WER 2000), 2000, Rio de Janeiro, Brasil. Anais... PUC-Rio, p. 109-125.

[NIXON 93] NIXON, B. *Dealing with Performance Requirements During the Development of Information Systems*. In: 1st International Symposium on Requirements Engineering (RE'93), 1993, San Diego, CA, USA. Proceedings... IEEE Computer Society Press, p. 42-49.

[NUSEIBEH 00] NUSEIBEH, B.; EASTERBROOK, S. *Requirements Engineering: A Roadmap*. In: International Conference on Software Engineering, 22., jun. 2000, Limerick, Ireland. Proceedings... ACM, 2000, p. 35-46.

[NUSEIBEH 94] NUSEIBEH, B.; KRAMER, J., FINKELSTEIN, A. *A Framework for Expressing the Relationships Between Multiple Views in Requirements Specifications*. IEEE Transactions on Software Engineering, v. 20, n. 10, p. 760-773, oct. 1994.

[NUSEIBEH 99] NUSEIBEH, B.; EASTERBROOK, S. *The Process of Inconsistency Management: A framework for understanding*. In: First International Workshop on the Requirements Engineering Process (REP'99), 1999, Florence, Italy. Proceedings... IEEE Computer Society Press, p. 364-368.

[PARNAS 00] PARNAS, D. *Requirements Documentation: Why a Formal Basis is Essential*. In: 4th International Conference on Requirements Engineering (ICRE 2000), 2000, Schaumburg, IL, USA. Proceedings... IEEE Computer Society Press, p. 81-82.

[PFLEEGER 98] PFLEEGER, S. L. *Software Engineering Theory and Practice*. Prentice Hall, 1998.

[PINHEIRO 00] PINHEIRO, F. A. C. *Formal and Informal Aspects of Requirements Tracing*. In: III Workshop de Engenharia de Requisitos (WER 2000), 2000, Rio de Janeiro, Brasil. Anais... PUC-Rio, p. 1-21.

[POHL 94] POHL, K. *The Three Dimensions of Requirements Engineering: A Framework and its Applications*. Information Systems, v. 19, n. 3, p. 243-258, 1994.

[POTTS 94] POTTS, C.; TAKAHASHI, K.; ANTÓN, A. I. *Inquiry-Based Requirements Analysis*. IEEE Software, v. 11, n. 2, p. 21-32, 1994

[PRESSMAN 97] PRESSMAN, R. S. *Software Engineering A Practitioner's Approach*. 4th Edition New York: Mc Graw Hill – 1997.

[PRIETO-DÍAZ 87] PRIETO-DÍAZ, R.; FREEMAN, P. *Classifying Software for Reusability*. IEEE Software, v. 4, n. 1, p. 7-16, 1987.

[RAMESH 98] RAMESH, B. *Factors Influencing Requirements Traceability Practice*. Comunications of the ACM, v. 41, n. 12, dec. 1998.

[ROBBINS 89] ROBBINS, S. P. *Organizational Behaviour: Concepts, Controversies and Applications*. 4th Edition. New Jersey: Prentice Hall, 1989.

[ROBERTSON 99] ROBERTSON, S.; ROBERTSON, J. *Mastering the Requirements Process.* Addison Wesley, 1999.

[ROBINSON 89] ROBINSON, W. N. *Integrating Multiple Specifications Using Domain Goals*. In: 5th International Workshop on Software Specification and Design, 1989, Pittsburgh, PA. Proceedings... IEEE Computer Society Press, p. 219–226.

[ROBINSON 89] ROBINSON, W. N. *Integrating Multiple Specifications Using Domain Goals*. In: 5th International Workshop on Software Specification and Design, 1989, Pittsburgh, PA. Proceedings... IEEE Computer Society Press, p. 219–226.

[ROBINSON 98] ROBINSON, W. N.; VOLKOV, S. *Supporting the Negotiation Life Cycle*. Communications of the ACM, v. 41, n. 5, p. 95-102, 1998.

[ROSS 77a] ROSS, D.; SCHOMAN, K. *Structured Analysis For Requirements Definition*. IEEE Transactions on Software Engineering, v. 3, n. 1, p. 6-15, jan. 1977.

[ROSS 77b] ROSS, D. *Structured Analysis (SA): A Language for Communicating Ideas*. IEEE Transactions on Software Engineering, v. 3, n. 1, p. 16-34, 1977.

