PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL

FACULDADE DE INFORMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Uma abordagem de *Data Warehouse* para Análise de Processos de Desenvolvimento de Software

Taisa Carla Novello

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre, pelo programa de Pós Graduação em Ciência da Computação da Pontifícia Universidade Católica do Rio Grande do Sul.

Orientadora: Profª. Drª. Karin Becker

Porto Alegre

2006

Dados Internacionais de Catalogação na Publicação (CIP)

N939a Novello, Taisa Carla
Uma abordagem de data warehouse para análise de processos de desenvolvimento de software / Taisa Carla Novello. – Porto Alegre, 2006.
153 f.

Diss. (Mestrado) – Fac. de Informática, PUCRS
Orientador: Profa. Dra. Karin Becker

1. Engenharia de Software. 2. Informática. 3. Banco de Dados - Arquitetura. I. Título.

CDD 005.1

Ficha Catalográfica elaborada pelo
Setor de Tratamento da Informação da BC-PUCRS

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
Reconhecido pelo Parecer No. 930/98.C.N.E. Homologação Publicada no D.O.U. de 30/12/98.

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada "***Uma Abordagem de Data Warehouse para Análise de Processos de Desenvolvimento de Software***", apresentada por Taísa Carla Novello, como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Sistemas de Informação, aprovada em 02/03/2006 pela Comissão Examinadora:

Profa. Dra. Karin Becker – Orientadora — PPGCC/PUCRS

Prof. Dr. Duncan Dubugras Alcoba Ruiz – — PPGCC/PUCRS

Prof. Dr. Ricardo Melo Bastos – — PPGCC/PUCRS

Profa. Dra. Nina Krahe Edelweiss – — PPGC/UFRGS

Homologada em...../...../....., conforme Ata No. ..... pela Comissão Coordenadora.

Prof. Dr. Fernando Luís Dotti
Coordenador.

AV. IPIRANGA, 6681 - PRÉDIO 16/SALA 106
90619-900 - PORTO ALEGRE -RS -BRASIL
TELEFONE: 55 51 33203611 FAX 55 51 3320.3621

RESUMO

A busca pela qualidade sobre produtos de software se faz cada vez mais presente e necessária em organizações de software. Neste sentido, essas organizações buscam opções de como medir e analisar quantitativamente a qualidade de seus processos de desenvolvimento. No entanto, organizações trabalham com diferentes projetos que, por sua vez, utilizam-se de diversos processos e métricas. Partindo desta premissa, tais organizações devem buscar alternativas de como prover uma visão unificada através da centralização dos dados dos diferentes projetos e ainda disponibilizar, a seus usuários, análises quantitativas de seus Processos de Desenvolvimento de Software (PDS) através de um Programa de Métricas (PM). Para tal, os modelos de qualidade de software sugerem a construção de um repositório organizacional de métricas. Contudo, a construção de um repositório que atenda as características tanto de suporte ao armazenamento dos dados, como da disponibilização de análises aos usuários organizacionais não mostra-se uma tarefa trivial.

Perante esta realidade, este trabalho descreve sucintamente a arquitetura de um ambiente de Data Warehousing que provê suporte a adoção de um PM através do armazenamento de dados resultantes de diferentes PDS em uma base de dados unificada e centralizada. Este volume dedica-se a apresentação de dois componentes deste ambiente: o modelo analítico, base do *Data Warehouse* (DW), e o componente de apresentação no qual definem-se recursos analíticos que facilitam as análises realizadas pelos usuários organizacionais.

O desenvolvimento de um repositório deve considerar tanto as especificidades do PM adotado como as do próprio ambiente dos PDS. Quanto às métricas que compõem o PM, algumas representam dados não aditivos que podem comprometer as análises a serem realizadas. Já, quanto ao ambiente, especificidades dos PDS dificultam a definição de um único modelo que comporte características distintas. Além do armazenamento dos dados, a forma como estes serão disponibilizados também deve ser considerada, uma vez que usuários possuem características e necessidades de análise distintas. Por conseqüência, a complexidade de se desenvolver um modelo e prover recursos de análise neste contexto é muito alta.

Desta forma, o modelo analítico proposto visa armazenar métricas e dados resultantes dos PDS, considerando as necessidades de análises e tratando tanto as especificidades do PM adotado como também as do ambiente do PDS. A definição dos recursos analíticos propostos,

considera usuários com diferentes perfis, bem como suas particularidades. Estes recursos visam satisfazer as necessidades de análise destes perfis disponibilizando informações através de vários níveis de granularidade e disponibilizando mecanismos que forneçam maior semântica aos dados. Assim, este trabalho provê uma infraestrutura que suporta dados resultantes de diferentes PDS e análises quantitativas que consideram diferentes perfis de usuários embasadas em um PM.

Palavras-Chaves: Processo de Desenvolvimento de Software, Métricas, Modelo Analítico, Análise e Qualidade.

ABSTRACT

Software quality is important and necessary for organizations. They are interested on how to measure and quantitative analyze the processes quality. As organizations work with different projects, processes and metrics, they should looking for alternatives that provide a unified vision using the centralization of different projects data and provide quantitative analysis about Software Development Processes (SDP) through a Metric Program (MP). Thus, the software maturity models suggest the construction of a organizational repository of metrics. However, the construction of a repository that attends the data management and analysis functionality is not a trivial task.

This work describes briefly the environment architecture for Data Warehousing that provides support for MP adoption through data stored that means different processes in a unified database. This volume describes two components: the analytic model based on the Data Warehouse (DW), and the presentation component that uses analytic resources to facilitate the users analyses. The development of a repository should consider the particularities of MP and the environment of PDS. About the metrics that compose MP, some of them represent non addictive data what can made the analysis difficult about the environment, the PDS features made difficult the unified model definition that supports distinct characteristics. Besides the data storage, the manner who it is available also must be considered once the users has different analysis needs. Thus, the complexity of the model development and analysis resource availability is high.

Thus, the analytic model proposed intends to store metrics and SDP data, according to analysis requirements, considering the MP features particularity and the SDP environment. The definition of analytic resources considers users with different profiles. These resources attend the analysis requirements for each profile and shows the information through many granularity levels, including mechanisms that provide more semantic for the data. Thus, this work provides an infra-structure that supports different SDP and quantitative analyses for different profiles base on MP.

Keywords: Software Development Processes, Software Metrics, Analitycal Model, Quality.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

AQ	*Área de Qualidade*
BAM	*Business Activity Monitoring*
BDHP	*Base de Dados de Histórico de Projetos*
BI	*Business Intelligence*
BO	*Baseline Original*
BPC	*Business Process Cockpit*
BPI	*Business Process Intelligence*
BR	*Baseline Revisado*
CASE	Computer-Aided Software Engeneering
CMM	*Modelo de Maturidade da Capacidade*
CMMI	*Capability Maturity Model Integrated*
DBA	*DataBase Administrator*
DSA	*Data Staging Area*
DW	*Data Warehouse*
ES	*Engenharia de Software*
ETC	*Extração, Transformação e Carga*
KLOC	*Kilo Lines of Code*
NA	*Navegador de Agregados*
KPAs	*Key Process Areas*
OLAP	*On-line Analytical Processing*
OSSP	*Organization Software Standard Process*
PDS	*Processo de Desenvolvimento de Software*
PDSP	*Project Definition Standard Process*

PDW	*Process Data Warehouse*
PM	*Programa de Métricas*
PME	*Process Mining Engine*
PFR	*Project Formal Review*
PMR	*Project Management Review*
RMM	*Repositório Multidimensional de Medidas*
SEI	*Instituto de Engenharia de Software*
SGPN	*Sistema Gerenciador de Processos de Negócio*
SGBD	*Sistema Gerenciador de Banco de Dados*
SMR	*Senior Management Review*
SQA	*Software Quality Assurance*
SQL	*Structure Query Language*
SW	*Software*
PUCRS	Pontifícia Universidade Católica do Rio Grande do Sul

SUMÁRIO

1 INTRODUÇÃO

1.1 Contexto Geral

A cada dia, o mercado de software está mais exigente. Esta exigência se traduz em produtos de software de maior qualidade, desenvolvidos em menor tempo e com menor custo. A busca pela qualidade motiva as organizações de desenvolvimento de software a adotarem alternativas de analisar e monitorar seus processos. Segundo [IEE91] *apud* [PAU93], processo é uma seqüência de passos executados com um dado propósito. No contexto deste trabalho, o conceito de processo refere-se a um conjunto de atividades e resultados associados que levam ao desenvolvimento de um software.

Os modelos de qualidade de software (e.g. *Software-Capability Maturity Model* (SW-CMM), [PAU93], *Capability Maturity Model Integrated* (CMMI), [SEI96], *Software Process Improvement and Capability Determination* (SPICE) [EMA97], entre outros) vêm sendo adotados por fornecerem sistemáticas voltadas à obtenção da qualidade tanto nos processos de desenvolvimento quanto nos produtos resultantes. Estes modelos induzem as organizações a definirem processos para guiar suas atividades de desenvolvimento abordando diferentes aspectos (e.g. gestão de processos, gestão de projeto, engenharia, entre outros). O termo Processo de Desenvolvimento de Software (PDS) é adotado neste trabalho para designar esta visão organizacional do conjunto de processos adotados por uma organização de software.

A qualidade do PDS pode ser mensurada através de um Programa de Métricas (PM). Um PM é composto por um conjunto de métricas que reflete os requisitos de análise de uma determinada organização, tendo como meta principal a institucionalização da estratégia e dos objetivos organizacionais por todos os níveis em que possa ser implantado [WOU99] *apud* [HAY03]. Neste sentido, o principal desafio na definição de um PM é identificar a relação entre os objetivos da organização e os objetivos das métricas [OTL01] *apud* [HAY03], vislumbrando as diversas necessidades de informação de cada nível organizacional.

Métricas organizacionais podem ser logicamente apresentadas em Áreas de Qualidade (AQ), refletindo expectativas e/ou necessidades de análise relacionadas de uma organização. A definição de um PM organizacional que contemple todas as AQ almejadas pela organização é de suma importância uma vez que a análise de uma métrica isolada possui pouco valor, mas a análise de métricas dentro do contexto de um PM torna possível o acompanhamento de toda a história de um processo ou organização.

Modelos de qualidade de software, como por exemplo, SW-CMM e CMMI, definem como um dos requisitos necessários ao amadurecimento de organizações de desenvolvimento de software a adoção de um PM e então, a construção de um repositório organizacional. O repositório organizacional, requisito para CMM nível 3, tem como objetivo prover uma visão unificada da organização através do armazenamento e manipulação dos dados resultantes da execução de processos.

A construção de tal repositório organizacional em organizações de software não é uma tarefa trivial. Estas trabalham simultaneamente com diferentes projetos que possuem particularidades quanto aos processos e ferramentas adotadas, bem como a forma como geram, armazenam e controlam seus dados, em particular aqueles necessários ao cálculo das métricas. Também deve se considerar a forma como a informação armazenada será disponibilizada à organização.

A apresentação dos dados à organização é de suma importância uma vez que esta pode determinar a usabilidade ou não do repositório organizacional desenvolvido. Neste sentido, através do repositório organizacional permite-se disponizar uma visão organizacional unificada capaz de auxiliar ao usuário na análise de projetos já desenvolvidos na organização, isto é, dados históricos desta. Assim, a apresentação da informação deve considerar os diferentes tipos de usuários organizacionais, suas distintas necessidades de informações, bem como suas limitações e familiaridade com a tecnologia envolvida.

Não existe uma infraestrutura de apoio aos diferentes PM organizacionais considerando todos estes pontos de variabilidade, tanto de armazenamento como de disponibilização de informação. Algumas propostas específicas podem ser encontradas em [OLI99, SUB99, PAL03, CAS04, GRI04, SAY02 e GOL04].

Assim, considerando a ausência de uma proposta que contemple os diversos aspectos que interferem na qualidade do PDS, a questão de pesquisa que norteou este trabalho foi: Como fornecer uma infraestrutura de apoio à análise e mensuração de processos de desenvolvimento de software que atenda a requisitos do SW-CMM nível 3?

1.2 Objetivo do Trabalho

Este trabalho vem ao encontro das necessidades reais de diversas organizações de software que buscam prover qualidade ao PDS através da adoção de modelos de qualidade e que confrontam-se com alguns requisitos de difícil desenvolvimento. Seu principal objetivo é auxiliar a estas organizações propondo um repositório organizacional para armazenamento de dados oriundos do PDS e recursos analíticos para análise dos dados do repositório.

O trabalho apóia-se em um estudo de caso realizado em uma operação de software de uma grande organização avaliada atualmente SW-CMM nível 3, localizada no Tecnopuc, PUCRS.

Complementando o objetivo geral, abaixo lista-se os objetivos específicos que guiam este trabalho:

- Propor um modelo analítico que permita o armazenamento de dados gerados pelos PDS;
- Prover mecanismos que disponibilizem maior semântica aos dados disponibilizados;
- Propor recursos que simplifiquem a análise de dados provenientes do Data Warehouse, DW considerando os diferentes tipos de usuários existentes e suas respectivas necessidades de análise sobre o PDS;
- Desenvolver um protótipo da camada de apresentação com seus recursos analíticos que disponibilize a análise do PDS através dos dados provenientes do DW.
- Escrever um relato de experiência quanto aos benefícios da adoção de um DW e de recursos analíticos.

1.3 Método e Estrutura de Pesquisa

O método de pesquisa que firmou o desenvolvimento desta dissertação foi o estudo de caso, caracterizado como de caso único [YIN01].

A Figura 1 representa as principais atividades desenvolvidas na condução desta pesquisa. Inicialmente estudou-se o ambiente em que o estudo de caso seria desenvolvido a fim de constatar quais as principais dificuldades organizacionais encontradas na adoção de um programa de métricas e de uma infra-estrutura de suporte.

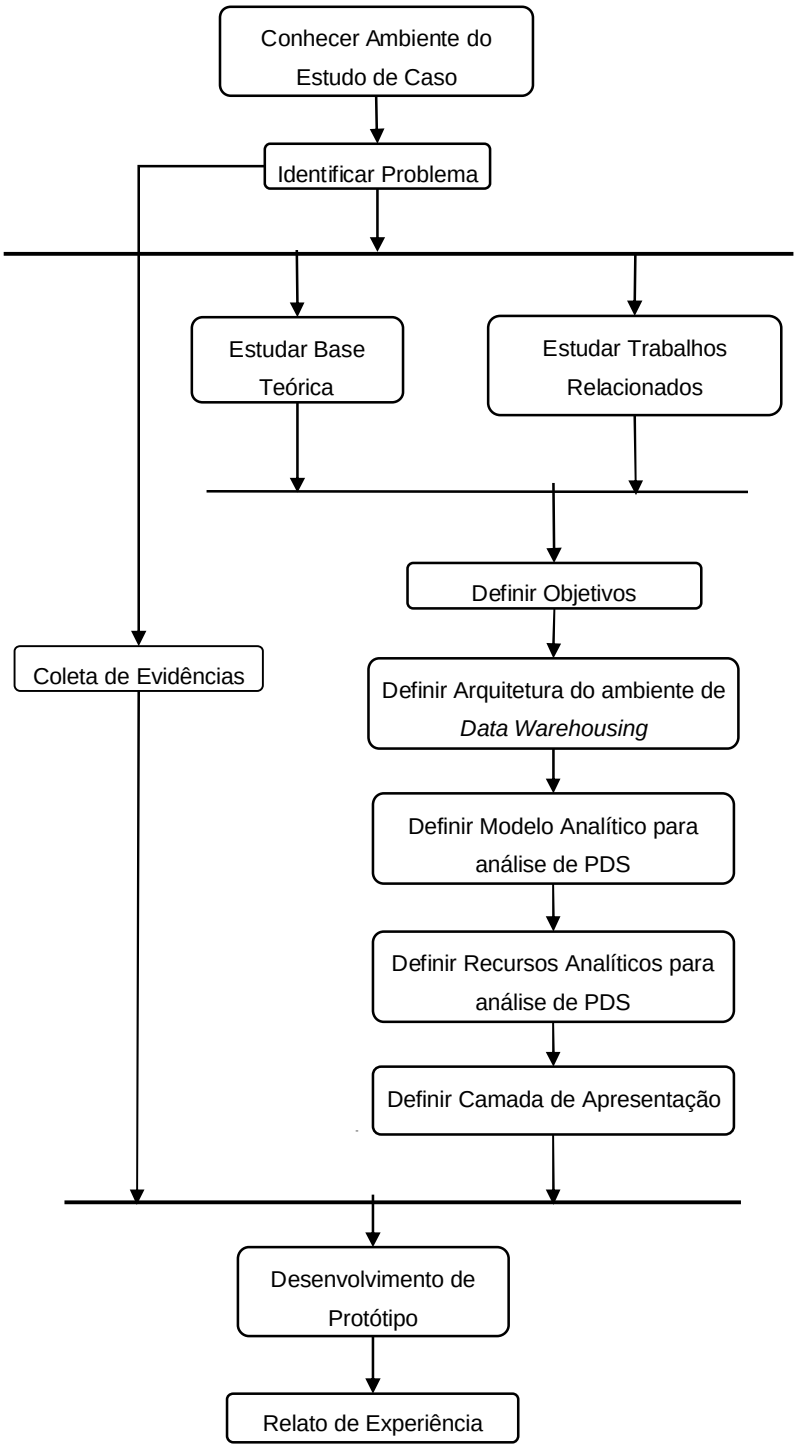

Figura 1: Atividades desenvolvidas na condução da pesquisa.

Após esta identificação, buscou-se na literatura propostas de repositórios de dados organizacionais a fim de se ter um entendimento e conhecimento geral do estado da arte. Posteriormente, focou-se em propostas de repositórios multidimensionais e de recursos para uma camada de apresentação que provenham o acompanhamento e análise quantitativa de dados. A restrição de escopo deve-se ao fato que tais propostas vão ao encontro da solução do problema motivador desta pesquisa.

Paralelamente à pesquisa literária, iniciou-se a organização e coleta das evidências necessárias para o desenvolvimento da pesquisa. A coleta de evidências estendeu-se durante todo o período de definições da pesquisa, finalizando-se somente quando iniciado o desenvolvimento do protótipo.

O estudo de diversas propostas, o embasamento teórico e as evidências adquiridas forneceram subsídios para a definição de uma arquitetura para um ambiente de *Data Warehousing*. Posteriormente, definiu-se o modelo analítico para análise de dados resultantes de PDS e, após definiu-se os recursos analíticos para análise dos dados oriundos do DW. Com a definição dos recursos analíticos finalizada, deu-se então a definição de uma camada de apresentação para suportá-los.

O passo seguinte foi o desenvolvimento do protótipo que implementa o modelo analítico do DW proposto, como também os recursos analíticos previstos para a camada de apresentação. Após, desenvolveu-se um relato de experiência real que descreve os benefícios da utilização de ambiente de *Data Warehousing* em produção em uma organização.

1.4 Estrutura do Trabalho

Este trabalho está dividido em 10 capítulos. O Capítulo 2 discute questões relativas à qualidade em PDSs, apresentando inicialmente uma visão geral do PDS e como os modelos de qualidade de software e a utilização de métricas podem auxiliar organizações a prover qualidade ao PDS. O Capítulo 3 discorre sobre principais conceitos de DW, sua arquitetura, métodos de modelagem do modelo analítico, recursos OLAP e analíticos para uma camada de apresentação. O Capítulo 4 apresenta o ambiente de estudo caso que motiva esta pesquisa dando maior ênfase aos aspectos apresentados no Capítulo 2. O Capítulo 5 apresenta trabalhos relacionados ao objetivo desta pesquisa.

Os Capítulos 6, 7 e 8 têm por objetivo apresentar ao leitor o trabalho de pesquisa desenvolvido. Para tal, o Capítulo 6 contextualiza o leitor quanto à arquitetura na qual está

inserida este trabalho. O Capítulo 7 apresenta detalhadamente o modelo analítico proposto, salientando aspectos de sua modelagem e análises providas. Já o Capítulo 8 apresenta todos os itens considerados na definição de recursos analíticos para uma camada de apresentação que visa atender a todas as perspectivas de análise organizacional. Neste, discorre-se sobre possíveis perfis de usuários, definição de recursos analíticos, a camada de apresentação propriamente dita, como também a definição do componente de análise composto da especificação de um algoritmo redirecionador de consultas ao DW desenvolvido.

No Capítulo 9 apresenta-se um relato de experiência quanta à adoção de um ambiente de *Data Warehousing*. O Capítulo 10 discorre sobre as conclusões, limitações e trabalhos futuros. Posteriormente, encontram-se as referências bibliográficas pesquisadas e anexos.

2 PROCESSO DE DESENVOLVIMENTO DE SOFTWARE E QUALIDADE

Este capítulo discute questões relativas à qualidade em PDSs apresentando inicialmente uma visão geral do PDS e após como modelos de qualidade de software e a utilização de métricas podem auxiliar as organizações a prover qualidade ao PDS.

2.1 Processo de Desenvolvimento de Software e Qualidade

Projetos de desenvolvimento de software podem possuir diferentes objetivos, tamanhos e formas de execução. Este fato tem dificultado o aparecimento de uma metodologia ou abordagem que defina como desenvolver um projeto de software com sucesso. Neste sentido, a Engenharia de Software (ES) nas últimas duas décadas tem centrado seus estudos em como prover qualidade ao processo para desenvolvimento de software.

Muitos são os conceitos encontrados na literatura para ES. Em [PRE01] apresenta-se a ES como o conjunto de quatro elementos fundamentais: métodos, ferramentas, processos e qualidade. Estes elementos são apresentados através de camadas, como mostrado na Figura 2, extraída de [PRE01] . A qualidade, representada como sustentação aos demais elementos deve possibilitar o desenvolvimento de um software de maior qualidade.

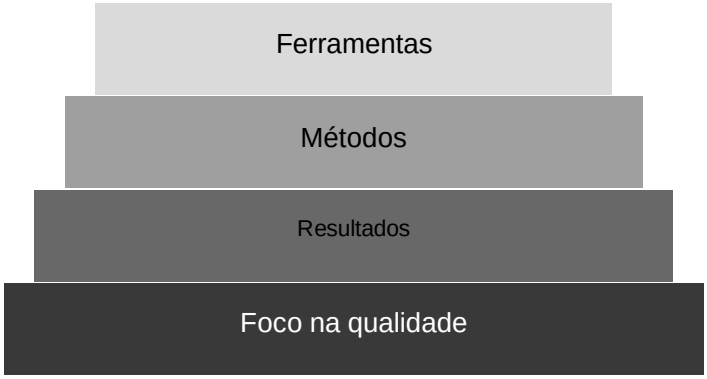

Figura 2 : As camadas da ES.

Os métodos em ES proporcionam os detalhes de como desenvolver um software, envolvendo um amplo conjunto de tarefas que incluem: planejamento e estimativa de projeto, análise de requisitos de software e de sistemas, projeto da estrutura de dados, arquitetura de programa e algoritmo de processamento, codificação, teste, manutenção, entre outros [PRE01].

As ferramentas da ES proporcionam o apoio automatizado ou semi-automatizado aos métodos. As ferramentas que apóiam os diversos métodos podem ser integradas de forma que a informação criada por uma ferramenta possa ser utilizada por outra, sendo estabelecido um sistema de suporte ao desenvolvimento de software chamado engenharia de software auxiliada por computador (CASE – *Computer-Aided Software Engineering*).

Os processos constituem o elo de ligação entre as ferramentas e os métodos, além de possibilitar o desenvolvimento racional do software. Os processos definem um conjunto de áreas chave (*Key Process Areas* - KPAs) [PAU93] *apud* [PRE01], que estabelecem a efetiva entrega do produto de software. As KPAs são a base para o gerenciamento dos projetos de software e estabelecem o contexto e a seqüência em que os métodos serão aplicados, produtos serão entregues, os controles que ajudam a assegurar qualidade e a coordenar as mudanças, e os marcos de referência que possibilitam aos gerentes de software avaliar o progresso do desenvolvimento [PRE01].

A resolução de problemas que circundam o PDS leva as organizações a buscar estratégias de desenvolvimento de software que considerem seus processos, métodos e ferramentas. Estas estratégias, chamadas modelos de processo de software ou ainda paradigmas da engenharia de software, são modelos de abstrações úteis que podem ser utilizadas para explicar diferentes abordagens do desenvolvimento de software [SOM04]. O modelo de processo deve ser escolhido tendo-se como base a natureza do projeto e da aplicação, os métodos e as ferramentas a serem utilizados e os produtos a serem entregues [PRE01]. Diferentes modelos de processos vêm sendo propostos, cada um considerando diferentes perspectivas e necessidades do processo de desenvolvimento de software. Neste sentido, abaixo apresenta-se uma breve descrição de alguns modelos de processo de software. Mais detalhes poderão ser encontrados em [PRE01, SOM04].

§ *Modelo Cascata ou Seqüencial linear*: o desenvolvimento ocorre linearmente, desde a análise dos requisitos, passando pela construção, teste de código e unidade, teste de subsistema e teste do sistema.

§ *Modelo de Prototipação*: o cliente e o desenvolvedor reúnem-se e definem objetivos globais para um protótipo de projeto rápido. Após, um processo de iteração ocorre a fim de satisfazer as necessidades do cliente e capacitar o desenvolvedor a compreender o que ainda precisa ser realizado.

- *Modelo Evolutivo ou Iterativo*: caracteriza-se por ser iterativo e permitir a engenheiros de software desenvolvem gradativamente versões de software [PRE01]. Difere-se da prototipação por ter como objetivo entregar um produto operacional a cada iteração. O modelo iterativo pode ser usado para reduzir riscos, uma vez que, a cada iteração, os riscos podem ser analisados. Este modelo apóia o desenvolvimento das abordagens de desenvolvimento incremental e espiral [SOM04].
- *Modelo Desenvolvimento Formal de Sistemas*: tem como base a transformação matemática formal de uma especificação de sistemas em um programa executável [SOM04].
- *Modelo baseado em Componentes ou Orientado a Reuso*: este modelo de processo é orientado a reuso com uma ampla base de componentes de software reutilizáveis, que podem ser acessados, e com alguma infra-estrutura de integração para esses componentes. Possui como vantagem a redução da quantidade de software a ser desenvolvida, de custo e riscos e como desvantagem a perda de algum controle sobre a evolução do sistema, uma vez que, novas versões dos componentes reutilizáveis podem não estar sob o controle da organização que utiliza estes componentes.

Finalmente, o processo de software pode ser caracterizado conforme mostra a Figura 3, extraída de [PRE01]. Um *Framework de Processos Comuns* constitui-se de um pequeno número de atividades aplicáveis a todos os projetos de software, considerando seu tamanho e complexidade. Estas atividades constituem o *Framework de Atividades.* O *Conjunto de Tarefas* permite agrupar as tarefas de engenharia de software, marcos de projetos, produtos a serem entregues e tarefas para garantir a qualidade do software *(Software Quality Assurance-SQA)*. O *Conjunto de Tarefas* permite a geração do *Framework de Atividades*. Já as *Atividades de Apoio* buscam garantir a qualidade do processo, a gerência de configuração de software e medições. As atividades de apoio são independentes de qualquer atividade do *Framework de Atividades* e ocorrem de forma independente do processo.

2.2 Modelos de Qualidade de Software

Há alguns anos, expressões como “maturidade de processo” e/ou “processos maduros” vêm se tornando mais comuns em organizações de software. Na segunda metade da década de oitenta, o *Software Engineering Institute* (SEI) iniciou o desenvolvimento de uma estrutura de

maturidade com o objetivo de ajudar organizações a melhorar seus processos de software. A partir de tal iniciativa surgiram os atuais modelos de qualidade.

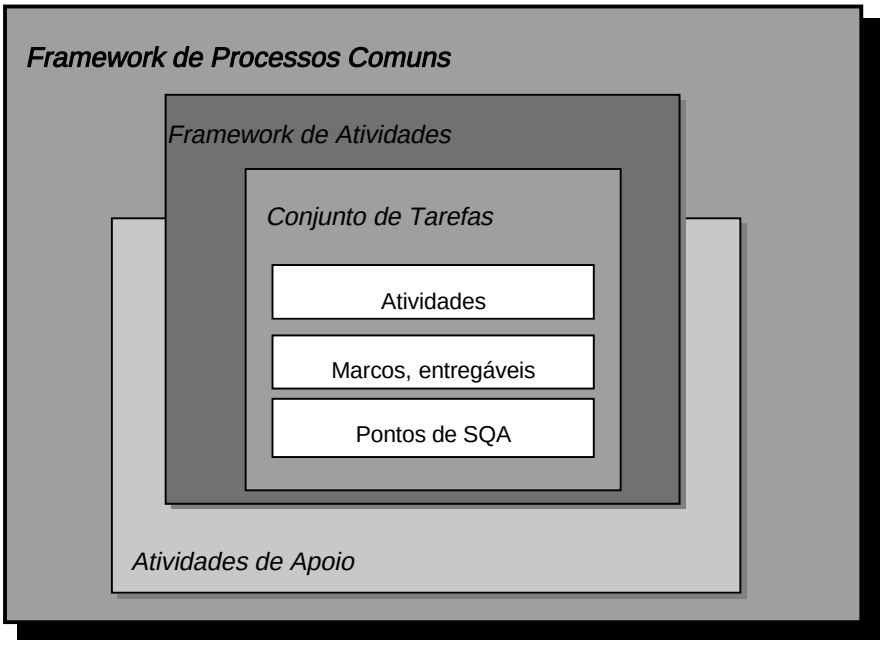

Figura 3: O processo de Software.

Os modelos de qualidade de software têm como objetivo auxiliar as organizações a melhorar seus processos através de guias de como obter o controle destes e evoluir em direção a uma cultura de engenharia de software e excelência de gestão.

Dois modelos que têm se destacado bastante na comunidade de ES são SW-CMM [PAU93] e CMMI [SEI96]. Estes modelos auxiliam organizações de software no processo de seleção das estratégias de melhoria, determinando a maturidade atual do PDS e identificando questões críticas para sua melhoria. Estes acreditam que focando em um conjunto limitado de atividades e trabalhando agressivamente para concluí-las com êxito, a organização pode melhorar o processo.

Nos modelos SW-CMM e CMMI a melhoria contínua do processo é baseada em pequenas etapas evolutivas, ao invés de fundamentar-se em inovações revolucionárias [PAU93]. Estas etapas evolutivas estão estruturadas em cinco níveis de maturidade que estabelecem fundamentos sucessivos para a melhoria contínua do processo. Cada nível de maturidade especifica certas características do processo. Seguindo estes modelos as organizações encontram um guia de como transformar um processo imaturo e "*ad hoc*" em um processo de maior maturidade.

As próximas seções apresentam uma breve descrição destes dois modelos de qualidade enfatizando-se em questões relacionadas a medições e métricas.

2.2.1 SW-CMM

A Figura 4, extraída de [PAU93], representa cada etapa evolutiva estabelecida pelo modelo SW-CMM**.**

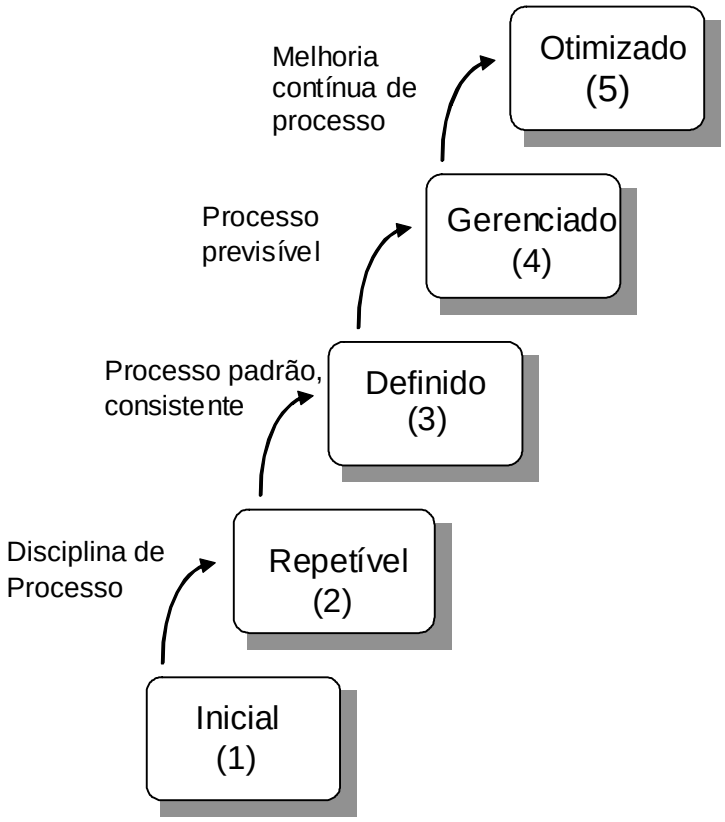

Figura 4: Níveis de maturidade do modelo SW-CMM.

No modelo SW-CMM o Nível 2 indica que a organização passou de um processo "*ad-hoc*", Nível 1, para um processo disciplinado através de controles de gestão de projeto. Neste nível a organização começa a definir e a coletar algumas métricas alinhadas com os objetivos da organização, e de suas necessidades de informação. A definição das métricas terá impacto sobre as análises que serão disponibilizadas e a conquista dos níveis de maturidade posteriores [PAU93].

No nível 3, o PDS organizacional passa a ser padronizado para a organização como um todo. Esta padronização dos PDS é um dos aspectos essenciais na busca da qualidade de software.

Neste sentido, a partir do nível 3 o PDS de toda a organização é regido por um conjunto de processos organizacionais (*Organization Standard Software Process* - OSSP) e um conjunto de processos de projetos (*Project Defined Software Process* - PDSP). O OSSP contém processos fundamentais e obrigatórios e deve ser seguido por todos os projetos da organização, enquanto que no PDSP cada projeto tem a liberdade de realizar adaptações do OSSP a fim de ajustá-lo às características específicas de seu projeto.

