Universidade Federal de Pernambuco
Centro de Informática

Uma Metodologia para Definição de Requisitos em Sistemas Data Warehouse

Por
Fábio Rilston Silva Paim
Dissertação de Mestrado

Recife
Fevereiro, 2003

Fábio Rilston Silva Paim

Uma Metodologia para Definição de Requisitos em Sistemas Data Warehouse

Dissertação apresentada à Coordenação da Pós-Graduação em Ciência da Computação do Centro de Informática como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação.

Orientador: Jaelson Freire Brelaz de Castro

Recife

Fevereiro, 2003

A Deus.

AGRADECIMENTOS

A Deus e aos bons espíritos, por toda a luz e inspiração.
À minha família, pelo apoio incondicional nos momentos difíceis, apesar da distância.
Ao meu orientador, pela paciência infinita em aturar uma pessoa tão difícil; pela competência indiscutível; e por acreditar em meu potencial quando tudo parecia perdido.
Aos membros da banca examinadora, pelas contribuições para o enriquecimento dessa dissertação.
A Káthia Marçal, minha eterna irmã, pela fé em mim sempre, mesmo nos momentos em que nem eu conseguia ter fé.
A Helena Cristina, pelo apoio e estímulo constantes à minha pesquisa, e pela oportunidade de desenvolver este trabalho no SERPRO.
A Mozart Rhamnuzia, por me permitir essa oportunidade única.
A Ana Elizabete Carvalho, pela parceria nas idéias e discussões.
A Cristina Ciferri, pela força e pelas contribuições valiosas a essa dissertação.
A Anjolina e Ruy pelo apoio nas horas difíceis e por nunca me deixarem desistir.
A Ângela Sakamoto, pela amizade e o carinho em todos os momentos,
A Rosa Nascimento e Ivson, pelo suporte fundamental nos primeiros meses.
Ao professor Décio Fonseca, por tudo o que sei sobre Data Warehouse.
A Elza, Erlan e Mônica, pela preocupação e por fornecerem meios para que esse trabalho fluísse mesmo longe de casa.
A Simone Ramos, pela paciência e colaboração na compilação do Estudo de Caso.
A Patrícia Batista, por dar vida ao framework DW-ENF.
A Suedy, Carlos Augusto e Leonardo pela colaboração com os dados sobre a ferramenta OLAP.
Aos demais colegas do projeto SAFE, pela dedicação e contribuições.
Ao cliente, sem o qual este trabalho não teria sentido.
Aos meus amigos, por assim continuarem, apesar da ausência constante de suas vidas.
A todos que direta ou indiretamente contribuíram para a realização deste trabalho.

RESUMO

No início da década de 90, a tecnologia de *Data Warehouse* floresceu como uma solução à necessidade crescente de informações integradas pela gerência estratégica das organizações, como parte de processos decisórios de alto nível. A partir da análise histórica de dados operacionais, sistemas *data warehouse* possibilitam a descoberta de tendências de negócio que se traduzem em aumento da vantagem competitiva com a definição de processos operacionais mais precisos.

Devido à complexidade e o volume de dados processados, o desenvolvimento de aplicações data warehouse tem sido fortemente influenciado, ao longo dos anos, pela organização física do banco de dados adotado e por outros aspectos implementacionais que têm como principal objetivo assegurar performance e integridade à aplicação. Contudo, apesar dos inegáveis benefícios, essa abordagem tem como conseqüência direta uma especificação de requisitos pobre, que não reflete de forma completa e adequada as necessidades do usuário. Em decorrência disso, soluções de suporte à decisão apresentam freqüentemente efeitos colaterais tais como altos índices de erro, escalabilidade limitada, baixa qualidade da informação agregada e um distanciamento acentuado entre desenvolvedores e usuários, contribuindo para o fracasso de inúmeros projetos.

Esse cenário sugere a utilização de técnicas de Engenharia de Requisitos em suporte a uma especificação de requisitos de alta qualidade em sistemas *data warehouse*. Nesse sentido, uma abordagem metodológica é requerida para garantir (*i*) a correta identificação dos requisitos do usuário, (*ii*) um projeto de sistema afinado e (*iii*) a definição de um modelo dimensional que preencha todas as necessidades de análise estratégica dos usuários finais.

Essa dissertação propõe uma metodologia para definição de requisitos em sistemas *data warehouse*. A abordagem adotada está centrada em um modelo de fases que direciona o processo de especificação dos requisitos, enquanto propõe um conjunto de artefatos para a coleta dos aspectos funcionais, não-funcionais e multidimensionais que integram a solicitação de serviço. Aliada a técnicas eficientes para gerenciamento de requisitos, a metodologia serve como um instrumento para rastrear as mudanças sofridas pelos requisitos ao longo do projeto, e assim possibilitar que ambas as visões dimensional e de requisitos estejam sempre ajustadas às necessidades do usuário.

Os princípios que servem como pano de fundo à metodologia foram concebidos ao longo do desenvolvimento de um projeto *data warehouse* de larga escala conduzido pelo SERPRO, uma empresa do Ministério da Fazenda, em sua maior unidade de negócio, a SUNAT – Superintendência de Negócios e Administração Tributária. Um estudo de caso ao final da dissertação apresenta as contribuições do uso da metodologia para o referido projeto.

ABSTRACT

In the early nineties, the Data Warehouse technology flourished as a promising solution to the increasingly growing need for integrated information from organizations' strategical level, as part of higher-order decision-support processes. Based on analysis of historical series of operational data, data warehouse systems allow for carving out business tendencies that can increase the company's competitive advantage by means of defining sharper operational processes.

Due to data volume and complexity, the development of data warehouse applications has been strongly influenced over the years by a number of implementation aspects such as database physical organization, which in common seek to provide performance and integrity to the underlying application. In spite of the undeniable benefits, however, a direct consequence of this approach is a poor requirements specification, which lacks a comprehensive, appropriate representation of user needs. As a result, decision-support systems often show side effects like high fault rates, restricted scalability, aggregated information low-quality and a gap between developers and users. All together these aspects contribute to failure in various projects.

Such scenario clearly advocates the use of Requirements Engineering techniques in support of a higher quality data warehouse requirements specification. In this sense, a methodological approach is required to guarantee (i) the correct comprehension of user requirements, (ii) a fine-tuned design phase and (iii) the construction of a dimensional schema that enables decision makers to perform all necessary analysis.

This dissertation aims at defining a methodology for requirements definition of data warehouse systems. The approach is centred in a phase-oriented framework that guides the requirements specification process, whereas proposes a set of artefacts to collect each functional, non-functional and multidimensional aspect pertaining user demand. Furthermore, such methodology, allied to efficient requirements management techniques, works as an instrument to trace the changes in user requirements along the project, in order to keep both dimensional and requirements visions aligned to the user's strategical needs.

The core principles behind the present methodology were developed throughout an ongoing large data warehouse project conducted by SERPRO, a Brazilian Finance Ministry Enterprise, in its largest business unit, SUNAT – Superintendência de Negócios e Administração Tributária *(Business and Tax Administration Superintendence). A case study in the end of this work presents the main contributions brought by the methodology application to the related project.*

CONTEÚDO

Índice de Figuras

Índice de Tabelas

Capítulo 1

Introdução

Este capítulo descreve as principais motivações para realização deste trabalho. As seções seguintes apresentam o escopo da dissertação e, finalmente, a estrutura do trabalho.

1.1 Motivações

No moderno mundo de negócios, a capacidade de responder de forma flexível às rápidas mudanças de ambiente tornou-se um fator primordial para o sucesso de qualquer empresa. À medida em que o progresso tecnológico caminha em velocidade exponencial, o ciclo de vida de desenvolvimento de produtos torna-se cada vez menor, obrigando as organizações a tomarem decisões tanto operacionais quanto estratégicas em intervalos decrescentes de tempo. Nesse cenário, a competitividade de uma organização passa a ser determinada, em grande parte, pela habilidade de seu alto escalão em tomar decisões precisas que garantam o sucesso de sua estratégia de negócio. Essas decisões devem estar baseadas em informações atualizadas, completas e de alta qualidade.

Em quase todos os setores de negócio e prestação de serviços, sistemas gerenciadores de bancos de dados (SGBDs) robustos e avançadas aplicações de software para processamento de transações *on-line* têm sido desenvolvidos com o objetivo de prover armazenamento persistente, bem como acesso eficiente e atualizado a grandes volumes de informação. Muito embora relevantes para a gerência dos processos operacionais, essas fontes de informação e os sistemas que as controlam possuem um potencial de suporte à decisão limitado. A maioria desses sistemas não consegue recuperar a informação na forma requerida para sustentar uma tomada de decisão rápida e inteligente. MOHANIA *et al.* (1999) destacam que aplicações convencionais têm dificuldade de responder a perguntas como "*quais foram os padrões de suprimento do produto X nas filiais da Cidade A no ano de 2002 e o quanto eles foram diferentes do ano anterior?*".

No início da década de 90, Sistemas *Data Warehouse* surgiram como uma solução para este e outros problemas de tomada de decisão. O princípio que diferencia esses sistemas do mundo OLTP (*On-Line Transaction Processing*) é a separação, tanto física quanto lógica, entre dados operacionais e o processamento de informações com fins estratégicos. A conexão entre fontes de dados operacionais e o sistema de suporte a decisão é estabelecida por um banco de dados intermediário – o *warehouse* – que atua como uma base de

informações comum de toda a empresa. Por um processo de extração, limpeza, transformação e agregação, o dado operacional é obtido das fontes provedoras e armazenado no *warehouse* segundo um esquema multidimensional que traduz, em última instância, o dado em informação *factual* (o que se deseja analisar) e *dimensional* (a perspectiva estratégica sobre a qual o fato pode ser analisado). A informação transpõe o enfoque OLTP para uma abordagem OLAP (*On-Line Analytical Processing*) (CODD *et al.*, 1993), pela qual fatos e dimensões interligados por um modelo intrincado, comumente referenciado como *cubo de dados* (BOEHNLEIN e ULBRICH-VOM ENDE, 1999), possibilitam executar consultas complexas ao *warehouse* de forma rápida e flexível.

Nos dias atuais, uma parcela significativa do orçamento em Tecnologia da Informação de muitas organizações é dedicada ao desenvolvimento de aplicações *Data Warehouse* (MOODY e KORTINK, 2000; SEN e JACOB, 1998; WATTERSON, 1998). Relatos de projetos bem-sucedidos na literatura reportam os altos níveis de satisfação do usuário e o retorno no investimento (GRAHAM *et al.*, 1996). Contudo, a despeito de todo o potencial, muitos projetos *Data Warehouse* ao redor do mundo fracassam em sua implantação (KELLY, 1997; SILVERSTON *et al.*, 1997). Algumas das razões por trás desses insucessos são: *(i)* especificações de sistema muito influenciadas pelos aspectos de implementação física do banco de dados - ainda um resquício dos primeiros anos de evolução da tecnologia, onde o objetivo primordial era a otimização das complexas consultas analíticas; *(ii)* no nível de abstração conceitual, os projetos não são concebidos tendo as necessidades dos usuários em mente (TRYFONA *et al.*, 1999); *(iii)* dificuldade em traduzir os requisitos do usuário em um modelo de fatos e dimensões; *(iv)* falhas nas cadeias de agregação entre fatos e dimensões, devido à especificação de relações multidimensionais incompatíveis (LEHNER *et al.*, 1998); *(v)* inexistência de uma metodologia padrão que direcione a construção de *data warehouses* de acordo com um processo passo a passo (HÜSEMAN *et al.*, 2000); *(vi)* altos índices de erro, provenientes da baixa qualidade da informação agregada; *(vii)* escalabilidade limitada, impedindo a evolução gradual das visões de negócio; *(viii)* distanciamento entre usuários e desenvolvedores, agravado pelo desconhecimento da tecnologia (VASSILIADIS, 2000);

(ix) modelagem da integração com fontes provedoras baseada em processos *ad-hoc*, ao invés de uma abordagem sistemática (CABIBBO e TORLONE, 1998). Este conjunto de fatores aponta para uma especificação deficiente dos requisitos de sistema e a ausência de um processo de especificação bem definido.

O cenário acima sugere a utilização de técnicas de *Engenharia de Requisitos* (VAN LAMSWEERDE, 2000) em suporte a uma especificação de requisitos de alta qualidade para sistemas *data warehouse*. Por meio de um *processo de engenharia de requisitos* (SOMMERVILLE e SAWYER, 1997), mecanismos apropriados podem ser canalizados para entender o que o cliente deseja; analisar suas necessidades; verificar a viabilidade; negociar uma solução compatível, especificando-a de forma a evitar ambigüidades; validar a especificação produzida; e gerenciar os requisitos de software enquanto são transformados em um sistema operacional (THAYER e DORFMAN, 1997). Aplicado sob uma abordagem metodológica ao projeto de *data warehouses*, esse processo permite garantir: *(i)* uma melhor identificação dos requisitos do usuário, *(ii)* um projeto de sistema afinado às características multidimensionais inerentes a aplicações de suporte à decisão, e *(iii)* a definição de um esquema dimensional que preencha todas as necessidades de análise estratégica dos usuários finais.

1.2 Escopo da Dissertação

Esta dissertação tem por objetivo a definição de uma metodologia para definição de requisitos em sistemas *data warehouse*. A abordagem proposta será descrita em um modelo de fases que direciona o processo de especificação dos requisitos e propõe um conjunto de artefatos gerados e utilizados para o relato dos aspectos funcionais, não-funcionais e multidimensionais que integram a solicitação de serviço. Ao longo do trabalho, será mostrado como o processo de requisitos padrão, adaptado para o domínio específico de sistemas de suporte à decisão, contribui para a geração de uma especificação de sistema aderente às características particulares do domínio proposto.

A influência do projeto arquitetônico de bancos multidimensionais na elaboração de uma especificação em alto nível dos requisitos de sistemas *data warehouse* será explorada ao longo das fases da metodologia, muito embora esse trabalho não se proponha a discutir a implementação de soluções arquitetônicas para essa classe de sistemas. Será demonstrado ainda que, aliada a ferramentas eficientes para gerenciamento de requisitos, a metodologia funciona como um instrumento para rastrear as mudanças sofridas pelos requisitos ao longo do projeto, e assim possibilitar que ambas as visões dimensional e de requisitos estejam sempre ajustadas às necessidades do usuário.

1.3 Estrutura da Dissertação

Além deste capítulo introdutório, esse trabalho consiste de mais cinco capítulos, os quais são descritos a seguir.

No capítulo 2 traçamos uma visão geral da Engenharia de Requisitos e seus fundamentos. Iniciamos tecendo considerações sobre o que sejam requisitos na Seção 2.2. Ainda na mesma seção, conceituamos Engenharia de Requisitos e discorremos, na Seção 2.3, sobre o seu papel no ciclo de vida do desenvolvimento de software. O processo de Engenharia de Requisitos e suas fases constituintes é detalhado na Seção 2.4, enquanto a Seção 2.5 descreve as principais abordagens para modelagem dos requisitos de uma aplicação de software. A Seção 2.6 resume então nossas considerações sobre o tema, ressaltando a importância da integração entre o processo de engenharia de requisitos e o desenvolvimento de *data warehouses*.

No capítulo 3 introduzimos a tecnologia de *Data Warehousing* e os desafios que esta acrescenta ao desenvolvimento tradicional de aplicações. Inicialmente, conceituamos o que sejam *data warehouses* e sua inserção no processo de *data warehousing* (Seção 3.2). Os principais conceitos relativos à modelagem de sistemas para suporte à decisão são definidos na Seção 3.3. O processo de *Data Warehousing* é detalhado na Seção 3.4. A Seção 3.5 traça, então, um perfil dos sistemas de suporte à decisão, ressaltando as peculiaridades destes em relação a sistemas transacionais convencionais. Em seguida, a Seção 3.6 examina

as metodologias atuais para desenvolvimento de sistemas *data warehouse* à luz de sua contribuição para uma especificação de requisitos eficiente. Ao final do capítulo, a Seção 3.6.6 resume nossas considerações sobre a necessidade de um processo de requisitos em suporte ao desenvolvimento de sistemas *data warehouse*.

Uma metodologia para a definição de requisitos em sistemas *data warehouse* é então proposta no capítulo 4. Inicialmente, tecemos na Seção 4.2 considerações sobre a natureza do processo de desenvolvimento de sistemas *data warehouse* e os requisitos essenciais à sua correta especificação. Na Seção 4.3, descrevemos brevemente a estrutura da metodologia. As seções de 4.4 a 4.7 detalham as fases da metodologia, enquanto a Seção 4.8 descreve a integração dessas fases com o novo ciclo de requisitos proposto para o desenvolvimento de aplicações *data warehouse*. A Seção 4.9 resume, ao final, nossas considerações sobre a abordagem descrita.

O capítulo 5 descreve as contribuições da aplicação da metodologia no desenvolvimento de um *data warehouse* de grande porte pelo SERPRO, em sua unidade de desenvolvimento regional da SUNAT-Recife. O capítulo inicia-se apresentando a empresa SERPRO (Seção 5.2) e a SUNAT (Seção 5.3). Em seguida, a estrutura e objetivos do projeto são descritos em maiores detalhes (Seção 5.4). A experiência com a aplicação da metodologia e os resultados alcançados são relatados na Seção 5.5. Na Seção 5.6, apresentamos nossas considerações finais sobre o estudo de caso.

O capítulo 6 discute as principais contribuições do trabalho para a melhoria da especificação e gerência de requisitos em sistemas *data warehouse* (Seção 6.1). Em seguida, descreve trabalhos futuros (Seção 6.2) que poderão ser realizados a partir dessa dissertação e, por último, fornece alguns possíveis direcionamentos para pesquisas futuras na área.

Capítulo 2

Engenharia de Requisitos

Este capítulo apresenta uma visão geral sobre Engenharia de Requisitos. Inicialmente, são tecidas considerações a respeito do que seja requisito, sua classificação e importância no processo de desenvolvimento de software. Em seguida, conceitua-se Engenharia de Requisitos e apresenta-se uma visão de seus processos e principais características, mostrando os benefícios da aplicação de tais fundamentos para o projeto de um software.

2.1 Introdução

Quando a "*Crise do Software*" (NAUR e RANDELL, 1969) foi nomeada nos anos 60, muito esforço foi direcionado para encontrar as causas da síndrome de problemas. Ainda na década de 70, investigações determinaram que deficiências nos **requisitos** estavam entre as mais importantes causas do problema: "*Em quase todo projeto de software a falha nos objetivos de desempenho e custo, requisitos inadequados desempenham o papel mais expressivo na falha do projeto*" (ALFORD e LAWSON, 1979). O desenvolvimento da especificação de requisitos "*em muitos casos parece trivial, mas provavelmente é a parte do processo que leva a mais falhas que qualquer outra*" (SCHWARTZ, 1975). Levantamentos recentes da comunidade européia, bem como do *Software Engineering Institute* (SEI), nos Estados Unidos, ainda confirmam deficiências na *especificação* e na *gerência de requisitos* como os principais aspectos responsáveis pela atual crise de software. No Brasil, estudos recentes corroboram essa constatação (PINHEIRO *et al.*, 2003).

Com a evolução e popularização dos computadores, a Engenharia de Software tem tentado resolver os problemas do desenvolvimento de software, em especial aqueles relacionados com os requisitos do software. O foco está em prevenir a insatisfação do cliente por meio de um melhor entendimento dos requisitos estabelecidos e derivados, e então utilizá-lo no projeto do software (ZULTNER, 1993). Quando os requisitos de um sistema são pobremente entendidos, possivelmente o produto final não será livre de falhas, e a versão do sistema entregue ao cliente não corresponderá ao produto encomendado inicialmente. De fato, em seu estudo sobre erros de software nos programas *Voyager* e *Galileo* da NASA, LUTZ (1993) apontou como principal causa de falhas de segurança erros na especificação de requisitos funcionais e de interface.

Adicionalmente, é comum que ao longo do ciclo de desenvolvimento ocorram mudanças nos requisitos do sistema. Tais mudanças acarretam uma série de efeitos colaterais ao projeto de software como atrasos no cronograma de entrega do sistema, necessidade de

alocar recursos adicionais, retrabalho, ou até mesmo a inviabilidade parcial ou total do projeto. Contraditoriamente, o que ocorre é que essas mudanças raramente são documentadas, e os impactos no projeto não são tratados adequadamente.

Com a evolução da Engenharia de Software, porém, várias empresas têm procurado aperfeiçoar os métodos para capturar, analisar, validar e gerenciar os seus requisitos. A Engenharia de Requisitos tem auxiliado os engenheiros de software nesse sentido, provendo mecanismos que auxiliam no entendimento do que o cliente precisa; na análise de suas necessidades; na avaliação da exeqüibilidade da solução proposta; na negociação de uma estratégia adequada à sua construção; na elaboração de uma especificação livre de ambigüidades; na validação dessa especificação junto aos clientes/usuários; e, finalmente, no gerenciamento dos requisitos coletados à medida em que estes são transformados em um sistema operacional (THAYER e DORFMAN, 1997).

Este capítulo apresenta uma visão geral sobre a Engenharia de Requisitos. Inicialmente, são tecidas considerações a respeito do que seja “requisito” (Seção 2.2). Em seguida, conceitua-se Engenharia de Requisitos e sua importância no processo de desenvolvimento de software (Seção 2.3). A Seção 2.4 descreve, então, as fases do processo de engenharia de requisitos e suas principais características. Posteriormente, são apresentados os métodos de modelagem mais utilizados para cada uma das visões organizacional, funcional e não-funcional dos requisitos de um sistema na Seção 2.5. A Seção 2.6 encerra o capítulo demonstrando nossas considerações finais sobre a importância da engenharia de requisitos para a qualidade de um projeto de *data warehouse*.

2.2 Visão Geral de Requisitos

Uma das primeiras medidas do sucesso de um sistema de software é verificar se ele atende às necessidades dos clientes (NUSEIBEH e EASTERBROOK, 2000). Num dos primeiros trabalhos realizados na área, BELL e THAYER (1976) observaram que muitos requisitos são inadequados, inconsistentes, incompletos e ambíguos e exercem um grande impacto na qualidade do software final. A partir dessa observação, eles concluíram que "*requisitos*

para um dado sistema não aparecem naturalmente; ao contrário, eles precisam ser projetados e revisados continuamente".

Estudos indicam que, quando detectados apenas depois do software implementado, erros em requisitos de software são até 20 vezes mais caros de se corrigir que qualquer outro tipo de erro, e até 200 vezes mais custosos do que seriam se corrigidos durante a fase de engenharia dos requisitos (BOEHM, 1981; 1984). Estudos recentes comprovam que os problemas relacionados aos requisitos do sistema afetam boa parte das organizações que desenvolvem e usam sistemas de software (JOHNSON, 2001; ESPITI, 1996). Corroborando essa constatação, novos estudos reconhecem a *Engenharia de Requisitos* como uma das fases mais importantes do processo de engenharia de software (VAN LAMSWEERDE, 2000; PINHEIRO *et al*., 2003).

Todavia, a fim de melhorar a qualidade dos requisitos, é necessário definir inicialmente o que são realmente "requisitos". Existem inúmeras definições disponíveis na literatura para o termo. Segundo KOTONYA e SOMMERVILLE (1997), um requisito pode descrever:

(1) Uma facilidade no nível do usuário; por exemplo, um corretor de gramática e ortografia;

(2) Uma propriedade muito geral do sistema; por exemplo, o sigilo de informações sem a devida autorização;

(3) Uma restrição específica no sistema; por exemplo, o tempo de varredura de um sensor;

(4) Uma restrição no desenvolvimento do sistema; por exemplo, a linguagem que deverá ser utilizada para o desenvolvimento do sistema.

Uma definição bastante simples para requisitos é dada por MACAULAY (1996), segundo o qual requisito é "*simplesmente algo de que o cliente necessita*". Ainda segundo JACKSON (1995), requisitos são fenômenos ou propriedades do domínio da aplicação que devem ser

executados, normalmente expressos em linguagem natural, diagrama informal ou outra notação apropriada ao entendimento do cliente e da equipe de desenvolvimento. Nesse trabalho, adotamos a definição de DORFMAN e THAYER (1990), os quais conceituam requisitos como sendo:

(a) uma capacidade do software necessitada pelo usuário para resolver um problema e atingir um objetivo;

(b) uma capacidade de software que um sistema (ou um seu componente) deve atingir ou possuir para satisfazer um contrato, padrão, especificação, ou outra documentação formalmente imposta.

Além de *requisitos*, outra definição necessária é a de *Engenharia de Requisitos*. Segundo o IEEE, engenharia de requisitos corresponde ao processo de aquisição, refinamento e verificação das necessidades do cliente para um sistema de software, objetivando-se ter uma especificação completa e correta dos requisitos de software (IEEE, 1984). ZAVE (1997) coloca que a engenharia de requisitos está relacionada com a identificação de metas para serem atingidas pelo sistema a ser desenvolvido, assim como a operacionalização de tais metas em serviços e restrições. No contexto dessa dissertação, adotaremos a definição de LOUCOPOULOS e KARAKOSTAS (1995), os quais definem engenharia de requisitos como o processo sistemático de desenvolvimento dos requisitos por meio de um processo iterativo e cooperativo de análise do problema, documentação das observações resultantes numa variedade de formatos, e checagem da acurácia do entendimento obtido. Em termos simples, o objetivo desse processo é desenvolver uma *especificação de requisitos* (COMPUTER, 1985), o que envolve uma integração de conceitos relacionados com aspectos de representação, sociais e cognitivos (POHL, 1993). Essa diversidade de fatores faz da engenharia de requisitos uma área eminentemente multidisciplinar, onde aspectos sociais e humanos desempenham um importante papel (ZAVE, 1997; NUSEIBEH e EASTERBROOK, 2000).

Existem pelo menos três grandes benefícios em desenvolver uma especificação de requisitos (LOUCOPOULOS e KARAKOSTAS, 1995):

(1) O documento gerado provê um ponto focal para o processo de comunicar um entendimento sobre um dado domínio, negócio e o próprio sistema-alvo. A especificação funciona, então, como uma linguagem para representar conteúdos;

(2) A especificação serve como parte de um acordo contratual, um artifício que pode se tornar bastante relevante para o relacionamento com agentes externos como clientes e fornecedores;

(3) A especificação pode ser usada para avaliar o produto final e assim ter um papel significativo em testes de aceitação do sistema.

Uma especificação de requisitos completa abrange uma ampla gama de requisitos. Tradicionalmente, os requisitos de software são classificados em requisitos *funcionais*, *não-funcionais* e *organizacionais* (SOMMERVILLE, 2001; ALENCAR, 1999):

- **Requisitos Funcionais**: são as declarações das funções que o sistema deve oferecer, como o sistema se comporta com entradas particulares e como o sistema deve se comportar em situações específicas. O termo função é usado no sentido genérico da operação que pode ser realizada pelo sistema, seja por meio de comandos dos usuários, ou seja pela ocorrência de eventos internos ou externos ao sistema. Em alguns casos, os requisitos funcionais podem também explicitamente definir o que o sistema **não deve** fazer.

- **Requisitos Não-Funcionais**: são as restrições nas funções oferecidas pelo sistema. Incluem restrições de tempo, restrições no processo de desenvolvimento, padrões, e qualidades globais de um software, como manutenibilidade, usabilidade, desempenho, custos e várias outras.

- **Requisitos Organizacionais**: derivados diretamente de procedimentos e políticas organizacionais e relacionados com os objetivos e metas da organização.

A necessidade de se estabelecer os requisitos de forma mais precisa é crítica na medida que o tamanho e a complexidade do software aumentam. Os requisitos exercem influência uns sobre os outros. Por exemplo, o requisito de que o software deve ter grande portabilidade pode implicar que o requisito desempenho não seja satisfeito.

2.3 Engenharia de Requisitos e o Ciclo de Vida do Desenvolvimento

Desenvolver sistemas constitui uma atividade de projeto (*design*) (LOUCOPOULOS e KARAKOSTAS, 1995). Um projeto de sistema tipicamente envolve os seguintes aspectos (DASGUPTA, 1991): (*a*) um conjunto de requisitos a ser satisfeito por algum artefato; (*b*) a saída do processo é algum tipo de documento de especificação de projeto; (*c*) o objetivo do desenvolvedor é gerar um projeto de modo que se alguma implementação desse projeto tiver que se materializar, então o artefato deve satisfazer os requisitos; (*d*) o desenvolvedor não detém conhecimento de um nenhum projeto prévio que satisfaz os requisitos.

Esses aspectos deixam aparente a estreita relação entre a engenharia de requisitos e a atividade de projetar software, bem como o grau de dificuldade enfrentado por engenheiros de sistema ao tentar traduzir requisitos do usuário numa solução de software. De fato, o software em si não possui valor se seus usuários não conseguem utilizá-lo para satisfazer suas necessidades de resolução de problemas. A *análise de requisitos* é o processo que visa a descoberta de requisitos; sua análise para identificar relevância, inconsistências, ausência de requisitos e praticidade; e a negociação dos requisitos finais para o sistema (KOTONYA e SOMMERVILLE, 1995).

Em contrapartida, os diversos relacionamentos e restrições que os requisitos exercem uns sobre os outros são muito difíceis de serem controlados. Principalmente se considerarmos que algumas decisões de projeto que afetam um ou mais requisitos só serão tomadas em fases adiantadas do desenvolvimento (STRAW01, 2001). Dessa forma, os requisitos precisam ser gerenciados ao longo de todo o ciclo de vida do projeto. Isso requer a definição e uso de processos, métodos e ferramentas dentro do contexto de uma estratégia

global de desenvolvimento, comumente referenciada na literatura como *modelo de processo de software* (PRESSMAN, 2001). Inúmeros modelos de processo incluem atividades de engenharia de requisitos em seu bojo (ROYCE, 1970; FLOYD, 1984; BOEHM, 1988, 1998a; ARANGO, 1988; McDERMID e ROOK, 1993; DAVIS e SITARAM, 1994; KRUCHTEN, 1999). Em geral, essas atividades atendem a três grupos principais de objetivos (ZAVE, 1997) conforme descrito na Tabela 1.

G1	**Investigação de objetivos, funções e restrições de uma aplicação de software**	Ultrapassar as barreiras de comunicação
		Gerar estratégias para converter objetivos vagos em propriedades ou restrições específicas
		Compreender prioridades e graus de satisfação
		Associar requisitos com os vários agentes do domínio
		Estimar custos, riscos e cronogramas
		Assegurar completude
G2	**Especificação do software**	Integrar representações e visões múltiplas
		Avaliar estratégias de soluções alternativas que satisfaçam os requisitos
		Obter uma especificação mais completa, mais consistente e menos ambígua
		Validar a especificação (*verificar que ela satisfaz aos requisitos*)
		Obter especificações que sejam apropriadas para as atividades de projeto (design) e implementação
G3	**Gerenciamento da evolução e das famílias do software**	Reutilização de requisitos durante o processo de evolução
		Reutilização de requisitos no desenvolvimento de software similares
		Reconstrução de requisitos

Tabela 1. Objetivos da Engenharia de Requisitos agrupados por categoria.

Normalmente, a engenharia de requisitos é considerada a atividade inicial do processo de desenvolvimento (CARVALHO, 2002). Informações sobre sistemas já existentes, necessidades das partes interessadas (*stakeholders*), padrões da organização, regulamentações, informações do domínio da aplicação, entre outros insumos, são fornecidos como entrada ao processo de análise dos requisitos, o qual produz como resultado *requisitos acordados*, uma *especificação de requisitos* e *um modelo de requisitos*. Quando a fase de requisitos é executada de forma incompleta ou inconsistente, os artefatos resultantes comprometem a qualidade do produto desenvolvido nas etapas seguintes do processo de software (Projeto, Implementação e Testes). Em contrapartida, os benefícios

que a aplicação correta dos princípios de engenharia de requisitos podem trazer ao processo de desenvolvimento incluem:

- Concordância entre desenvolvedores, clientes e usuário sobre o trabalho a ser feito e quais os critérios de aceitação do sistema.
- Uma base precisa para a estimativa dos recursos (custo, pessoal, prazos, ferramentas e equipamentos).
- Melhoria na usabilidade, manutenibilidade e outras qualidades do sistema.
- Atingir os objetivos com o mínimo de desperdício.

Na seção seguinte, descrevemos como o processo de engenharia de requisitos é estruturado de forma a atingir esses objetivos.

2.4 O Processo de Engenharia de Requisitos

De acordo com a *International Standards Organization* (ISO), o termo processo é definido como "*um grupo de atividades inter-relacionadas que se caracterizam por uma série de entradas específicas que agregam valor e fornecem uma série de saídas específicas para clientes externos e internos*".

O processo de engenharia de requisitos é um conjunto estruturado de atividades que são seguidas para derivar, validar e manter um documento de requisito (KOTONYA e SOMMERVILLE, 1997). Uma descrição completa do processo deve incluir: (*a*) quais atividades são executadas; (*b*) a estruturação e o cronograma delas; (*c*) quem é o responsável por cada uma das atividades; (*d*) suas entradas e saídas; e (*e*) as ferramentas usadas para suportar a engenharia de requisitos.

Embora o processo de Engenharia de Requisitos varie de uma organização para outra, podendo diferir inclusive na mesma organização, suas entradas e saídas, na maioria dos

casos, são similares. O processo de Engenharia de Requisitos é um processo de projeto (*design*) com entradas e saídas, como descrito na Tabela 2.

ENTRADA OU SAÍDA	TIPO	DESCRIÇÃO
Informações dos Sistemas Existentes	Entrada	Refere-se a informações gerais sobre o sistema que será substituído ou criado e de outros sistemas com o qual o sistema deverá interagir.
Necessidades das partes interessadas (*stakeholders*)	Entrada	Refere-se a uma descrição das necessidades das partes interessadas para apoiar seu trabalho.
Padrões corporativos	Entrada	Refere-se a padrões e normas adotadas pela empresa para o desenvolvimento de sistemas, incluindo métodos para o desenvolvimento, práticas para garantia de qualidade, etc..
Normas e regulamentos	Entrada	Normas e regulamentos externos que se apliquem ao sistema.
Informações do Domínio	Entrada	Informações gerais sobre o domínio do sistema.
Requisitos Definidos	Saída	Descrição dos requisitos levantados, avaliados e aprovados pelas partes interessadas.
Especificação do Sistema	Saída	Uma especificação mais detalhada do sistema a ser desenvolvido.
Modelos do Sistema	Saída	Um conjunto de modelos que descrevem o sistema a partir de diferentes perspectivas.

Tabela 2. Entradas e Saídas do Processo de Engenharia de Requisitos.
(KOTONYA e SOMMERVILLE, 1997)

Dentre as várias propostas para modelos de processo de engenharia de requisitos existentes na literatura, adotamos para o contexto dessa dissertação o modelo proposto por KOTONYA e SOMMERVILLE (1997). O modelo está baseado num ciclo em espiral (Figura 1), onde o início do processo ocorre com a execução de atividades de **elicitação de requisitos**, seguidas pela **análise e negociação** desses requisitos, sua **documentação** e posterior **validação**. O processo se repete ao longo dos ciclos da espiral tantas vezes quantas sejam necessárias, até que se atinja um ***ponto de decisão de aceitação*** do documento final de requisitos. Para NUSEIBEH e EASTERBROOK (2000), esse modelo espiral constitui-se, na prática, de um conjunto de processos interativos, interelacionados e com retroalimentação (*feedback*), que podem cobrir todo o ciclo de vida do projeto de software. Não obstante, a aplicabilidade do modelo dependerá, dentre outros fatores, do

domínio da aplicação considerada. No capítulo 4, adaptamos e estendemos a proposta de KOTONYA e SOMMERVILLE (1997) para contemplar aspectos de requisitos peculiares ao domínio de aplicações *data warehouse* e sua inerente multidimensionalidade. Por ora, descrevemos nas seções seguintes a natureza e o propósito das fases que integram originalmente o referido modelo.

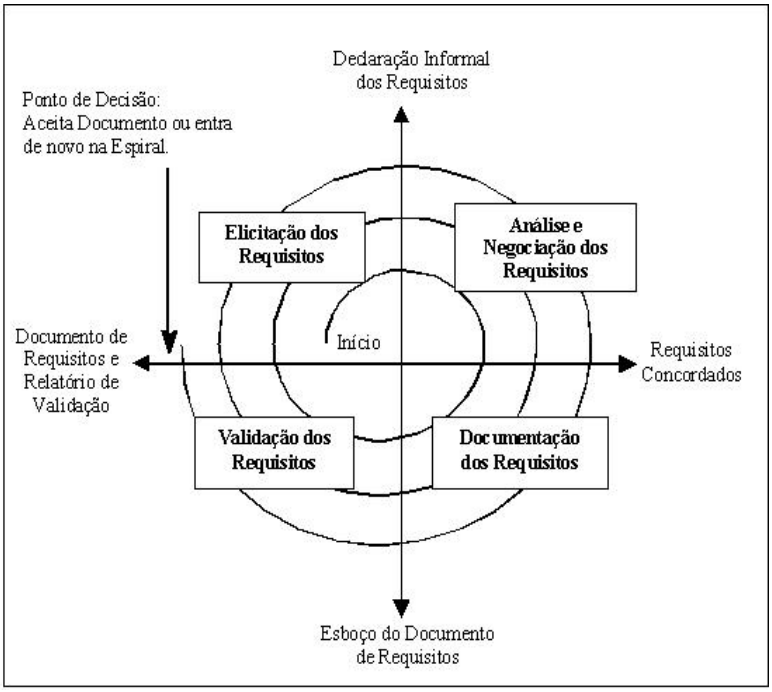


Figura 1. Processo da Engenharia de Requisitos e o Modelo Espiral.

2.4.1 Elicitação de Requisitos

Elicitar requisitos é a atividade relacionada com a identificação dos requisitos do sistema, a partir de consulta aos representantes de cada grupo de usuários; da análise de documentos do domínio em questão; do conhecimento desse domínio; e/ou de pesquisas de mercado. A finalidade geral do processo de elicitação é identificar os fatos que compõem os requisitos do sistema, de forma a prover o mais correto entendimento do que dele é esperado. Outro papel relevante dessa atividade é a descoberta das necessidades das diferentes classes de usuários (SHARP *et al.*, 1999). Aliado a esse processo de descoberta, uma elicitação de

requisitos consistente requer também uma criteriosa análise da organização, do domínio da aplicação e dos processos organizacionais (KOTONYA e SOMMERVILLE, 1997).

Dessa forma, elicitar requisitos apresenta inúmeros desafios. Segundo TORANZO (2002), os principais problemas que afetam a etapa de elicitação podem ser classificados em quatro categorias:

(i) **Problemas de Escopo**. Vinculados ao desconhecimento de informações do sistema relativas à *organização* (restrições, objetivos, metas e expectativas), ao *ambiente organizacional* (fatores sociais, econômicos e políticos), ao *projeto* e à sua *interface computacional* (necessidades/restrições de software e hardware do sistema);

(ii) **Problemas de Entendimento**. Relacionados a dificuldades de comunicação entre desenvolvedores e usuários, e à falta de entendimento dos requisitos informados entre ambas as partes, incluindo problemas humanos relativos à aquisição e disseminação do conhecimento;

(iii) **Problemas Técnicos**. Nessa categoria estão inclusos alguns problemas que afetam o sucesso do processo de elicitação, tais como (*a*) mudanças tecnológicas no software e hardware; (*b*) usuários exigindo sistemas cada vez maiores e complexos; (*c*) mudanças de requisitos com o tempo; (*d*) existência de muitas fontes de informação para os requisitos; (*e*) dificuldades com o reuso de conhecimento pelos analistas;

(iv) **Problemas de Comportamento Humano**. Decorrentes da interação entre as pessoas, envolvendo aspectos tanto individuais quanto coletivos.

Para resolver esses problemas, várias técnicas de elicitação têm sido propostas. A seguir, descrevemos as principais categorias de técnicas de elicitação de requisitos:

- ***Técnicas Tradicionais*** – utilizadas regularmente em várias áreas do conhecimento, como por exemplo a aplicação de questionários, a observação e a análise de documentos (WIERINGA, 1996; MACEDO e LEITE, 1999).

- ***Técnicas de Elicitação de Grupo*** – visam melhor compreender o pensamento e comportamento dos grupos, com o intuito de captar de maneira mais detalhada as necessidades dos usuários. Como principais exemplos dessa categoria, destacamos as técnicas de *Workshop* e *Brainstorming* (LEFFINGWELL e WIDRIG, 2000), as sessões de JAD (*Joint Application Design*) (SOLTYS e CRAWFORD, 1999) e RAD (*Rapid Application Development*) (ROBINSON, 1996).

- ***Prototipação*** – um protótipo é uma versão inicial do sistema que está em fase de desenvolvimento. Em sistemas de hardware, protótipos são usados com freqüência para testar e experimentar os projetos de sistema. Em sistemas de software, protótipos auxiliam na elicitação e validação dos requisitos de sistema (KOTONYA e SOMMERVILLE, 1997). A prototipação é especialmente recomendada quando há um alto grau de incerteza ou quando é necessário um rápido feedback dos usuários (DAVIS, 1992).

- ***Técnicas de Modelagem*** – nos últimos anos, técnicas de modelagem vêm conquistando maior importância para as organizações, desenvolvedores e usuários. Elas apresentam um modelo específico das informações que serão adquiridas, utilizando-o para orientar o processo de elicitação. Exemplos já consagrados na literatura incluem métodos baseados em metas como KAOS (***K****nowledge* ***A****cquisition in aut****O****mated* ***S****pecification*) (VAN LAMSWEERDE *et al.*, 1991) e *i** (YU, 1995); e técnicas baseadas em cenários, como *casos de uso* (JACOBSON, 1992) e CREWS (MAIDEN, 1998), as quais procuram representar as tarefas que os usuários executam normalmente e aquelas que estes desejam executar (LEITE *et al.*, 1997; SCHNEIDER e WINTERS, 1998).

- ***Técnicas Cognitivas*** – originalmente desenvolvidas para a aquisição de conhecimento para sistemas baseados em conhecimento, como por exemplo a análise de protocolo, *laddering*, *card sorting* e *repertory grids* (SHAW e GAINES, 1996).

- ***Técnicas Contextuais –*** surgiram como uma alternativa às técnicas tradicionais e conginitivas. Inclui técnicas como a etnografia e a análise social (VILLER e

SOMMERVILLE, 1999), para as quais o trabalho é uma atividades social que envolve um grupo de pessoas que cooperam para o desenvolvimento de diferentes tarefas. A natureza dessa cooperação é freqüentemente complexa e variante de acordo com as pessoas envolvidas, o ambiente físico e a organização na qual o trabalho acontece. Nesse contexto, técnicas contextuais exploram a *observação* para desenvolver um entendimento completo e detalhado de culturas particulares, como no caso da etnografia, onde uma sociedade ou cultura é observada durante um extenso período, após o qual são apresentadas informações sobre todas as suas práticas constituintes e sua importância no processo produtivo.

- ***Reuso de Requisitos* –** do ponto de vista de requisitos, "reuso" objetiva reutilizar ao máximo o conhecimento existente durante o desenvolvimento do sistema. Existem inúmeras situações onde o reuso de requisitos é recomendado: (*a*) quando os requisitos esclarecem um aspecto do domínio da aplicação, como por exemplo restrições operacionais; (*b*) quando o requisito está relacionado com uma interface comum entre aplicações no mesmo ambiente; (*c*) quando requisitos refletem procedimentos de uso repetido e padrão ao longo do sistema; (*d*) quando os requisitos descrevem políticas da empresa que podem ser reutilizadas em outros sistemas. Apesar das dificuldades inerentes, técnicas como análise de domínio (PRIETO-DIAZ, 1990), clichês (BALZER, 1988) e casos de uso (JACOBSON, 1992) demonstram que o reuso de especificações de requisitos de software pode contribuir para o aumento da produtividade e a unificação de processos internos.

2.4.2 Análise e Negociação de Requisitos

A análise e negociação de requisitos envolve atividades que visam descobrir problemas com os requisitos de sistema e estabelecer um acordo de mudanças que satisfaça todos os afetados. O processo de análise e negociação é caro e demorado porque requer pessoas qualificadas e experientes para dedicar tempo à leitura cuidadosa de documentos e identificação das implicações contidas nas declarações presentes nesses artefatos

(KOTONYA e SOMMERVILLE, 1997). Na maioria das vezes, atividades de análise e negociação de requisitos são executadas de forma paralela ou intercalada, em conjunto com atividades de elicitação de requisitos.

Durante o processo de análise, os requisitos são examinados a fim de detectar omissões, redundâncias, incompletudes e/ou requisitos irreais. A preocupação está em identificar os requisitos que são realmente necessários ao desenvolvimento do sistema e ao atendimento das necessidades dos clientes.

A negociação envolve a discussão dos conflitos encontrados entre requisitos e a busca por uma solução de comum acordo que debele o conflito e atenda aos anseios de todas as partes envolvidas. Os requisitos finais devem exprimir um compromisso que é governado pelas necessidades da organização em geral, os requisitos específicos de diferentes partes interessadas, e restrições de projeto, implementação, prazo e orçamento para o desenvolvimento do software (KOTONYA e SOMMERVILLE, 1997). Para tanto, os modelos de negociação geralmente identificam as principais necessidades dos usuários, priorizando-as a fim de assegurar que os requisitos mais críticos a elas relacionados sejam atendidos o quanto antes. Isso envolve a interação entre gerente de projeto e clientes para o planejamento e organização das versões intermediárias do sistema final.

A seguir são apresentadas as técnicas mais comumente utilizadas no processo de análise e negociação:

- ***Listas de Verificação (Checklists)* –** correspondem a listas de questões que o analista pode fazer uso para avaliar cada requisito. *Checklists* são úteis porque provêem uma referência sobre o que procurar e reduzem as chances de que requisitos importantes deixem de ser checados. Ao final da checagem, pode-se disponibilizar uma lista de inconsistências encontradas, as quais podem ser solucionadas por meio de negociação ou, caso necessário, nova elicitação de requisitos.

- ***Matrizes de Interação –*** utilizadas para denotar a interação entre requisitos e facilitar o processo de análise de possíveis conflitos entre estes. A forma mais simples de construção dessas matrizes é usar uma tabela e rotular suas linhas e colunas com identificadores de requisitos. Valores numéricos indicam a relação entre cada um dos requisitos mapeados, evidenciando conflitos ou sobreposições. Requisitos com altos valores correlacionados devem ser cuidadosamente analisados para avaliar o impacto que essas relações, e eventuais mudanças nas mesmas, possam ter no desenvolvimento do sistema.

- ***Prototipação –*** os protótipos criados na etapa de elicitação podem ser aperfeiçoados na etapa de análise e negociação, possibilitando uma análise mais rica dos requisitos do sistema. Além disso, protótipos contribuem para um maior envolvimento entre as partes interessadas durante as atividades de elicitação, análise e negociação de requisitos.

- ***Reuniões –*** são provavelmente o meio mais eficiente de negociação e resolução de conflitos entre requisitos. Reuniões de negociação são conduzidas em três etapas: (*i*) explicação sobre a natureza dos problemas de requisitos; (*ii*) discussão entre as partes de como resolver os problemas, observando as prioridades estabelecidas; (*iii*) resolução dos conflitos pela tomada de ações corretivas.

2.4.3 Documentação de Requisitos

Nesta fase, os requisitos acordados são anotados num documento que reúne, num nível apropriado de detalhe, o escopo de requisitos que servirá como base para o processo de desenvolvimento do software. O documento de requisitos serve como um contrato entre usuários e desenvolvedores, e deve ser formatado e estruturado de acordo com padrões organizacionais (KOTONYA e SOMMERVILLE, 1997).

De acordo com a norma IEEE (IEEE, 1998), o documento de requisitos deverá possuir declarações não-ambíguas, ser completo, verificável, consistente, modificável, rastreável e utilizável durante todas as fases do ciclo de vida do requisito. Alguns autores conceituam o

documento de requisitos de software como sendo o meio utilizado para descrever as restrições quanto às características do produto e ao processo de desenvolvimento, a interface com outras aplicações, a informação sobre o domínio da aplicação e informações de suporte ao conhecimento do problema. Nesse sentido, o documento de requisitos deve promover o suporte à verificação, validação e gerenciamento do projeto; promover a redução do tempo total e esforço dedicado ao processo de criação do software, evitando o retrabalho quanto à especificação, codificação e testes do sistema; permitir o rastreamento e gerenciamento dos requisitos ao longo da evolução do sistema.

Existem muitas maneiras diferentes de estruturar um documento de requisitos. Este pode ser definido como um único artefato, ou ser formado pela associação de diferentes artefatos que provêem visões de requisitos específicas para cada faceta do desenvolvimento, uma abordagem comum nos *modernos documentos de requisitos de software* (LEFFINGWELL e WIDRIG, 2000). No geral, requisitos são descritos em linguagem natural, e como tal são passíveis de apresentar efeitos colaterais como ambigüidade e generalidade. Para tratar desses efeitos, a definição dos requisitos pode ser complementada por modelos tanto gráficos quanto matemáticos (RUMBAUGH *et al.*, 1991; DELISLE e GARLAN, 1990). Algumas organizações desenvolvem inclusive suas próprias notações para documentação dos requisitos (VAN LAMSWEERDE, 2000a; CYSNEIROS e LEITE, 2001).

KOTONYA e SOMMERVILLE (1997) argumentam que, com o objetivo de garantir que toda informação necessária esteja presente, as organizações devem definir seus próprios padrões de documentação de requisitos. Um exemplo de padrão aceito internacionalmente é o IEEE/ANSI 830-1998 (IEEE, 1998).