[RUMBAUGH 91] RUMBAUGH, J.; BLAHA, M.; PREMERLANOI, W.; EDDY, F.; LORENSEN, W. *Object Oriented Modeling and Design*. Prentice Hall, 1991.

[SHAW 96] SHAW, M. *Three Patterns that help explain the development of Software Engineering*. In: Dagstuhl Seminar 9635 on History of Software Engineering, ago. 1996, p. 52-56. Disponível em: <http://www.dagstuhl.de/DATA/Seminars/96/>.

[SIDDIQI 96] SIDDIQI, J.; SHEKARAN, M. C. *Requirements Engineering: The Emerging Wisdom*. IEEE Software, v. 13, n. 2, p. 15-19, mar. 1996

[SWARTOUT 82] SWARTOUT, W.; BALZER, R. *On the Inevitable Intertwining of Specification and Implementation*. Communications of the ACM, v. 25, n. 7, p. 438-440, 1982.

[WASSERMAN 79] WASSERMAN, A. *A Specitication Method for Interactive Information Systems*. In: Specification of Reliable Software (SRS), 1979. Proceedings... IEEE Catalog No. 79 CH1401-9C, p. 68-79.

[WEIDENHAUPT 98] WEIDENHAUPT, K. et al. *Scenario Usage in System Development: A Report on Current Practice*. IEEE Software, v. 15, n. 2, p. 34-45, 1998.

[YOURDON 78] YOURDON, E. N.; CONSTANTINE, L. L. *Structured Design.* Yourdon Press, 1978.

[YU 97] YU. E. *Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering*. In: 3rd IEEE International Symposium on Requirements Enginnering (RE'97), 3., 1997, Washington D. C., USA. Proceedings... IEEE Computer Society Press, p. 226-235.

[YUE 87] YUE, K. *What Does It Mean to Say that a Specification is Complete?* In: Fourth International Workshop on Software Specification and Design (IWSSD-4), 1987, Monterey, CA, USA. Proceedings... IEEE Computer Society Press.

[ZAVE 97a] ZAVE, P. *Classification of Research Efforts in Requirements Engineering.* ACM Computing Surveys, v. 29, n. 4, p. 315-321, 1997.

[ZAVE 97b] ZAVE, P.; JACKSON, M. *Four Dark Corners of Requirements Engineering*. ACM Transactions on Software Engineering and Methodology, v. 6, n. 1, p. 1-30, 1997.

Apêndice 1

A aplicação do esquema facetado da seção 5.4 a alguns itens da referência bibliográfica deste trabalho.

1	**referência**	Structured Design E. N. Yourdon & L. L. Constantine Yourdon Press – 1978	[YOURDON 78]
	Descrição / classificação	Método para a elaboração de projeto de *software* englobando uma seqüência de passos para decomposição de um sistema em funções e a representação dos elementos dessa decomposição em uma notação específica.	
		Método-procedimento / representação / documentação / sistemas de informação	
2	**referência**	Structured Analysis For Requirements Definition D. Ross & K. Schoman IEEE Transactions on Software Engineering, V. 3(1) – Jan/1977 pp.6-15	[ROSS 77a]
	Descrição / classificação	Método para a elaboração de projeto de *software* envolvendo um processo e uma notação para representação dos elementos de composição do projeto de *software* sob a ótica de decomposição da abordagem (funções e dados).	
		Método-procedimento / representação / documentação / sistemas de informação	
3	**referência**	Structured Analysis and System Specification T. DeMarco Prentice Hall – 1979	[DEMARCO 79]
	Descrição / classificação	Método para elaboração de projeto de *software* adotando a abordagem da decomposição funcional seguida da decomposição estrutural. Prescreve passos e notação para a condução do processo.	
		Método-procedimento / representação / documentação / sistema de informação	
4	**referência**	Object Oriented Modeling and Design J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy & W. Lorensen Prentice Hall – 1991	[RUMBAUGH 91]
	Descrição / classificação	Método para análise e projeto de *software*, prescrevendo uma abordagem de decomposição estrutural seguida de decomposição funcional. Apresenta também uma notação para a representação das decomposições obtidas.	
		Método-procedimento / representação / documentação / sistemas de informação	