Também, o Nível 3 cita a necessidade de um repositório que estabeleça e mantenha os processos organizacionais juntamente com suas métricas, e que possibilite algumas análises sobre estas. O repositório deve armazenar produtos e métricas dos processos previamente definidos pela organização e, ainda, trabalhar com a informação de forma que esta possa ser utilizada para avaliação quantitativa dos processos. Através das métricas estabelecidas no Nível 2 e do repositório do Nível 3, torna-se possível estimar e planejar atividades de projetos, bem como ter também uma visão unificada e quantitativa sobre a qualidade dos projetos organizacionais.

O Nível 4 visa estabelecer e manter o entendimento quantitativo dos PDS organizacionais provendo suporte à qualidade através do controle do desempenho dos processos e dos *baselines* (estabelecimento de linhas básicas [PRE01]), disponibilizando ainda modelos para gerenciar quantitativamente os projetos da organização.

O Nível 5 tem como propósito identificar inovações que provenham maior qualidade ao desenvolvimento dos processos e, através dessas inovações, aperfeiçoar as análises quantitativas sobre os mesmos. Neste sentido, os Níveis 4 e 5 visam o aperfeiçoamento de estimativas e otimização, tendo por base o repositório organizacional.

Cada um dos níveis de qualidade do modelo SW-CMM, com exceção do Nível 1, são decompostos em várias áreas chave de processo. As áreas chave de processo indicam as áreas nas quais uma organização deve focar seus esforços para a melhoria de seu PDS. Exemplos de áreas chave são: planejamento, acompanhamento e supervisão, engenharia de produto de software, entre outras.

As áreas chave de processo são organizadas por características comuns. As características comuns são atributos que determinam se a implementação e a institucionalização de uma área chave de processo serão eficazes, repetíveis e duradouras [PAU93]. Dentre as cinco características comuns, salientamos neste trabalho a Medição e Análise.

A característica comum de Medição e Análise descreve a necessidade de medir as atividades executadas e analisar estas medições [PAU93]. Segundo [PAU93] exemplos de métricas são tratadas no modelo SW-CMM como informação adicional, uma vez que a variabilidade existente em ambientes de projeto e/ou organizações pode conduzir a necessidades diferentes.

2.2.2 CMMI

O modelo CMMI possui uma diferença básica em sua implantação em relação ao modelo SW-CMM. Ao implantar o modelo CMMI, a organização deve escolher a forma de representação do modelo que lhe convém, sendo que as duas representações disponíveis são:

- Contínua: perspectiva da capacidade das áreas de processo, medindo resultados de cada área individualmente;
- Estágios: perspectiva de maturidade da organização, enfatizando conjuntos de áreas de processo que definem estágios comprovados de maturidade de processos. Provê a execução de etapas evolutivas, semelhante ao modelo SW-CMM.

Seguindo a representação em estágios, o CMMI pode ser decomposto em cincos níveis de maturidade, apresentados na Figura 5, extraída de [CMM02].

A estrutura do modelo CMMI muito se assemelha à estrutura do SW-CMM, o que é explicado pelo fato do modelo CMMI ter sido elaborado a partir de uma revisão do modelo SW-CMM. Neste sentido, os principais componentes do modelo CMMI são seus objetivos específicos e genéricos que guiam a organização quanto à melhoria de processo, suas práticas específicas e genéricas que explicitam formas de como a organização alcançará seus objetivos específicos e genéricos e as áreas de processos que constituem-se de conjuntos de práticas relacionadas que quando executadas coletivamente satisfazem os objetivos considerados importantes para a melhoria significativa de uma área.

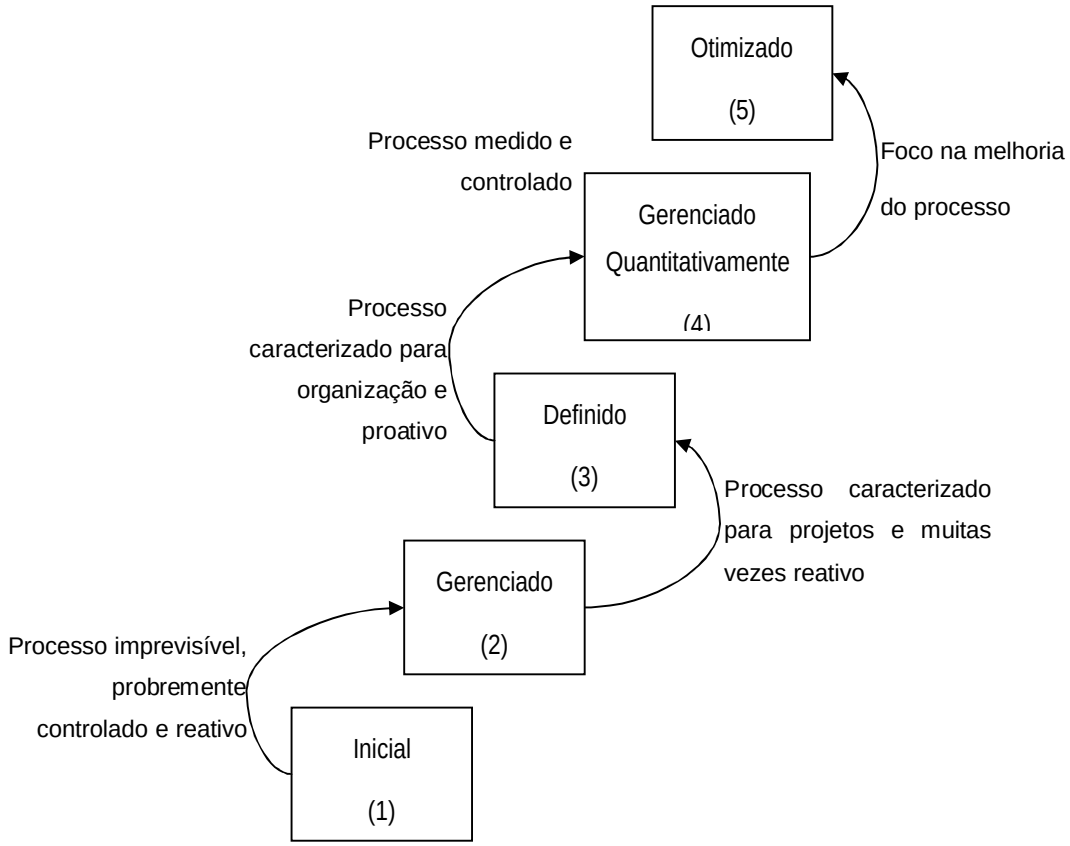

Figura 5: Os cincos níveis de maturidade no modelo CMMI.

Todas as áreas de processos são comuns tanto à representação contínua como por estágio, sendo que na representação contínua as áreas de processos são organizadas em categorias de ao invés de níveis. A Tabela 1 apresenta as áreas de processos organizadas por categorias. A Tabela 2, extraída de [CMM02], apresenta as áreas de processo organizadas por nível de maturidade.

A área de processo Medição e Análise apóia todas as áreas de processo provendo práticas específicas que guiam os projetos e a organização na definição das métricas e dos resultados que estas poderão gerar para apoiar a tomada de decisões e ações corretivas do processo em tempo apropriado [SEI96]. A área de processo de Medição e Análise, tanto no SW-CMM ou no SMMI possui os seguintes objetivos:

- Alinhar objetivos de Medição e a Análise a fim de identificar as informações necessárias aos objetivos;
- Fornecer os resultados obtidos para que sejam utilizados na tomada de decisões e ações corretivas apropriadas.

Tabela 1: Áreas de Processo por categoria no Modelo CMMI.

Categorias	Áreas de Processos
Gerência de Processo	**Foco no Processo da Organização** **Definição do Processo da Organização** **Treinamento Organizacional** **Desempenho do Processo Organizacional** **Inovação e Desenvolvimento Organizacional**
Gerência de Projeto	**Planejamento do Projeto** **Controle e Monitoração de Projetos** **Gerência de Acordo com Fornecedores** **Gerência Integrada de Projeto** **Gerência de Riscos** **Gerência Integrada de Fornecedores** **Gerência Quantitativa de Projeto**
Engenharia	**Gerência de Requisitos** **Desenvolvimento de Requisitos** **Integração de Produto** **Solução Técnica** **Verificação** **Validação**
Suporte	**Gerência de Configuração** **Garantia de Qualidade do Processo e Produto** **Medição e Análise** **Análise e Resolução de Causas** **Desenvolvimento Organizacional** **Análise e Resolução de Causas**

Tabela 2: Áreas de Processo por Nível no Modelo CMMI.

Nível de Maturidade	Áreas de Processos
5- Otimizado	**Inovação e Desenvolvimento Organizacional** **Análise e Resolução de Causas**
4- Gerenciado Quantitativamente	**Desempenho do Processo Organizacional** **Gerência Quantitativa de Projeto**
3- Definido	**Desenvolvimento de Requisitos** **Solução Técnica** **Integração de Produto** **Verificação** **Validação** **Gerência de Riscos** **Gerência Integrada de Fornecedores** **Gerência Integrada de Projeto** **Definição do Processo da Organização** **Foco no Processo da Organização** **Treinamento Organizacional**
2- Gerenciado	**Controle e Monitoração de Projetos** **Garantia de Qualidade do Processo e Produto** **Gerência de Configuração** **Gerência de Acordo com Fornecedores** **Gerência de Requisitos** **Planejamento do Projeto** **Medição e Análise**
1- Inicial	**Não se aplica**

2.2.3 Considerações sobre o SW-CMM e CMMI

Quanto maior for o nível de maturidade de uma organização (SW-CMM ou CMMI), mais instrumentalizados com medidas consistentes e bem definidos são seus processos [PAU93]. [BRO04] analisa organizações de software e sugere diversas áreas onde organizações poderiam se beneficiar de medições e análise. Através desta análise, [BRO04] aponta que 22% das fraquezas e oportunidades identificadas no gerenciamento de processos estão relacionadas a assegurar a qualidade da medição e análise.

Neste sentido, um dos principais objetivos dos modelos de qualidade é prover às organizações uma visão compartilhada das métricas de desempenho de seus processos de desenvolvimento de software. Para isto, estes estabelecem práticas quanto à definição, coleta, armazenamento, análise e usabilidade de medições, a serem adotadas em cada nível de maturidade conquistado. O grau de medição das organizações tende a aumentar à medida que a maturidade vai sendo conquistada [FEN97].

2.3 Métricas

Uma proposta disciplinada de medição e análise dos dados pode ser crucial para o sucesso de um software ou sistemas de uma organização [GOL99] e [GOL03] *apud* [BRO04]. A qualidade de software pode ser adquirida através da avaliação quantitativa de dados gerados a partir do processo de desenvolvimento. A avaliação quantitativa deve ser claramente definida para que não fique somente na intuição. Neste sentido, deve-se definir um conjunto apropriado de métricas de software. A utilização de métricas visa a realização de avaliações ao longo do PDS, permitindo verificar se o nível de qualidade exigido está sendo satisfeito. As métricas de software reduzem a subjetividade da avaliação provendo o controle da qualidade do software através de uma base quantitativa para tomada de decisões.

O padrão IEEE 1061 [IEE98] define termos relacionados a métricas de qualidade software. Em [KAN04], é realizada uma discussão crítica sobre a dificuldade de estabelecer um programa de métricas e sobre a validade de métricas estabelecidas na área de engenharia de software. Também é proposto um arcabouço para avaliar e compreender métricas. Assim, na literatura relacionada a este assunto, não há um consenso quanto aos termos relacionados. Abaixo apresenta-se alguns termos que utilizados neste trabalho.

- Atributo: propriedade física ou abstrata mensurável de uma entidade (e.g. tamanho, defeito, esforço);
- Medição: ato ou processo de atribuir um número ou categoria a uma entidade;
- Métrica ou Métrica de qualidade de Software: função cujas entradas são dados de software e a saída é um único valor numérico, interpretado como o grau de qualidade de determinado atributo do software (e.g. variação de tamanho, variação de esforço, densidade de defeitos);
- Valor ou resultado de métrica: o valor ou elemento de saída de uma métrica, exemplo: variação de tamanho de um produto é igual a 5%;
- Indicador: representa *status* da qualidade decorrente do resultado de uma métrica. Pode ser utilizado para monitorar a qualidade através da análise de tendências em um processo de desenvolvimento. Segundo [BUG03], indicadores são fáceis de compreender, mostram tendência de desempenho (“bom” ou “ruim”) e proporcionam que ações possam ser tomadas rapidamente (e.g. a métrica densidade de defeitos, poderia possuir os seguintes indicadores: baixa (menor que 0,3 defeitos/KLOC), média (de 0,3 a 0,7 defeitos/KLOC) e alta (maior que 0,7 defeitos/KLOC));
- Fator de qualidade: valor ou palavra associada a um atributo de software que representa sua qualidade. (e.g. variação de tamanho <=10%, densidade de defeitos <=0.3 defeitos/KLOC);
- Valor crítico: valor de uma métrica validada, usado para identificar um nível de qualidade como inaceitável, exemplo: variação de tamanho de um produto é igual a 20%;

De acordo com o Padrão IEEE 1061 [IEE98], métricas podem ser classificadas como base e derivada. Uma métrica base não depende da medida de nenhum outro atributo, e são também denominadas métricas fundamentais. Já métricas derivadas normalmente representam funções calculadas em função de mais de uma métrica base. Alguns exemplos de métricas derivadas são *Produtividade* (tamanho do código / esforço), *Densidade de Defeitos* (número de defeitos / tamanho), entre outras. Ao longo deste trabalho utilizaremos estes termos, contudo, em casos onde o tipo da métrica não interfere sobre o assunto tratado referencia-se apenas métricas ou métrica de software.

A definição de um programa de métricas pode fornecer informações tanto sobre o PDS quanto sobre o produto gerado. Do ponto de vista do processo, é possível monitorar as fases e atividades que compõem o ciclo de vida de desenvolvimento do software, atribuindo métricas sobre a sua realização. As atividades, por sua vez, produzem saídas tangíveis, tais como código fonte e/ou documentação de projeto que também podem ser monitoradas através de medições.

As métricas podem ser agrupadas em Áreas de Qualidade (AQ), as quais representam as diferentes expectativas e/ou necessidades de análise da organização. Um exemplo de AQ poderia ser *Qualidade*, uma vez que qualidade pode ser o agrupamento de diversas características. A noção de qualidade é geralmente relacionada a ausência de defeitos, conforme afirma Fenton e Pfleeger em [FEN97] “Métricas de qualidade: defeitos”. Assim, a AQ *Qualidade* poderia conter as métricas: densidade de defeitos, eficiência de revisão, entre outras. Cabe salientar que uma AQ pode representar diferentes atributos, como por exemplo, a métrica densidade de defeitos que possui em sua função os atributos tamanho e defeitos. Outros exemplos de AQs: tempo contendo variação de cronograma e duração, esforço contendo variação de esforço no *baseline* original e revisado, entre outras.

A adoção de um programa de métricas inclui a definição e/ou seleção de métricas que podem gerar informações úteis à organização. Para tal, deve-se levantar e analisar quais as metas organizacionais e o que se deseja descobrir e/ou mostrar através das métricas definidas e/ou selecionadas. Por outro lado, o programa de métricas estabelecido define diretamente os dados que deverão ser coletados do ambiente transacional. Neste sentido, a organização deve preocupar-se com a forma de como as diferentes estruturas e/ou ciclos de vida de PDS existentes na organização armazenam a informação requerida pelo PM e como esta será coletada.

Um dos aspectos mais importantes na utilização de um programa de métricas é a forma como seus resultados são avaliados e analisados. Primeiramente deve-se realizar a interpretação de seus resultados investigando as diferenças entre o fator de qualidade e o respectivo resultado obtido. A validação das métricas tem como propósito verificar se o programa de métricas pode predizer específicos fatores de qualidade. Os resultados obtidos das métricas indicarão se o fator de qualidade sobre um processo foi ou será alcançado em um futuro próximo. A validação das métricas não se dá de forma unificada, deve-se respeitar a relação entre cada métrica com seu fator de qualidade específico para uma determinada aplicação [IEE98].

A implantação de um programa de métricas não elimina a necessidade do julgamento humano em avaliações de software [IEE98]. Normalmente, os resultados obtidos apóiam pessoas que disponibilizam pouco tempo para a realização de análises estatísticas. Assim, a apresentação das métricas também é considerada como um dos fatores cruciais para o sucesso da implantação de um programa de métricas.

2.4 Considerações

Prover qualidade ao PDS é um desafio cada vez mais perseguido por organizações. Ao longo da revisão teórica realizada neste capítulo identificaram-se diversos aspectos que influenciam na qualidade de um produto de software.

Inicialmente, apresentou-se as características do PDS sua estrutura e alguns modelos de processos previstos pela ES. Após discorreu-se sobre dois modelos de qualidade (SW-CMM e CMMI) enfatizando sua estrutura e quesitos necessários a cada nível de maturidade. Nestes, atenção especial foi despendida à Medição e Análise, por ser foco deste trabalho.

Sendo um PM parte essencial da Medição e Análise a última seção deste capítulo apresentou fatores a serem considerados na implantação de um programa de métricas, como também o vocabulário a ser utilizado ao longo deste trabalho.

Como visto, a qualidade resultante da adoção de modelos de qualidade depende dos mais diversos fatores. Assim, deve-se considerar que a adoção de modelos não é sinônimo de um PDS de qualidade. Para prover qualidade ao PDS é necessário que toda a organização adote uma cultura de qualidade durante toda a execução do PDS, utilize-se de métricas para sua quantificação e posteriormente análise, e tome ações no sentido de prover melhor performance ao PDS.

3 *DATA WAREHOUSING*

Este capítulo apresenta uma visão genérica sobre Data Warehousing, e após apresenta os principais conceitos relacionados à modelagem analítica e à camada de apresentação.

O processo de construção, acesso e manutenção de um *Data Warehouse* (DW) é denominado *Data Warehousing*. Este processo tem como objetivo integrar e gerenciar dados extraídos de diferente fontes de informação de uma organização e assim possibilitar a esta uma visão única de seu negócio.

O DW é uma base de dados que proporciona aos usuários uma única fonte de informação a respeito dos seus negócios agrupando os dados históricos de uma organização, provenientes de qualquer banco de dados, planilha eletrônica, documentos textuais, entre outros [KIM98]. Segundo Inmon [INM97], o DW diferencia-se de bases de dados convencionais por ser:

- Orientado por assuntos: sempre armazena dados importantes sobre temas específicos da organização, conforme o interesse dos usuários que irão utilizá-lo;
- Integrado: integra dados provenientes de fontes distintas de forma a obter uma única forma de representação;
- Variante no Tempo: dados são dependentes do tempo. A cada ocorrência ocorrida na variável tempo, uma nova entrada deve ser criada no DW;
- Não Volátil: uma vez que um dado é inserido, este não pode ser modificado ou excluído.

3.1 Elementos de um Ambiente de *Data Warehousing*

Um ambiente de *Data Warehousing* é composto por quatro grandes componentes genéricos. A Figura 6 apresenta os quatro elementos citados em [KIM98]. O primeiro componente que aparece são os sistemas legados (também denominados sistemas operacionais ou transacionais). Estes são os sistemas onde organizações registram as transações do dia-dia de seus negócios, e são mantidos fora do ambiente de *Data Warehousing* por se ter pouco ou nenhum controle sobre o conteúdo e formato de seus dados [KIM98]. Os objetivos dos sistemas legados diferenciam-se dos de um DW. Estes possuem

como prioridades principais seu desempenho e disponibilidade e normalmente não existe nenhuma preocupação com a padronização de dados e/ou compartilhamento destes através da organização.

O segundo elemento previsto, o *Data Staging*, tem como principal objetivo buscar todos os dados destes diferentes sistemas legados e prepará-los para carga no DW. Para tal, na área do *Data Staging*, várias técnicas são aplicadas sobre os dados brutos dos sistemas legados a fim de deixá-los condizentes com o padrão exigido por um determinado DW.

No DW propriamente dito, os dados são organizados, armazenados e disponibilizados para serem consultados pelo usuário. Os dados são organizados através de um modelo dimensional, projetado de acordo com as necessidades de uma organização. A modelagem dimensional é abordada com mais detalhes na Seção 3.2 .

O último elemento é composto das ferramentas que provêem acesso aos dados, também chamado de camada de apresentação. É considerado o principal do ambiente de Data Warehousing segundo [KIM98]. O termo ferramenta é utilizado para se referir a uma grande variedade de recursos que podem ser utilizados para disponibilizar dados analíticos a usuários organizacionais [KIM98]. Mais detalhes sobre camada de apresentação podem ser encontrados na Seção 3.3.

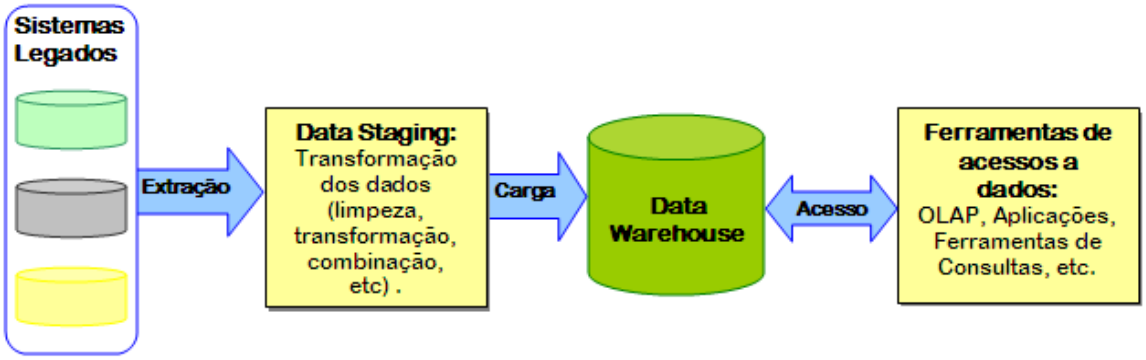

Figura 6: Elementos básicos de um DW.

3.2 Modelagem Dimensional

Modelagem dimensional é a técnica mais usada em DW, por ser a mais adequada para análise dos dados no ambiente gerencial [KIM98, INM97]. Este tipo de modelagem distribui os dados entre tabelas dimensões e fatos.

As tabelas de dimensões armazenam as descrições textuais do negócio. Por exemplo, a tabela dimensão de um projeto de software poderia armazenar dados sobre as características

de projeto de sofware como cliente, nome, tecnologia utilizada entre outras. Tabelas de dimensão caracterizam-se por possuir uma única chave primária.

Uma tabela fato normalmente armazena as medidas numéricas do negócio, como por exemplo esforço despendido por determinada tarefa, tamanho de uma determinada versão, etc. As tabelas fato caracterizam-se por suas chaves primárias serem uma composição de chaves estrangeiras das tabelas dimensões. O DW pode possuir várias tabelas fatos, cada uma representando um assunto ou negócio diferente dentro da organização. Há três tipos de tabelas fato: completamente aditivas, semi-aditivas e não aditivas. Tabelas fato completamente aditivas são as mais úteis pois podem ser sumarizadas por qualquer dimensão do modelo. As tabelas fato semi-aditivas podem ser sumarizadas somente por algumas dimensões. Já as fatos não aditivas não podem ser sumarizadas em nenhuma dimensão. Neste caso, a análise deve-se realizar registro a registro (por exemplo, taxas e índices). A solução para este tipo de fato é transformar a medida não aditiva em elementos aditivos para então armazená-lo na tabela fato.

Um dos modelos utilizados é o modelo estrela. Este apresenta uma estrutura otimizada para análise dos diferentes objetivos dos usuários. A modelagem multidimensional do tipo estrela é representada por uma tabela fato relacionada a várias tabelas dimensão. Variações do modelo estrela existem, como por exemplo o modelo constelação de fatos, composto por várias tabelas fato ligadas a um conjunto de dimensões conformadas.

A elaboração de um modelo dimensional exige a combinação das necessidades informacionais de uma organização com os dados que realmente estejam disponíveis nesta. Para tal, [KIM98] propõe um guia composto de etapas de decisão que podem facilitar este processo, alertando porém, que não há uma abordagem que possa ser aplicada diretamente a qualquer organização. Neste sentido, deve-se sempre considerar as necessidades decisicionais para qual o DW está sendo construído.

A construção de um modelo dimensional é um processo "*top-dow*". Ou seja, em um primeiro momento identificam-se os processos que representam os assuntos ou negócios da organização e que deverão ser contemplados pelo DW. Para tal, consideram-se os diferentes tipos de usuários organizacionais e suas expectativa de análise sobre os dados que deverão ser armazenados no DW. Após o levantamento da abrangência do DW na organização, torna-se possível atribuir tabelas fato a cada um dos processos contemplados e identificar possíveis dimensões do modelo. Porém, a análise "*top-down*" deve estar conciliada com a análise "*bottom-up*" dos dados de suas fontes originais a fim de haver um ajuste da informação

necessária à informação disponível na organização. O ajuste da informação definirá o nível de detalhe no qual cada dado será armazenado, ou seja, a granularidade do DW. Assim, com a granularidade definida realiza-se a definição das dimensões do modelo e das medidas que serão armazenadas nas tabelas fatos.

Maiores detalhes e informaçõ*es* sobre modelagem dimensional e projetos de *Data Warehousing* podem ser obtidos em [KIM98] e [INM97].

3.3 Camada de Apresentação

O DW fornece dados integrados e históricos que contemplam toda uma organização, desde a alta direção, que necessita de informações mais resumidas e estratégicas, até as gerências de níveis operacionais, onde dados detalhados ajudam a observar aspectos mais táticos da organização ou de um processo específico. A camada de apresentação deve permitir que os diversos tipos de usuários organizacionais tenham acesso aos dados do DW, segundo seus objetivos e perfis [KIM98, POE98].

Satisfazer os diferentes objetivos de análise encontrados em uma organização exige conhecimento do negócio e envolvimento dos usuários organizacionais com o projeto de DW. Somente desta forma ter-se-á o domínio do negócio e o real conhecimento das necessidades organizacionais.

A forma como a camada de apresentação disponibiliza a informação também é muito importante. O usuário final normalmente não posssui conhecimento suficiente para buscar a informação no formato dimensional e, mesmo que este consulte diretamente o DW para encontrar respostas às suas indagações, as consultas realizadas completamente "*ad hoc"* sem um objetivo específico e padrão pode resultar em "histórias" contadas de formas ligeiramente diferentes. Neste sentido, técnicas analíticas devem garantir a satisfação de grande ou total parte dos usuários organizacionais através de parâmetros que permitam o estabelecimento de uma estrutura analítica consistente [KIM98].

Abaixo, apresenta-se uma breve descrição sobre as necessidades de análise de diferentes perfis de usuários de um DW identificados em [POE98]:

- Executivo: deseja acesso fácil e rápido ao *status* da organização. Necessitam de relatórios pré-definidos que possam ser rapidamente localizados e acessados. Estes devem ser disponibilizados de forma gráfica e suportar detalhes quando necessário;

- Novato ou casual: acessa as informações ocasionalmente. Necessita de relatórios pré-definidos e pode se interessar em relatórios gerados a partir de um conjunto de parâmetros. Opções de análise podem ser decisivas para este usuário;
- Analista do negócio: acessa as informações diariamente, não possui o conhecimento técnico para o desenvolvimento de relatórios. Necessita de auxílio inicialmente e após deve analisar resultados sob diferentes expectativas desejando modificações, customizações e geração de novos relatórios.
- Especialista: usuário conhecedor de tecnologia. Necessita alterar parâmetros e manipular conjuntos de resultados. Facilmente gera seus próprios relatórios e os exporta para planilhas eletrônicas para futuras manipulações. Também freqüentemente desenvolve relatórios que podem ser compartilhados com outros usuários da organização;
- Desenvolvedor da aplicação: este usuário é treinado para gerar/analisar relatórios para outros usuários, definir padrões de relatórios, localização e nomenclatura. Freqüentemente preocupa-se com a performance dos relatórios.

Como visto, cada um dos níveis de usuários possui diferentes necessidades de análise e diferentes métodos de como acessar estes dados. Assim, do ponto de vista do usuário do negócio, a camada de apresentação deve oferecer ferramentas de consulta e visualização de dados com diferentes graus de sofisticação, ferramentas de relatórios padronizados, e aplicações específicas. As ferramentas devem ajustar-se às necessidades de diferente pefis de usuários, e estas necessidades traduzem o nível de suas atividades na organização (i.e. estratégico, gerencial, operacional).

É importante também que utilize-se de um vocabulário que pemita a usabilidade intuitiva da ferramenta por seus usuários. Isto possibilitará que a informação possa ser buscada com maior facilidade e que decisões decorrentes destas análises possam ser tomadas mais rapidamente.

3.3.1 OLAP

A tecnologia OLAP (*On-line Analytical Processing*), [KIM98], surgiu devido à necessidade que executivos e gerentes possuem em dispor de informações de sua organização de forma sintetizada, através de comparações, visões personalizadas e análises históricas.

As ferramentas de apresentação OLAP permitem que consultas variadas sejam realizadas sobre as medidas aditivas, navegando pelo espaço multidimensional. Estas facilidades permitem visualizar dados segundo diferentes pontos de vista (dimensões), e níveis de abstração (mais ou menos detalhado), permitindo consultar ou gerar novos fatos.

A técnologia OLAP oferece funções que permitem derivar medidas, envolvendo comparações entre períodos, percentual de diferença, médias, somas acumulativas, bem como funções estatísticas. Consultas parametrizadas ou visões podem ser produzidas através destas técnicas e então disponibilizadas a usuários.

As principais vantagens de uma ferramenta OLAP, referem-se à capacidade de visualização de forma interativa das informações sob várias formas, conforme a necessidade de detalhamento. Algumas das principais características OLAP são apresentadas abaixo:

a) *drill down:* permite aumentar o nível de detalhe da informação, diminuindo o grau de granularidade;
b) *drill up:* ao contrário do *drill down*, possibilita aumentar o grau de granularidade, diminuindo o detalhamento da informação;
c) *slice and dice:* consiste em mudar a ordem das dimensões alterando assim a orientação segundo a qual os dados são visualizados. Altera linhas por colunas de maneira a facilitar a compreensão dos usuários;
d) *drill through:* ocorre quando o usuário passa de uma informação contida em uma dimensão para outra;
e) *drill across:* consiste em fazer com que duas ou mais tabelas de fato que compartilham dimensões sejam combinadas.

Aplicativos podem disponibilizar análises pré-definidas e parametrizáveis, relacionadas a diferentes áreas da organização. As análises podem ser apresentadas através de diferentes tipos de gráficos, tabelas e relatórios.

3.3.2 Outros recursos analíticos

Como recursos analíticos adicionais aos tradicionais ferramentas OLAP, mineração de dados e novas técnicas vêm se mostrando cada vez mais necessárias no anseio de trazer maior semântica à apresentação de informação a organização, considerando os diferentes perfis de usuários e facilitando a forma como os dados podem ser analisados de forma a garantir a rapidez na tomada de decisão.

Novos recursos analíticos, como descritos em [GOL04] e [HEL02], objetivam considerar os diferentes níveis de conhecimento, tanto tecnológico como de estratégia organizacional dos usuários. Normalmente, os usuários que necessitam da informação para tomar decisões estratégicas disponibilizam pouco tempo para analisar a informação, e pouco ou nenhum conhecimento de como navegar em ferramentas OLAP. Neste sentido, não considerar as limitações e necessidades dos diferentes perfis de usuários encontrados pode resultar na não utilização de uma camada de apresentação.

A utilização de recursos analíticos que apresentam informações na forma gráfica (e.g gráficos e *dashboards*) permite que a informação organizacional torne-se mais democrática, isto é, que seu significado é compreensível para um número maior de usuários da organização. No entanto, a utilização de gráficos pode ainda não ser suficiente para usuários que desconhecem a estratégia organizacional. Por exemplo, ao verificar a taxa de 5% de remoção de defeitos de um determinado produto, um usuário que não conhecer a meta organizacional para tal fator, e não saberá se esta taxa é um bom ou mal resultado.

Neste sentido, *dashboards* diferenciam-se dos gráficos convencionais, disponibilizando a informação segundo indicadores de qualidade previamente definidos pela organização. Assim, ao analisar determinada informação, o usuário reconhece imediatamente qualquer desvio de desempenho. Os *dashboards* podem ser gerados para mostrar diferentes assuntos de uma organização.

Há algum tempo *dashboards* vêm sendo utilizados por gerentes e executivos para acompanhar o desempenho de seus negócios. A visualização de informação através destes é um importante recurso analítico uma vez que eles podem tanto ilustrar o desempenho da organização como um todo, quanto focar em diferentes aspectos, indicando seu respectivo nível de qualidade.

O acompanhamento efetivo do desempenho organizacional torna-se cada vez mais importante, existindo a uma grande necessidade de detectar-se desvios de performance rapidamente para que sejam rapidamente tratados. Neste sentido, a utilização de alertas, possivelmente acoplados a *dashboards*, mostra-se um outro importante recurso analítico. O alerta pode disparar automaticamente avisos aos usuários sempre que identificar comportamentos não condizentes com o indicador de qualidade organizacional. Aplicações já começam a utilizar-se destes recursos. Um exemplo da utilização de recursos analíticos no âmbito de sistemas de informação para o acompanhamento de PDSs é explorado em [SEL04, KIM98 e POE98].

3.4 Medodologia de Desenvolvimento

O projeto de um DW depende do desenvolvimento e integração de várias tarefas e ferramentas. Considerando a grande dificuldade em coordenar estas múltiplas características Kimball em [KIM98] uma propõe a metodologia para desenvolvimento de projetos deste tipo, cuja principais fases encontram-se ilustradas na Figura 7.