2.4.4 Validação de Requisitos

A validação de requisitos é definida como o processo que certifica que o modelo de requisitos gerado esteja consistente com as necessidades e intenções de clientes e usuários. Essa visão da atividade de validação retrata um processo contínuo, ocorrendo na maior parte do tempo em paralelo com as fases de elicitação e especificação (análise, negociação

e documentação). De fato, a validação não é só aplicada ao modelo final de requisitos, mas também a todos os modelos intermediários gerados ao longo do processo de requisitos.

Validar os requisitos significa confirmar que o documento de requisitos e suas especificações complementares reflitem as reais necessidades dos clientes e usuários finais, autenticando os artefatos que servirão de base para as fases subseqüentes do processo de desenvolvimento. Em geral, a validação deve criar os meios necessários para que ocorra um consenso entre as partes envolvidas no projeto, geralmente com objetivos conflitantes. Validação não é, pois, uma tarefa fácil e pode requerer várias sessões de trabalho até que todos encontrem não somente pontos de concordância a respeito dos requisitos, mas principalmente visualizem as implicações futuras de suas decisões. Nesse sentido, a participação de especialistas de domínio contribui sobremaneira para a orientação de clientes, usuários e desenvolvedores na resolução de possíveis impasses.

Os elementos de entrada para a validação de requisitos são o documento de requisitos, diagramas complementares, padrões e conhecimento sobre a organização. Como saída, obtém-se uma lista de requisitos acordados, ações acordadas e documentação revisada. De fato, enquanto fases como *projeto* e *implementação* possuem o documento de requisitos como fonte para verificação de seus resultados, a validação de requisitos não possui uma representação similar que possa ser usada como base. Ou seja, não existe um meio para demonstrar que uma especificação de requisitos esteja correta em relação a outras representações do sistema (KOTONYA e SOMMERVILLE, 1997). Contudo, existe uma variedade de técnicas que podem ser aplicadas para apoiar o processo de validação:

- ***Revisões –*** provavelmente a técnica mais utilizada para validação de requisitos, revisões envolvem um grupo de pessoas lendo e analisando a documentação de requisitos à procura de possíveis problemas. A revisão de requisitos constitui-se de uma reunião formal na qual um engenheiro de requisitos apresenta cada um dos requisitos para crítica e identificação de inconsistências pelo grupo. As inconsistências encontradas são

registradas para posterior discussão. O grupo de revisão deve então tomar decisões que se materializam em ações a serem executadas para corrigir os problemas identificados.

- ***Prototipação –*** se um protótipo foi desenvolvido para fins de elicitação de requisitos, faz sentido usá-lo posteriormente para a validação desses requisitos. Porém, protótipos para validação devem ser mais completos e conter requisitos suficientes para garantir que facilidades projetadas para o sistema estão de acordo com o uso prático esperado por seus usuários. Protótipos de elicitação normalmente apresentam funcionalidades ausentes e podem não contemplar mudanças acordadas durante o processo de análise dos requisitos. É, portanto, necessário dar continuidade ao desenvolvimento do protótipo durante a etapa de validação.

- ***Testes de Requisitos –*** técnicas baseadas em cenário permitem elicitar e analisar requisitos, enquanto facilitam a criação de casos de teste para os cenários identificados (CYSNEIROS, 2002). A execução dos requisitos implementados pode ser simulada para demonstrar que todos os requisitos do sistema tenham sido considerados e estejam de acordo com o esperado. Se dificuldades existem no sentido de derivar casos de teste para um dado requisito, isso implica que algum tipo de problema com a definição do requisito pode existir (KOTONYA e SOMMERVILLE, 1997). O objetivo com os casos de teste, porém, deve ser o de validar os requisitos e não o sistema.

- ***Sistemas Especialistas –*** segundo LOUCOPOULOS e KARAKOSTAS (1995), o processo de validação poderia se beneficiar de ferramentas automatizadas que possuam conhecimento de algum aspecto do processo de engenharia de requisitos para o qual estão capacitadas. Esse aspecto pode ser tanto o conhecimento de um *método* (ex. como aplicar a análise estruturada (YOURDON, 1989) para efetuar engenharia de requisitos), ou o conhecimento sobre um domínio, i.e., conhecimento a respeito do domínio para o qual o software deve ser modelado. Sistemas Especialistas ainda estão restritos ao ambiente acadêmico, contudo é esperado um grande impacto com sua incorporação às próximas gerações de ferramentas CASE.

2.4.5 Gerenciamento de Requisitos

As atividades de Documentação e Validação de requisitos devem formar um ciclo ao longo do qual serão realizadas múltiplas iterações até que a especificação de requisitos seja aprovada numa validação final sem restrições. Paralelamente a essas atividades, deve-se desenvolver o *gerenciamento de requisitos*. Essa atividade consiste em administrar as inevitáveis mudanças dos requisitos propostos. Tais mudanças surgem, principalmente, quando são alteradas as prioridades do negócio, quando se identificam erros ou omissões nos requisitos, ou quando novos requisitos são definidos. Os principais objetivos do gerenciamento de requisitos são: (*i*) gerenciar mudanças nos requisitos acordados; (*ii*) gerenciar as relações entre requisitos; (*iii*) administrar as dependências entre os documentos de requisitos e outros documentos produzidos durante o ciclo de vida do software.

Gerenciamento de requisitos é executado por meio da implementação de *rastreabilidade*. Gerenciamento e rastreamento de requisitos são reconhecidos como importantes pré-requisitos para desenvolver software de alta qualidade (POHL, 1996; GOTEL e FINKELSTEIN, 1994; JARKE, 1998; TORANZO e CASTRO, 1999) e definidos como a habilidade de descrever e seguir a vida de um requisito, em ambas as direções (POHL, 1996; GOTEL e FINKELSTEIN, 1994; RAMESH, 1998). Rastreamento de requisitos é um fator importante para prover com integridade uma documentação completa dos requisitos, assim como ajudar no processo de gerenciamento de mudanças nesses requisitos (TORANZO, 2002).

A área de rastreamento de requisitos situa-se entre duas ilhas: pré e pós-rastreamento, conectadas pela especificação dos requisitos do sistema. *Pré-rastreamento* está relacionado a alguns aspectos da vida do requisito antes da sua inclusão na especificação dos requisitos. *Pós-rastreamento* está relacionado a alguns aspectos da vida do requisito após a sua inclusão na especificação dos requisitos (GOTEL e FINKELSTEIN, 1994). Desse modo, rastreamento de requisitos associa fonte de informação a requisitos, bem como requisitos a artefatos de *design* e vice-versa.

Gerenciar requisitos envolve manipular uma grande quantidade de informação, que deve ser veiculada entre vários indivíduos ao redor da organização (KOTONYA e SOMMERVILLE, 1997). Ferramentas CASE (ASG, 2001) provêm suporte automatizado a esse processo, facilitando o intercâmbio das informações, o armazenamento estruturado dos requisitos coletados, e a publicação dos requisitos analisados via meios corporativos tais como a internet ou intranet. Ferramentas como *Requisitepro®* (RATIONAL, 2001) e *Doors®* (TELELOGIC, 2002) já são uma realidade no ambiente de desenvolvimento das organizações de software e trazem algumas vantagens ao processo de desenvolvimento, dentre elas a integração com ferramentas proprietárias existentes e a montagem de *matrizes de rastreabilidade* a partir do repositório de requisitos.

2.5 Modelagem de Requisitos

O desenvolvimento de uma especificação envolve o mapeamento de fenômenos do mundo-real na forma de símbolos dentro de uma linguagem de especificação. Em particular, a qualidade de uma especificação de requisitos e, em última instância, a qualidade do sistema de informação por ela representado depende em larga escala da habilidade do engenheiro de software em extrair e entender o conhecimento a respeito do domínio sendo modelado, do *universo de discurso*, e do sistema de informação em si mesmo (LOUCOPOULOS e KARAKOSTAS, 1995).

Devido à natureza cognitiva das atividades de engenharia de requisitos, os engenheiros são forçados a capturar grandes volumes de conhecimento a respeito da empresa e do contexto que motiva o desenvolvimento de uma aplicação, os quais são subseqüentemente abstraídos na forma de uma especificação formal. Os formalismos utilizados com esse propósito são conhecidos como *modelos conceituais*. A especificação definida em termos de um modelo conceitual representa abstrações, conclusões e restrições sobre o domínio da aplicação. Essa especificação funciona como um ponto de referência para quaisquer procedimentos de desenvolvimento ou manutenção do sistema-alvo. Por outro lado, ela serve de guia para a condução de novas atividades de análise de domínio, elicitação, especificação e negociação.

Modelos também provêm a base para a documentação e evolução do software (VAN LAMSVEERDE, 2000).

Grande parte das técnicas de modelagem de requisitos tradicionais enfocam a especificação dos requisitos ditos *funcionais*. Uma especificação de requisitos funcionais tem como objetivo a descrição das funções fundamentais que perfazem os componentes de software da aplicação. Uma função é descrita em termos de *entradas*, *saídas* e do *processamento* exigido para que os componentes do sistema transformem os insumos de entrada em resultados (saídas) úteis de acordo com as necessidades do usuário. Uma visão dinâmica das funcionalidades de um sistema deve contemplar aspectos como *controle* (seqüência e paralelismo entre atividades), posicionamento no *tempo* (início e fim de atividades), bem como o *comportamento* do sistema quando da ocorrência de exceções.

McDERMID (1994) afirma, contudo, que quando a especificação funcional torna-se o único ponto focal de um processo de análise de requisitos, então decisões são tomadas ainda na fronteira do sistema antes que o devido entendimento sobre as reais necessidades dos usuários finais seja alcançado. Em outras palavras, uma modelagem puramente funcional tende a desviar a atenção de outros aspectos importantes da especificação do sistema, tais como seus objetivos principais, restrições ao seu desenvolvimento, ou características desejáveis do sistema como performance, segurança, usabilidade, dentre outras. LOUCOPOULOS e KARAKOSTAS (1995) adicionam que só após considerar todos esses aspectos é que o engenheiro consegue entender de forma adequada as razões por trás das necessidades e aspirações daqueles que definem os requisitos. Tal visão oferece um alicerce mais consistente ao desenvolvimento da aplicação, permitindo acomodar dados que possibilitam a resolução de conflitos de requisitos num nível organizacional; atribuição de prioridades para os requisitos coletados; ou até mesmo avaliar cenários alternativos para a satisfação dos requisitos estabelecidos.

Nesse contexto, um consenso existente na comunidade de requisitos (BUBENKO *et al.*, 1994; JACKSON e ZAVE, 1993; LOUCOPOULOS e KATSOULI, 1992; NELLBORN *et*

al., 1992; CHUNG *et al.*, 2000; MYLOPOULOS *et al.*, 2001; CASTRO *et al.*, 2002) é o de que uma especificação de requisitos deve modelar não somente aspectos funcionais da aplicação, mas também aqueles relacionados com a *organização* (requisitos do domínio) onde o sistema deverá operar, além de outros que definem *restrições* a serem impostas ao seu desenvolvimento (requisitos não-funcionais). As seções seguintes discutem as principais abordagens para a modelagem de requisitos organizacionais, funcionais, e não-funcionais.

2.5.1 Modelagem de Requisitos Organizacionais

Uma *organização* pode ser vista como uma estrutura social que inclui desde indivíduos até grupos de indivíduos, ou até mesmo outras organizações (LOUCOPOULOS e KARAKOSTAS, 1995). Todos esses participantes compartilham *recursos* (material e informação) e provêm *serviços* que contribuem para os objetivos gerais da organização. Um modelo de requisitos organizacionais descreve conhecimento e considerações importantes sobre o ambiente para o qual a aplicação se destina. Esse modelo define tipicamente (*a*) estruturas organizacionais; (*b*) objetivos e metas; (*c*) atividades, processos e produtos; (*d*) agentes e papéis de trabalho.

A modelagem organizacional visa descrever os requisitos ilustrativos de um *sistema social* com seus agentes, papéis, metas, responsabilidades e aspectos similares. Um dos problemas da fase de requisitos do ciclo de vida de desenvolvimento de software encontra-se no fato de que nem sempre as tarefas de cada uma das atividades são bem distribuídas entre cada um dos participantes da equipe de desenvolvimento (KOTONYA e SOMMERVILLE, 1997). Modelos organizacionais (CASTRO *et al.*, 2001) definem papéis a serem representados por cada um dos componentes da equipe e as tarefas de cada um dos papéis dentro do processo escolhido.

As primeiras abordagens para inserção de modelos organizacionais numa especificação de requisitos incluem os trabalhos de MacDONALD (1986) e MERCURIO *et al*. (1990). Trabalhos posteriores expandem a visão sobre o modelo organizacional, reconhecendo a

vantagem de se examinar a organização a partir de múltiplas perspectivas diferentes (DOBSON, 1992; NELLBORN *et al.*, 1992; YU e MYLOPOULOS, 1994). Essas abordagens consideram modelos organizacionais como ferramentas adequadas à construção gradual de uma especificação de requisitos funcionais e não-funcionais, e a sua interligação com aspectos gerais da organização. Conceitos como *dependência de objetivos* e *metas* fornecem uma ligação entre requisitos estratégicos de alto nível e requisitos operacionais e provêm oportunidades para a análise de correspondência entre expectativas de alto nível e os processos atuais (ANTON *et al.*, 1994).

Em particular, a necessidade de uma análise das metas (*goals*) durante o desenvolvimento do sistema é amplamente aceita na comunidade acadêmica (MITTERMEIR *et al.*, 1990; DARDENNE *et al.*, 1993; YU e MYLOPOULOS, 1994; CHUNG *et al.*, 2001). Alguns autores (LOUCOPOULOS e KARAKOSTAS, 1995; CASTRO *et al.*, 2001; VAN LAMSWEERDE, 2001) afirmam que desenvolvedores de software devem entender não somente o que estão desenvolvendo, mas também o propósito do sistema em questão. Nesse contexto, um sistema é considerado como a realização de um conjunto de metas organizacionais. A rede de metas interligadas gerada deriva uma infinidade de alternativas de implementação. Essas alternativas devem ser elaboradas de forma a determinar o grau com o qual um conjunto de metas dá suporte a uma dada operacionalização. Um exemplo dessa abordagem aplicada à indústria é reportado em (POTTS, 1994).

Em resumo, a modelagem de metas organizacionais é importante por várias razões: (*a*) leva à incorporação de componentes de requisitos de acordo com metas de negócio da empresa; (*b*) justifica e explica a presença desses componentes; (*c*) pode ser usado para atribuir responsabilidades a agentes do sistema e compromissos entre partes interessadas com relação à aplicação; (*d*) fornece uma base para a detecção e resolução de conflitos que surgem a partir dos múltiplos pontos de vista entre agentes humanos (DARDENNE *et al.*, 1993). Várias técnicas têm sido propostas com o intuito de modelar ambientes organizacionais, incorporando as diretrizes anteriores. Dentre as principais estratégias,

podemos citar *i** (YU, 1995), BUBENKO e WANGLER (1993), KAOS (VAN LAMSWEERDE *et al.*, 1991) e TROPOS (CASTRO *et al.*, 2001).

Embora esta dissertação não utilize diretamente nenhuma das técnicas anteriormente descritas, a metodologia proposta incorpora algumas das preocupações da análise organizacional ao promover (*a*) a vinculação entre necessidades analíticas do cliente e as funcionalidades que a aplicação *data warehouse* deve implementar; (*b*) ao enfocar o estabelecimento de papéis, responsabilidades e prioridades durante o desenvolvimento dos aspectos funcionais e não-funcionais da aplicação; e (*c*) ao utilizar uma abordagem *goal-oriented* (orientada a metas) para a resolução de problemas de projeto multidimensional no contexto da organização, descrita em detalhes no capítulo 4.

2.5.2 Modelagem de Requisitos Funcionais

As abordagens iniciais para modelagem dos requisitos de software eram primordialmente dirigidas à identificação e descrição do conteúdo e estrutura do sistema, i.e., seus *requisitos funcionais*. Ainda na década de 70, a modelagem de requisitos funcionais era dominada pelos chamados métodos de desenvolvimento estruturados. ROSS e SCHOMAN (1977a) foram precursores nesse campo ao descrever em seu artigo de forma elegante o papel das metas (*goals*), pontos de vista (*viewpoints*) (EASTERBROOK, 1994), dados, operações, agentes e recursos como elementos potenciais de uma *ontologia* para a Engenharia de Requisitos. Em trabalho seguinte, os mesmos autores introduziram a técnica de modelagem SADT – *Structured Analysis and Design Technique* (ROSS e SCHOMAN, 1977b) tendo como base a formalização de uma declaração do sistema por meio de múltiplos modelos interligados, cada um representando uma visão dos dados e operações que compõem o sistema e como esses elementos estão relacionados.

Outras técnicas semi-informais foram desenvolvidas no final dos anos 70, notadamente, diagramas de Entidade-Relacionamento (CHEN, 1976) para a modelagem de dados, análise estruturada (DeMARCO, 1978) para a modelagem de operações passo a passo, e diagramas de transição de estado (WASSERMAN, 1979) para a modelagem das interações com o

usuário. Essas técnicas ganharam popularidade pela simplicidade e foco eficiente em aspectos individuais da modelagem do sistema. Em contrapartida, sofriam de escopo e expressividade limitados devido ao pouco suporte ontológico e a facilidades de estruturação limitadas (VAN LAMSWEERDE, 2000).

A linguagem de modelagem de requisitos RML (GREENSPAN *et al.*, 1994) introduziu conceitos semanticamente mais ricos como *generalização*, *agregação* e *classificação*, servindo de embrião para as primeiras abordagens orientadas a objeto. Várias abordagens para engenharia de requisitos baseadas em conceitos de orientação a objetos foram propostas em seguida (COLBERT, 1989; COAD e YOURDON, 1989; RUMBAUGH *et al.*, 1991; JACOBSON, 1992). O paradigma orientado a objetos trouxe uma série de benefícios à modelagem de requisitos funcionais tais como modularidade, encapsulamento, abstração de aspectos de alto nível e reusabilidade. O aperfeiçoamento das técnicas orientadas a objeto deu origem à OMT (*Object Modelling Technique*) (RUMBAUGH *et al.*, 1991) e posteriormente ao método de Jacobson (JACOBSON, 1992), hoje a abordagem mais difundida para especificação de um modelo de requisitos de sistema.

JACOBSON (1992) descreve dois modelos em suporte à fase de definição de requisitos. O *modelo de requisitos*, idealizado para capturar os requisitos do usuário pela especificação de todo o escopo de funcionalidades que a aplicação deve executar, em colaboração direta com o usuário. E o *modelo de análise*, que constitui a base para a estrutura do sistema, especificando todos os objetos lógicos a serem incluídos e seus relacionamentos.

O modelo de requisitos de JACOBSON (1992) consiste de casos de uso, descrições de interface e modelos de domínio de problema. O modelo de casos de uso utiliza *atores* para modelar o que existe externamente ao sistema e casos de uso para representar as operações a serem executadas pelo sistema (Figura 2). Um caso de uso descreve uma seqüência de ações que um sistema executa com o intuito de produzir um resultado de valor para um ator particular (LEFFINGWELL e WIDRIG, 2000). Atores são diferenciados de usuários, os quais são pessoas que usam o sistema, enquanto atores representam *papéis* que esses

usuários devem desempenhar. Atores interagem com o sistema por meio da seqüência de passos relatada nas especificações de casos de uso.

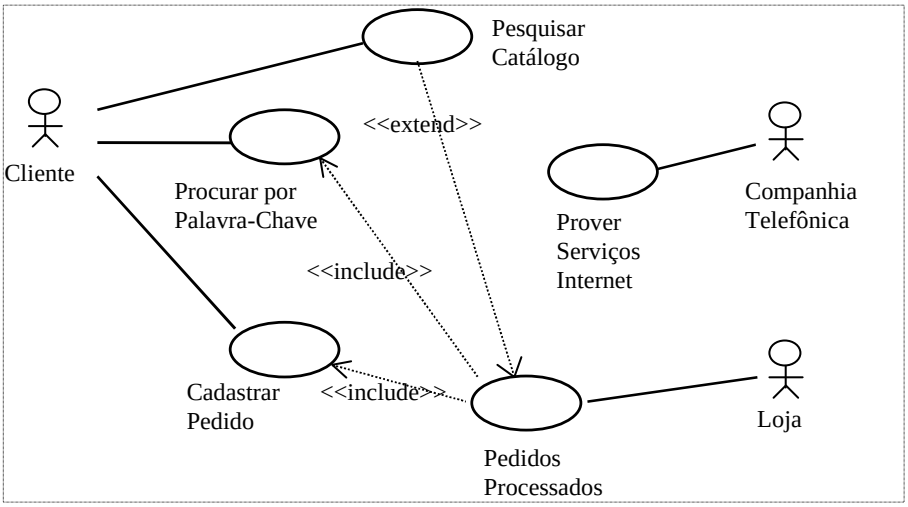


Figura 2. Exemplo de Modelo de Casos de Uso de Jacobson.

Um dado caso de uso pode representar diferentes *cenários* de execução do sistema. Cenários estão para casos de uso assim como instâncias estão para classes, significando que um cenário é basicamente uma instância de um caso de uso. Vários caminhos (cenários) podem ser seguidos dependendo do contexto na execução do sistema. Considera-se que o caminho básico para realizar um caso de uso, sem problemas e sem erros em nenhum dos passos da seqüência, é denominado de **cenário primário**. Neste tipo de cenário, a execução dos passos para realizar a funcionalidade básica do caso de uso, é obtida com sucesso. Por outro lado, caminhos alternativos, bem como situações de erro, podem ser representados através de **cenários secundários** ou **alternativos**. Cenários secundários descrevem seqüências alternativas e de erros que podem ocorrer em um cenário primário associado com um caso de uso. Cenários secundários podem ser descritos separadamente ou como extensão da descrição de um cenário primário.

Diagramas de casos de uso são parte representados na Linguagem de Modelagem Unificada ou UML (*Unified Modeling Language*) (BOOCH *et al.*, 1999). A UML oferece ainda outros recursos para refinar fluxos de eventos em casos de uso. A idéia consiste

basicamente em incluir relacionamentos que permitam descrever diversos aspectos de comportamento entre casos de uso. Os relacionamentos permitidos pela UML incluem:

- relacionamento do tipo **<<*include*>>:** quando for detectado no sistema um conjunto de **passos comuns** a vários casos de uso, pode-se criar um caso de uso com estes passos, com potencial para ser reutilizado por outros casos de uso. A idéia consiste em abstrair em um **caso de uso específico**, um comportamento comum a vários casos de uso, podendo os demais casos de uso fazer uso do mesmo (i.e. incluí-lo) quando necessário.

- relacionamento do tipo <<***extend***>>: utilizado quando existe uma **seqüência opcional** ou **condicional** de passos que se quer incluir em um caso de uso. Esta seqüência de passos deve ser descrita em um caso de uso específico, que poderá ser utilizado por outros casos de uso em certo ponto de sua execução.

- relacionamento do tipo **<<*generalization*>>:** generalização entre casos de uso tem o mesmo significado de generalização entre classes na orientação a objetos. Isto significa que um caso de uso "filho" **herda** o comportamento e estrutura do caso de uso "pai". Considera-se que um caso de uso "filho" é uma **especialização** do caso de uso "pai", podendo adicionar nova estrutura e comportamento este. Os passos comuns a ambos os casos de uso são definidos, assim, no caso de uso "pai", enquanto o caso de uso "filho" detalha os passos variantes que descrevem o caminho alternativo.

Estudos recentes têm demonstrado a importância do uso de técnicas baseadas em casos de uso e cenários para a especificação de requisitos de sistema (WEIDENHAUPT *et al.*, 1998; JARKE e KURKI-SUONIO, 1998). Adicionalmente, modelos baseados em casos de uso demonstram-se mais eficazes que questionários e entrevistas na resolução de problemas de comunicação e transferência de conhecimentos entre desenvolvedores, devido à proximidade conseguida com as sessões de "relato de histórias" promovidas pelos casos de uso. Outro progresso importante é a possibilidade de se integrar modelagem funcional com modelagem organizacional (SANTANDER, 2002).

2.5.3 Modelagem de Requisitos Não-Funcionais

Requisitos Não-Funcionais (RNF) desempenham um papel crucial no projeto e desenvolvimento de um sistema de informação. Erros, omissões ou falhas em tratar de maneira apropriada esses tipos de requisitos têm sido apontados entre os problemas mais caros e de maior dificuldade de correção uma vez que o sistema tenha sido finalizado (LOUCOPOULOS e KARAKOSTAS, 1995).

De forma irrefutável, requisitos não-funcionais têm uma grande influência na qualidade do software a ser desenvolvido (FICKAS, 1987; DARDENNE *et al.*, 1991; JARKE *et al.*, 1993). Contudo, apesar dessa importância, a maioria dos métodos de especificação de requisitos é dominada por abordagens centradas na modelagem de requisitos funcionais. Por trás desse fato reside a natureza informal de representação desse tipo de requisito. A despeito dos avanços da engenharia de requisitos no campo da criação de modelos de requisitos de sistema, a tecnologia corrente ainda não está totalmente amadurecida para superar essa informalidade e representar adequadamente requisitos de natureza não-funcional. As abordagens mais nem sucedidas nesse sentido trabalham com o princípio de que requisitos não-funcionais devem ser trabalhados de forma a satisfazer duas premissas fundamentais: eles devem ser os mais "*objetivos*" e *testáveis* possíveis. Um requisito não-funcional é dito "objetivo" se expressa um desejo, meta ou opinião pessoal do usuário, e é testável se existe algum processo que consiga aplicar condições de validação para o mesmo.

O tratamento preciso e rigoroso de requisitos não-funcionais é caracterizado por duas abordagens principais (MYLOPOULOS *et al.*, 1992):

- **Orientada a Produto:** considera requisitos não-funcionais do ponto de vista de avaliação do produto final. Essa abordagem busca elaborar definições formais dos requisitos não-funcionais de forma que o sistema possa ser avaliado quanto ao grau de atendimento a esses requisitos. A avaliação ocorre ao final do processo de desenvolvimento;

- **Orientada a Processo:** essa abordagem procura desenvolver técnicas para justificar decisões de projeto durante o ciclo de desenvolvimento. Ao invés de avaliar o produto final, a abordagem orientada a processo tenta racionalizar o seu desenvolvimento em termos dos requisitos não-funcionais que devem ser atendidos. Decisões de projeto podem afetar positiva ou negativamente os requisitos não-funcionais. Essas relações de dependência servem de base para que o projetista do sistema escolha dentre alternativas de desenvolvimento que melhor satisfazem os requisitos não-funcionais impostos.

Métodos orientados a produto surgiram no início dos anos 90 (KELLER *et al.*, 1990) com um enfoque *quantitativo* sobre requisitos de qualidade. Todavia, as técnicas derivadas da abordagem orientada a processo e inspiradas em princípios do raciocínio *qualitativo* (KUIPERS, 1995) conquistaram um lugar de destaque como alternativas mais econômicas e complementares aos casos de requisitos de alto nível não facilmente formalizáveis (VAN LAMSWEERDE, 2000). Uma abordagem qualitativa de destaque é o ***Framework NFR*** (CHUNG *et al.*, 2000) para captura e análise sistemática de requisitos não-funcionais. Durante o processo de desenvolvimento, o *framework* trata requisitos não-funcionais como metas (*goals*) a serem atendidas. Metas se interelacionam de forma sinergística ou conflitante. O *framework* explora as dependências entre metas para (*a*) auxiliar na seleção de alternativas de projeto aderentes a aspectos não-funcionais estabelecidos; (*b*) analisar os compromissos entre alternativas; (*c*) vincular decisões de projeto a requisitos não-funcionais; (*d*) justificar as decisões tomadas; (*e*) e detectar defeitos de projeto (CHUNG e NIXON, 1995).

Um conceito central no *framework* é a noção de *softgoals*[1]. Um *softgoal* representa uma meta que não possui uma definição precisa e/ou critério de satisfação determinado (*ex.* "*o sistema deve possuir uma interface amigável*"). *Softgoals* exercem influência positiva ou

[1] O termo *softgoal* não possui uma tradução direta para o português. Numa tradução livre poderíamos descrevê-lo como uma "meta solta" ou "meta imprecisa". Para manter a integridade com a notação proposta por CHUNG *et al.* (2000), preferimos utilizar como referência o termo original em inglês.

negativa entre si (*ex*. o uso de confirmações sucessivas de senha durante a execução de uma operação bancária aumenta a *confiabilidade* do sistema, mas diminui a *amigabilidade* da interface). Quando existe evidência suficientemente positiva a favor e pouca evidência negativa contra um *softgoal*, diz-se que este é satisfeito (ou "*satisficed*" como descrito em CHUNG *et al.* (2000) - *ex*. clientes do banco que utilizam sua interface Web preferem confirmar mais vezes suas senhas desde que isso garanta a confiabilidade da operação, logo, confirmações sucessivas de senha satisfazem a meta "*confiabilidade*", a despeito da perda em *amigabilidade*). Essas interdependências podem ser investigadas em termos da construção, análise e revisão incremental e interativa de diagramas SIG - *Softgoal Interdependency Graphs* (Gráfico de Interdependência de *Softgoals*) (Figura 3), de acordo com o ciclo a seguir: (*i*) desenvolver as metas (*goals*) e sua decomposição em hierarquias de *softgoals* do tipo AND/OR; (*ii*) definir catálogos de técnicas de desenvolvimento denominadas *métodos de operacionalização*, relativos a um dado requisito não-funcional; (*iii*) identificar correlações entre *softgoals* e identificar seus graus de criticidade; (*iv*) analisar compromissos em alternativas de projeto; (*v*) identificar um cenário operacional que melhor satisfaz os requisitos de qualidade; (*vi*) relacionar a alternativa de projeto encontrada com requisitos funcionais no sistema-alvo.

Nuvens finas num SIG denotam *softgoals*, os quais são descritos pela seguinte notação: *Tipo [Tópico1, Tópico2, ...]*. *Tipo* corresponde a um aspecto não-funcional (ex. *segurança*) e *Tópico* identifica um assunto dentro do sistema-alvo ao qual o *softgoal* está associado (ex. *contas_bancárias*). Tópicos podem ser decompostos em atributos, indicados por um "." em seguida à descrição do tópico (ex. *contas_bancárias.saldo_médio*). Nuvens com a borda grossa indicam uma operacionalização de *softgoal*. As linhas entre nuvens grossas e finas representam o grau em que a operacionalização do *softgoal* (grossa) satisfaz um requisito não-funcional (nuvem fina). O grau de intensidade da satisfação é denotado pelos sinais "+" e "-" ao lado das linhas. Exclamações, por sua vez, indicam a criticidade do *goal*.

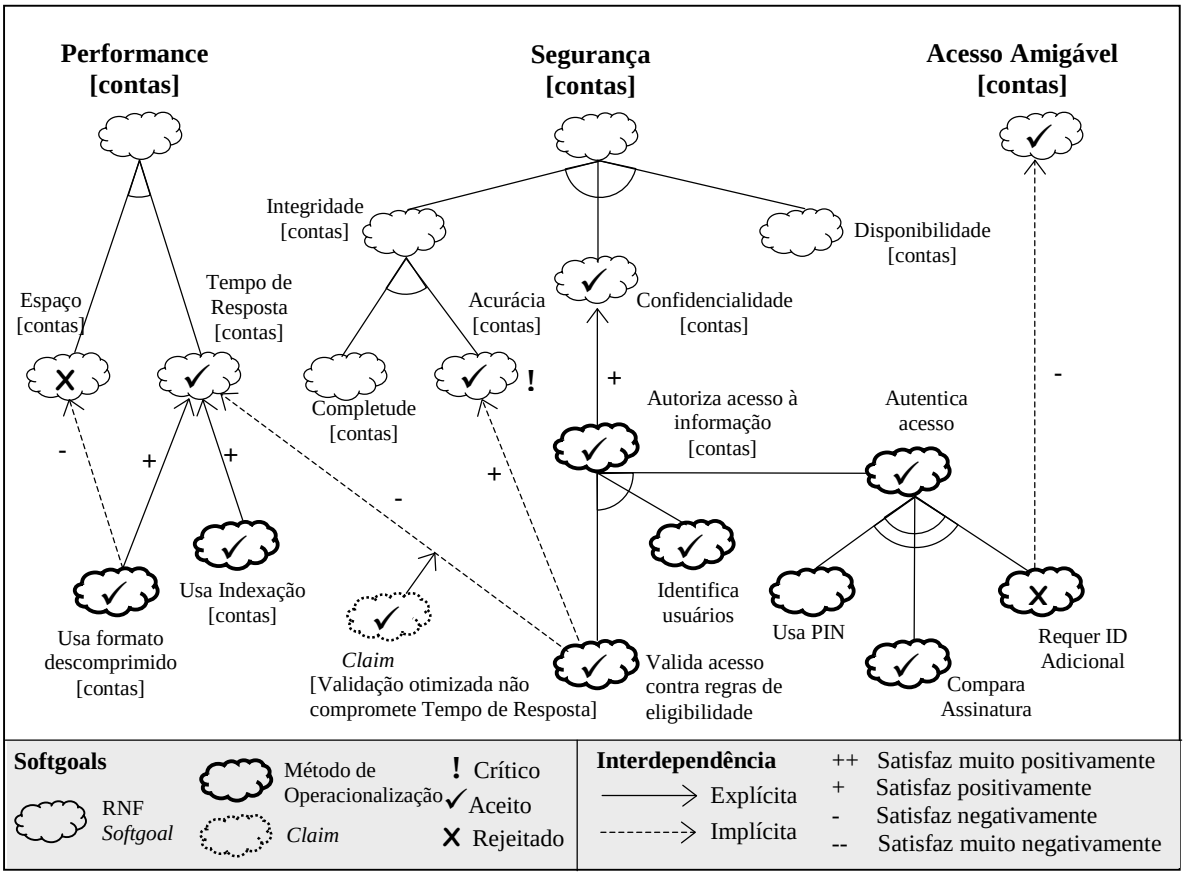


Figura 3. Exemplo de Gráfico de Interdependência de *Softgoals*[2].

Arcos simples representam a união (AND) de todos os *softgoals* originados de um outro *softgoal*, enquanto arcos duplos representam uma seleção (OU) entre *softgoals*. Os símbolos "✓" e "**x**" são usados para indicar a seleção (aceitação) ou negação (rejeição) de um *softgoal*[3]. Por fim, nuvens tracejadas representam uma justificativa denominada *claims*, e apontam o raciocínio utilizado no processo de aceitação/rejeição de um *softgoal* baseado nas características de domínio tais como prioridades e volume de trabalho (SUBRAMANIAN e CHUNG, 2001).

[2] Adaptado de (CHUNG *et al*, 2000).

[3] A notação mais recente no *Framework NFR* utiliza valores numéricos (métricas) dentro das nuvens de *softgoals* para registrar o grau pelo qual uma solução arquitetônica gerada satisfaz os vários requisitos não-funcionais. Por razões de simplicidade, adotamos nesse trabalho a notação original.

A Figura 3 demonstra esses conceitos. No topo da figura, *softgoals* (nuvens finas) representam requisitos não-funcionais de mais alto nível num sistema de controle de contas bancárias (*Segurança*, *Performance* e *Acesso Amigável*). Os tópicos entre colchetes identificam o assunto sendo investigado à luz desses requisitos, i.e., as **contas bancárias**. Para facilitar a investigação, os *softgoals* são decompostos em requisitos menores (ex. *Acurácia[contas]*). As nuvens grossas denotam os métodos que operacionalizam esses requisitos e as setas o grau de influência. No exemplo, o uso de *indexação* para identificar as contas satisfaz positivamente o requisito *Tempo de Resposta*. Conflitos podem surgir da influência mútua entre métodos e requisitos não-funcionais. Por exemplo, o uso de *formato descomprimido* representa explicitamente um ganho para o tempo de resposta no acesso às contas, mas implicitamente influencia de forma negativa no aproveitamento de espaço em disco. Esse conflitos podem ser resolvidos com a análise do gráfico e determinação de um raciocínio para aceitação/rejeição de um *softgoal* em detrimento de outro. Por exemplo, a contribuição do *uso de formato descomprimido* e *indexação* para o *Tempo de Resposta* é mais positiva que a influência negativa (implícita) que o formato descomprimido exerce sobre o softgoal *Espaço[contas]*, o que faz com que haja a aceitação do *Tempo de Resposta* e a conseqüente rejeição de *Espaço[contas]*. Raciocínio similar pode ser aplicado ao *softgoal* "*Validar acesso contra regras*". Baseado na sua influência positiva para o requisito crítico *acurácia* das contas e na justificativa (*claim*) de que uma validação otimizada não compromete o tempo de resposta, temos que o *softgoal* é aceito em detrimento de sua ação possivelmente negativa sobre o *Tempo de Resposta*.

O *Framework NFR* tem sido usado numa variedade de domínios para modelar requisitos não-funcionais (CHUNG e NIXON, 1995; SUBRAMANIAN e CHUNG, 2001; CHUNG *et al.*, 1994; JARKE *et al.*, 1993; ALVES, 2001). Em (PAIM e CASTRO, 2002a), estendemos o *Framework NFR* para permitir o tratamento de aspectos não-funcionais em aplicações *data warehouse*. Um catálogo de Tipos RNF e métodos de operacionalização especificamente definidos para o domínio *data warehouse* é reutilizado durante a especificação dos requisitos para investigar alternativas de projeto multidimensional. Essa abordagem e sua aplicação são descritas em detalhes nos capítulos 4 e 5 respectivamente.

2.6 Considerações Finais

Os últimos 25 anos têm evidenciado um grande esforço por parte das organizações de software e da comunidade acadêmica no sentido de intensificar a adoção e a melhoria de processos para uma engenharia de requisitos de alta-qualidade. Embora difícil, essa tarefa revela-se cada vez mais crítica para o sucesso dos projetos de software, o que vem elevar a importância das técnicas, conceitos e processos da Engenharia de Requisitos para o perfeito entendimento das necessidades dos clientes e conseqüente derivação de um produto de software de qualidade.

Nesse capítulo, apresentamos os benefícios da aplicação desses mecanismos para uma correta aquisição, análise, especificação e gerência dos requisitos do sistema. Não obstante, a inserção desses mecanismos no processo de desenvolvimento de sistemas *data warehouse* ainda se configura uma lacuna a ser preenchida. Conforme será demonstrado no capítulo seguinte, a maioria das abordagens para o desenvolvimento de aplicações em suporte a processos de decisão ainda coloca em primeiro plano aspectos ligados à implementação do repositório de dados, em detrimento de uma visualização das metas do tomador de decisão, e sua tradução subseqüente em requisitos de sistema que, num último estágio, determinam as características analíticas do *data warehouse*. Com essa motivação em mente, iremos mostrar, ao longo dessa tese, que a integração entre engenharia de requisitos e o desenvolvimento de aplicações *data warehouse* é uma realidade factível e, acima de tudo, essencial para o sucesso do projeto.

Capítulo 3

Sistemas Data Warehouse

Este capítulo apresenta a tecnologia de Data Warehousing. Inicialmente, conceituamos "data warehouse", mostrando sua inserção no processo de data warehousing. O processo é então detalhado, juntamente com os principais conceitos relativos ao paradigma mutidimensional de suporte à decisão. Em seguida, estabelecemos um comparativo entre sistemas de suporte à decisão e sistemas transacionais, ao qual unimos uma análise das metodologias atuais para desenvolvimento de sistemas data warehouse, como motivação para a necessidade de uma especificação de requisitos adequada a essas aplicações.

3.1 Introdução

Surgida no início dos anos 90, a tecnologia de *Data Warehousing* (CHAUDHURI e DAYAL, 1997) foi proposta como uma solução genérica para suprir a necessidade de informações gerenciais das organizações (MOODY e KORTINK, 2000). Desde então, tomou de assalto a indústria de informática, impulsionada pelas enormes possibilidades de extração de informações estratégicas, a partir de dados "escondidos" em sistemas computacionais cotidianos. O sucesso da solução pode ser comprovado pelo aumento dos investimentos em projetos *data warehouse* ao redor do mundo, cuja cifra saltou de 4,5 bilhões de dólares em 1996 (BOEHNLEIN e ULBRICH-VOM ENDE, 1999) para uma previsão de 52 bilhões de dólares em 2002 (WATTERSON, 1998).

Data Warehousing requer uma mudança considerável nas relações entre desenvolvedores e usuários, ao promover a construção de aplicações baseadas num modelo *self-service*, em contraste com o modelo *orientado-a-relatório* tradicional (MOODY e KORTINK, 2000). Num ambiente *data warehouse*, usuários finais acessam dados e criam seus próprios relatórios diretamente, por meio de ferramentas de consulta amigáveis que implementam pesquisas *ad-hoc* a uma base (*warehouse*) organizacional consolidada. A independência de soluções específicas, pouco flexíveis para o processamento de informações gerenciais, é um dos grandes atrativos por trás dos investimentos em *data warehouse*. Essas características de independência e flexibilidade, contudo, escondem um complexo processo de extração, tratamento e carga de dados operacionais, que pode envolver centenas de bases mantidas por dezenas de sistemas provedores.

De fato, o desenvolvimento de sistemas *data warehouse* é bastante diferente do desenvolvimento dos sistemas convencionais que fornecem dados para o repositório central. O primeiro não envolve somente os *requisitos* de informação dos tomadores de decisão, mas também a estrutura e requisitos alocados dos demais. Ambas as visões operacional e estratégica têm de ser unidas num mesmo pacote multidimensional, em atendimento a requisitos de análise corporativos (PAIM e CASTRO, 2002a). Nesse

contexto, tanto os requisitos do *data warehouse* quanto aqueles alocados aos sistemas provedores exercem uma influência dinâmica no desenvolvimento da aplicação, ilustrada pelas possíveis mudanças nos requisitos do usuário e por variações na estrutura das fontes de dados.

Para melhor entender a importância e as peculiaridades da definição de requisitos em sistemas *data warehouse*, esse capítulo apresenta uma visão geral da tecnologia de *Data Warehousing*. Inicialmente, conceituamos o que sejam *data warehouses* e sua inserção no processo de *data warehousing* (seção 3.2). Os principais conceitos relativos à modelagem de sistemas para suporte à decisão são definidos na seção 3.3. O processo de *Data Warehousing* é então detalhado na seção 3.4. A seção 3.5 traça um comparativo entre sistemas de suporte à decisão e sistemas transacionais convencionais. Em seguida, a seção 3.6 examina as metodologias atuais para desenvolvimento de sistemas *data warehouse* à luz de sua contribuição para uma especificação de requisitos eficiente. Ao final do capítulo, a seção 3.6.6 resume nossas considerações sobre a necessidade de um processo de requisitos em suporte ao desenvolvimento de sistemas *data warehouse*.

3.2 Data Warehouse e Data Warehousing

O termo *Data Warehouse* foi inicialmente usado por DEVLIN e MURPHY (1988), mas foi INMON (1996) quem se consagrou como o pai do *data warehouse* ao criar a definição mais aceita na literatura:

> *"Data Warehouse é uma coleção de dados orientada a assunto, integrada, não-volátil, e variante no tempo em suporte a decisões gerenciais"*.

O dado é organizado por **assuntos**, tais como "produtos eletrônicos", permitindo que usuários trabalhem com termos do seu dia-a-dia. Em aplicações convencionais, o dado é organizado para atender a uma funcionalidade específica (ex. *análises químicas*), tornando-o muitas vezes incompreensível para usuários pouco especializados. A orientação a assunto confere a aplicações *data warehouse* a flexibilidade requerida para a análise gerencial dos

dados, ao passo que possibilita a sua estruturação de acordo com as áreas de atuação e objetivos estratégicos da organização.

Sistemas *data warehouse* **integram** dados provenientes de bases localizadas nos diversos sistemas operacionais da organização, provendo uma visão consistente e unificada do cenário operacional. Para tanto, o dado é extraído do sistema provedor, traduzido para um formato adequado às análises estratégicas, e tratado para eliminar inconsistências, incompatibilidades e ausência de conteúdo essencial. Esse processo torna a tarefa de combinar as diferentes informações para análise muito mais fácil para o usuário final.

O dado é dito **não-volátil** pois, ao contrário de aplicações comuns, é mantido na base por muitos anos, e normalmente nunca é excluído. A retenção do dado torna possível a consulta histórica sobre informações armazenadas em longos períodos de tempo, subsidiando, dessa forma, a análise de tendências a partir do estudo de séries complexas de dados.

Em *data warehouses* o dado é **variante no tempo**, i.e., é armazenado conforme suas inúmeras variações ao longo de uma linha de tempo, de forma que consultas podem ser executadas para recuperar o estado da informação num dado período de tempo. Nesse sentido, o dado é regularmente inserido para manter o registro histórico das operações de toda a organização ao longo de sua existência.

O desafio de sistemas *data warehouse* é transformar o dado operacional, distribuído heterogeneamente pela organização, em informação estratégica agregada, para suporte a processos de tomada de decisão que garantam a competitividade requerida pelo negócio. Para esse fim, a construção de um *data warehouse* está apoiada num processo sistemático que compreende algoritmos, ferramentas, técnicas e uma arquitetura especialmente concebida para facilitar o armazenamento de grandes volumes de dados e viabilizar a obtenção de informação estratégica a partir da execução de consultas complexas ao repositório, dentro de aspectos restritos de tempo de resposta e vazão (*throughput*). Esse processo sistemático é denominado *Data Warehousing* e os sistemas dele resultantes

comumente referenciados como **Sistemas de Suporte à Decisão** (SSD), ou simplesmente **Sistemas Data Warehouse**[4].

3.3 O Paradigma Multidimensional

Devido à não-volatilidade do dado e à complexidade das consultas *ad-hoc*, o projeto de sistemas *data warehouse* é fortemente influenciado pela modelagem do **dado**, ao contrário de aplicações tradicionais onde o foco reside na definição de um modelo de **transações** eficiente. MOODY e KORTINK (2000) argumentam, contudo, que as técnicas tradicionais de modelagem de dados são inadequadas ao projeto de ambientes *data warehouse* pelas seguintes razões:

(1) A estrutura de banco de dados resultante é muito complexa para que usuários finais possam compreender. No melhor dos casos, o modelo gerado é composto por uma centena de tabelas interligadas por uma complexa rede de relacionamentos. Até mesmo consultas simples requerem o estabelecimento de *joins* (junções) entre múltiplas tabelas, o que se constitui num mecanismo propenso a erro e inapropriado ao acesso eficiente dos grandes volumes de dados armazenados no *warehouse*;

(2) Modelos tradicionais enfatizam a *normalização* (CODD, 1972) como forma de evitar dados redundantes e conseqüentes anomalias de atualização. Em ambientes *data warehouse*, porém, o dado não é atualizado e sua redundância demonstra ser, ao contrário, um artifício para maximizar a eficiência das consultas ao repositório (KIMBALL, 1998).

[4] O termo *data warehouse* é usado concomitantemente na literatura para designar ora o banco de dados de aplicações de suporte à decisão, ora o produto (sistema) gerado como resultado de um processo de *Data Warehousing*. No contexto da presente dissertação, adotaremos esse último uso do termo para referenciar as aplicações de suporte à decisão. O banco de dados, quando assim não expressamente referenciado, será citado como *repositório* ou ainda '*warehouse*'.

De fato, projetar sistemas *data warehouse* requer o desenvolvimento de técnicas completamente diferentes das adotadas no projeto de sistemas tradicionais (GOLFARELLI e RIZZI, 1999). O Paradigma Multidimensional (AGRAWAL *et al.*, 1995) foi criado para permitir a modelagem de esquemas lógicos mais adequados ao processamento analítico típico de aplicações *data warehouse*. A idéia básica por trás de modelos multidimensionais é a separação entre dados quantitativos e qualitativos (SHOSHANI, 1982), por meio de um conceito conhecido como **multidimensionalidade** (ABELLÓ *et al.*, 2000; BREITNER, 1997; LEHNER *et al.*, 1998), no qual a informação é classificada de acordo com fatos e dimensões. *Fatos* são dados quantitativos mensuráveis que representam uma atividade específica de negócio que se deseja analisar (ex. *movimentações financeiras*). Fatos podem ser analisados sob vários pontos de vista ou perspectivas diferentes, com base em aspectos qualitativos associados ao seu conteúdo (ex. *movimentações financeiras dos clientes por agência e período*). A combinação de aspectos qualitativos interelacionados dentro de um mesmo ponto de vista é denominada *Dimensão* (ex. a hierarquia conta→agência→banco pode constituir a dimensão "Cliente"). Esse arranjo dos dados dá origem a uma estrutura *n*-dimensional referenciada na literatura como *hipercubo* ou simplesmente *cubo de dados* (CHAUDHURI e DAYAL, 1997), conforme ilustrado pela Figura 4.

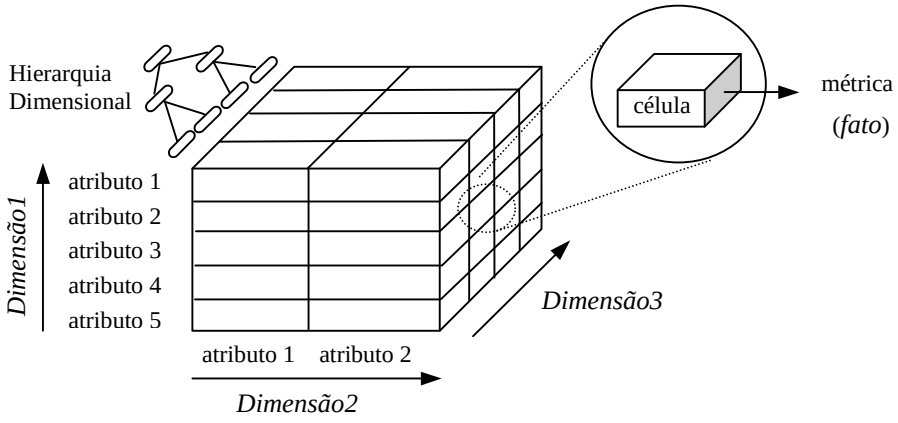


Figura 4. Cubo (*Hipercubo*) de Dados Multidimensional
(BOEHNLEIN e ULBRICH-VOM ENDE, 1999).