5	**referência**	Object-Oriented Software Engineering – A Use Case Driven Approach I. Jacobson, M. Christerson, P. Jonsson, G. Övergaard Addison Wesley – 1992	[JACOBSON 92]
	Descrição / classificação	Método de desenvolvimento de *software* baseado em decomposição estrutural seguida de decomposição funcional (paradigma orientado a objetos), concentrando-se na prescrição de um método (com sua respectiva notação) para a representação da interação do sistema com seus diversos atores.	
		Método-procedimento / representação / documentação / sistemas de informação	
6	**referência**	Object Oriented Analysis and Design with Applications Grady Booch Addison Wesley – 1994	[BOOCH 94]
	Descrição / classificação	Método de desenvolvimento de *software* baseado no paradigma orientado a objetos que prescreve procedimentos e notações para a representação das dimensões estrutural e funcional do sistema.	
		Método-procedimento / representação / documentação / sistemas de informação	
7	**referência**	Requirements Engineering in the Year 00: A Research Perspective Axel van Lamsweerde Proc of the 22nd International Conference on Software Engineering June 4 - 11, 2000, Limerick Ireland	[LAMSWEERDE 00]
	Descrição / classificação	Uma análise da pesquisa na área de Engenharia de Requisitos até o ano de 2000, focalizando a evolução das técnicas e abordagens e os problemas ainda em aberto e que deveriam ser foco da pesquisa nos próximos anos.	
		Posicionamento	
8	**referência**	Object-Oriented Development: The Fusion Method Derek Coleman et al. Prentice Hall – 1994	[COLEMAN 94]
	Descrição / classificação	Método para a condução de projeto de *software* baseado no paradigma orientado a objetos, prescrevendo processo para a condução das atividades e notação específica para representação de suas dimensões estática, dinâmica e de controle.	
		Método-procedimento / representação / documentação / sistemas de informação	

9	**referência**	Classification of Research Efforts in Requirements Engineering Pamela Zave ACM Computing Surveys, 29(4): 315-321 – 1997	[ZAVE 97a]
	Descrição / classificação	Artigo propondo uma classificação da pesquisa na área de Engenharia de Requisitos.	
		Posicionamento	
10	**referência**	Requirements Engineering: A Roadmap Bashar Nuseibeh & Steve Easterbrook Proc of the 22nd International Conference on Software Engineering June 4 - 11, 2000, Limerick Ireland	[NUSEIBEH 00]
	Descrição / classificação	Analisa as realizações da pesquisa na área de Engenharia de Requisitos para Sistemas de *Software* e indica algumas questões importantes de pesquisa para o futuro.	
		Posicionamento	
11	**referência**	Software Requirements and Specifications Michael Jackson Addison Wesley – 1995	[JACKSON 95]
	Descrição / classificação	Livro estabelecendo conceitos da área de Engenharia de Requisitos e fornecendo definições da terminologia na área.	
		Conceituação / especificação / outros sistemas	
12	**referência**	Four Dark Corners of Requirements Engineering Pamela Zave & Michael Jackson ACM Transactions on Software Engineering and Methodology, 6(1): 1-30	[ZAVE 97b]
	Descrição / classificação	Texto apontando as principais deficiências conceituais que causam ambigüidade no entendimento das necessidades da Engenharia de Requisitos. Cria um modelo conceitual para o entendimento dos requisitos separando a parte interna da máquina (*hardware* + *software*) do mundo externo no qual ela irá operar.	
		Conceituação / especificação	

13	referência	On the Inevitable Intertwining of Specification and Implementation William Swartout and Robert Balzer Communications of the ACM 25(7): 438-440 - 1982	[SWARTOUT 82]
	Descrição / classificação	Ampla discussão sobre a sobreposição (circularidade) das atividades de especificação e implementação no processo de *software*.	
		Conceituação / especificação	
14	referência	Requirements Engineering With Viewpoints Gerald Kotonya and Ian Sommerville Software Engineering Journal V11 N1 - January/1996	[KOTONYA 96]
	Descrição / classificação	Prescrição de um método para a condução do processo de requisitos para sistemas interativos baseado em perspectivas.	
		Método-procedimento / sistemas de informação	
15	referência	Techniques for Requirements Elicitation Joseph A. Goguen e Charlotte Linde Proceedings of The International Symposium on Requirements Engineering, 1993	[GOGUEN 93]
	Descrição / classificação	Análise das técnicas aplicáveis à elicitação de requisitos, enfatizando, particularmente, a forma como cada uma delas lida com as questões sociais da atividade.	
		Compêndio / consenso / elicitação	
16	referência	The Process of Inconsistency Management: A framework for understanding Bashar Nuseibeh & Steve Easterbrook First International Workshop on the Requirements Engineering Process (REP'99) Florence, Italy, September 2-3, 1999.	[NUSEIBEH 99]
	Descrição / classificação	Análise e indicação das principais atividades do processo de gerenciamento de inconsistências no processo de Engenharia de Requisitos, com a sustentação de que a obtenção da consistência é cara e, por vezes, indesejada Propõe uma estrutura segundo a qual as inconsistências são acompanhadas e analisadas para que se decida qual a orientação adotar em relação a cada uma delas.	
		Método-procedimento / representação / análise e negociação	