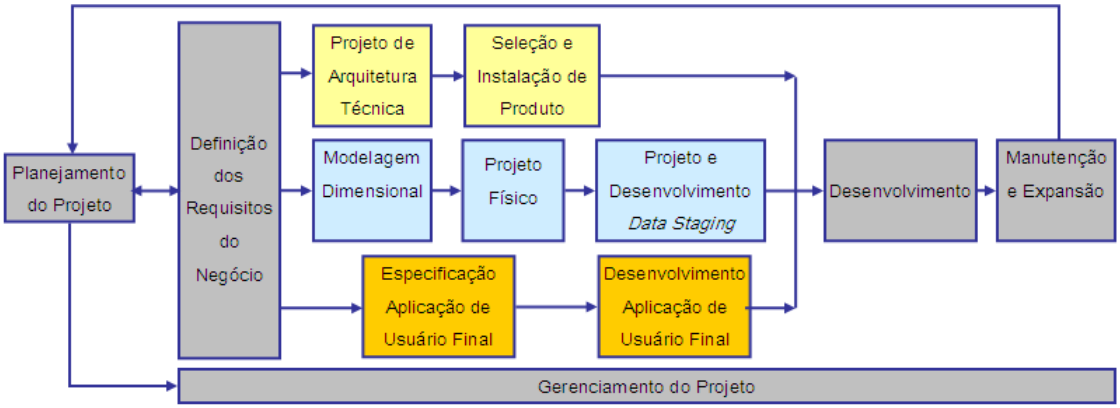

Figura 7: Metodologia - Projeto de Data Warehousing.

Conforme a metodologia proposta, o desenvolvimento do projeto de *Data Warehousing* inicia-se pela fase de *planejamento de projeto*. O planejamento define o escopo de todo projeto resultante. Nesta fase avalia-se a aptidão da organização para o desenvolvimento de um DW, estabelece-se um escopo e um objetivo prévio, obtém-se recursos e, então inicia-se o projeto.

A *definição dos requisitos do negócio*, segunda fase prevista, mostra-se crucial uma vez que decisões tomadas guiarão o restante do projeto. Esta fase tem por objetivo alinhar o *Data Warehousing* com os requisitos do negócio. O projetista deve possuir entendimento das necessidades do negócio e as trazer para o projeto de DW. Usuários do negócio e suas necessidades devem ser considerados durante todo o desenvolvimento do projeto.

O *gerenciamento do projeto* inicia-se com definição dos requisitos organizacionais e estende-se durante todo o tempo de desenvolvimento do projeto. Este possui como atividades principais: manter e gerenciar o plano de projeto sua documentação e seu escopo e ainda o desenvolvimento e manutenção de um plano de comunicação a fim de gerenciar as expectativas de seus usuários.

Finalizada a *definição dos requisitos do negócio* e iniciado o *gerenciamento do projeto,* três fluxos de atividades podem ser desenvolvidos paralelamente:

- Fluxo de atividades relacionadas aos dados: a *modelagem dimensional*, o *projeto físico* e o *projeto e desenvolvimento de um Data Staging*;
- Fluxo de atividades relacionadas à tecnologia: *projeto de arquitetura técnica* e *seleção e instalação de produtos,* as quais proverão suporte ao projeto de *Data Warehousing*;
- Fluxo de atividades relacionadas à aplicação: atividades relativas à especificação *da aplicação de usuário final* e *desenvolvimento da aplicação de usuário final.*

A *definição dos requisitos de negócio* realizada anteriormente define as necessidades analíticas dos usuários do negócio. A *modelagem dimensional* suporta estes requisitos, como discutido na Seção 3.2.

O *projeto físico* foca na definição da estrutura física necessária para suportar o banco de dados lógico. Este inclui também a definição de uma nomenclatura padrão e configuração de um ambiente para o banco de dados.

O *projeto e desenvolvimento de um Data Staging* possui três atividades principais: extração, transformação e carga. O *Data Staging* é responsável por extrair os dados do ambiente transacional, transformá-los em um formato adequado ao DW e, então realizar a carga destes para o DW. Este processo, de extração, tranformação e carga de dados, ETC, é considerado uma das maiores dificuldades na implantação de um projeto de *Data Warehousing*.

O ambiente de *Data Warehousing* requer a integração de diversas tecnologias. O *projeto de arquitetura técnica* estabelece uma estrutura de suporte a estas tecnologias. Para tal, esta deve considerar: os requisitos do negócio, o ambiente tecnológico atual e aspirações futuras. A *seleção e instalação de produtos* visam definir os componentes que integrarão a arquitetura técnica, selecioná-los e testá-los. Com exemplos de componentes pode ser citados: a plataforma de hardware, sistema gerenciador de banco de dados, entre outros. O processo de avaliação dos componentes é definido de acordo com a necessidade da arquitetura. Uma vez que o produto encontra-se avaliado e selecionado este é integrado a arquitetura de *Data Warehousing*.

A fase de *especificação e de desenvolvimento da aplicação de usuário* final envolvem definir mecanismos para que os usuários finais tenham acesso aos dados armazenados no DW. A aplicação de usuário final pode variar de um conjunto pré-definidos de relatórios OLAP ao desenvolvimento de uma camada de apresentação, como discutido na Seção 3.3.

Os fluxos de atividades de tecnologia, de dados e de aplicação convergem para o *desenvolvimento* do projeto. Neste sentido, um grande planejamento é necessário para garantir a união dos mais diferentes aspectos destas atividades. A fase de *manutenção e expansão* envolve as seguintes áreas: suporte o projeto desenvolvido (a fim de mantê-lo sempre atendendo as expectativas do usuário), treinamentos da equipe para incentivar a troca de idéias e garantir atualização tecnológica.

3.5 Considerações Finais

O desenvolvimento de projetos de *Data Warehousing* é um objetivo perseguido por muitas organizações. Contudo muitos são os relatos de fracassos encontrados na literatura. Assim, a revisão bibligráfica que transcorreu neste capítulo teve por objetivo apresentar as várias etapas que compõem o desenvolvimento de um projeto deste tipo.

O capítulo iniciou com uma breve explicação a respeito dos elementos que compõem um *Data Warehousing*. Após, devido aos objetivos do presente trabalho, apresentou-se detalhadamente a modelagem multidimensional e a camada de apresentação com alguns possíveis recursos analíticos.

A finalização do capítulo deu-se com a apresentação da metodologia de desenvolvimento de projetos de *Data Warehousing* proposta em [KIM98]. Nesta seção apresenta-se cada uma das atividades envolvidas juntamente com a seqüência que devem ser executadas e aspectos que devem ser considerados no seu desenvolvimento.

As atividades descritas pela metodologia de desenvolvimento proposta por Kimball guiam organizações no desenvolvimento de projetos de *Data Warehousing.* Contudo, nada é mais importante para o desenvolvimento de um projeto que o respeito aos requisitos do usuário final. O sucesso de um *Data Warehousing* estará seriamente comprometido se os requisitos do usuário final não forem atendidos durante a fase de desenvolvimento e apresentados de forma condizente às necessidades e restrições de análise do usuário final.

4 ESTUDO DE CASO

Este capítulo descreve o estudo de caso que motivou o desenvolvimento deste trabalho. Neste sentido, apresenta-se a realidade encontrada na organização e as necessidades quanto à qualidade de seus produtos.

4.1 Contexto do Estudo de Caso

O estudo de caso relatado neste trabalho foi realizado na operação de software de uma grande organização. Durante o desenvolvimento desta pesquisa, esta possuía avaliação SW-CMM nível 2 com implantação de várias práticas do nível 3. O presente trabalho inseriu-se nos esforços desta organização para avaliação nível 3 no final do ano de 2005.

A organização encontra-se muito empenhada em implantar uma “cultura de qualidade” [SOM04], e para tal, despende muito esforço na busca de alternativas para atingí-la. Ela possui uma equipe dedicada ao projeto de alcance do SW-CMM nível 3 e às atividades de SQA. Assim, os dados e evidências necessários ao andamento desta pesquisa foram coletados através da inserção da autora no âmbito desta equipe e no convívio com seis projetos da organização, de agosto de 2004 até o presente momento.

No período inicial desta pesquisa na organização, essa se empenhava em definir uma OSSP condizente a sua realidade. Os processos organizacionais sofriam constantes alterações, o que dificultava o seu acompanhamento, o PM adotado encontrava-se praticamente definido somente faltando algumas métricas. As reuniões de acompanhamento dos projetos eram baseadas no PM recém definido; mas a coleta dos dados, o cálculo das métricas e sua apresentação eram realizados manualmente, consumindo grande esforço da equipe do projeto e dando margem a muitas inconsistências.

Após, a OSSP incluiu todos os processos vislumbrados como ideais pela organização. A estabilidade da OSSP facilitou a implantação dos processos organizacionais em todos os projetos. O PM igualmente estabilizou-se. Contudo, a dificuldade de coleta de dados e cálculo das métricas para posterior análise persistia. O presente trabalho insere-se nesta realidade, e buscou alternativas para prover à organização uma infra-estrutura que facilitasse a adoção de um PM e uma visão unificada da realidade organizacional através de um repositório organizacional provido de mecanismos de análise sobre o PDS.

As seções que seguem apresentam os resultados das atividades desenvolvidas na organização do estudo de caso. Sinteticamente, estas atividades foram:

- Estudo do PM adotado inicialmente pela organização juntamente com a identificação de suas respectivas fontes de dados;
- Estudo da OSSP, informação gerada por esta e forma como foi adotada pelos projetos da organização;
- Cruzamento dos dados oriundos da execução dos processos da OSSP e os dados necessários ao PM;
- Estudo dos dados oriundo dos diferentes projetos da organização, visando identificar heterogenidades entre projetos;
- Estudo dos possíveis perfis organizacionais e suas expectativas de análise;
- Análise de problemas encontrados.

No decorrer do trabalho, as necessidades levantadas por este estudo de caso, assim como a proposta para análise do PDS da organização, foram apresentadas acompanhadas e aprovadas pela organização através de apresentações e relatórios diversos [NOV04, NOV04a e NOV04b].

A seguir, inicialmente apresenta-se a metodologia de pesquisa adotada no andamento deste trabalho. Após, as atividades acima descritas são apresentadas em detalhes através das seções que apresentam a organização quanto ao modelo de qualidade e PM adotados, quanto à forma como o PDS está estruturado, em perspectiva de organização e de projeto e quanto aos perfis de usuários organizacionais. Finaliza-se este capítulo analisando as dificuldades encontradas frente aos objetivos deste trabalho e as necessidades da organização quanto à análise do PDS.

4.2 Metodologia da Pesquisa

O método de pesquisa utilizado neste trabalho é estudo de caso, adotado conforme [YIN01]:

> *"Um estudo de caso é uma investigação empírica que investiga um fenômeno contemporâneo dentro de seu contexto da vida real, especialmente quando os limites entre o fenômeno e o contexto não estão claramente definidos."*

Neste estudo de caso foram identificados necessidades, dificuldades e requisitos que influenciam a adoção de um PM e uma infra-estrutura de suporte a ele.

Conforme [YIN01], o método de estudo de caso pode ser de caso único e de casos múltiplos. Os fundamentos lógicos citados por [YIN01] para utilização do método de estudo de caso único são apresentados abaixo:

- Quando o caso único representa um caso decisivo para se testar uma teoria;
- Quando o caso único representa um caso raro ou extremo;
- Quando o caso único é um caso revelador;
- Quando o caso único é um caso representativo, pode representar um "projeto" típico entre muitos projetos diferentes. Parte-se do princípio de que as lições que se aprendem desses casos fornecem muitas informações sobre as experiências da pessoa ou instituição usual.

O estudo de caso desenvolvido neste trabalho caracteriza-se como estudo de caso único por ter sido desenvolvido em uma única organização de desenvolvimento de software. Considerando-se os objetivos, em termos de análise e mensuração de processos que devem ser tratados para a passagem do nível 2 ao nível 3 do SW-CMM, e que esta organização apresente tais características semelhantes, este estudo de caso enquadra-se como caso representativo.

A unidade de análise é a entidade central do problema de pesquisa. Conforme [YIN01], a definição da unidade de análise está relacionada à maneira como as questões iniciais da pesquisa foram definidas e devem ser semelhantes àquelas previamente estudadas por outras pessoas, para que se possam comparar às descobertas de forma clara e definida. Embora seja normalmente definida como sendo indivíduos, grupos ou organizações, ela pode também ser uma atividade, um processo, um aspecto ou uma dimensão comportamental [LAZ95].

A mensuração e análise do PDS da organização do estudo de caso é a unidade de análise desta pesquisa, a qual se caracteriza também como um projeto de pesquisa holístico devido ao fato de considerar uma única unidade de análise.

A coleta das evidências é a base para a realização do estudo de caso. Os dados utilizados em um estudo de caso podem se basear em muitas fontes de evidências (dados). Segundo [YIN01], as evidências de um estudo de caso podem vir de seis fontes distintas:

documentos, registros em arquivo, entrevistas, observação direta, observação participante e artefatos físicos.

A coleta de evidências desta pesquisa centrou-se em:

a) documentação, análise de documentos organizacionais relacionados à forma como a organização estrutura e conceitua seu PDS, e ainda, documentos requeridos pelo modelo de qualidade adotado, como: documento de padronização dos processos organizacionais, Guia de métricas organizacional, guia de ciclos de vida, entre outros;

b) entrevistas pessoais realizadas com o gerente do projeto SW-CMM na organização alvo;

c) registros em arquivos, análise de cronogramas e banco de dados de projetos;

d) observação da participante através de realização de apresentações que validaram alguns dados e forneceram informações adicionais para o andamento de trabalho;

e) observação direta na equipe de trabalho do projeto SW-CMM através de bolsa no período em que a organização transitava do SW-CMM nível 2 para SW-CMM nível 3, a saber, de agosto de 2004 até o presente momento.

Para realizar a análise das evidências torna-se necessária a identificação de estratégias analíticas, bem como dos métodos de análise. A análise de evidências consiste em examinar, categorizar, classificar em tabelas ou recombinar as evidências tendo em vista proposições inicias de um estudo. Existem basicamente duas estratégias, uma que se baseia nas proposições teóricas, e outra que desenvolve uma descrição de caso [YIN01].

A primeira, proposições teóricas, consiste em seguir os objetivos e o projeto original que incentivaram o estudo. Baseia-se, presumivelmente, em proposições que refletem um conjunto de questões da pesquisa, as revisões feitas na literatura sobre o assunto e as novas interpretações que possam surgir. A segunda estratégia analítica é desenvolver uma estrutura descritiva a fim de organizar o estudo de caso. A abordagem descritiva pode ajudar a identificar as ligações causais apropriadas a serem analisadas [YIN01].

A investigação deste estudo de caso baseia-se em várias fontes de evidências. Assim, a estratégia analítica utilizada refere-se à descrição do ambiente encontrado através das diversas fontes enfatizando os aspectos relacionados ao PDS e sua mensuração.

4.3 Modelo de Qualidade e Métricas

Dentre as várias alternativas de modelos de qualidade que buscam auxiliar organizações a prover qualidade ao PDS, a organização em questão optou pela utilização do modelo SW-CMM, descrito na Seção 2.2.1. Como já citado, atualmente a organização possui avaliação SW-CMM nível 2, mas utiliza-se de várias técnicas e processos do nível 3.

Um dos quesitos requeridos pelo SW-CMM é a utilização de medições a partir do nível 2. O PM definido pela organização [DOC05] estrutura-se em quatro colunas: métricas, objetivo, dados coletados e função da métrica.

Segundo a nomenclatura utililizada neste trabalho a primeira coluna do PM, denominada métricas, é constituída por métricas derivadas enquanto que os dados coletados constituem-se em sua maioria de métricas base, necessárias ao cálculo das métricas derivadas.

Contudo, a organização desconsidera que dados coletados, em especial os que representam métricas base de estimativas e realizações (e.g. esforço realizado e esforço no *baseline original*, tamanho realizado e tamanho no *baseline original*, entre outros), são úteis a análises do PDS. Um exemplo de sua utilização pode ser encontrado em reuniões de acompanhamento de projetos onde apresentações comparando estimativas e realizações através de gráficos possuem um efeito maior que apenas apresentação das variações destas.

A Tabela 3 apresenta o PM da organização segundo a nomenclatura utilizada neste trabalho.

O conceito de retrabalho, citado no PM da organização, tem como significado a ação tomada para adequar itens com defeitos ou não conformidade às exigências ou especificações. O retrabalho, especialmente o imprevisto, é uma causa bastante freqüente de atrasos em projetos, na maioria das áreas de aplicação. A equipe do projeto deve fazer o máximo esforço possível para minimizar o retrabalho [PMB00].

A realidade de uma organização SW-CMM nível 2 é a realidade onde cada projeto representa um “mundo” diferente [PAU93]. Isto pôde ser observado na organização em questão realcionado à forma como os projetos armazenam os dados necessários ao cálculo das métricas. Esta heterogenidade encontrada nos projetos é um dos fatores a ser considerado quando da utilização do PM.

Tabela 3: Programa de Métricas Organizacional.

Métricas	Objetivo	Dados a serem coletados	Função da Métrica
Variação de Cronograma	Entregar no prazo	Data Inicial Baseline Original - DIBO Data Inicial Baseline Revisado – DIBR Data Inicial Real - DIR Data Término Baseline Original - DTBO Data Término Baseline Revisado - DTBR Data Término Real - DTR	Variação de cronograma (BR) =(DTR-DTBR)*100/(DTBR-DIBR) Variação de cronograma (BO)=(DTR-DTBO)*100/(DTBO - DIBO)
Variação de Esforço	Melhorar acurácia de estimativas	Esforço Baseline Original – EBO Esforço Baseline Revisado - EBR Esforço Real - ER	Variação de esforço (EBR)= (ER-EBR)*100/(EBR) Variação de esforço (BO)= (ER-EBO)*100/(EBO)
Satisfação do Cliente	Maximizar a satisfação do cliente	Índice de Satisfação do Cliente - SC	Satisfação do cliente = Média(SC)
Variação de Tamanho	Melhorar acurácia de estimativas	Tamanho Baseline Original (em KLOC) - TBO Tamanho Baseline Revisado (em KLOC) - TBR Tamanho Real (em KLOC) - TR	Variação de tamanho (BR)= (TR-TBR)*100/(TBR) Variação de tamanho (BO)= (TR-TBO)*100/(TBO)
Custo da Qualidade	Minimizar retrabalho	Custo Real – CR Custo Atividade Revisão - CARV Custo Fase de Teste - CFT Custo Atividade de Qualidade - CAQ Custo Atividade de Retrabalho – CART	Custo da qualidade= ((CARV + CFT + CAQ+ CART) / CR)*100
Volatilidade de Requisitos	Analisar a estabilidade dos Requisitos	Número de requisitos adicionados – RA Número de requisitos removidos - RR Número de requisitos modificados - RM Número de requisitos aprovados pelo cliente – RAC	Volatilidade de requisitos= ((RA+RR + RM)/RAC)
Eficiência de Remoção de defeitos	Melhorar a qualidade dos entregáveis	Número de defeitos encontrados internamente - DI Número de defeitos encontrados no cliente - DE	Eficiência de remoção= (DI/ (DI + DE))
Densidade de Defeitos Entregues	Fornecer visibilidade da qualidade do produto entregue ao cliente	DE TR (em KLOC)	Densidade de defeitos entregues= DE/TR
Densidade de Defeitos Internos	Melhorar a qualidade dos entregáveis	DI TR (em KLOC)	Densidade de defeitos (Internos)= DI/TR
Densidade de Defeitos	Melhorar a qualidade dos entregáveis	DE DI TR (em KLOC)	Densidade de defeitos = (DE + DI) / TR
Eficiência de Revisão	Melhorar a produtividade	Esforço Atividade de Revisão - EAR Número de defeitos encontrados em revisões – PR	Eficiência de revisão = PR/ EAR
Produtividade	Melhorar a produtividade	Esforço Real – ER Tamanho Real – TR	Produtividade= ER/TR
Variação de Custo	Melhorar acurácia de estimativas	Custo Baseline Original - CBO Custo Baseline Revisado - CBR Custo Real - CR	Variação de custo (CBR)= (CR-CBR)*100/(CBR) Variação de custo (BO)= (CR-CBO)*100/(CBO)

O PM organizacional definido contempla as seguintes AQs: esforço, duração, custo, tamanho, qualidade e requisitos. As diferentes AQs permitem que a organização realize análises considerando diferentes aspectos do processo de desenvolvimento.

A Tabela 4 explicita as fontes de coleta dos dados dos diferentes projetos organizacionais, organizados pelas respectivas AQs.

Tabela 4: Fontes de Coleta para o PM.

Área de Qualidade	Fonte de Informação
Esforço	Planilha Eletrônica Project Server
Duração	Project Server
Custo	Project Server
Tamanho	Project Server
Qualidade	Planilha Eletrônica Bugzilla Clear Quest
Requisitos	Planilha Eletrônica Requesite Pro

À medida que o PM foi implantado na organização, novas necessidades foram identificadas. A inserção de novas métricas, como Custo da Qualidade, se fizeram necessárias sendo então incluídas ao PM organizacional. Quanto à apresentação dos resultados das métricas identificou-se que muitas vezes estes eram insuficientes para análise realizada pelo usuário organizacional devido a falta de parâmetros que identificassem se o resultado obtido condizia ou não à expectativa da organização. Neste sentido, vislumbrou-se a necessidade de algum mecanismo que facilitasse a análise das métricas e que fosse capaz de posicionar, para um usuário, a qualidade de seu processo quanto ao nível de qualidade esperado pela organização.

4.4 Processo de Desenvolvimento de Software

A organização alvo deste trabalho presta serviços de desenvolvimento e manutenção de software. Diversos projetos de software são desenvolvidos paralelamente e a escolha das tecnologias e ferramentas que apóiam o PDS são dependentes exclusivamente da especificação dos requisitos do cliente.

Como definido pelo modelo SW-CMM, a organização definiu uma OSSP que é seguida por toda a organização e cada um dos projetos organizacionais definiu e segue sua própria PDSP. Através da utilização do OSSP a organização obteve a padronização de alguns ativos de projetos, como a nomenclatura de fases de desenvolvimento dos projetos e dos tipos de atividades. Porém, a padronização de algumas características poderia dificultar o

desenvolvimento do projeto. Como exemplo, cita-se diferentes modelos de processo de software adotados pelos projetos (e.g. iterativo, seqüencial), diferentes ferramentas de apoio ao PDS (e.g. Project Server, Excel), diferentes formas de classificar uma mesma informação (e.g. severidade de defeitos), entre outras.

4.5 Perfis de Análise de Usuários da Organização

A provisão de recursos analíticos para o acompanhamento do PDS deve considerar o usuário final. Assim, considerando os papéis organizacionais desempenhados por diferentes pessoas na organização. Foram identificados, através de observação direta, os seguintes perfis de análise:

- Organizacional: composto por pessoas que ocupam papéis de Gerente Sênior e executivos da organização;
- Projeto: composto por pessoas que ocupam papéis de Gerente de Projeto, Gerente de Software e *Software Quality Assurance* (SQA);
- Liderança de Projeto: pessoas responsáveis por alguma parte específica do PDS (e.g. teste, gerência de configuração, construção, projeto).

Considerando os perfis acima descritos, a Tabela 5 descreve sucintamente as expectativas de análise levantadas, através de observação direta na empresa alvo, para cada um dos perfis.

Tabela 5: Perfis de Análise Organizacional.

Perfil	Expectativa de Análise
Organizacional	§ Foco na organização, desempenho organizacional; § Visão sintetizada de todos os projetos organizacionais; § Métricas comparando projetos: § em andamento e concluídos § diferentes tipos de projetos § em um determinado período de tempo.
Projeto	§ Foco em um projeto específico; § Métricas comparando projetos: § em andamento e concluídos § tipos de projetos § em um determinado período de tempo § Métricas sobre um projeto: § variando no tempo, histórico e real.
Liderança de Projeto	§ Foco em um projeto específico e em suas fases e atividades; § Idem ao perfil de Projeto excluindo-se métricas referentes a custo; § Métricas sobre a fase e atividades: § variando no tempo, histórico e real.

4.6 Necessidades e Problemas

A definição do PM atende uma necessidade de SW-CMM nível 2. Já, a definição da OSSP, da PDSP, e o desenvolvimento de um repositório que estabeleça e mantenha os processos organizacionais, juntamente com suas métricas e algumas análises sobre estas, são características de SW-CMM nível 3.

Para tanto, o desenvolvimento de uma infra-estrutura de apoio ao PM definido e de um repositório organizacional munido de recursos analíticos no cenário acima descrito, confronta-se com as várias realidades de projeto. A nomenclatura e o formato dos ativos gerados mostram-se distintos nos projetos organizacionais, o que dificulta sua centralização e entendimento. Também, a não formalização de termos organizacionais dificulta a criação de um repositório. Um exemplo desta realidade, é representado pelos termos “versão” e “projeto de software”. Estes termos inicialmente não possuíam características que os diferenciasse. Neste sentido, a necessidade de explicitar diversos conceitos utilizados no dia-dia organizacional se fez necessária uma vez que um repositório não poderia suportar termos ambíguos.

O PM definido pela organização juntamente com a OSSP que determina geração de alguns ativos poderia garantir a consistência da informação necessária ao repositório organizacional. Contudo, identificou-se um “*gap*” entre a informação que a organização vislumbrava disponibilizar no repositório e a informação realmente disponibilizada pelos projetos. Como exemplo desta realidade, o esforço realizado e estimado em alguns projetos é coletado por atividade, em outros o esforço realizado é armazenado por tipo de atividade e o esforço estimado armazenado por atividade.

Assim, características como as descritas, identificadas através das fontes de evidências, foram fator determinante de muitas das decisões tomadas durante o desenvolvimento desta pesquisa.

5 TRABALHOS RELACIONADOS

Este capítulo descreve alguns trabalhos relacionados aos objetivos desta pesquisa, análise de PDS. A apresentação destes é realizada conforme aspectos considerados no desenvolvimento deste trabalho, aspectos de como os dados resultantes do PDS são armazenados e apresentados aos usuários finais.

Há algum tempo organizações buscam alternativas para acompanhar permanentemente seus PDSs. Métodos mais rústicos utilizam-se de planilhas eletrônicas. Porém, a utilização de novas tecnologias combinadas às necessidades específicas das organizações de software torna os repositórios de dados uma opção atrativa. Os repositórios de dados podem ser desenvolvidos visando o armazenamento de diferentes aspectos do PDS e a utilização de diferentes tecnologias.

Data Warehousing tem sido de grande valia em repositórios de dados organizacionais. Novas tendências vêm surgindo tendo como diferencial o monitoramento dos dados armazenados e a preocupação de como estes serão apresentados ao usuário final [HEL02].

Este capítulo apresenta alguns tópicos e propostas de como planilhas eletrônicas e repositórios de dados podem ser utilizados no acompanhamento de PDS e finaliza discorrendo sobre os elementos destas novas tendências de projetos de *Data Warehousing*.

5.1 Planilhas Eletrônicas - *Trackers*

As planilhas eletrônicas sempre estiveram presentes no dia-dia organizacional. Devido ao fato de serem tão familiares ao ambiente organizacional, acabaram sendo adotadas para o acompanhamento do PDS. Para isto a organização define padronizações de como os dados de projetos são representados nas diferentes planilhas, as relações destes para o cálculo das métricas, bem como variadas formas de análise através de gráficos. A estas planilhas, destinadas ao acompanhamento do PDS, denominaremos de *Trackers*.

A utilização de *Trackers* tem a vantagem de não requerer muito investimento financeiro da organização. Por outro lado, dificilmente a padronização estabelecida do *Tracker* suportará o armazenamento das diversas características das diferentes estruturas de

ciclos de vidas de PDS presentes em uma organização como também, para prover análises que suporte a todo o PM da organização.

A análise de métricas através do *Tracker* exige grande esforço na coleta dos dados para seu cálculo. Os dados devem ser inseridos e atualizados manualmente no *Tracker* a cada atualização destes no ambiente transacional do projeto. O processo de inserção de dados em um *Tracker* pode ser ainda mais oneroso em projetos com grandes quantidades de dados e/ou freqüentemente atualizados.

As diferentes padronizações de *Tracker* através dos projetos da organização é outro aspecto a ser considerado, uma vez que a informação disponibilizada em cada um dos projetos pode não ser uniforme (e.g. métricas em diferentes unidades, informações não existentes no projeto, etc.). Contudo, mesmo utilizando-se de um único formato de *Tracker*, este não provê uma visão consolidada dos projetos organizacionais.

Através da utilização de *Trackers* a organização dispõe da análise de cada um dos projetos da organização, contudo se esta desejar conhecer o desempenho da organização como um todo deverá dispor de algum mecanismo que extraia os dados dos diferentes *Trackers* utilizados e os centralize através de uma base para a realização de consultas.

Os dados armazenados em um *Tracker* refletem um determinado momento de um único projeto, não permitindo a análise de dados históricos deste projeto ou mesmo de diversos projetos. A análise histórica de dados de um projeto ou de vários igualmente necessita da centralização dos dados em uma base de forma a permitir que dados sejam consultados e/ou comparados.

A análise dos dados disponibilizada através do *Tracker* é totalmente dependente de sua formatação, isto é, dependente dos parâmetros inicialmente especificados durante sua formatação. Neste sentido, a expansão ou a modificação das métricas disponibilizadas pelo *Tracker* depende de uma reformatação deste. O *Tracker* é utilizado e mantido pelos projetos sem a existência de algum tipo de controle. Neste sentido, não existem garantias sobre os dados coletados para o cálculo das métricas e sua posterior análise.

Uma vez que o formato e as métricas comportadas pelo *Tracker* determina as análises disponibilizadas, estas também determinaram a abrangência dos dados disponibilizados sobre os usuários da organização. Devido ao fato de não prover consultas, sua especificação determina se ele satisfaz as necessidades de análise de um grupo de usuário ou de uma organização inteira.

Concluindo, a análise provida pelo *Tracker* é pré-definida de acordo com o formato adotado e os recursos de análise disponibilizados aos usuários restringem-se aos recursos disponibilizados por planilhas eletrônicas. A Figura 8 apresenta um exemplo de um *Tracker* utilizado para acompanhamento de PDS.

		Análise			Projeto			Construção			PR	Retrabalho	Tamanho	Tamanho	Tamanho
Iteração	ID	%	Baseline	Actual	%	Baseline	Actual	%	Baseline	Actual			Baseline	Actual	%
Programa		98%	3276	6948	83%	2289	2385	81%	6428	14369	383	3916	1007	866	
I2	FREQ22	100%	98	153	100%	63	74	100%	182	868	38	297	29.2	29.2	100%
I2	FREQ24	100%	83	153	100%	53	109	100%	153	317	53	45	24.5	24.5	100%
I2	FREQ25	100%	27	118	100%	18	06	100%	51	205	08	73	7.6	7.6	100%
I3	FREQ16	100%	42	121	100%	28	31	100%	120	141	09	42	12.3	12.3	100%
I3	FREQ17	100%	129	484	100%	83	89	100%	239	658	49	325	38.5	38.5	100%
I3	FREQ18	100%	129	264	100%	83	63	100%	239	365	16	193	38.5	38.5	100%
I3	FREQ21-2	100%	13	57	100%	09	13	100%	25	268	04	111	4.0	4.0	100%
I3	FREQ23	100%	67	284	100%	44	27	100%	125	168	06	151	19.8	19.8	100%
I3	FREQ26	100%	83	299	100%	53	66	100%	153	567	09	196	24.5	24.5	100%
I3	FREQ27	100%	129	175	100%	83	47	100%	239	322	07	168	38.5	38.5	100%
I3	FREQ28	100%	27	89	100%	18	17	100%	49	52	03	25	7.6	7.6	100%
I4B1	FREQ04	100%	27	36	100%	18	14	50%	49	05	00	00	7.6	5.6	73%
I4B1	FREQ15	100%	83	189	100%	53	208	90%	153	369	10	45	24.5	23.2	95%
I4	FREQ29-1	100%	212	864	100%	135	92	100%	394	990	09	303	64.0	64.0	100%
I4	FREQ29-2	100%	53	192	100%	34	28	100%	99	105	03	35	16.0	16.0	100%
I4	FREQ29-3	100%	53	197	100%	34	35	100%	99	99	03	42	16.0	16.0	100%
I4	FREQ29-4	100%	53	164	100%	34	09	100%	99	146	04	33	16.0	16.0	100%
I4	FREQ29-5	100%	53	167	100%	34	38	100%	99	176	04	19	16.0	16.0	100%
I4	FREQ29-6	100%	53	235	100%	34	14	100%	99	302	32	91	16.0	16.0	100%
I4	FREQ29-7	100%	53	166	100%	34	16	100%	99	96	01	06	16.0	16.0	100%
I4	FREQ30	100%	160	108	100%	103	14	100%	296	197	02	95	47.9	47.9	100%
I4B2	FREQ31	90%	17	01	0%	12	00	0%	31	00	00	00	4.7	1.2	26%
I4B2	FREQ32	100%	17	00	0%	12	00	0%	31	00	00	00	4.7	1.3	29%
I4B1	FREQ33	100%	114	350	100%	73	140	90%	210	158	05	16	33.9	32.0	95%
I4B1	FREQ34	100%	96	68	100%	62	111	75%	178	220	02	04	28.6	24.8	87%
I4B2	FREQ35	5%	37	00	0%	24	00	0%	67	00	00	00	10.5	0.2	1%
I4B2	FREQ36	100%	183	354	0%	117	00	0%	340	00	06	40	54.9	15.7	29%
I4B2	FREQ37	90%	318	125	0%	203	00	0%	591	00	00	00	95.7	24.6	26%
I4	NEREQ03	100%	10	00	100%	10	00	100%	42	00	00	00	5.5	5.5	100%

Figura 8: Exemplo de Tracker.

5.2 Repositórios Organizacionais

Repositórios de dados possuem como objetivo armazenar dados provenientes de PDS centralizando-os e possibilitando a organização uma visão centralizada dos diferentes projetos de uma organização. Os repositórios podem ser implementados vislumbrando diferentes aspectos do PDS e tecnologias para seu desenvolvimento. Estes podem variar de implementações calçadas em um SGBD com inserção manual de dados a sofisticados ambientes de *Data Warehousing*. Abaixo, três propostas de repositórios encontradas na literatura são discutidas. Cada uma foca diferentes aspectos tanto quanto ao PDS, como às tecnologias utilizadas. Porém todas enfatizam o armazenamento de métricas para mensuração quantitativa da qualidade do PDS.

5.2.1 Banco Histórico

O Banco Histórico [SUB99] é uma proposta de repositório implementada por uma empresa do ramo de prestação de serviços de tecnologia e desenvolvimento de software. Este repositório tem como objetivo armazenar as métricas e as lições aprendidas de projetos executados. A coleta dos dados é realizada por ferramentas especialmente desenvolvidas, as quais são baseadas em modelos que disponibilizam, desde sua origem, um conjunto mínimo de dados comuns a todos os projetos, requeridos para análise de métricas definidas pela organização.