Cada dimensão consiste de uma série de elementos ou *atributos* (características) que a descrevem em níveis de detalhe diferenciados, dispostos numa organização hierárquica (*hierarquia dimensional*), através da qual valores numéricos (*métricas*) representativos de fatos podem ser agregados ou classificados. Os níveis dentro da hierarquia conferem uma *granularidade*, i.e., um nível de detalhe maior ou menor ao fato. A interseção entre atributos de um cubo dimensional forma uma *célula*, a qual guarda uma métrica quantitativa de um fato. Por exemplo, numa cadeia de lojas de departamento, a **venda de produtos** é um *fato*, que pode ser analisado sob a perspectiva das lojas que efetuaram vendas ao longo do ano. **Produto**, **loja**, e **tempo** são então *dimensões* pelas quais os valores de venda podem ser consolidados (Figura 5-a). Através de cada dimensão, valores das vendas (*métricas do fato*) são agregados em níveis crescentes de detalhe, fornecendo ao usuário final visões diferenciadas sobre as vendas da empresa (Figura 5-b).

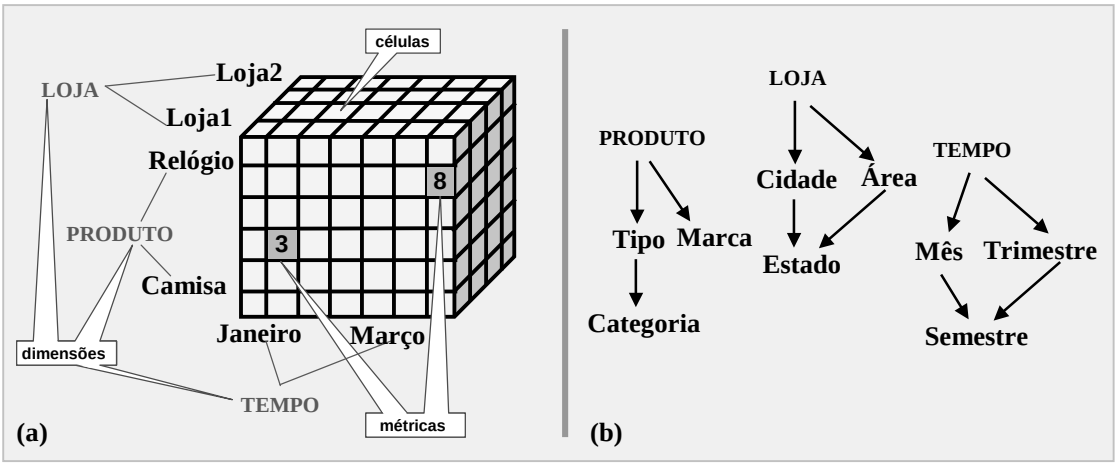


Figura 5. Elementos multidimensionais no cubo de dados[5].
(a) fatos, dimensões, células e métricas.
(b) hierarquias dimensionais e seus atributos.

A estrutura multidimensional do cubo de dados fornece uma plataforma mais flexível para a execução de consultas analíticas complexas, maximizando a eficiência do esforço computacional exigido e minimizando, ao mesmo tempo, a quantidade de tabelas e relações

[5] adaptado de (TRUJILLO *et al*, 2001).

no repositório de dados. Acessos ao cubo de dados, por sua vez, denotam operações envolvendo volume intenso de dados e alta carga computacional. Tais operações não são bem representadas por métodos de acesso tradicionais, concebidos para otimizar o acesso transacional concorrente a pequenas quantidades de informação (TRUJILLO e PALOMAR, 1998). Especialmente projetada para suprir essa deficiência, a tecnologia OLAP (*On-Line Analytical Processing*) (CODD *et al.*, 1993) possibilita respostas rápidas e consistentes em consultas a dados agregados, independente do tamanho do banco ou da complexidade do modelo multidimensional. Utilizando o cubo de dados como plataforma-base, o modelo OLAP implementa visões configuráveis dos dados em diferentes ângulos e níveis de agregação, por meio de operações especiais (Figura 6) tais como:

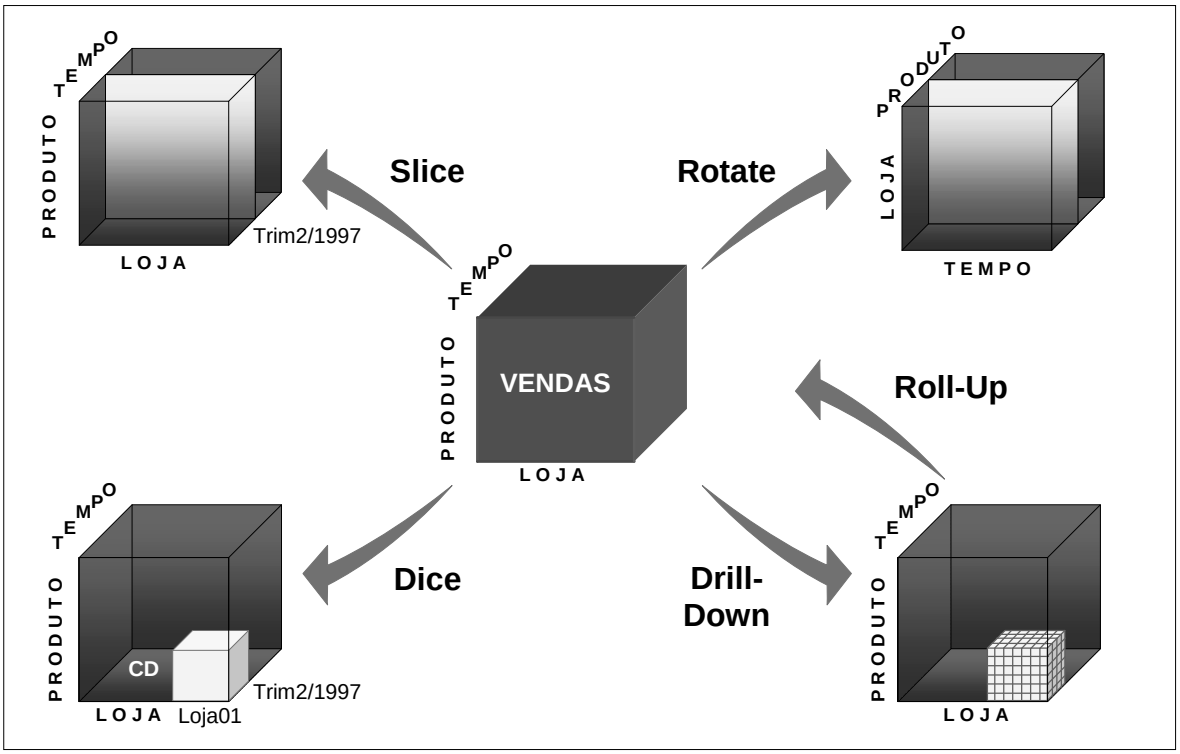


Figura 6. Operadores OLAP.

- SLICE/DICE: O foco do usuário é transferido para uma camada de dados particular ou sub-cubo de dados, respectivamente, pela secção de fatias (*slices*) do cubo ou pela extração de um sub-cubo (*dice*) de dados agregados, fixando-se valores de dimensão. Por exemplo, o

total das vendas do segundo semestre de 1997 pode ser obtido pelo "*Slice*" da Figura 6. O seu refinamento para o produto "CD" e Loja 01 é representado pelo "*Dice*" logo abaixo na mesma figura;

- ROTATE (Rotação) ou PIVOT: Modifica a orientação do cubo, ao trocar a posição das dimensões entre os eixos do cubo, gerando uma nova configuração de análise dimensional. No exemplo da Figura 6, uma rotação alterna a visão "produtos vendidos em cada loja no período" para a vertente "lojas que no período venderam determinados produtos";

- DRILL-DOWN/ROLL-UP: Utiliza as hierarquias dimensionais para oferecer visões mais refinadas ou mais agregadas dos dados. Na Figura 6, as vendas de um dado produto pode ser refinada (*drilled-down*) ao longo das dimensões "tempo" e "loja", e vice-versa.

Sistemas OLAP são tipicamente implementados sobre bancos de dados relacionais. O Modelo Relacional foi introduzido por CODD em 1970 (CODD, 1970) e propõe a representação de bancos de dados como uma coleção de *relações* ou *tabelas* de valores. Desde então, o modelo Entidade-Relacionamento (E/R) (CHEN, 1976) tem sido adotado para a modelagem conceitual de tabelas e relacionamentos entre estas. Entretanto, existe um consenso na comunidade científica de que modelos relacionais tradicionais gerados a partir da técnica E/R não são adequados ao projeto de *data warehouses* (BOEHNLEIN e ULBRICH-VOM ENDE, 1999; KIMBALL, 1998; BULOS, 1996; PEDERSEN e JENSEN, 1999; SAPIA *et al.*, 1998). KIMBALL *et al*. (1998a) resume bem esse consenso ao colocar que:

1. Os esquemas relacionais resultantes de uma modelagem tradicional contrariam uma premissa chave em *data warehousing* que é a recuperação intuitiva e em alta-performance dos dados;

2. Usuários finais não conseguem entender ou navegar um modelo E/R complexo, que pode em alguns casos representar centenas de entidades;

3. Softwares comuns não conseguem aplicar consultas analíticas a modelos relacionais tradicionais de forma eficiente. Otimizadores de consulta que tentam suprir essa deficiência são notórios por efetuar escolhas inadequadas, com grande perda de performance;

4. Modelos E/R não são extensíveis o suficiente para acomodar mudanças nos requisitos de negócio do sistema. Em conseqüência, toda a estrutura de entidades e relacionamentos, bem como as funcionalidades da aplicação nela apoiadas, têm de ser revistas e adaptadas quando da inclusão de novos elementos ou mudança em requisitos de projeto. Em justaposição a essa característica, em ambientes *data warehouse* os requisitos do usuário estão sujeitos a mudanças constantes, tornando a evolução do esquema conceitual um aspecto primordial para o sucesso do projeto.

Novos esquemas[6] de representação foram, então, desenvolvidos para modelar os requisitos de dado inerentes ao paradigma multidimensional. Os esquemas mais adotados são o *Estrela* e o *Flocos de Neve*. No esquema **Estrela**, métricas e atributos de dimensão são mapeados através de dois tipos específicos de tabelas: uma *Tabela-Fato*, que armazena as métricas a serem analisadas; e *Tabelas-Dimensão*, formadas pela hierarquia de atributos qualitativos de uma dada dimensão. Tabelas-dimensão são conectadas ao redor da tabela-fato, descrevendo uma estrutura em forma de estrela (Figura 7). A representação de uma dimensão como uma única tabela leva à *desnormalização* dos atributos que a compõem, i.e., a redundância de atributos é permitida dentro da tabela. Uma forte justificativa para a adoção desta política reside na otimização conseguida sobre o processamento de consultas complexas, o que torna o esquema estrela o preferido para a representação de *data warehouses* (KIMBALL, 1998).

[6] Do ponto de vista semântico, a modelagem de entidades, seus atributos e relacionamentos para representação conceitual de estruturas multidimensionais não constitui um novo *modelo*, mas sim um *esquema* de dados (TRYFONA *et al*, 1999). O esquema gerado utiliza os mesmos princípios da modelagem E/R para dispor as entidades numa configuração mais adequada ao acesso rápido e facilitado ao dado agregado.

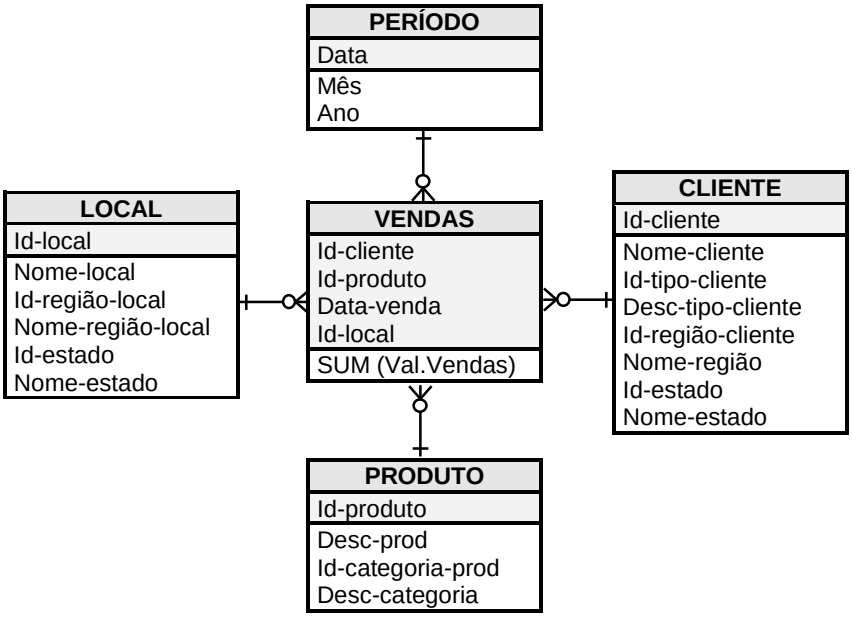


Figura 7. Esquema Estrela.

No esquema **Flocos-de-Neve** (*Snowflake*), tabelas do tipo *dimensão* são transformadas (normalizadas) com base na *terceira forma normal* (CODD, 1972), eliminando a redundância entre atributos (Figura 8).

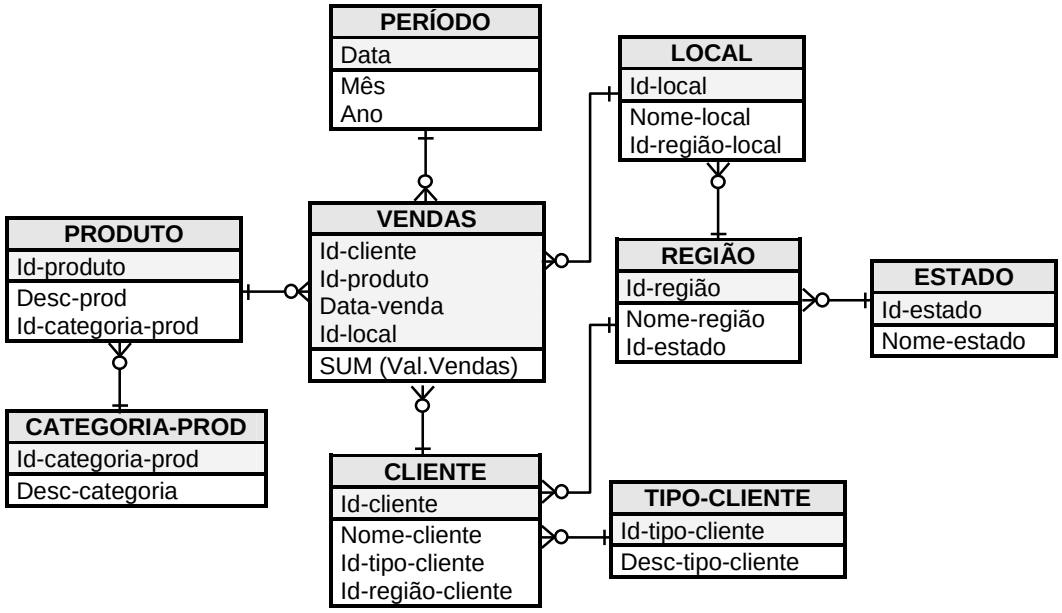


Figura 8. Esquema Flocos de Neve (*Snowflake*).

Para cada hierarquia de dimensão, uma tabela em separado é gerada. A despeito da perda comparativa em performance com relação ao modelo estrela, o esquema flocos-de-neve tem a vantagem de simplificar a execução de operações OLAP sobre o modelo multidimensional (MOHANIA *et al.*, 1999), enquanto preserva a flexibilidade e simplicidade dos esquemas multidimensionais.

A escolha do modelo adequado depende da análise comparativa entre custos de armazenamento e o desempenho requerido nas consultas ao repositório. Ademais, algumas variações sobre os dois modelos apresentados são sugeridas na literatura. O modelo em ***Constelação*** (Figura 9) apresenta esquemas-estrela com tabelas-fato hierarquicamente interligadas por meio de dimensões comuns (TESTE, 2001), possibilitando a recuperação de dados entre *visões de negócio* situadas em tabelas-fato autônomas.

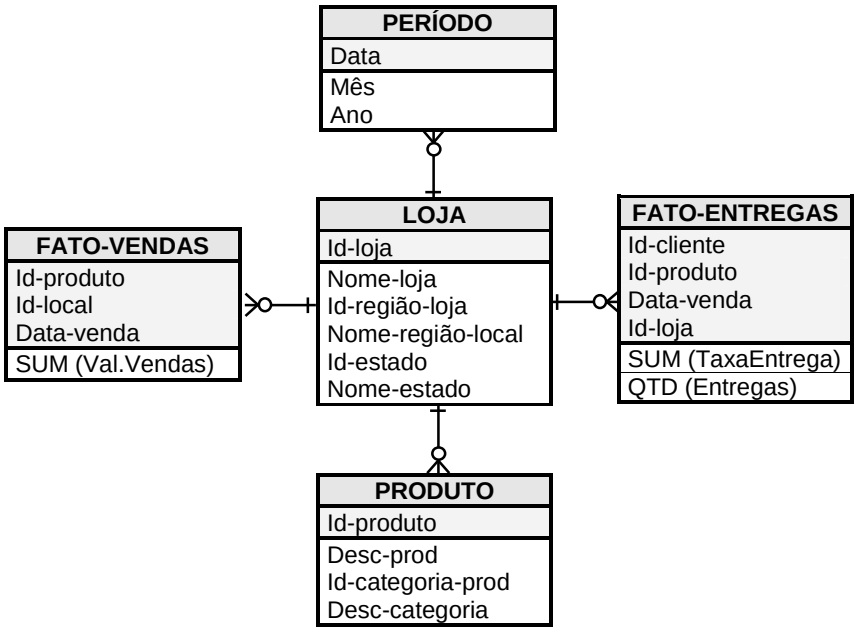


Figura 9. Esquema Constelação.

3.4 O Processo de Data Warehousing

Ambientes *Data Warehouse* integram o repositório de dados e esquemas multidimensionais a um processo definido, responsável por dar vida ao *warehouse* e suas tabelas. O processo de *Data Warehousing* compreende quatro grandes fases:

(a) **Extração** dos Dados, a partir de fontes operacionais heterogêneas (em sua maioria sistemas legados), distribuídas ao longo de inúmeros sistemas operacionais internos e/ou externos à organização. Dados são processados num esquema periódico, podendo ser adquiridos diretamente da fonte provedora, segundo condições pré-estabelecidas entre as partes, ou (mais comum) fornecidos por esta com base em padrões de integração definidos pelos projetistas do *data warehouse*. Os dados podem estar em formatos que vão desde tabelas relacionais a arquivos ASCII, o que requer sua tradução e adequação para a estrutura-padrão do repositório.

(b) **Transformação** do dado bruto, o que envolve basicamente a adaptação, limpeza e consolidação das informações antes de serem integradas ao *warehouse*. O objetivo principal dessa etapa é eliminar as diferenças semânticas entre o dado extraído e o esquema multidimensional adotado. A consolidação final resulta da execução de uma seqüência de atividades, dentre as quais (*i*) a equivalência entre atributos (ex. *as duas primeiras posições do campo "status" na fonte são equivalentes ao "estado civil" no warehouse*); (*ii*) resolução de sobreposições (*overlappings* – ex. *os campos "matrícula" e "cod-cliente", representando a mesma informação, com formatos diferentes*); (*iii*) definição de chaves primárias e secundárias no *warehouse*; (*iv*) tratamento de diferenças semânticas (ex. *traduzir o dado* "Sexo" *de "M" e "F" para "01" e "02", respectivamente*); e (*v*) complementação de dados ausentes (*valores nulos, em branco ou zerados*).

(c) **Carga** dos dados no repositório. Essa fase envolve a geração de programas para alimentação do banco de dados, num processo eminentemente *batch*. Devido ao alto volume de dados, técnicas especiais como processamento paralelo ou incremental

são utilizadas para aumentar a eficiência e garantir a entrega dos dados dentro dos requisitos de carga (*data prevista, taxa de erros máxima, quantidades dentro do volume acordado*). Durante essa fase, outras atividades são desempenhadas tais como a criação de índices, classificação e agregação de dados, particionamento de bases e checagem de integridade (controle de qualidade).

(d) **Consulta** aos dados consolidados, utilizando-se a estrutura multidimensional montada durante a fase de modelagem e as facilidades de ferramentas OLAP (ex. "*obter a soma das vendas de produtos no ano 2000 categorizada por loja, tipo de produto, cidade e estado*"). Aplicações *data warehouse* diferenciam-se de sistemas convencionais neste ponto pois a implementação de consultas se apóia na adaptação da ferramenta OLAP para as necessidades do usuário, ao invés de desenvolver interfaces proprietárias por intermédio de linguagens de programação. O desafio maior é tornar a interface aderente aos padrões de interoperabilidade e requisitos de qualidade estabelecidos pelo usuário. Por exemplo, o usuário pode definir que a interface seja desenvolvida para o ambiente Web e permita a geração instantânea de gráficos estatísticos a partir dos dados apresentados, com a mesma performance de aplicações cliente/servidor proprietárias com as quais está acostumado.

De fato, a construção de aplicações de suporte à decisão é um processo complexo e contínuo, influenciado diretamente pela dinâmica dos ciclos de vida das fontes provedoras. O subprocesso de Extração/Transformação/Carga, formado pelas fases (a), (b) e (c) respectivamente, é especialmente sensível a mudanças no conteúdo ou formato dos dados extraídos, as quais podem impactar toda a cadeia de alimentação do *data warehouse*. Para quebrar a complexidade desse processo em partes mais facilmente administráveis e permitir a coordenação dos grupos envolvidos, uma arquitetura em camadas (Figura 10) foi idealizada, onde cada nível encapsula ferramentas, métodos e procedimentos utilizados no ciclo de *data warehousing*.

Após a fase de Extração/Transformação/Carga (ETC), o dado integrado ao *warehouse* representa a consolidação de uma variedade de informações operacionais relevantes à tomada de decisão na organização. Essa gama de informações, contudo, nem sempre é significativa para todos os segmentos de negócio dentro da corporação. Alguns departamentos podem estar mais interessados, por exemplo, em realizar análises estratégicas sobre dados de vendas, em detrimento de montantes de reposição de estoque.

Por outro lado, o planejamento da construção de um *data warehouse* corporativo pode envolver dezenas de milhões de registros e uma centena de elementos multidimensionais, entre fatos e dimensões, aumentando os riscos da adoção de uma estratégia monolítica para montagem das diversas visões de negócio, todas a um mesmo tempo.

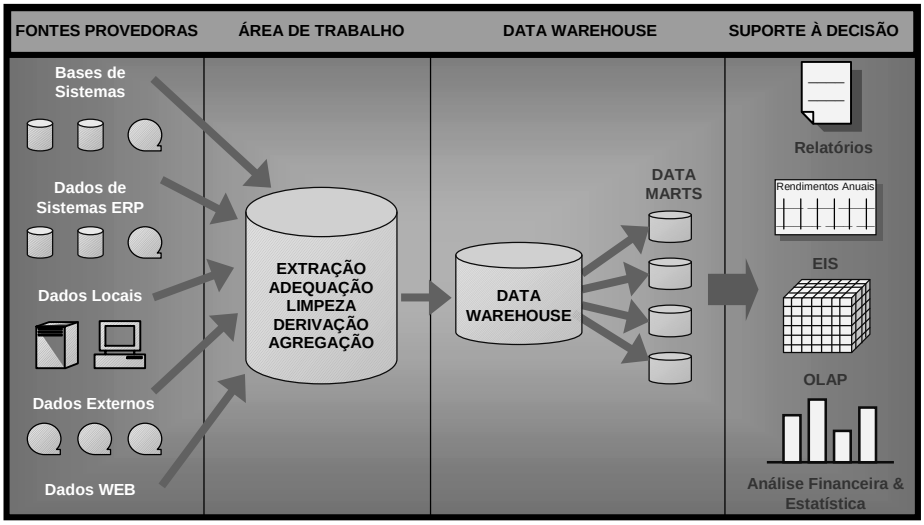


Figura 10. Arquitetura e Processo de *Data Warehousing* [7].

Inúmeros autores (MOODY e KORTINK, 2000; WATTERSON, 1998; KIMBALL, 1998; BREITNER, 1997; POWER, 2000) argumentam que uma solução mais viável é a divisão do *data warehouse* global em subconjuntos de dados significativos, que são alimentados de acordo com a visão departamental a que atendem.

[7] Adaptado de (WATTERSON, 1998).

Esses *mini-data warehouses* são denominados ***Data Marts*** e constituem os blocos de construção dos modernos *data warehouses*. Segundo BREITNER (1997), *data marts* representam o ponto de partida para o projeto do *Data Warehouse Corporativo* da empresa, sendo desenvolvidos individualmente para cada departamento e depois combinados para formar uma visão de negócios global. Em muitos casos, a estratégia de construir *data warehouses* a partir de *data marts* revela-se uma opção mais rápida e barata, ao passo que o produto gerado funciona como protótipo para demonstração e validação evolutiva dos requisitos dos usuários. WATTERSON (1998) coloca ainda que os *data marts* possibilitaram que projetos em *data warehouse* deixassem de ser vistos como empreendimentos custosos e arriscados, e passassem a serem encarados como algo que toda organização necessita como forma de manter a sua competitividade no mercado.

3.5 Aplicações OLTP versus OLAP

Aplicações OLAP diferenciam-se significativamente de sistemas convencionais ou OLTP (*On-Line Transaction Process*), projetados para oferecer suporte ao processamento de transações *on-line*.

No mundo OLTP, uma grande quantidade de usuários acessa concorrentemente coleções de dados armazenadas num repositório central ou distribuídas ao longo de poucas bases de dado. O processamento pode ser caracterizado por um número relativamente fixo de transações pré-definidas e simples de *read*/*write* (leitura/gravação), as quais afetam uma quantidade pequena de registros (um registro por vez). O acesso a dados individuais é facilitado pela adoção de representações normalizadas dos dados, cujo objetivo maior é eliminar redundâncias. O tamanho dos arquivos de dado do sistema é restrito a alguns gigabytes de informação. O processo de desenvolvimento de novos sistemas utiliza técnicas tradicionais para modelar as funcionalidades a serem implementadas, que variam de acordo com a natureza da aplicação.

Em contraste, aplicações OLAP são geralmente operadas por um pequeno número de analistas de negócio ou executivos, numa ordem de acessos bastante inferior à de sistemas

OLTP. O acesso é limitado a transações *read-only* (somente leitura), por sua vez extremamente complexas e associadas a um grande volume de dados que inclui informação histórica (muitos registros de uma única vez), armazenada ao longo de dezenas ou centenas de tabelas-fato. Conseqüentemente, o volume total de dados alcança a magnitude de terabytes. Além disso, ao invés de pré-definidas, essas "transações" são construídas de forma *ad-hoc* pelo usuário, com base em critérios de seleção que usam as diversas tabelas-dimensão para materializar diferentes perspectivas sobre o dado. Considerando-se que o dado não é atualizado, e em face ao volume de processamento, o uso de modelos desnormalizados encontra na redundância dos dados uma solução para otimização das consultas. Uma outra característica de aplicações de suporte à decisão é a obediência a um processo-padrão de desenvolvimento, centrado numa arquitetura em camadas que implementa funcionalidades para a recuperação, tratamento e consolidação de informações heterogêneas (Seção 3.4). comuns a todas as aplicações *data warehouse*. A Tabela 3 resume as principais diferenças entre sistemas OLAP e OLTP.

Característica	Sistemas OLTP	Sistemas OLAP
Unidade	**Transação**	**Dado**
Número de Usuários	**Alto**	**Baixo**
Complexidade das Transações	**Baixa**	**Alta**
Natureza da Transação	**Simples, Pré-Definida**	**Dinâmica e Complexa**
Foco	**Registros individuais**	**Milhões de registros**
Tamanho das fontes	**Gigabyte**	**Gigabyte-Terabyte**
Atualidade dos dados	**Valores recentes**	**Valores recentes e históricos**
Modelo de Dados	**Normalizado**	**Desnormalizado**
Processo de Desenvolvimento	***Customizado***	**Padrão**
Dados	**Inseridos/Alterados**	**Inseridos**
Informação Primária	**Detém o controle**	**Controlada por fontes externas**

Tabela 3. Características de Sistemas OLAP e OLTP.

3.6 Principais Metodologias para Desenvolvimento de Sistemas Data Warehouse: Um Enfoque de Requisitos

Data Warehousing é, ao mesmo tempo, uma área nova e um processo de difícil condução devido às suas peculiaridades e à intrincada cadeia de fatores que têm de ser integrados e gerenciados para viabilizar o desenvolvimento de aplicações em suporte à decisão. Nesse sentido, TRYFONA *et al*. (1999) argumenta que uma metodologia que forneça uma plataforma em alto nível para suporte ao projeto de bancos de dados multidimensionais é uma necessidade premente. Tal metodologia é a chave para tornar os projetos de *data warehouse* mais próximos do **entendimento** do usuário e garantir a independência de aspectos de implementação. Benefícios imediatos da adoção de um processo organizado são (*i*) a definição de um esquema conceitual formal, completo, e livre de ambigüidades; (*ii*) detecção antecipada de erros de modelagem, com redução do impacto de requisitos alocados erroneamente em fases avançadas do ciclo de vida do projeto; e (*iii*) aplicações mais facilmente extensíveis. WATTERSON (1998) coloca ainda que metodologias desempenham um importante papel não somente no planejamento e na construção bem-sucedida de grandes *data warehouses* escaláveis, mas também na sua entrega rápida.

VASSILIADIS (2000) argumenta, contudo, que, ao reler os dois livros clássicos sobre *data warehouse* (INMON, 1996; KIMBALL, 1998), "*tem-se a impressão de que eles provêm dicas e soluções para fragmentos do processo como um todo, ao invés de uma metodologia concreta para o profissional de* data warehouse". Como conseqüência, os projetos tendem a focar em soluções com nítido apelo físico (*otimização de consultas, integração de dados heterogêneos, manutenção de visões materializadas,* dentre outros aspectos), enquanto se distanciam do correto entendimento dos requisitos do usuário. Inúmeras metodologias têm sido propostas tanto pela comunidade científica quanto pela indústria (HADDEN e KELLY, 1997; KIMBALL *et al.*, 1998a; GOLFARELLI e RIZZI, 1999; HÜSEMANN *et al.*, 2000; MOODY e KORTINK, 2000; NASH e ONDER, 2002; ASCENTIAL, 2002) com o intuito de prover uma visão de mais alto nível ao projeto de aplicações *data warehouse*.

Em comum esses trabalhos possuem a concepção de um modelo organizado numa seqüência de fases, que compõem um ciclo de vida completo de desenvolvimento.

Embora a maior parte das abordagens caracterizem o início do ciclo de desenvolvimento com uma fase de *Levantamento* ou *Análise de Requisitos*, constata-se que estas falham em seu objetivo maior, ao oferecer um processo pouco consistente (não sistemático) de análise dos requisitos de negócio que irão nortear a construção da aplicação. Aspectos como ambigüidades, conflitos e omissões em requisitos não são tratados, enquanto questões relacionadas com a implementação do *warehouse* são elevadas a um primeiro plano, sem antes haver um entendimento sólido dos objetivos do usuário. Técnicas informais de elicitação, documentação, análise e validação de requisitos (não necessariamente em todas essas categorias) desempenham um papel mais ou menos significativo nesse momento, variando de acordo com a abordagem (PAIM *et al.*, 2002). Todavia, mecanismos para o gerenciamento preciso e controlado dos requisitos não são providos.

O foco se concentra, em seguida, no projeto de uma arquitetura multidimensional robusta o bastante para acomodar toda a carga de dados operacionais, com uma boa performance de acesso. KIMBALL (1998) coloca, contudo, que os requisitos de negócio dos usuários influenciam quase toda decisão tomada ao longo da implementação do *data warehouse*. Ainda segundo este, o modelo de requisitos determina “o quê” deve estar disponível no sistema, como essa informação deve estar organizada e em que periodicidade. Mesmo arquiteturas robustas podem ser seriamente afetadas por mudanças nos requisitos do sistema, o que requer um alto nível de gerenciamento. Assim, uma boa especificação de requisitos é a chave para o desenvolvimento de aplicações *data warehouse* de qualidade. Nesse trabalho é proposta uma abordagem metodológica para a definição de requisitos em sistemas *data warehouse* que suplanta as deficiências das metodologias atuais, por meio da incorporação de técnicas de Engenharia de Requisitos ao processo de *Data Warehousing*.

A seguir, as principais metodologias para desenvolvimento de sistemas *data warehouse* e sua abordagem para análise dos requisitos são discutidas.

3.6.1 O Método Hadden-Kelly

A metodologia Hadden-Kelly (HADDEN e KELLY, 1997) é uma das mais difundidas na indústria e licenciada para grandes empresas de informática como a **Software AG**, **Reston**, **VA**, dentre outras. Earl Hadden e Sean Kelly propuseram um método evolucionário, centrado na construção rápida de *data warehouses*/*data marts*, com refinamento posterior da infraestrutura tecnológica, regras de negócio, processo de ETC (Seção 3.4) e requisitos de negócio do usuário. Resultados colhidos a cada fase são realimentados no modelo para a melhoria progressiva de novas *releases* (versões intermediárias). O método está divido em quatro fases: (1) **Preparação**, na qual define-se o "caminho ótimo" (*fundamentos de negócio*, *ambiente técnico*, *disponibilidade e qualidade do dado*, *recursos necessários*) para a construção do *data warehouse/data mart*. Como saída, essa etapa apresenta um ***Plano de Ação*** para a organização; (2) **Planejamento**, focado em objetivos, dados e áreas específicas de negócio que serão alvo do *data warehouse*. Um ***Plano de Implementação*** descrevendo as atividades de construção do projeto é preparado e prioridades são atribuídas de acordo com a importância dos *data marts* para a estratégia de negócio; (3) **Construção**, correspondendo à implementação de um *data mart* (e/ou parte do *Data Warehouse*) que atenda a um objetivo particular de negócio. Inclui o mapeamento de dados operacionais para o modelo do *warehouse*, definição e geração de programas de extração e transformação, controle de qualidade dos dados, dentre outras atividades; (4) **Operação**, caracterizada pela condução de atividades de backup e recuperação de dados, monitoração de performance, e melhoria do processo com base nos *feedbacks* obtidos.

A abordagem evolucionária do método flexibiliza o gerenciamento de mudanças em requisitos, reduzindo o impacto geral no desenvolvimento. A fase de *Preparação* trabalha articulada com o *Planejamento* para coletar um escopo bem definido do projeto. Contudo, observa-se um foco maior na gerência do projeto em contraposição à gerência dos requisitos de sistema. Tanto essa atividade, quanto a coleta e validação dos requisitos, não são apoiadas por uma sistemática específica, o que é agravado pela forte influência de

fatores implementacionais, reflexo da ausência de uma visão de requisitos de mais alto nível.

3.6.2 Business Development Lifecycle (BDL)

A metodologia criada por KIMBALL *et al*. (1998a) tem como pano de fundo um *framework* conceitual que descreve uma seqüência de etapas de alto-nível requeridas para o projeto, desenvolvimento e implantação efetivos de um *data warehouse*. O ciclo de vida proposto inicia-se com o **planejamento** do projeto. Nessa etapa são definidos o escopo da aplicação, os critérios de validação e a oportunidade de negócio que justifica sua implementação. KIMBALL *et al*. (1998a) acredita que a probabilidade de sucesso de um projeto *data warehouse* é consideravelmente aumentada se um entendimento consistente dos requisitos dos usuários é estabelecido. Dessa forma, o *framework* propõe, em seguida ao planejamento, uma etapa de **definição dos requisitos de negócio**, cujo objetivo principal é alcançar o entendimento dos requisitos de negócio que motivam a construção do *data warehouse* e traduzi-los para considerações de projeto. A fase subseqüente de **modelagem dimensional** constrói, a partir dos requisitos levantados, um modelo multidimensional que implementa as necessidades estratégicas dos usuários e garante extensibilidade ao *warehouse*.

As demais fases que integram o *framework* tratam de aspectos relacionados com a **implementação física** do modelo multidimensional, **definição da infraestrutura** para suporte aos processos de Extração/Transformação/Carga e **implantação** do *data warehouse*. Em relação a metodologias similares, a abordagem de KIMBALL *et al*. (1998a) se destaca pela separação entre requisitos de negócio e projeto físico da aplicação; pela busca da conformidade de requisitos multidimensionais comuns com o modelo corporativo; e pelo uso de sessões de *entrevista* e *reuniões* para elicitar, analisar e negociar requisitos de sistema. Apesar de bem estruturada, a fase de requisitos não provê meios para a gerência eficiente dos requisitos coletados, particularmente no que tange a mudanças de escopo. Os modelos (*templates*) de artefato propostos para documentação dos requisitos apresentam

um baixo nível de abstração em detrimento de uma representação detalhada de aspectos funcionais e não-funcionais do *data warehouse*. Ademais, a metodologia não descreve de forma clara como a multidimensionalidade do modelo é extraída a partir dos requisitos do negócio.

3.6.3 Golfarelli e Rizzi

GOLFARELLI e RIZZI (1999) propõem um *framework* genérico para o projeto de *data warehouse*, estruturado ao longo de seis fases : (*i*) análise do sistema de informação; (*ii*) especificação de requisitos; (*iii*) projeto conceitual; (*iv*) refinamento e validação do esquema conceitual; (*v*) projeto lógico; e (*vi*) projeto físico.

A fase de **análise** trata de um aspecto bastante peculiar a aplicações *data warehouse*, comparadas a aplicações convencionais: a existência de documentação a priori sobre a fonte provedora de dados. A referida documentação é analisada conjuntamente por projetistas e equipe do sistema provedor, para produzir os esquemas de integração iniciais. Esses esquemas tornam-se entrada para a fase de **especificação de requisitos**, onde as necessidades do usuário são filtradas, produzindo como saída uma especificação preliminar de fatos e dimensões.

Do ponto de vista gráfico, um esquema multidimensional é elaborado a partir da especificação preliminar, durante a fase de **projeto conceitual**, para acomodar definições sobre a natureza de dimensões, métricas, hierarquias e atributos pertinentes ao sistema. Na etapa de **refinamento e validação**, um conjunto inovador de expressões para instância de fato permite determinar se o modelo hierárquico gerado atende a todas as necessidades de consulta e agregação de dados do sistema. O **projeto lógico** subseqüente recebe como entrada o esquema dimensional validado e produz um esquema multidimensional suficientemente capaz de otimizar as consultas a serem operadas sobre o repositório. A sexta e última etapa do processo de desenvolvimento envolve a conversão da visão lógica dos dados para um **projeto físico** mais adequado à implementação no banco de dados, onde considerações sobre os índices a serem adotados desempenham importante papel.

A despeito de ainda em evolução, o trabalho de Golfarelli e Rizzi traz como importante contribuição a definição de um *framework* completo, que contempla todas as fases do desenvolvimento de um *data warehouse*, apoiado num modelo conceitual formal para data warehouses denominado *Dimensional Fact Model* (GOLFARELLI *et al.*, 1998). A consistência das fases de projeto (conceitual, lógico e físico), todavia, não se reflete na fase de especificação de requisitos, eminentemente centrada na elicitação de requisitos, sem o apoio de um procedimento sistemático. A determinação de fatos e dimensões que compõem o modelo depende da experiência de administradores de bancos de dados e engenheiros de software, não sendo estabelecida uma ligação entre requisitos multidimensionais e necessidades de negócio dos usuários. Por fim, o *framework* não aborda a documentação e o gerenciamento dos requisitos de negócio.

3.6.4 Conceptual Data Warehouse Design (CDWD)

Em (HÜSEMANN *et al.*, 2000), uma metodologia para projeto multidimensional intitulada *Conceptual Data Warehouse Design* (CDWD) é sistematicamente construída com base na modelagem tradicional de bancos de dados (BATINI *et al.*, 1992). A exemplo de Golfarelli e Rizzi (1999), os autores defendem um estudo de modelagem conduzido numa seqüência de passos, que vão da análise e especificação de requisitos, ao projeto físico do banco. O enfoque maior, contudo, está nas fases de **levantamento de requisitos** e **projeto conceitual**. Esta última fase é tida como "*a mais importante*" do projeto de sistemas *data warehouse*, e tem a função de derivar o esquema de dimensões, métricas e hierarquias que formará a base da aplicação. Para tanto, o modelo fornece: (*a*) diretrizes para a modelagem de um bom esquema de dados e restrições de integridade dentro de um contexto multidimensional; (*b*) um formalismo gráfico que captura apropriadamente a distinção entre atributos e propriedades num nível dimensional; e, finalmente, (*c*) uma aplicação direta dos benefícios demonstrados em (LEHNER *et al.*, 1998) sobre o uso de *formas normais multidimensionais* (FNM) no projeto de *data warehouse*, para garantia de corretude nos resultados das agregações. O projeto conceitual utiliza uma *representação tabular* dos

requisitos multidimensionais coletados, além de informações suplementares (*restrições de integridade*, *requisitos derivados*) adicionadas informalmente.

A contribuição principal do modelo CDWD é apresentar técnicas algorítmicas bem elaboradas para obtenção dos produtos da fase de projeto conceitual, algebricamente fundamentadas nas dependências funcionais entre atributos, e na teoria de Formas Normais Multidimensionais. O resultado é um esquema multidimensional livre de anomalias de agregação, que assegura a eficiência necessária às consultas analíticas. Para tanto, os autores propõem uma abordagem para evidenciar os níveis de agregação entre métricas e dimensões. Quatro níveis são definidos: no *Nível 1*, todas as operações de agregação são possíveis (SOMA, MÉDIA, DESVIO PADRÃO, MÁXIMO, MÍNIMO, VARIÂNCIA, CONTAR); no *Nível 2*, todas as operações fazem sentido, **exceto** a SOMA de valores; no *Nível 3*, apenas a operação CONTAR faz sentido; e no *Nível 4*, a métrica não pode ser agregada na dimensão. Mesmo assim, o modelo CDWD não garante que o esquema resultante reflita as reais necessidades de suporte à decisão do usuário. Problemas surgem quando, na prática, o esquema multidimensional desejado não garante que todas as métricas sejam funcionalmente dependentes dos mesmos níveis terminais de dimensão (ex. métricas residuais como “saldo devedor” ou discretas como “condição do tempo”), ocasionando a estratificação do esquema em inúmeras tabelas-fato, o que pode afetar sua performance e representatividade (a situação se agrava em esquemas do tipo “Constelação”). Ademais, o entendimento dos requisitos de negócio, o impacto de mudanças nos atributos e regras dimensionais, e o uso de técnicas confiáveis para a coleta e análise desses requisitos, dentre outros aspectos fundamentais para o sucesso do projeto, não são explorados pelo modelo.

3.6.5 ITERATIONS®

ITERATIONS® (ASCENTIAL, 2002) é um produto da empresa **Ascential Software** (antiga Prism Solutions Inc.) e é inspirado nas idéias de Bill Inmon (INMON, 1996), às quais foram somadas a experiência adquirida nos mais de 400 projetos *data warehouse* implementados pela empresa ao redor do mundo.

ITERATIONS® possui um processo de desenvolvimento repetível, organizado em módulos (Figura 11). As atividades requeridas para a execução completa de um módulo são identificadas com seus respectivos artefatos de saída. **Módulos** são agrupados na forma de *Trilhas*, ou em *Fases*. **Trilhas** representam conjuntos distintos de módulos que devem ocorrer em paralelo, ao longo do ciclo de desenvolvimento. **Fases** representam um agrupamento progressivo de módulos. Fases são usadas para monitorar e gerenciar os projetos. **Papéis** para cada integrante da equipe são definidos e seus interelacionamentos referenciados no ciclo de vida. Tipicamente módulos são encerrados antes do início de uma fase subseqüente, e um *checkpoint* é executado para revisar e garantir a completude de todos os artefatos de entrega e tarefas associadas.

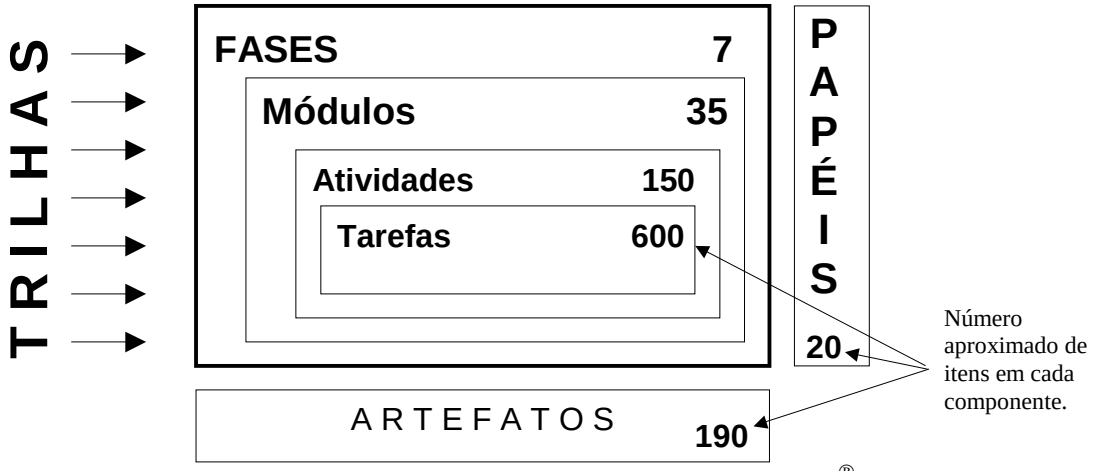


Figura 11. *Framework* do método ITERATIONS®.

ITERATIONS™ traz algumas inovações importantes para a especificação de requisitos de um projeto *data warehouse*: uma *trilha* dedicada à análise dos requisitos de negócio, negociação de escopo e definição de critérios de aceitação; papéis bem definidos; *templates* de artefatos para registro de requisitos multidimensionais identificados; *checklists* (listas de verificação) para cada uma das atividades; definição de um idioma comum para todos os envolvidos no projeto; dentre outros fatores. O processo proposto, contudo, ainda é muito voltado à especificação de esquemas conceituais otimizados, prescindindo de uma visão de requisitos de mais alto nível. Apesar da boa integração entre *trilhas* de requisitos e projeto conceitual, as atividades e técnicas que permeiam esses caminhos não capturam de forma

adequada aspectos importantes tais como a interação entre usuários e repositório de dados; requisitos de qualidade da aplicação; necessidades estratégicas e funcionalidades correlatas; dentre outros. A metodologia também não provê suporte à gerência de mudanças e priorização de requisitos de sistema.

3.6.6 Quadro Comparativo

O quadro comparativo descrito na Tabela 4 resume as deficiências das metodologias de desenvolvimento de sistemas *data warehouse* atuais em relação à incorporação de práticas de engenharia de requisitos ao ciclo de vida da aplicação.

	Hadden-Kelly	**Kimball *et al.* (BDL)**	**Golfarelli e Rizzi**	**Hüsemann *et al.* (CDWD)**	**ITERATIONS®**	**Nossa Metodologia**
Boas Práticas em Engenharia de Requisitos						
Processo Iterativo com refinamentos sucessivos	✓	✓			✓	✓
Gerenciamento de Mudanças em Requisitos	P					✓
Modelagem de Aspectos Organizacionais	P				✓	✓
Técnicas de Elicitação de Requisitos		P	P	P	P	✓
Técnicas de Análise & Negociação dos Requisitos		P			P	✓
Técnicas de Documentação dos Requisitos		P		P	P	✓
Técnicas de Validação dos Requisitos			P		P	✓
Especificação do Sistema em Alto Nível		P				✓
Modelo de Requisitos Detalhado						✓
Separação entre Requisitos e Aspectos Implementacionais	P	✓		P	P	✓
Práticas de Requisitos recomendadas para sistemas DW						
Planejamento da Gerência de Requisitos	P	P			P	✓
Modelagem de Hierarquias Múltiplas			✓	✓		
Modelagem da Aditividade dos Fatos		✓	✓	✓		✓
Modelagem da Cardinalidade das dimensões		✓				✓
Conformidade de Requisitos Multidimensionais		✓				✓
Conformidade de Requisitos Funcionais						✓

*Legenda: ✓= atende completamente (ou em sua maior parte) à prática; **P** = atende apenas parcialmente à prática; **Brancos** = não atende à prática.*

Tabela 4. Quadro comparativo entre o enfoque de requisitos das metodologias para desenvolvimento de data warehouse e a abordagem proposta nessa dissertação.

Nota-se que a maior parte das metodologias procura fornecer meios para a elicitação e documentação dos requisitos do *data warehouse*, com uma ênfase maior na modelagem de aspectos mais diretamente relacionados ao domínio multidimensional tais como hierarquias multidimensionais e aditividade de fatos. Métodos como ITERATIONS® (ASCENTIAL, 2002) apresentam uma boa cobertura das fases do ciclo tradicional de requisitos. Apesar disso, as abordagens analisadas provêm técnicas que cobrem apenas parcialmente os esforços de elicitação, análise, documentação e validação de requisitos requeridos em sistemas *data warehouse*.