17	**referência**	A Cost-Value Approach for Prioritizing Requirements Joachim Karlsson & Kevin Ryan IEEE Software 14(5) – October/1997 pp.67-74	[KARLSSON 97]
	Descrição / classificação	Método para a priorização de requisitos baseado em uma técnica de custo-benefício (Analytichal Hierarchy Process).	
		Método-procedimento / consenso / análise e negociação / sistemas de informação	
18	**referência**	Requirements Engineering: The Emerging Wisdom Jawed Siddiqi & M Chandra Shekaran IEEE Software 13(2) – Mars/1996 pp. 15-19	[SIDDIQI 96]
	Descrição / classificação	Considerações sobre as várias correntes que definem requisitos e a apresentação de questões práticas que apontam na direção de se considerar com maior atenção o contexto do desenvolvimento, a acomodação da falta de completitude e o reconhecimento da natureza evolutiva da engenharia de requisitos.	
		Posicionamento	
19	**referência**	Requirements Validation Through Viewpoint Resolution Julio Cesar Sampaio do Prado Leite & Peter Freeman IEEE Transactions on Software Engineering 17(12) – December/1991	[LEITE 91]
	Descrição / classificação	Método para a integração de especificações produzidas por diferentes analistas em uma única representação que represente o consenso entre eles.	
		Método-procedimento / consenso / análise e negociação	
20	**referência**	Integrating Multiple Specifications Using Domain Goals Proc. 5th International Workshop on Software Specification and Design Pittsburgh, PA IEEE Computer Society Press, 1989, pp. 219 – 226	[ROBINSON 89]
	Descrição / classificação	Técnica para a integração de especificações, baseada em negociação, utilizando-se objetivos como elementos de priorização de requisitos.	
		Método-procedimento / consenso / análise e negociação	

21	**referência**	Negotiation and the Role of the Requirements Specification Social Dimensions of Systems Engineering: People, processes, policies and software development London: Ellis Horwood, pp144-164, 1993	[EASTERBROOK 93a]
	Descrição / classificação	Analisa a hipótese, freqüentemente assumida pela pesquisa na Engenharia de Software, da existência da especificação de requisitos. Argumenta, através da análise do papel da especificação de requisitos no processo de *software*, que é preciso entender o processo de requisitos e suas necessidades conflitantes, em especial a questão da negociação.	
		Conceituação	
22	**referência**	A Survey of Empirical Studies of Conflict. CSCW: Co-operation or Conflict? (pp.1-68) Easterbrook, S. M., Beck, E. E., Goodlet, J. S., Plowman, L., Sharples, M., & Wood, C. C. London: Springer-Verlag, 1993	[EASTERBROOK 93b]
	Descrição / classificação	Faz um breve relato das questões envolvendo conflitos e das várias maneiras de se lidar com eles focalizando nas questões de interesse ao trabalho cooperativo suportado por computador (*Computer Supported Cooperative Work*).	
		Compêndio / consenso / análise e negociação	
23	**referência**	Co-ordinating Conflicting ViewPoints by Managing Inconsistency S. M. Easterbrook, A. C. W. Finkelstein, J. Kramer, and B. A. Nuseibeh Workshop on Conflict Management in Design International Conference on Artificial Intelligence in Design Lausanne, Switzerland, 15-18 August 1994	[EASTERBROOK 94a]
	Descrição / classificação	Apresenta uma estrutura para a manutenção de diferentes especificações (viewpoints) ao longo do processo de desenvolvimento. Sustenta a fragilidade de especificações monolíticas, defendendo a existência de especificações com representações múltiplas dos requisitos.	
		Método-procedimento / representação / análise e negociação	