O acesso aos dados armazenados no repositório é também realizado através de uma ferramenta proprietária de consulta, onde os parâmetros da consulta desejada são passados à ferramenta, que retorna dados que podem ser exportados para planilhas eletrônicas para a geração de gráficos. O Banco Histórico possibilita a seus usuários uma visão consolidada de todos os projetos da organização ou de um projeto específico. Porém, por armazenar informações apenas no nível geral de projeto, não possibilita a análise de desempenho de níveis mais detalhados do projeto, como fase ou atividades.

A utilização do Banco Histórico dentro da organização teve início com 75 projetos, sendo estendido posteriormente para 145 projetos. Como principais benefícios de sua adoção relata-se institucionalização do PM da organização e os benefícios da utilização de métricas comuns a todos os projetos, o que permitiu a realização de comparações e análises de desempenho entre eles. Como trabalho futuro o autor cita a necessidade de informações de fases de projetos. Contudo, pesquisou-se na literatura o andamento deste trabalho e nada foi encontrado.

5.2.2 Repositório de Métricas

O Repositório de Métricas realatado em [OLI99] é composto de duas bases de dados. A primeira, chamada de meta base de dados, armazena todos os dados e atributos dos modelos dos processos. A segunda, denominada base de processos, é subdividida em base de processos e base reais. A base de processos é responsável por armazenar a definição de um processo, enquanto que a base de dados realizados captura os dados provenientes da execução dos passos de cada instância de processo definido, tais como tempo, esforço, custo, horário e progresso, defeito de produto/problemas, recurso, etc.

Cada processo do Repositório de Métricas é vinculado a um papel de usuário, desta forma realizando-se o controle de acesso sobre os dados armazenados. No momento em que o usuário executa um processo, dados previamente especificados são coletados e armazenados na base de dados reais. A evolução das métricas coletadas depende de sua especificação na base de dados realizados.

A proposta deste trabalho tem como objetivo principal prover uma forma de coleta de métricas a partir da execução de processos. Contudo, não demonstra preocupação em [OLI99] com a forma na qual consultas devem ser realizadas e como seus resultados serão disponibilizados aos usuários. Ainda, apresenta como trabalho futuro a integração das bases de dados com a instrumentalização da coleção de métricas derivadas.

Nenhuma evolução deste trabalho foi encontrada na literatura. O Repositório de Métricas mostra-se útil por centralizar todos os dados resultantes da execução de processos. Porém, a falta de mecanismos para a consulta aos dados armazenados é uma limitação crítica.

5.2.3 Repositório Multidimensional de Medidas

O Repositório Multidimensional de Medidas (RMM) [PAL03] tem como intuito refletir visões compartilhadas da organização sobre análise de tendências e acompanhamento de processos organizacionais. O RMM utiliza-se de *Data Warehousing* e disponibiliza consultas sobre os dados através de cubos OLAP. Este repositório é flexível, permitindo a adoção de diferentes medidas sobre seus processos, dando suporte à organização ao longo da evolução de sua maturidade.

O RMM considera as contantes alterações sofridas por organizações em seus ambientes e em seus dados através da utilização do modelo de qualidade CMMI. Em ambientes dinâmicos, repositórios de medidas estáticos tornam-se inadequados. Neste sentido, o RMM disponibiliza uma base de dados genérica com alto nível de flexibilidade [PAL03].

A flexibilidade provida pelo repositório se dá através do desenvolvimento de metadados, que provêm o armazenamento de definições de medidas e as relações entre elas. Também, o conjunto de relações entre entidades é definido e armazenado como uma nova entidade no repositório, a fim de apoiar tanto a visão hierárquica quanto a visão multidimensional sobre os dados. O diagrama de classes do modelo de dados do RMM é representado através da Figura 9, extraída de [PAL03].

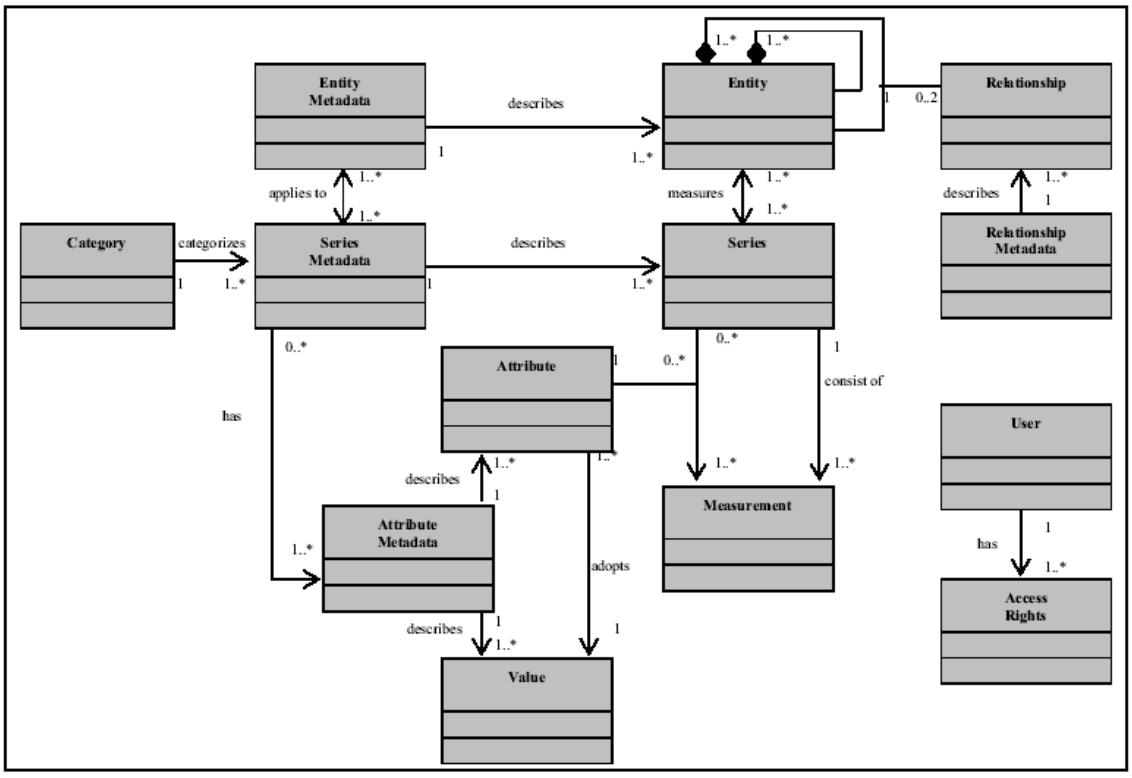

Figura 9: Entidades e relacionamentos que compõem o RMM.

A disponibilização de informação aos usuários finais é realizada através de um portal WEB. Este oferece uma lista de relatórios pré-definidos. Novos relatórios devem ser derivados a partir de linguagem SQL ou através dos cubos OLAP, nem sempre familiares a todos os usuários da organização.

A arquitetura do RMM, ilustrada na Figura 10 (extraída de [PAL03]) é composta basicamente de:

- Gerenciador de indicadores e tendências, o qual apresenta a informação baseada em relatórios e gráficos pré-definidos;
- Capacidades analíticas e *drill-down/drill-up*, que foram projetadas para apoiar os gerentes e a equipe de desenvolvimento com relatórios dinâmicos, exportação para planilhas eletrônicas e funcionalidades de *drill-down/drill-up*;
- Gerência e controle de qualidade, que permite definir novas medidas, conceder privilégios e realizar auditorias sobre a qualidade dos dados do sistema;
- Mecanismos analíticos (OLAP), que provêem a capacidade de computar medidas derivadas e agregá-las por múltiplas dimensões;

§ Repositório de Medidas, ou seja, repositório multidimensional propriamente dito; e

§ Coleta manual e automática de dados.

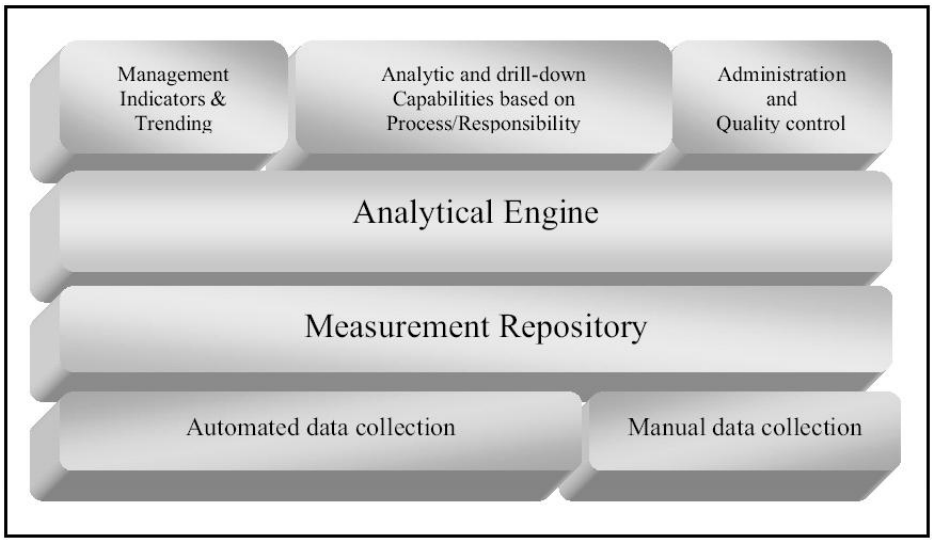

Figura 10: Arquitetura RMM.

A inserção de dados no RMM é realizada de duas formas: automatizada e manual. Contudo este assunto não é detalhado em [PAL03]. A primeira repousa sobre mecanismos de ETC, e é destinada a grandes volumes de dados. A coleta manual é realizada através da WEB, e é voltada a pequenos volumes. Como trabalho futuro, o autor aponta o desenvolvimento de uma interface para coletar dados de bancos de dados de sistemas legados diretamente.

Esta proposta de repositório de dados diferencia-se das duas apresentadas anteriormente por prover que métricas podem ser analisadas através das diferentes granularidades que compõem o modelo analítico do RMM. Também provê uma maior flexibilidade quanto às métricas armazenadas e sua evolução, através do suporte de metadados.

Contudo, o material encontrado, não apresenta como a flexibilidade provida pelo modelo do RMM garante a homogeneidade dos dados dos diferentes projetos e a manutenção da consistência ao longo da evolução do PM. Quanto à disponibilização da informação aos usuários finais, este também deixa a desejar por não considerar os diferentes perfis de usuário que poder-se-ão utilizar-se destes dados. Seus recursos analíticos apóiam-se quase que somente em relatórios OLAP e planilhas eletrônicas, o que dificulta o acesso de usuários não conhecedores destes recursos. O RMM foi desenvolvido considerando as necessidades da

Enterprise Performance Unit da Ericsson Research Canadá, no entanto, não são disponibilizados os resultados alcançados pela implantação deste projeto.

5.3 *Business Performance Management* (BPM)

Propostas de repositórios como as apresentadas na Seção 5.2 proporcionam uma estrutura unificada para o armazenamento dos dados resultantes de processos e/ou projetos de software executados em organizações. Porém, seus recursos de análise sobre os dados armazenados mostram-se aquém do esperado em termos de análise e monitoramento de processos. Propostas de *Business Performance Management* (BPM), ou gerenciamento do desempenho de negócios [GOL04] focam em qualquer processo de negócio e assemelham-se as propostas anteriores por serem centradas em um repositório, normalmente uma base analítica, contudo estas enfocam na qualidade da informação disponibilizada ao usuário final.

O BPM baseia-se em *Data Warehousing* e na proposta de monitoramento de processos e/ou projetos em execução, mas possuem como diferencial a preocupação de como os dados armazenados poderão ser úteis aos usuários organizacionais. O BPM tem com presuposto garantir que as estratégias do nível gerencial e operacional estejam de acordo com as estratégias do modelo tático [GOL04]. Para tal, integra uma tecnologia chamada *Business Activity Monitoring* (BAM), monitoramento de atividades de negócios [HEL02], que enfatiza na diminuição do tempo entre a ocorrência de um evento no ambiente transacional e a tomada de decisão no nível estratégico organizacional em virtude do evento ocorrido.

A proposta do BPM está voltada às estratégias da organização, como ilustrado na Figura 11, extraída de [GOL04]. Nesta proposta os indicadores representam as estratégias e metas da organização (indicadores alvo) e o desempenho atual da organização (indicadores reais). As ações e decisões do nível tático e operacional e a estratégia organizacional são decorrentes dos valores apresentados nos indicadores de desempenho organizacional.

A realidade organizacional vivida por grande parte das organizações atualmente demanda cortes de custo, estratégias para aumento do lucro e a necessidade de reagir rapidamente. A análise dos dados armazenados nas propostas apresentadas anteriormente demanda tempo, hoje cada vez mais valorizado, também a realidade do desenvolvimento de produtos dentro das organizações está cada vez mais voltado a processos e seu gerenciamento voltado a medições.

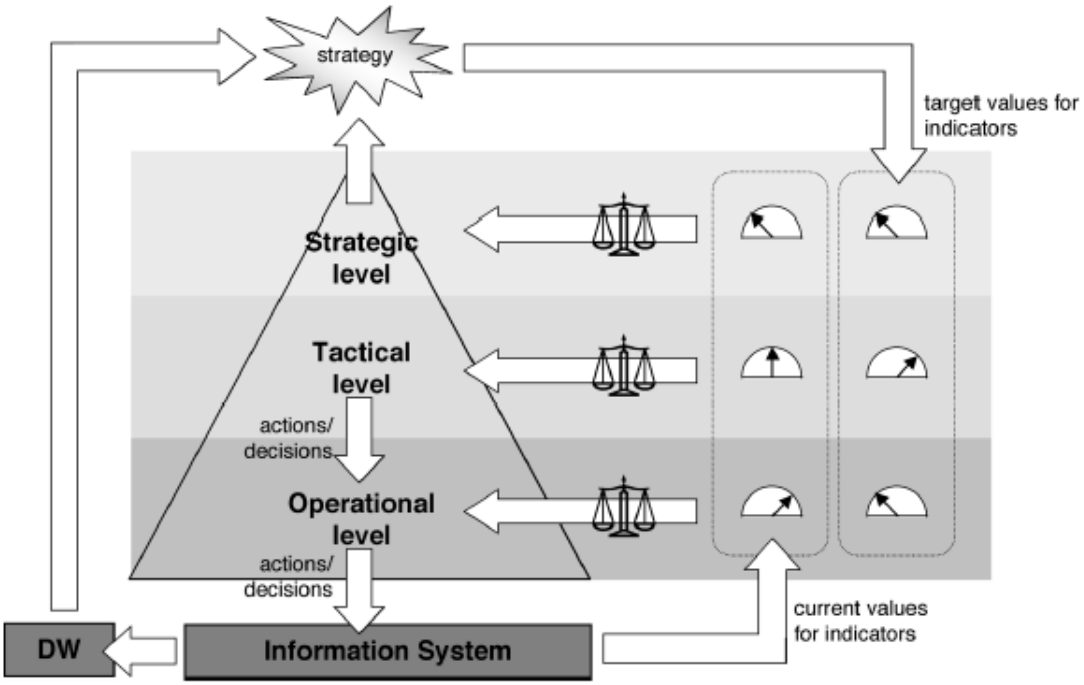

Figura 11: Proposta de BPM.

Visando atender a esta realidade, [GOL04] propõe a arquitetura de BPM ilustrada pela Figura 12. O lado esquerdo da arquitetura proposta é composto de um DW clássico, uma ferramenta de ETL que extrai, transforma e limpa dados oriundos do ambiente transacional, um DSA denominado ODS que integra estes dados no DW e ferramentas OLAP que disponibilizam estes dados. Já, o lado direito da arquitetura proposta visa à tecnologia BAM, para tal constitui os seguintes componentes:

- *Right-Time Integrator* – RTI, responsável por integrar em tempo correto os dados do ambiente transacional, do DW, do EAI (Enterprise Application Integration), sistema de integração de aplicação organizacional e dos dados atuais (*data streams*);
- *Dynamic Data Store* – DDS, base de dados que armazena dados por um curto período de tempo para que estes sejam utilizados por ferramentas de mineração e regras de inferência;
- *KPI manager*, gerenciador dos indicadores organizacionais necessários aos diferentes níveis de *dashboards* e relatórios gerados;
- Conjunto de ferramentas de mineração responsáveis por extrair padrões relevantes quanto aos dados do ambiente transacional;

§ *Rule Engine*, regras de inferência que ininterruptamente monitoram os eventos filtrados pelo RTI ou descobertos pelas ferramentas de mineração, disponibilizando aos usuários alertas em tempo correto para que possam interagir e tomar as ações necessárias.

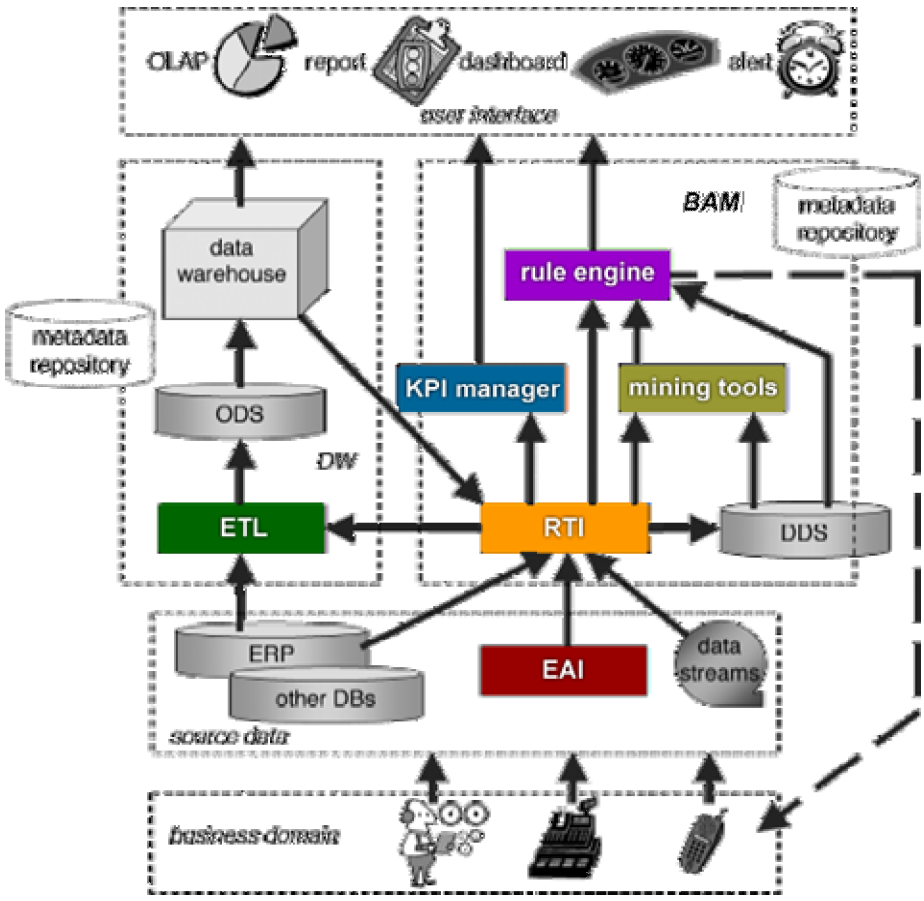

Figura 12: Arquitetura - BPM.

A informação deve ser disponibilizada ao usuário no momento correto para que este consiga tomar ações no sentido de reverter um comportamento não esperado. Neste sentido, enfatiza-se a necessidade de reduzir o tempo entre a integração de dados oriundos do ambiente transacional, do DW e sua análise [GOL04].

Analisar os dados no tempo correto implica diminuir o tempo entre a ocorrência de um evento no ambiente transacional e a tomada de ação necessária da organização, já que o valor empresarial de uma ação pode diminuir à medida que o tempo passa. A Figura 13, extraída de [MAR03], representa os componentes que estão presentes no intervalo entre a ocorrência de um evento e a tomada de ação correspondente [HAC03]. Estes se encontram brevemente descritos abaixo.

1) Intervalo de dados: é o tempo entre a ocorrência do evento e os dados serem armazenados e disponibilizados para análise;

2) Intervalo de análise: o tempo entre a disponibilidade dos dados para análise e a geração de informação através da análise deste. Nesta fase são aplicados métricas e indicadores de qualidade sobre dados;

3) Intervalo de decisão: é o tempo entre disponibilizar a informação à pessoa responsável e esta absorvê-la e responder de forma apropriada, tomando decisões.

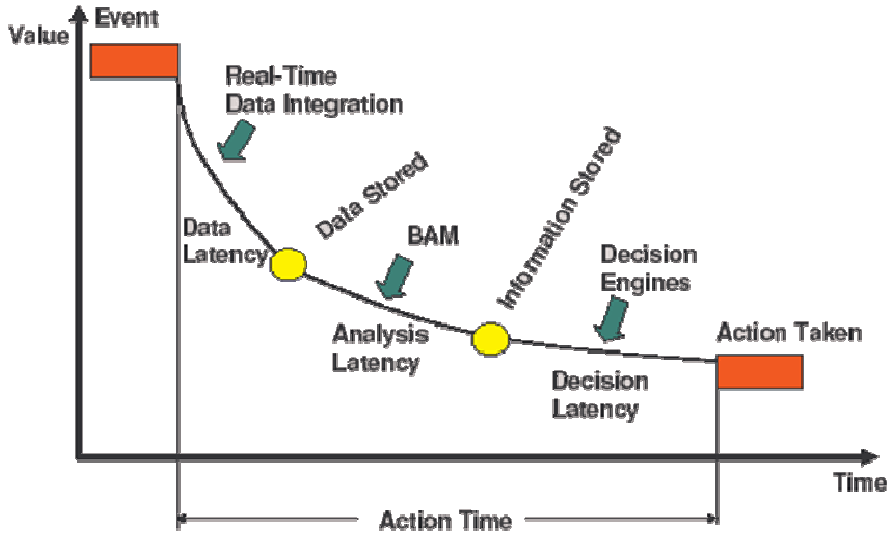

Figura 13: Relação Tempo X Tomada de ação.

Contudo, além do intervalo entre a ocorrência e a disponibilização da informação, também é de suma importância a forma ou qualidade de como a informação é disponibilizada aos usuários. A qualidade da informação torna-se importante devido ao tempo de interpretação que esta pode consumir, o que inclui precisão de distribuição, contexto e formato da informação. Um exemplo de informação de difícil interpretação, onde poucos usuários entenderiam, é dado pelo alerta: “Alarme = 423, Dados = 419, 1630, 18 e Ação = 17”. Um alerta pode ocorrer sempre que identificado algum evento no ambiente transacional, este deve propagar a informação que descreve o evento ocorrido a todos os usuários a que este possa interessar [MAR03]. Assim, em contrapartida a mensagem que segue poderia facilmente ser interpretada: “Uma ordem remessa foi processada sem um registro de faturamento. Pare a remessa. Clique aqui para informação mais detalhada”.

Quanto à qualidade do alerta disparado ao usuário, deve-se considerar que ao ocorrerem eventos não esperados, este deve utilizar-se do canal de comunicação que mais rapidamente possa ser acessado pelo usuário alvo para tomar uma decisão. Alguns exemplos

de canais de comunicação que podem ser utilizados para disparar alertas são: SMS, mensagem instantânea, e-mail, sinal sonoro, entre outras.

A utilização de uma estratégia voltada à medição e a preocupação em prover informação de qualidade exigem que o desempenho dos processos sejam mensurados continuamente e comparados através de indicadores de qualidade e outros recursos analíticos, como *dashboards* [GOL04].

Os *dashboards* proporcionam aos usuários um quadro visual do que está acontecendo na organização, através de uma única interface o usuário visualiza todas as métricas de interesse da organização. Paralelamente ao *dashboard*, o perfil do usuário (desenvolvedor, líder, gerente, etc.) determina a informação que este deve ter acesso de acordo com o papel que exercido na organização.

Os indicadores de qualidade comunicam tendências e metas específicas de uma organização. Devem-se definir indicadores que satisfaçam a todos os perfis de usuários da organização. A definição dos indicadores pode estar baseada em dados históricos, relacionamentos entre indicadores ou modelos de predição. Indicadores de qualidade podem ser disponibilizados através de *dashboards* ou de alertas disparados aos usuários.

5.3.1 *Business Process Intelligence* (BPI)

Uma proposta de BPM pode ser encontrada em [CAS04] e [GRI04]. Estes apresentam um conjunto de conceitos e uma arquitetura de implementação denominados *Business Process Intelligence* (BPI). O BPI tem como objetivo prover a seus usuários mecanismos de análise e previsibilidade sobre execuções de quaisquer processos de negócio.

A Figura 14, extraída de [GRI04], apresenta a arquitetura do BPI. Esta é composta basicamente de três componentes chaves: PDW *Loader*, *Process Mining Engine* (PME) e *Business Process Cockpit* (BPC).

O *PDW Loader* é responsável pela extração de dados de logs de execução de processos de negócio, limpeza, cálculo de métricas e carga no DW de processos (*Process Data Warehouse* - PDW). Já o PME através de técnicas de de mineração de dados aplicadas sobre os dados armazenados no PDW oferece mecanismos para análise e predição sobre os processos de negócio. As consultas sobre o PDW podem ser realizadas utilizando ferramentas OLAP convencionais ou através de uma interface dedicada denominada BPC que tem por

objetivo oferecer suporte a usuários organizacionais na análise, monitoramento e gerenciamento de seus processos de negócios [GRI04].

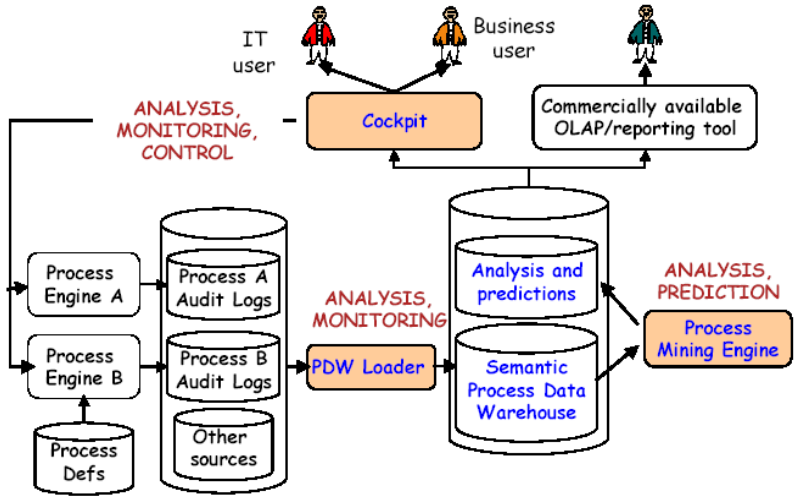

Figura 14: Arquitetura - BPI.

O PDW baseia-se no modelo multidimensional representado pela Figura 15, extraída de [GRI04]. As tabelas do PDW estão organizadas através de um esquema de constelação de fatos. Neste, alterações do estado de processos, serviços e nodos são armazenados em tabelas fatos, enquanto que as definições de processos, de serviços, de nodos de dados, recursos e comportamentos, são dimensões sob as quais os fatos podem ser analisados. Também, o PDW contém um conjunto de agregados de informação que descrevem o desempenho de métricas, como por exemplo, eficiência de um recurso. Mais detalhes a respeito deste modelo analítico podem ser encontrados em [GRI04].

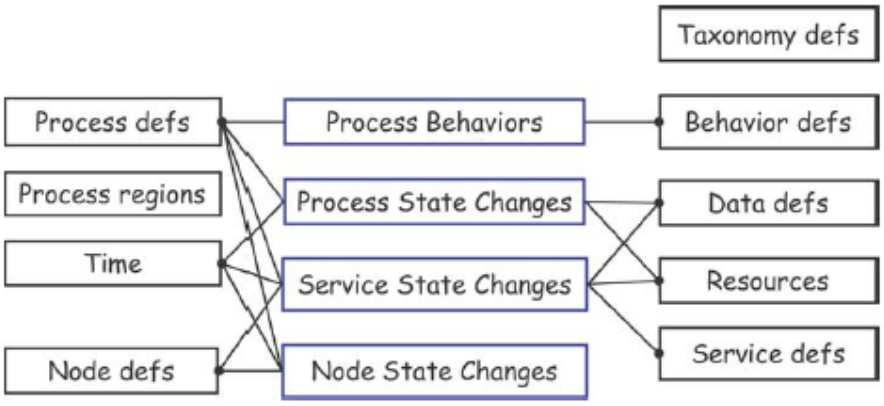

Figura 15: Modelo Multidimensional - PDW.

Como mencionado, os dados armazenados no PDW podem ser analisados através de ferramentas OLAP comerciais ou através do BPC. O BPC tem como meta atingir a todos os usuários organizacionais. Preocupa-se em disponibilizar uma interface simples que não limite a flexibilidade e as funcionalidades previstas pela ferramenta. Prover a análise e monitoramento do negócio envolve desenvolver técnicas que permitam ao usuário definir, monitorar e mensurar a qualidade do negócio através de métricas [GRI04]. Neste sentido, o BPC:

- Disponibiliza uma varidade de relatórios, preocupando-se com a semântica dos dados através de conceitos de visualização e técnicas especificamente projetadas para exibir processos de negócio em execução;
- Permite monitorar processos, serviços, recursos e outras entidades relacionadas a processos informando aos usuários seu desempenho atual em relação à qualidade esperada, e notificando-o em casos de desvios identificados;
- Permite administrar processos correntes ajustando o processo com parâmetros de configuração do sistema (e.g. prioridade de processo) e notificar eventos do processo.

O BPC permite visualizar os dados da execução de processos através de métricas sob diferentes perspectivas. Uma perspectiva identifica a entidade de processo que é o foco da análise. Por exemplo, sobre a perspectiva de serviço, o usuário poderia visualizar métricas e estatísticas sobre *web services* invocados durante a execução de processos de negócios. Segundo [GRI04], as seguintes perspectivas são pertinentes à análise em nível de negócio: sistema (como um todo), processo, recurso e serviço. A interface disponibilizada pelo BPC pode ser visualizada através da Figura 16, extraída de [GRI04].

Cada uma das perspectivas definidas foca em um objetivo específico e facilitam a análise apresentando algumas informações estatísticas básicas (e.g. tempo de execução e desempenho), informações correspondentes a valores de processo (e.g. rendas e custos) e informações a respeito do comportamento do processo.

Já, quanto à semântica provida ao processo de negócio que permite ao usuário analisar o comportamento deste, é atribuída pelo BPI através de três conceitos: comportamento, taxonomia e região de processo.

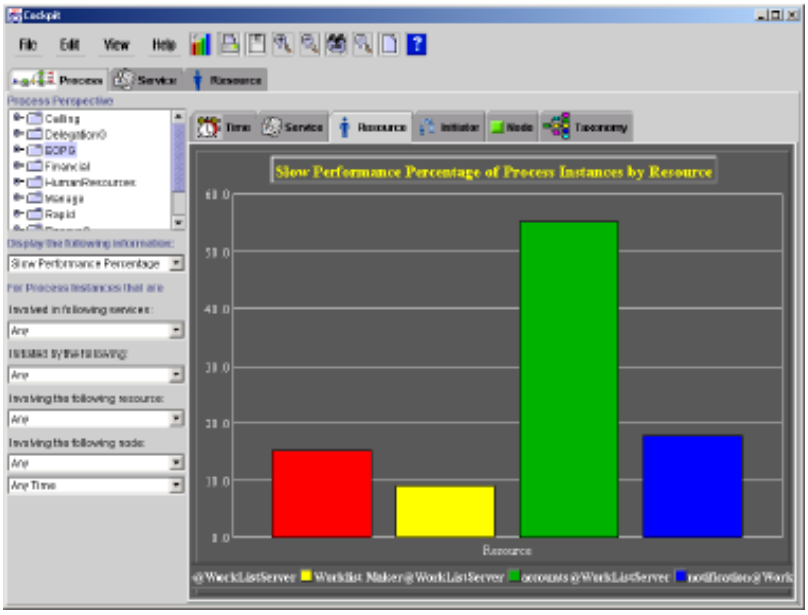

Figura 16: Interface BPC.

§ Comportamentos: habilitam a identificação de instâncias processo que possuem características do interesse do analista, possivelmente por corresponder a uma alta ou baixa qualidade. Por exemplo, processos que duram mais que 10 dias ou processos que entram em *loop*. O BPI possui uma biblioteca de modelos de comportamentos implementados como visões SQL, estes são detectados através de valores *booleanos* e vinculados a mecanismos de alertas definidos pelo BPI;

§ Taxonomias: classificam processos segundo às características definidas pelo usuário, cada taxonomia pode possuir diversas categorias. As categorias representam os comportamentos definidos que permitem analisar os processos e então entender suas causas. Por exemplo, identificação de comportamentos de processos que foram executados em mais de N dias.

§ Regiões de processo: definem regiões do processo identificando um nodo inicial e um ou um conjunto de nodos finais de um determinado processo.

A evolução do BPI é apresentada em [CAS05] como *Intelligent Business Operation Management* (iBOM). O iBOM é uma plataforma que permite o gerenciamento automatizado, inteligente, orientado a processo, bem como a optimização do processo baseada nas metas organizacionais. O iBOM provê uma visão da organização e execução de processos, identificando desvios e predizendo anomalias, além de propor melhorias aos processos de negócios.

Como adicional ao BPI, o iBOM define uma série de componentes que se comprometem a estratégia organizacional provendo a organização a análise e o monitoramento de processos através de métricas que podem ser definidas por usuários associados a processos e disponibilizadas através de relatórios.