Em particular, as técnicas sugeridas para especificação de requisitos funcionais e não-funcionais não delineiam de maneira completa e precisa requisitos do tipo *necessidades*, *funcionalidades*, *conhecimento do negócio*, *fatores de qualidade*, dentre outros essenciais a um bom projeto da aplicação. Adicionalmente, requisitos organizacionais como *papéis*, *responsabilidades* e *metas* de projeto são alvo de apenas duas (uma delas de forma parcial) das abordagens consideradas. A despeito da preocupação com a separação entre requisitos e aspectos implementacionais, uma especificação em alto nível do sistema somente é obtida (mesmo que parcialmente) na abordagem de KIMBALL *et al*. (1998a).

A metodologia proposta nesse trabalho suplanta essas deficiências, oferecendo um arcabouço de técnicas, artefatos e procedimentos que possibilitam uma definição precisa e em alto nível dos requisitos do *data warehouse*. Em comum com as demais metodologias, nossa abordagem apresenta um processo iterativo de análise e especificação dos requisitos, por meio de refinamentos sucessivos; procedimentos e artefatos para a definição de requisitos multidimensionais (exceto *multiplicidade de hierarquias dimensionais*); e a preocupação com o planejamento da gerência dos requisitos do sistema. Em relação aos fatores anteriormente colocados, o presente trabalho fornece, contudo, uma visão mais detalhada e especificamente moldada à definição de requisitos em *data warehouse*, a qual agrega alguns diferenciais importantes como a gerência de mudanças em requisitos e a conformidade de requisitos comuns, aspectos pouco explorados nas abordagens atuais.

3.7 Considerações Finais

Conforme visto neste capítulo, o processo de desenvolvimento de sistemas *data warehouse* difere bastante daquele aplicado a produtos de software convencionais. Transformar o dado operacional em informação estratégica para suporte ao processo de decisão requer a definição e implementação de um esquema lógico e processo de alimentação próprios, dando origem a uma nova categoria de aplicações que se destaca por suas características bastante peculiares.

O alto volume de dados, a complexidade do paradigma multidimensional e a intrincada cadeia de fatores que devem ser integrados tornam imperativa a adoção de um processo sistemático para a construção dessas novas aplicações. Embora várias metodologias tenham sido propostas nesse sentido, as abordagens atuais ainda não oferecem um processo de definição e gerenciamento dos requisitos de negócio consistente, o que, em contrapartida, é um fator essencial para o sucesso da implementação final. Nesse trabalho, propomos uma metodologia para definição de requisitos em *data warehouse* que suplanta as deficiências das abordagens de desenvolvimento atualmente praticadas dentro desse domínio específico. No capítulo seguinte, iremos mostrar como essa abordagem metodológica pode contribuir para o correto entendimento das necessidades estratégicas dos clientes e a conseqüente definição de uma solução de suporte à decisão de qualidade.

Capítulo 4

Metodologia para Definição de Requisitos em Sistemas Data Warehouse

Este capítulo descreve nossa proposta de metodologia para definição de requisitos em sistemas data warehouse. Num primeiro momento, teceremos considerações sobre a natureza, peculiaridades e requisitos inerentes a uma especificação de data warehouse como pano de fundo para a definição da metodologia. Em seguida, descrevemos a metodologia e suas fases, detalhando seus artefatos e processos constituintes. Demonstramos, posteriormente, a integração dessas fases com o novo ciclo de requisitos proposto para o desenvolvimento de aplicações data warehouse. Ao final, resumimos a importância da abordagem proposta no contexto da integração entre o processo de data warehousing e as práticas de engenharia de requisitos.

4.1 Introdução

Sistemas *Data Warehouse* emergiram como uma poderosa tecnologia para extração e integração de dados operacionais heterogêneos numa forma mais propícia à execução de análises estratégicas, em benefício de um sonho antigo para qualquer organização: tomar decisões acertadas sobre o seu futuro. O desenvolvimento de tais aplicações, todavia, é bastante diferente daquele comumente aplicado aos sistemas convencionais. De fato, *data warehouses* não envolvem apenas os requisitos estratégicos ditados pelos *tomadores de decisão*, mas também considerações sobre a estrutura e requisitos alocados às fontes provedoras. Ambas as visões operacional e estratégica têm de ser unidas num modelo de requisitos que congrega desde funcionalidades puramente voltadas ao suporte à decisão até restrições de qualidade.

Nesse contexto, o sucesso do projeto depende, mais do que nunca, de construir-se a mais precisa especificação de requisitos como base para a montagem de uma arquitetura de suporte à decisão que atenda a todas as necessidades estratégicas dos altos níveis organizacionais. Conforme evidenciado no Capítulo 2, a Engenharia de Requisitos propõe métodos e procedimentos já consagrados nesse sentido. Uma abordagem metodológica baseada em processos de engenharia de requisitos é requerida não somente para garantir o correto entendimento dos requisitos do cliente, mas principalmente para possibilitar o tratamento adequado dos requisitos funcionais, não-funcionais e sobretudo daqueles inerentes ao domínio multidimensional.

Apesar dessa atraente perspectiva, a integração entre processos da engenharia de requisitos e o processo de desenvolvimento tradicional de *data warehouses* permanece ainda um campo aberto. Projetos envolvendo essa classe de aplicações ainda são muito influenciados por aspectos operacionais em detrimento de uma análise prévia acurada dos requisitos que motivam o sistema, trazendo conseqüências que remetem ao fracasso dos projetos, conforme evidenciado no Capítulo 1. As abordagens mais atuais para construção de *data warehouses* apresentam, por sua vez, pouca consistência no levantamento, análise e

gerenciamento desses requisitos, conforme comprova a revisão das metodologias atuais presente ao final do Capítulo 3.

Para atacar esse problema, esse trabalho propõe uma metodologia para definição de requisitos em sistemas *data warehouse*. Nossa abordagem define um método iterativo e orientado por fases para a especificação e gerenciamento dos requisitos em aplicações de suporte à decisão. A metodologia descreve uma série de *templates* de artefatos para a coleta e documentação dos requisitos funcionais e não-funcionais que determinam o escopo da aplicação, bem como de suas variantes orientadas ao domínio multidimensional. Aliada a mecanismos eficientes de gerência dos requisitos, a metodologia age como um instrumento para o acompanhamento das mudanças em requisitos do usuário ao longo do projeto, permitindo aos engenheiros de sistema efetuarem manutenções apropriadas e em tempo no projeto da aplicação (PAIM *et al.*, 2002; PAIM e CASTRO, 2002a).

Nesse capítulo, detalhamos a proposta acima descrita. Inicialmente, tecemos na Seção 4.2 considerações sobre a natureza, peculiaridades e requisitos inerentes a uma especificação de *data warehouse*. Na Seção 4.3 descrevemos brevemente a metodologia. As seções de 4.4 a 4.7 detalham as fases da metodologia, enquanto que a Seção 4.8 descreve a integração dessas fases com o novo ciclo de requisitos proposto para o desenvolvimento de aplicações *data warehouse*. A Seção 4.9 resume, ao final, nossas considerações sobre a abordagem descrita nesse capítulo.

4.2 Especificação de Sistemas Data Warehouse

Data Warehouses são frequentemente encarados como sendo aplicações puramente de banco de dados. Sob essa ótica, a construção de um *data warehouse* segue os moldes de métodos tradicionais para o projeto de bancos de dados como BATINI *et al.* (1992) e VOSSEN (2000). Esses métodos estruturam o desenvolvimento numa sequência de fases através das quais um modelo de dados é evoluído de uma visão conceitual até sua correspondente representação física. É comum que a primeira fase desse ciclo envolva uma análise e especificação de requisitos. Contudo, essa especificação trata usualmente aspectos

particulares do *data warehouse* como conteúdo do esquema multidimensional, performance e visões materializadas (BARALIS *et al.*, 1997), ao invés de modelar uma visão conceitual independente, que mapeie fatores de mais alto nível, fundamentais para o sucesso do projeto, tais como objetivos e necessidades dos usuários, restrições operacionais, papéis envolvidos, responsabilidades, passos de interação e requisitos de qualidade.

Nesse trabalho argumentamos que, mais que bancos de dados, *data warehouses* caracterizam sistemas completos de informação com um processo de desenvolvimento específico, conforme descrito no capítulo 3. Em tais aplicações, requisitos de software e arquiteturais exercem entre si mútua influência, o que requer uma especificação detalhada de requisitos para assegurar a qualidade dos dados gerados e subsidiar o controle efetivo do processo como um todo.

A seguir tecemos algumas considerações sobre a natureza de aplicações *data warehouse* e definimos os principais requisitos que devem compor uma boa especificação de requisitos para um *data warehouse*.

4.2.1 Uma Abordagem *Twin Peaks*

Especificar e implementar um ambiente *Data Warehouse* é uma tarefa altamente complexa que requer um suporte metodológico capaz de tratar de forma harmoniosa requisitos de especificação conceitual e requisitos de projeto arquitetural. Uma alternativa para reduzir a distância entre classes de requisitos de origem aparentemente distinta no tempo é adotar um processo iterativo para produzir progressiva e simultaneamente uma especificação detalhada de requisitos, que trate aspectos conceituais e de projeto igualmente.

Para tanto, a metodologia aqui descrita fundamenta-se nos princípios do Processo de Desenvolvimento *Twin Peaks* (Dois Picos), proposto por NUSEIBEH (2001). De fato, uma definição de requisitos em sistemas *data warehouse* não pode ser conduzida sem levar em consideração restrições arquitetônicas advindas da multidimensionalidade característica dessas aplicações, as quais exercem influências tanto estáticas quanto dinâmicas no escopo

do sistema, obrigando os desenvolvedores a focarem igualmente em ambos os "picos" de atuação (projeto e especificação) durante todo o ciclo de desenvolvimento, conforme preconizado por NUSEIBEH (2001). No contexto dessa dissertação, estamos preocupados com os aspectos arquitetônicos ligados à definição de um esquema multidimensional para o *data warehouse* e à adaptação da ferramenta OLAP a essa estrutura (ex. *formato de atributos do esquema, conformidade de fatos e dimensões ao esquema global, prototipação OLAP, impacto de mudanças para a arquitetura relacional*).

No modelo *Twin Peaks*, enquanto requisitos de software e arquitetura são definidos concorrentemente, seus conteúdos distintos são preservados, i.e., as atividades de estruturação e especificação do problema (*definição de requisitos*) são separadas da (mas associadas à) estruturação e especificação da solução (*arquitetura*). O modelo engloba os três princípios de gerenciamento identificados por Barry Boehm em (BOEHM, 2000):

(i) **IKIWISI**[8]. Os requisitos freqüentemente só emergem depois que uma significativa análise de modelos e protótipos foi conduzida, e os usuários tiveram a oportunidade de visualizar e prover *feedback* sobre esses modelos e protótipos;

(ii) **COTS**[9]. Cada vez mais, o desenvolvimento de um sistema envolve a identificação e seleção de pacotes de software, o que requer a identificação em contrapartida dos requisitos desejáveis (geralmente expressos na forma de funcionalidades ou serviços), e o casamento dessas características com os produtos comercialmente disponíveis;

(iii) **Mudança Rápida.** A gerência de mudanças constitui-se um dos problemas fundamentais do desenvolvimento de software, e sua solução reside na identificação dos requisitos centrais que definem o sistema, i.e., aqueles que expressam os objetivos primordiais dos usuários e são mais prováveis de persistir por um período maior de tempo.

[8] ***I****'ll* ***K****now* ***I****t* ***W****hen* ***I*** ***S****ee* ***I****t*, ou "Eu saberei quando eu vir".
[9] *Components-off-the-Shelf*.

Nossas experiências iniciais sustentam que esses princípios são igualmente aplicáveis a sistemas *data warehouse*. Com relação ao princípio "IKIWISI", os requisitos que motivam a aplicação de suporte à decisão são freqüentemente clarificados quando as partes interessadas avaliam o sistema em desenvolvimento por meio de protótipos e versões preliminares de esquema dimensional, criados durante iterações sucessivas. Embora a utilização de componentes de prateleira (COTS) não seja uma tendência em aplicações *data warehouse*, o reuso de requisitos previamente definidos num modelo corporativo (*data warehouse global*) pode ser extremamente importante para a evolução de uma solução *Data Mart*. Adicionalmente, uma abordagem que enfatiza a distinção entre requisitos centrais do *data warehouse* e requisitos estritos de um *Data Mart* subordinado melhora a integração das partes componentes do modelo multidimensional. Consequentemente, facilita-se a gerência das mudanças rápidas, às quais esse tipo de aplicação está constantemente exposto, em requisitos. Nossa abordagem enfatiza todos esses níveis de gerenciamento.

4.2.2 Requisitos Essenciais em Sistemas Data Warehouse

Para melhor correlacionar os requisitos do problema (especificação) com os requisitos da solução adotada (arquitetura), os seguintes conceitos devem ser investigados durante a modelagem dos requisitos de um *data warehouse*:

(*i*) **Representar fatos e suas propriedades**. Fatos são elementos centrais em *data warehouses*. Analisar os requisitos do usuário implica identificar *fatos* e suas propriedades por intermédio das métricas escondidas na demanda do usuário. Por exemplo, a especificação de uma aplicação *data warehouse* para análise da situação de hipotecas habitacionais deve considerar o fato "*pagamento das prestações*", e suas propriedades discretas "*montante pago*" e "*saldo devedor*", como requisitos essenciais do sistema. Um outro requisito determinante do potencial de suporte à decisão do *data warehouse* é o grau de *aditividade* dessas métricas, i.e., a capacidade da métrica de ser "operada" (somada, contada, extraída a média) numa dada perspectiva de análise do tomador de decisão;

(*ii*) **Distinguir e conectar dimensões a fatos**. Dimensões fornecem a chave para a análise estratégica das métricas de fato armazenadas no repositório de dados. Sua identificação é essencial para a correta modelagem do sistema. Muitas vezes, a análise de uma simples declaração do usuário dá margem à descoberta de um número de dimensões candidatas. Por exemplo, a sentença "*analisar as vendas mensais de itens individuais em estoque por loja*" claramente explicita as dimensões ***tempo***, ***produto*** e ***loja***, as quais estão conectadas ao fato ***montante de vendas***. Perceber as dimensões e fatos (e sua correta associação) inseridas nos requisitos do usuário é um aspecto crítico na análise dos requisitos de sistemas *data warehouse*, bem como um ponto com forte tendência a erros de concepção.

(*iii*) **Garantia de Agregabilidade (*Summarizability*)**. Um requisito não-funcional essencial a aplicações *data warehouse* é a garantia de corretude dos resultados agregados ao longo do esquema multidimensional, em qualquer combinação de dimensões e fatos associados. Essa característica é conhecida como *Summarizability* (LENZ e SHOSHANI, 1997), aqui referenciada como *Agregabilidade*. O problema está em que nem todas as possíveis agregações de métricas ao longo de dimensões fazem sentido no contexto de uma certa aplicação. Por exemplo, dada uma dimensão "mutuários", somar valores ou extrair médias dos saldos devedores de suas hipotecas são operações bastante plausíveis; no contexto da dimensão "tempo", porém, a soma desses valores de saldo parece sem sentido, o que não invalida que se extraiam médias e outras análises estatísticas a partir dessa métrica. Muito embora a existência de problemas de agregabilidade seja um fato, falhas desnecessárias na construção do esquema dimensional podem ser evitadas expressando-se claramente os requisitos que restringem a agregação de dados na aplicação, bem como estabelecendo-se uma conformidade entre fatos e dimensões comuns ao mesmo tempo a *Data Marts* e o modelo dimensional global do *Data Warehouse*. No contexto dessa metodologia, adotamos a abordagem de HÜSEMANN *et al*. (2000) para evidenciar os níveis de agregabilidade entre métricas e dimensões (vide Seção 3.6.4). A tabela contendo os níveis de classificação encontra-se descrita no corpo do artefato **Especificação de Requisitos Multidimensionais** (Apêndice 1);

(*iv*) **Representar a integração com fontes provedora**s. Em ambientes *data warehouse*, o dado é coletado a partir de diferentes fontes internas e/ou externas à organização. Essa atividade envolve, além de importar dados de todo o tipo de base, considerações sobre requisitos informais oriundos diretamente do negócio, que requerem a definição de ações específicas para sua inclusão no repositório (ex. *importar dados de um banco de dados pessoal, geração de interfaces para alimentação de dados manual pelo cliente, leitura de planilhas e relatórios gerenciais*). Entender os requisitos e procedimentos relacionados com esse processo de integração é fundamental para o projeto de um sistema *data warehouse* de qualidade, bem como para a segurança de que a aplicação definida não apresenta resultados inconsistentes;

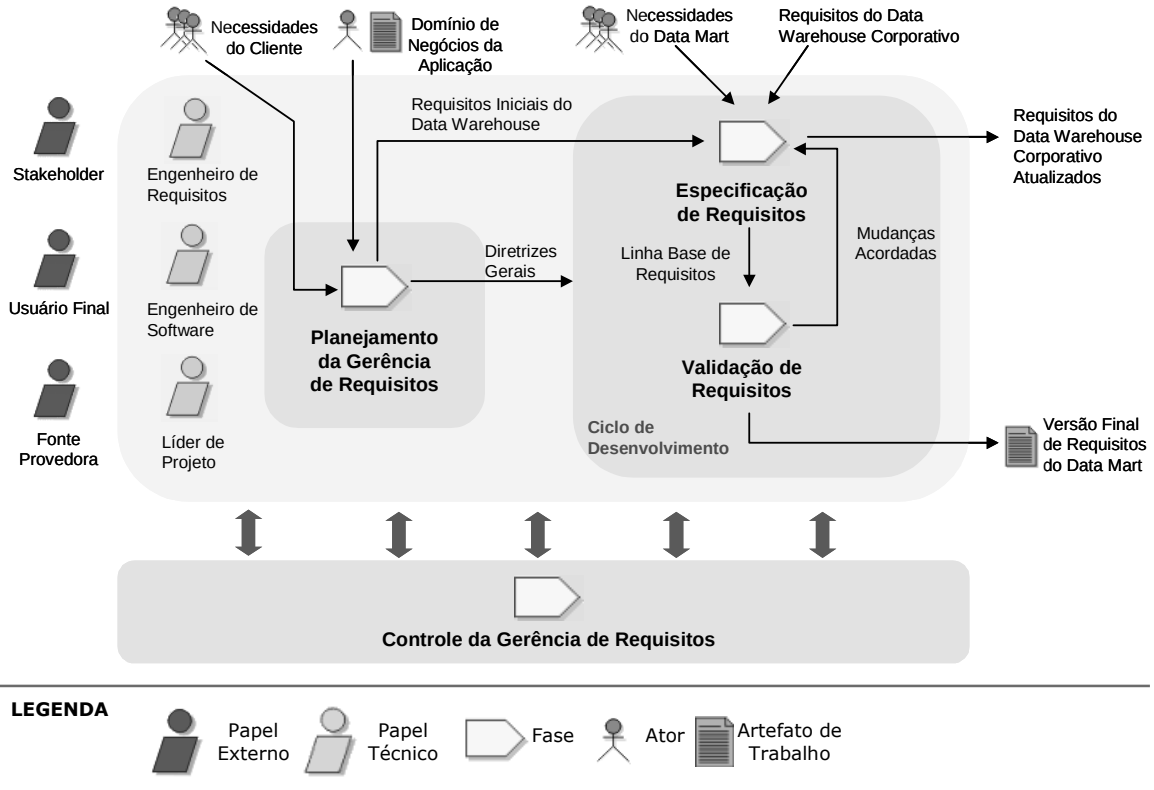
(*v*) **Acompanhamento rápido de mudanças em requisitos do usuário**. Um problema comum em sistemas convencionais, acompanhar as mudanças em requisitos assume uma importância exponencial no desenvolvimento de aplicações *data warehouse*. Até mesmo mudanças ditas "insignificantes" podem comprometer o ciclo de alimentação do repositório e a validade dos dados derivados, afetando o potencial de suporte à decisão da aplicação. Por exemplo, uma mudança na lei de composição de um atributo de dimensão para substituir "brancos" por "zeros", mesmo que impactando apenas os dígitos mais à esquerda e preservando o conteúdo do campo no repositório, pode implicar a revisão de todo o processo de transformação e carga dos dados;

(*vi*) **Documentação de alta qualidade**. Diferentemente de sistemas convencionais, cuja documentação muitas vezes precisa ser "concebida do zero", *data warehouses* sempre envolvem desenvolvedores com o exame de documentação operacional pré-existente para definir os meios e processos para integração dos dados operacionais. A não ser por raras exceções, tal documentação legada não possui a qualidade necessária, transformando a extração dos requisitos técnicos em uma atividade custosa e propensa a falhas na interpretação (ex. *a falta de manutenção da documentação pode levar à definição de regras de extração baseadas em campos cujo formato não reflete a realidade, ou que simplesmente foram exportados para outra tabela*). Dessa forma, uma documentação de

alto nível, projetada especialmente para acomodar requisitos de suporte à decisão e disseminada entre ambos os lados (fornecedor e consumidor) provê uma interface genérica para a extração desses requisitos.

4.3 Metodologia

A metodologia proposta nesse trabalho está estruturada numa sequência de fases, conforme ilustrado na Figura 12. Cada fase define a aplicação em níveis decrescentes de abstração, à medida em que os requisitos do projeto são reunidos para formar uma **linha base** (*baseline*) de requisitos, i.e., um conjunto itemizado de características ou requisitos, a ser entregue numa versão específica da aplicação (LEFFINGWELL e WIDRIG, 2000).



Notação: *Rational Unified Process*® 2001-A (RATIONAL, 2001a)

Figura 12. Descrição em alto nível da Metodologia.

A etapa inicial da metodologia (*Planejamento da Gerência de Requisitos*) procura estabelecer diretrizes para o processo de aquisição, documentação e gerência dos requisitos no ambiente *data warehouse*, tendo em vista as necessidades do usuário e o domínio do negócio. Identificado um padrão de comportamento para o projeto, soluções *data mart* individuais são então definidas em uma fase posterior de *Especificação*, considerando-se os requisitos iniciais do *data warehouse* num primeiro momento e as necessidades de suporte à decisão para o *data mart* apresentadas pelos clientes.

Durante essa fase, um modelo de requisitos para o *data mart* é gradualmente construído e integrado ao modelo de requisitos pré-existente do *data warehouse* corporativo, o qual é retroalimentado e enriquecido em especificações subsequentes, num processo iterativo de elicitação, análise, negociação, documentação e conformidade de requisitos. A *linha base* de requisitos produzida a partir desse processo serve de entrada à fase de *Validação*, na qual é analisada para identificar e remover possíveis erros de concepção. O produto final da metodologia é uma versão (*release*) acordada da especificação de requisitos do *data mart*, juntamente com o modelo de requisitos do *data warehouse* atualizado.

Ao redor de todo o processo, uma atividade de suporte denominada *Controle da Gerência de Requisitos* efetua o monitoramento constante de mudanças em requisitos do sistema. Para cada uma das partes componentes da metodologia, artefatos pré-definidos servem de instrumento para registro e análise dos requisitos coletados ao longo do ciclo de desenvolvimento. Nas próximas seções, descrevemos as fases componentes da metodologia em detalhes.

4.4 Planejamento da Gerência de Requisitos

Antes de dar início à tarefa de elicitar requisitos, regras para um efetivo processo de especificação e gerenciamento de requisitos precisam ser estabelecidas. Essas diretrizes gerais ajudam a nortear o uso da metodologia e evitar que a ausência de um padrão comum introduza inconsistências ao processo de requisitos. As diretrizes envolvem o controle da

aquisição, documentação e gerência dos requisitos de sistema, e podem ser definidas em termos de regras de negócio, procedimentos e processos devidamente ajustados e acordados entre as partes interessadas, com o objetivo de esclarecer os seguintes pontos:

(1) **Papéis e Responsabilidades**. Um erro comum em projetos de software é a ausência da atribuição clara de tarefas e responsabilidades. A definição prévia de papéis e deveres dentro do processo de requisitos, bem como a atribuição das pessoas responsáveis por exercer essas funções é decisiva para que a especificação dos requisitos de sistema e sua gestão sejam bem sucedidas.

(2) **Premissas de Integração com Fontes de Dados**. Deve-se estabelecer regras claras para o intercâmbio de dados entre fontes provedoras e o *data warehouse*, apoiadas num padrão único de integração. Periodicidade, prioridades de carga e mecanismos de controle de qualidade, entre outros aspectos, irão constituir a base das diretrizes para o processo de alimentação do repositório de dados.

(3) **Diretrizes para a Gerência dos Requisitos**. A equipe de desenvolvimento, juntamente com os demais grupos envolvidos (quando necessário) são responsáveis por definir, de forma clara: critérios de rastreabilidade dos requisitos; restrições de projeto quanto a prazos de entrega e pontos de controle; critérios para especificação de requisitos (tipos, atributos, regras de numeração e artefatos para coleta).

Os pontos acima encerram uma lista mínima, mas não exaustiva, de aspectos, a qual pode ser certamente complementada com outros pontos relevantes, de acordo com a natureza da aplicação. Em qualquer caso, o **Plano de Gerenciamento de Requisitos** resultante deve incorporar uma relação de itens que descrevam (*a*) como os requisitos de sistema serão identificados e estruturados; (*b*) quais são os critérios para o acompanhamento da gerência dos requisitos ao longo do projeto, e (*c*) quais são os responsáveis por essas atividades. É importante notar que o planejamento da gerência dos requisitos deve estar consoante com o escopo geral do projeto. Isso requer que versões preliminares dos documentos de **Visão** do

Data Warehouse e **Especificação de Requisitos Multidimensionais** (Seção 4.5.3) sejam elaboradas, para consolidar, respectivamente, aspectos fundamentais como:

(1) **Objetivos do Projeto**. Devido à alta flexibilidade provida pelo modelo dimensional e pelas ferramentas OLAP, usuários de sistemas para suporte à decisão tendem a encarar todas as suas necessidades de análise estratégica como algo factível. Em contrapartida, desenvolvedores tendem a projetar a construção do *data warehouse* sob um ponto de vista eminentemente técnico, modelando funcionalidades que destoam das reais necessidades de suporte à decisão de seus clientes. Um equilíbrio entre essas abordagens deve ser alcançado por meio da formalização clara dos objetivos do projeto entre todos os participantes, incluindo as restrições que irão limitar o seu desenvolvimento, antes do início de qualquer atividade. Por exemplo, certas consultas poderão não ser alvo de uma primeira versão do sistema devido à impossibilidade de extrair os dados necessários à composição das dimensões dentro do prazo.

(2) **Escopo Multidimensional**. Qual o nível de granularidade em cada *Data Mart* (ex. *as consultas para a tabela-fato "Vendas" terão como granularidade o cliente*)? Quais restrições legais ou operacionais restringem a análise multidimensional dos dados (*limitações da ferramenta, prazos legais que impedem a extração, etc.*)? De que formas os usuários gostariam de ver os dados agregados ao longo das dimensões (*dimensões hierárquicas, restrições de aditividade, etc.*)? As respostas para essas e outras questões influenciam diretamente a modelagem dos requisitos, e como tal devem ser capturadas como requisitos gerais do projeto.

Templates (modelos) dos artefatos citados são apresentados no Apêndice 1.

4.5 Especificação de Requisitos

O sucesso de um processo de engenharia de requisitos depende da habilidade de, a partir de declarações individuais de requisitos ainda informais e confusas, chegar a uma especificação precisa que é compreendida por, e acordada entre todos os interessados

(LOUCOPOULOS e KARAKOSTAS, 1995). No desenvolvimento de aplicações *data warehouse*, esse processo enseja uma abordagem cíclica para a aquisição, representação e avaliação dos requisitos e produção gradual de uma especificação de projeto. Nesse sentido, um processo iterativo demonstra-se mais apropriado ao suporte desse fluxo de atividades. Os requisitos iniciais (brutos) de um dado *Data Mart* percorrem uma seqüência de iterações segundo um modelo espiral (Figura 13), ao longo do qual os requisitos são analisados, negociados entre as partes envolvidas, documentados e avaliados para garantir a conformidade com o modelo corporativo do *Data Warehouse*. O produto de cada iteração pode ser tanto um conjunto mais refinado de requisitos – os quais servem de entrada para iterações subseqüentes, quanto uma versão intermediária (*linha base*) da especificação do *data mart* que reflete a real percepção dos usuários a respeito da aplicação. Como a especificação de requisitos em *data warehouse* espelha o processo de *data warehousing* padrão, o produto final de cada iteração tende a ser cada vez mais próximo de uma *linha base* de requisitos acordada entre as partes, à medida que informações cada vez mais refinadas são retroalimentadas no ciclo, reduzindo o esforço e o tempo gasto com a especificação da solução, o que dá a aparência concêntrica à espiral da Figura 13.

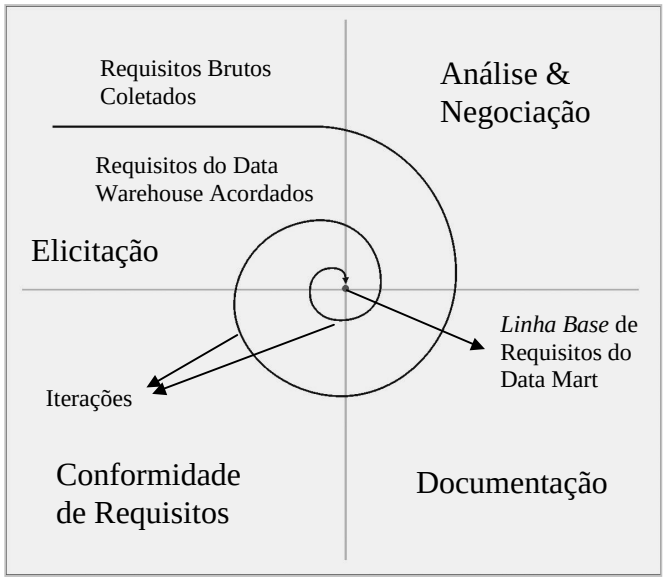


Figura 13. Ciclo de Especificação de Requisitos em *Data Warehouses*.

Projetar sistemas *data warehouse* requer uma preocupação extra com o reuso de requisitos previamente acordados. Seguindo-se o princípio de que a construção do *data warehouse* organizacional envolve um processo de desenvolvimento e integração a longo prazo de *data marts* individuais, a única forma de assegurar uma integração eficiente é especificar requisitos multidimensionais (fatos e dimensões) comuns ao esquema corporativo de maneira tal que estes possuam o mesmo significado em todos os *data marts*.

Dessa forma, requisitos novos devem ser postos em conformidade com os requisitos já incorporados ao *data warehouse*, evitando assim redundâncias e garantindo, ao mesmo tempo, aderência ao modelo global. Processo similar pode ser aplicado aos procedimentos e regras de negócio surgidos ao longo das etapas de construção dos *data marts*. O feedback e a conformidade adequados são providos nesse ponto pela fase de Conformidade de Requisitos (Seção 4.5.4), na qual os requisitos do sistema são comparados com um padrão global para verificar sua aderência e possibilidade de reuso. As seções seguintes descrevem os (sub) processos que compõem a fase de especificação de requisitos em sistemas data warehouse.

4.5.1 Elicitação

Essa fase objetiva implementar um processo de descoberta de requisitos para o *data warehouse*, com ênfase em aspectos multidimensionais, por meio da comunicação com as partes interessadas. Da mesma forma que em sistemas convencionais, a fase de elicitação de requisitos em *data warehouses* requer conhecimento do domínio da aplicação e organizacional de ambos usuários e engenheiros de software. Para auxiliar nessa atividade, propomos as seguintes técnicas clássicas de elicitação:

- **Entrevistas.** Levantar questões junto às partes interessadas a respeito das consultas analíticas a serem executadas é especialmente indicado para amenizar a inabilidade natural dessas partes em descrever, em termos concretos, suas necessidades de suporte à decisão. As respostas obtidas são um meio rico de identificação de requisitos multidimensionais, além de requisitos do sistema como um todo. GIOVINAZZO (2000)

apresenta inúmeros exemplos de sessões de entrevista úteis para a extração de informações, enquanto KIMBALL *et al.* (1998a) fornece recomendações importantes sobre como realizar uma entrevista (Tabela 5).

1	Procure obter informações sobre experiências anteriores com data warehouse na empresa.	**2**	Sempre que possível, entreviste o cliente em seu local de trabalho. Isso evita atrasos e facilita a coleta de documentação do negócio.
3	Alterne entrevistas com representantes de diversas áreas do cliente, procurando confrontar suas respostas antes do final de todas as sessões.	**4**	Nunca faça a pergunta: "*O que você deseja em seu data warehouse*?". Ao contrário, procure entender o negócio e o processo inerente de tomada de decisão.
5	Inicie o processo com uma plenária para apresentação do grupo. Durante os dias de reunião, nunca marque mais do que três entrevistas num dia.	**6**	Elabore um resumo ou ata de reunião poucas horas após a entrevista e repasse a todos para validação.
7	Nunca grave as entrevistas. Sempre copie tudo.	**8**	Não misture chefes e subordinados na mesma sala.
9	Identifique os gargalos de informação para os executivos e os sistemas envolvidos.	**10**	Em caso de dúvida, nunca interprete, sempre confirme com o cliente a resposta correta.

Tabela 5. Dicas para realização de entrevistas.

- ***Workshops*.** A técnica de *workshop* visa reunir as partes interessadas de um projeto por um período curto, mas intensivo, de tempo para discutirem aspectos relevantes para o desenvolvimento da aplicação. O *workshop* é conduzido por um membro da equipe de projeto ou um facilitador externo e requer uma preparação mais especializada do que a de uma simples reunião, o que envolve questões de logística, distribuição prévia do material de trabalho e sensibilização sobre os benefícios da sessão. *Workshops* de Requisitos para *data warehouse* são destinados a encorajar o consenso sobre o foco multidimensional da aplicação; alcançar um acordo rápido entre as partes envolvidas sobre o curso de ações e seqüência de produção dos *data marts*; resolver conflitos de interesse; formar um time de projeto coeso; obter resultados rápidos sobre o escopo do *data warehouse*, principais funcionalidades e restrições operacionais. *Workshop* é uma das técnicas mais poderosas para elicitar requisitos e realmente consegue influenciar o sucesso de um projeto, mesmo quando utilizado sem a presença de outra técnica (LEFFINGWELL e WIDRIG, 2000).

- **Prototipação**. Utilizados dentro do processo como exemplos experimentais do projeto, protótipos demonstram aos interessados como as funcionalidades do sistema irão ajudar na tomada de decisão. Protótipos simulam o comportamento do sistema e esclarecem dúvidas relativas ao *data warehouse*/*data mart* em desenvolvimento tais como performance e amigabilidade das consultas analíticas; cobertura dimensional projetada; usabilidade da interface; e outras características funcionais e não-funcionais. As idéias consolidadas provêm *feedback* essencial para o aperfeiçoamento da aplicação e do esquema multidimensional que lhe serve de base. Os protótipos devem explorar ambos os lados de fatores como consultas complexas/simples; alto/baixo volume de dados; único usuário/múltiplos usuários simultâneos.

- **Cenários.** Desenvolver uma série de cenários de interação ajuda engenheiros de software a clarear e detalhar requisitos funcionais do sistema, sob a forma de casos de uso. As principais transações que compõem uma aplicação *data warehouse* tendem a obedecer a um esquema de casos de uso (UML) como o representado na Figura 14.

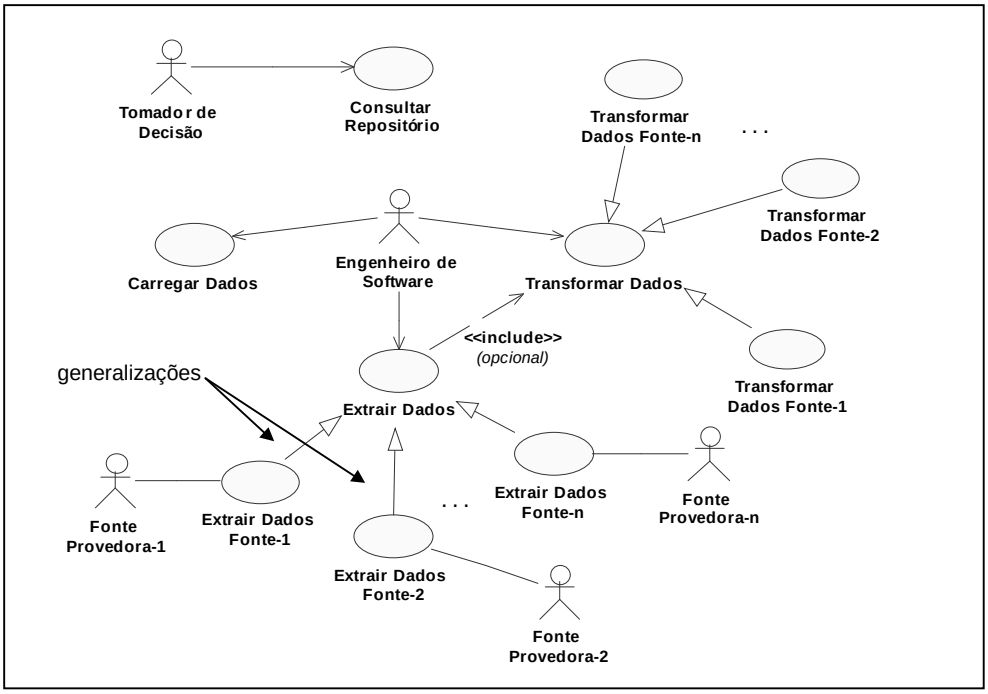


Figura 14. Template de Casos de Uso em UML para Sistemas *Data Warehouse*.

A fase de especificação irá focar em instanciar esse modelo padrão para a realidade do *data warehouse* alvo, e descrever as regras de negócio, procedimentos e funcionalidades de análise que tornam uma aplicação de suporte à decisão única em relação a outra. Conforme evidenciado pelo modelo padrão, os casos de uso permitem o reuso de comportamento comum a diferentes *data marts*.

- **Análise de Requisitos Não-Funcionais**. Tratar requisitos de qualidade durante o projeto de aplicações *data warehouse* requer uma noção das diferentes necessidades e visões de negócio de cada parte interessada. JARKE *et al*. (1999) coloca que os tomadores de decisão estão geralmente interessados na qualidade do dado armazenado, sua atualidade no tempo e facilidade de acesso pelas ferramentas OLAP. Por outro lado, os desenvolvedores estão preocupados com o grau de acessibilidade a dados operacionais; a performance e pontualidade do processo de carga; a consistência do modelo estrela; dentre outros aspectos. Nesse contexto, poucos métodos têm se revelado tão extensíveis e poderosos quanto o *Framework NFR* (CHUNG *et al.*, 2000) para a modelagem desses requisitos não-funcionais. Uma das aplicações possíveis do *Framework* na definição de sistemas *data warehouse* está na escolha de alternativas arquiteturais que melhor satisfaçam a relação "critérios de qualidade do usuário *versus* requisitos funcionais do sistema". A seguir, identificamos e definimos os principais requisitos não-funcionais para aplicações *data warehouse* (Tabela 6). Tendo como base esses requisitos, estendemos o *Framework NFR* para definir um conjunto de Tipos-NFR e catálogos de métodos de operacionalização específicos para ambientes *data warehouse*, e o denominamos *Data Warehouse-Extended NFR Framework* (DW-ENF). Os catálogos e tipos do *framework* DW-ENF podem ser reusados durante a fase de especificação de requisitos para investigar alternativas de projeto que melhor acomodam os requisitos coletados em função das restrições de qualidade impostas pelo cliente (PAIM e CASTRO, 2002a). Adicionalmente, a relação dos requisitos não-funcionais para *data warehouse* (e seus requisitos de qualidade derivados) serve como um guia para a montagem do artefato **Especificação de Requisitos Não-Funcionais**, conforme descrito na Seção 4.5.3.

1. Performance		
O grau de eficiência com que a arquitetura do *data warehouse* responde a um pedido de processamento.		
1.1	**Tempo**	Performance ótima de tempo da arquitetura *data warehouse*.
1.1.1	**Tempo de Resposta**	Tempo de Resposta pequeno ou reduzido para as consultas analíticas.
1.1.2	**Tempo de Processamento**	Tempo de Resposta pequeno ou reduzido para o processamento em *backstage*.
1.2	**Espaço**	Utilização eficiente da memória de processamento.
1.2.1	**Memória Principal**	Grau de otimização do uso da memória principal.
1.2.2	**Memória Secundária**	Grau de otimização do uso da memória secundária.
2. Segurança		
O grau de proteção da informação e comportamento confiável do data warehouse e dos seus dados.		
2.1	**Integridade**	Grau de precisão e validade dos dados multidimensionais.
2.1.1	**Acurácia**	Precisão dos dados armazenados e resultados de agregação.
2.1.1.1	**Consistência**	Coerência lógica dos dados do *data warehouse*.
2.1.1.1.1	**Aderência ao Domínio**	Adequação dos dados aos padrões do domínio.
2.1.1.1.2	**Agregabilidade**	Habilidade de agregar corretamente os dados ao longo das dimensões.
2.1.1.2	**Minimalidade**	Grau até o qual redundância indesejada é evitada durante o processo de integração.
2.1.2	**Corretude**	Extensão com que a especificação do *data warehouse* mapeia as fontes de informação para satisfazer as necessidades do usuário.
2.1.2.1	**Rastreabilidade**	Capacidade de associar os requisitos das partes interessadas ao esquema do *data warehouse*.
2.1.3	**Completude**	Grau com que o conhecimento essencial do *data warehouse* é implementado de forma apropriada no esquema multidimensional e dados armazenados.
2.2	**Disponibilidade**	Grau com que a fonte provedora ou o *data warehouse* está perfeitamente disponível para uso por todos os *stakeholders*.
2.2.1	**Confiabilidade**	Percentual de tempo em que a fonte de dado ou o sistema *data warehouse* permanence disponível para uso considerando-se aspectos de maturidade, tolerância a falhas e recuperabilidade.
2.2.2	**Distributividade**	Capacidade do *data warehouse* de atingir todos os tomadores de decisão.
2.3	**Confidencialidade**	Capacidade do *data warehouse* de oferecer proteção contra o acesso não-autorizado.
3. Multidimensionalidade		
Habilidade de representar os requisitos de suporte à decisão e prover acesso ao dado dimensional e factual.		
2.1	**Conformidade**	Habilidade de representar aspectos comuns do *data warehouse* de forma idêntica ao longo de toda a arquitetura do sistema.
2.2	**Integrabilidade**	Capacidade de integrar adequada e eficientemente informação operacional.
2.2.1	**Pontualidade**	Grau com que a frequência de atualização dos dados atende às necessidades dos usuários.
2.3	**Accessibilidade**	Possibilidade de acesso ao dado para consulta.
2.4	**Interpretabilidade**	Extensão com a qual os dados podem ser interpretados para modelar eficientemente o *data warehouse*.
2.4.1	**Interpretabilidade da Documentação**	Grau com que a documentação na fonte provedora é entendível.
2.4.2	**Interpretabilidade do Dado**	Grau de confiabilidade da descrição do dado.
4. Usabilidade		
Grau com que o software *data warehouse* é de fácil uso.		
4.1	**Operacionalidade**	Facilidade com que o *data warehouse* pode ser operado.
4.2	**Flexibilidade**	Extensão com que as características do *data warehouse* facilitam consultas *ad hoc*.
4.3	**Capacidade de Aprendizagem**	A capacidade física e intelectual requerida para aprender a utilizar a aplicação.

Tabela 6. Principais Requisitos Não-Funcionais para Data Warehouse.

Cabe ressaltar que, para todas as técnicas anteriormente propostas, especialistas de domínio devem auxiliar o trabalho dos engenheiros de requisitos, de forma que não apenas os requisitos necessários, mas também os requisitos **corretos** sejam identificados.

4.5.2 Análise & Negociação

Requisitos descobertos na fase de elicitação, juntamente com o modelo de requisitos já existente, são entrada para um processo detalhado de análise que visa checar os requisitos quanto a omissões, conflitos, sobreposições e inconsistências. Os artefatos gerados devem ser revisados para garantir que a especificação segue os padrões de qualidade e principais diretrizes do projeto multidimensional, e que a justa equivalência entre aspectos conceituais e arquiteturais esteja sendo mantida. A equipe de projeto deve assegurar que os requisitos definidos com o cliente estejam coerentes com o esquema multidimensional projetado e, principalmente, com as limitações da ferramenta OLAP escolhida. Em contrapartida, a análise permite identificar pontos de ajuste na concepção multidimensional adotada até o momento, evitando que falhas possam vir a comprometer a estabilidade do projeto em fases avançadas de seu desenvolvimento.

Uma técnica importante para dar apoio na execução dessa fase são as Listas de Verificação (*Checklists*) de Requisitos. Por *lista de verificação* definimos uma lista de perguntas dirigida a examinar cada requisito por meio da leitura dos documentos que os registram. A lista pode ser implementada como uma tabela na qual as linhas representam itens nomeados com identificadores dos requisitos, como o exemplo para sistemas *data warehouse* ilustrado na Tabela 7.

4.5.3 Documentação

Essa fase é o coração de nossa metodologia. O propósito aqui é prover uma documentação completa e detalhada dos requisitos elicitados, de maneira a torná-los compreensíveis a todos os interessados. Em verdade, a documentação dos requisitos é gerada ao longo do processo de desenvolvimento, não estando restrita a esta fase especificamente (vide Figura

15). Dentro de nossa abordagem, desenvolvemos um conjunto de *templates* de artefatos para acomodar os requisitos funcionais e não-funcionais de um *data warehouse*, e seus requisitos de domínio correlatos.

Item	Questionamento
Agregação Automática	Todos os níveis de dimensão definidos garantem uma completa e direta *agregação automática* (adição, soma, contagem,...), em termos do esquema multidimensional elaborado?
Representação de Fatos e Dimensões	Todas as necessidades das partes interessadas estão representadas no esquema multidimensional projetado no que diz respeito a fatos e dimensões?
Conexão entre Fatos e Dimensões	O conjunto total de níveis dimensionais está correta e completamente associado ao conjunto de fatos a ser analisado?
Completude da Integração	Todos os procedimentos e requisitos de integração encontram-se definidos de forma a possibilitar a correta incorporação dos dados provenientes de fontes externas?
Qualidade da Documentação	Todos os documentos elaborados servem como ferramentas efetivas para modelar as necessidades dos usuários em termos de requisitos, dentro dos padrões de qualidade estabelecidos pela empresa?
Requisitos Desnecessários	O requisito documentado corresponde a uma real necessidade dos usuários ou serve apenas como um "embelezamento" do sistema?
Ambiguidade de Requisitos	Existe ambiguidade entre os requisitos, i.e., um dado requisito pode ser interpretado de formas diferentes por pessoas diferentes? Quais são as possíveis interpretações de cada requisito e como isso pode ser melhor trabalhado no projeto?
Testabilidade dos Requisitos	Todos os requisitos são testáveis, ou seja, eles estão declarados de forma tal que testes podem ser derivados para comprovar que o sistema atende aos requisitos do usuário?
Conformidade dos Requisitos	Podemos na prática navegar (*drill*) entre tabelas-fato através do esquema de dimensões comuns sem incorrer em perda de dados ou inconsistência nas consultas?

Tabela 7. Exemplo de Lista de Verificação de Requisitos em Sistemas Data Warehouse.

Os *templates* são meta-documentos que descrevem o seu próprio propósito e uso apropriado (vide Apêndice 1). Os artefatos foram inspirados no modelo de documentação para a fase de Requisitos proposto no *Rational Unified Process* (RUP) (KRUCHTEN, 1999) e na abordagem para organização da especificação de requisitos em uma *família de produtos* de LEFFINGWELL e WIDRIG (2000), a qual se demonstra bastante apropriada à filosofia de

construção de *data warehouses* a partir de *data marts* individuais. A seguir, descrevemos cada um desses artefatos.