24	referência	Supporting the Negotiation Life Cycle Willian N. Robinson and Slav Volkov GSU – CIS – 96 – 05	[ROBINSON 98]
	Descrição / classificação	Análise do processo de negociação com o objetivo de propor uma estrutura capaz de possibilitar o entendimento da negociação automática (no sentido de que elas acontecem através do uso de algum mecanismo de comunicação remota, como a Internet) e estudar as possibilidades de influenciá-lo para obter consenso no menor tempo possível.	
		Método-procedimento / consenso / análise e negociação	
25	referência	Software Requirements Negotiation and Renegotiation Aids: A Theory-W Based Spiral Approach Barry Boehm, Prasanta Bose, Ellis Horowitz & Ming June Lee Proceedings of the 17th International Conference on Software Engineering (ICSE-17), Seattle, USA, 1995 pp. 243- 253	[BOEHM 95]
	Descrição / classificação	Propõe os principais elementos de um sistema de apoio ao modelo Win-Win Spiral Process de negociação de requisitos.	
		Método-procedimento / consenso / análise e negociação	
26	referência	Attempto Controlled English (ACE) Norbert E. Fuchs & Rolf Schwitter CLAW 96, The First International Workshop on Controlled Language Applications Katholieke Universiteit Leuven – 26-27/March/1996	[FUCHS 96]
	Descrição / classificação	Proposta de redução do vocabulário da língua inglesa visando utilizá-la como linguagem formal na especificação de sistemas.	
		Investigação-solução / especificação / validação	
27	referência	Requirements Modeling with UML Discussed Paulo Sérgio Naddeo Dias Lopes & Marcos Ribeiro Pereira Barreto	[LOPES 02]
	Descrição / classificação	Discute a adequação do uso de UML na modelagem de requisitos.	
		Conceituação / representação / documentação / sistemas de informação	

28	**referência**	Goal -directed Requirements Acquisition Anne Dardene, Axel van Lamsweerd & Stephen Fickas Science of Computer Programming 20 (1993) pp 3-50	[DARDENE 93]
	Descrição / classificação	Discussão sobre os principais conceitos observados na fase de aquisição e modelagem de requisitos e que originam a construção de modelos de requisitos. Propõe um metamodelo contendo os principais elementos conceituais que devem ser capturados e seus inter-relacionamentos sustentando que modelos de requisitos são instâncias do metamodelo proposto.	
		Conceituação / representação / documentação	
29	**referência**	Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering Eric Yu Proceedings of the 3rd IEEE International Symposium on Requirements Enginnering (RE'97)	[YU 97]
	Descrição / classificação	Método para levantamento e documentação do contexto organizacional no qual irá se inserir o desenvolvimento do sistema. Captura o contexto através do registro dos objetivos de alto nível, dos interessados em sua condução (pessoas) e das relações entre eles na fase inicial do processo de requisitos.	
		Método-procedimento / representação / sistemas de informação	
30	**referência**	From Object-Oriented to Goal-Oriented Requirement Analysis John Mylopoulos, Lawrence Chung & Eric Yu Communications of the ACM 42(1), Jan/1999	[MYLOPOULOS 99]
	Descrição / classificação	Estabelece o conceito de objetivos e define que diversos fatores contribuem positiva ou negativamente para que se atinja o objetivo identificado. Caracteriza o processo de determinação dos requisitos como um processo de intensa negociação onde é imprescindível conhecer as razões por trás dos requisitos elicitados e documentados. Introduz um mecanismo palpável para o tratamento de requisitos não funcionais.	
		Conceituação / consenso / sistemas de informação	
31	**referência**	Requirements Tracing Mathias Jarke Communications of the ACM, 41(12), Dezembro/1998	[JARKE 98a]
	Descrição / classificação	Esclarece rastreabilidade classificando-a em quatro categorias caracterizadas por como requisitos se relacionam com os demais elementos do contexto do desenvolvimento do sistema.	
		Conceituação / gerenciamento	