5.4 Considerações

Diferentes alternativas vêm sendo utilizadas por organizações para analisar e monitorar seus PDSs. Este capítulo explorou três diferentes alternativas. A primeira, planilha eletrônica, é uma alternativa que não requer grande investimento financeiro contudo, deixa a desejar quanto à análise dos dados disponibilizados. A segunda, os repositórios proporcionam à organizações a centralização de dados resultantes do PDS e provêem uma visão consolidada sobre estes. Contudo, mas em sua maioria o acesso disponibilizado a estes dados por parte dos usuários organizacionais normalmente se dá de forma bastante restrita. E finalmente, a terceira alternativa, a utilização do BPM focado em qualquer processo de negócio, representa uma tendência de projetos de DW diferenciando-se dos demais por preocupar-se com o monitoramento e a apresentação dos dados aos usuários finais. Ao final apresentou-se uma proposta de projeto de BPM com todas suas características de extração, modelagem e apresentação dos dados aos usuários finais.

6 ARQUITETURA DE *DATA WAREHOUSING* PARA ACOMPANHAMENTO DE PDS

Este capítulo apresenta uma arquitetura de Data Warehousing na qual este trabalho encontra-se inserido. Apresenta também as decisões de projeto que nortearam o desenvolvimento desta arquitetura.

Diante da realidade organizacional apresentada no capítulo 4 e do reconhecimento da necessidade de uma infra-estrutura de apoio a um PM como requisito básico do SW-CMM nível 3, é proposto para este estudo de caso um ambiente de *Data Warehousing* para apoio à análise sobre o desempenho de projetos concluídos e o monitoramento de projetos em andamento através do PM. Uma versão preliminar desta pesquisa foi apresentada em [RUI05].

Dentro deste ambiente, o presente trabalho é responsável pela especificação, implementação e modelagem de dois aspectos:

a) Um DW voltado ao armazenamento de métricas e/ou dados necessários ao cálculo de métricas do PDS, considerando o PM definido na organização alvo;

b) Uma camada de apresentação que provê recursos de análise de projetos concluídos.

A seguir apresenta-se a arquitetura proposta para este ambiente de *Data Warehousing*, juntamente com uma breve descrição de seus componentes.

6.1 Arquitetura de *Data Warehousing*

O ambiente de *Data Warehousing* proposto tem por objetivo a extração de dados oriundos do ambiente transacional do PDS, o armazenamento e a sua disponibilização aos usuários organizacionais através de recursos de análise e de monitoramento, se enquadrando nas novas tendências de *Data Warehousing* [GOL04]. A arquitetura proposta para este ambiente é apresentada na Figura 17.

Esta arquitetura é composta de 6 componentes básicos: componente de integração de aplicações, componente de integração de dados, componente repositório, componente de monitoramento, componente de análise e componente de apresentação. O presente trabalho é responsável especificamente pelo componente repositório, componente análise e componente

de apresentação. Os componentes de integração de aplicações e de dados são detalhados em [CUN05].

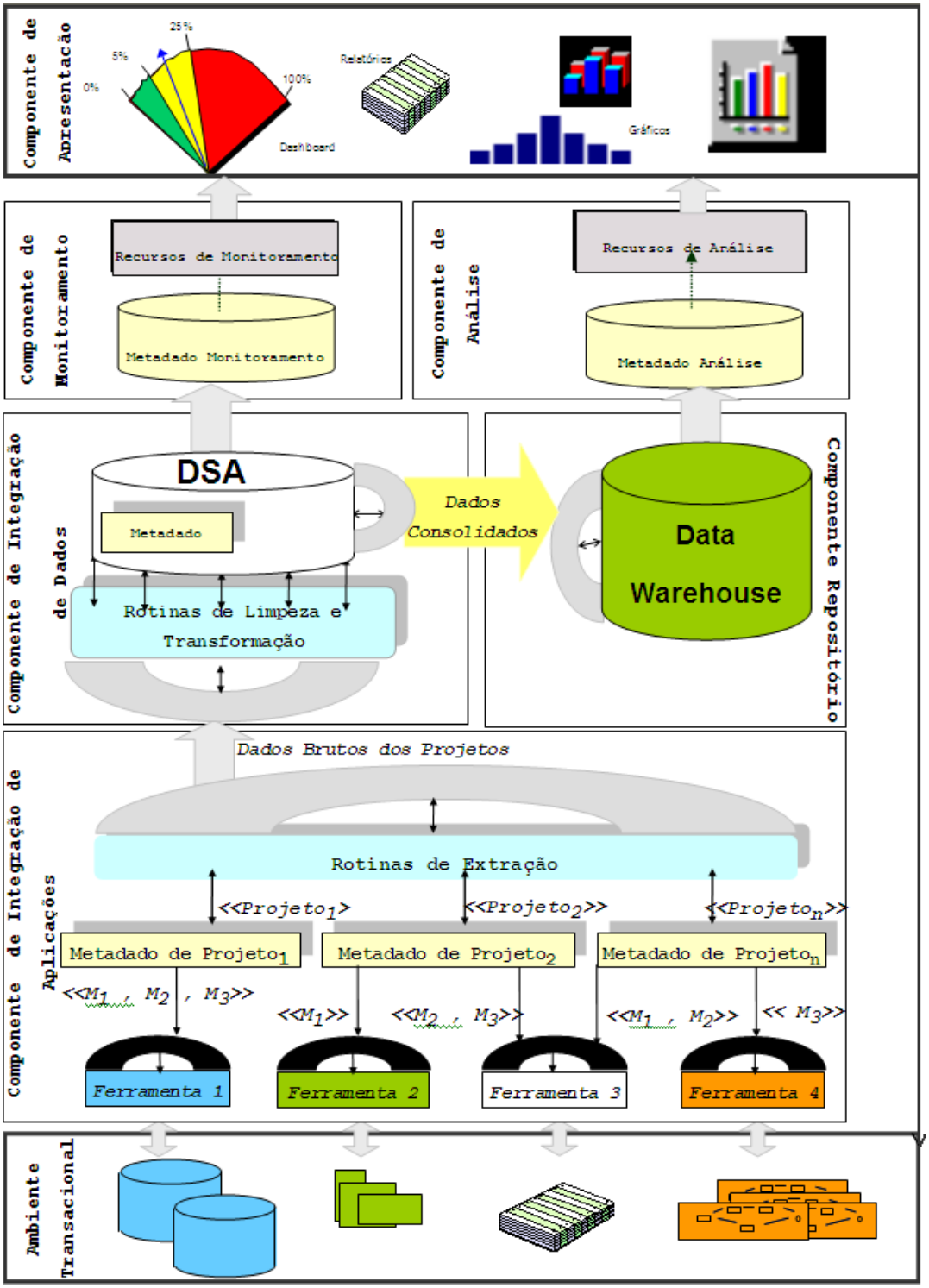

Figura 17: Arquitetura proposta para um ambiente de Data Warehousing.

O componente de integração de aplicações é responsável por extrair os dados brutos dos projetos provenientes das diversas ferramentas e carregá-los no *Data Staging Area* (DSA). Neste ambiente, adotou-se uma abordagem de baixa intrusão seguindo um padrão arquitetural orientado a serviços [ALO04], onde os serviços atuam como *wrappers*. Cada *wrapper* aborda a extração de dados considerando uma ferramenta em particular, com foco no modelo de dados proprietário desta. Além disso, cada um dos projetos de software é descrito por metadados, expressos em XML *Schema,* que parametrizam a implementação dos *wrappers*. Os metadados de projeto definem as ferramentas adotadas e como os dados requeridos para o PM (métricas e atributos das dimensões) são armazenados nessas ferramentas, de acordo com as características próprias de cada projeto. As rotinas de extração exploram os metadados de projeto para localizar o *wrapper* correto e guiar a extração baseada no mapeamento estabelecido entre o dado bruto e o dado requerido [CUN05].

No componente integração de dados, os dados extraidos das ferramentas do ambiente transacional são limpos e transformados no DSA, através das rotinas de limpeza e transformação, auxiliadas pelos metadados organizacionais. Após esse processo, os dados consolidados são carregados no DW. O DW integra estes dados resultantes do PDS e consolidados através do DSA, visando a disponibilização de informação que auxilie a organização no acompanhamento quantitativo de seu PDS através de um PM. Este componente também adota uma abordagem orientada a serviço.

O modelo de dados analítico que guia o processo de ETC representado pelos componentes de integração de aplicações e de dados é uma das contribuições do presente trabalho, sendo discutido com detalhes no Capítulo 7.

A camada de apresentação do ambiente de *Data Warehousing* é formada pelos componentes de análise, monitoramento e apresentação. Os componentes de análise e de apresentação desenvolvidos no contexto deste trabalho são detalhados no Capítulo 8. O desenvolvimento destes considera os requisitos e restrições de análise dos possíveis perfis de usuário encontrados na organização, como discutido na Seção 4.5.

O componente de análise permite que o usuário acompanhe os dados históricos de projetos já concluídos na organização. Para tal, este utiliza-se de dados de projetos armazenados no DW, metadados e recursos analíticos. Os metadados de análise são utilizados para construir as consultas necessárias ao cálculo das métricas sem esforço dos usuários. Para tal, o usuário interage com o componente de apresentação fornecendo os parâmetros de sua consulta e visualizando os resultados.

O componente de monitoramento tem como objetivo principal prover ao usuário o monitoramento de projetos ainda em andamento permitindo que este rapidamente detecte desvios de desempenho e tome ações apropriadas. Neste sentido, o componente de monitoramento utiliza-se de dados consolidados oriundos do DSA, metadados e recursos analíticos. O monitoramento utiliza-se de dados oriundo do DSA por estes possuírem menor latência de dados do que os dados do DW. Isto é, o intervalo de tempo entre a extração dos dados do ambiente transacional e o armazenamento no DSA é menor do que para o armazenamento no DW. Neste sentido, o monitoramento com tais dados provê visões mais próximas à realidade de um determinado momento. Os metadados facilitam o cálculo das métricas voltadas ao monitoramento de projetos e posteriormente recursos analíticos que visam facilitar o monitoramento de dados são disponibilizados através da camada de apresentação.

O componente de apresentação proposto, pode ser dividido em dois módulos distintos, um que visa à análise do PDS organizacional e outro que visa o monitoramento deste. Os dois módulos diferenciam-se quanto às opções disponibilizadas ao usuário, entre as métricas e os parâmetros de consultas. O componente de análise suporta, através de uma infra-estrutura, cálculos de métricas sobre diferentes tipos de dados armazenados do ambiente transacional, diferenciando-se de ferramentas tradicionais de OLAP. O componente de análise que compõe a camada de apresentação do ambiente de *Data Warehousing* é detalhado no Capítulo 8.

A implantação desta arquitetura em uma organização provê uma infra-estrutura que apóia desde a extração de dados do ambiente transacional até sua apresentação através de uma camada de apresentação. Considerando a heterogenidade do ambiente do PDS, as funcionalidades descritas mostraram que é possível uma proposta de baixa intrusão que facilite a análise e o monitoramento do PDS através de métricas.

A integração dos componentes propostos torna a organização capaz de identificar comportamentos inadequados presentes no PDS e fornece dados que apóiam a tomada de decisão baseada em fatos verídicos e confiáveis, por serem extraídos automaticamente do ambiente transacional, sem esforço extra. Assim, esta infra-estrutura possibilita ao usuário um tempo de tomada de decisão mais eficaz provendo a este a utilização de argumentos apoiados em fatos. A agilidade e a precisão da informação é considerada grande diferencial nas organizações atuais. Resultados preliminares desta proposta podem ser encontrados em [RUI05].

7 MODELO ANALÍTICO PARA ANÁLISE DE PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

Este capítulo apresenta um modelo multidimensional, parte de um projeto de Data Warehousing, que visa o armazenamento e acompanhamento de dados oriundos de PDSs através de métricas. Apresenta também as decisões de projeto necessárias que levaram ao modelo analítico final.

O modelo analítico apresentado neste Capítulo integra o projeto do ambiente de *Data Warehousing* apresentado no Capítulo 6. Ele é a base de um DW que tem por objetivo prover um repositório de dados unificado e centralizado, que permita a análise dos dados resultantes de todos os projetos de desenvolvimento de uma organização através de um PM organizacional.

O desenvolvimento deste modelo analítico considerou os requisitos de uma organização de sofware SW-CMM nível 2 em busca do SW-CMM nível 3, como especificado no estudo de caso apresentado no Capítulo 4. A próxima seção apresenta algumas questões que se fizeram necessárias para o desenvolvimento do projeto do modelo analítico em seguida, apresenta detalhadamente o modelo analítico resultante, juntamente com suas tabelas dimensões, tabelas fatos e demais características.

7.1 Projeto - Modelo Analítico

Segundo a metodologia seguida, (seção 3.4), a construção de um modelo analítico deve iniciar pela identificação dos assuntos que serão contemplados pelo DW e que são do interesse da organização em questão, juntamente com suas informações quantitativas. Estes assuntos devem ser identificado junto aos possíveis usuários do DW, desta forma garantindo o comprometimento dos mesmos com o projeto a ser desenvolvido.

No desenvolvimento do modelo analítico incialmente considerou-se os perfis de usuários da organização que utilizar-se-ão do DW desenvolvido, os quais foram apresentados na Tabela 5 (Capítulo 4). Após, identificou-se os assuntos que circundavam suas necessidades de análise quanto ao PDS e ao PM definidos pela organização.

A Figura 18 apresenta os fatores que determinam a estrutura do modelo analítico desenvolvido. O desesenvolvimento do modelo analítico norteou-se na realidade do ambiente organizacional em questão. Nesta, as necessidades de análise organizacional sobre o PDS são

representadas por diferentes perspectivas de análise, áreas de qualidade, identificadas através do PM adotado e os diferentes perfis de usuários encontrados em uma organização. Uma vez que as necessidades de análises requeridas sobre o PDS visam a análise de dados e métricas através de estruturas de ciclos de vida adotadas por projetos (*e.g.* esforço por fase, tamanho por versão, etc) propõe-se que o modelo analítico reflita esta. Também, deve-se considerar as limitações quanto à disponibilidade dos dados, determinadas pelas fontes de informações. Identificados estes fatores do ambiente (necessidade de análise organizacional, estrutura dos projetos e disponibilidade dos dados), o suporte da organização quanto às limitações encontradas (*e.g.* não padronização de unidades de medidas entre projetos, adoção de diferentes nomenclaturas para fases, etc) determinará o modelo analítico resultante.

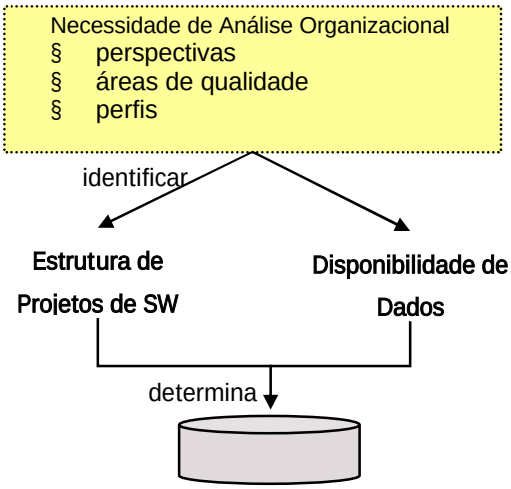

Figura 18: Fatores que guiaram o desenvolvimento do modelo analítico.

Quanto à necessidade de analisar diferentes aspectos sobre o PDS e de visualizar as métricas organizacionais sob as diferentes AQ identificou-se a necessidade de analisar o PDS quanto a características que estão presentes em projetos, versões, fases, iterações, defeitos e atividades. Por exemplo, quanto às atividades identificou-se a necessidade de analisá-las por tipo (trabalho, retrabalho, revisão e qualidade); já os defeitos devem ser analisados quanto a sua severidade (alta, média e baixa), tipo (interno, externo ou melhoria) e fase de origem. Um levantamento detalhado quanto às necessidades relativas às métricas na estrutura de PDS encontra-se em [NOV04a].

A estrutura do ciclo de vida adotado pelos projetos de SW é de suma importância para o desenvolvimento do modelo analítico uma vez que este deverá refleti-lá visando os objetivos de análise organizacional. Na Seção 2.1 apresentou-se diversas características de PDS, entre elas as diversas estruturas e formatos de ciclo de vida possíveis de serem adotados.

Assim, considerando o estudo de caso, foram analisados na organização em questão projetos desenvolvidos no período de agosto de 2004 março de 2006, verificando-se que estrutura dos ciclos de vida adotados pelos projetos possuem poucas variações. Os projetos são organizados em versões de software, cujo desenvolvimento pode ou não estar organizado em iterações. Se estiver organizado em iterações, cada uma destas é constituída de diversas fases. Caso não possuam iterações, o desenvolvimento de versão está dividido diretamente em fases. Em ambos casos, as fases resultam na execução de diversas atividades. A estrutura de projeto pode ser visualizada através da Figura 19. Assim, conclui-se que nesta organização deve-se dar suporte tanto a ciclos de vida seqüenciais quanto a iterativos.

A disponibilidade dos dados é determinada pela identificação das fontes de dados utilizadas no processo de ETC. Durante esta investigação das fontes de dados de cada uma das necessidades informacionais se detectou que os projetos não possuem uniformidade quanto ao armazenamento de um mesmo dado, o que certamente tornou mais oneroso o processo de extração dos dados. As fontes de dados identificadas e consideradas pelo ETC são apresentadas na Tabela 4 e um relato detalhado sobre todos os problemas de heterogeneidade encontrados é desenvolvido em [CUN05].

Alguns do problemas encontrados quanto aos dados necessários foram:

- não adoção de *baselines* por alguns projetos;
- falta de padronização quanto à nomenclatura de fases e tipos de atividades, bem como para classificação e categorização de defeitos;
- dados coletados em diferentes níveis de abstração (*e.g.* estimativas em nível de fase ou de atividade) nos projetos;
- termos como projeto e versão em alguns projetos usados como sinônimos;
- mesmo dado coletado em diferentes unidades (*e.g.* tamanho em KLOC, *Use Case Points*, Pontos de Função, etc).

O cruzamento entre as informações levantadas sobre a necessidade de análise organizacional, estruturas de projetos, adotadas e disponibilidade dos dados são de suma importância ao posterior desenvolvimento do modelo analítico, uma vez que determinam a granularidade do modelo a ser desenvolvido.

Neste sentido, quanto a granularidade, devido a não existência de estimativas em nível de atividade em alguns projetos a organização optou por não prover estimativas por atividade.

Provê-la acarretaria em grande esforço a projetos que não as tinham. Contudo, quanto às limitações de nomenclatura de fases, tipos de atividades e a classificação e categorização de defeitos que igualmente poderia comprometer a granularidade e qualidade dos dados disponibilizados pelo modelo, recebeu-se o suporte organizacional no sentido da adoção de uma padronização deste dados através dos projetos da organização.

Outra questão a ser analisada quanto ao modelo analítico desenvolvido é a forma como este comporta o PM. O PM adotado, Tabela 3, é constituído em sua maioria por métricas derivadas representadas por funções não aditivas, mais especificamente razões. Como explanado na Seção 3.2, valores não aditivos impõem restrições sérias quanto à forma como podem ser sumarizadas para produzir resultados corretos em OLAP. A fim de se resolver este problema, conforme sugerido por [KIM98], tomou-se a decisão de armazenar somente as métricas base nas tabelas fato do modelo analítico, isto é métricas bases que constituem métricas derivadas (*e.g.* esforço realizado, esforço – BO, etc). Desta forma, o DW possibilita a realização de uma maior variedade de consultas sobre os fatos do modelo, através de *drills* que exploram os fatos em diferentes granularidades. Assim, as métricas que representam funções não aditivas são extraídas do DW através de consultas SQL, usando os recursos da camada de apresentação. Novas métricas também podem ser derivadas a partir das métricas aditivas armazenadas.

A Figura 19 apresenta a estrutura de projeto comportada pelo o ambiente de *Data Warehousing.* Esta estrutura de projeto considera as características de projeto e métricas armazenadas, no ambiente transacional, por cada um de seus componentes. Assim, segundo o ambiente transacional projetos de software podem ser compostos de uma ou mais versões. As versões de projetos podem estar relacionadas a iterações ou fases, na Figura 19 representada por “Etapa”. Fases, estão relacionadas a atividades. As atividades estão vinculadas a um nome, ao esforço realizado (ER) e as realizações de cronograma tanto em projetos iterativos como seqüenciais. Já, as estimativas de esforço e as estimativas de cronograma são armazenadas ao longo das fases iterações.

Os defeitos no ambiente transacional são armazenados sempre relacionados a uma classificação de sua categoria, severidade, tipo e fase de origem, tanto em projetos seqüenciais como em projetos iterativos. Em projetos iterativos, estes também estão relacionados à iteração em que foram originados.

As versões de projetos são vinculadas no ambiente transacional ao nome da versão, tamanho e custo estimados e realizados, requisitos e a satisfação do cliente. Não há

estimativas para requisitos, defeitos e satisfação de cliente; os valores armazenados representam realizações. As versões armazenam estas informações tanto em projetos iterativos como seqüenciais.

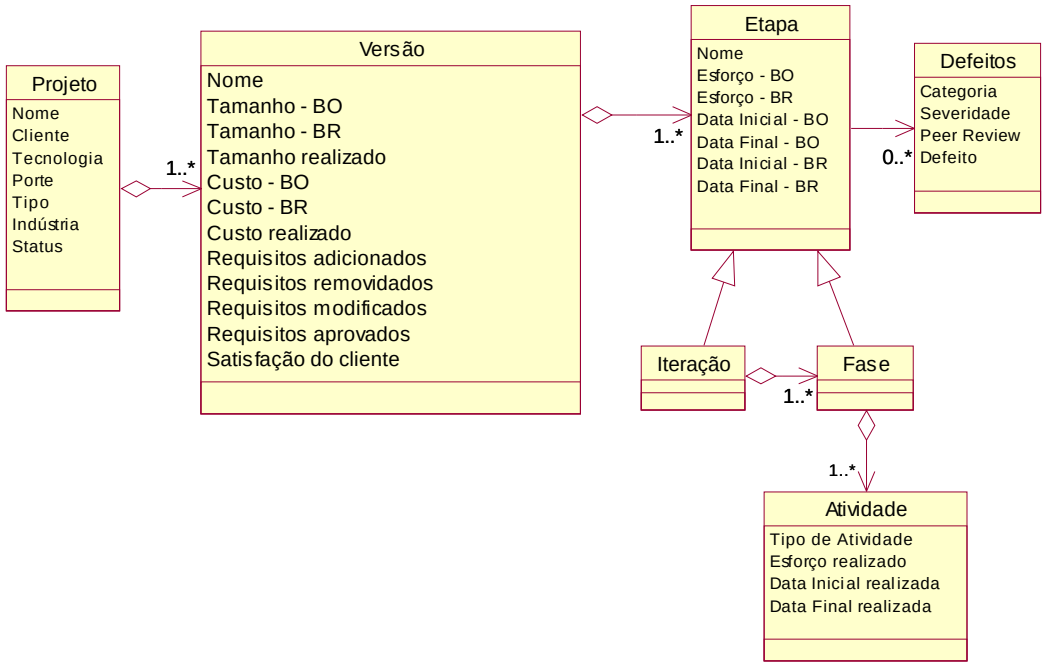

Figura 19: Estrutura de projeto considerada pelo modelo analítico.

7.2 Modelagem - Modelo Analítico

As tabelas do modelo analítico representam um esquema do tipo constelação de fatos. As tabelas fato, descritas sucintamente na Tabela 6, representam as métricas de projetos em diferentes níveis de granularidade, sendo que a informação de menor granularidade prevista pelo modelo, é representada pela Fato_Atividade e a de maior granularidade pela Fato_Release.

Tabela 6: Fatos do Modelo Analítico.

Nome do Fato	Descrição do Fato
Fato_Release	Armazena as métricas de requisitos, satisfação do cliente, tamanho e custo estimados e reais de uma versão de projeto.
Fato_Iteracao	Armazena as métricas de esforço e duração estimados e reais de uma iteração.
Fato_Defeito	Armazena os defeitos de uma versão em determinada fases de iterações.
Fato_Fase	Armazena as métricas de esforço e duração estimados e reais de uma fase.
Fato_Atividade	Armazena as métricas de esforço realizado de determinado tipo de atividade (i.e. Trabalho, Retrabalho, Revisão e Qualidade).

As tabelas dimensão descrevem as características de projeto, versão, iteração, fase, atividade, defeito, entre outras. A Tabela 7 nomeia e descreve sucintamente cada uma das dimensões do modelo.

Tabela 7: Dimensões do Modelo Analítico.

Nome da Dimensão	Descrição da Dimensão
Dim_Projeto	Armazena nome de projetos.
Dim_Porte_Projeto	Armazena porte de projetos.
Dim_Tecnologia	Armazena tecnologias de projetos.
Dim_Industria	Armazena o tipo de indústria de projetos. (Ex. finanças, governos e outras).
Dim_Tipo_Projeto	Armazena tipos de projeto. (Ex. manutenção, desenvolvimento entre outros)
Dim_Cliente	Armazena os nomes dos clientes de projetos.
Dim_Release	Armazena os nomes das versões de projetos.
Dim_Iteração	Armazena os nomes das iterações de versões.
Dim_Fase	Armazena os nomes das fases de versões.
Dim_Atividade	Armazena os tipos de atividades (Ex. Trabalho, Retrabalho, Revisão e Qualidade) de uma fase.
Dim_Defeito	Armazena as informações referentes a defeitos. Estes são armazenados por categoria (interno ou externo) , severidade (baixa, média ou alta) e peer review.
Dim_Status	Armazena *status* de versões como em desenvolvimento ou concluídas.
Dim_Tempo	Armazena datas inicial e final de atividade, fase, iterações e versões (data, ano, mês, dia e semestre).
Dim_Tipo_Fato	Armazena um identificador que determina se um fato é uma estimativa (baseline original, baseline revisado) ou o registro de uma realização.

Visando uma melhor apresentação do modelo analítico desenvolvido, este é representado através de seus fatos. As figuras 20, 21, 22, 23 e 24 apresentam respectivamente os fatos: Fato_Release, Fato_Defeito, Fato_Iteracao, Fato_Fase, e Fato_Atividade com suas métricas e dimensões relacionadas.

O Fato_Release (Figura 20) armazena os atributos de tamanho, custo, duração, requisitos e a satisfação do cliente. O armazenamento destes no Fato_Release deve-se à granularidade na qual cada uma destas informações é armazenada no ambiente transacional. As dimensões são definidas considerando-se as diferentes necessidades de análise dos usuários. Assim, verificou-se que estes desejam analisar estes atributos por versões ou projetos possivelmente, ainda focando em algumas características específicas de projetos. Quanto a projetos, as propriedades foram distribuídas em diversas dimensões, já que não existe uma relação de ordem entre elas, o que é assumido entre atributos de uma dimensão em tecnologia OLAP.

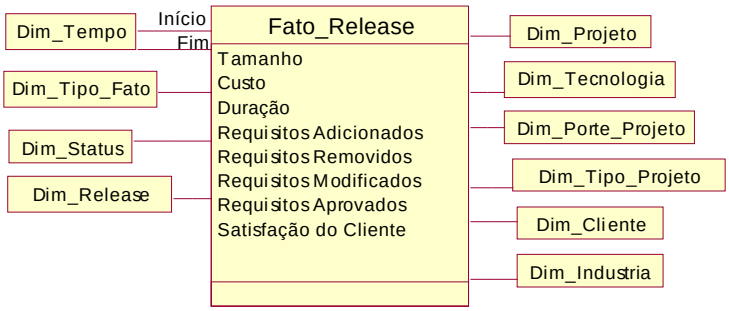

Figura 20: Modelo Analítico – Fato_Release.

Fato_Release relaciona-se com a dimensão Dim_Tipo_Fato, a qual determina se um fato é uma estimativa ou uma realização. As métricas de tamanho, custo e duração possuem valores estimados e reais. Já as métricas que correspondem a requisitos e satisfação do cliente somente possuem valores realizados uma vez que não há estimativas sobre estes.

Como já comentado anteriormente, os defeitos estão relacionados à fase na qual foram detectados tanto em projetos iterativos como seqüenciais. Assim, a tabela Fato_Defeito (Figura 21) encontra-se relacionada às dimensões fase e iteração, além de todas as dimensões que representam as propriedades do projeto e de versão (Dim_Release). Assume-se sempre que projetos seqüenciais possuem uma única iteração.

A organização alvo não realiza estimativas para defeitos, somente armazena-se o número de defeitos encontrados, junto com a respectiva caracterização do defeito. Devido a esta característica a tabela Fato_Defeito não se encontra relacionada à tabela Dim_Tipo_Fato. Portanto, as consultas sobre defeitos podem explorar as características atribuídas ao defeito isto é, categoria (defeito interno, defeito externo ou melhoria), severidade (alta, média ou baixa) e *peer review* ou defeito de teste e ainda as propriedades sobre projetos, versões, fases onde se originaram e iterações.

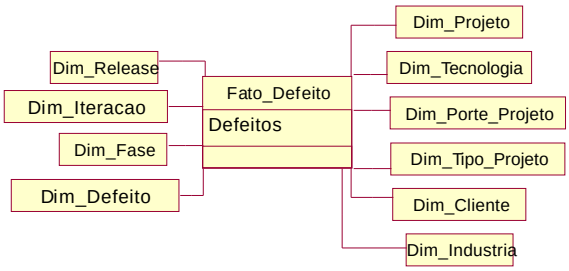

Figura 21:Modelo Analítico - Fato_Defeito.

Fato_Iteracao (Figura 22) armazena o esforço e a duração estimados e reais de iterações. Portanto, o Fato_Iteraçao também encontra-se relacionado à dimensão que determina valores estimados e realizados, Dim_Tipo_Fato. As durações (estimadas e realizadas) são calculadas através das datas armazenadas nas iterações. Já, o esforço representa o esforço acumulado do nível de atividade no ambiente transacional. Através deste fato é possível consultar o esforço e a duração de projetos iterativos através das diferentes características de projetos, versões e iterações.

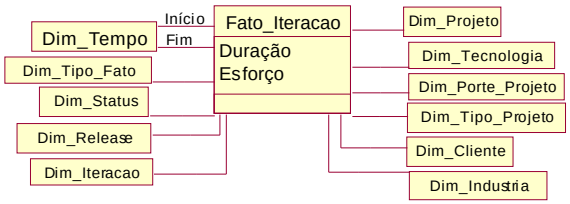

Figura 22: Modelo Analítico - Fato_Iteracao.

Fato_Fase (Figura 23) armazena esforço e as durações estimados e reais de fases de projetos. As durações (estimadas e realizadas) são calculadas através das datas armazenadas por fases. Já, o esforço representa o esforço acumulado do nível de atividade no ambiente transacional. Por prover dados estimados e reais, este fato encontra-se relacionados a Dim_Tipo_Fato.

Através deste fato é possível consultar o esfoço e duração através das diferentes características de projetos, versões e fases providas pelo modelo. Contudo, a dimensão Dim_Iteracao que disponibiliza dados por iterações a projetos iterativos não é relacionada ao Fato_Fase. Esta decisão se fez necessária devido ao fato de uma fase poder se repetir entre as iterações de um projeto. Assim, ao se consultar a variação de uma determinada fase que pode estar em diversas iterações resultados errôneos poderiam ser retornados, uma vez que o desenvolvimento de uma iteração nem sempre ocorre de forma seqüencial.

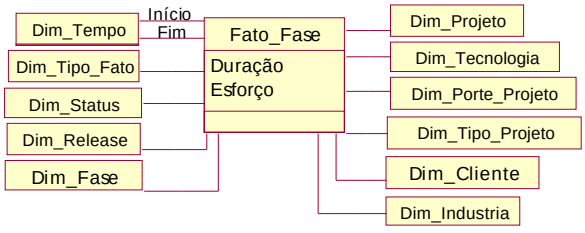

Figura 23: Modelo Analítico - Fato_Fase.

Fato_Atividade (Figura 24) armazena o esforço realizado de fases e iterações de versões, tanto de projetos seqüenciais como iterativos. Por não armazenar estimativas, este fato não se relaciona com a dimensão Dim_Tipo_Fato. Na dimensão Dim_Atividade, as atividades são classificadas em: trabalho, retrabalho, revisão e qualidade. As consultas sobre o esforço realizado por tipo de atividade podem ser realizadas tanto em projetos seqüenciais como iterativos, uma vez que o esforço realizado é coletado em ambos tipos de projetos no nível de atividade e relacionado a suas fases e iterações de origem. Assim, como com Fato_Defeito, assume-se que projetos seqüenciais são compostos de uma única iteração. Consultas podem ser realizadas através das diferentes características de projetos, iterações, fases e tipos de atividades providas pelo modelo.

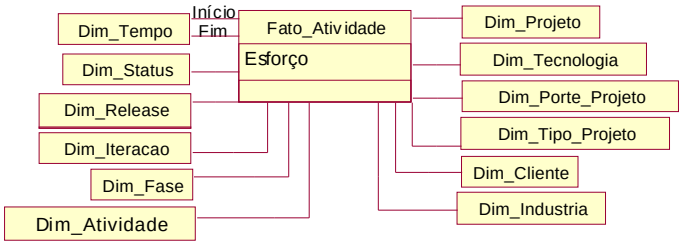

Figura 24: Modelo Analítico - Fato_Atividade.

7.3 Implementação

O modelo analítico foi implementado através de um DW desenvolvido utilizando o SGBD SQL Server 2000 da Microsoft. Cada uma das tabelas que compõem o DW resultante juntamente com seus atributos, tipos e número de registros armazenados pode ser analisada através do Anexo I. Versões preliminares deste modelo foram relatadas em [NOV05] e [RUI05].

7.4 Considerações

Neste capítulo foram apresentadas as decisões que guiaram o desenvolvimento do modelo analítico, salientando-se cada aspecto da modelagem do DW. Entre estas podem ser citadas: a definição inicial dos requisitos de análise que deveriam ser disponibilizados, o levantamento de possíveis tabelas dimensões e fatos, a forma como as métricas seriam armazenadas nas tabelas fato do mesmo, entre outras.

Após, discorreu-se sobre os diversos aspectos do desenvolvimento do modelo analítico, bem como as informações armazenadas em cada uma das tabelas fatos e dimensões.

A utilização do modelo analíto apresentado provê à organização a centralização de informações oriundas do PDS e a análise voltada às métricas definidas.