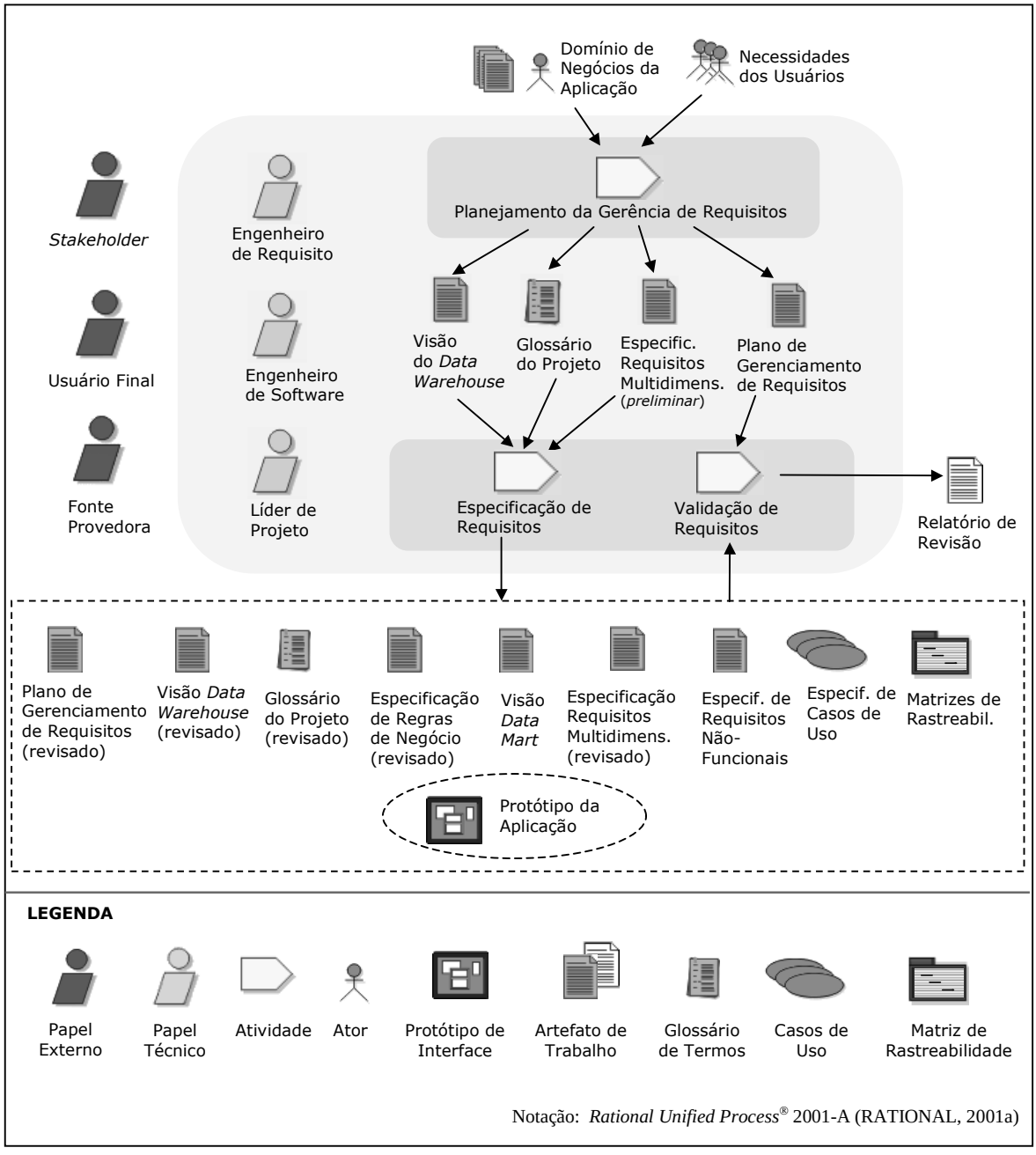


Figura 15. Descrição do Processo de Documentação.

- **Plano de Gerenciamento de Requisitos**. Documenta aspectos de gerência essenciais ao controle da engenharia de requisitos do projeto, conforme discutido na Seção 4.4.

- **Glossário**. Coleta e organiza toda a terminologia e conceitos específicos dos domínios de problema tratados no corpo do *data warehouse* da organização. O principal objetivo do Glossário é melhorar o entendimento comum de termos fundamentais para o desenvolvimento da aplicação entre todas as partes envolvidas.

- **Visão do *Data Warehouse***. Descreve os requisitos gerais do *data warehouse* corporativo, incluindo considerações sobre sua motivação e escopo estratégico atacado; objetivos e diretrizes de projeto; e outros aspectos referentes ao sistema como um todo.

- **Visão do *Data Mart***. Coleta as necessidades de alto nível dos usuários dos *data mart*, além das funcionalidades e atores que dão suporte a essas necessidades. Ademais, o artefato cobre os principais marcos de projeto a serem atendidos em cada etapa de construção dos *data mart*.

- **Especificações de Caso de Uso**. Detalham os passos para a implementação das funcionalidades projetadas para o *data warehouse/data mart*, representando a seqüência de operações que devem ter lugar até que um resultado final seja obtido.

- **Especificação de Requisitos Multidimensionais**[10]. Especifica os aspectos inerentes à multidimensionalidade da aplicação, incluindo a relação de fatos e dimensões que compõem o esquema dimensional e considerações sobre sua granularidade, aditividade, e conformidade à arquitetura global.

[10] Esse artefato foi acrescido a partir da consolidação de informações contidas anteriormente nos documentos de Visão, conforme descrito em (PAIM *et al*., 2002), devido à importância e volume desses requisitos e para facilitar o estudo de questões como a conformidade de fatos e dimensões.

- **Especificação de Requisitos Não-Funcionais**. Complementa as especificações de caso de uso, descrevendo os requisitos não-funcionais da aplicação que não são cobertos pelo modelo de casos de uso. O artefato documenta ainda restrições de projeto e limitações legais.

- **Especificação de Regras de Negócio**. A função desse documento é registrar todas as regras de negócio que regulam a operacionalização das funcionalidades dos *data mart*. Adicionalmente, conforme o volume de regras identificadas, documentos distintos, orientados por assunto, podem ser produzidos, para agrupar categorias específicas de regras.

  Os quatro últimos artefatos constituem juntos o que foi referenciado no Capítulo 2 como "*Moderno Documento de Requisitos de Software*".

- **Relatório de Revisão**. Um relatório simples descrevendo as ações acordadas após uma sessão de validação de requisitos (vide Seção 4.5.2).

- **Matrizes de Rastreabilidade**. Artefato primordial para a gerência de mudanças em requisitos do sistema, conforme descrito na seção 4.7.

4.5.4 Conformidade de Requisitos

Conformidade de requisitos é uma etapa particular da especificação de *data warehouses*. Projetos *data warehouse* somente podem ser bem-sucedidos se suas peças de montagem fazem sentido em dois universos interdependentes: a visão orientada a assunto dos *Data Marts* e o esquema global do *Data Warehouse*. Tentativas de definir peças isoladas do *data warehouse* que, ao final, não podem ser funcionalmente integradas concorrem fortemente para o fracasso do projeto. Antes de construir um *data mart* departamental ou o *data warehouse* de toda a organização, é imperativo que a equipe de desenvolvimento considere a arquitetura multidimensional proposta dentro de uma visão corporativa de todos os dados da organização. Dentro desse princípio, KIMBALL *et al.* (1998a) coloca que, quando se espera construir um *data warehouse* que seja robusto e resistente em face à contínua

evolução dos seus requisitos, deve-se aderir a uma especificação de *data mart* na qual dimensões e fatos comuns estão em conformidade entre todos os *data marts*.

Uma dimensão está em conformidade quando tem o mesmo significado para toda tabela-fato à qual está ligada. A dimensão "Tempo" é um exemplo clássico de dimensão em conformidade, visto que uma das funções primárias em um *data warehouse* é a soma de métricas de fato ao longo do tempo. De maneira similar, um fato está em conformidade com o esquema global do *data warehouse* se a mesma terminologia para representação do seu conteúdo é usada ao longo de todos os *data marts* constituintes. Exemplos típicos de fatos que requerem conformidade são *vendas do produto, lucro obtido, preços padrão* e *custos padrão*. Quando presentes em mais de uma tabela-fato, esses fatos devem possuir o mesmo formato, mesma regra de formação, mesma lei de cálculo e definidos no mesmo contexto dimensional.

Em nossa abordagem, o conceito de conformidade é estendido a um patamar maior de abstração, onde todos os requisitos comuns do sistema são postos em conformidade. Definimos que um requisito está em conformidade se este é comum a vários (ou todos os) *data marts* e é descrito identicamente em cada uma das diferentes visões de assunto do *data warehouse*. Mais do que apenas aspectos multidimensionais, requisitos em conformidade respondem por cada funcionalidade, característica ou restrição ao desenvolvimento do sistema que guarda o mesmo raciocínio por todo o projeto, devendo ser, portanto, representados de forma única. Em outros termos, conformidade de requisitos é mais uma das múltiplas facetas do reuso de requisitos.

Requisitos em conformidade trazem os seguintes benefícios para a especificação de sistemas *data warehouse*:

(i) Evitam redundância e ambiguidade entre requisitos que permeiam o *data warehouse* como um todo;

(ii) Permitem que dimensões comuns sejam relacionadas com múltiplos fatos no mesmo espaço de banco de dados (*database space*);

(iii) Em conjunto com abordagens orientadas a cenário (ex. *casos e uso*), possibilita o reuso de conhecimento previamente acordado no escopo do projeto, promovendo, dessa forma, a melhoria da qualidade;

(iv) Melhora a consistência das interfaces do usuário e do conteúdo dos dados agregados onde quer que seja feito uso do modelo comum (em conformidade);

(v) Possibilita operações *drill-across* entre *data marts*, i.e., a recuperação de dados factuais a partir de tabelas-fato localizadas em diferentes visões de assunto dentro do *data warehouse* da organização;

(vi) Facilita a integração requerida entre *data marts*, permitindo que a arquitetura multidimensional trabalhe como um todo único;

(vii) Propicia escalabilidade e facilita a evolução do *data warehouse*;

(viii) Facilita a aderência a padrões de projeto e organizacionais.

Cabe ao time de projeto estabelecer, publicar, manter e reforçar a conformidade de requisitos. Após cada fase de documentação, todos os documentos de especificação devem ser analisados para identificar aqueles requisitos que representem aspectos gerais do *data warehouse*. Engenheiros de software devem estar atentos para reconhecer sobreposições e similaridades entre requisitos e proceder a ajustes na sua especificação para mantê-los em conformidade, quando possível, com um modelo de requisitos global, promovendo inclusive requisitos a artefatos de mais alto nível de abstração se necessário.

4.6 Validação de Requisitos

Considerando-se uma aplicação *data warehouse*, é provável que, após várias iterações executando as fases previamente descritas, alguns erros de interpretação e/ou concepção equivocadas a respeito das capacidades analíticas a serem entregues ainda persistam. Muitas vezes, nem o grupo de usuários nem a equipe de desenvolvimento estão confiantes o suficiente do que o produto sendo entregue é capaz de fazer.

Para validação do que foi definido como a especificação para o *data warehouse*, propomos a junção entre as técnicas de *Revisão* e *Prototipação* como uma estratégia efetiva para detectar e remover defeitos na especificação, antes que estes se tornem parte do "pacote *data mart*". Durante a reunião de revisão, a *linha base* final do *data mart* é apresentada para todos os envolvidos, e descrita em termos de seus requisitos funcionais e de qualidade. O protótipo desenvolvido na ferramenta OLAP auxilia o reconhecimento pelos usuários dos aspectos arquiteturais que implementam os requisitos especificados.

Quando problemas são identificados, a especificação é revista no contexto do requisito funcional, não-funcional ou do domínio multidimensional que dá origem à inconsistência. O time de validação deve gerar imediatamente uma lista de ações em resposta a cada um dos casos, e acordar entre os envolvidos com as ações propostas. O processo de desenvolvimento retorna à fase de especificação, onde as ações são aplicadas para adequar a especificação dos requisitos às definições corretas. É importante que as ações considerem tanto uma revisão conceitual quanto arquitetônica do esquema implementado. O time de prototipação, juntamente com os especialistas em banco de dados, deve efetuar uma análise dos impactos das alterações acordadas no esquema multidimensional implementado. Outro fator importante durante a validação dos requisitos do *data warehouse* é incluir especialistas de domínio que não estiveram envolvidos com a engenharia dos requisitos do sistema. Revisores externos enriquecem o processo de análise, na medida em que não estão atrelados a noções preconcebidas da solução.

4.7 Controle da Gerência de Requisitos

Requisitos não podem ser gerenciados eficientemente sem que haja *rastreabilidade* (TORANZO, 2002). O IEEE (IEEE, 1984) define que uma especificação de requisitos de software é rastreável se: (*a*) a origem de cada um dos seus requisitos é clara e (*b*) se esta facilita a referência de cada um dos requisitos no desenvolvimento futuro. Conforme evidenciado ao longo desse trabalho, a rastreabilidade e a gerência de mudanças durante o

desenvolvimento de sistemas *data warehouse* devem ser conduzidas em todas as fases, sobretudo nas esferas de requisitos e arquitetural.

A primeira visão tem como meta a gerência das mudanças ocorridas em requisitos acordados e análise dos impactos no modelo de requisitos. A segunda visão estende essa investigação à arquitetura do banco de dados, para identificar de forma complementar os impactos que as mudanças provocam no esquema multidimensional do *data warehouse*. Ferramentas de suporte existem para ambas as atividades de investigação (*Doors®* (TELELOGIC, 2002), *Requisitepro®* (RATIONAL, 2001), *SLATE®* (EDS, 2002), *System Architect®* (POPKINS, 2002), *Oracle9iDesigner®* (ORACLE, 2002a)). A utilização desses instrumentos torna-se obrigatória em ambientes *data warehouse*, em vista de envolver grandes quantidades de requisitos e atributos de banco de dados. A escolha das ferramentas corretas irá depender de inúmeros fatores como *suporte local*, *banco de dados adotado no projeto, limitações da ferramenta, integração entre aplicativos*, dentre outros.

Matrizes de Rastreabilidade são o componente mais largamente utilizado em ferramentas de suporte à gerência de requisitos para demonstrar dependência entre requisitos (TORANZO, 2002). Linhas e colunas da matriz representam os requisitos do sistema, enquanto um símbolo marca a relação de dependência entre eles. Ao ler uma das linhas da matriz, pode-se constar todas as dependências para um dado requisito. A análise de impacto implica uma comparação biunívoca. Para tanto, grupos de diferentes matrizes devem ser definidos para permitir a completa investigação dos requisitos (*ex*. necessidades do usuário *versus* funcionalidades; funcionalidades *versus* fatos; fatos *versus* dimensões; fatos *versus* atributos dimensionais; regras de negócio *versus* passos de caso de uso; e muitas outras). Outros recursos como *grupos de discussão, publicação na Web, controle do versionamento de requisitos* e *geração automatizada de relatórios* estão entre o largo espectro de técnicas tornadas disponíveis por ferramentas de requisitos.

Do ponto de vista arquitetural, mudanças no modelo de requisitos afetam diretamente o modelo de dados que governa o repositório em ambas as visões "*data mart*" e "*data*

warehouse". Ferramentas CASE (*Oracle9iDesigner®* (ORACLE, 2002a), *System Architect®* (POPKINS, 2002)) oferecem suporte à análise dos impactos no modelo de dados, incluindo facilidades como busca automatizada dos requisitos dimensionais alocados ao banco de dados e identificação do número (e nível de detalhe) de componentes do banco afetados pela mudança. Essas facilidades reduzem o esforço de análise e execução de manutenções em campos de tabelas no banco de dados, garantindo uma gerência efetiva da solução arquitetural.

4.8 Ciclo de Definição de Requisitos em Sistemas Data Warehouse

As etapas da metodologia definida nas seções anteriores oferecem um arcabouço para a aplicação das melhores práticas em engenharia de requisitos ao desenvolvimento de sistemas *data warehouse*. Contudo, para que essas etapas realmente produzam efeito, elas devem ser integradas ao processo de desenvolvimento da aplicação. A integração entre metodologia e processo de desenvolvimento introduz um ciclo de definição de requisitos próprio para sistemas *data warehouse*, conforme descrito pela Figura 16.

A fase de Planejamento da Gerência de Requisitos deriva as diretrizes para o início do ciclo de definição (vide Seção 4.4). As fases que compõem o núcleo do processo de requisitos (*Elicitação*, *Análise & Negociação*, *Documentação* e *Conformidade de Requisitos*) fornecem o suporte para a execução das atividades de definição dos requisitos (conforme Seções 4.5.1 a 4.5.4). O ciclo se inicia com a definição do escopo do projeto *data warehouse*. Como os requisitos de qualidade têm grande influência no projeto de arquitetura do sistema, a análise dessa influência tem lugar nos primeiros estágios do ciclo, tendo como base o *framework* de requisitos não-funcionais (Seção 4.5.1). Essa atividade perdura durante toda a especificação do *data warehouse* até a seleção de uma solução ideal (PAIM e CASTRO, 2002a).

Estabelecidos os requisitos principais do *data warehouse* da organização, o ciclo segue com o amadurecimento do modelo de requisitos para os *data marts* que o irão integrar. O primeiro estágio nesse sentido é a definição de um escopo para o *data mart*, onde serão

estabelecidas as necessidades e funcionalidades sobre as quais o modelo de requisitos do *data mart* irá se apoiar e que irão integrar a *Visão* de negócio específica.

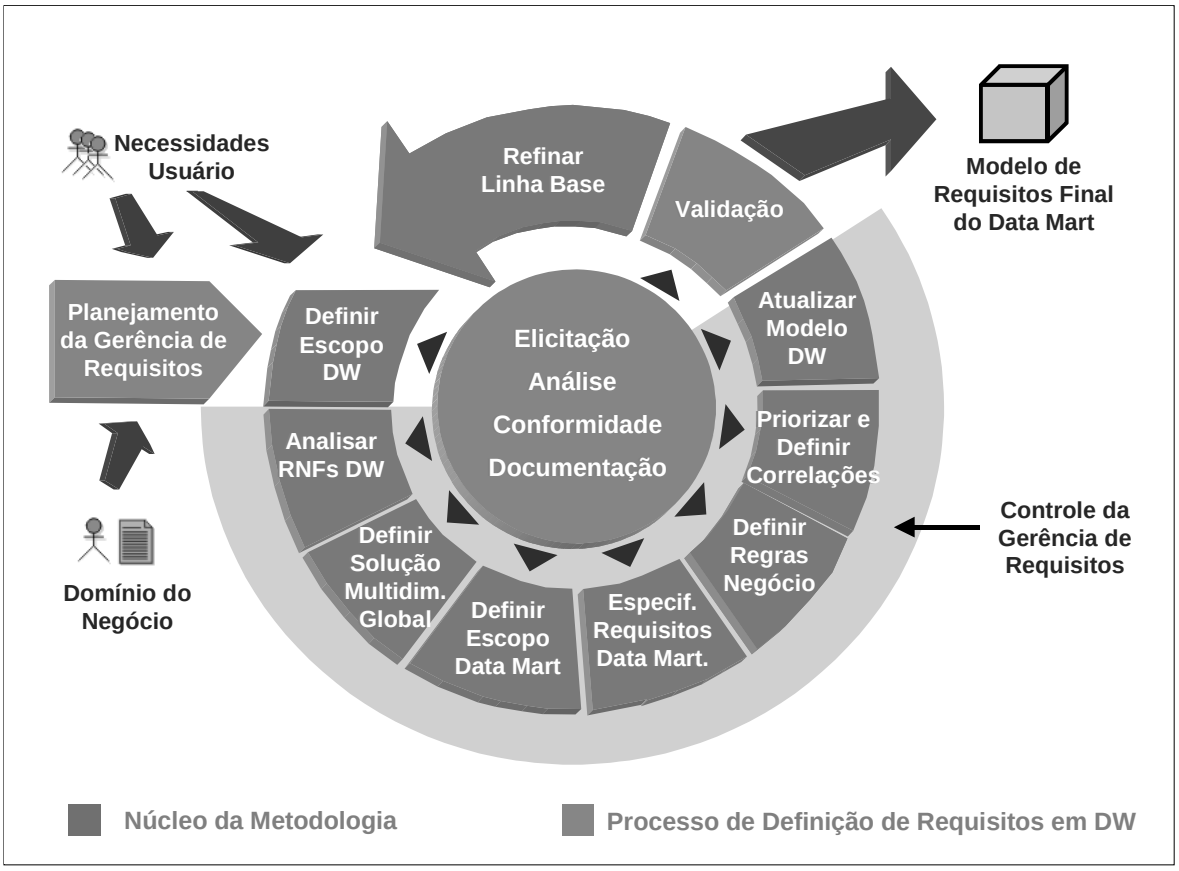


Figura 16. Ciclo de Definição de Requisitos em Data Warehouses.

Os requisitos multidimensionais e funcionais que implementam as funcionalidades divisadas e requisitos de qualidade pré-estabelecidos são capturados, analisados e documentados no modelo de casos de uso, juntamente com as regras de negócio que os regulam (Seção 4.5.3). Em seguida, correlações entre requisitos são estabelecidas para permitir o gerenciamento da *linha base* de requisitos, tendo como suporte, ao longo de todo o processo, os procedimentos e ferramentas fornecidos pela atividade de Controle da Gerência de Requisitos (Seção 4.7). A linha base de requisitos será objeto de validações para o seu refinamento contínuo até a constituição de um modelo de requisitos final do *data mart*. Esse refinamento promove a revisão e adequação do modelo corporativo de requisitos

do *data warehouse*, garantindo a conformidade necessária à sua constante evolução (vide Seção 4.6).

O ciclo anteriormente descrito não pressupõe um processo em **cascata**. Em verdade, as atividades se interpõem de acordo com o nível de maturidade da linha base no tempo. Não obstante, cada atividade faz uso em maior ou menor grau das fases situadas no núcleo da metodologia, segundo a natureza do trabalho a que estão relacionadas. A Tabela 8 a seguir demonstra a importância ocupada por cada uma dessas fases dentro do ciclo de definição de requisitos, conforme nossas experiências iniciais com a implantação da metodologia no Projeto SAFE (vide Tabela 9 e Capítulo 5).

	Fases do Ciclo de Requisitos	Elicit.	Anal.&Neg.	Docum.	Conform.	Produtos
Especificação do *Data Warehouse*	1. Planejamento da Gerência de Requisitos	65%*	30%	5%		PGR, GLO, VDW, ERM
	2. Definir Escopo do Data Warehouse	65%	15%	20%		VDW, GLO
	3. Analisar Requisitos Não-Funcionais	70%	15%	15%		ERNF
	4. Definir Solução Multidimensional Global	40%	50%	10%		ERM, VDW
Especificação do *Data Mart*	5. Definir Escopo do Data Mart	45%	5%	30%	20%	VDM
	6. Especificar Requisitos do Data Mart	25%	20%	25%	30%	CDU, ERNF, ERM, PRO
	7. Definir Regras de Negócio	55%	5%	30%	10%	ERN
	8. Priorizar e Associar Requisitos			100%		MTR
	9. Atualizar Modelo Corporativo			80%	20%	ERM
Validação	10. Validar Linha Base		50%	50%		RR

LEGENDA: *PGR=Plano de Gerenciamento de Requisitos; GLO=Glossário; VDW=Visão do Data Warehouse; VDM=Visão do Data Mart; ERM=Especificação de Requisitos Multidimensionais; ERNF=Especificação de Requisitos Não-Funcionais; CDU=Especificações de Casos de Uso; PRO=Protótipo da Aplicação;MTR=Matrizes de Rastreabilidade; RR=Relatório de Revisão.*

** Média dos percentuais de participação por fase das atividades do núcleo da metodologia, conforme Tabela 8 (Dados arredondados)*

Tabela 8. Freqüência de distribuição do núcleo da metodologia pelo Ciclo de Definição.

Fases do Ciclo de Requisitos	**Data Mart 1** (H/h)*				**Data Mart 2** (H/h)			
	Elicit.	**Anal.**	**Docum.**	**Confrm.**	**Elicit.**	**Anal.**	**Docum.**	**Confrm.**
1. Planejamento da Gerência de Requisitos	109,0	50,5	8,5	0,00	89,0	43,0	7,0	0,00
2. Definir Escopo do Data Warehouse	175,0	40,0	54,0	0,00	144,5	36,0	42,0	0,00
3. Analisar Requisitos Não-Funcionais	94,0	19,0	21,5	0,00	76,0	17,0	19,0	0,00
4. Definir Solução Multidimensional Global	215,0	269,0	54,0	0,00	178,0	218,0	49,0	0,00
5. Definir Escopo do Data Mart	91,0	10,0	61,0	40,5	72,0	12,0	50,0	34,0
6. Especificar Requisitos do Data Mart	301,0	242,0	304,0	362,5	250,0	200,0	250,0	304,0
7. Definir Regras de Negócio	222,0	21,0	121,0	40,5	184,0	17,0	101,0	36,0
8. Priorizar e Associar Requisitos	0,00	0,00	202,0	0,00	0,00	0,00	167,0	0,00
9. Atualizar Modelo Corporativo	0,00	0,00	81,0	20,5	0,00	0,00	67,0	15,0
10. Validar Linha Base	0,00	66,0	67,0	0,00	0,00	54,0	57,0	0,00
TOTAL NO CICLO	1207,00	717,50	974,00	464,00	993,50	597,00	809,00	389,00

** Quantidade de Homem/Hora gasta com a atividade-núcleo na fase (dados:Projeto SAFE/SERPRO (vide capítulo 5)).*

Tabela 9. Horas gastas no Estudo de Caso para executar atividades do núcleo da metodologia por fase do Ciclo de Requisitos.

4.9 Considerações Finais

Nesse capítulo mostramos como a integração entre o processo de engenharia de requisitos e o desenvolvimento de sistemas *data warehouse* pode ser alcançada com a definição de uma metodologia para a coleta, análise, especificação e gerenciamento de requisitos, ao longo de uma seqüência de fases. Na medida em que instanciam o processo tradicional de engenharia de requisitos para as características específicas de ambientes *data warehouse*, as fases da metodologia congregam requisitos de sistema para a modelagem de uma solução em estreita aderência a metas estratégicas dos usuários finais.

Ao definir um conjunto de artefatos, procedimentos e ferramentas para apoio à aplicação interativa de cada uma de suas fases, a abordagem aqui proposta não só fornece mecanismos para a correta definição de sistemas de suporte à decisão, como também possibilita uma gerência constante sobre a transposição da solução definida para o mundo operacional. No capítulo seguinte exemplificaremos como os princípios da metodologia podem ser aplicados ao desenvolvimento de um *data warehouse* de grande porte e as contribuições advindas desse processo.

Capítulo 5

Estudo de Caso

Este capítulo apresenta um estudo de caso conduzido na empresa SERPRO com o objetivo de validar a utilização da metodologia proposta na especificação e gerência dos requisitos de um projeto data warehouse de grande porte. Descrevemos, inicialmente, a empresa e o projeto que serviu de motivação para o estudo. Em seguida, detalhamos a utilização da metodologia para a definição da solução data warehouse. Finalmente, os resultados desse caso prático são descritos, assim como as principais lições aprendidas.

5.1 Introdução

Apresentamos nesse capítulo um exemplo de aplicação real da metodologia proposta na especificação e gerência dos requisitos de um projeto *data warehouse* de grande porte, conduzido pela empresa SERPRO para o grande cliente na administração direta federal.

O projeto, intitulado Sistema de Análises Fiscais e Estratégicas - SAFE, objetiva criar um repositório contendo informações sobre as atividades de fiscalização e administração de tributos federais. SAFE permitirá ao alto escalão administrativo do cliente realizar consultas analíticas para subsidiar um processo de tomada de decisão que propicie a melhoria do desempenho de suas atividades, em suas várias áreas de atuação. O projeto é conduzido no SERPRO pela unidade de desenvolvimento regional da Superintendência de Negócios e Arrecadação Tributária – SUNAT, em Recife.

Um dos maiores desafios do projeto é desenvolver uma especificação de requisitos que reflita o atendimento de todas as necessidades estratégicas do cliente e, ao mesmo tempo, assegure a escalabilidade do projeto em vista da diversidade dos assuntos incorporados ao *data warehouse*. A complexidade do projeto requer que uma gerência efetiva dos requisitos seja implementada como única forma de administrar a multiplicidade dos requisitos do sistema.

Esse capítulo descreve como a aplicação da metodologia objeto dessa dissertação tem auxiliado a equipe de projeto da SUNAT-Recife na construção do sistema SAFE e superação dos desafios impostos pela definição e gerência de seu modelo de requisitos. O capítulo inicia-se apresentando a empresa SERPRO (Seção 5.2) e a SUNAT (Seção 5.3). Em seguida, a estrutura e objetivos do projeto SAFE são descritos em maiores detalhes (Seção 5.4). A experiência com a aplicação da metodologia e os resultados alcançados são relatados na Seção 5.5. A seção 5.6 tece considerações sobre a utilização da metodologia para a definição dos demais *data mart* do projeto SAFE. Na última sessão, apresentamos nossas considerações finais.

5.2 A Empresa SERPRO

O SERPRO é uma empresa de informática vinculada ao Ministério da Fazenda, cuja função principal é a execução de serviços de tratamento de informações e processamento de dados para o governo federal. A Empresa está sediada em Brasília, com representações regionais em 10 capitais (Belém, Belo Horizonte, Curitiba, Fortaleza, Porto Alegre, Recife, Rio de Janeiro, Salvador, São Paulo e a própria capital federal) e um quadro em 2003 de 8.774 funcionários, sendo 2.472 desenvolvedores. Como empresa de informática, a visão do SERPRO é prover "*o melhor em tecnologia da informação para o sucesso dos clientes*". Essa visão se coaduna com a missão maior da empresa de "*fornecer soluções baseadas em tecnologia da informação para êxito das decisões e operações de seus clientes, com inovação, qualidade e segurança, a preços competitivos*".

A Empresa encontra-se estruturada em Unidades de Gestão (UG) responsáveis por um segmento da Administração Pública. Cada segmento atende a pelo menos um órgão federal (com características e necessidades próprias), por intermédio de projeções (*representações da unidade corporativa*) da UG em cada uma das unidades organizacionais. A natureza diversa desses clientes e a descentralização do desenvolvimento em suas diversas representações exigem do SERPRO a manutenção de um complexo parque de desenvolvimento e o envolvimento direto com as mais diferentes plataformas, incluindo tecnologias inovadoras como *data warehouse*.

5.3 A SUNAT – Superintendência de Negócios e Administração Tributária

A Superintendência de Negócios de Administração Tributária - SUNAT é uma Unidade de Gestão de Negócios do SERPRO, criada em primeiro de agosto de 1995 com a missão de tornar disponível ao Governo Federal, todas as informações pertinentes à Arrecadação, Tributação e Fiscalização de impostos federais de forma integrada. A SUNAT tem a preocupação com o desenvolvimento de seus produtos e serviços permanentemente dentro dos requisitos de qualidade exigidos, a saber: segurança, confiabilidade, precisão nos

dados, rapidez na resposta, respeito às peculiaridades do cliente, eficiência nos controles de qualidade, diversidade de tecnologia de comunicação, disponibilidade das informações, inovação e criatividade.

Essa exigência, aliada ao processamento de um volume maciço (mais de 500 milhões de registros/mês) de informações que requerem sigilo e segurança, impulsionam a SUNAT à adoção de tecnologias de ponta, como a Internet e *Data Warehouse*, para facilitar o processo de gestão e a tomada de decisão do seu cliente. O complexo ambiente desse cliente exige, em contrapartida, a criação de uma infra-estrutura diferenciada e a implantação das melhores práticas de gestão do processo de desenvolvimento, como forma de assegurar a qualidade do produto final. Nesse sentido, destacam-se as iniciativas pioneiras da SUNAT na institucionalização do Processo SERPRO de Desenvolvimento de Soluções (PSDS) (PAIM e TAVARES, 2002); na criação e difusão de uma política organizacional para a gerência de requisitos no SERPRO (CARVALHO, 2002); e na implantação e qualificação da sua unidade Recife no nível 2 de maturidade do *Capability Maturity Model* (CMM) do SEI – *Software Engineering Institute* (TAVARES *et al.*, 2002).

5.4 O Projeto SAFE

O ambiente operacional do cliente contempla a execução de uma ampla gama de atividades que vão desde a coleta, fiscalização e administração dos tributos federais até a gestão de seus processos internos. Administrar esse ambiente complexo requer a construção de soluções de automação para as mais variadas áreas de negócio. O SERPRO tem contribuído ao longo de 38 anos para o desenvolvimento de soluções de software que agilizam o trabalho de auditores fiscais e técnicos do governo. Todavia, a alta administração do cliente tem percebido que, para atingir o alto grau de excelência desejado na execução de suas atividades e prestação de serviços à população, é necessária uma visão integrada do trabalho realizado em seus diversos segmentos de negócio.

O Projeto SAFE foi concebido com a premissa de dotar o cliente de uma visão integrada e corporativa de seus resultados operacionais para a tomada de decisões estratégicas. O *Data*

Warehouse desenvolvido coleta e armazena dados provenientes das inúmeras bases de sistema do cliente, localizadas no ambiente de produção controlado pelo SERPRO. Os dados são agregados numa perspectiva orientada a assunto para permitir a realização de consultas OLAP complexas. *Assunto* é definido como uma área de negócio chave do cliente. Os assuntos são modelados por meio de soluções *data mart* distintas, integradas ao *data warehouse* corporativo.

Tabelas-Fato armazenam as métricas de fatos relativos ao desempenho do cliente em cada um dos assuntos selecionados para compor o *data warehouse*. Conectadas a cada tabela-fato, dezenas de tabelas-dimensão compõem modelos estrela interconectados, formando uma complexa arquitetura *data warehouse*. O sistema foi desenvolvido numa arquitetura em três camadas. O servidor de aplicação Web publica instâncias de interface OLAP geradas a partir da ferramenta *Microstrategy®* (MICROSTRATEGY, 2003), as quais interligam os usuários finais ao repositório de dados e permitem operar consultas ao *warehouse* a partir de qualquer ponto dentro da intranet do cliente, num esquema "24 X 7" (24 horas por dia, 7 dias por semana).

5.5 Aplicando a Metodologia ao Projeto SAFE

O Projeto SAFE foi totalmente desenvolvido, desde o seu início em janeiro de 2001, segundo as diretrizes da metodologia aqui proposta. Devido a restrições de tempo impostas pelo cliente, a estratégia escolhida para os primeiros *data mart* desenvolvidos foi atacar os requisitos mais críticos, i.e., aqueles relacionados com pontos críticos no sistema, a saber: (*a*) integração com sistemas provedores; (*b*) levantamento e especificação de requisitos multidimensionais; e o (*c*) controle da mudança de requisitos pelo usuário. Os processos e artefatos especificados na metodologia foram, então, intensivamente aplicados para resolver os pontos críticos, com ênfase nas atividades de especificação e controle da gerência de requisitos. O *data mart* escolhido como piloto corresponde à Visão “Ação Fiscal PJ” do *data warehouse*, a qual consolida fatos sobre as ações de fiscalização sobre o contribuinte pessoa jurídica. O desenvolvimento seguiu duas grandes iterações, com uma

linha base intermediária para validação pelo cliente. A equipe de projeto foi composta de oito desenvolvedores - incluindo um líder de projeto, um especialista em *data warehouse* e um Administrador de Banco de Dados (DBA) Projetista - e dois engenheiros de requisitos (nos quais eu me incluo), responsáveis pelo suporte à implantação da metodologia. Após a experiência bem-sucedida com o primeiro *data mart*, o cliente solicitou estudos para a viabilização de mais sete *data marts*, expandindo o escopo inicial do projeto SAFE, que passou a envolver duas equipes distintas em dois grandes pólos de desenvolvimento da SUNAT: Recife e Rio de Janeiro. No prazo de um ano (fevereiro de 2002), dois *data marts* foram concluídos com sucesso (Visões "Ação Fiscal PJ" e "Ação Fiscal PF"), e o projeto segue agora com o desenvolvimento e integração ao *data warehouse* corporativo de mais dois *data marts* (Visões "Arrecadação" e "Revisão Parametrizada").

A seguir, descrevemos o processo de implantação e as contribuições da metodologia para a construção do modelo de requisitos do projeto SAFE, em sua Visão "Ação Fiscal PJ", desenvolvida pela SUNAT-Regional Recife.

5.5.1 Planejando a Gerência dos Requisitos

Antes de dar início à especificação do sistema SAFE, equipe de projeto e partes interessadas procuraram estabelecer um entendimento comum sobre o escopo e objetivos do *data warehouse*, e sobre as regras que iriam direcionar o seu desenvolvimento. Essa fase envolveu representatos do cliente numa experiência inovadora de discutir metas, critérios gerenciais e aspectos técnicos do *data warehouse* em conjunto com o corpo técnico do SERPRO.

A discussão teve lugar durante um *workshop* realizado entre os dias 16 e 20 de Janeiro de 2001. As diferenças de conhecimento e perfis entre os participantes foram compensadas pela novidade do tema *data warehouse* para a grande maioria. De fato, a experiência revelou-se um passo crítico no sentido de evitar concepções errôneas a respeito do *data warehouse* que poderiam comprometer o seu desenvolvimento se descobertas em fases posteriores à Definição dos Requisitos. O especialista em *data warehouse* teve um papel

fundamental no esclarecimento dos principais conceitos sobre a tecnologia, de forma a nivelar o entendimento de todos sobre as potencialidades e limitações da aplicação SAFE. O nivelamento facilitou a definição das regras para a identificação e gerência dos requisitos do projeto, atividade coordenada pelos engenheiros de requisito e documentada utilizando-se o *template* (modelo) para o *Plano de Gerenciamento de Requisitos* apresentado no Apêndice 1. O conteúdo final do Plano[11] está descrito na Figura 17. Ao final do *workshop*, pôde-se perceber não somente um aumento da confiança dos clientes no projeto, mas também uma considerável redução na distância entre estes e a equipe técnica. O workshop também serviu para demonstrar a importância de um *Glossário de Termos* da aplicação para o nivelamento do entendimento entre as partes, considerando-se a ampla gama de domínios de negócio a serem cobertos pelo sistema e a novidade de certos termos inerentes à filosofia *data warehouse*.

[11] Por razões de espaço e sigilo entre o SERPRO e o cliente, o conteúdo das sessões internas aos artefatos exemplificados nesse Estudo de Caso conterá apenas **extratos** da documentação oficial para o projeto SAFE. Serão omitidas quaisquer referências a nomes de pessoas (exceto o meu), departamentos e sistemas em ambas as empresas, e demais informações consideradas de natureza confidencial dentro dos critérios de segurança e confidencialidade dos dados que regem o contrato SERPRO-Cliente, descritos na Portaria No. 782/97. O quadro de versões dos documentos apresentará, a título de exemplo e sempre que possível, informações referentes à última versão do artefato.

Histórico de Revisões

Data	Versão	Descrição	Autor	Revisor
15/10/2001	2.3	Versão acordada com o Cliente em reunião de 10/10/2001 no SERPRO-Recife	**(sigiloso)**	Fábio Rilston

Plano de Gerenciamento de Requisitos

1. Projeto

Sistema de Análises Fiscais e Estratégicas (SAFE)

Projeto desenvolvido para o cliente XXXXXXX, sob gestão da sua Coordenação de Tecnologia, conforme definição e escopo constante ao Documento de Visão do Data Warehouse para o referido projeto.

2. Objetivos do Plano

O Plano aqui contido descreve os procedimentos que irão regular a gerência dos requisitos do sistema no que se refere a mudanças de conteúdo, validações, criação e ateste de linhas base (*baselines*) de requisitos, especificação de novos requisitos, dentre outros. Esses regulamentos devem definir o comportamento e as responsabilidades dos atores (usuários e desenvolvedores) em cada fase do desenvolvimento do data warehouse.

3. Papéis e Responsabilidades

Papel	Responsabilidades	Responsável
Gestor do Projeto (Cliente)	Definir e aprovar, juntamente com o Líder do Projeto, os procedimentos para a gestão da especificação de requisitos do projeto.	**(informação sigilosa)**
Gerente Senior	Prover os recursos necessários à execução das atividades de análise e gestão dos requisitos no âmbito da empresa SERPRO.	**(informação sigilosa)**
Líder de Projeto	Definir junto com o Gestor os procedimentos para a definição e gestão dos requisitos. Alocar recursos e coordenar a aplicação dos procedimentos. Especificar, junto com a equipe de projeto, o modelo de requisitos do data warehouse. Validar o modelo com o Gestor em eventos regulares.	**(informação sigilosa)**

Figura 17. Plano de Gerenciamento de Requisitos do Projeto SAFE.

Engenheiro de Requisitos	Definir critérios para a identificação, acompanhamento e validação dos requisitos. Orientar equipe do projeto e gestor na definição e aplicação das práticas de gerência dos requisitos.	**Fábio Rilston Silva Paim (outro nome em sigilo)**
Engenheiro de Sistema (*Analista de Sistemas*)	Auxiliar o Líder de Projeto na elaboração do modelo de requisitos do data warehouse. Desenvolver a aplicação em estrita observância às práticas de gestão definidas nesse Plano.	**(informação sigilosa)**
Programador	Desenvolver as unidades de implementação necessárias à satisfação das funcionalidades específicas do data warehouse. Efetuar a alocação dos requisitos ao produto final em estrita obediência às regras de gestão definidas nesse Plano.	**(informação sigilosa)**
DBA Projetista	Fornecer dados que permitam a gestão do requisitos alocados ao repositório de dados. Zelar pela correta implementação dos requisitos ao esquema multidimensional.	**(informação sigilosa)**

4. Procedimentos para Gerência dos Requisitos

4.1. Extração de Dados

- Nos casos de extrações sob a responsabilidade da equipe da fonte provedora, os dados provenientes das bases dos sistemas que se integram com o data warehouse serão recebidos através de arquivos "texto" gerados e consolidados pelos próprios sistemas-fonte, em consonância com a especificação de caso de uso do processo de extração correspondente e com o layout padrão do data warehouse.
- Nos casos de extrações sob a responsabilidade da equipe SAFE, alterações aplicadas sobre as bases objeto da extração deverão ser formalmente comunicadas com antecedência pelas equipes responsáveis ao líder do projeto SAFE, para que este possa proceder à análise de impactos e adequação do modelo de requisitos do sistema à nova configuração de dados.

4.2. Carga de Dados

- Toda carga no repositório deverá ser procedida de uma checagem dos totais agregados referentes a cada assunto, tendo como base os totais acumulados fornecidos pelo provedor, ou contabilizados pela Equipe do Projeto SAFE. Na existência de inconsistência comprovada dos dados, o Líder de Projeto deve ser comunicado para proceder à revisão da especificação e do projeto arquitetônico.

Figura 15. Plano de Gerenciamento de Requisitos do Projeto SAFE (*continuação*).

4.3. Procedimentos Gerais

- Mudanças em requisitos do sistema somente serão aceitas se formalmente cadastradas no sistema de Controle de Demandas do cliente. O Líder de Projeto terá 5 (cinco) dias úteis para emitir parecer sobre a viabilidade da solicitação.
- Somente serão aceitas alterações de requisitos após a fase de construção do data mart em casos de extrema necessidade, com aprovação conjunta do Cliente e do SERPRO.
- Os requisitos multidimensionais especificados e extraídos das fontes provedoras devem ser alocados ao banco de dados de acordo com as seguintes prioridades: requisitos conformados primeiro e requisitos específicos do data mart em seguida; requisitos dimensionais de maior granularidade antes dos de menor granularidade. O DBA projetista deverá validar o esquema resultante da primeira operação e reportar de imediato possíveis inconsistência de integração para o Líder de Projeto.
- Apenas requisitos pertencentes a uma linha base acordada de requisitos poderão ser alocados ao repositório e ao modelo de projeto do sistema. A aprovação deverá ser dada por um comitê composto do Líder de Projeto, Gestor-Cliente e um Gerente de Configuração local.
- Os procedimentos descritos no corpo deste documento deverão ser revisados a cada inclusão de um novo data mart na estrutura corporativa do data warehouse.

5. Artefatos

Serão utilizados todos os artefatos definidos na metodologia para definição e gerência de requisitos da SUNAT-Recife. Nenhum artefato extra será necessário.

6. Identificação de Requisitos

6.1. **Tipos de Requisitos**

ATR – Ator	**CDU** – Caso de Uso	**DIM** – Dimensão
FAT – Fato	**FUN** – Funcionalidade	**NEC** – Necessidade
PFA – Passo Flux. Alternativo	**PFP** – Passo Flux. Principal	**RNF** – Req. Não-Funcional
RNG – Regra de Negócio	**GFAT** – Grupo de Fato	**SFAT** – Subgrupo de Fato

6.2. **Atributos**

6.2.1. **Situação**

Proposto	Descreve requisitos que estão ainda em discussão e não foram devidamente revisados e aceitos pelos envolvidos no projeto.
Aprovado	Utilizado para identificar requisitos cuja análise de viabilidade considerou como exeqüíveis e necessários à condução do projeto.
Implementado	Requisitos incluídos em alguma linha base do código.
Validado	Todos os envolvidos aprovaram a implementação do requisito.
Cancelado	A incorporação do requisito foi desconsiderada. É necessário registrar o motivo associado.

6.2.2. **Prioridade**

Alta	Requisitos de alta relevância para a aplicação. A não incorporação deste requisito no sistema afeta a satisfação final do cliente.

Figura 15. Plano de Gerenciamento de Requisitos do Projeto SAFE (*continuação*).

Média	Identifica requisitos que são úteis ao sistema, mas cuja ausência não compromete o funcionamento geral do aplicativo.
Baixa	Requisitos que não comprometem o sucesso do produto junto ao usuário.

6.2.3. **Risco**

Alto	O requisito é muito preocupante para a equipe de desenvolvimento, e representa um alto impacto para o sucesso do produto
Médio	A equipe de desenvolvimento sente-se confortável em implementar o requisito mesmo com restrições impostas, sendo que há ressalvas caso a implementação do requisito seja incompleta.
Baixo	A equipe de desenvolvimento está confiante de que o risco de implementação é pequeno, e não causa danos maiores ao projeto.

6.2.4. **Benefício**

Crítico	O requisito é essencial. O fracasso em sua implementação significará o não atendimento das necessidades do cliente.
Importante	O requisito é importante para o sistema. Sua não implementação afeta a satisfação do usuário e/ou o valor agregado do produto, mas não impede o funcionamento do sistema dentro dos padrões mínimos.
Útil	O requisito é útil, porém não essencial à satisfação do cliente.

6.2.5. **Aditividade do Fato**

Aditivo	O fato é aditivo ao longo de todas as dimensões.
Semi-Aditivo	O fato não é aditivo ao longo de alguma dimensão.
Não-Aditivo	O fato não é aditivo ao longo de nenhuma dimensão.

6.3. **Regra de Identificação**

Os requisitos são numerados pela ferramenta RequisitePro® segundo a seguinte regra de formação: XXXP.F, onde XXX=sigla que identifica o requisito; P=número do requisito-pai; F=número do requisito-filho.

7. Critérios de Rastreabilidade

- Toda Funcionalidade deve estar associada a pelo menos uma Necessidade;
- Toda dimensão deve estar atrelada a pelo menos um fato;
- Todo Caso de Uso deve satisfazer a uma dada funcionalidade;
- Todo Passo do fluxo alternativo deve estar associado a pelo menos um Passo do Fluxo principal.

Figura 15. Plano de Gerenciamento de Requisitos do Projeto SAFE (*continuação*).

5.5.2 Definindo o Escopo de SAFE

Ainda durante o workshop relatado na seção anterior, as partes envolvidas no projeto procuraram estabelecer um acordo sobre o escopo do *data warehouse* SAFE, o que envolveu a análise detalhada do cenário e oportunidade de negócio que motivam a construção do sistema, bem como do contexto geral dos *data marts* a serem criados para suprir as necessidades estratégicas do cliente. Essa visão integrada da aplicação foi necessária para a definição de pontos cruciais para o projeto do *data warehouse* como papéis das partes interessadas e demais atores envolvidos nos processos de extração, transformação, carga e consulta; nível de granularidade dos *data marts*; tempo de resposta médio em consultas; restrições sobre quantidade de critérios de pesquisa em relação ao volume de dados, tempo de resposta e complexidade da interface; dentre outros.

Pela própria complexidade do projeto, apenas um escopo geral foi definido no *workshop*, o qual foi evoluído ao longo de novas sessões de análise que requereram o uso intensivo das fases de especificação da metodologia (exceto *Conformidade de Requisitos*). Os engenheiros de requisito, com o auxílio do especialista do domínio, conduziram *entrevistas* informais com os clientes, procurando identificar uma visão geral dos aspectos multidimensionais do sistema SAFE no contexto da organização-cliente. O resultado desse trabalho foi registrado e pré-validado com o cliente no *Documento de Visão do Data Warehouse*, o qual está descrito na Figura 18.

As informações coletadas nas entrevistas permitiram consolidar um *Glossário do Projeto*, (Figura 19), um escopo multidimensional preliminar para a aplicação (documentada no artefato referenciado à Seção 5.5.6) e uma especificação inicial de requisitos não-funcionais (Seção 5.5.3). Uma boa prática de validação adotada nesse ponto foi a leitura dos documentos ao final dos trabalhos entre todos os participantes, como forma de garantir que as informações registradas refletissem um acordo entre as partes.

Histórico de Revisões

Data	Versão	Descrição	Autor	Revisor
21/03/2002	3.1	Atualização da estrutura conforme decisões da reunião com o Gestor-Cliente em 18/03/2002.	**(sigiloso)**	Fabio Rilston **(revisor 2)**

Documento de Visão do Data Warehouse

1. Posicionamento do Projeto SAFE

1.1. **Cenário**

Apesar de inúmeros sistemas já terem sido desenvolvidos pelo SERPRO para o cliente, até o momento não existe uma ferramenta que permita a consulta integrada das informações relevantes de cada uma destas aplicações. Neste contexto, os usuários do alto escalão do cliente dependem de dados informativos na forma de relatórios e interfaces gráficas individualizadas para suporte ao seu processo de tomada de decisão. Embora abrangente em muitos casos, a visão dos dados no âmbito de assuntos independentes dificulta uma visão única e integrada por parte do cliente. Ademais, as soluções hoje providas não suportam a execução dinâmica de consultas complexas sobre suas bases, devido a restrições de implementação e de volume de processamento, em muitos casos alto.

1.2. **Oportunidade de Negócio**

O SERPRO pode mais uma vez atender ao seu cliente, dentro da sua vocação como empresa pública, desenvolvendo um sistema que venha a suprir a necessidade da instituição de uma base integrada para suporte à tomada de decisões gerenciais. O sistema SAFE (*Sistema de Análises Fiscais e Estratégicas*) irá possibilitar a execução de consultas analíticas complexas sobre os diversos assuntos que compõem a fiscalização federal, representando uma poderosa ferramenta para o acompanhamento do trabalho realizado por seus agentes em todo o território nacional, o que há muito é um desejo do cliente. Adicionalmente, o SAFE incorporará informações relativas a assuntos externos à fiscalização – como, por exemplo, Recursos Humanos - ampliando a visão do usuário do alto escalão do cliente a respeito de seu negócio.