32	**referência**	Formal and Informal Aspects of Requirements Tracing Francisco A. C. Pinheiro III Workshop de Engenharia de Requisitos - WER 2000 Julho/2000	[PINHEIRO 00]
	Descrição / classificação	Discute a rastreabilidade a partir do princípio de que a atividade deixa sinais no processo de requisitos. Dessa forma, rastreabilidade consiste em identificar e acompanhar os sinais deixados naturalmente pela simples condução do processo de requisitos.	
		Conceituação / gerenciamento	
33	**referência**	Factors Influencing Requirements Traceability Practice Balasubramaniam Ramesh Comunications of the ACM 41(12), Dezembro/1998	[RAMESH 98]
	Descrição / classificação	Estudo de caso sobre a aplicação da rastreabilidade em um conjunto de empresas de desenvolvimento de *software*. O autor conduz uma análise sobre o efeito dos fatores ambientais, organizacionais e técnicos que influenciam a condução da rastreabilidade no processo de *software*.	
		Estudo de caso / gerenciamento	
34	**referência**	Requirements Engineering as the Reconciliation of Technical and Social Issues Joseph A. Goguen, pp.165-199 Requirements Engineering Social and Technical Issues ed. J. A. Goguen and M. Jirotka, Academic Press (1994)	[GOGUEN 94]
	Descrição / classificação	Discute a natureza social do processo de requisitos em contrapartida à natureza técnica e formal do projeto de *software*, sustentando que Engenharia de Requisitos seja a ponte de reconciliação entre esses dois mundos. Apresenta uma classificação dos métodos de condução da engenharia de requisitos baseado na premissa de que cada método traz consigo uma teoria de organizações (ainda que não articulada), que constitui uma teoria sociológica implícita.	
		Conceituação	

35	**referência**	Requirements Engineering Annotated Bibliography Steve Easterbrook & Bashar Nuseibeh **http://www.cs.toronto.edu/~sme/CSC2106S/readings.pdf**	[EASTERBROOK 02]
	Descrição / classificação	Coletânea de referências bibliográficas sobre Engenharia de Requisitos com foco na profundidade na qual abordam o tema.	
		Compêndio	
36	**referência**	The Three Dimensions of Requirements Engineering: A Framework and its applications Klaus Pohl Information Systems Vol. 19, No. 3, pp. 243-258, 1994	[POHL 94]
	Descrição / classificação	Identificação dos principais objetivos da Engenharia de Requisitos e de seus produtos, apontando as três principais dimensões através das quais se estabelece a disciplina.	
		Conceituação	
37	**Referência**	Integrating non-functional Requirements into data model Cysneiros, L.M. & Leite, J. C. S. P. 4th International Symposium on Requirements Engineering IEEE Software, Ireland – June 1999 pp. 162-171	[CYSNEIROS 99]
	Descrição / classificação	Propõe um método para representar requisitos não-funcionais em modelos de projeto utilizando diagramas de Entidade-Relacionamento.	
		Método-procedimento / representação / documentação / sistemas de informação	
38	**Referência**	Non-functional Requirements For Object-Oriented Modelling Neto, J. M. S., Leite, J. C. S. P., Cysneiros, L. M. Proceedings III Workshop de Engenharia de Requisitos – WER 2000 Julho/2000 – pp. 109-125	[NETO 00]
	Descrição / classificação	Propõe uma estratégia para, utilizando o Léxico Estendido do Domínio, construir os cenários e o diagrama de classes de forma a considerar os requisitos não funcionais.	
		Método-procedimento / representação / docuentação / sistemas de informação	

39	**referência**	Problems and Deficiencies of UML as a Requirements Specification Language Martin Glinz Proceedings of the 10th International Workshop on Software Specification and Design San Diego – November/2000, pp. 11-22	[GLINZ 00b]
		Discussão sobre a adequação da UML à modelagem de requisitos, focalizando no diagrama de casos de uso e na impossibilidade de se registrar elementos necessários aos requisitos e que sejam externos à fronteira do sistema (como relações entre atores, por exemplo).	
	Descrição / classificação	**Estudo de caso / representação / documentação / sistemas de informação**	
40	**referência**	A Framework for Integrating Non-Functional Requirements into Conceptual Models Luiz Márcio Cysneiros, Julio Cesar Sampaio do Prado Leite & Jaime de Melo Sabat Neto Requirements Engineering Journal 6(2) / 2001 – pp. 97-115	[CYSNEIROS 01]
	Descrição / classificação	Propõe uma técnica para integrar requisitos não funcionais em modelos de projeto de *software*, em particular aqueles utilizando diagramas de Entidade Relacionamento, na abordagem estruturada, e diagramas de classes, na abordagem orientada a objetos.	
		Método-procedimento / representação / documentação / sistemas de informação	