Apesar dos diversos benefícios providos pela implantação deste modelo nota-se que a falta de uma camada de apresentação que considere as restrições tecnológicas dos usuários da organização influência no uso do DW. Acredita-se que a adoção de uma camada de apresentação com recursos analíticos que facilite a análise do PDS quanto às métricas organizacionais tornaria o DW em questão uma ferramenta de considerável apoio à análise dos projetos da organização.

Neste sentido, o próximo capítulo apresenta a proposta de uma camada de apresentação para este DW munida de diversos recursos analíticos baseados nas necessidades de todos os possíveis usuários de um DW. Além da análise de dados do DW, a camada de apresentação também provê monitoramentos de dados.

8 RECURSOS ANALÍTICOS PARA UMA CAMADA DE APRESENTAÇÃO

Este capítulo descreve alguns aspectos de implementação em produção na organização alvo do estudo de caso. Após, relata a experência da adoção de um ambiente de Data Warehousing para análise do PDS através de seu Programa de Métricas, ressaltando os benefícios percebidos.

O mercado de software provê diversas ferramentas voltadas à análise de dados amazenados em DW através da tecnologia OLAP como Oracle [ORA05], Microsoft Analysis Services [MIC05], Cosmos [COS05], entre outras. Contudo, estas normalmente desconsideram várias das características prioritárias à usabilidade de camadas de apresentações, tais como os diferentes perfis de usuários que poderão utilizar-se dos dados analíticos.

Este trabalho propõe uma camada de apresentação que suporta a análise e o monitoramento de PDS, cujos recursos analíticos são voltados às especificidades do PDS. Esta proposta visa-se atender às seguintes questões:

- Ferramentas OLAP oferecem total suporte à análise de dados aditivos apresentando limitações em dados semi ou não aditivos. Como pode ser observado na Tabela 3, o PM organizacional é composto em sua maioria de métricas calculadas através de razões. Sendo assim, se calculadas através de recursos OLAP (sumarizações) produzirão resultados incorretos;
- Os mecanismos OLAP são genéricos demais, e requerem um certo conhecimento técnico. Nem todos os usuários detêm este conhecimento, ou sentem-se motivados, ou até mesmo não dispõem de tempo para aplicá-los, a fim de elaborar análises rotineiras sobre o PDS, ou investigar causas para possíveis desvios de desempenho de processos;
- A informação deve ser apresentada tanto de forma detalhada, através de tabelas e gráficos, como de forma a permitir o reconhecimento imediato da situação dos projetos na organização frente aos indicadores de qualidade estabelecidos.

Visando sanar deficiências das ferramentas OLAP no cenário de um ambiente de *Data Warehousing* voltado à análise de PDS, a camada de apresentação proposta tem como princípios oferecer recursos analíticos adequados aos diferentes perfis de análise encontrados em organizações, as quais consideram as especificidades de análise e monitoramento do PDS, provêem maior semântica sobre os dados disponibilizados e conseqüentemente, demandam um menor tempo do usuário. Os recursos analíticos considerados pela camada de apresentação foram identificados através do documento [NOV04a].

O ambiente de *Data Warehousing* proposto no Capítulo 6 possui três componentes responsáveis pela apresentação dos dados ao usuário final: componentes de análise, de monitoramento e o componente de apresentação. Especificamente, este trabalho abordou o desenvolvimento dois destes, a saber, o componente de análise e o componente de apresentação. Neste sentido, as próximas seções descrevem respectivamente uma visão geral do componente de apresentação proposto e após, o componente de análise juntamente com cada um de seus recursos analíticos.

8.1 Componente de Apresentação

A Figura 25 apresenta a interface do componente de apresentação. Este possui como recursos principais: perspectivas de análise, áreas de qualidade e de métricas, filtros temporais e área de visualização.

No modelo analítico, as tabelas fato quantificam métricas base de projetos, enquanto que as dimensões qualificam estes dados. No contexto desta interface, as consultas realizadas pelo usuário são compostas pelas restrições de qualificação de dados representadas através das perspectivas e filtros temporais, bem como das restrições quantitativas representadas na interface através de uma lista de métricas que determinam quais as formas como os dados qualitativos serão calculados.

As perspectivas de análise representam o modelo analítico através de um componente visual em forma de árvore. As áreas de qualidade (AQ) são definidas segundo a abrangência do PM adotado, agrupando métricas relacionadas, as quais são mostradas na área de métricas conforme a AQ selecionada. Os filtros temporais permitem que as análises disponibilizadas sejam filtradas por intervalo de tempo. A área de visualização disponiliza o resultado da consulta realizada pelo usuário através de componentes gráficos.

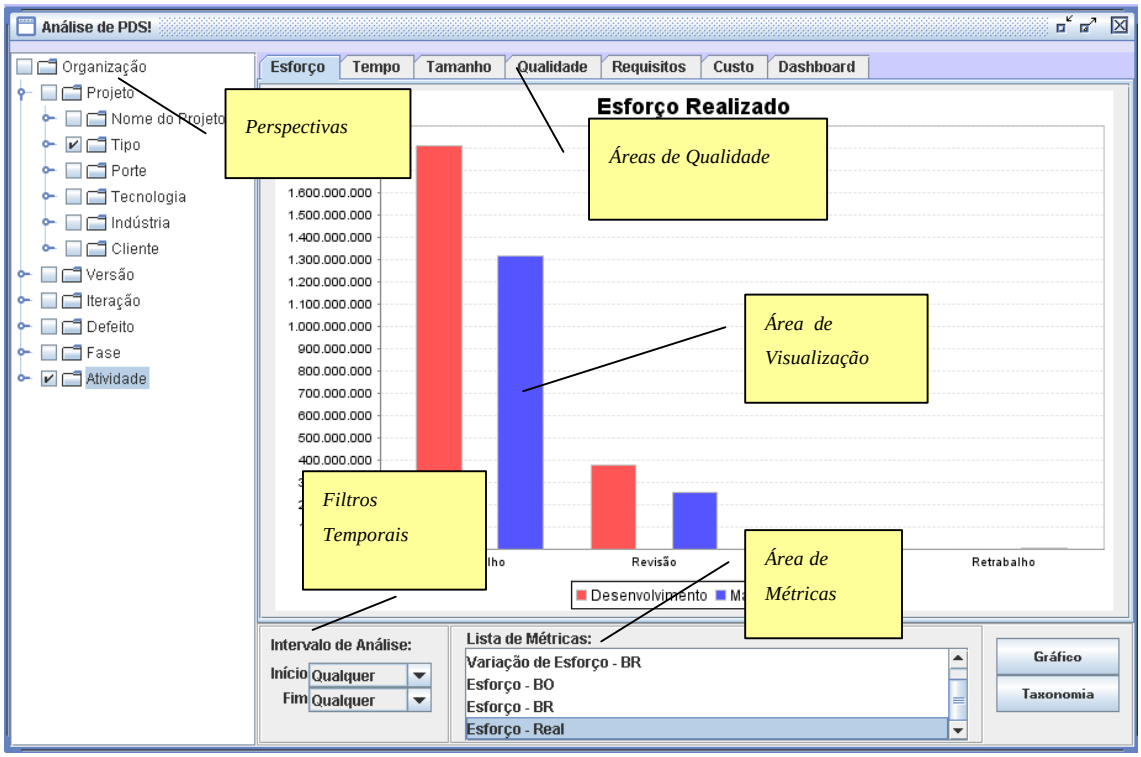

Figura 25: Interface - Componente de Apresentação.

8.1.1 Restrições Qualitativas : Perspectivas

As perspectivas refletem os diferentes níveis de detalhe nos quais os dados sobre o PDS podem ser analisados. Estes níveis refletem a estrutura do projeto definida considerada pelo DW, correspondendo em sua maioria, às dimensões do modelo analítico.

As perspectivas são apresentadas através de uma árvore, a qual organiza hierarquicamente, e em um único componente, a estrutura do projeto em suas diferentes granularidades, as respectivas dimensões associadas, bem como atributos e respectivos valores. A Tabela 8 representa a relação entre as perspectivas do componente de apresentação e as dimensões encontradas no modelo analítico, a perspectiva organização foi incluída devido à necessidade organizacional de se comparar diferentes projetos da organização.

Para realizar as restrições qualitativas, o usuário deve selecionar os nodos da árvore que representam as dimensões e atributos desejados. A Figura 26 apresenta três diferentes momentos de interação do usuário com as perspectivas. O primeiro (Figura 26a) representa como esta é disponibilizada inicialmente ao usuário, apresentando a estrutura de análise do PDS. A segunda (Figura 26b) apresenta todas as possíveis qualificações dentro desta estrutura em termos de atributos e dimensões. Finalmente a terceira (Figura 26c) apresenta alguns

valores oriundos do DW, os quais podem ser usados para restringir as consultas do usuário. Através da seleção ilustrada na Figura 26c, a consulta define interesse na análise de projetos desenvolvidos utilizando tecnologia Java, sendo que o nome dos projetos é a propriedade utilizada para apresentar o resultado.

Tabela 8: Perspectiva e dimensão.

	Perspectiva	Dimensão
Organização	Organização	
Projeto	Projeto	Dim_Projeto
	Nome Projeto	
	Tipo	Dim_Tipo_Projeto
	Porte	Dim_Porte_Projeto
	Tecnologia	Dim_Tecnologia
	Indústria	Dim_Industria
	Cliente	Dim_Cliente
Versão	Versão	Dim_Release
	Nome da Versão	
Iteração	Iteração	Dim_Iteração
	Nome da Iteração	
Defeito	Defeito	Dim_Defeito
	Categoria	
	Severidade	
	Peer Review	
Fase	Fase	Dim_Fase
	Nome da Fase	
Atividade	Atividade	Dim_Atividade
	Tipo Atividade	

8.1.2 Restriçoes Quantitativas: Áreas de Qualidade e de Métricas

As Áreas de Qualidade (AQ), como apresentado na Seção 2.3, representam as diferentes expectativas e/ou necessidades de análise da organização. As AQ facilitam a análise realizada pelo usuário, uma vez que este nem sempre conhece quais as métricas que estão incluídas no PM.

Ao selecionar uma AQ, o componente de apresentação automaticamente mostra na Área de Métricas as métricas do PM relacionadas àquela AQ. A Figura 27 apresenta dois exemplos de AQ juntamente com as métricas disponibilizadas ao usuário. O usuário deve estabelecer sua restrição quantitativa selecionando primeiro uma AQ, e depois, uma ou mais métricas da Área de Métricas.

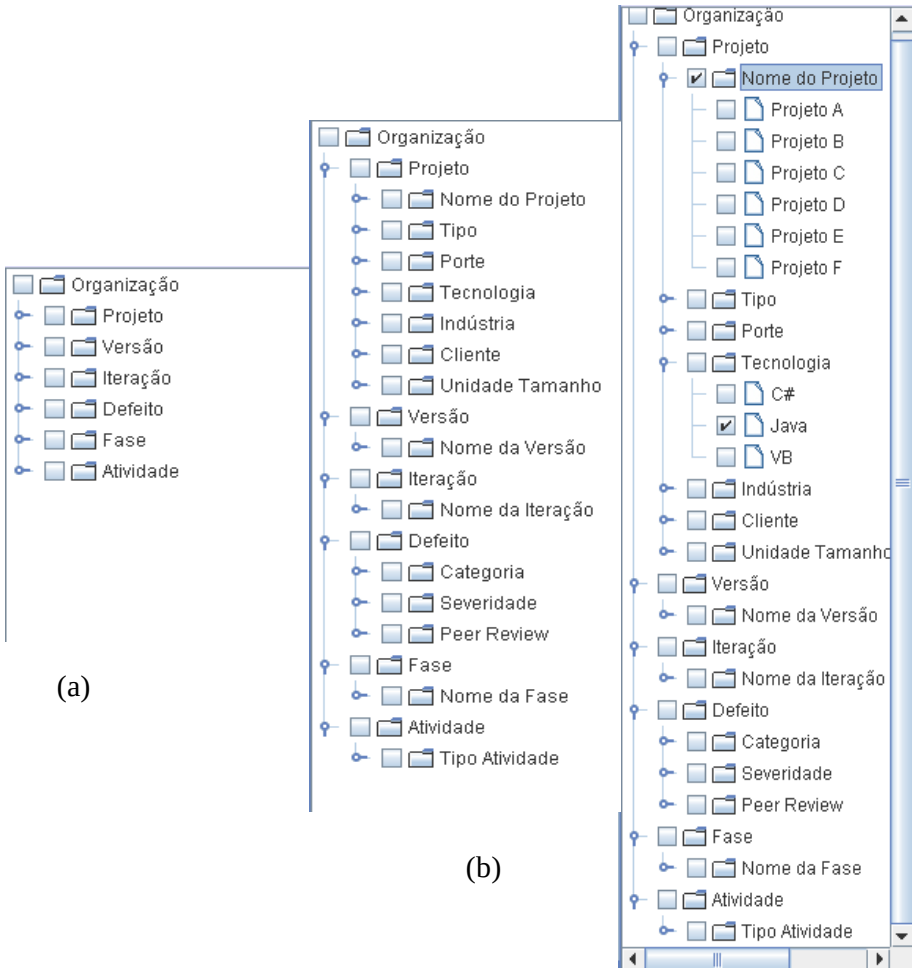

Figura 26: Perspectivas.

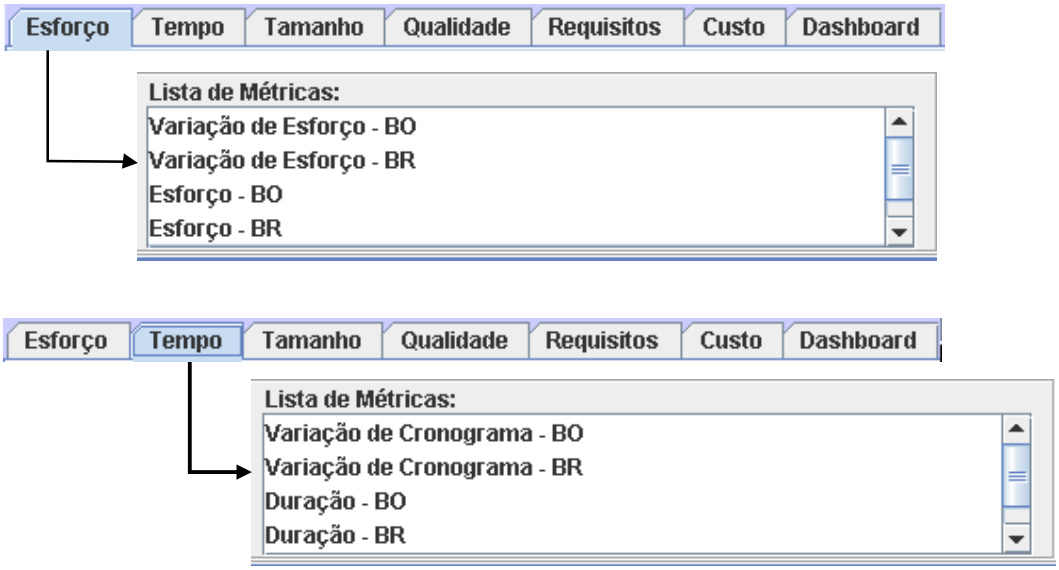

Figura 27: Áreas de Qualidade e Métricas disponibilizadas.

Observa-se na Figura 27 que o último elemento gráfico da AQ denomina-se *Dashboard*. Este apresenta várias métricas relacionadas a distintas AQ, conforme seus indicadores. O *Dashboard* é apresentado com detalhes na Seção 8.1.4.3.

A Tabela 9 apresenta a distribuição das métricas pelas diversas AQ previstas. Neste estudo de caso, as métricas disponibilizadas na Área de Métricas (Tabela 9), agrupadas por AQ, correspondem a todas as métricas identificadas em uma organização (Tabela 3). Adicionalmente, foram incluídas outras métricas base, muitas das quais são utilizadas no cálculo das métricas derivadas. Por exemplo, Esforço BO e Esforço Real são utilizadas para cálculo de variação de Esforço BO, e Tamanho Real e Esforço Real são utilizados no cálculo da produtividade organizacional. Estas métricas foram incluídas devido ao interesse demonstrado pela organização em sua análise.

Tabela 9: Métricas e Áreas de Qualidade.

Áreas de Qualidade	Métricas
Esforço	Variação de Esforço Baseline Original Variação de Esforço Baseline Revisado Esforço Baseline Original Esforço Baseline Revisado Esforço Real Produtividade
Tempo	Variação de Cronograma Baseline Original Variação de Cronograma Baseline Revisado Duração Baseline Original Duração Baseline Revisado Duração Real
Tamanho	Variação de Tamanho Baseline Original Variação de Tamanho Baseline Revisado Tamanho Baseline Original Tamanho Baseline Revisado Tamanho Real
Qualidade	Eficiência de Remoção de defeitos Densidade de Defeitos Entregues Densidade de Defeitos Internos Densidade de Defeitos Eficiência de Revisão Defeitos Entregues Defeitos Internos Defeitos de Peer Review Satisfação do Cliente
Requisitos	Volatilidade de Requisitos Requisitos adicionados Requisitos removidos Requisitos modificados Requisitos aprovados pelo cliente
Custo	Variação de Custo Baseline Original Variação de Custo Baseline Revisado Custo Baseline Original Custo Baseline Revisado Custo Real Custo da Qualidade

A vinculação das AQs aos perfis de usuários de uma organização mostra-se outro recurso analítico interessante. Através desta torna-se possível restringir análises de AQ de acordo com o perfil de usário que está realizando a análise. Um exemplo é o perfil liderança de projeto, o qual não pode ter acesso à AQ custo. Contudo, esta possibilidade não foi explicitamente explorada neste trabalho.

8.1.3 Restrições Temporais: Filtros Temporais

Os filtros temporais restringem a seleção qualitativa (perpectivas) e quantitativa (AQ e métricas) do usuário de acordo com o tempo no qual as versões de projetos transcorreram. Para tal, busca-se as datas iniciais e/ou finais de todas as versões de projetos.

Selecionando somente um tempo inicial no componente de apresentação, a consulta considerará somente versões iniciadas a partir do período inicial definido; selecionando somente um tempo final, serão selecionadas versões que estiveram ativas até o período final definido. E, se o usuário selecionar um período inicial e um final, serão selecionadas as versões que estiveram ativas durante o intervalo de tempo definido.

A Figura 28 exemplifica uma consulta com restrição temporal. Nesta, o usuário deseja analisar a Variação de Esforço - BR de versões de projetos desenvolvidos utilizando a tecnologia Java, referente ao período de Abril/2004 a Julho/2005.

8.1.4 Opções de Visualização

Dashboards, taxonomias e gráficos são utilizados para facilitar a análise da informação disponibilizada ao usuário. Embora todas estas opções apresentem informações de forma gráfica, estas são voltadas a expectativas de análises bastante diferenciadas.

8.1.4.1 *Gráficos*

Os gráficos disponibilizam valores correspondentes a uma ou mais métricas segundo as seleções do usuário. Por exemplo, o gráfico ilustrado na

Figura 29 apresenta a Variação de Esforço - BR de versões de projetos desenvolvidas em Java. No exemplo, através deste recurso consegue-se analisar quantitativamente a percentagem de Variação de Esforço – BR de cada projeto.

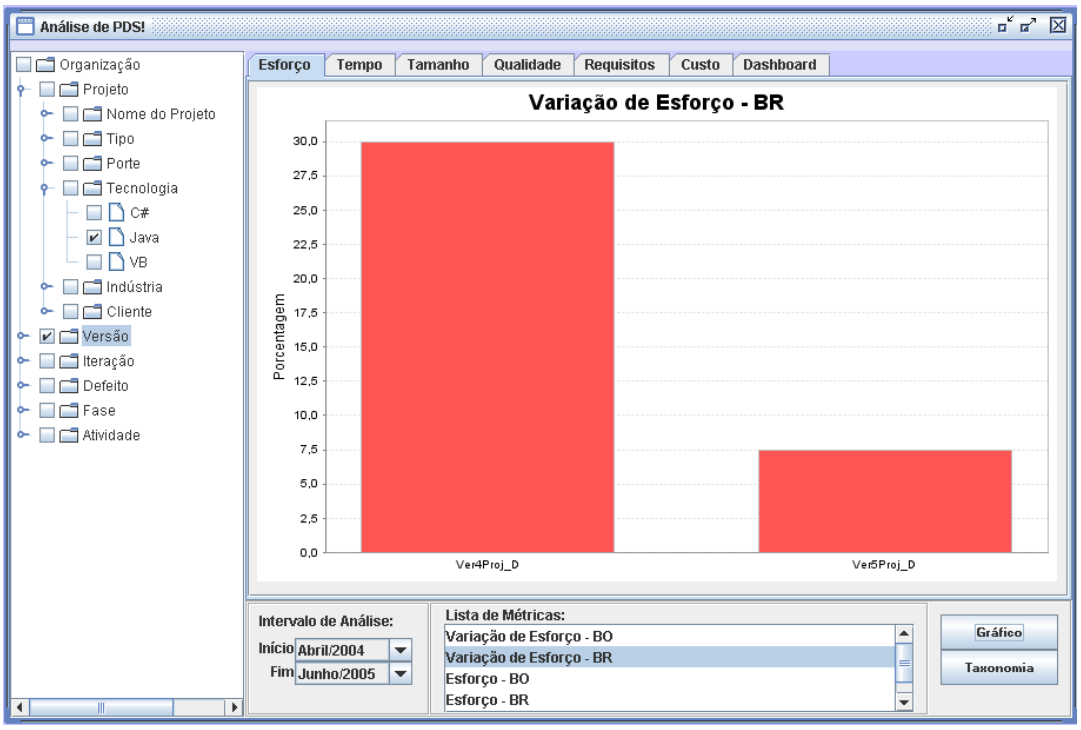

Figura 28: Exemplo de gráfico com intervalo temporal.

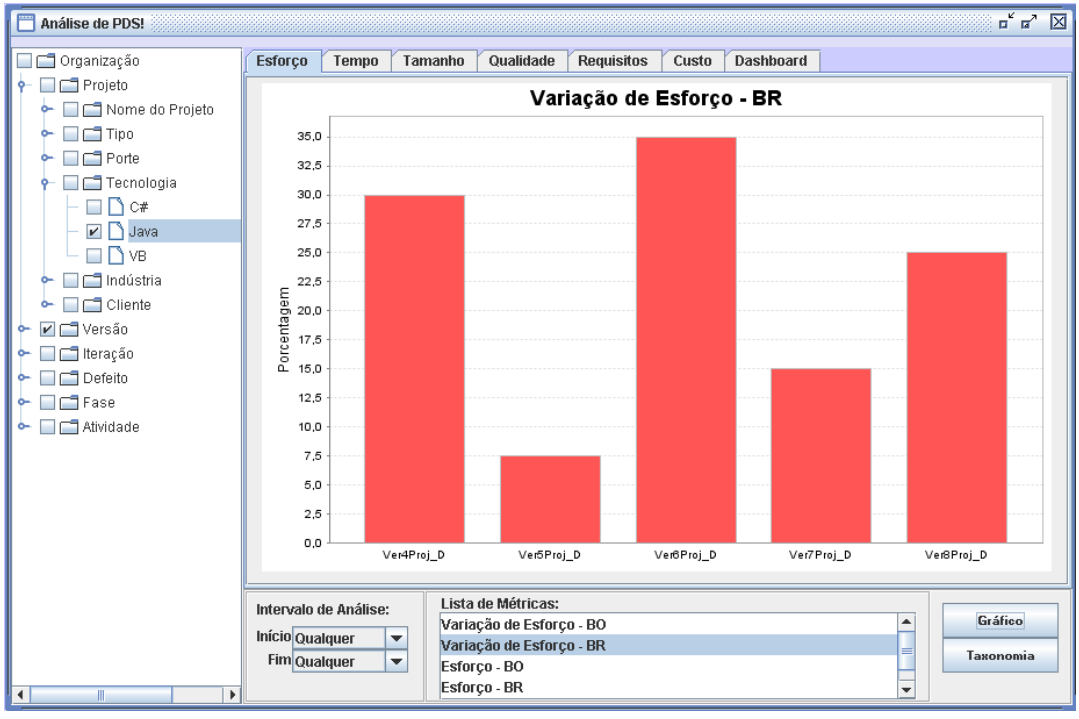

Figura 29: Gráfico de Variação de Esforço - BR.

8.1.4.2 *Taxonomias*

A definição de taxonomias de categorias, baseada nos indicadores de métricas, discutido na Seção 5.3, provê maior semântica à apresentação dos dados organizacionais. Por exemplo, uma taxonomia para representar a métrica satisfação de clientes pode possuir as categorias Satisfaz (com indicador média do índice >4) e Não_Satisfaz (com indicador média do índice <=4). Já, a taxonomia que representa Variação de Esforço – BR pode possuir as categorias: Bom (com indicador <=5%), Aceitável (com indicador >5% e <25) e Regular (com indicador >=25%). A Figura 30 representa a definição de categorias e indicadores da taxonomia para Variação de Esforço – BR.

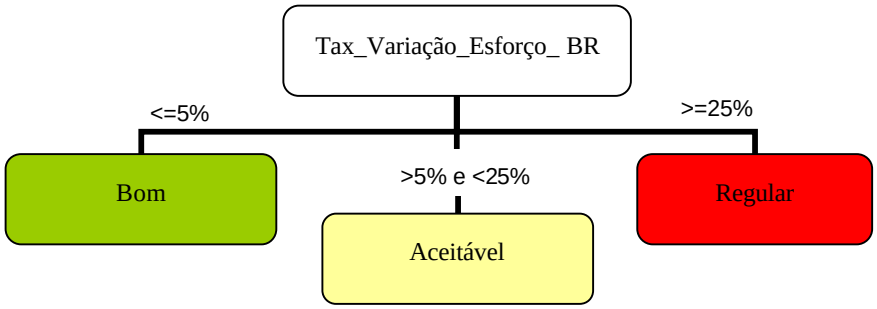

Figura 30: Taxonomia Variação de Esforço - BR.

Taxonomias podem ser definidas para cada uma das métricas da organização. Para visualizar a representação gráfica das taxonomias, o usuário deve selecionar uma única métrica na Lista de Métricas e clicar no botão “Taxonomia”. Os indicadores e as categorias de taxonomias devem estar previamente armazenados no banco de dados.

A distribuição de dados através das categorias de uma taxonomia permite ao usuário final identificar rapidamente problemas de desempenho apresentados por dados retornados por suas consultas. A Figura 31 apresenta a consulta apresentada na Figura 29 conforme a distribuição de seus dados pela taxonomia definida na Figura 30. Como pode-se obervar, 50% das versões de versões de projetos de tecnologia Java apresentam desempenho regular, e 50% revelam desempenho aceitável.

A apresentação de dados de projetos através das taxonomias disponibiliza uma visão organizacional sobre uma determinada métrica. A utilização deste recurso é de grande importância principalmente para os perfis gerenciais que não necessitam visualizar valores específicos, mas sim sua distribuição de acordo com os indicadores de qualidade da organização.

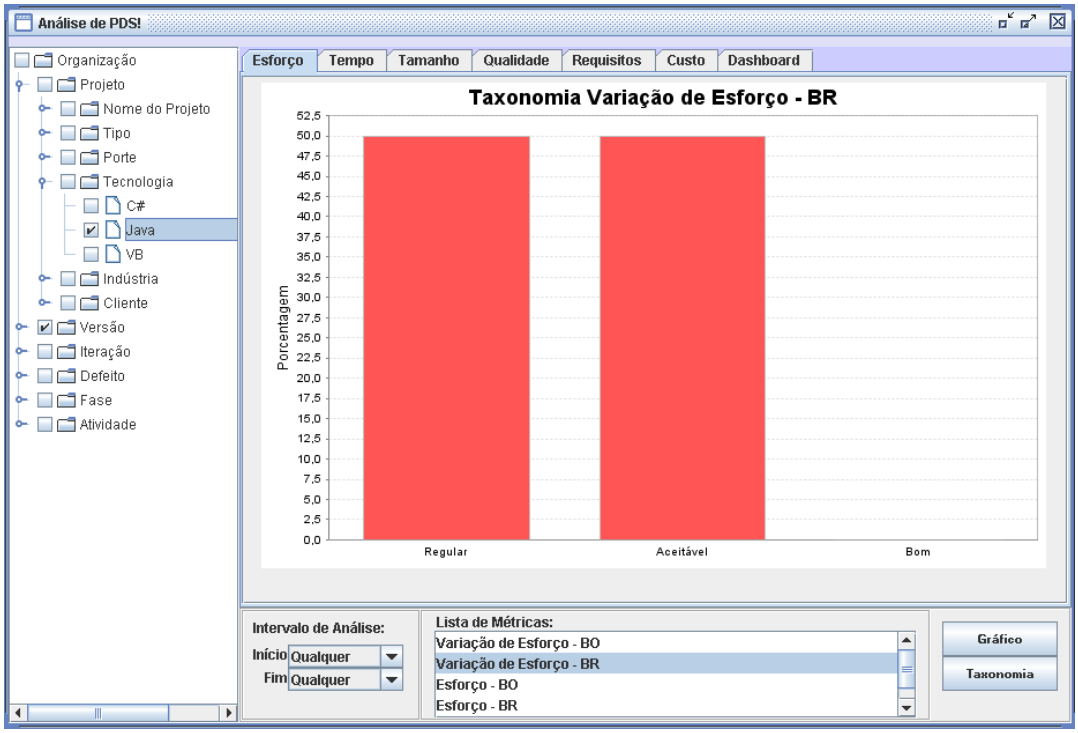

Figura 31: Gráficos para análise de Variação de Esforço - BR.

Outro aspecto considerado quanto à utilização de taxonomias é o fato de muitos dos usuários da organização desconhecerem os indicadores de qualidade organizacionais. Através das taxonomias há o compartilhamento dos indicadores de qualidade por todos os usuários e conseqüentemente, um maior comprometimento em alcançar as metas organizacionais. Neste sentido, a distribuição dos dados através das categorias definidas por uma taxonomia agrega semântica aos dados. O usuário não necessita ser profundo conhecedor dos valores esperados para acompanhar o desempenho de seus projetos.

Ao visualizar o gráfico provido pela

Figura 29, observa-se o desempenho das diversas versões de projetos desenvolvidas em Java. Contudo, através deste gráfico, não é possível analisar como está o desempenho destas versões diante das metas organizacionais. Assim, ao analisar a Figura 31, que apresenta os dados através das taxonomias definidas, rapidamente se detecta que 50% das versões desenvolvidas em Java estão com um desempenho regular e os outros 50%, com um desempenho aceitável.

O conhecimento dos perfis de usuários e dos requisitos de análise organizacional permite que a organização defina um conjunto de taxonomias que represente as expectativas de análise e as apresente através de um *dashboard*, este assunto é abordado na próxima seção.

8.1.4.3 *Dashboards*

Um *dashboard*, como apresentado na Seção 3.3.2, disponibiliza informação segundo indicadores de qualidade previamente definidos pela organização, os quais são representados pelas categorias das taxonomias.

Os *dashboards* são voltados a mostrar diferentes assuntos de uma organização de uma forma sintética. Considerando que as necessidades de análise requeridas por uma organização é representada por um PM, o *dashboard* provê uma visão organizacional através das taxonomias definidas para todas as métricas contempladas pelo PM ou somente para as mais importantes. Assim, quando se altera o conjunto de taxonomias, o *dashboard* é atualizado automaticamente, segundo a nova necessidade de análise organizacional.

Os *dashboards* consideram os perfis gerenciais da organização, os quais não dispõem de tempo para analisar detalhes do desempenho organizacional. Normalmente estes perfis requisitam visualizações do desempenho da organização como um todo, talvez de um projeto específico, ou no máximo de uma versão. Neste sentido, a camada de apresentação proposta disponibiliza ao usuário organizacional, *dashboards* com as visões da organização, de projeto ou de uma versão específica.

O cálculo do *dashboard* que provê uma visão organizacional é realizado através da média das métricas encontradas em versões já concluídas de todos os projetos. O *dashboard* que provê uma visão de projeto é calculado através das médias das versões concluídas de um projeto específico. Finalmente, o *dashboard* que provê visão de versões apresenta os valores da própria versão.

Exemplificando, considere que uma determinada organização tenha definido o conjunto de taxonomias, através do modelo analítico, representado pela Tabela 10. Suponha também que o DW tenha dados de 5 projetos, cada um contendo suas respectivas versões. A Tabela 11 apresenta exemplos fictícios de projetos, valores de métricas e as sumarizações destas para a visão da organização.

Tabela 10: Exemplo de Taxonomias definidas.

Taxonomia	Indicador definido		
Nome	*Bom*	*Aceitável*	*Regular*
Tax_Variacao_Esforço_BR	<=5%	>5% e <20%	>=20%
Tax_ Densidade de Defeitos Internos	>10%	>=10% e >=20%	>20%
Tax_Variacao_Cronograma_BR	<=5%	>5% e <25%	>=25%

Tabela 11: Exemplos de valores de organização - projeto - versão.

Projeto	Versão	Variação de Esforço - BR	Densidade de Defeitos Internos	Variação de Cronograma - BR
Projeto A	Ver1Proj_A	5%	10%	7%
Média - Projeto A		*5%*	*10%*	*7%*
Projeto B	Ver1Proj_B	30%	20%	15%
Média - Projeto B		*30%*	*20%*	*15%*
Projeto C	Ver1Proj_C	7.5%	12%	5%
Média - Projeto C		*7.5%*	*12%*	*5%*
Projeto D	Ver1Proj_D	51%	25%	16%
	Ver2Proj_D	49%	27%	9%
	Ver3Proj_D	33%	20%	11%
	Ver4Proj_D	30%	30%	16%
	Ver5Proj_D	7.5%	25%	15%
	Ver6Proj_D	35%	22%	13%
	Ver7Proj_D	15%	20%	20%
	Ver8Proj_D	25%	30%	17%
Média - Projeto D		*31%*	*15%*	*25%*
Projeto E	Ver1Proj_E	15%	20%	29%
	Ver2Proj_E	8%	18%	25%
	Ver3Proj_E	23%	23%	23%
	Ver4Proj_E	23%	25%	27%
	Ver5Proj_E	11%	19%	25%
	Ver6Proj_E	17%	17%	21%
	Ver7Proj_E	10%	15%	23%
Média - Projeto E		*15%*	*25%*	*20%*
Média Organizacional		*16.2%*	*16.4%*	*11%*

A Figura 32 apresenta o *dashboard* que proporciona ao usuário uma visão do desempenho organizacional. Para ter esta visualização, o usuário selecionou a opção Organização nas perspectivas e após a aba *Dashboard*, localizada ao lado das AQ. Como pode ser observado, a organização está tendo um desempenho *aceitável* quanto às 3 métricas consideradas importantes para esta, uma vez que foram contempladas com a definição de taxonomias. Contudo, a Variação de Esforço – BR e Densidade de Defeitos exigem atenção, uma vez que os valores obtidos encontram-se muito próximos à indicação de desempenho *regular*. Já o valor obtido para a Variação de Cronograma – BR, encontra-se mais próximo ao valor que indica um desempenho bom.