1.3. **Descrição**

O sistema SAFE possibilita a usuários de alto escalão do cliente realizar consultas analíticas complexas sobre bases de dados de inúmeros assuntos que permeiam a atividade fiscalizadora deste orgão. Uma interface Web provê a acessibilidade a todos esses usuários em qualquer local do país. Através desta interface, usuários podem selecionar qual assunto desejam analisar, e um conjunto de critérios sobre os quais as consultas analíticas serão montadas. Um repositório de dados suporta a execução otimizada de consultas, armazenando dados de diversos sistemas do cliente obtidos após um processo de extração, agregação e carga destas informações.

1.4. **Objetivos**

- Montar uma base integrada dos dados operacionais do cliente XXXXXXX.
- Fornecer uma visão estratégica única de todos os assuntos do negócio Fiscalização de Tributos.
- Permitir consultas *ad hoc* sobre a base integrada a todos os funcionários do alto escalão do cliente.
- Garantir a integridade da informação extraída e agregada no repositório corporativo.
- Oferecer escalabilidade para a evolução contínua da visão estratégica integrada.

Figura 18. Documento de Visão do *Data Warehouse* SAFE.

1.5. **Cobertura**

O sistema SAFE é de acesso privativo dos auditores fiscais, delegados, inspetores e superintendentes que atuam no quadro da organização-cliente, em todas as suas projeções, bem como dos funcionários que integram o alto escalão desta instituição, conforme determinações constantes da portaria No. 18, de 16/04/2001. No âmbito do SERPRO, apenas o Líder de Projeto possui autorização para acesso ao repositório de dados.

1.6. **Solução Tecnológica**

SAFE utiliza uma arquitetura relacional (ROLAP) e implementa um esquema estrela ("constelação)". O banco de dados Oracle na versão 8.1.7.2 está localizado numa máquina RISC IBM RS6000 rodando sistema operacional AIX 4.3.3. A interface OLAP escolhida é a ferramenta MicroStrategy na sua versão 7.1.

2. **Papéis de Partes Interessadas (*Stakeholders*) e Atores**

2.1. **Gestor**

Descrição	Responsável no cliente pela coordenação do projeto.
Papel no desenvolvimento da aplicação	Definir as necessidades a serem atendidas pelo sistema. Estabelecer as funcionalidades requeridas, restrições operacionais, e requisitos de interface. Identificar juntamente com o Engenheiro de Sistema os requisitos de ordem funcional, não-funcional e de domínio do data warehouse. Especificar e solicitar atualizações na visão dos dados relativos aos diversos assuntos integrados.
Insumos ao Sistema	Necessidades do usuário, Solicitação de Alteração de Requisitos, Requisitos Funcionais, Requisitos de Qualidade, Restrições Operacionais e de Negócio, Modelo preliminar de Interface.

2.2. **Líder de Projeto**

Descrição	Papel responsável pela direção, controle e administração do projeto.
Papel no desenvolvimento da aplicação	Gerenciar as etapas de desenvolvimento, e interagir com a equipe para a perfeita acomodação dos requisitos do usuário. Representar o projeto perante o cliente. Interagir com as equipes provedoras de dado para o perfeito funcionamento do processo de carga.
Insumos ao Sistema	Requisitos de Sistema, Recursos alocados, Cronograma de desenvolvimento, Especificação de Requisitos.

3. Relação de Data Marts

Relação das Visões que compõem o Data Warehouse, por ordem de implantação.

3.1. **Visão Ação Fiscal PJ**

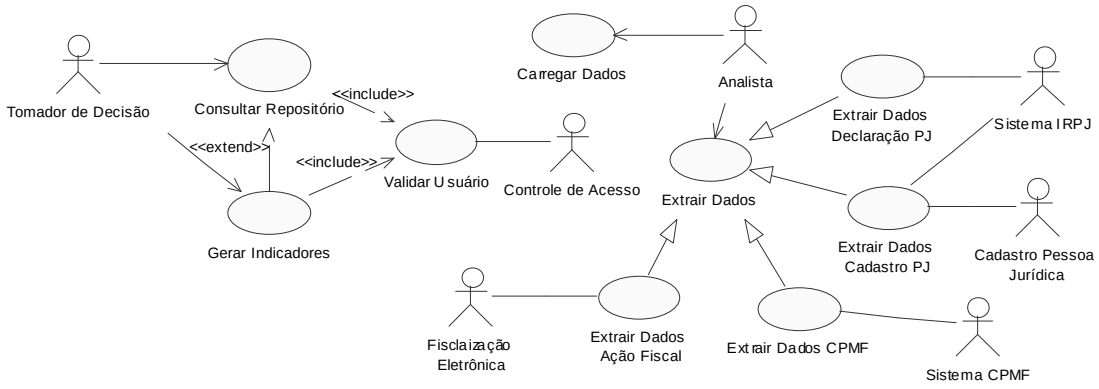
Oferecer uma visão integrada das ações de fiscalização sobre o contribuinte pessoa jurídica.

3.2. **Visão Ação Fiscal PF**

Oferecer uma visão integrada das ações de fiscalização sobre o contribuinte pessoa física.

Figura 16. Documento de Visão do Data Warehouse SAFE. (*continuação*)

4. Fronteira Funcional



5. Critérios de Qualidade

- Todos os Data Marts devem estar rigorosamente de acordo com o Padrão de Interfaces do cliente para o ambiente Data Warehouse. As interfaces serão validadas por especialistas do cliente em homologações com a participação do SERPRO.
- Antes da carga definitiva no repositório, todas as informações de fato e dimensões deverão ser checadas conforme procedimentos descritos no Plano de Gerenciamento de Requisitos, para assegurar a integridade de dados e de domínios em relação ao modelo corporativo. Os resultados devem ser avaliados e liberados pelo Líder de Projeto.
- A informação armazenada numa Visão do data warehouse não poderá possuir uma defasagem maior que 2 meses em relação ao dado no sistema de origem.

6. Restrições de Projeto

- O orçamento do projeto não comporta para o ano de 2001 a liberação de mais que dois data marts, ao custo médio de R$ **(info. sigilosa)**.
- Devido ao volume de demandas em andamento, as seguintes fontes provedoras não poderão fornecer dados ao data warehouse até o mês de Julho de 2001: **(info. sigilosa)**.
- Extrações e transformações nas bases de Cadastro só poderão ocorrer numa data específica no mês. Exceções serão processadas fora do horário comercial, via apuração especial.
- Cargas iniciais não poderão envolver mais que 30 processos executando no UNIX.

7. Marcos do Projeto

Evento	Data
Validação 1º. Protótipo Visão Ação Fiscal PJ	11/03/2001
Validação 2º. Protótipo Visão Ação Fiscal PJ	29/03/2001
Entrega e Validação de Linha Base de Requisitos	16/04/2001
Carga ambiente de produção	22/05/2001
.....	

Figura 16. Documento de Visão do Data Warehouse SAFE. (*continuação*)

Histórico de Revisões

Data	Versão	Descrição	Autor	Revisor
15/06/2002	3.0	Atualização da terminologia do projeto para a visão Arrecadação.	Fábio Rilston	**(sigiloso)**

Glossário do Projeto

1. Introdução

(omitida por razões de espaço, texto original descrito no template para o artefato – Apêndice 1)

2. Organização

Os conceitos e termos aqui definidos encontram-se organizados em Grupos, os quais determinam aspectos específicos do escopo do sistema, para facilitar o entendimento pelo leitor. Os grupos criados agrupam informações de **caráter geral**, referentes ao sistema como um todo; **por assunto**, que explicam os termos específicos de cada assunto tratado em SAFE; ou ainda relevantes para o entendimento dos **aspectos multidimensionais** do sistema, dentro do seu enfoque como ferramenta de data warehouse. Este documento será atualizado frequentemente, ao longo do processo, quando a definição de novos termos tornar-se necessária.

3. Definições

3.1. **Termos Gerais**

Tomador de Decisão

Pessoa no âmbito do cliente responsável pela análise e tomada de ações estratégicas.

Tributo

Toda prestação pecuniária compulsória, em moeda ou cujo valor nela se possa exprimir, que não constitua sanção de ato ilícito, instituída em lei e cobrada mediante atividade administrativa plenamente vinculada [CTN].

CNAE

Código Nacional de Atividade Econômica. Discrimina a natureza das atividades econômicas pelas quais os tributos são agrupados.

3.2. **Visão Ação Fiscal PJ**

COFINS

Contribuição para Financiamento da Seguridade Social. É um tributo cobrado pela União sobre o faturamento bruto das pessoas jurídicas, destinado a atender programas sociais do Governo Federal. Sua alíquota, que era de 2%, foi aumentada para 3% em fevereiro de 1999.

Unidade Cadastradora

É o local de análise dos pedidos encaminhados pela Internet e atendimento dos pedidos de baixa perante o CNPJ, de iniciativa do contribuinte.

Figura 19. Extrato do Glossário de Termos do Projeto SAFE.

3.3. **Data Warehouse**

Dimensão

Conjunto de atributos que definem uma perspectiva única em cima de um dado armazenado num esquema multidimensional.

Fato

Dado numérico ou factual que representa uma atividade específica de negócio.

OLAP (*On-Line Analytical Processing*)

Termo introduzido em [Codd93] para definir a categoria de processamento baseada em consultas analíticas sobre grandes bases de dados históricas para o suporte à tomada de decisão.

4. Referências

[CTN] Código Tributário Nacional (CTN): http://www.receita.fazenda.gov.br/

[Codd93] Codd, E. F., Codd, S. B., Salley, C. T. "Providing OLAP (Online Analytical Processing) to User Analyst: An IT Mandate". Arbor Software White paper, http://www.arborsoft.com/OLAP.html, 1993.

Figura 17. Extrato do Glossário de Termos do Projeto SAFE.

5.5.3 Analisando Requisitos Não-Funcionais

Durante entrevistas com os clientes, ainda na fase de elicitação, os requisitos de qualidade do *data warehouse* foram discutidos com os clientes. O foco desse trabalho foi o estabelecimento de padrões de qualidade para controle de aspectos vitais ao funcionamento da aplicação como freqüência de atualizações, volume máximo de dados, quantidade de acessos simultâneos, flexibilidade da interface, restrições legais, e outros tantos. O *framework* DW-ENF (Capítulo 3 e Apêndice 2) serviu como uma *lista de verificação* (*checklist*) para os principais requisitos não-funcionais em sistemas *data warehouse*, dando uma segurança maior no tratamento desses aspectos. Requisitos não-funcionais referentes a visões específicas do *data warehouse* (*data marts*) não foram considerados nesse momento. A **Especificação de Requisitos Não-Funcionais** para o projeto SAFE, descrita pela Figura 20, apresenta o resultado desse trabalho.

Histórico de Revisões

Data	Versão	Descrição	Autor	Revisor
03/04/2001	2.2	Inclusão de requisitos de Acessibilidade.	**(sigiloso)**	Fábio Rilston

Especificação de Requisitos Não-Funcionais

1. Usabilidade

1.1. A aplicação deve refletir, sempre que possível, a estrutura de menus descrita no padrão de interfaces do cliente.

1.2. A aplicação deverá prover facilidades como *help on-line*, menus dinâmicos, *tool tips*.

1.3. Não pode haver relatório pré-definido. O usuário deverá ser capaz de criar a sua própria consulta ou relatório, através da escolha de critérios de montagem.

2. Acessibilidade

2.1. O sistema deve estar disponível para todos os seus usuários 24 (vinte e quatro) horas por dia, durante os 7 (sete) dias da semana.

2.2. É obrigatória a escolha dos critérios Nível Operacional e Período, nas consultas ao repositório.

2.3. O Sistema deverá permitir ativar/desativar a coluna de Totais, em qualquer consulta.

2.4. A cada acesso do usuário para consulta, somente 5 dimensões são possíveis para seleção ao mesmo tempo.

2.5. O sistema deve permitir que os usuários naveguem entre fatos de assuntos diferentes.

3. Performance

3.1. O tempo médio de resposta a consultas simples ao repositório deverá ser de, no máximo, 1 minuto. Para consultas complexas (envolvendo mais que 30 milhões de registros ou 3 tabelas-fato), este tempo poderá ficar na faixa de 1 a 3 minutos.

3.2. O sistema deverá suportar até 500 usuários ativos num dado momento.

3.3. O sistema deverá suportar até 200 acessos de uma única vez.

4. Precisão dos Dados

4.1. O sistema deverá proceder à atualização periódica dos dados das fontes provedoras, dentro dos prazos estabelecidos entre o clientee as equipes responsáveis no SERPRO.

4.2. Os dados devem ser fornecidos pelas fontes externas de acordo com o cronograma mensal de trabalho definido e acordado entre o líder do projeto SAFE e o responsável pela fonte provedora. A impossibilidade de entrega dos dados deverá ser comunicada com antecedência mínima de 7 dias úteis para replanejamento da agenda e especificação de procedimentos extras para carga interna.

4.3. Durante os processos de carga e extração, os dados deverão refletir a real posição na fonte provedora, mesmo que isto signifique uma inconsistência do dado.

5. Segurança

5.1. O sistema está definido para operar conjuntamente com o Sistema de Controle de Acesso, e garantir assim o acesso aos dados apenas às pessoas devidamente autorizadas.

Figura 20. Especificação de Requisitos Não-Funcionais do projeto SAFE.

5.5.4 Definindo a Arquitetura Multidimensional

Definido o escopo geral da aplicação, as preocupações voltaram-se para a arquitetura a ser adotada. Técnicos e clientes compartilhavam as idéias de KIMBALL (1998) de que um esquema multidimensional "estrela" com tabelas desnormalizadas proveria a eficiência necessária para as consultas à base corporativa. Contudo, todos concordavam que alguma forma de investigação era requerida para comprovar esse e outros aspectos relevantes para a definição da arquitetura do *data warehouse*, como indexação e interligação de tabelas.

Considerando-se o escopo estabelecido e as premissas de qualidade básicas definidas na *Especificação de Requisitos Não-Funcionais* para o projeto, o *framework* DW-ENF foi usado para investigar uma arquitetura ideal para o sistema (essa etapa teve uma forte participação do DBA Projetista). Vamos reportar essa análise do ponto de vista dos requisitos de qualidade que foram fundamentais para a definição da arquitetura multidimensional[12], a saber: (***r₁***) o tempo de resposta das consultas não deve exceder 1 minuto; (***r₂***) o sistema deve fornecer sempre a informação mais atualizada possível; (***r₃***) o sistema deve permitir que os usuários naveguem entre fatos de assuntos diferentes.

Pode-se perceber a partir do *framework* DW-ENF que esses requisitos de qualidade estão diretamente relacionados com a *performance* e o *potencial multidimensional* esperado do sistema SAFE. Para determinar a arquitetura ótima, partiu-se então para investigar os seguintes aspectos: o *tempo de resposta* das consultas, a *pontualidade* das cargas de dados e os *métodos de acesso* a serem implementados. Utilizando os catálogos de tipo e métodos do *framework*, construímos o Gráfico de Interdependência de *Softgoals* (SIG) ilustrado na Figura 21.

[12] Por restrições de espaço, atemo-nos a esse grupo de requisitos para exemplificar a contribuição do *framework* para a identificação de uma arquitetura multidimensional ideal, muito embora a investigação prática tenha envolvido outras variantes como *características relacionais do banco de dados Oracle, otimização de espaço em disco, esquema de carga de dados, freqüência de acesso ao dado*, dentre outras.

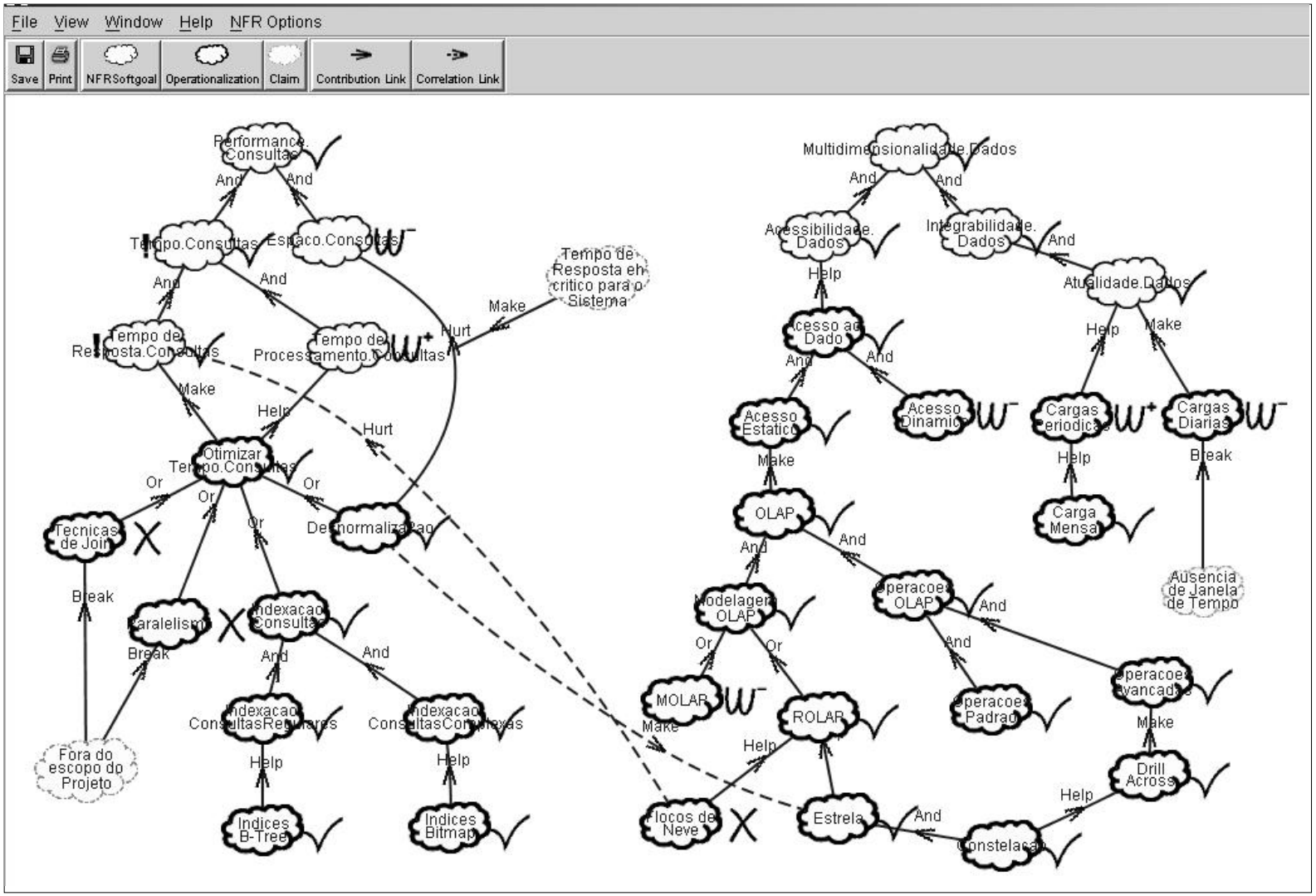


Figura 21. Gráfico de Interdependência de *Softgoals* para investigação da arquitetura no sistema SAFE.

Para geração do SIG, foi utilizada a ferramenta OME3 (OME3, 2000), cuja notação difere um pouco da notação original de CHUNG *et al.* (2000) descrita no Capítulo 2. Os símbolos especiais "W–" e "W+" indicam respectivamente que o softgoal foi "fracamente rejeitado" ou "fracamente aceito". OME3 também utiliza as palavras HELP, MAKE, HURT e BREAK em substituição aos sinais "+","++", "–" e "– –" respectivamente. As demais regras da notação original permanecem válidas.

A partir do requisito r_1, assinalamos o requisito não-funcional "*performance no tempo*" como crítico para o sistema, indicado pelo sinal "**!**" ao lado do *softgoal* **Tempo.Consultas**. Um dado importante nessa análise foi a aquisição de um servidor de dados robusto, com capacidade para armazenamento e acesso rápido a *terabytes* de dados, o que tornou o requisito adjacente "*performance em disco*" uma preocupação menor no desenvolvimento do sistema.

Ainda analisando a performance no tempo, *tempo de resposta* prevalece sobre *tempo de processamento*, pois a freqüência de carga supera o tempo para processamento dos dados em significância, o que pode ser inferido a partir do requisito r_2 e das facilidades do servidor de dados mencionado anteriormente. A criticidade do tempo de resposta é evidenciada pelo sinal "**!**" ao lado do *softgoal* que o representa.

A investigação do aspecto Tempo concentrou-se então no fator "*tempo de resposta*". O SIG da Figura 21 demonstra que, para otimizar o tempo de resposta, o projetista do banco pode optar por fazer uso de quaisquer dos seguintes métodos: técnicas de Join e Indexação (O'NEIL e GRAEFE, 1995), paralelismo de processadores e/ou desnormalização. Porém, a ferramenta *Microstrategy®*, escolhida para implementar o ambiente OLAP, controla e otimiza automaticamente *join* (junções) de dados. Além disso, há a expectativa de que o *warehouse* do projeto SAFE armazene algumas centenas de milhões de registros, o que elimina a necessidade de investimentos em processamento paralelo, mais indicado para quantidades superiores a *gigabytes*. Com base na argumentação anterior, o projetista pode rejeitar, respectivamente, os softgoals **Técnicas de Join** e **Paralelismo**.

A investigação continuou, então, com uma análise das técnicas de indexação. A Interface Web de SAFE possibilita aos usuários finais executarem tanto consultas complexas quanto outras de médio porte. Consultas complexas freqüentemente efetuam *joins* entre múltiplas tabelas, o que é enfatizado pela necessidade de executar operações *drill across* (Capítulo 4), implícita no requisito r_3. Consultas de médio porte, ao contrário, retornam uma quantidade pequena de dados a partir de poucas tabelas-fato. Os dois tipos de consulta requerem abordagens de indexação diferenciadas, indicadas pela decomposição do *softgoal* **Indexação.Consultas** em **Indexação.ConsultasRegulares** e **Indexação.Consultas Complexas**. Consultas regulares são melhor suportadas por índices do tipo *B-Tree* (O'NEIL e QUASS, 1997), enquanto consultas complexas requerem índices mais sofisticados como os do tipo *Bitmap* (CHAUDHURI e DAYAL, 1997). Ambas as técnicas foram escolhidas, então, por satisfazerem os requisitos de performance das consultas.

Por outro lado, o requisito r_3 sugere fortemente a adoção de uma derivação do esquema "estrela" conhecida como *constelação* (Capítulo 3), onde esquemas "estrela" individuais estão hierarquicamente interligados por dimensões comuns. As interligações entre tabelas-fato provêem a habilidade requerida pelo cliente de navegar entre assuntos diferentes, o que é indicado na Figura 21 pela interdependência fortemente positiva entre os *softgoals* (métodos) **Constelação** e ***Drill Across***. *Desnormalização*, por sua vez, torna-se um *softgoal* elegível, na medida em que é a base para a implementação de esquemas "estrela". Apesar de desnormalização ferir o aspecto "performance em disco", este não é um ponto crítico do sistema, conforme visto anteriormente. Ademais, a outra escolha lógica, "Flocos de Neve", não pode ser aceita pois fere o *softgoal* crítico "*tempo de resposta*".

Numa última análise, o projetista é confrontado com a necessidade de processamento pontual das cargas de dados. O requisito r_2 sugere a adoção de uma abordagem diária para a carga. Essa opção, contudo, não pode ser satisfeita devido à falta de uma "janela de tempo" no ambiente de produção do SERPRO larga o suficiente para realizar a extração de dados de todas as fontes. De fato, a menor periodicidade com que os dados podem ser carregados, dentro dos limites do ambiente, é mensal. Dessa forma, uma periodicidade mensal foi aceita

e a restrição acordada junto ao cliente. O resultado da investigação validou as concepções iniciais de técnicos e partes interessadas sobre o esquema em "estrela" e proporcionou uma visão mais bem definida dos demais aspectos (indexação e integração de tabelas). A conclusão então alcançada foi a adoção de uma arquitetura baseada em um esquema "*constelação*" para facilitar operações *drill-across* e a implementação de tabelas desnormalizadas, índices *Bitmap* e *B-Tree* para satisfazer os requisitos de performance da aplicação. Essas conclusões foram interadas à **Especificação dos Requisitos Multidimensionais**, na seção 1.1 "Escopo e Diretrizes Gerais".

5.5.5 Definindo o Escopo da Visão "Ação Fiscal PJ"

Tendo estabelecido uma visão global do *data warehouse* SAFE, o esforço se concentrou na especificação do primeiro *data mart* a ser entregue, correspondente a uma visão das ações fiscais conduzidas sobre contribuintes pessoa jurídica. Um *workshop* reunindo todas as partes interessadas foi organizado com esse intuito na semana de 5 a 9 de março de 2001, e teve como primeira meta estabelecer o escopo geral do *data mart*, o que incluiu a identificação do seu propósito, no contexto do cliente e do projeto SAFE; o seu público alvo; e os papéis, dentre aqueles já definidos para o *data warehouse*, que estarão envolvidos com o desenvolvimento da Visão "Ação Fiscal PJ", com suas respectivas responsabilidades.

Um exercício fundamental nessa etapa foi identificar as *necessidades*, i.e., as metas de negócio do cliente que motivam a construção do *data mart*; e as *funcionalidades* a serem implementadas para satisfazer essas metas. Necessidades foram mapeadas de acordo com seu *benefício* para a visão-alvo, enquanto as funcionalidades foram categorizadas segundo sua *importância* para o alcance da necessidade. O resultado foi anotado no **Documento de Visão do Data Mart** (Figura 22), e transportado já no padrão de requisitos definido no *Plano de Gerenciamento de Requisitos* para a ferramenta RequisitePro®, juntamente com a relação de requisitos já elicitada para o *data warehouse* global.

Histórico de Revisões

Data	Versão	Descrição	Autor	Revisor
26/03/2001	1.4	Versão inicial sujeita a revisão.	**(sigiloso)**	Fábio Rilston

Documento de Visão do Data Mart

1. Visão Geral

1.1. **Identificação**

Visão Ação Fiscal PJ

1.2. **Propósito**

Oferecer uma visão integrada dos dados referentes às ações de fiscalização sobre o contribuinte pessoa jurídica.

1.3. **Audiência**

O acesso à Visão Ação Fiscal PJ é privativo dos auditores fiscais, para efetuar cruzamentos entre fatos da fiscalização federal; delegados e inspetores para elaboração do programa de fiscalização regional; e demais cargos do alto escalão do cliente para tomada de decisão sobre as atividades de fiscalização.

2. Papéis Envolvidos

Papel	Responsável
Gestor Cliente	**(nome protegido por sigilo)**
Gerente Sênior	**(nome protegido por sigilo)**
Líder de Projeto	**(nome protegido por sigilo)**
Engenheiro de Requisitos	Fábio Rilston Silva Paim

3. Necessidades do Usuário

Necessidade #1	**Benefício**
Selecionar contribuintes candidatos à fiscalização com base nos valores declarados e recolhimento de CPMF.	Crítico
Funcionalidade(s) de Suporte:	**F1, F2, F3, F4, F5, F11, F15*.**

Necessidade #2	**Benefício**
Avaliar o desempenho dos fiscais no exercício atual e anteriores.	Crítico
Funcionalidade(s) de Suporte:	**F1, F2, F3, F4, F7, F18.**

Necessidade #3	**Benefício**
Visualizar os dados fiscais de um contribuinte individual.	Importante
Funcionalidade(s) de Suporte:	**F1, F5, F8, F15, F27.**

** algumas funções poderão não estar descritas posteriormente por razões de espaço e sigilo.*

Figura 22. Documento de Visão do *Data Mart* "Ação Fiscal PJ".

4. Funcionalidades

ID	Descrição	Prioridade
F1	O sistema deverá permitir detalhar/agrupar as informações apresentados por critérios pré-estabelecidos.	Alta
F2	O sistema deverá permitir alternar linhas por colunas, correspondentes às dimensões na matriz de dados.	Alta
F3	O sistema deverá permitir mover (*pivotar*) linhas correspondentes às dimensões na matriz de dados para colunas (ou vice-versa).	Alta
F4	O sistema deverá permitir ativar/desativar uma dimensão na tela de consulta.	Média
F5	O sistema deverá extrair, adequar e carregar dados das declarações DIPJ.	Alta
F6	O sistema deverá extrair, adequar e carregar dados da arrecadação relativos à pessoa jurídica.	Alta
F7	O sistema deverá extrair, adequar e carregar dados das ações fiscais de 1991 a 2002.	Alta
F8	O sistema deverá exibir indicadores financeiros para auxílio à análise de dados de compras e notas fiscais.	Baixa

5. Marcos da Visão "Ação Fiscal PJ"

Evento	Data
Conclusão 1º. Protótipo Visão Ação Fiscal PJ	08/04/2002
Validação 1º. Protótipo Visão Ação Fiscal PJ	11/04/2002
Entrega e Validação de Linha Base de Requisitos Preliminar	19/04/2002
Validação 2º. Protótipo Visão Ação Fiscal PJ	30/04/2002
Entrega e Validação de Linha Base de Requisitos	16/05/2002
Homologação Versão Final Visão Ação Fiscal PJ	31/07/2002
Implantação em produção	05/08/2002

Figura 20. Documento de Visão do Data Mart "Ação Fiscal PJ".

5.5.6 Modelando os Requisitos do Data Mart

De posse das necessidades e funcionalidades, deu-se início, ainda no *workshop* citado na seção anterior, ao detalhamento do modelo de requisitos da Visão "Ação Fiscal PJ". Esse trabalho envolveu três frentes distintas:

(1) **Definição dos Requisitos Multidimensionais**: clientes e técnicos puderam conceituar, em termos de *fatos* e *dimensões*, um esquema multidimensional adequado ao escopo de necessidades e funcionalidades, a partir das descrições desses requisitos. Fatos, por sua vez, foram definidos em termos de suas *métricas*, *aditividade* e conformidade com o

esquema global. Fatos comuns como *movimentação financeira*, *ação fiscal* e *cobrança* foram identificados e registrados na seção correspondente do artefato *Especificação de Requisitos Multidimensionais*, com sua respectiva unidade de representação padrão. Dimensões foram definidas em termos de *atributos*, com suas respectivas cardinalidades, as quais se demonstraram úteis para avaliar a complexidade das consultas resultantes e melhor dimensionar a configuração de banco. Uma análise do panorama futuro do *data warehouse* revelou dimensões comuns, a saber: *atividade econômica*, *temporal*, *contribuinte*, *nível organizacional*, dentre outras. Ao final, atividades de *Análise & Negociação* procuraram: (*a*) investigar os níveis de agregação entre fatos e dimensões, e sua adequação às futuras consultas analíticas; (*b*) garantir que fatos comuns estivessem definidos dentro do mesmo contexto dimensional. O artefato **Especificação de Requisitos Multidimensionais** já existente foi revisado e acrescido dos requisitos multidimensionais identificados ao final desta etapa (Figura 23).

(2) **Detalhamento dos Requisitos Funcionais**: as funcionalidades divisadas serviram de base para a especificação de um modelo de casos de uso para a Visão "Ação Fiscal PJ". O modelo organizou comportamentos funcionais similares em termos do modelo padrão de casos de uso proposto no Capítulo 4 e serviu de embrião para o modelo de casos de uso global do projeto SAFE, representado pelo diagrama da Seção 4 do *Documento de Visão do Data Warehouse*. A especificação dos casos de uso foi associada à confecção de protótipos do sistema na ferramenta *MicroStrategy®* (MICROSTRATEGY, 2003), para dar maior visibilidade ao cliente durante as reuniões de validação da especificação, conduzidas em marcos do projeto (vide *Visão do Data Warehouse*). A leitura dos passos do fluxo de interação que compõem cada caso de uso em reuniões com o cliente demonstrou-se uma técnica bastante eficiente para a descoberta de requisitos ausentes, ambíguos ou inconsistentes, aumentando a percepção do cliente sobre a aplicação e contribuindo para gerar uma especificação mais ajustada do sistema. Como exemplo de casos de uso no sistema SAFE, descrevemos aqui especificações referentes à "Extração de Dados" (juntamente com uma de suas especializações) (Figura 24 e Figura 25) e à "Consulta ao Repositório" (Figura 26).

Uma outra preocupação nessa etapa foi com a identificação das regras de negócio que regulam o comportamento do sistema descrito nos casos de uso. As regras foram organizadas em *genéricas* (i.e., aquelas que regulam o funcionamento do *data warehouse* como um todo) e *específicas* (relacionadas com um dado *data mart*) e devidamente referenciadas dentro dos casos de uso (*vide especificação do caso de uso "Extrair Dados Cadastro Pessoa Jurídica"*). Um extrato dessa estrutura está ilustrado na Figura 27.

(3) **Análise dos Requisitos Não-Funcionais**. Os requisitos de qualidade para o *Data Mart* foram analisados, mas não foi encontrada nenhuma restrição de qualidade específica. Foram mantidos então os mesmos requisitos não-funcionais definidos para o projeto como um todo.

Ao final da etapa, a lista de verificação para requisitos de sistemas *data warehouse* proposta no Capítulo 4 (Seção 4.5.2, *Análise & Negociação*) foi aplicada para verificar a completude do modelo de requisitos gerado. Foi identificada a necessidade de um maior detalhamento dos casos de uso de extração de dados quanto a questões sobre "domínio de atributos", essenciais para que as equipes dos sistemas provedores pudessem especificar os programas de extração. As especificações foram revistas e ajustadas.

Histórico de Revisões

Data	Versão	Descrição	Autor	Revisor
25/04/2002	3.5	Revisão da granularidade dos data marts com a integração prevista com a estrela de Arrecadação.	Fábio Rilston **(autor 2)**	**(sigiloso)**

Especificação de Requisitos Multidimensionais

1. Ambiente Multidimensional

1.1. **Escopo e Diretrizes Gerais**

O ambiente data warehouse do projeto SAFE está construído sobre uma arquitetura ROLAP (OLAP-Relacional), implementando um esquema estrela especial denominado "constelação" para permitir a navegação (*drill-across*) entre diferentes visões de dados. O esquema multidimensional é implementado no banco de dados Oracle 8.1 por meio de tabelas desnormalizadas e particionadas conforme as restrições de volume. Como esquema de indexação, utilizam-se índices do tipo *Bitmap* e *B-Tree* para satisfazer requisitos de performance. A ferramenta OLAP MicroStrategy 7.1 é a base para o desenvolvimento das interfaces de cada Visão do data warehouse, em conformidade com o padrão visual estabelecido pelo cliente para o ambiente, o qual encontra-se descrito no site de publicação do projeto.

1.2. **Granularidade**

A Visão "Ação Fiscal PJ" terá como nível de granularidade geral as empresas/estabelecimentos. Na dimensão Bens e Direitos, o menor nível de granularidade será "Classe do Bem"; Para as demais dimensões, será considerado, para efeito de consulta, o menor nível de granularidade definido no esquema multidimensional;

1.3. **Diagrama Multidimensional**

O esquema multidimensional global do data warehouse encontra-se particionado em várias visões ou "estrelas" que podem ser consultadas no site da Administração de Dados do SERPRO (ADSite) localizado no endereço http://10.14.52.67/adsite/ . Apenas funcionários do SERPRO e do cliente devidamente cadastrados têm direito de acesso à página supracitada. O site e todo o seu conteúdo são protegidos pelos critérios de segurança e confidencialidade dos dados que regem o contrato SERPRO-Cliente, descritos na Portaria No. 782/97.

1.4. **Restrições Multidimensionais**

1.4.1. A ferramenta MicroStrategy não trabalha com intervalos de valores para pesquisa nos dados, portanto consultas dessa natureza não são permitidas.

1.4.2. A ferramenta MicroStrategy não permite consultas que envolvam a associação entre um fato e duplos relacionamentos (auto-relacionamento) entre dimensões. Visões separadas devem ser criadas na estrutura do repositório nestes casos.

1.4.3. Tabelas-fato ou dimensão devem ser particionadas caso o seu conteúdo exceda 5 milhões de registros. A regra de particionamento obedecerá ao domínio de um campo escolhido (*ex*. Unidade organizacional). Quando esse critério não puder ser adotado, um critério auxiliar como "data de carga" pode ser usado, sob responsabilidade do Administrador do Banco.

Figura 23. Especificação dos Requisitos Multidimensionais no Projeto SAFE.

2. Requisitos Multidimensionais Corporativos

2.1. **Dimensões Conformadas**

2.1.1. **Temporal**

Atributos	Descrição	Cardinalidade
Dia	Data do dia da ocorrência do fato, no formato YYYYMMDD.	2920
Decêndio	Número identificador de decêndios	288
Mês	Número identificador do mês.	96
Trimestre	Número de identificação do trimestre.	32
Ano	Número identificador do ano.	8

2.1.2. **Atividade Econômica**

Atributos	Descrição	Cardinalidade
Atividade	Número da atividade econômica.	568
Subsetor	Número da divisão do setor da atividade econômica.	225
Setor	Número sequencial do setor econômica.	66
GrupoSetor	Número sequencial do grupo-setor econômico.	24

2.1.3. **Unidade Organizacional**

Atributos	Descrição	Cardinalidade
Município	Número sequencial do município.	5574
Estado	Número da unidade da federação.	35
Região Política	Número da região política da RF.	13
País	Número sequencial do país.	269

2.2. **Fatos Conformados**

Fatos	Descrição	Métricas do Fato	Formato	Aditiv.	Tipo	Fórmula
Movimentação Financeira	Movimento pecuniário anual do contribuinte.	Valor Movimento Anual	M17,2	**S**	Derivado	(**info. sigilosa**)
		Valor CPMF Recolhida	M17,2	**A**	Simples	
		Qtd. Contribuintes c/ Movimentação	N1	**A**	Simples	
Ação Fiscal	Situação das atividades de fiscalização por superintendência.	Qtd. Ações Fiscais	N2	**A**	Simples	
		Qtd. Ações Fiscais p/ Tributo	N2	**A**	Simples	
		Qtd. Ações Fiscais p/ Operação	N2	**A**	Simples	
Cobrança	Resultados da cobrança do tributo por exercício.	Qtd. Contribuinte Autuado	N8	**A**	Simples	
		Qtd. Créditos Tributários	N8	**A**	Simples	
		Valor Consolidado Auto Inf.	M17,2	**A**	Derivado	(**info. sigilosa**)
		....				

LEGENDA: FORMATO..... M=Moeda; A=Alfanumérico; N=Numérico; D=Data.
ADITIVIDADE.... A=Aditivo; S=Semi-Aditivo;N=Não-Aditivo.

3. Requisitos Multidimensionais dos Data Mart

3.1. **Visão Ação Fiscal PJ**

3.1.1 **Dimensões**

Figura 23. Especificação dos Requisitos Multidimensionais no Projeto SAFE (*cont.*).

3.1.1.1 **Pessoa Jurídica**

Atributos	Descrição	Cardinalidade
CNPJ	Identificação do contribuinte no Cadastro de Pessoa Jurídica	12.905.000
CNAE	Número da atividade econômica	1741
Município	Número do munícipio de domicílio fiscal do contribuinte	5574
Estado	Número da unidade da federação	35

3.1.1.2 **Estabelecimento Comercial**

Atributos	Descrição	Cardinalidade
No. Registro	Número de registro do estabelecimento na Junta Comercial	13.998.000
Vigência	Indicador da Vigência: "Extinto" ou "Em Vigor"	13.998.000
Tipo Estab.	Tipo do Estabelecimento	5

3.1.2 **Fatos**

Fatos	Descrição	Métricas do Fato	Formato	Aditiv.	Tipo	Fórmula
Arrecadação	Valores arrecadados da pessoa jurídica	Valor Arrecadação Bruta	M17,2	**A**	Derivado	(**info. sigilosa**)
		Qt. Parcelas DARF	N9	**S**	Simples	
Cadastro Pessoas Jurídicas	Quantitativo de CNPJs por situação	Qtd. CNPJ Ativo	N9	**A**	Simples	
		Qtd. CNPJ Inativo	N9	**A**	Simples	
Malha PJ	Resultados de apuração da Malha PJ.	Qtd. Incidência em Malha	N9	**A**	Simples	
		Vlr. Aumento Restituição	N17,2	**S**	Simples	
		Vlr. Multas Malha	N17,2	**A**	Simples	
Aduaneira	Importação e Exportação de Mercadorias	Tempo Médio Desembaraço	Hora	**S**	Derivado	TempoTotal/ Qtde. Mercador.

3.2. **Visão Ação Fiscal PF**
(*não detalhada nesse estudo de caso*)

4. Restrições de Agregação

Devido ao volume de dados, serão representadas apenas as restrições com nível menor que 1 (um), i.e., aquelas que não permitem todos os tipos de operação de agregação, segundo a classificação presente no Apêndice abaixo.

Métrica	Dimensão	Nível de Restrição
Quantidade de Declarações em Malha PJ	Tempo	Nível 2
Quantidade de CNPJ Válidos	Tempo	Nível 3
Quantidade de CPF Válidos	Tempo	Nível 3
Percentual de Crescimento Arrecadação	TODAS	Nível 4

5. Apêndice – Classificação do Nível de Restrição Agregacional

Nível	Funções de agregação aplicáveis
1	{ SUM, AVG, MAX, MIN,STDDEV, VAR, COUNT }
2	{ AVG, MAX, MIN, STDDEV, VAR, COUNT }
3	{ COUNT }
4	{ }

Figura 23. Especificação dos Requisitos Multidimensionais no Projeto SAFE (*cont.*).

Histórico de Revisões

Data	Versão	Descrição	Autor	Revisor
09/04/2001	1.3	Promoção das correções-padrão para o caso de uso genérico.	**(sigiloso)**	**(sigiloso)**

Especificação de Caso de Uso

1. Descrição do Caso de Uso

1.1. **Nome**

Extrair Dados

1.2. **Propósito**

Descrever os passos comuns do processo de extração de dados do projeto SAFE.

2. Fluxo de Eventos Básico

P1. Abrir Arquivo(s) da Fonte Provedora

P2. Obter Dados Operacionais

P3. Traduzir Dados

P4. Corrigir Inconsistências Padrão

Para os dados *inválidos*, *não-informados* ou para os quais o conteúdo *não se aplique* à condição de extração, formatar o conteúdo do campo de saída respectivamente com os códigos -7, -8 e -9. (Vide passo alternativo A2 para informações complementares)

P5. Agregar Dados

P6. Totalizar Dados

Gravar os totais de dados acumulados por nível de classificação. Gravar totais gerais de dado agregado, classificação do dado agregado, qtde. registros lidos, qtde. registros processados e qtde. registros rejeitados.

P7. Gerar Arquivo de Carga

Gerar um arquivo de saída no ambiente de destino (Grande Porte ou Ambiente Oracle), com as informações extraídas, conforme leiaute para a Carga constante do Anexo desse documento. O caso de uso se encerra.

3. Fluxos Alternativos

A1. Arquivo Indisponível

No passo P1, se o arquivo estiver indisponível, uma mensagem de erro é formatada no procedimento BATCH e o caso de uso se encerra.

A2. Registro Rejeitado

No passo P3, se algum dado constante do registro de entrada apresentar uma das inconsistências a seguir:

Figura 24. Caso de Uso Geral da Extração de Dados Operacionais.

- Dado fora do domínio estabelecido pela Fonte Provedora;
- Data Inválida;
- Dados nulos ou brancos (quando esta condição não for admissível);

O sistema rejeita o registro inconsistente e gera um LOG dos casos de inconsistência. O caso de uso retoma do passo P1 com a leitura de outro registro.

4. Pré-Condições

- As bases das fontes provedoras devem estar disponíveis para acesso antes do processamento.
- A extração dos dados nas fontes provedoras somente pode ser realizada dentro do prazo acordado com a equipe responsável.

5. Pós-Condições

Não se aplica.

6. Pontos de Generalização

G1. Extrair Dados Cadastro Pessoa Jurídica
Detalha os passos de P1 a P4 para a extração de dados cadastrais do contribuinte pessoa jurídica a partir da base CNPJ.

G2. Extrair Dados Declaração Pessoa Jurídica
Detalha os passos de P1 a P4 para a extração de dados da Declaração de Renda do contribuinte pessoa jurídica a partir da base DIPJ.

G3. Extrair Dados CPMF
Detalha os passos de P1 a P4 para a extração de dados sobre o recolhimento de Contribuição sobre Movimentações Financeiras a partir da base CPMF.

G4. Extrair Dados Ação Fiscal
Detalha os passos de P1 a P4 para a extração de dados sobre as atividades de fiscalização sobre o contribuinte pessoa jurídica a partir da base de Fiscalização Eletrônica.

Figura 24. Caso de Uso geral da Extração de Dados Operacionais (*continuação*).

Histórico de Revisões

Data	Versão	Descrição	Autor	Revisor
20/04/2001	1.6	Correções nos passos segundo sugestão da equipe IRPJ.	**(sigiloso)**	**(sigiloso)**

Especificação de Caso de Uso

1. Descrição do Caso de Uso

1.1. **Nome**

Extrair Dados Cadastro Pessoa Jurídica

1.2. **Propósito**

Descrever os passos para a extração e transformação dos dados cadastrais da pessoa jurídica, a partir da Fonte CNPJ.

2. Fluxo de Eventos Básico

P1. Abrir Arquivo(s) da Fonte Provedora
Abrir o Cadastro de Pessoas Jurídicas.

P2. Obter Dados Operacionais
Obter a UA do Contribuinte, Município, Período (Ano e Mês da ocorrência do fato), Atividade Econômica, Situação Cadastral, Natureza Jurídica, Perfil e Faixa de Qtde. de Empregados de todas as empresas e estabelecimentos no Cadastro a partir de 1995.

P3. Traduzir Dados
Converter o conteúdo dos dados obtidos para o leiaute do DW para a Tabela-Fato "Ação Fiscal PJ", de acordo com as regras abaixo:

P3.1. Gravar o conteúdo da UA do Contribuinte no atributo "Domicílio Fiscal", considerando-se a UA com 7 posições numéricas.

P3.2. Gravar o Município de Localização no atributo "Município", considerando-se o município no CNPJ com 4 posições numéricas.

P3.3. Converter a Atividade Econômica do contribuinte para o campo "Atividade Econômica", de acordo com a regra de negócio 2.1 descrita no artefato *Especificação de Regras de Negócio*.

P4. Corrigir Inconsistências Padrão

P5. Agregar Dados
Obter as quantidades de empresas e estabelecimentos, classificados por Domicílio Fiscal, Município, Período, Atividade Econômica, Situação Cadastral, Natureza Jurídica, Perfil e Faixa de Qtde. de Empregados.

P6. Totalizar Dados

P7. Gerar Arquivo de Carga

Figura 25. Especialização do Caso de Uso "Extrair Dados" para a Fonte "Cadastro Pessoa Jurídica".

3. Fluxos Alternativos

A1. Obter UA do Contribuinte

No passo P3, nos casos em que a UA do Contribuinte esteja inválida, acessar a base de Declaração IRPJ e recuperar a UA de jurisdição do contribuinte, atribuindo-a ao Domicílio do Declarante. Continuar o caso de uso no passo P3.

4. Pré-Condições

- A base de Declarações IRPJ deve estar disponível para a execução do passo A1.
- A primeira extração (carga inicial) deve ser feita a partir de Janeiro/1995 até o mês anterior ao mês corrente. Para os meses posteriores, os arquivos devem ser gerados nos primeiros dias do mês anterior ao mês corrente. Por exemplo, no início de outubro deve ser gerado o arquivo referente ao mês de Setembro.

5. Pós-Condições

- O atributo "Atividade Econômica" deve estar alinhado com brancos à direita. Por exemplo, para apresentar a informação A, esse atributo deve ser formatado com "Ab" (onde "b" representa um espaço em branco).
- Os campos Domicílio Fiscal, Município, Período, Quantidade de Empresas e Quantidade de estabelecimentos que apresentam conteúdo diferente dos domínios de exceção (-7, -8 e -9) devem estar alinhados com zeros à esquerda. Os casos de exceção (-7, -8 e -9) dos campos Domicílio Fiscal e Município devem ser alinhados com brancos à direita. Por exemplo, o campo de Domicílio Fiscal, deve ser formatado com -8bbbbb ou -9bbbbb (onde "b" representa um espaço em branco).

Figura 25. Especialização do Caso de Uso "Extrair Dados" para a Fonte "Cadastro Pessoa Jurídica" (*continuação*).

Histórico de Revisões

Data	Versão	Descrição	Autor	Revisor
06/04/2001	1.7	Incluir fluxos alternativos para tratamento da Visão Arrecadação.	**(sigiloso)**	**(sigiloso)**

Especificação de Caso de Uso

1. Descrição do Caso de Uso

1.1. **Nome**

Consultar Repositório

1.2. **Propósito**

Descrever o processo de consulta aos dados no Repositório do Projeto SAFE.

2. Fluxo de Eventos Básico

P1. Tomador de Decisão Efetua LOGIN
O Tomador de Decisão acessa o sistema. O direito de acesso do usuário ao sistema é checado conforme o caso de uso indicado no Ponto de Inclusão **I1**.

P2. Tomador de Decisão Seleciona Assunto
O sistema exibe uma lista de Assuntos. O Tomador de Decisão seleciona um dos assuntos.