A Figura 33 apresenta o *dashboard* que seria disponibilizado a um usuário interessado em acompanhar o desempenho do Projeto A quanto aos indicadores organizacionais. Para tal, o usuário selecionou nas perspectivas o Projeto A e após selecionou a aba *Dashboard.* Este *dashboard* permite observar que o projeto possui um desempenho *bom* em quase todas as métricas definidas como essenciais pela organização. A exceção encontra-se na métrica Densidade de Defeitos Internos, onde o desempenho do projeto enquadrou-se como *aceitável.*

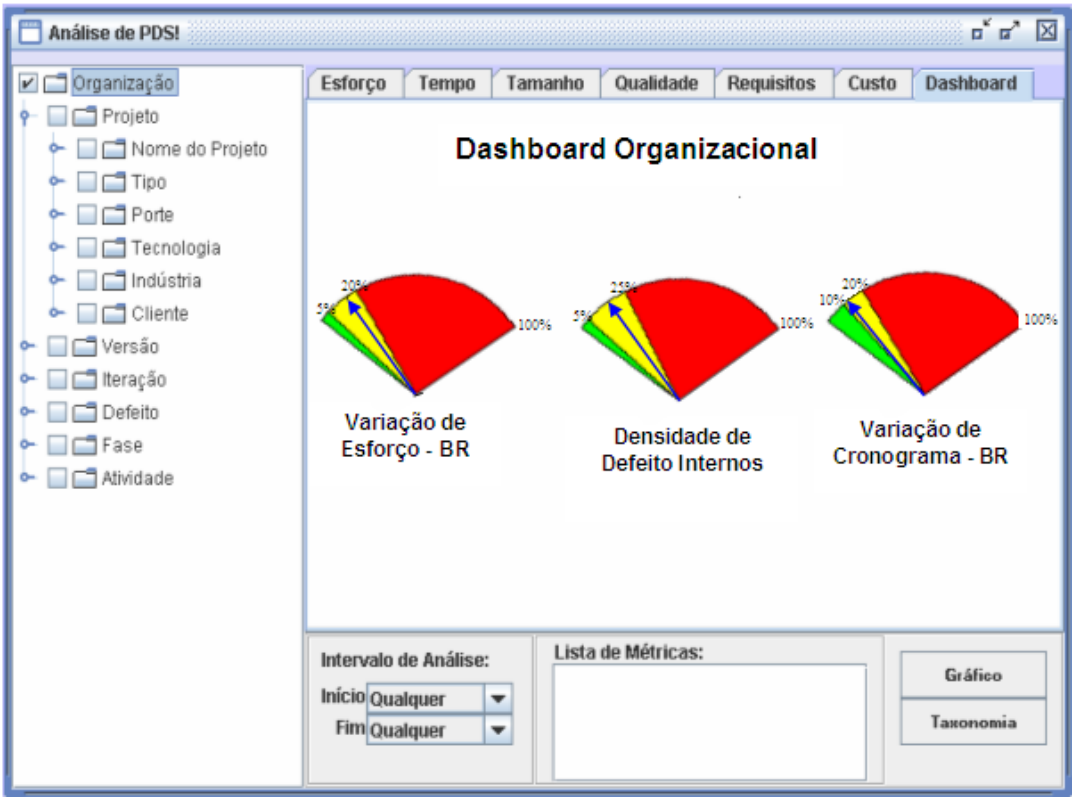

Figura 32: Dashboard – Visão Organizacional.

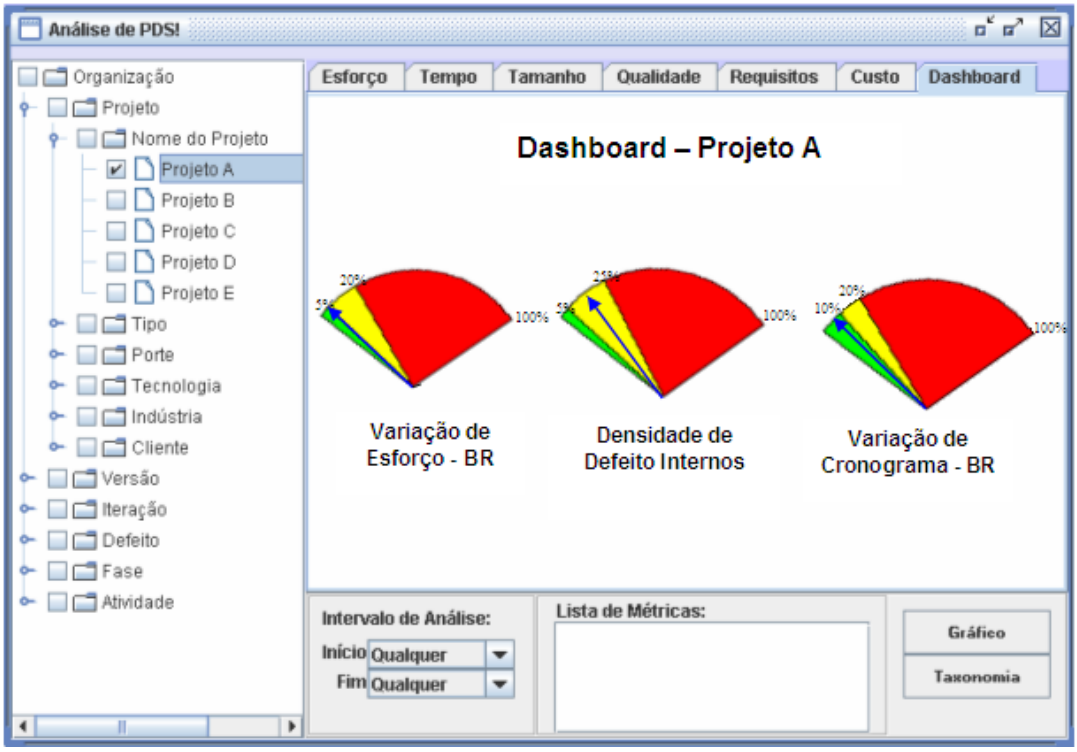

Figura 33: Dashboard – Visão Projeto A.

Finalmente, a Figura 34 apresenta o *dashboard* da versão de um projeto específico. No exemplo o usuário selecionou a versão identificada como Ver3Proj_De pela visualização do *dashboard* desta, observa-se que a mesma possui problemas de desempenho. Quanto à Variação de Esforço – BR e a Defeitos Internos, esta apresenta um desempenho considerado *regular*. Somente quanto ao desempenho da Variação de Cronograma – BR, esta versão mostra um desempenho *bom*.

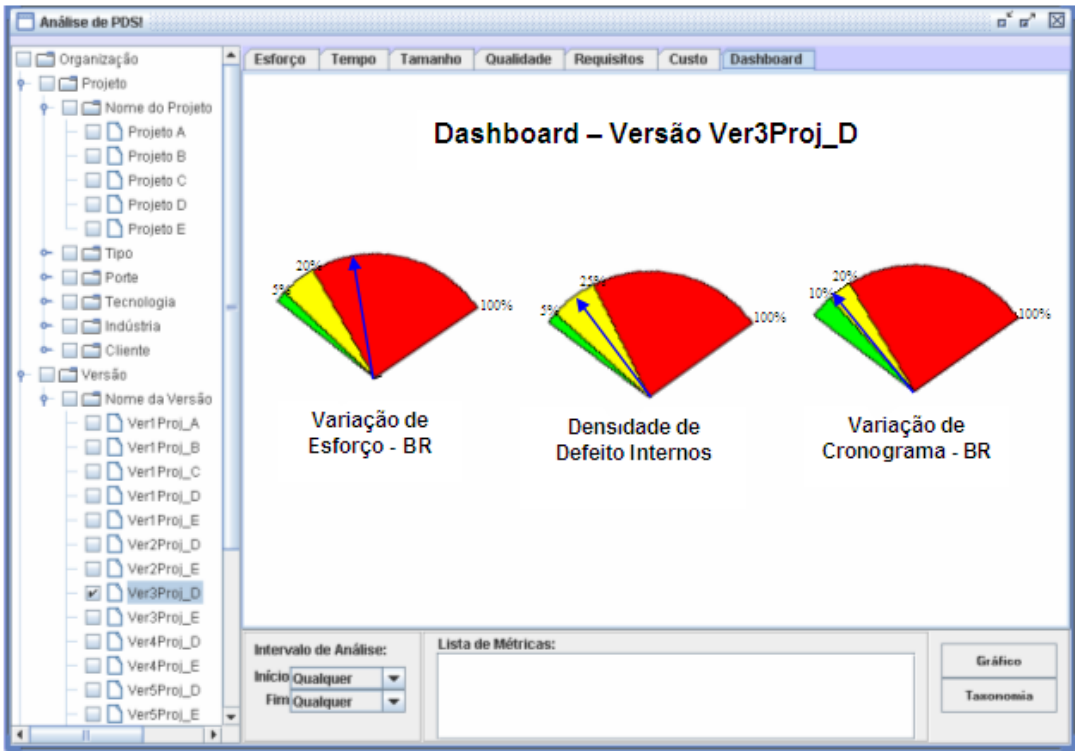

Figura 34: Dashboard – Visão Versão Ver3Proj_D.

8.2 Componente de Análise

Os recursos analíticos apresentados na seção anterior facilitam a geração de consultas para análise dos dados realizada pelo usuário da camada de apresentação. Contudo, para que estes dados sejam disponibilizados se faz necessária uma infra-estrutura que intermedie a consulta gerada através dos componentes gráficos da interface e sua execução propriamente dita no DW. O termo redirecionador de consultas é utilizado no contexto de ambientes de *Data Warehousing* para designar componentes funcionais com este papel [KIM98], sendo o navegador de agregados um dos tipos de redirecionadores mais comuns [BEC04].

O redirecionador de consultas em questão recebe os vários parâmetros oriundos da interface de usuário, e através de metadados definidos, monta as consultas SQL

correspondentes que serão realizadas sobre o DW. O projeto deste redirecionador de consultas foi baseado na estrutura do modelo analítico e na natureza das possíveis consultas que poderão ser requeridas pelo usuário através do componente de apresentação. Como pode se observado, a maioria das métricas que compões o PM são razões sobre métricas base, as quais, em sua maioria, devem ser calculadas individualmente por envolverem tabelas fatos e/ou dimensões diferentes. Por exemplo, para o cálculo da métrica Variação de Esforço – BR são necessários o esforço – BR (estimado) e o esforço – ER (real), os quais são armazenados nas tabelas Fato_Fase e Fato_TipoAtividade, respectivamente. Além disto, uma consulta que resulte no cálculo destes dois valores ao mesmo tempo é extremamente complexa, pois esta deve considerar diferentes granularidades de cada um dos valores utilizados, bem como os diferentes usos de uma mesma tabela.

A Figura 35 apresenta a estrutura e interação do redirecionador de consultas com outros compenentes do ambiente de *Data Warehousing*. Os metadados têm por objetivo auxiliar no mapeamento dos parâmetros selecionados na interface para os respectivos atributos do DW, de forma a obter todos os elementos necessários às cláusulas das estruturas de consultas SQL previamente definidas. O algoritmo do direcionador interage com os metadados e, de posse dos dados mapeados, escreve as estruturas de consultas SQL necessárias. Finalmente, estas são executadas no DW, de onde retornará os resultados.

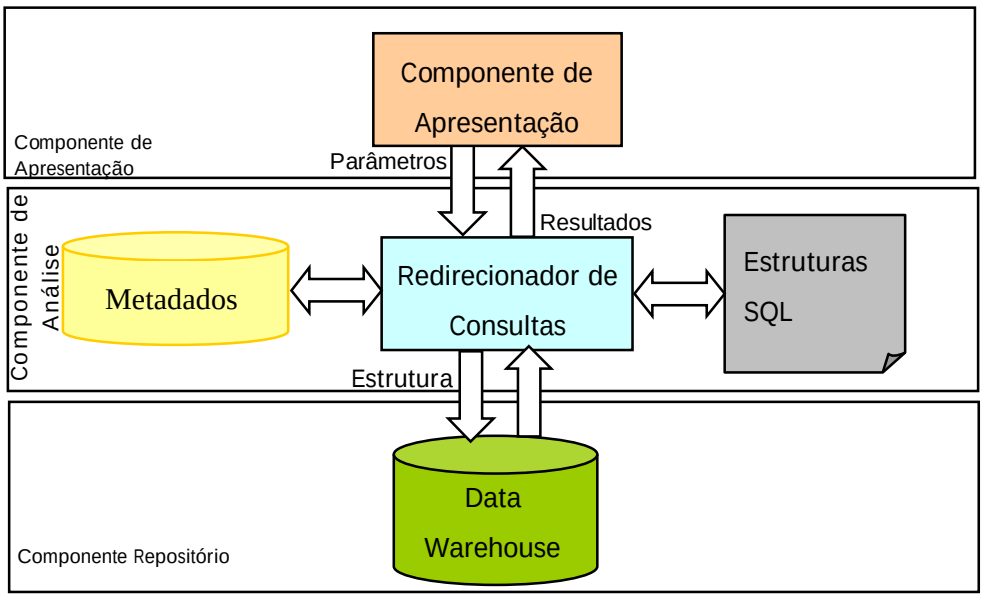

Figura 35: Estrutura Redirecionador de consultas.

Os parâmetros oriundos da interface gráfica podem ser divididos em três grupos, como especificado na Seção 8.1: qualitativos, quantitativos e parâmetros temporais. Os parâmetros qualitativos podem ser classificados de duas formas: parâmetros que representam atributos (e.g nome de projeto) e parâmetros que representam valores (e.g. nome de projeto = “A”).

A próxima seção apresenta a definição das estruturas de consultas SQL que deverão ser geradas. Após, apresenta-se brevemente cada um dos metadados especificados transcorrendo sobre seus objetivos e finaliza-se apresentando como o algoritmo interage para a realização das consultas.

8.2.1 Definição de Estruturas das Consultas SQL

A consulta sobre um modelo analítico voltado a métricas de software com diversos níveis de granularidade deve ser cuidadosamente composta de forma a garantir a correção do resultado final. A natureza das métricas que compõem o PM, aliada à estrutura do modelo analítico definido, requer a definição de diferentes tipos de estrutura de consultas.

A diferença entre dados estimados ou reais é definida pela dimensão Dim_Tipo_Fato. Assim, enquanto uma possível análise de intervalo de tempo requerida por um usuário recai sobre versões de projetos em desenvolvimento no tempo selecionado, as possíveis métricas selecionadas para esta mesma consulta podem necessitar de dados sobre fatos que correspondem a estimativas e/ou realizações (e.g. métricas de variações). Neste sentido, três estruturas de consultas SQL fazem-se necessárias: (1) a primeira estabelece a restrição temporal considerando somente fatos reais; (2) a segunda considera o retorno da consulta temporal e ainda os parâmetros quantitativos e qualitativos na busca de métricas base, consultadas em fatos reais ou estimados, que constituem um métrica derivada; e então (3) a terceira estrutura considera os resultados retornados das duas anteriores buscando o resultado de métricas derivadas.

A Figura 36 apresenta a interação destas diferentes estruturas de consultas definidas. Assim, inicialmente realiza-se a consulta temporal sobre o DW. Por sua vez, esta é utilizada para restringir as consultas sobre cada uma das métricas base que compõem uma métrica derivada. Por fim, as diferentes métricas base são utilizadas para cálculo de métricas derivada, consolidando todos os resultados. Ressalta-se que, como o usuário pode selecionar uma ou mais métricas, estes passos repetir-se-ão de acordo com o número de métricas selecionadas pelo usuário.

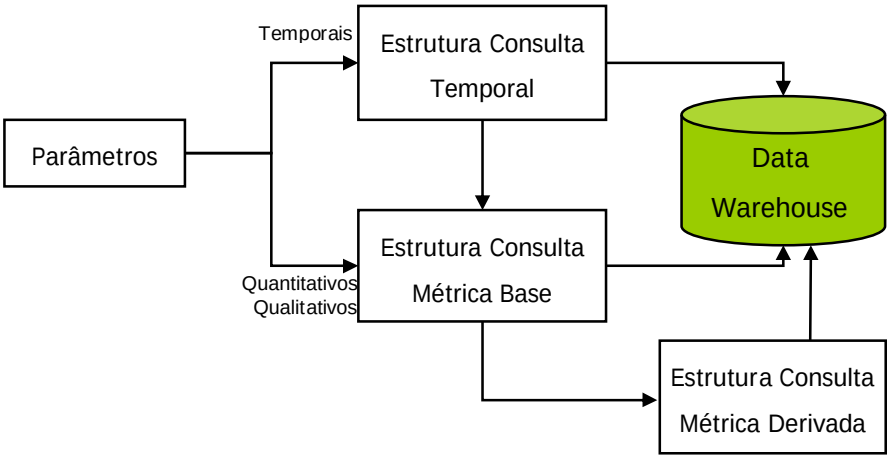

Figura 36: Interação de estruturas SQL definidas.

As estruturas para os diferentes tipos de consultas visam definir os componentes necessários a cada cláusula destas e, assim, nortear o mapeamento dos parâmetros oriundos da interface considerando os metadados definidos.

8.2.1.1 *Estrutura Consulta Temporal*

A consulta temporal tem por objetivo buscar versões de projetos que respeitam um período especificado pelo usuário (versões iniciadas após tal data, concluídas até tal data, ou iniciadas e terminadas dentro de um intervalo especificado). Os parâmetros temporais correspondem a início e/ou fim de um período. A consulta temporal é armazenada como visão a fim de facilitar a recuperação posterior dos dados.

O gráfico apresentado pela Figura 28 exemplica uma consulta com restrições temporais enquanto que a Figura 37 ilustra a consulta, visão, temporal correspondente. Salienta-se que nesta o atributo “Dim_Tipo_Fato=1” corresponde a fatos reais e os colchetes “[]” representam clausulas opcionais.

```
CREATE VIEW Consulta_Temporal
AS
SELECT Fato_Release.Id_Release AS Versao
FROM   Dim_Tempo as Inicio,  Dim_Tempo as Fim, Fato_Release, Dim_Tipo_Fato
WHERE (Inicio.Data > = CONVERT (DATETIME, '2004-04-01 00:00:00', 102))) AND
          (Fim.Data < = CONVERT (DATETIME, '2005-01-07 00:00:00', 102))) AND
           Dim_Tipo_Fato.Tipo = 1  AND
           Fato_Release.Id_Tempo_Ini = Inicio. Id_Tempo AND
           Fato_Release.Id_Tempo_Fim = Fim. Id_Tempo AND
           Fato_Release.Id_Tipo_Fato = Dim_Tipo_Fato.Id_Tipo_Fato
```

Figura 37: Aplicando a estrutura de consulta temporal.

A Figura 38 apresenta a estrutura de uma consulta temporal SQL, juntamente com todos seus componentes.

```
CREATE VIEW Consulta_Temporal
AS
SELECT Fato_Release.Id_Release AS Versao
FROM   [Dim_Tempo_Ini], [Dim_Tempo_Fim],  Fato_Release,  Dim_Tipo_Fato
WHERE [Tempo_Ini] AND
        [Tempo_Fim]  AND
        Dim_Tipo_Fato.Tipo = 1  AND
       Junção
```

Figura 38: Estrutura Consulta Temporal.

Abaixo, apresenta-se cada uma das cláusulas que compõem a estrutura SQL definida.

a) Cláusula Select

Composta somente pelo atributo que identifica versões desenvolvidas no intervado especificado.

b) Cláusula From

Comporta o tempo da restrição temporal definida pelo usuário (Dim_Tempo_Ini e Dim_Tempo_Fim) as quais são armazenadas na tabela Dim_Tempo, o fato que determina a versão analisada (Fato_Release) e, finalmente a dimensão Dim_Tipo_Fato é usada para restringir datas de início e fim reais.

c) Cláusula Where

Esta cláusula estabelece as restrições envolvendo a verificação das datas (Tempo_Ini e/ou Tempo_Fim), o tipo de fato (valor "1" representa fato real) e as condições da junção das tabelas listadas na cláusula From (Junção).

8.2.1.2 *Consulta sobre Métricas*

As consultas geradas sobre o PM podem ser classificadas como segue:

- Consultas – Métricas base: este tipo de consulta não exige grande esforço na sua estruturação uma vez que todos os valores necessários encontram-se armazenados diretamente em um único fato do modelo analítico. Estes valores são obtidos diretamente da base de dados, possivelmente utilizando funções de agregação. Exemplos de métricas calculadas com este tipo de consulta são: esforço real, esforço BR, duração real, entre outras.
- Consultas – Métricas Derivadas: gerar consultas para cálculo de métricas derivadas pode ser bastante complexo uma vez que cada uma das métricas base utilizadas neste cálculo pode exigir tabelas (fatos e dimensões) específicas. Neste sentido, dispondo as consultas sobre as métricas base já realizadas, este tipo de consulta tem como objetivo unificar estas caonsultas através da função de uma determinada métrica derivada. Por exemplo, tendo-se o esforço real (ER) e o esforço BR (EBR) poderia se aplicar a função da métrica derivada de Variação de Esforço – BR que é ((ER – EBR)*100/EBR).

O redirecionador de consultas inicialmente decompõe as consultas sobre métricas derivadas em consultas sobre todas as métricas bases necessárias. As consultas às métricas base que compõem uma métrica derivada devem realizar-se sobre os mesmos parâmetros qualitativos, bem como envolver as mesmas versões filtradas na consulta temporal. A cada uma das consultas sobre métricas base executada dá-se o nome de consulta componente. Ao

final, todas as métricas base são reunidas através da função da métrica derivada. Ressalta-se que este procedimento é executado para cada métrica derivada escolhida.

A Figura 39 apresenta um exemplo de como uma consulta por métrica derivada seria decomposta em métricas base e após, consolidada através de uma consulta que inclui a função da métrica derivada. A consulta exemplifica a busca de Variação de Esforço – BR de versões de tecnologia Java, referente ao período de Abril/2004 a Julho/2005.

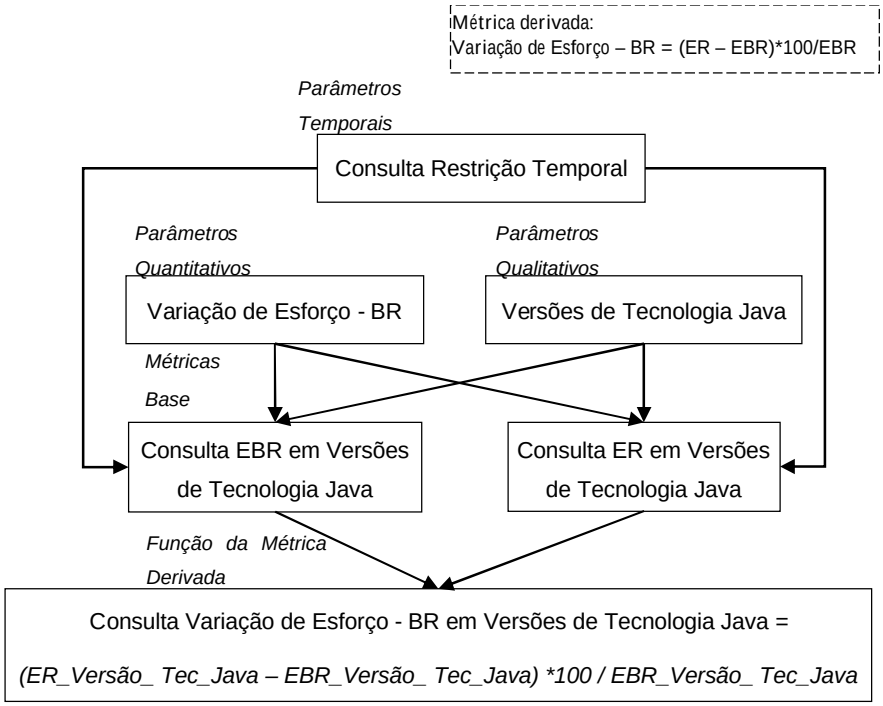

Figura 39: Exemplo de consulta por métrica derivada.

8.2.1.2.1 Estrutura Consulta - Métricas Base

Considerando o exemplo da Figura 28, para o cálculo de Variação de Esforço – BR são necessárias duas métricas base, Esforço Real (ER) e Esforço – Baseline Revisado (EBR). As consultas geradas para este exemplo são ilustradas a seguir. A Figura 40 apresenta a consulta para obtenção do ER enquanto que a Figura 41 apresenta a obtenção do EBR. Como observado, as consultas para busca das métricas base são armazenadas como visões para que possam ser facilmente recuperadas para o cálculo da métrica derivada.

```
CREATE VIEW ER
AS
SELECT Dim_Release.Nome, SUM(Fato_Atividade.Esforço)AS ER
FROM  Dim_Release, Dim_Tecnologia, Fato_Atividade, Dim_Tipo_Fato
WHERE Dim_Tecnologia.Nome= "Java" AND
      Dim_Tipo_Fato.Tipo = 1 AND
      Fato_Atividade.ID_Release = Dim_ Release.ID_ Release AND
      Fato_Atividade.ID_Tipo_Fato = Dim_Tipo_Fato. ID_Tipo_Fato AND
      Fato_Atividade.ID_Tecnologia = Dim_ Tecnologia. ID_Tecnologia AND
      [Fato_Atividade.ID_Release IN (Consulta_Temporal.Versao) AND
      Fato_Atividade.ID_Release = Consulta_Temporal.Versao]
GROUP BY Dim_ Release.Nome;
```

Figura 40: Aplicando a estrutura de consulta Métrica Base - ER.

```
CREATE VIEW EBR
AS
SELECT Dim_ Release.Nome, SUM(Fato_Fase.Esforço)AS EBR
FROM  Dim_ Release, Dim_Tecnologia, Fato_Fase, Dim_Tipo_Fato
WHERE Dim_Tecnologia.Nome= "Java" AND
      Dim_Tipo_Fato.Tipo = 3 AND
      Fato_Fase.ID_ Release =Dim_ Release.ID_ Release AND
      Fato_Fase.ID_Tipo_Fato= Dim_Tipo_Fato. ID_Tipo_Fato AND
      Fato_Fase.ID_Tecnologia = Dim_ Tecnologia. ID_Tecnologia AND
      [Fato_Fase.ID_Release IN (Consulta_Temporal.Versao) AND
       Fato_Fase.ID_Release = Consulta_Temporal.Versao]
GROUP BY Dim_ Release.Nome;
```

Figura 41: Aplicando a estrutura de consulta Métrica Base - EBR.

A estrutura da consulta SQL para busca de métricas base no DW considera todos os elementos necessários a cada uma de suas cláusulas. A Figura 42 apresenta a estrutura consolidada juntamente com os elementos especificados.

```
CREATE VIEW Met_Base_Text
AS
SELECT Atrib_Dim_User, Função_ Metr_Base(Metr_Base)AS Met_Base_Text
FROM  Dimensao_User, Dim_Metr_Base, Fato, Dim_Tipo_Fato
WHERE Val_Dim_User AND
      Val_ Metr_Base AND
      Val_Tipo_Fato AND
      Fato.ID_Release IN (Consulta_Temporal.Versao) AND
      Junção
GROUP BY Atrib_Dim_User;
```

Figura 42: Estrutura Consulta a Métrica Base.

Abaixo apresenta-se cada um dos elementos que compõe cada uma das cláusulas de uma estrutura de consulta SQL definida para métricas base.

a) Cláusula Select

Comporta os parâmetros qualitativos selecionados pelo usuário e ainda da métrica base selecionada ou necessária ao cálculo de uma métrica derivada. Os atributos qualitativos selecionados determinam a função que acompanha a métrica base.

A estrutura SQL denomina os parâmetros qualitativos que representam atributos do DW como Atrib_Dim_User (*e.g.* Dim_Release.Nome), a função que acompanha a métrica base como Função_Metr_Base (AVG, SUM ou vazio, no caso de atributo quantitativo ser uma data), a métrica base é denominada como Metr_Base (*e.g.* Fato_Atividade.Esforço, Fato_Fase.Esforço, etc) e a expressão retornada pela métrica base como Met_Base_Text (*e.g.* ER, EBR, etc).

A Função_Metr_Base utilizada na consulta a métrica base depende da consulta realizada pelo usuário e da métrica a ser consultada. Por exemplo, ao consultar ER ou EBR de projetos utiliza-se AVG, uma vez que o esforço é armazenado na granularidade de versão. Já, ao consultar esforço realizado ou estimado de versões utiliza-se SUM, uma vez que o esforço é armazenado na granularidade de versão e finalmente ao buscar-se datas não utiliza-se nenhuma função então, vazio.

b) Cláusula From

Comporta as dimensões selecionadas pelo usuário (Dim_User), as dimensões adicionais necessárias ao cálculo das métricas base (Dim_Metr_Base), a dimensão que determina o tipo de fato a ser consultado (Dim_Tipo_Fato), o fato propriamente dito (Fato) e a consulta temporal previamente executada (Consulta_Temporal).

c) Cláusula Where

Esta cláusula expressa os parâmetros qualitativos que representam restrições de valores. Estes podem ser selecionados pelo usuário (Val_Dim_User) isto é, o usuário pode consultar por determinado valor (*e.g.* Dim_Tecnologia.Nome = "Java"), ou ainda podem ser restrições de valores necessárias ao cálculo de algumas métricas, Val_Metr_Base (*e.g.* Dim_Atividade.Tipo = "Revisão"), restrições quanto ao fato a ser consultado, Val_Tipo_Fato (*e.g.* Dim_Tipo_Fato.Tipo = 3) e finalmente as restrições quanto as junções de todas as tabelas da consulta (Junção). Caso a consulta definida pelo usuário possua restrição temporal parte-se da premissa que esta já foi previamente executada e retornou o identificador das versões que deverão ser consideradas nas consultas das métricas base (Consulta_Temporal.Versao).

d) Cláusula Group by

Composta pelos atributos especificados nos parâmetros qualitativos selecionados pelo usuário (Atrib_Dim_User).

8.2.1.2.2 Estrutura Consulta Métrica Derivada

A consulta métrica derivada tem por objetivo consolidar as consultas previamente realizadas sobre as métricas base e aplicar a função da métrica derivada selecionada pelo usuário.

O número de consultas realizadas sobre métricas base é determinado pela métrica derivada selecionada. Neste sentido, denomina-se cada uma das consultas realizadas sobre as métricas base como Consulta_Componente. Como já ressaltado, os parâmetros qualitativos selecionados pelo usuário se mantêm por todas as consultas às métricas base, sendo assim iguais em todas as consultas componente geradas.

Considerando o exemplo da Figura 28, para o cálculo de Variação de Esforço – BR e as consultas sobre as métricas base necessárias, ER na Figura 40 e EBR na Figura 41. A Figura 43 apresenta a consulta gerada a partir destas consultas previamente geradas.

```
SELECT ER.Nome, (ER.ER - EBR.EBR)*100/EBR.EBR AS [Variação de Esforço - BR]
FROM ER, EBR
WHERE ER.Nome = EBR.Nome
```

Figura 43: Aplicando a estrutura de consulta Métrica Derivada.

Assim, considerando os elementos necessários as cláusulas da estrutura SQL para consultar a métrica derivada, a Figura 44 apresenta a estrutura SQL consolidada juntamente com os elementos especificados.

```
SELECT Comp_Atrib_Dim_User, Funcao_Metrica_Derivada AS Met_Derivada_Text
FROM Consultas_Componente
WHERE Junção
```

Figura 44: Estrutura Consulta a Métrica Derivada.

Abaixo apresenta-se os elementos definidos para cada uma das cláusulas desta consulta SQL.

a) Cláusula Select

Comporta os parâmetros qualitativos selecionados pelo usuário (Comp_Atrib_Dim_User). Uma vez que os parâmetros qualitativos se repetem por todas consultas componentes, busca-se os parâmetros qualitativos da primeira consulta componente somente. A cláusula *Select* também comporta a função da métrica derivada selecionada pelo usuário, Funcao_Metrica_Derivada (*e.g* (ER.ER - EBR.EBR)*100/EBR.EBR) e o nome atribuído a esta, Met_Derivada_Text (*e.g* Variação de Esforço - BR).

b) Cláusula From

Esta cláusula referencia todas as consultas componentes necessárias ao cálculo da métrica derivada em questão.

c) Cláusula Where

A cláusula *Where* armazena a junção das consultas componentes (Junção).

8.2.2 Metadados

Os metadados definidos têm por objetivo mapear os parâmetros selecionados na interface para as respectivas dimensões, fatos e atributos do DW. Para tal, foram definidos quatro metadados: Meta_Qual, Meta_Metrica, Meta_Juncao e Meta_Funcao_Metrica. O Anexo II apresenta para o estudo de caso, um exemplo da instanciação de cada metadado, considerando o DW, apresentado no Capítulo 7. A Figura 45 representa sinteticamente cada metadado definido juntamente com seus atributos.

Figura 45: Metadados definidos.

O metadado Meta_Qual auxilia a mapear as restrições estabelecidas pelo usuário através das perspectivas. Cada nodo selecionado na árvore (param_Qual) é uma *string*, contendo a relação do usuário. Assim, exemplos de param_Qual são: “Nome Projeto”, “Tecnologia = Java”, entre outros. Assume-se a operação busca() definida sobre a classe Meta_Qual, a qual recupera a instância de Meta_Qual que se refere a um dado param_Qual passado como argumento. A Tabela 12 descreve os atributos das instâncias de Meta_Qual.