P3. Tomador de Decisão Seleciona Critérios de Pesquisa
O sistema disponibiliza como critérios padrão a Unidade Organizacional e o Período. De acordo com o assunto, o sistema identifica a relação de demais critérios de pesquisa, conforme os passos alternativos a seguir:

A1. Apresentar Critérios da Visão "Ação Fiscal PJ"

A2. Apresentar Critérios da Visão "Ação Fiscal PF"

P4. Tomador de Decisão Seleciona os Critérios Desejados
O usuário seleciona os valores respectivos para cada critério e escolhe a opção "Executar Pesquisa".

P5. Exibir Resultado da Consulta
O sistema disponibiliza os resultados da consulta analítica. O Tomador de Decisão seleciona, a partir da tela de resultado, uma (ou nenhuma) das seguintes funções:

A10. Escolher Métrica

A11. Ativar Dimensão

A12. Alternar Dimensões

A13. Fazer *Pivot*

A14. Detalhar/Agrupar Valores

Figura 26. Especificação de Caso de Uso para a "Consulta ao Repositório".

P6. Tomador de Decisão Encerra Aplicativo
O usuário fecha a janela Web. O caso de uso se encerra.

3. Fluxos Alternativos

A1. Apresentar Critérios da Visão "Ação Fiscal PJ"
No passo P3, se o assunto for "Ação Fiscal PJ", o sistema disponibiliza os critérios Grupo CNAE, Faixa de Quantidade de Funcionários, Opção de Tributação e Situação Cadastral. O caso de uso executa o fluxo alternativo A9, Pesquisar Valores.

A2. Apresentar Critérios da Visão "Ação Fiscal PF"
(*não detalhado nesse estudo de caso*)

A9. Pesquisar Valores
Baseado nas tabelas de domínio, o sistema verifica o conjunto de valores possíveis para cada critério. O sistema monta a lista de valores para cada critério. O caso de uso retorna a partir do passo P4.

A10. Escolher Métrica
No passo P5, o usuário seleciona uma métrica a partir de uma lista de métricas específicas do assunto. O sistema substitui os valores da métrica vigente pelos valores da métrica selecionada. O caso de uso retorna ao passo P5.

A11. Ativar Dimensão
No passo P5, o usuário ativa/desativa a visualização dos dados referentes a uma dimensão, clicando no quadro correspondente, ou nos ícones (+) para ativar e (-) para desativar. O caso de uso retorna ao passo P5.

4. Pré-Condições

- O usuário deve estar previamente habilitado no Sistema de Controle de Acesso do DW.

5. Pós-Condições

Não se aplica.

6. Pontos de Inserção

I1. Validar Usuário
Valida o CPF e a senha do usuário junto ao Sistema de Controle de Acesso.

7. Pontos de Extensão

E1. Gerar Indicadores
Apresenta indicadores econômicos para auxiliar o entendimento sobre valores agregados da fiscalização.

Figura 26. Especificação de Caso de Uso para a "Consulta ao Repositório" (*cont.*).

Histórico de Revisões

Data	Versão	Descrição	Autor	Revisor
09/08/2002	4.2	Inclusão de novas regras de cálculo para a Visão Arrecadação	**(sigiloso)**	**(sigiloso)**

Especificação de Regras de Negócio

1. **Regras Genéricas**

1.1 **Extração e Transformação**

1.1.1 **Normalizar a data de recepção da declaração**

Se a data de entrega da declaração estiver em branco OU for inválida OU for maior que data atual
Fazer a "Data de Recepção" no repositório = Ano de exercício da declaração +
Mês da data do prazo de entrega + "01" (Dia)
Fim-se

1.1.2 **Preencher o campo UA de Jurisdição**

Se a UA de Jurisdição do contribuinte estiver em branco
Recuperar da base CPF a partir do código do CPF
Senão
Recuperar do próprio registro que está sendo trabalhado
Fim-se

1.2 **Carga de Dados**

(*não detalhado nesse estudo de caso*)

2. **Ação Fiscal PJ**

2.1 **Converter Atividade Econômica**

```
Se Código da Atividade Econômica no Cadastro CNPJ = 01 ou 02
   Mover  "A" para "Atividade Econômica"
Senão
   Se Código da Atividade Econômica =  05
      Mover  "B" para "Atividade Econômica"
   Senão
      Se Código da Atividade Econômica = 10 ou 11 ou 13 ou 14
         Mover  "C" para "Atividade Econômica"
      Senão
         (...)
     Senão
         Somar 1 à Qtde. Atividades Inválidas
         Rejeitar o registro
     Fim-se
  Fim-se
Fim-se
```

Figura 27. Extrato da Especificação de Regras de Negócio do sistema SAFE.

5.5.7 Estruturando o Repositório de Requisitos

Ao final da etapa anterior, os requisitos coletados foram inseridos no repositório da ferramenta RequisitePro®, a partir do qual podiam ser gerenciados pelo líder de projeto e técnicos do projeto. Os critérios de identificação de requisitos definidos no *Plano de Gerenciamento de Requisitos* permitiram capturar atributos para cada tipo de requisito, tais como *risco*, *situação*, *prioridade*, *aditividade*, *formato*, dentre outros, a partir dos quais foi possível classificar, priorizar e gerenciar a alocação desses requisitos ao sistema. Após classificados, os requisitos foram interelacionados de acordo com suas dependências funcionais (*ex*. passos do fluxo do caso de uso *versus* dimensões envolvidas), possibilitando assim uma gerência dos impactos e riscos quando da mudança de requisitos. *Matrizes de Rastreabilidade* foram criadas, a partir dessas correlações, para demonstrar os impactos numa forma gráfica, como no exemplo da Figura 28.

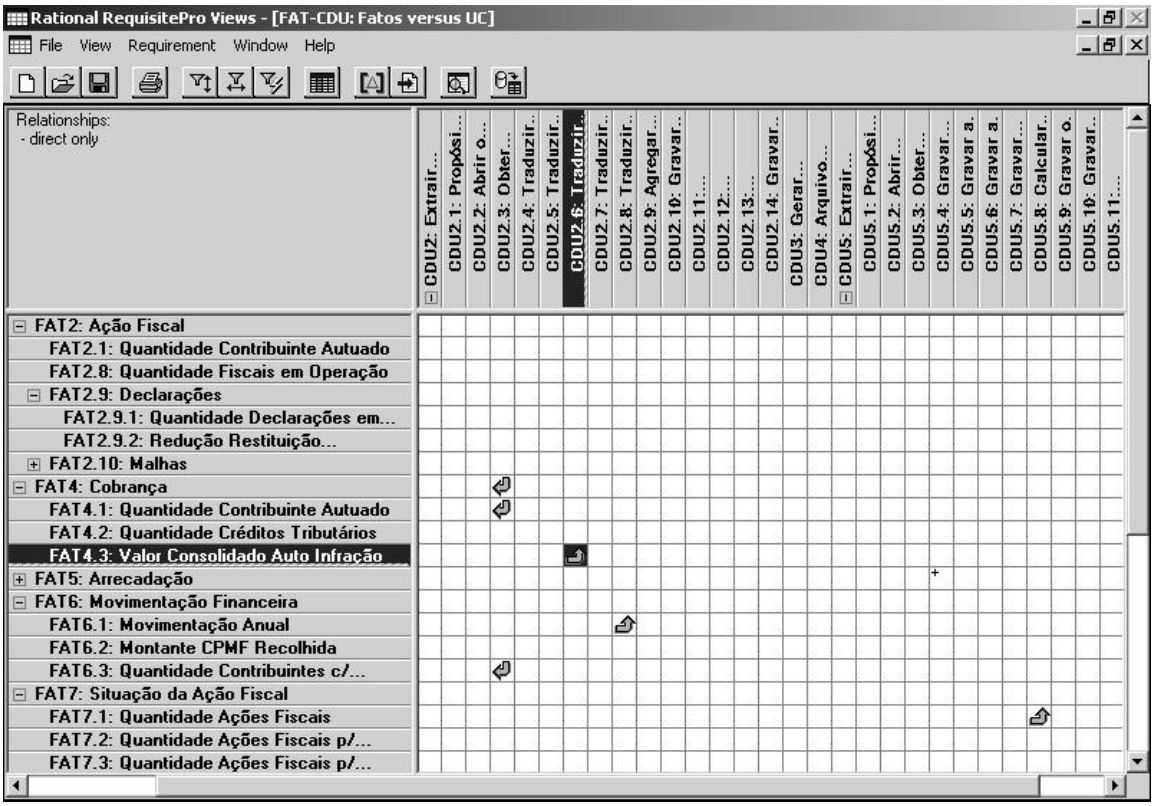


Figura 28. Exemplo de Matriz de Rastreabilidade (Fatos *versus* Casos de Uso).

As linhas da matriz denotam os requisitos do tipo "*fato*" armazenados no repositório da ferramenta. As colunas representam requisitos do tipo "caso de uso". As setas em azul na figura indicam os rastreamentos do tipo DE e PARA permitidos pela ferramenta, i.e., o requisito pode ser analisado quanto ao impacto ***para*** outro requisito causado por uma mudança aplicada ao seu conteúdo; ou quanto ao impacto que a mudança ***de*** outro registro causa em sua especificação. A ferramenta possui ainda recursos como "filtros de dados" que permitem restringir a consulta ao repositório de requisitos por variadas condições (ex. *requisitos afetados*, *situação do requisito*, *responsável pelo requisito*) e limitar a quantidade de dados retornada pela consulta.

Apesar de outras ferramentas para gerenciamento de requisitos apresentarem recursos mais elaborados para o rastreamento de mudanças (*vide* TORANZO, 2002), a abordagem clássica descrita anteriormente e suportada pela ferramenta RequisitePro® demonstrou-se bastante útil para a visualização pela equipe de projeto dos impactos causados em alterações propostas na especificação tanto do *data mart* quanto do *data warehouse*.

5.5.8 Validando e Refinando a *Linha Base* de Requisitos

As atividades descritas nas seções anteriores transcorreram-se ao longo de duas grandes iterações de desenvolvimento (Janeiro a Março/2001; Abril a Julho/2001), ao final das quais reuniões de validação foram estabelecidas para garantir a conformidade da *linha base* de requisitos produzida com as necessidades do cliente.

Ressaltamos aqui que os clientes adicionaram uma nova perspectiva ao processo de validação ao promoverem as reuniões para *workshops* abertos, onde os documentos de requisito eram discutidos entre todos os participantes, incluindo *observadores externos*, conforme recomendado em nossa abordagem. Os observadores foram escolhidos dentre técnicos experientes do SERPRO-Recife e contribuíram com questionamentos desprovidos de vínculo com os "vícios" de concepção do sistema, permitindo evidenciar aspectos conceituais e arquitetônicos não muito bem especificados (ou até mesmo esquecidos) tanto pelo lado do SERPRO quanto do cliente. Como exemplo, foi verificado que a dimensão

comum “Unidade Organizacional” não estava refletindo aspectos peculiares da estrutura do cliente como a vinculação de delegacias comuns a delegacias especiais, denominadas DRJ (Delegacias Regionais de Julgamento), com “*status*” de Superintendência. A hierarquia da dimensão apenas previa a vinculação direta entre delegacias e superintendências. A linha base foi então revista (*refinada*) e as ações estabelecidas foram aplicadas aos documentos, repositório de requisitos e esquema multidimensional. Ao final da segunda iteração, uma versão (*release*) final de requisitos estava pronta para ser entregue ao cliente e demais envolvidos com o desenvolvimento do *data warehouse*.

5.5.9 Gerenciando Mudanças

O repositório final de requisitos para o projeto SAFE em sua visão “Ação Fiscal PJ” continha aproximadamente 980 itens. Tal quantia, conforme argumentado anteriormente no corpo dessa dissertação, seria muito difícil de gerenciar sem o auxílio de uma ferramenta apropriada. Para adicionar um grau maior de dificuldade, como é natural em sistemas *data warehouse*, 60% desse total sofreu pelo menos uma mudança ou ajuste ao longo do processo de definição. A ferramenta RequisitePro® foi usada intensivamente para gerenciar o impacto dessas mudanças, possibilitando ao líder de projeto negociar ajustes com o cliente numa base mais sólida. As *matrizes de rastreabilidade* mostraram-se, num primeiro instante, um meio eficiente para visualização das dependências entre os requisitos. À medida em que o repositório de requisitos crescia, sentiu-se a necessidade de introduzir atributos de requisitos num nível mais alto de abstração para facilitar a visualização das associações entre estes, o que não eliminou a necessidade de executar análises parciais sobre o repositório em casos de mudanças de grande porte. Contudo, pôde-se observar que tal processo de análise seria praticamente impossível sem o auxílio de ferramental.

Ao passo em que os requisitos eram alocados ao sistema, sua gerência passava a requerer também uma análise dos impactos na arquitetura do projeto. A ferramenta Designer/2000® da Oracle® foi utilizada para enriquecer a gerência de mudanças e demonstrar os impactos em ambas as esferas conceituais e arquitetônicas. O DBA projetista teve papel fundamental

nessa análise, executando consultas aos *application systems* (ORACLE, 2002) do banco Oracle® para relatar as implicações de mudanças no esquema multidimensional.

5.6 Considerações sobre a aplicação da metodologia aos demais Data Mart

Após a implantação da Visão "Ação Fiscal PJ", o Projeto SAFE teve continuidade com a definição, implementação e entrega de mais uma Visão (*data mart*) para o cliente, referida como "Ação Fiscal PF". Atualmente estão em fase de construção mais dois *data marts*. No geral, a utilização da metodologia procedeu-se de forma uniforme no desenvolvimento dessas Visões e o nível de satisfação do cliente se manteve acima dos 80%. Cabe relatar, porém, dois pontos importantes surgidos com a aplicação da metodologia às demais Visões:

- Mais regras para controle da carga dos dados foram incorporadas ao Plano de Gerenciamento de Requisitos do projeto, visando solucionar problemas de sincronismo e duplicidade de informações. O caso de uso de carga foi alterado para referenciar um *metamodelo* criado especificamente para gerenciar processos de carga e extração, o qual armazena informações tais como localização da fonte, identificação de dimensões e fatos dentro do esquema integrado, momento de carga, taxa de erro aceitável, dentre outras. A implementação do metamodelo contribuiu sobremaneira para a automação dos processos do *data warehouse* e revelou a necessidade de incorporar a definição desses requisitos à etapa de especificação da metodologia. Esse ponto deverá ser alvo de uma próxima versão da metodologia;

- O maior volume de dados (4.306 requisitos (aprox.) em 4 *data marts*) e a complexidade introduzida para a rastreabilidade do modelo de requisitos forçou a criação de *tipos* de requisito intermediários para mapear o relacionamento entre requisitos funcionais e multidimensionais em diferentes níveis de abstração (ex. passos do caso de uso *versus* fatos da visão/fatos por assunto/fatos por divisão de assunto – vide Figura 28). O mesmo princípio foi aplicado às regras de negócio. Essa medida não evitou que filtros fossem aplicados às matrizes de rastreabilidade para facilitar a análise de impactos, o

que comprovou ser o volume de dados um dos fatores críticos para a gerência de requisitos em *data warehouse*.

5.7 Considerações Finais

Aplicar a metodologia ao ciclo de definição de requisitos do projeto SAFE operou uma grande mudança no paradigma *cliente-desenvolvedor* dentro da empresa SERPRO. O cliente sentiu-se parte integrante do processo de construção, o que ajudou sobremaneira no desenvolvimento de um produto de qualidade. Ademais, o uso de ferramentas para gerenciamento dos requisitos deu aos clientes uma visão mais profunda dos impactos que mudanças em requisitos podem acarretar ao desenvolvimento de um sistema, em especial de um *data warehouse*.

Por outro lado, desenvolvedores puderam notar a importância de uma boa especificação de requisitos para o sucesso do projeto. As técnicas propostas pela metodologia serviram para firmar conceitos que já estavam sendo introduzidos na empresa com o processo de implantação das práticas de nível 2 do Modelo CMM (PAULK *et al.*, 1991), tais como a necessidade de análise e validação de documentos de requisitos; e o uso de matrizes de rastreabilidade para a gerência de mudanças. Comparado com experiências anteriores no desenvolvimento de sistemas para suporte à decisão, os engenheiros puderam constatar a eficácia da metodologia em extrair e representar requisitos multidimensionais. Ainda, a equipe técnica pôde observar a influência dos aspectos qualitativos na seleção de uma arquitetura de software, ao utilizar o *framework NFR* (CHUNG *et al.*, 2000); e a importância do reuso de requisitos, principalmente em ambientes *data warehouse*, nos quais a conformidade entre requisitos dimensionais é fundamental para a sua integração e escalabilidade (PAIM *et al.*, 2002).

Apesar dos benefícios, a experiência no projeto SAFE revelou alguns pontos que não são cobertos pela metodologia tais como a especificação de hierarquias dimensionais múltiplas e os pontos de adequação quando manutenções são exigidas sobre o *data mart*/*data warehouse* implantado. Conforme descrito na Seção 5.6, o volume de dados mostrou ainda

ser um ponto crítico para o gerenciamento de requisitos em sistemas *data warehouse*, comprometendo parte da visibilidade proporcionada pelas matrizes de rastreabilidade.

Tais pontos serão objeto de estudo em novas versões da metodologia. No momento, contudo, uma análise final indica que o uso da metodologia traz inúmeros benefícios ao projeto de aplicações *data warehouse*, conforme demonstrado pela pequena incidência de erros nas Visões "Ação Fiscal PJ" e "Ação Fiscal PF" implementadas para o projeto SAFE; a cobertura das características multidimensionais obtida; a entrega dos *data marts* dentro do prazo; e a grande satisfação do cliente com os recursos do *data warehouse*. De imediato, a implantação do *data warehouse* (*i*) proporcionou uma significativa redução no percentual de apurações especiais (consultas de auditoria sobre a base de dados), anteriormente requeridas para análise de dados estratégicos, representando uma economia de 46% ao mês nos custos de análise estratégica do cliente; (*ii*) agilizou a etapa de programação das ações de fiscalização, permitindo estabelecer, de uma única vez, os parâmetros e cruzamentos de consultas de interesse dos usuários e a elaboração de estratégias de fiscalização mais abrangentes; (*iii*) proveu uma visão integrada da fiscalização, antes somente possível por meio de complicadas tabelas de correlação entre dados operacionais, e promoveu a disseminação desse conhecimento por todos os escalões da organização-cliente.

Capítulo 6

Conclusões e Trabalhos Futuros

Este capítulo mostra as principais contribuições do trabalho para a melhoria do processo de definição de requisitos em sistemas data warehouse. Em seguida, descreve trabalhos futuros que poderão ser realizados a partir dessa dissertação e finalmente fornece alguns possíveis direcionamentos para pesquisas futuras na área.

6.1 Conclusões

Aplicações *Data Warehouse* propõem uma nova visão para o processo de definição e gerenciamento de requisitos de um sistema. Diferentemente de sistemas convencionais, o processo complexo e especializado de desenvolvimento de tais aplicações, aliado à multidimensionalidade peculiar a esse domínio de software requerem uma adaptação do processo de engenharia de requisitos tradicional (SOMMERVILLE e SAWYER, 1997) para canalizar mecanismos que permitam entender, de forma correta, as necessidades de análise estratégica e suporte à decisão da organização; verificar a viabilidade e negociar uma solução de software compatível com a visão multidimensional adotada; validar a especificação produzida contra os requisitos estabelecidos; e gerenciar esses requisitos enquanto o ambiente *data warehouse* é construído.

Com essa motivação em mente, definimos nesse trabalho uma metodologia para a especificação e gerência de requisitos em aplicações *data warehouse*. Nossa abordagem fornece um arcabouço de técnicas, artefatos e procedimentos projetados para auxiliar engenheiros de software na tarefa de elaborar uma definição de requisitos precisa do *data warehouse* e seus *data marts*, segundo um processo iterativo de discussão entre as partes interessadas; especificação detalhada do modelo de requisitos coletado; e gerência do ciclo de vida do modelo ao longo do desenvolvimento do projeto. A metodologia estende, ao mesmo tempo: (*i*) o processo de engenharia de requisitos tradicional, para tratar de forma original aspectos próprios ao domínio de *data warehouse* como multidimensionalidade, aditividade de fatos e conformidade de requisitos; (*ii*) e as abordagens de desenvolvimento de aplicações *data warehouse* existentes, fornecendo um processo orientado a fases para a especificação detalhada, evolutiva e em alto nível dos requisitos do sistema.

Nesse sentido, a metodologia resolve alguns dos problemas por trás de inúmeros fracassos em projetos *data warehouse* ao (*a*) promover o entendimento correto entre desenvolvedores e clientes por meio de técnicas de elicitação, análise e negociação de requisitos; (*b*) estabelecer a separação entre aspectos conceituais e de implementação para gerar uma

especificação de requisitos autônoma e reutilizável; (*c*) vincular o modelo de requisitos da aplicação às reais necessidades do tomador de decisão e derivar o esquema multidimensional a partir de um conjunto conciso de necessidades e funcionalidades associadas; (*d*) fornecer uma estrutura de documentação adequada e padronizada para descrever processos e requisitos entre as partes envolvidas que definem a dinâmica do sistema, em especial aqueles referentes à integração com fontes provedoras; (*e*) evidenciar a existência de falhas ou restrições nas cadeias de agregação entre fatos e dimensões, e assim possibilitar a construção de um esquema multidimensional mais robusto; (*f*) ressaltar a complexidade das interelações entre fatos e dimensões no modelo de requisitos com forma de definir opções de consultas analíticas mais adequadas; (*g*) garantir a conformidade entre os requisitos da aplicação, tanto em relação àqueles ligados à multidimensionalidade do *data warehouse* corporativo, quanto a fatores de ordem funcional (e não-funcional) comuns à aplicação como um todo; (*h*) planejar uma estratégia de gerência dos requisitos da aplicação para evitar desvios na condução do projeto em relação aos seus objetivos primordiais; (*i*) estabelecer uma gerência efetiva das mudanças em requisitos e seus impactos para o desenvolvimento do *data warehouse*; dentre outras características unicamente providas por esta abordagem.

Ao longo de um estudo de caso real, mostramos como essa abordagem metodológica, aplicada ao desenvolvimento de um *data warehouse* de grande porte, garantiu uma melhor identificação e representação dos requisitos do usuário, o que propiciou definir tanto um esquema dimensional quanto processos para alimentação do repositório de dados em atendimento às necessidades de suporte à decisão do cliente. Além desses benefícios, ressaltamos outras contribuições dessa dissertação:

- Definição de um modelo de fases genérico para suporte à aplicação de práticas de engenharia de requisitos ao ciclo de desenvolvimento de *data marts* e *data warehouses*. Por meio do modelo proposto, o processo de engenharia de requisitos é adaptado para as características próprias de aplicações de suporte à decisão, servindo de ponte entre a engenharia de requisitos e o processo de *data warehousing*;

- Definição de uma forma de documentação padrão para a especificação de sistemas *data warehouse*, baseada numa coleção de artefatos especificamente desenhados para dar apoio à descoberta e documentação dos requisitos de tais aplicações. Os artefatos são descritos na forma de *templates* (modelos) auto-explicativos, que facilitam a sua instanciação e disseminação entre a equipe do projeto;

- Definição de um *template* para a modelagem de casos de uso em UML para sistemas *data warehouse*;

- Definição de uma abordagem para a conformidade de requisitos em aplicações de suporte à decisão, como forma de garantir a integração e evolução da aplicação e, em particular, do seu modelo de requisitos multidimensionais;

- Definição de uma lista de verificação (*checklist*) específica em auxílio à análise e negociação de requisitos, durante a fase de especificação de sistemas *data warehouse*;

- Conceituação e exemplificação do funcionamento de um ciclo de definição de requisitos para *data warehouse* e sua contribuição para a definição e gerenciamento dos requisitos do sistema durante o seu ciclo de desenvolvimento;

- Definição de uma extensão do Framework NFR de CHUNG *et al*. (2000), denominada DW-ENF (*Data Warehouse-Extended NFR Framework*), para auxiliar na análise da influência dos requisitos não-funcionais no projeto do *data warehouse* e na seleção de uma arquitetura ideal para a aplicação (PAIM e CASTRO, 2002a).

Contudo, a experiência prática no contexto do projeto SAFE permitiu identificar algumas dificuldades na utilização da metodologia, bem como alguns pontos de requerem melhorias:

- A metodologia não fornece meios para o tratamento adequado da especificação de hierarquias múltiplas em dimensões. Todavia, acreditamos que uma correta abordagem para o tema exige alguma forma de representação visual, possivelmente gráfica, dos atributos e sua organização nas estruturas hierárquicas, para complementar a análise

desses requisitos. Esse fator sugere a construção de um suporte ferramental para superar os limites que o papel impõe a tal representação. Viabilizar essa infraestrutura, porém, estava além do foco dessa dissertação;

- Mais do que analisar o impacto de mudanças, sentiu-se a necessidade de instrumentos para o controle do fluxo de solicitações de mudança do cliente. Processos e ferramentas nesse sentido não estão definidos na nossa abordagem atual;

- A metodologia provou-se eficaz para projetos de desenvolvimento de *data warehouse*. Contudo, sua adequação a projetos de manutenção requer ajustes, notadamente no seu arcabouço de artefatos, para torná-la aplicável a esses casos;

- A abordagem de CHUNG *et al*. (2000) e sua derivação, o *Framework* DW-ENF, apesar de serem de fácil entendimento e de existir uma ferramenta (OME3, 2000) que automatiza a checagem *bottom-up* da satisfação dos *softgoals*, exigem um tempo de dedicação dos técnicos envolvidos para dominarem sua notação. Os gráficos de *softgoal* podem, por outro lado, adquirir uma extensão que dificulta a sua análise mesmo em ambientes apropriados como o da ferramenta OME3.

- A metodologia não provê uma heurística para o reconhecimento sistemático de dimensões e fatos a partir das descrições de necessidades e funcionalidades. Todavia, até onde sabemos, uma solução adequada ainda não existe para esse problema. A proposta de BOEHNLEIN e ULBRICH-VON ENDE (1999) é uma das que melhor trata a questão, fornecendo um método passo a passo para a derivação de métricas de fato, dimensões e restrições de integridade a partir de requisitos de negócio. Contudo, a abordagem não elimina a subjetividade do tema, especialmente em relação à derivação de dimensões, ainda sujeita a uma certa dose de criatividade por parte do engenheiro da aplicação;

- A experiência com o projeto SAFE demonstrou a necessidade de enriquecer a fase de especificação da metodologia para oferecer um tratamento apropriado de requisitos para

metadado, um aspecto que vem sendo reconhecido como importante para tornar o *data warehouse* mais ágil e robusto (KIMBALL, 1998a; STAUDT *et al*., 1999).

6.2 Trabalhos Futuros

Certamente muito dos pontos de melhoria apresentados na seção anterior serão alvo de trabalhos futuros. Pela própria urgência do tema para a continuidade do projeto SAFE, deveremos buscar num primeiro instante definir uma versão da metodologia adequada ao tratamento de requisitos em projetos *data warehouse* de manutenção. Em seguida, estudos serão conduzidos no sentido de incorporar novos artefatos e procedimentos à metodologia para possibilitar a especificação de um modelo de metadados para o *data warehouse*.

Alguns temas interessantes para pesquisa também podem ser derivados a partir de nossa proposta:

- Investigar como abordagens mais sofisticadas de rastreamento tais como as descritas em (TORANZO, 2002) podem ser incorporadas à metodologia para expandir o potencial de gerenciamento de mudanças e impactos nos requisitos do *data warehouse*;

- Como forma de prover uma especificação conceitual mais completa para a aplicação *data warehouse*, um possível trabalho seria a integração entre a metodologia aqui proposta e a abordagem de TRUJILLO *et al*. (2001) para modelagem conceitual de *data warehouses*. A proposta em questão introduz uma extensão da UML (BOOCH *et al*., 1999) para representação de elementos multidimensionais do esquema "estrela" e assim encapsular requisitos do *data warehouse* num modelo conceitual. Esse requisitos, descobertos durante a etapa de especificação do *data warehouse* em nossa abordagem, serviriam de entrada para a ferramenta CASE "*GOLD Model*" desenvolvida pelos autores, que por sua vez se integra com ferramentas OLAP para geração das tabelas relacionais. Uma forma de viabilizar essa junção de abordagens seria a integração entre a ferramenta de gerenciamento de requisitos adotada (*ex*. RequisitePro®) e o referido CASE.

- Outro trabalho de pesquisa consiste na extensão da metodologia para melhor tratar requisitos organizacionais. Uma técnica importante nesse sentido é a i* (YU, 1995). Sua junção ao nosso trabalho ampliaria o poder de representação da técnica de caso de uso em relação a fatores organizacionais tais como *razões internas* (incluindo tarefas, recursos, objetivos, *softgoals* e ligações do tipo meio-fim e de decomposição de tarefa) associadas a atores organizacionais.

REFERÊNCIAS BIBLIOGRÁFICAS

ABELLÓ, A., SAMOS, J., SALTOR, F., 2000. “Benefits of an Object Oriented Multidimensional Data Model”. Lecture Notes in Computer Science, a. 1944. In: Proceedings of Objects and Database 2000 (ECOOP Workshop), France, pg. 141.

AGRAWAL, R., GUPTA, A., SARAWAGI, S., 1995. “Modeling Multidimensional Databases”. Research Report, IBM, Almaden Research Center (Set).

ALENCAR, F., 1999. “Mapeando a Modelagem Organizacional em Especificações Precisas”. Centro de Informática, Universidade Federal de Pernambuco, Tese de Doutorado, Dezembro/1999.

ALFORD, M., LAWSON, J., 1979. *Software Requirements Engineering Methodology (Development)*. RADC-TR-79-168, U. S. Air Force Rome - Air Development Center, Griffiss, AFB, NY (Jun), (DDc-AD-A073132).

ALVES, C., 2001. “Seleção de Produtos de Software Utilizando uma Abordagem Baseada em Engenharia de Requisitos”. Centro de Informática, Universidade Federal de Pernambuco, Tese de Mestrado, Março/2001.

ANTON, A. I., McCRACKEN, W. M., POTTS, C., 1994. "Goal Decomposition and Scenario Analysis in Business Process Reengineering". In: Proceedings of the 6th International Conference on Advanced Information Systems Engineering (CAiSE'94), Springer, Utrecht, NL (Jun), pp. 94-104.

ARANGO, G., 1988. Domain Engineering for Software Reuse. Ph.D. Thesis, Department of Computer Science, University of California, Irvine.

ASCENTIAL, 2002. *ITERATIONS® – Proven Methodology for Successful Business Intelligence Infrastructure*. Ascential Software White Paper, September/2002. http://www.ascentialsoftware.com/cgi-bin/litlib.cgi?URL=iterations.pdf.

ASG, 2001. *Requirements Engineering Tools*. The Atlantic Systems Guild Inc., http://www .systemsguild.com/GuildSite/Robs/retools.html

BALZER, R. *et al*., 1988. RADC System/Software Requirements Engineering Testbed Research and Development Program. Report TR-88-75, Rome Air Development Center, Griffiths Air Force Base, NY (Jun).

BARALIS, E., PARABOSCHI, S., TENIENTE, E., 1997. "Materialized View Selection in Multidimensional Database". In: Proceedings of the 23rd Very Large Database Conference (VLDB97), Athens, Greece, pp. 156-165.

BATINI, C., CERI, S., NAVATHE, S. B., 1992. *Conceptual Database Design – An Entity-Relationship Approach*. Benjamin/Cummings, Redwood City.

BELL, T. E., THAYER, T. A., 1976. "Software Requirements: Are They Really a Problem?". In: Proceedings of International Conference on Software Engineering (ICSE-2), San Francisco, CA, pp. 61-68.

BOEHM, B. W., 1981. *Software Engineering Economics*. Prentice Hall, Englewood Cliffs, NJ.

BOEHM, B. W., 1984. *Verifying and Validating Software Requirements and Design Specification*. IEEE software, Vol. 1 (Jan), pp. 94-103.

BOEHM, B. W., 1988. "A Spiral Model for Software Development and Enhancement". *Computer*, Vol. 21, No. 5 (May), pp. 61-72.

BOEHM, B. W., 1998a. "Using the WINWIN Spiral Model: A Case Study". *Computer*, Vol. 31, No. 7 (Jul), pp. 33-44.

BOEHM, B., 2000. "Requirements that Handle IKIWISI, COTS and Rapid Change". *Computer*, 33(7):99-102, IEEE Computer Software Press.

BOEHNLEIN, M., ULBRICH-VOM ENDE, A., 1999. "Deriving Initial Data Warehouses Structures from the Conceptual Data Models of the Underlying Operational Information Systems". In: Proceedings of Workshop on Data Warehousing and OLAP (DOLAP), Kansas City, MO, USA.

BOOCH, G., RUMBAUGH, J., JACOBSON, I., 1999. *The UML User Guide*. Addison-Wesley, NY.

BREITNER, C. A., 1997. "Data Warehousing and OLAP: Delivering Just-In-Time Information for Decision Support". In: Proceedings of the 6th International Workshop for Oeconometrics, 6(4), Karlsruhe, Germany (Jun).

BUBENKO, J. A., WANGLER, B., 1993. "Objectives Driven Capture of Business Rules and of Information System Requirements". IEEE Systems Man and Cybernetics'93 Conference, Le Touquet, France.

BUBENKO, J., ROLLAND, C., LOUCOPOULOS, P., de ANTONELLIS, V., 1994. "Facilitating 'fuzzy to formal' Requirements Modelling". In: Proceedings of IEEE International Conference on Requirements Engineering (ICRE), Colorado Springs, USA.

BULOS, D., 1996. *A New Dimension*. In: Database Programming & Design: 6/1996, pp. 33-37.

CABIBBO, L., TORLONE, R., 1998. "A Logical Approach to Multidimensional Databases". Lecture Notes in Computer Science a. 1377. In: Proceedings of the 6th International Conference on Extending Database Technology (EDBT98), Valencia, Spain (Mar), pp. 183-197.

CARVALHO, A. E., 2002. "Uma Estratégia para Implantação de uma Gerência de Requisitos Visando a Melhoria dos Processos de Software". Centro de Informática, Universidade Federal de Pernambuco, Tese de Mestrado, Fevereiro/2002.

CASTRO, J., KOLP, M., MYLOPOULOS, J., 2001. “A requirements-driven development methodology”. In: Proceedings of the 13th International Conference on Advanced Information Systems Engineering (CAiSE-01), Interlaken, Switzerland.

CASTRO, J., KOLP, M., MYLOPOULOS, J., 2002. “Towards Requirements-Driven Information Systems Engineering: The TROPOS Project”. *Information Systems* 27(6): 365-389.

CHAUDHURI, S., DAYAL, U., 1997. "An Overview of Data Warehousing and OLAP Technology". ACM SIGMOD Record (Mar), Vol. 26, No. 1, pp. 65-74.

CHEN, P., 1976. "The Entity-Relationship Model - Towards a Unified View of Data", ACM Transaction on Database Systems, Vol. 1, No. 1, pp. 9-36.

CHUNG, L., NIXON, B. A., YU, ERIC, 1994. "Using Quality Requirements to Systematically Develop Quality Software". In: Proceedings of the 4th International Conference on Software Quality, McLean, VA, USA (Oct).

CHUNG, L., NIXON, B., 1995. "Dealing with Non-Functional Requirements: Three Experimental Studies of a Process-Oriented Approach". In: Proceedings of the IEEE 17th International Conference on Software Engineering (ICSE), Seattle (Apr), pp. 25-37.

CHUNG, L., NIXON, B., YU, E., MYLOPOULOS, J., 2000. *Non-Functional Requirements in Software Engineering*. Kluwer Publishing.

CHUNG, L., SUBRAMANIAN, N., 2001. “Process-Oriented Metrics for Software Architecture Adaptability". In: Proceedings of the 12th International Symposium on Software Reliability Engineering (ISRE), Hong Kong, China.

COAD, P., YOURDON, E., 1991. *Object-OrientedAnalysis*. Prentice-Hall, Second Edition.

CODD, E. F., 1972. “Further Normalization of the Database Relational Model”. In: R. Rustin (editor), Database Systems, number 6 in Courant Institute Computer Science Symposia Series, Prentice Hall, Englewood Cliffs, NJ, pp. 33-64.

CODD, E. F., CODD, S. B., SALLEY, C. T., 1993, "Providing OLAP (Online Analytical Processing) to User Analyst: an IT Mandate". Arbor Software White paper, http://www.arborsoft.com/OLAP.html.

CODD, E.F., 1970. “A Relational Model for Large Shared Databanks”. Communications of the ACM (Jun), 13(6), pp. 377-387.

COLBERT, E., 1989. “The Object-Oriented Software Development Method: A Practical Approach to Object-Oriented Development. In: Proceedings of the Conference on Tri-Ada'89 - Ada Technology in Context: Application, Development and Deployment, Pittsburgh, Pennsylvania, USA (Jan), pp. 400-415.

COMPUTER, 1985. Special Issue on Requirements Engineering, IEEE Computer.

CYSNEIROS, G., 2002. “Ferramenta para o Suporte do Mapeamento da Modelagem Organizacional em i* para UML”. Centro de Informática, Universidade Federal de Pernambuco, Tese de Mestrado, 2002.

CYSNEIROS, L. M., LEITE, J., 2001. "Using the Language Extended LExicon to Support NFR Elicitation". In: Proceedings of the 5th Workshop on Requirements Engineering, Buenos Aires, Argentina (Nov).

DARDENNE, A., VAN LAMSVEERDE, A., FICKAS, S., 1991. “Goal Directed Requirements Acquisition in Requirements Elicitation”. In: Proceedings of the 6th International Workshop on Software Specification and Design, Como, Italy.

DARDENNE, A., VAN LAMSWEERDE, A., FICKAS, S., 1993. “Goal Directed Requirements Acquisition”. *Science of Computer Proggraming*, 20, pp.3-50.

DASGUPTA, S., 1991. *Design Theory and Computer Science*. Cambridge University Press, Cambridge.

DAVIS, A., 1992. “Operational Prototyping: A New Development Approach”. *IEEE Software* (Sep/Oct).

DAVIS, A., SITARAM, P., 1994. “A Concurrent Software Process Model for Software Development”. *Software Engineering Notes*, ACM Press, Vol. 19, No. 2 (Apr), pp. 38-51.

DELISLE, N., GARLAN, D., 1990. “A Formal Specification of an Oscilloscope”. *IEEE Software*, 7(5):29-36.

DeMARCO, T., 1978. *Structured Analysis and System Specification*. Yourdon Press.

DEUTSCH, L. P., 1989. "Design Reuse and Frameworks In The Smalltalk-80 Programming System". *Software Reusability*, Vol II, Eds. Ted J. Biggerstaff e Alan J. Perlis, ACM Press.

DEVLIN, B. A., MURPHY, P. T., 1988. “An Architecture for a Business and Information System”. *IBM Systems Journal*, 27(1).

DOBSON, J. S., 1992. “A Methodology for Managing Organizational Requirements”. University of Newcastle upon Tyne, UK.

DORFMAN, M., THAYER, R. H., 1990. *Standards, Guidelines and Examples of System and Software Requirements Engineering*. Los Alamitos, CA, IEEE Computer Society Press.

EASTERBROOK, S., 1994. “Resolving Requirements Conflicts with Computer Supported Negotiation”. J. Goguen, M. Jirotka (ed.s) *Requirements Engineering: Social and Technical Issues*, Academic Press, 41-66.

EDS, 2002. *System Level Automation Tool for Engineers* (SLATE®). http://www.tdtech.com.

ESPITI, 1996. European Software Institute, European User Survey Analysis. Report USV_EUR 2.1 ESPITI Project.

FICKAS, S. F., 1987. *Automating the Software Specification Process*. Technical Report 87-05, Computer Science Department, University of Oregon (Dec).

FLOYD, C., 1984. "A Systematic Look at Prototyping". In: Buddle R. (eds.) *Approaches to Prototyping*, Springer-Verlag, Berlin.

GIOVINAZZO, W., 2000. *Object-Oriented Data Warehouse Design*. Prentice Hall, first edition.

GOLFARELLI, M., MAIO, D., RIZZI, S., 1998. "The Dimensional Fact Model: A Conceptual Model for Data Warehouses". *International Journal of Cooperative Information Systems*, Vol. 7(2&3), pp.215-247.

GOLFARELLI, M., RIZZI, S., 1999. *Designing the Data Warehouse: Key Steps and Crucial Issues*. Journal of Computer Science and Information Management, Vol. 2, No. 3.

GOTEL, O., FINKELNSTEIN, A., 1994. "An Analysis of the Requirements Traceability Problem". In: Proceedings of the First International Conference on Requirements Engineering, Colorado Springs, USA (Apr).

GRAHAM, S., COBURN, D., OLESON, C., 1996. *The Foundations of Wisdom: A Study of the Financial Impact of Data Warehousing*. International Data Corporation (IDC) Ltd., Canada.

GREENSPAN, S., BORGIDA, A., MYLOPOULOS, J., 1994. " A Requirements Modeling Language and Its Logic". *Information Systems*, Vol. 11, No. 1, pp. 9-23.

HADDEN, E., KELLY, S., 1997. *The Hadden-Kelly Data Warehouse Method 4.0.* Hadden & Company – Management Consultants, http://www.hadden-kelly.com/method.htm.

HÜSEMANN, B., LECHTENBÖRGER, J., VOSSEN, G., 2000. "Conceptual Data Warehouse Design". In: Proceedings of the 2nd International Workshop on Design and Management of Data Warehouses (DMDW), M. Jeusfeld, H. Shu, M. Staudt, G. Vossen (eds.), Stockholm, Sweden (Jun).

IEEE, 1984. IEEE Std. 830 - IEEE Guide to Software Requirements Specification. The Institute of Electrical and Electronics Engineers, New York, USA.

IEEE, 1998. IEEE/ANSI 830-1998, Recommended Practice for Software Requirements Specifications, IEEE, NY.

INMON, W. H., 1996. *Building the Data Warehouse*. John Wiley & Sons, 2nd edition.

JACKSON, M., 1995. *Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices*. First edition, Addison-Wesley, Massachusetts, USA.

JACKSON, M., ZAVE, P., 1993. “Domain Descriptions”. In: Proceedings of the IEEE International Symposium on Requirements Engineering, IEEE Computer Society Press, San Diego, CA, pp. 56-64.

JACOBSON, I., 1992. *Object Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley, New York.

JARKE, M. KURKI-SUONIO, R. (eds.), 1998. *Special Issue on Scenario Management*, IEEE Transactions on Software Engineering, 24:12.

JARKE, M., 1998. “Requirements Tracing”. ACM Communications, 41(12), pp. 32-36 (Dec).

JARKE, M., BUBENKO, J., ROLLAND, *et al*., 1993. “Theories Undelying Requirements Engineering: An Overview of NATURE at Genesis”. In: International Symposium on Requirements Engineering, San Diego, CA (Jan).

JARKE, M., JEUSFELD, M., QUIX, C., VASSILIADIS, P., 1999. "Architecture and Quality in Data Warehouses: An Extended Repository Approach". *Information Systems*, Vol. 24, No. 3, pp. 229-253.

JOHNSON, J. H., 2001. “Micro Projects Cause Constant Change”. In: Extreme Programming Conference (XP2001), Villasimius, Cagliari, Italy (May). http://www.xp2001.org/conference/news.html

KELLER, S. E., KAHN, L. G., PANARA, R. B., 1990. “Specifying Software Quality Requirements with Metrics”. In: Thayer, R. H., Dorfman M. (eds.) *System and Software Requirements Engineering*, IEEE Computer Society Press, Washington, DC, pp. 145-163.

KELLY, S., 1997. *Data Warehousing in Action*. New York, John Wiley & Sons.

KIMBALL, R., 1998. *The Data Warehouse Toolkit*. New York, John Wiley & Sons.

KIMBALL, R., THORNTHWAITE, W., REEVES, L., ROSS, M., 1998a. *The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing and Deploying Data Warehouses.* New York, John Wiley & Sons.

KOTONYA, G., SOMMERVILLE, I., 1997. *Requirements Engineering: Processes and Techniques*. Wiley, John & Sons Inc.

KRUCHTEN, P., 1999. *The Rational Unified Process*. Addison-Wesley, USA.

KUIPERS, B., 1995. *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*. MIT Press.

LEFFINGWELL, D., WIDRIG, D., 2000. *Managing Software Requirements: A Unified Approach*. G. Booch, I. Jacobson, J. Rumbaugh (eds.) The Object Technology Series, Addison-Wesley, NY.

LEHNER, W., ALBRECHT, J., WEDEKIND, H., 1998. "Normal Forms for Multidimensional Databases". In: Proceedings of the 8th International Conference on Statistical and Scientific Database Management (SSDBM), IEEE Computer Society.

LEITE, J., CASTRO, J., PINHEIRO, F., 1999. *Plataforma Tecnológica em Engenharia de Requisitos: Estratégias para o Aumento da Qualidade no Desenvolvimento de Sistemas*. http://www.cic.unb.br/~facp/per/perhome.html

LEITE, J., *et al*., 1997. "Enhancing a Requirements Baseline with Scenarios". In: proceedings of the Third IEEE International Symposium on Requirements Engineering, IEEE Computer Society Press, Los Alamitos, CA, USA, pp. 44-53.

LENZ, H-J, SHOSHANI, A., 1997. "Summarizability in OLAP and Statistical Databases". In: Proceedings of the 9th International Conference on Statistical and Scientific Databases, pp. 132-143.

LOUCOPOULOS, P., KARAKOSTAS, V., 1995. *System Requirements Engineering*, McGraw-Hill, London.

LOUCOPOULOS, P., KATSOULI, E., 1992. *Modelling Business Rules in an Office Environment*. ACM SIGOIS (Aug).

LUTZ, R. R., 1993. "Analyzing Software Requirements Errors in Safe-Critical, Embedded Systems". In: Proceedings of the First International Symposium on Requirements Engineering (RE'03), San Diego, CA, IEEE, pp. 126-133.

MACAULAY, L. A., 1996. *Requirements Engineering*. Springer, London.

MacDONALD, I. G., 1986. “Information Engineering”. In: Olle T. W., Sol H. G., e Verrijn-Stuart A. A. (eds.) *Information System Design Methodologies: Improving the Practice*, Elsevier/North Holland, Amsterdam.

MACEDO, N., LEITE, J., 1999. “Elicit@99: Um Protótipo de Ferramenta para a Elicitação de Requisitos”. In: Proceedings of the II (Ibero-American) Workshop on Requirements Engineering (WER99), Buenos Aires, Argentina (Sep).

MAIDEN, N., 1998. “CREWS-SAVRE: Scenarios for Acquiring and Validating Requirements”. *Automated Software Engineering*, 5(4): 419-446.

McDERMID, J., 1994. “Requirements Analysis: Orthodoxy, Fundamentalism and Heresy”. In: Jirotka M. e Goguen J. A. (eds.) *Requirements Engineering: Social and Technical Issues*, Academic Press, London, pp. 17-40.

McDERMID, J., ROOK, P., 1993. “Software Development Process Models”. In: Software Engineer’s Reference Book, CRC Press, pp. 15/26-15/28.

MERCURIO, V., MEYERS, B. F., NISBET, A. M., RADIN, G., 1990. *AD/Cycle Strategy and Architecture*. IBM Systems Journal, 29(2).

MICROSTRATEGY, 2003. “An Architecture for Next-Generation Business Intelligence”. MicroStrategy White Paper, 2003.

MITTERMEIR, R. T., ROUSOPOULOS, N., YEH, R. T., NG, P. A., 1990. “An Integrated Approach to Requirements Analysis”. In: P. Ng, R. Yeh (eds.) *Modern Software Engineering: Foundation and Current Perspectives*, Van nostrand Veinhold, NY, pp. 450-461.

MOHANIA, M., SAMTANI, S., RODDICK, J., KAMBAYASHI, Y., 1999. "Advances and Research Directions in Data Warehousing Technology". Research Report ACRC-99-006, School of Computer and Information Science, University of South Australia.

MOODY, D. L., KORTINK, M. A. R., 2000, “From Enterprise Models to Dimensional Models: A Methodology for Data Warehouse and Data Mart Design”. In: Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW), M. Jeusfeld, H. Shu, M. Staudt, G. Vossen (eds.), Stockholm, Sweden (Jun).

MYLOPOULOS, J., CHUNG, L., LIAO, S., WANG, H., YU, E., 2001. “Exploring Alternatives during Requirements Analysis”. IEEE Software (Jan/Feb), pp. 2-6.

MYLOPOULOS, J., CHUNG, L., NIXON, B., 1992. "Representing and Using Non-Functional Requirements: A Process-Oriented Approach". *IEEE Transactions on Software Engineering*, Vol. 18, No. 6 (Jun), pp. 483-497.

NASH, T., ONDER, J., 2002. *The Approach to Building a Business-Driven Data Warehouse*. Headstrong Company White Paper.

NAUR, P., RANDELL, B. (eds.), 1969. *Software Engineering: Report on a Conference sponsored by the NATO Science Commission*. Scientific Affairs Division, NATO, Brussels (Jan).

NELLBORN, C., BUBENKO, J., GUSTAFSSON, M., 1992. “Enterprise Modelling - The Key to Capturing Requirements for Information Systems”. Deliverable 3-1-3-R1, ESPRIT III Project 6612 (Nov).

NUSEIBEH, B., 2001. "Weaving the Software Development Process Between Requirements and Architecture". In: Proceedings of the ICSE2001 International Workshop: From Software Requirements to Architectures (STRAW-01), Toronto, Canada.

NUSEIBEH, B., EASTERBROOK, S., 2000. “Requirements Engineering: A Roadmap”. In: Proceedings of the 22nd International Conference on Software Engineering (ICSE), Limerick, Ireland (Jun).

OME3, 2000. *OME3: Organization Modelling Environment*, Eric Yu (Supervisor) e Lin Liu (Researcher), University of Toronto, CANADA, http://www.cs.toronto.edu/ km/ome/.

O'NEIL, P., GRAEFE, G., 1995. "Multiple-table Joins Through Bitmapped Join Indexes". SIGMOD Record, Vol. 24, No. 3 (Sep), pp. 8-11.

O'NEIL, P., QUASS, D., 1997. “Improved Query Performance With Variant Indexes”. In: Proceedings of the ACM SIGMOD Conference on Management of Data, Tucson, Arizona, USA (May), pp. 38-49.

ORACLE, 2002. “Documenting the Enterprise Using Oracle9i Designer”. Oracle White Paper, July, http://otn.oracle.com/products/designer/pdf/otn_9ides_doc_ent_wp.pdf.

ORACLE, 2002a. *Oracle9i Developer Suite*. http://www.oracle.com/ip/develop/ ids/ index.html?designer.html

PAIM, F. R. S., CARVALHO, A. E., CASTRO, J. B., 2002. “Towards a Methodology for Requirements Analysis of Data Warehouse Systems". In: Anais do XVI Simpósio Brasileiro de Engenharia de Software (SBES'2002), Gramado, Rio Grande do Sul, Brasil, pp. 146-161.

PAIM, F. R. S., CASTRO, J. B., 2002a. “Enhancing Data Warehouse Design with the NFR Framework”. In: Proceedings of the 5th Workshop on Requirements Engineering (WER2002), Valencia, Spain (Nov).

PAIM, F. R. S., TAVARES, H. C., 2002. "Implementing a Living Software Process". In: Proceedings of the First ICSE Workshop on Software Quality (WoSQ), Orlando, Flórida, USA (May).

PAULK, M. C., *et al*., 1991. *Capability Maturity Model for Software*. Technical Report (CMU/SEI-91-TR-24, ADA240603), Pittsburgh, PA, Software Engineering Institute, Carnegie-Mellon University.

PEDERSEN, T. B., JENSEN, C. S., 1999. “Multidimensional Data Modeling for Complex Data”. In: Proceedings of 15th International Conference on Data Engineering (ICDE), IEEE Computer Society, Sydney, Australia, pp. 336-345.

PINHEIRO, F., LEITE, J., CASTRO, J., 2003. “Requirements Engineering Technology Transfer: An Experience Report”. To appear at *The Journal of Technology Transfer*, Kluwer Academic Publishers.

POHL, K., 1993. “The Three Dimensions of Requirements Engineering”. In: Rolland C., Bodart F., Cauvet C. (eds.) 5th International Conference on Advanced Information Systems Engineering (CAiSE'93), Springer-Verlag, Paris, pp. 275-292.

POHL. K., 1996. *Process Centered Requirements Engineering*. Number 5 in Advanced Software Development Series, Wiley & Sons Ltd., England.

POPKINS, 2002. *Popkin's System Architect*. http://www.popkin.com/

POTTS, C., 1994. *Requirements completeness, Enterprise Goals and Scenarios*. Research Report, College of Computing. Georgia Tech, USA.

POWER, D. J., 2000. *Decision Support Systems Hyperbook*. Cedar Falls, IA: DSSResources.COM, PDF version, http://dssresources.com/dssbook/.

PRESSMAN, R., 2001. *Software Engineering: A Practitioner's Approach*, 5th. Edition, McGraw-Hill Series in Computer Science, New York.

PRIETO-DIAZ, R., 1990. “Domain Analysis: An Introduction”. ACM SIGSOFT, Software Engineering Notes, Vol. 15, No. 2 (Apr), pp. 47-54.

RAMESH, B., 1998. “Factors Influencing Requirements Traceability Practice”. Communications of the ACM, Vol. 41, No.12 (Dec), pp. 37-44.

RATIONAL, 2001. “Achieving ROI with Rational Requirements Management Tools”. Rational Software and IDC White Paper, December, http://www.rational.com/media/whitepapers/roirm.pdf

RATIONAL, 2001a. *Rational Unified Process: Artifacts Notation*. www.rational.com/products/rup/index.jsp

ROBINSON, R., 1996. "Put The Rapid Into RAD". *Datamation*, Vol. 42, No. 4 (Feb), 80(1).

ROSS, D., 1977a. “Structured Analysis (SA): A Language for Communicating Ideas”. *IEEE Transactions on Software Engineering*, 3(1): 16-34.

ROSS, D., SCHOMAN, A., 1977b. “Structured Analysis for Requirements Definition”. *IEEE Transactions on Software Engineering* (special issue on requirements analysis), 3(1):6-15.

ROYCE, W. W., 1970. “Managing the Development of Large Software Systems: Concepts and Techniques”. In: Western Electronic Show and Convention, Vol. 14 of WESCON Technical Papers, Los Angeles, CA, (Aug), pp. 1-9.

RUMBAUGH, J. *et al*., 1991. *Object-Oriented Modeling and Design*. First Edition, Prentice Hall, Englewood Cliffs, NJ, 1991.

SANTANDER, V. F. A. , 2002. “Integrando Modelagem Organizacional com Modelagem Funcional”. Centro de informática, Universidade Federal de Pernambuco, Tese de Doutorado, Dezembro/2002.

SAPIA, C., BLASCHKA, M., HÖFLING, G., DINTER, B., 1998. “Extending the E/R Model for the Multidimensional Paradigm”. In: Proceedings of the International Workshop on Data Warehouse and Data Mining (DWDM, in connection with ER'98) (Nov), Singapore.

SCHNEIDER, G., WINTERS, J., 1998. *Applying Use Cases: A Practical Guide*. Addison-Wesley, New York.

SCHWARTZ, J., 1975. *Construction of Software, Problems and Practicalities, in Practical Strategies for Developing Large Software Systems*. E. Horowitz (ed.), Addison-Wesley, Reading, MA.

SEN, A., JACOB, V. S., 1998. “Industrial Strength Data Warehousing”. Communications of the ACM (Sep).

SHARP, H., FINKELSTEIN, A., GALAL, G., 1999. “Stakeholder Identification in the Requirements Engineering Process”. Workshop on Requirements Engineering Process, Florence, Italy.

SHAW, M., GAINES, B., 1995. “Requirements Acquisition”. *Software Engineering Journal*, vol. 11.

SHOSHANI, A., 1982. “Statistical Databases: Characteristics, Problems and Some Solutions.” In: Proceedings of the 8th International Conference on Very Large Data Bases (VLDB), Mexico City, Mexico (Sep), pp. 208-213.

SILVERSTON, L., INMON, W.H., GRAZIANO, K., 1997. *The Data Model Resource Book*. New York, John Wiley & Sons.

SOLTYS, R., CRAWFORD, A., 1999. "JAD for Business Plans and Designs". http://www.thefacilitator.com/htdocs/article11.html

SOMMERVILLE, I., 2001. *Software Engineering*. Sixth Edition, Addison Wesley.

SOMMERVILLE, I., SAWYER, P., 1997. *Requirements Engineering: A Good Practice Guide*, Wiley, John & Sons Inc.

STAUDT, M., VADUVA, A., VETTERLI, T., 1999. "The Role of Metadata for Data Warehousing". Technical Report 99.06, Department of Information Technology, University of Zurich (Sep).

STRAW01, 2001. First International Workshop From Software Requirements to Architectures (STRAW'01), Toronto, Canada (May). http://www.cin.ufpe.br/~straw01/

SUBRAMANIAN, N., CHUNG, L., 2001. "Software Architecture Adaptability: An NFR Approach". In: Proceedings of the International Workshop on Principles of Software Evolution (IWPSE'01), Vienna, Austria (Sep). IEEE Computer Society Press.

TAVARES, H. C., PAIM, F. R., CARVALHO, A. E., 2002. "Implantando CMM Nível 2: A Estratégia SERPRO". In: Proceedings of the First Simpósio Brasileiro de Qualidade de Software (SBQS2002), Gramado, Rio Grande do Sul (Out).

TELELOGIC, 2002. *Telelogic DOORS/ERS*. http://www.telelogic.com/products/doorsers/index.cfm

TESTE, O., 2001. "Towards Conceptual Multidimensional Design in Decision Support Systems". In: Proceedings of the Fifth East-European Conference on Advances in Databases and Information Systems, Vilnius, Lithuania (Sep), pp. 25-28.

THAYER, R. H., DORFMAN, M., 1997. *Software Requirements Engineering*, 2nd edition, IEEE Computer Society Press.

TORANZO, M. A., 2002. "Uma Proposta para Melhorar o Rastreamento de Requisitos de Software". Centro de Informática, Universidade Federal de Pernambuco, Tese de Doutorado, Dezembro/2002.

TORANZO, M., CASTRO, J., 1999. "A Comprehensive Traceability Model to Support the Design of Interactive Systems". In: Proceedings of the International Workshop on Interactive System Development and Object Models (WISDOM99), Lisboa, Portugal.

TRUJILLO, J., PALOMAR, M., 1998. “An Object-Oriented Approach to Multidimensional Database Conceptual Modeling”. In: Proceedings of the ACM 1st International Workshop on Data Warehousing and OLAP (DOLAP), Washington D.C., USA.

TRUJILLO, J., PALOMAR, M., GOMEZ, J., SONG, I-Y., 2001. *Designing Data Warehouses with OO Conceptual Models*. IEEE Computer, special issue on Data Warehouses, vol. 34, no. 12 (Dec), pp. 66-75.

TRYFONA, N., BUSBORG, F., CHRISTIANSEN, J. G. B., 1999. "starER: A Conceptual Model for Data Warehouse Design". In: Proceedings of Workshop on Data Warehousing and OLAP (DOLAP), Kansas City, Missouri, USA.

VAN LAMSWEERDE, A., DARDENNE, A., DUBISY, F., 1991. “The KAOS Project: Knowledge Acquisition in Automated Specification of Software”. In: Proceedings of the AAAI Spring Symposium Series, Stanford University (Mar).

VAN LAMSWEERDE, A., 2000. "Requirements Engineering in the year 00: A Research Perspective". In: Proceedings of the 22nd International Conference on Software Engineering (ICSE), Limerick, Ireland (Jun).

VAN LAMSWEERDE, A., 2000a. “Formal Specification: A Roadmap”. In: Proceedings of the Conference on the Future of Software Engineering, Limerick, Ireland (Jun), pp.147-159.

VANDIVIER, S. E., 2001. “Data Warehouse Design & Discoverer 4.1”. AVANCO International White Paper, http://www.avanco.com/n/wp3.html.

VASSILIADIS, P., 2000. “Gulliver in the Land of Data Warehousing: Practical Experiences and Observations of a Researcher”. In: Proceedings of the 2nd International Workshop on Design and Management of Data Warehouses (DMDW), M. Jeusfeld, H. Shu, M. Staudt, G. Vossen (eds.), Stockholm, Sweden, pp. 12.1-12.16.

VASSILIADIS, P., 2000a. “Data Warehouse Modeling and Quality Issues”. PhD thesis, Department of Electrical and Computer Engineering, National Technical University of Athens, Greece.

VASSILIADIS, P., BOUZEGHOUB, M., QUIX, C., 1999. “Towards Quality-oriented Data Warehouse Usage and Evolution”. In: Proceedings of the 11th Conference on Advanced Information Systems Engineering (CAiSE '99), Heidelberg, Germany.

VILLER, S., SOMMERVILLE, I., 1999. “Social Analysis in the Requirements Engineering Process: From Ethnography to Method”. In: Proceedings of the 4th International Symposium on Requirements Engineering, Limerick, Ireland (Jun).

VOSSEN, G., 2000. *Data Models, Database Languages and Database Management System*. Fourth edition, Oldenbourg, Germany.

WASSERMAN, A., 1979. “A Specification Method for Interactive Information Systems”. In: Proceedings of SRS – Specification of Reliable Software, IEEE Catalog No. 79 CH1401-9C, pp. 68-79.

WATTERSON, K., 1998. “Second Generation Data”. *SunExpert Magazine* (Oct), pp. 58-65.

WEIDENHAUPT, K., POHL, K., JARKE, M., HAUMER, P., 1998. “Scenario Usage in System Development: A Report on Current Practice - Extented Abstract”. In: Proceedings of the Third IEEE International Conference on Requirements Engineering (ICRE’98), Colorado, USA (Apr).

WIERINGA, R. J., 1996. *Requirements Engineering: Frameworks for Understanding*. John Wiley e Sons, New York.

YOURDON, E., 1989. *Modern Structured Analysis*. Prentice Hall, NY.

YU, E., 1995. “Modelling Strategic Relationships for Process Reengineering”. Phd Thesis, University of Toronto.

YU, E., MYLOPOULOS, J., 1994. “Understanding ‘why’ in Software Process Modeling, Analysis and Design”. In: Proceedings of the 16th International Conference on Software Engineering, Sorrento, Italy.

ZAVE, P., 1997. *Classification of Research Efforts in Requirements Engineering*. ACM Computer Surveys, Vol. 29, No. 4.

ZULTNER, R., 1993. *TQM for Technical Teams*. Communications of the ACM 36(10), pp. 79-91.

GLOSSÁRIO

Artefatos	Um conjunto tangível de informações que: (*i*) é criado, modificado e utilizado pelos desenvolvedores (em papéis específicos) ao executarem as atividades, (*ii*) representa uma entidade que tem responsáveis e (*iii*) pode ser colocado sob controle de versão. Um artefato pode ser por exemplo um documento ou um modelo visual.
Checklist	Uma lista que enumera atividades, produtos ou outros itens a serem observados ou verificados. *Checklists* são lembretes, ajudam a não esquecer detalhes importantes.
CMM	Modelo de Maturidade de Capacitação (CMM) *do Software Engineering Institute – SEI*, da *Carnegie-Mellon University*. Tem como proposta a obtenção de um processo mensurável e controlado para atividades, com ações de melhorias contínuas, permitindo que a empresa busque a maturidade gradativa e planejada do processo de desenvolvimento e manutenção de software.
Critério	Requisito que restringe a execução de uma atividade.
Cubo de Dados (*Hipercubo*)	Metáfora utilizada para esclarecer a estrutural multidimensional dos dados em um data warehouse, na qual cada célula no cubo corresponde a uma unidade de dados, representativa da intersecção entre "n" dimensões num determinado ponto do espaço, onde um valor (métrica) que se deseja investigar quantifica um fato do mundo real.
Desenvolvedor	É um membro da equipe técnica ou gerencial que participa das atividades de desenvolvimento e manutenção de software. Um único desenvolvedor pode desempenhar diferentes papéis no processo. Exemplo: Um mesmo desenvolvedor pode ser arquiteto e criador de componentes.
Equipe de Projeto	Grupo de pessoas que têm responsabilidade pela execução das atividades de desenvolvimento e manutenção de software (*por exemplo*, análise de requisitos, projeto, código e teste) em um determinado projeto.
Framework	Modelo integrado de componentes que demonstra a interação e as relações possíveis entre esse componentes, dentro de um domínio de problema específico.
Marco de Projeto (*Milestone*)	É um demarcador de fases importantes do desenvolvimento do software. Marcos determinam pontos nos quais líderes de projeto e clientes tomam decisões cruciais sobre cronograma, orçamentos, produtos de entrega e requisitos do projeto. Podem ser considerados como pontos de sincronização, onde um conjunto bem definido de objetivos é atingido, artefatos são concluídos, decisões são tomadas, para então ir (ou não) para a próxima fase.
OLAP	*On-Line Analytical Processing*. Categoria de processamento *on-line* que permite o acesso interativo, consistente e rápido a um grande volume de dados sob diversas perspectivas (dimensões) diferentes.
OLTP	*On-Line Transaction Processing*. Tipo de processamento no qual o computador responde imediatamente ao pedido do usuário. Cada pedido é considerado uma *transação*.

Papel	Um papel é uma definição precisa e bem delimitada de uma função e responsabilidades a serem desempenhadas por uma ou mais pessoas. *Comentário* - Um papel pode ser desempenhado por mais de uma pessoa simultaneamente, por exemplo: um arquiteto (papel) pode ser interpretado por um grupo de desenvolvedores com habilidades complementares. Para desempenhar um determinado papel de maneira adequada o desenvolvedor deve possuir as habilidades necessárias à sua interpretação.
Plano	Documento que descreve em linhas gerais como um objetivo será alcançado e o que será necessário para alcançá-lo.
Procedimento	Descrição passo-a-passo de uma seqüência de tarefas para a realização de uma atividade. Descreve tarefas a serem executadas e identifica regras para desenvolvê-las.
Processo iterativo	Em um contexto de ciclo de vida de software, é o tipo de processo que envolve o gerenciamento de uma cadeia de versões (*releases*) executáveis.
Produto	O subconjunto de artefatos de software que são entregues ao cliente ou usuário final são denominados produtos de software.
PSDS	Processo SERPRO de Desenvolvimento de Soluções. Processo de desenvolvimento criado pelo SERPRO para organizar as atividades relacionadas com a criação, entrega e manutenção de sistemas de software para seus clientes. O PSDS baseia-se em dois processos modernos e amplamente aceitos pela comunidade de informática: o RUP (*Rational Unified Process*) e o USDP (*Unified Software Development Process*), e foi concebido como um produto de software que, como tal, pode ser mantido e evoluído por meio da atualização das páginas Web que o compõem.
ROLAP	*Relational OLAP*. Modelo arquitetural de data warehouse no qual o dado é armazenado em tabelas relacionais seguindo um esquema “estrela”, sendo posteriormente traduzido para o formato de um “cubo” multidimensional sobre o qual consultas podem ser realizadas.
Snowflake	Esquema “Flocos de Neve”. Configuração de dados multidimensionais que utiliza tabelas dimensionais normalizadas.
Parte Interessada ou *Stakeholder*	Pessoa que esteja interessada com o sucesso do desenvolvimento do sistema ou qualquer pessoa que seja afetada pelo sistema, por exemplo: usuário final, desenvolvedor, gerente, etc.
Template	Padrão a ser utilizado como guia para produzir determinado produto. Quando bem estruturado, fornece "slots" para capturar e organizar informações. Textos com orientações auxiliam a utilizá-lo adequadamente.
Tomada de decisão	Processo de estabelecimento de ações para o direcionamento estratégico de uma empresa.
Versão Final (*Release*)	É um conjunto de artefatos relativamente completo e consistente - possivelmente incluindo uma construção - entregue a um usuário interno ou externo; a entrega de tal conjunto.
SEI	*Software Engineering Institute*. Instituição de pesquisa e desenvolvimento de software financiada pelo Departamento de Defesa (DoD) americano, cuja missão é incentivar e prover meios para a melhoria dos processos de engenharia de software nas empresas.

APÊNDICE 1 - *Templates dos Artefatos de Requisitos*

Histórico de Revisões

Data	Versão	Descrição	Autor	Revisor

Plano de Gerenciamento de Requisitos

1. Projeto
<nome do projeto data warehouse.>

2. Papéis e Responsabilidades
< identifica os papéis e respectivas responsabilidades das pessoas envolvidas com a gerência dos requisitos do projeto.>

Papel	Responsabilidades	Responsável

3. Procedimentos para Gerência dos Requisitos
<descreve os procedimentos que irão regular a gerência dos requisitos do sistema no que se refere a mudanças de conteúdo, validações, criação e ateste de linhas base de requisitos, especificação de novos requisitos, dentre outros. Esses regulamentos devem definir o comportamento a ser seguido pelos atores (usuários e desenvolvedores) em cada fase do desenvolvimento da aplicação. A lista de procedimentos deve ser convenientemente organizada em seções referentes a cada fase, de forma a facilitar a leitura. Além de uma seção com Procedimentos Gerais, outras seções possíveis são Extração de Dados, Transformação de Dados e Carga de Dados, mas fases que refletem preocupações mais atuais em projetos data warehouse podem ser também incorporadas como Segurança e Atualização dos Dados.>

4. Artefatos (*opcional*)
< enumera e descreve brevemente os artefatos de projeto que deverão conter definições de requisitos.>

5. Identificação de Requisitos
<descreve como os requisitos serão classificados, identificados e numerados.>

5.1. **Tipos de Requisitos**
<especifica todos os tipos de requisitos funcionais, não-funcionais e de domínio associados ao projeto. Tipos comuns são Dimensão, Fato, Ator, Necessidade, Funcionalidade, Caso de Uso, e muitos outros. Para facilitar a classificação, um mnemônico específico deve ser criado para identificar cada tipo definido (ex.: DIM=Dimensões).>

5.2. **Atributos**
<lista todos os atributos de requisitos que serão utilizados para avaliação, acompanhamento, priorização e gerência dos requisitos do data warehouse. Atributos úteis são Risco, Esforço, Prioridade, Situação, Benefício e Nível de Agregação.Cada atributo deve ser descrito em termos de uma faixa de classificação, a qual deve ir de um nível de importância baixo até um mais alto (ex.: Prioridade pode ser classificada em Alta, Média e Baixa). Uma pequena descrição deve seguir cada nível de atributo.>

5.3. **Regra de Identificação**

<explicita a regra de identificação dos requisitos. Quando um algoritmo de numeração é provido pela ferramenta de gerenciamento de requisitos, especificar sua lógica básica.>

6. Critério de Rastreabilidade

<descreve quaisquer regras adicionais e diretrizes para o correto rastreamento dos requisitos. Restrições aplicáveis tais como "cada dimensão aprovada deve estar ligada a pelo menos uma tabela-fato"devem ser descritas nessa seção. >

Histórico de Revisões

Data	Versão	Descrição	Autor	Revisor

Glossário do Projeto

1. Introdução

O propósito do Glossário é prover uma terminologia comum sobre o projeto para todos os grupos envolvidos. Esse documento cataloga todo o tipo de informação que um leitor – possivelmente não familiarizado com a filosofia de sistemas para suporte à decisão – possa necessitar para compreender os conceitos desenvolvidos no projeto. As informações coletadas devem definir um mapeamento específico sobre o domínio do problema, explicando detalhadamente os aspectos mais relevantes. O conjunto de termos resultante pode ser usado como um dicionário de dados informal, capturando a definição de dados e deixando que os demais documentos concentrem-se no que o sistema deve fazer com a informação.

2. Organização

<descreve como o documento está organizado internamente. O agrupamento de termos pode ser bastante útil nos casos de grande volume ou variedade de informação, e sua organização deve ser descrita claramente nesta seção.>

3. Definições

<os termos definidos aqui formam a substância essencial do documento. Eles podem ser definidos em qualquer ordem, mas geralmente a ordem alfabética garante um acesso mais rápido à informação.>

3.1. **<Grupo de Termos>** (*opcional*)

<especifica o nome do grupo.>

<Nome do Termo>

<a descrição para o termo é apresentada aqui. Fornece o máximo de informações necessaries ao leitor para entender o significado do termo.>

3.<n>. **<outro Grupo de Termos>** (*opcional*)

<outro Nome de Termo>

<descrição do termo.>

4. Referências

<Essa seção deve fornecer uma lista completa de todos os documentos referenciados no corpo do Glossário. Cada documento deve ser identificado pelo título, número do relatório (se aplicável), data, e organização de publicação. Especifique as fontes a partir das quais a referência pode ser obtida. Essa informação pode ser atendida por meio de referência a outro documento ou a um apêndice.>

Histórico de Revisões

Data	Versão	Descrição	Autor	Revisor

Documento de Visão do Data Warehouse

1. Posicionamento do Projeto

1.1. **Cenário**
<descreve o cenário que motiva a construção do data warehouse.>

1.2. **Oportunidade de Negócio**
<resume a oportunidade de negócio sendo atacada pelo projeto, e os benefícios principais advindos do seu desenvolvimento, considerando-se o cenário previamente descrito. >

1.3. **Descrição**
<descreve resumidamente o propósito do sistema, enquanto provê uma visão geral dos processos que integram o seu funcionamento.>

1.4. **Objetivos**
<estabelece claramente os objetivos acordados com as partes interessadas, por meio de frases descritivas curtas.>

1.5. **Cobertura**
<descreve o alcança da aplicação no contexto de seu público alvo, condições e restrições gerais de acesso.>

1.6. **Solução Tecnológica**
<documenta a abordagem tecnológica escolhida para implementar o data warehouse e suas características funcionais associadas.>

2. **Papéis de Stakeholders e Atores**

2.<n>. **<Nome da Parte Interessada/Ator >**

Descrição	*<breve descrição sobre o ator ou parte interessada.>*
Papel no desenvolvimento da aplicação	*<descreve o papel e a importância da parte interessada ou ator para o desenvolvimento da aplicação.>*
Insumos ao Sistema	*<lista todos os produtos a serem entregues pela parte interessada ou ator para servirem de entrada ao processo de desenvolvimento.>*

3. Relação de Data Marts

3.1. **<Nome do Data Mart>**
<O objetivo principal do Data Mart é descrito aqui.>

3.<n>. **<outro Data Mart>**
<outra descrição de Data Mart.>

4. Fronteira Funcional

<apresenta o diagrama de use cases que descreve a fronteira functional do sistema.>

5. Critérios de Qualidade

<define os padrões de qualidade a serem adotados para medir a qualidade da aplicação.>

6. Restrições do Projeto

<descreve os aspectos que restringem o desenvolvimento do projeto, tanto em relação à arquitetura multidimensional quanto a limitações operacionais e legais.>

7. Marcos do Projeto (*Milestones*)

<enumera pontos de controle do projeto tanto internos quanto relativos à entrega de linhas base e documentação de requisitos para o cliente.>

Histórico de Revisões

Data	Versão	Descrição	Autor	Revisor

Documento de Visão do Data Mart

1. Visão Geral

1.1. **Identificação**
<introduz a identificação pela qual o Data Mart sera referenciado no projeto data warehouse.>

1.2. **Propósito**
<descreve os principais propósitos estratégicos endereçados pelo Data Mart.>

1.3. **Audiência**
<identifica o público alvo da visão provida pelo Data Mart.>

2. Papéis Envolvidos
<identifica, dentre os papéis definidos no Visão do Data Warehouse, aqueles diretamente relacionados com o desenvolvimento do Data Mart, incluindo seus respectivos representantes.>

Papel	Responsável

3. Necessidades do Usuário
<descreve os problemas-chave percebidos pelas partes interessadas e usuários finais, e estabelece uma associação entre essas necessidades e funcionalidades em potencial a serem implementadas para sua satisfação. Essa seção também identifica os atores relacionados com o desenvolvimento das funcionalidades e sua importância nesse processo.>

Necessidade #<n>		**Benefício**
<descrição da necessidade do usuário.>		[Crítico, Importante, Útil]
Funcionalidade(s) de Suporte:	*<relação de identificadores da(s) funcionalidade(s) que dão suporte à realização da necessidade.>*	

4. Funcionalidades

ID	Descrição	Prioridade
F<n>	*<descrição da funcionalidade>*.	[Alta, Média, Baixa]

5. Marcos da Visão "<identificador do data mart>"
<enumera pontos de controle do desenvolvimento do Data Mart tanto internos quanto relativos à entrega de linhas base e documentação de requisitos para o cliente.>

Histórico de Revisões

Data	Versão	Descrição	Autor	Revisor

Especificação de Requisitos Multidimensionais

1. Ambiente Multidimensional

<descreve o cenário multidimensional requerido pela aplicação.>

1.1. **Escopo e Diretrizes Gerais**

<descreve o escopo multidimensional geral da aplicação e estabelece diretrizes para o projeto do esquema multidimensional tais como nível de granularidade, processo de agregação, número máximo de fatos e dimensões envolvidas numa consulta, dentre outras.>

1.2. **Diagrama Multidimensional**

<representação gráfica do esquema multidimensional adotado, mostrando as entidades conceituais representativas de fatos e dimensões e seus interelacionamentos. Para esquemas complexos, o diagrama pode ser substituído por um link para a sua representação em ferramenta CASE externa.>

1.3. **Restrições Multidimensionais**

<define as restrições do esquema multidimensional que impõem limites ao desenvolvimento do projeto.>

2. Requisitos Multidimensionais Corporativos

<descreve os requisitos multidimensionais estritamente relacionados com o data warehouse corporativo.>

2.1. **Dimensões Conformadas**

<detalha a estrutura das dimensões conformadas e prove uma guia de referência para a construção dos Data Marts individuais.>

2.1.<n>. **<Nome da Dimensão>**

<uma breve descrição do propósito da dimensão deve ser acrescida aqui. A estrutura dimensional é descrita preenchendo-se a tabela a abaixo.>

Atributos	**Descrição**	**Formato/Tamanho** (*opcional*)	**Cardinalidade**
<nome do atributo>	*<descrição do atributo>*	*TipoFormato+Tamanho*	*<número máximo de ocorrências>*

2.2. **Fatos Conformados**

<lista os fatos conformados e suas correspondentes unidades de medida.>

Fatos	**Descrição**	**Métricas do Fato**	**Formato**	**Tipo**	**Aditividade**	**Fórmula**
<nome do fato>	*<descrição>*	*<nome da métrica>*	Tipo+Tam.	[Simples, Derivado]	[Aditivo, Não-Aditivo, Semi-Aditivo]	*<descreve a expressão matemática que dá origem ao fato derivado.>*

3. Requisitos Multidimensionais dos Data Mart

<define os requsitos multidimensionais com respeito à arquitetura dos Data Mart.>

3.<n>. **<Nome do Data Mart >**

3.<n>.<1>. **Dimensões**

3.<n>.<1>.<m> **<Nome da Dimensão>**

Atributos	**Descrição**	**Formato/Tamanho** (*opcional*)	**Cardinalidade**
<nome do atributo>	*<descrição do atributo>*	*TipoFormato+Tamanho*	*<número máximo de ocorrências>*

3.<n>.<2>. **Fatos**

Fatos	**Descrição**	**Métricas do Fato**	**Formato**	**Tipo**	**Aditividade**	**Fórmula**
<nome do fato>	*<descrição>*	*<nome da métrica>*	Tipo+Tam.	[Simples, Derivado]	[Aditivo, Não-Aditivo, Semi-Aditivo]	*<descreve a expressão matemática que dá origem ao fato derivado.>*

4. Restrições de Agregação

<demonstra as restrições de agregação entre fatos e atributos de dimensão. Com grandes volumes de fatos e dimensões, sugere-se representar apenas os pares de (*fato*, *dimensão*) com nível de restrição inferior a 1 (*vide apêndice abaixo*).*>*

Métrica	**Dimensão**	**Nível de Restrição**
<nome da métrica>	*<nome da dimensão>*	<identificação do nível>

5. Fontes Provedoras *(opcional)*

<Nos casos de extrações efetuadas pela própria equipe do data mart, mapeia cada campo de fonte de dado em função de seu tamanho e formato.>

5.<n>. **<Nome da Fonte>**

Campos	**Formato/Tamanho**
<nome do campo>	*TipoFormato+Tamanho*

6. Apêndice – Classificação do Nível de Restrição Agregacional *

Nível	**Funções de agregação aplicáveis**
1	{ SUM, AVG, MAX, MIN, STDDEV, VAR, COUNT }
2	{ AVG, MAX, MIN, STDDEV, VAR, COUNT }
3	{ COUNT }
4	{ }

* Adaptado da proposta de classificação HÜSEMAN *et al*. em "Conceptual Data Warehouse Design". Proceedings of the 2nd International Workshop on Design and Management of Data Warehouses (DMDW), Estocolmo, Suécia, Junho, 2000.

Histórico de Revisões

Data	Versão	Descrição	Autor	Revisor

Especificação de Caso de Uso

1. Descrição do Caso de Uso

1.1. **Nome**

<Iniciar com verbo no Infinitivo. Utilizar nome completo, onde as primeiras letras devem ser maiúsculas, exceto preposições. Este nome deve retratar claramente a ação a ser realizada.>

1.2. **Propósito**

< Deve oferecer uma idéia geral do propósito do caso de uso.>

2. Fluxo de Eventos Básico

< Descrever o cenário utilizado pelos atores para atingir o objetivo com sucesso. O caso de uso deve descrever o que o ator faz e o que o sistema responde em troca, na forma de um diálogo entre esses elementos. A numeração dos passos deve iniciar em 1. A numeração dos subpassos deve preservar o índice do passo-pai, acrescido ao número identificador do subpasso, iniciando em 1.>

P<n>. <identificação do passo básico>

<descrição do passo>

P<n.m> *<descrição do subpasso>*

3. Fluxos Alternativos

< Descrever os cenários alternativos utilizados pelos atores. A numeração dos procedimentos deve iniciar em 1. Os caminhos alternativos devem percorrer um fluxo completo, demostrando:(a) o passo a que estão associado; (b) a condição que aciona a entrada no caminho alternativo; (c) as ações tomadas no caminho alternativo; (d) a ação de retorno.>

A<n>. < identificação do passo alternativo>

<descrição do passo>

4. Pré-Condições

< identifica as condições que devem ser satisfeitas para que o caso de uso possa iniciar.>

5. Post-Conditions

identifica as condições que podem ser garantidas como verdadeiras ao final do caso de uso.>

6. Pontos de Inclusão[13] (*opcional*)

<Pontos de inclusão são referências a casos de uso externos que descrevem um comportamento comum a vários casos de uso. Identificar o passo do fluxo básico ao qual a inclusão está associada, descrevendo as condições e o momento para sua ativação.>

[13] Para identificação de inclusões, extensões e generalizações, o artefato adota o padrão para casos de uso sugerido pelo Rational Unified Process (RUP) (KRUCHTEN, 1999).

I<n>. <identificação do ponto de inclusão>
<descrição do ponto de inclusão>

7. Pontos de Extensão (*opcional*)
<Pontos de extensão são referências a outros casos de uso externos que complementam o fluxo de eventos do corpo do caso de uso "chamador". Identificar o passo do fluxo básico ao qual a extensão está associada, descrevendo as condições e o momento para sua ativação.>

E<n>. <identificação do ponto de extensão>
<descrição do ponto de extensão>

8. Pontos de Generalização (*opcional*)
<Pontos de Generalização referenciam casos de uso Filho que sobrepõem passos do caso de uso Pai para adicionar uma perspectiva particular a um dado cenário. Defina os passos comuns no caso de uso Pai, deixando os passos que variam apenas indicados por uma identificação. No caso de uso Filho, faça o oposto, detalhando apenas os passos variantes que foram indicados no caso de uso Pai, descrevendo o caminho alternativo.>

G<n>. < identificação do ponto de generalização>
<resumo do ponto de generalização>

Histórico de Revisões

Data	Versão	Descrição	Autor	Revisor

Especificação de Requisitos Não-Funcionais

1. <Categoria do Requisito Não-Funcional>

<Existe um grande número de requisitos não-funcionais (RNF) descritos na literatura. Nem todos esses requisitos se encaixam perfeitamente ao perfil da aplicação sendo desenvolvida. Os engenheiros de sistema e de requisitos devem definir os RNF que melhor se adequam ao assunto tratado pelo Data Warehouse/Data Mart. As seguintes categorias de RNF estão normalmente associadas ao desenvolvimento de sistemas data warehouse:

[Restrições Multidimensionais, Requisitos Legais, Restrições de Negócio, Confiabilidade, Performance, Interoperabilidade, Requisitos Organizacionais, Periodicidade de processos, Suportabilidade, Acessibilidade e Usabilidade].O framework DW-ENF descrito no Apêndice 2 propõe, ainda, uma árvore de classificação suplementar, especificamente voltada para o ambiente data warehouse.

(Certas coleções de RNFs podem referir-se a um Data Mart específico. Nesses casos, grupos de itens de RNF por visão do data warehouse devem ser formados para melhor organizar sua descrição, facilitando a sua leitura e posterior captura).
O seguinte padrão de organização pode ser adotado:>

2.1. <Nome do Data Mart > (*quando aplicável*)

2.1.1 *< descrição do requisito não-funcional. Quando a abordagem de agrupamento por Data Mart não for aplicável, reduzir a numeração do RNF de um nível e prosseguir com sua descrição.>*

2.<n>. <outro Data Mart>

2.<n>.1 *<outra descrição do requisito não-funcional >*

<n>. <outra Categoria de Requisito Não-Funcional >

Histórico de Revisões

Data	Versão	Descrição	Autor	Revisor

Especificação de Regras de Negócio

1. **Regras Genéricas**
   <*identifica as regras de negócio que abrangem todo o escopo do data warehouse.*>

   1.1 <**Identificação da Regra**>
   <*descrição da regra.*>

   1.<n> <**outra Identificação de Regra** >
   <*outra descrição de regra de negócio.*>

2. <**Identificação do Data Mart>**
   <*esta seção documenta as regras aplicáveis a um Data Mart específico.*>

   2.1 <**Identificação da Regra**>
   <*descrição da regra.*>

   2.<n> <**outra Identificação de Regra** >
   <*outra descrição de regra de negócio.*>

<n>. <**outro Data Mart>**

APÊNDICE 2 – O *Framework* DW-ENF

1. Introdução

O complexo processo de extração, transformação e carga de dados em aplicações *data warehouse*, bem como suas facilidades para consulta analítica das informações agregadas são governados por uma multiplicidade de fatores de qualidade. Em praticamente todos os projetos dessa natureza, clientes ávidos por um meio robusto para a tomada de decisões requerem soluções que se integrem de maneira precisa e pontual com inúmeras fontes provedoras de dados heterogêneos; apresentem resultados confiáveis de forma acurada; ofereçam uma interface OLAP flexível; e façam tudo isso apoiados num esquema multidimensional completo e enxuto. A gama de exigências anteriores revela uma série de requisitos de ordem não-funcional tais como integridade, acessibilidade, performance e usabilidade, além de outros específicos do domínio multidimensional, que precisam ser atendidos para a satisfação plena do cliente. Essa afirmação demonstra claramente a necessidade de se utilizar técnicas de modelagem de requisitos não-funcionais como o *Framework NFR* (CHUNG *et al.*, 2000) em prol de uma especificação de soluções *data warehouse* de qualidade.

Nesse breve apêndice, apresentamos uma extensão do *Framework NFR* para tratamento dos aspectos de qualidade de sistemas *data warehouse*, a qual denominamos *Data Warehouse-Extended NFR Framework* (DW-ENF) (PAIM e CASTRO, 2002a). O *framework* DW-ENF define catálogos de Tipos de Requisitos Não-Funcionais e métodos operacionais correlatos para posterior reuso durante a fase de especificação da aplicação. Os catálogos permitem que o engenheiro de software possa selecionar, dentre uma combinação de fatores qualitativos e de implementação, aqueles que melhor atendem às necessidades de suporte à decisão. Na Seção 2, detalhamos os requisitos não-funcionais que integram o *framework* e apresentamos a árvore hierárquica que descreve o catálogo de tipos. Ao final, exemplificamos catálogos de métodos operacionais para os dois principais tipos de requisitos não-funcionais explorados nessa dissertação: Performance e Multidimensionalidade.

2. Framework DW-ENF: Catálogos de Tipos e Métodos

O *Framework DW-ENF* detalha um amplo catálogo com os mais importantes requisitos não-funcionais em aplicações *data warehouse*. O catálogo foi desenvolvido a partir da proposta original de CHUNG *et al.* (2000) e tem como base vários trabalhos sobre a qualidade de dados e de software em aplicações convencionais (CHUNG e NIXON, 1995; CHUNG e SUBRAMANIAN, 2001; SUBRAMANIAN e CHUNG, 2001; PRESSMAN, 2001) e de suporte à decisão (LENZ e SHOSHANI, 1997; KIMBALL *et al.*, 1998a; JARKE *et al.*, 1999; VASSILIADIS *et al.*, 1999; VASSILIADIS, 2000a). Fazemos notar, contudo, que o referido catálogo não encerra uma lista exaustiva de Tipos e Métodos, mas ao contrário, enfoca apenas as principais categorias de fatores de qualidade que influenciam no projeto de um *data warehouse*. A Figura 29 oferece uma visão hierárquica dessas categorias e seus requisitos de qualidade associados, enquanto as Figuras 30 e 31 ilustram respectivamente os catálogos de métodos para os requisitos Performance e Multidimensionalidade.

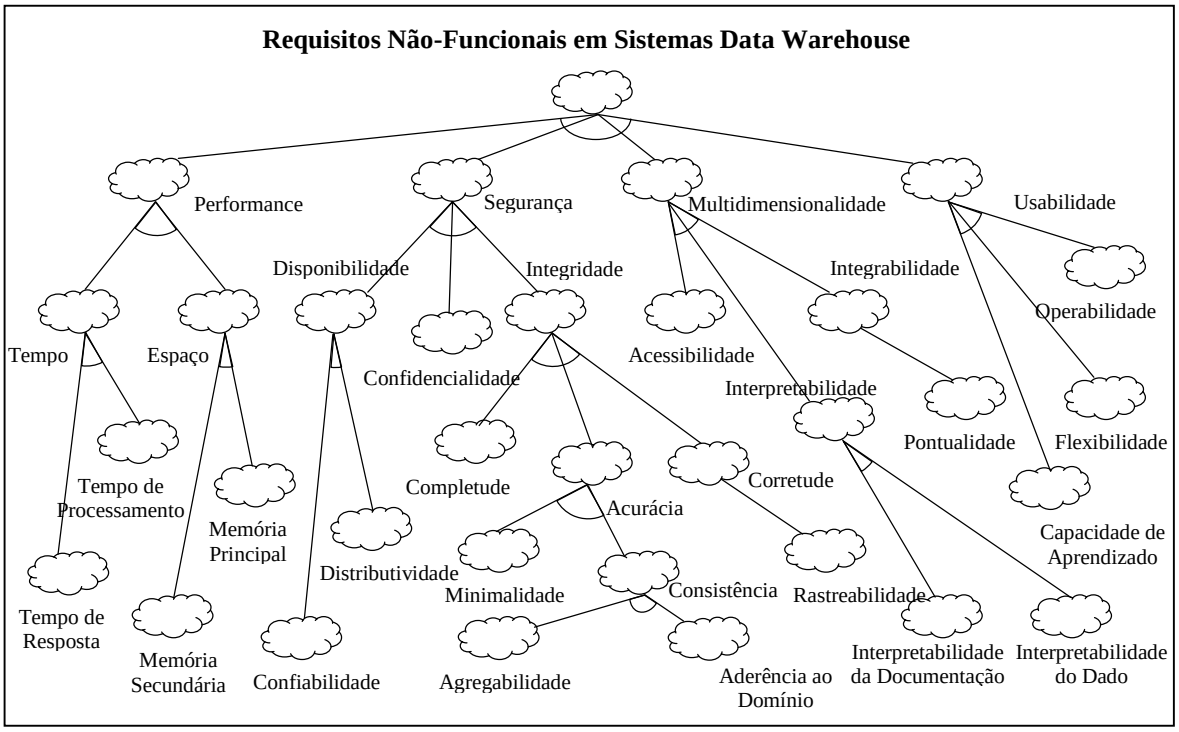


Figura 27. Árvore Hierárquica representando o Catálogo de Tipos RNF para Data Warehouse.

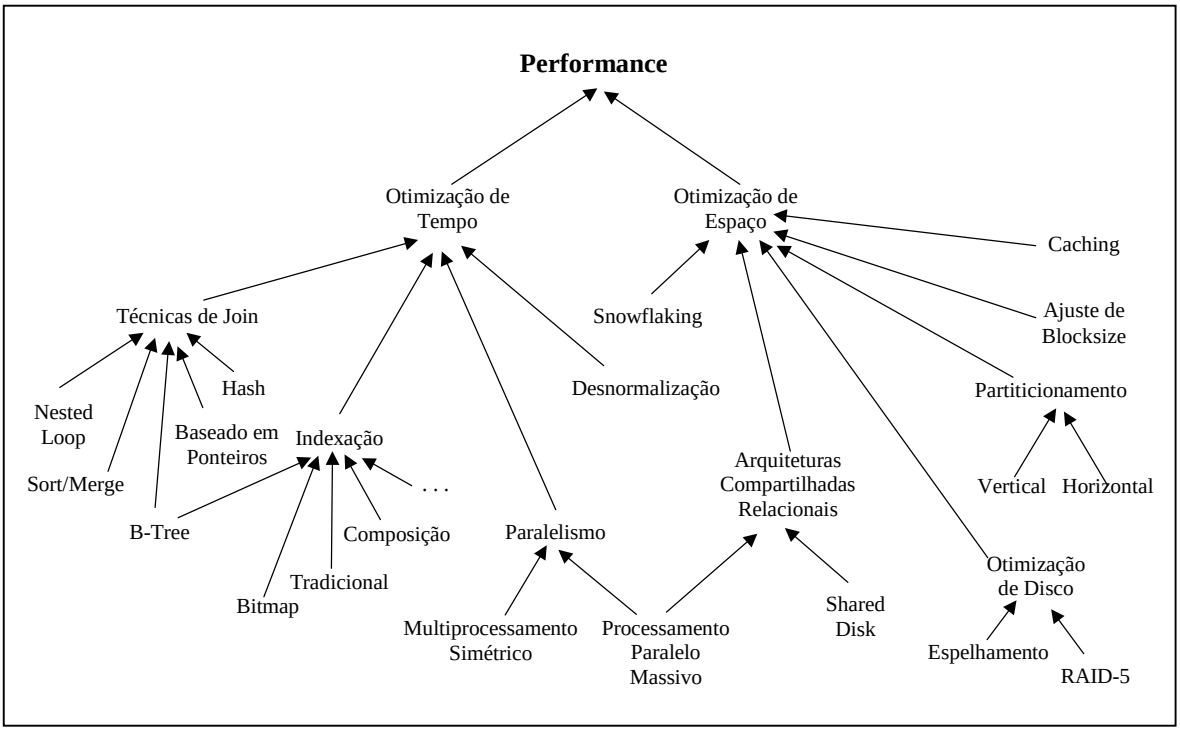


Figura 28. Métodos Operacionais para Performance.

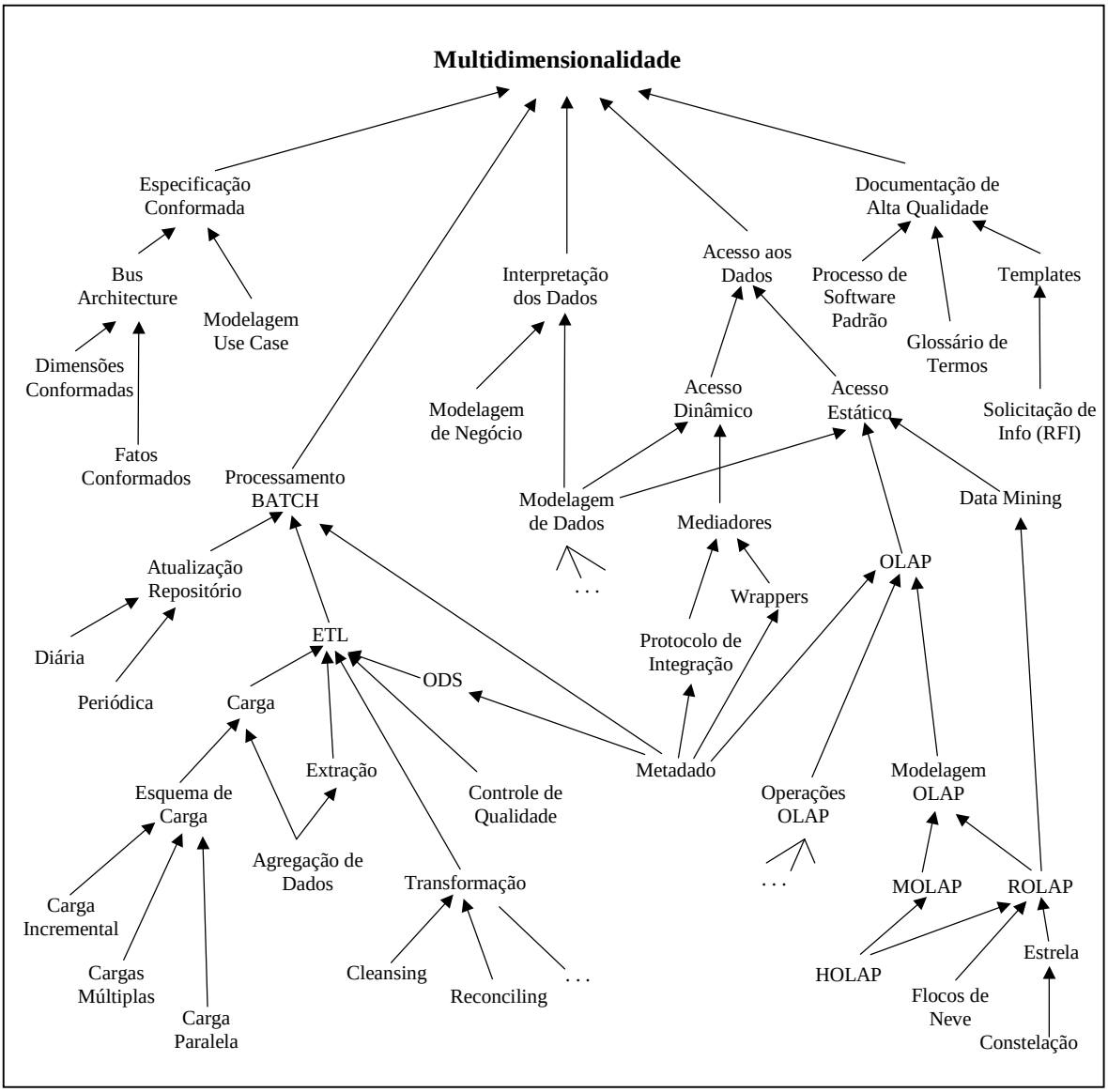


Figura 29. Métodos Operacionais para Multidimensionalidade.