O metadado Meta_Metrica auxilia a definir as consultas componentes necessárias para uma métrica (base ou derivada) selecionada pelo usuário, juntamente com os atributos, tabelas e condições necessárias a cada uma delas. Cada métrica da Área de Métricas selecionada é uma string (param_Quant), que contém o seu nome. Assume-se que a operação busca() definida sobre Meta_Metrica recupera instâncias que casam com o param_Quant. Assim, cada instância representa uma métrica que é buscada através de uma consulta componente. A Tabela 13 descreve os atributos do Meta_Metrica.

Tabela 12: Metadado Meta_Qual.

Atributo	Objetivo
Qual_Str	Armazena os possíveis parâmetros qualitativos que poderão ser selecionados pelos usuários. Exemplos: Nome do Projeto, Tecnologia, entre outros.
Dim_User	Armazena as dimensões dos possíveis atributos Qual_Str. Exemplos: Dim_Projeto, Dim_Tecnologia, entre outros.
Atrib_User	Armazena os possíveis atributos de cada uma das dimensões de um modelo analítico. Exemplos: Dim_Projeto.Nome, Dim_Tecnologia.Nome, entre outros.
Val_User	Este parâmetro é preenchido pelo usuário, quando pertinente, com valor para um determinado atributo. Exemplos: Tecnologia Java, Projeto A, entre outros.

Tabela 13: Metadado Meta_Metrica.

Atributo	Objetivo
Metrica_Str	Armazena todos os possíveis parâmetros quantitativos que poderão ser selecionados pelos usuários. Exemplo: Variação de Esforço – BR, Esforço realizado.
Metrica_Base_Text	Armazena uma métrica base necessária para o cálculo da métrica_Str. Exemplo: considerando a métrica Variação de Esforço – BR uma instância de Meta_Metrica conteria o valor de ER e outra o valor de EBR.
Metrica_Base	Armazena o nome da tabela fato e atributo correspondente, tal como definido no esquema do DW. Este atributo só possui valor se tipo for numérico. Exemplo: para a Metrica_Base_Text ER o valor de Metrica_Base é Fato_Atividade.Esforco.
Dim_Met_Base	Armazena a dimensão que auxiliará na consulta de uma Metrica_Base_Text. Exemplo: para a Metrica_Base_Text DE (Defeito Externo) a Dim_Metrica_Base é Dim_Defeito.
Atrib_Metr_Base	Armazena o atributo da Dim_Met_Base que auxiliará na consulta de uma Metrica_Base_Text. Exemplo: para a Metrica_Base_Text DE o Atrib_Metrica_Base é Dim_Defeito.Categoria.
Val_Metr_Base	Armazena um valor da Dim_Met_Base que auxiliará na consulta de uma Metrica_Base_Text. Exemplo: para a Metrica_Base_Text DE o Val_Metrica_Base é Dim_Defeito.Categoria= “DE”.
Tipo	Armazena o tipo de uma determinada Metrica_Base_Text. Os tipos definidos são: numérico e data. Exemplo: o tipo da métrica base ER é um numérico.
Dim_Tipo_Fato	Armazena a dimensão que define o tipo de fato que deverá ser consultado para uma determinada Metrica_Base_Text.
Val_Tipo_Fato	Armazena o valor da dimensão que define o tipo de fato (realizado ou estimado) que deverá ser consultado para uma determinada Metrica_Base_Text. Exemplo: para a Metrica_Base_Text = ER o Val_Tipo_Fato = 1 (realizado), enquanto que com a Metrica_Base_Text EBO, o Val_Tipo_Fato = 3 (estimado no BR).
Fato	Armazena o fato a ser consultado para uma determinada métrica base. Se a Metrica_Base_Text possuir tipo data, este atributo não possui valor. Exemplo: o esforço realizado é armazenado no Fato_Fase.

Após o mapeamento dos parâmetros quantitativos e qualitativos sobre os metadados Meta_Qual e Meta_Metrica obtém-se a grande maioria dos componentes necessários as estruturas de consultas SQL definidas. Metadados também são necessários para juntar todas as possíveis dimensões utilizadas na consulta com um fato relacionado.

Assim, o metadado Meta_Juncao tem como objetivo juntar uma tabela dimensão qualquer a uma tabela fato. Assume-se uma operação *busca()* que recebe como parâmetro o

nome de uma dimensão e de um fato, retorna um texto com a expressão de junção destes. A Tabela 14 descreve cada um dos atributos que compõem o metadado Meta_Juncao e uma breve exemplificação deste.

Tabela 14: Metadado Meta_Juncao.

Atributo	Objetivo
Dimensao	Armazena todas as possíveis dimensões do DW que poderão ser consultadas. Exemplo: Dim_Projeto.
Fato	Armazena um possível fato do DW a qual a dimensão está relacionada. Exemplo: Fato_Release.
Juncao	Armazena a condição de junção de dimensão e fato.Exemplo: Fato_Release.Id_Projeto = Dim_Projeto.Id_Projeto.

Os metadados apresentados até o momento auxiliam na busca dos componentes que compõem a estrutura de consulta de uma única métrica base. Contudo, após realizar cada uma das consultas às métricas base, é necessário buscar a função que deve ser aplicada para o cálculo da métrica derivada. Neste sentido, o metadado denominado Meta_Funcao_Metrica tem como objetivo associar a métrica derivada com sua função de cálculo. Assume-se que a função *busca()* sobre a classe Meta_Função_Metrica, retorna uma instância que condiz ao nome de uma dada métrica. A Tabela 15 descreve cada um dos atributos que compõem o metadado Meta_Funcao_Metrica, exemplificando-os.

Tabela 15: Metadado Meta_Funcao_Metrica.

Atributo	Objetivo
Metrica_Str	Armazena todos os possíveis parâmetros quantitativos que poderão ser selecionados pelos usuários. Exemplo: Variação de Esforço – BR, Variação de Cronograma BO, ER, entre outros.
Função	Armazena todas as funções das possíveis possíveis Metrica_Str. Exemplo: (ER-EBR)*100/EBR para a Métrica_Str igual a Variação de Esforço – BR e (DATEDIFF([day]; DTR; DTBO))*100/Duracao_BO para a Métrica_Str igual a Variação de Cronograma - BO.

8.2.3 Algoritmo do Redirecionador

O algoritmo redirecionador desenvolvido visa converter os parâmetros qualitativos, quantitativos e temporais selecionados pelo usuário em uma série de consultas SQL sobre o DW com o apoio de metadados. Neste sentido, este algoritmo recebe como parâmetros de entrada:

1) *param_Temporal*: corresponde aos parâmetros temporais selecionados pelo usuário;

2) *param_Dimensao*: corresponde aos parâmetros qualitativos selecionados pelo usuário;

3) *param_Metrica*: corresponde aos parâmetros quantitativos selecionados pelo usuário.

O algoritmo cerne da geração de consultas a métricas base denomina-se execute_Principal, Figura 48. Contudo, este algoritmo utiliza-se de outros algoritmos que o auxiliam a montagem da consulta à métrica base e derivada. Estes são: execute_Consulta_Temporal, execute_Meta_Qual, monta_Consultas_Componente, execute_Funcao_Metr_Base e execute_Consulta_Base e monta_Consulta_Derivada. Abaixo, apresenta-se de forma sintética cada um dos passos de execução que o compõe o algoritmo desenvolvido:

1. Se houver uma restrição temporal estabelecida pelo usuário, uma consulta temporal dever ser executada considerando seus parâmetros temporais. Neste sentido, o parâmetro param_Temporal armazena em sua primeira posição o tempo inicial e na segunda o tempo final, podendo ter apenas um destes, ou os dois, determinando a busca por um período.

   1.1. O método execute_Consulta_Temporal, Figura 49, tem como objetivo montar a estrutura da consulta temporal conforme estrutura especificada, Figura 38, e executá-la.

   1.2. Para realizar a junção das tabelas que compõem a consulta temporal utiliza-se a operação busca do metadado Meta_Juncao passando como parâmetros o fato que deverá ser consultado a dimensão de tempo (inicial e final) e a ainda a dimensão que determina o tipo do fato. Como retorno deste mapeamento tem-se todas as junções necessárias;

2. Mapear cada um dos parâmetros qualitativos na instância do metadado Meta_Qual correspondente. Este passo tem por objetivo definir os atributos, dimensões e valores do DW referentes à seleção do usuário. Como saída deste mapeamento obtém-se um objeto, o_Qual, para cada parâmetro qualitativo. Cada objeto o_Qual é composto pelos seguintes campos: qual_Str, atrib_User, dim_User e val_User. O objeto o_Qual é gerado através do método execute_Meta_Qual, Figura 50. Um conjunto destes objetos que armazenam os dados dos parâmetros qualitativos é denominado no algoritmo de Cj_Qual. A Figura 46

exemplifica o resultado deste passo supondo os seguintes parâmetros qualitativos: nome da versão e tecnologia Java.

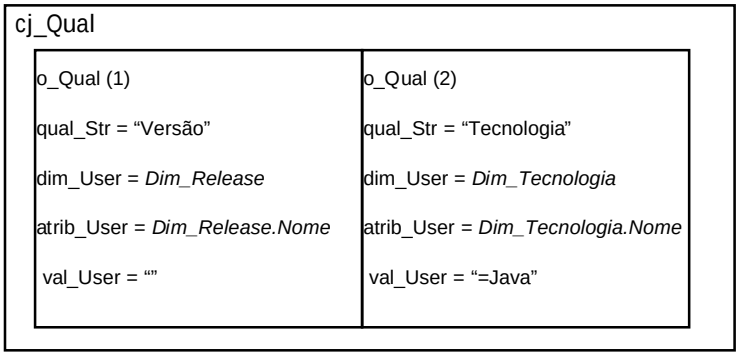

Figura 46: Exemplo algoritmo passo 1.

3. Mapear parâmetros quantitativos em instâncias do metadado Meta_Metrica. No algoritmo, cj_Metrica é um conjunto de conjuntos (cj_Metrica_Base). Cada subconjunto refere-se a uma métrica (geralmente derivada) selecionada pelo usuário. Por sua vez, cada subconjunto (cj_Metrica_Base) contém como elementos as métricas base necessárias a seu cálculo (o_Metrica_Base). Um o_Metrica_Base é composto pelos seguintes campos: metrica_Str, metr_Base_Text, metrica_Base, dim_Metr_Base, atrib_Metr_Base, val_Metr_Base, tipo, dim_Tipo_Fato, val_Tipo_Fato e fato. A Figura 47 exemplifica este passo supondo ser passado o seguinte parâmetro quantitativo: Variação de Esforço – BR, para a qual são necessárias as métricas base ER e EBR. Caso houvesse mais de uma métrica selecionada pelo usuário, haveria outros cj_Metrica_Base.

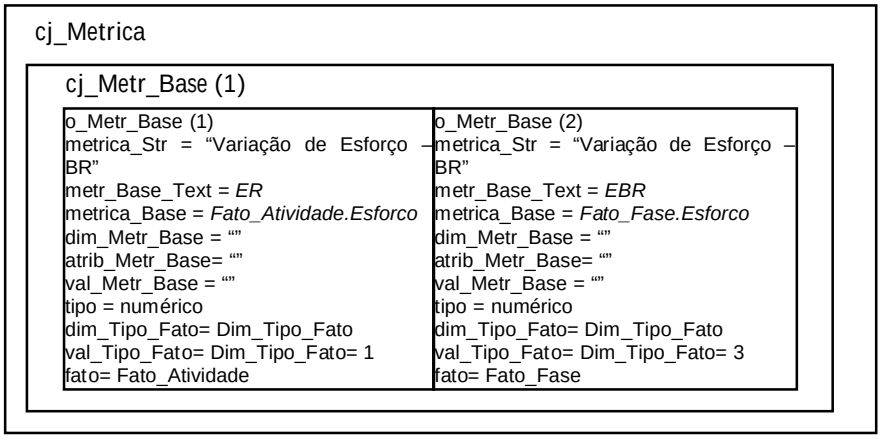

Figura 47: Exemplo algoritmo passo 2.

De posse dos componentes necessários à montagem da estrutura SQL definida para métricas base, os próximos passos do algoritmo visam montar cada consulta componente e a respectiva consulta consolidada.

4. Para cada uma das métricas selecionadas pelo usuário, selecionar o respectivo conjunto de métricas base e;

   4.1. Definir as consultas componentes necessárias, Figura 51, este envolve:

      4.1.a) Definir a função da métrica base. Para tal, desenvolveu-se o método funcao_Metr_Base, Figura 52, que recebe como parâmetro cj_Qual e o tipo da métrica base e retorna funcao_Metrica_Base. Dependendo do atributo qualitativo selecionado pelo usuário e do tipo da métrica base o método pode retornar: a função AVG, se selecionado projetos (apresenta-se média de versões) e tipo “int” (poderia ser “Date”, neste caso não teria função), SUM para qualquer atributo qualitativo e tipo “int” e vazio se o tipo da métrica base for “Date”;

      4.1.b) Buscar as condições de junção de tabelas necessárias consulta da métrica base que está sendo montada com base no metadado Meta_Juncao. Utiliza-se a operação busca do metadado Meta_Juncao passando como parâmetros todas as dimensões selecionadas pelo usuário, armazenadas no objeto o_Qual, o fato que deverá ser consultado e o fato com o tipo de tabela fato a ser consultada (estimada ou real). Como retorno deste mapeamento tem-se um vetor com todas as junções necessárias denominado cj_juncao;

      4.1.c) Escreve-se a consulta. Para tal, definiu-se o método *execute_Consulta_Base,* Figura 53, que recebe como parâmetros cj_Qual, o_Metrica_Base, funcao_Metrica_Base, consulta_Temporal e cj_juncao e retorna uma consulta SQL. O método *execute_Consulta_Base* tem como objetivo disponibilizar todos os parâmetros de entrada conforme a estrutura SQL definida em Figura 42. Após, adiciona-se à consulta retornada em uma estrutura denominada componente_consulta.

Executado o método *execute_Consulta_Base* tem-se como retorno todas as consultas componentes necessárias a consulta para a métrica derivada corrente. Neste sentido, retorna-se ao algoritmo principal, onde os próximos passos deste visam montar a consulta para a

métrica derivada. Para tal, inicialmente deve-se buscar da função da métrica derivada que será utitilizada.

4.2. Buscar a função para o cálculo de métrica derivada, com base no metadado Meta_Funcao_Metrica. Passa-se como parâmetros o atributo Metrica_Str de qualquer um dos objetos pertencentes ao cj_Metrica_Base corrente (neste caso o algoritmo seleciona o primeiro objeto).

4.3. O método, Figura 54, escreve a consulta a métrica derivada conforme a Figura 43. Para tal, tem-se como apoio todas as consultas componentes previamente executadas e armazenadas. Este método tem como objetivo reunir as consultas componentes armazenadas no sentido de formar uma única consulta SQL que retornará o resultado a camada de apresentação. Abaixo, as atividades que estão inclusas neste algoritmo.

4.3.1. Percorre todas as posições de cada consulta componente, captura-se o campo Atrib_Dim_User da primeira consulta componente uma vez que este representa a seleção do usuário e se repete em todas as consultas componentes, após concatena-se a consulta à função da métrica derivada retornada, estes pertencerão a cláusula *Select* da consulta derivada gerada.

4.3.2. A cláusula *From* da consulta final é composta dos nomes de cada componente armazenado.

4.3.3. Escreve a consulta da métrica derivada.

4.3.4. Adicionar a estrutura da métrica derivada em uma estrutura denominada cj_consulta_Metr_Derivada;

Após retorna-se ao passo 4 a fim de verificar se existe um novo conjunto de métricas base a ser consultado. Caso não houver, retorna-se como resultado deste algoritmo a estrutura da métrica derivada para que seja executada no DW e apresentada a camada de apresentação.

```
Algoritmo: execute_Principal
Entrada: param_Temporal, param_Dimensao, param_Metrica;
Saída: cj_consulta_Metr_Derivada;

(1)Se param_Temporal <> ""
consulta_Temporal = execute_Consulta_Temporal(param_Temporal);

(2)Para cada p em param_Dimensao
/* para cada seleção qualitativa do usuário
        cj_Qual.add(execute_Meta_Qual(p))

(3)Para cada n em param_Metrica
/* para cada métrica selecionada pelo usuário busca o conjunto de dados correspondente
      cj_Metrica.add(Meta_Metrica.busca(n))

/*laço que percorre todas as métricas selecionadas pelo usuário
(4)Para cada cj_Met_Base em cj_Metrica.size()

        /* Método que monta todas consultas componentes para uma métrica derivada
        (4.1)consulta_componente = monta_Consultas_Componente(cj_Met_Base, cj_Qual,
        consulta_Temporal);

        /* pego um objeto do conjunto de métricas base, para pegar o nome da métrica
        (4.2)o_Met_Base = cj_Met_Base(1)
        metrica= o_Met_Base.Metrica_Str

        (4.3)cj_Metrica_Derivada= monta_Consulta_Derivada(consulta_componente, metrica)
return (cj_Metrica_Derivada);
```

Figura 48: Algoritmo execute_Principal.

```
Algoritmo: execute_Consulta_Temporal
Entrada: param_temporal;
Saída: consulta_Temporal;
consulta = "";

(1.1)consulta.concat("CREATE VIEW Consulta_Temporal AS
                SELECT Fato_Release.Id_Release
                FROM Dim_Tempo as Inicio, Dim_Tempo as Fim, Fato_Release, Dim_Tipo_Fato
                WHERE ");

        /* se houver restrição temporal de tempo inicial
        Se (param_Temporal.get(1).toString())<> ""
           consulta.concat("(Inicio.Data > = CONVERT (DATETIME,"+ param_Temporal.get(1).toString()+
", 102))) AND");

        /* se houver restrição temporal de tempo final
        Se (param_Temporal.get(2).toString())<> ""
           consulta.concat((Fim.Data < = CONVERT (DATETIME,"+ param_Temporal.get(1).toString()+ ",
102))) AND");

        /* adiciona tipo de fato a ser consultado
        consulta.concat("Dim_Tipo_Fato.Tipo = 1 AND");

        /* realiza junção com tempo inicial e final
        (1.2)consulta.concat (Meta_Junçao.busca(Fato_Release, Dim_Tempo).toString()+ "AND");
        consulta.concat (Meta_Junçao.busca(Fato_Release, Dim_Tempo) .toString()+ "AND");

        /* realiza junção com dimensão do tipo fato
        consulta.concat (Meta_Junçao.busca(Fato_Release, Dim_Tipo_Fato) .toString());

        /* consulta temporal montada; executa-se no DW, retornando o objeto consulta_Temporal
        /* necessidade de identificar as versões presentes na restrição temporal
        consulta_Temporal = execute(consulta);

        return (consulta_Temporal);
```

Figura 49: Algoritmo execute_Consulta_Temporal.

```
Algoritmo: execute_Meta_Qual
    Entrada: param_Dimensao;
    Saída: o_Qual;

    /* separa a string do param dimensão em atributo e valor
    Array array_Aux = param_Dimensao.split(“:”)
    param_Dim_user= array_Aux.get(1)
    param_Val_user= array_Aux.get(2)

    o_Qual = Meta_Qual.busca(param_Dim_user)

    /* Se restrição de valor
    Se param_Val_user <> null
       o_Qual.Val_User =  o_Qual.Dim_User + “=“ + param_Val_User

    return (o_Qual);
```

Figura 50: Algoritmo execute_Meta_Qual.

```
Algoritmo: monta_Consultas_Componente
Entrada: cj_Metrica_Base, cj_Qual, consulta_Temporal;
Saída: consulta_Componente;
    //Monta a consulta de cada uma das métricas base necessária a uma métrica derivada
    Para cada o_Metrica_Base de cj_Met_Base.size()

            /*o tipo da métrica base determina a função
            tipo= o_Metrica_Base.Tipo
            /*retorna uma função por métrica base (AVG, SUM ou vazio)
(4.1.a)     funcao_Metrica_Base= funcao_Metr_Base(tipo, cj_Qual)

           //* junção das tabelas de uma métrica base, um fato por métrica_Base
             fato = o_Metrica_Base.Fato
            //* com cada atributo selecionado pelo usuário
(4.1.b)     Para cada k de cj_Qual.size()
                  o_Qual = cj_Qual.get(k)
                 cj_juncao.add(Meta_Juncao.busca(fato, k.Dim_User))

            //* com a dimensões da métrica base também (Dim_Metrica_Base e Dim_Tipo_Fato)
            Se o_Metrica_Base.Dim_Met_Base <> null
                       cj_juncao.add(Meta_Juncao.busca(fato,o_Metrica_Base.Dim_Met_Base))
                      cj_juncao.add(Meta_Juncao.busca(fato,o_Metrica_Base.Dim_Tipo_Fato))

           //* aqui já tenho todos os componentes de consulta de uma métrica base pronta, agora
        montar a consulta p/ uma métrica base
(4.1.c) consulta_Componente.add(execute_Consulta_Base(cj_Qual,o_Metrica_Base,
        funcao_Metrica_Base, cj_juncao, consulta_Temporal)

return consulta_Componente();
```

Figura 51: Algoritmo monta_Consultas_Componente.

```
Algoritmo: funcao_Metr_Base
Entrada: tipo, cj_Qual;
Saída: funcao_Metr_Base;
funcao_Metr_Base =  “”

/*se o usuário selecionar projeto deve-se fazer uma média das versões, por isso AVG.
Para cada i de cj_Qual.size()
        o_Qual = cj_Qual.get(i)
        dim_user=  o_Qual.Dim.User.toString()
            Se dim_user = “Dim_Projeto”
               funcao_Metr_Base = “AVG”

/*se tiver item selecionado, a métrica for numérica e não tiver projeto selecionado
Se cj_Qual.size()>1 e tipo = numérico e funcao_Metr_Base = “”
        funcao_Metr_Base = “SUM”

return (funcao_Metr_Base);
```

Figura 52: Algoritmo execute_Funcao_Metr_Base.

```
Algoritmo: execute_Consulta_Base
Entrada: cj_Qual, o_Metrica_Base, funcao_Metrica_Base, cj_juncao, id_Int_Analise;
Saída: consulta;
        /*percorre o cj_Qual para armazenar cada linha
        Para cada i em cj_Qual.size()
         /* armazena uma linha do cj_Qual com dimensão, atributo e valor selecionado pelo usuário
             o_Qual = cj_Qual.get(i)
            consulta.concat("CREATE VIEW"+  o_Qual.metr_Base_Text + "AS"
                             SELECT FROM  WHERE GROUP BY ")
            /* como pode ter mais de uma dimensão, atrib, valor selecionado armazeno cada um em uma
string   para passar para a consulta
             dimensao_aux.concat (o_Qual.Dim_User + "," )
             atributo_aux.concat (o_Qual.Atrib_User + "," )
              valor_aux.concat(o_Qual.Val_User)

        /*percorre o conjunto juncao para armazenar cada linha
        Para cada i em cj_juncao.size()
                /* armazena uma linha do param_Qual com dimensão, atributo e valor selecionado pelo
                usuário
                o_juncao = cj_juncao.get(junção (i))
                /* como pode ter mais de uma junção
                juncao_aux.concat (o_juncao.Junção + " AND " )

        /*percorre todos os projetos que estão no intervalo de análise
        Para cada i em consulta_Temporal.size()
        /*para cada projeto do intervalo compara c/ o id projeto fato em questao
                int_Analise_Aux.concat(o_Metrica_Base.Fato.toString()+ ".Id_Projeto = "+
        consulta_Temporal.get(i) )


                /* como estamos fazendo para uma métrica base é tudo unitário
                consulta.concat("SELECT"+ atributo_aux.toString()+ ",")

                Se o_Metrica_Base.Atrib_Met_Base.toString()<> ""
                   consulta.concat(o_Metrica_Base.Atrib_Met_Base.toString()+ ",")

                Se funcao_Metrica_Base.Funcao_Met_Base.toString()<> ""
                   consulta.concat(funcao_Metrica_Base.Funcao_Metrica_Base.toString()+ ","
                   +"("
                   + o_Metrica_Base.Metrica_Base.toString()
                   +")"+ "AS "
                   + o_Metrica_Base.Met_Base_Text
                   "FROM" +dimensao_aux.toString()+ ","
                   + o_Metrica_Base.Fato.toString()+ ",")

                Se o_Metrica_Base.Dim_Met_Base.toString()<> ""
                   consulta.concat(o_Metrica_Base.Dim_Metrica_Base.toString()+ ",")

                Se funcao_Metrica_Base.Dim_Met_Base.toString()<> ""
                   consulta.concat(funcao_Metrica_Base.Dim_Met_Base.toString()+ ","
                   + "Dim_Status"
                   "WHERE"+" Dim_Status.Status = "Fechado" AND ")

                Se valor_aux.toString()<> ""
                   consulta.concat(valor_aux.toString()+ " AND ")

                Se o_Metrica_Base.Val_Met_Base.toString()<> ""
                   consulta.concat(o_Metrica_Base.Val_Met_Base.toString()+ " AND "

                Se int_Analise_Aux ()<> ""
                   consulta.concat(int_Analise_Aux.toString()+ " AND "

                  + junção_aux.toString()+

                  "GROUP BY"+ atributo_aux.toString()+ ")


return (consulta);
```

Figura 53: Algoritmo execute_Consulta_Base.

```
Algoritmo: monta_Consulta_Derivada
Entrada: consulta_componente, metrica;
Saída: cj_consulta_Metr_Derivada;
select, from = “”;

(4.3.1)Para cada k em consulta_componente.size()
        componente = consulta_componente.get(k)
        select = componente.Atrib_Dim_User
        //adiciona a função da métrica derivada
        select.concat(”,” + metrica)
        (4.3.2)from.concat(componente.nome)

(4.3.3)consulta_Metr_Derivada = (“SELECT”+ select + “FROM” + from  )
(4.3.4)cj_consulta_Metr_Derivada.add (consulta_Metr_Derivada)

return (cj_consulta_Metr_Derivada)
```

Figura 54: Algoritmo monta_Consulta_Derivada.

8.2.4 Implementação Componente de Análise

O componente de análise não foi completamente desenvolvido, remanescendo a integração do algoritmo redirecionador de consultas à interface e ainda, a implementação do *dashboard* definido. A proposta utiliza-se de linguagem Java para seu desenvolvimento.

8.3 Considerações sobre o Componente de Monitoramento

O componente de monitoramento difere em muitos aspectos do componente de análise. O monitoramento tem como um de seus principais objetivos detectar desvios rapidamente e prover a tomada de ação em projetos em desenvolvimento, enquanto que o componente de análise possibilita ao usuário analisar projetos já concluídos.

As métricas utilizadas pelo componente de monitoramento têm o propósito de possibilitar ao usuário visualizar a performance de projetos quanto aos custos e prazos inicialmente planejados. Neste sentido, métricas específicas para este fim, como o conjunto de métricas denominadas de *Earned Value* [SOL01] possibilitam o acompanhamento dos custos e prazos do projeto durante todo o seu desenvolvimento. Métricas de análise, se aplicadas neste contexto, não agregam valor as informações disponibilizadas uma vez que desvios de performance somente seriam detectadas ao final dos valores estimados. As métricas de monitoramento permitem o controle do que deve ser ou estar sendo executado em cada momento do projeto.

Devido à urgência de detecção de desvios em projetos em andamento no componente de monitoramento, diferente do componente de análise, a latência entre os dados do ambiente transacional e a informação disponibilizada ao usuário deve ser a menor possível. Neste sentido, o componente de monitoramento tende a utilizar-se das informações armazenadas no

DSA evitando o tempo necessário para que os dados sejam armazenados no DW provendo os dados mais rapidamente aos usuários finais.

A urgência em se detectar desvios de performance também deve ser considerada pelos mecanismos que disponibilizam a informação ao usuário final. Assim, além dos mecanismos já providos pelo componente de análise (gráficos, indicadores e *dashboards*) o componente de monitoramento deve dispor de alertas adaptados aos perfis de usuários, ou seja, para um determinado usuário o melhor alerta pode ser um e-mail enquanto que para outro pode ser uma mensagem SMS.

Acredita-se que a camada de apresentação definida poderia comportar facilmente o componente de monitoramento. Contudo, deveria se passar a considerar a estrutura do DSA que compõe o DW e então seria necessário definir novas estruturas de consultas SQL que visassem as métricas de monitoramento a serem comportadas pelo algoritmo redirecionador de consultas, como também novos recursos analítico como mensagens instantâneas e por mail que poderiam automaticamente serem disparadas alertando aos usuários organizacionais quanto a desvios de performance. O componente de monitoramento não foi especificado devido a limitação de tempo.

9 RELATO DE EXPERIÊNCIA

Este capítulo descreve alguns aspectos de implementação em produção na organização alvo do estudo de caso. Após, relata a experência da adoção de um ambiente de Data Warehousing para análise do PDS através de seu Programa de Métricas, ressaltando os benefícios percebidos.

9.1 Implementação em Produção

O agendamento da avaliação CMM nível 3 da organização alvo programado para novembro de 2005 fez com que esta necessitasse de um protótipo operacional do ambiente de *Data Warehousing* para análise de PDS a partir de maio daquele ano. Assim, a organização implementou uma versão modificada do modelo analítico proposto, a qual é acessada através de uma ferramenta de *Business Intelligenge* (BI), customizada de acordo com os principais requisitos analíticos. Assim, mesmo que esta implementação não tenha sido desenvolvida pela aluna, a mesma seguiu os requisitos e especificações definidos neste trabalho.

O modelo analítico em produção difere em poucos aspectos do modelo apresentado no Capítulo 7. As principais diferenças referem-se à eliminação da dimensão Dim_Tipo_Fato e à inclusão de dados agregados em tabelas de diferentes granularidades. Muitas das decisões tomadas nesta implementação consideram as limitações da ferramenta de BI adquirida, a qual é totalmente baseada em OLAP. Procedimentos de ETL semi-automatizados também foram desenvolvidos para carga dos dados no DW. Esta implementação utiliza recursos Microsoft, a saber, MS SQL Server 2000 e MS QSL Server DTS.

A ferramenta de BI permite o acesso ao repositório através de interface WEB, e oferece ambiente tradicional de OLAP. Este é composto de uma tabela, denominada *Pivot Table*, um *browser* navegador que lista todas as dimensões, fatos e atributos que compõem o DW, uma área para geração do gráfico e algumas opções de configurações. O usuário constrói suas consultas selecionando os dados necessários no *browser*, arrastando-os para a *Pivot Table*, e aplicando sobre estas as operações de *drill-down*, *drill-up*, *pivoting*, etc. O usuário ainda tem a possibilidade de gerar gráficos a partir dos dados da *Pivot Table*. A ferramenta foi customizada no sentido de disponibilizar alguns relatórios pré-definidos e *dashboards* que mostram o desempenho de métricas de acordo com o conjunto de indicadores definidos. Três *dashboards* estão disponíveis: organização, projeto e versão.

9.2 Relato de Experiência

Este protótipo possibilitou à organização fazer uso de funcionalidades críticas previstas no ambiente de *Data Warehousing.* Este tornou-se operacional em maio de 2005, e foram carregados dados históricos de projetos desde outubro de 2004. Desde então, cargas bi-semanais são realizadas, e a organização vem utilizando este ambiente em atividades formais e não formais. Entre as atividades formais encontram-se as diversas reuniões de acompanhamento de projetos que ocorrem mensalmente. Estas são: PMR (*Project Management Review*), que visa prover visibilidade da performance do projeto a seus gerentes; PFR (*Project Formal Review*), que visa prover visibilidade do projeto ao cliente; e, finalmente, SMR (*Senior Management Review*), que visa dar visibilidade ao gerente sênior da organização sobre a performance dos projetos. O ambiente de *Data Warehousing* agilizou o preparo destas reuniões através de visões e relatórios pré-definidos, também resultando em uma maior credibilidade sobre os resultados apresentados. A conquista deve-se ao fato de os dados apresentados nestas reuniões serem suportados pelo DW desenvolvido.

Em atividades não formais, utiliza-se este ambiente na investigação de áreas de deficiências, buscando melhorar a qualidade do processo e do produto gerado; na geração de estimativas mais precisas, focada em dados históricos; e ainda, na solidificação da credibilidade organizacional, entre seus clientes em atividades comerciais, através da ênfase dada à utilização de um sistema que apóia a quantificação da qualidade de seus processos.

O ambiente de *Data Warehousing* possibilitou o uso efetivo e extensivo do PM adotado. Um dos principais benefícios é que o DW proveu o estabelecimento da padronização das métricas organizacionais. Anteriormente à adoção do ambiente de *Data Warehousing,* a organização já realizava algumas coletas e análises de métricas sobre os dados dos projetos organizacionais. Contudo, por diversas razões esta não permitia a comparação entre projetos de forma confiável. Primeiro, projetos capturavam seus dados em momentos distintos, impossibilitanto assim a comparação de projetos relativos ao mesmo período de tempo. Segundo, um mesmo dado era coletado em unidades diferentes, dificultando a comparação destes. E, finalmente, não havia uma padronização quanto às métricas a serem utilizadas, assim projetos poderiam utilizar-se de métricas diferentes para analisar um mesmo aspecto de seu PDS. Neste sentido, a padronização provida pelo modelo analítico desenvolvido garante que todos os dados estarão seguindo o padrão organizacional ao serem armazenados e disponibilizados para análise. Outro benefício diz respeito à consistência e regularidade da coleta dos dados. Através do ambiente provido, cargas referentes aos dados resultantes do

PDS são realizadas regularmente, de maneira semi-automática. Outro benefício deve-se à comunicação do desempenho organizacional e ao acesso aos recursos analíticos provido pela interface de BI através da geração de relatórios, análises pré-definidas, indicadores e através do poder de investigativo disponibilizado pelo manuseio da ferramenta OLAP.

A visibilidade sobre o PDS provida pelo ambiente de *Data Warehousing*, permite à organização quantificar objetivamente, através de métricas, o desempenho e a qualidade do produto de software. Neste sentido, as análises OLAP permitem às equipes de qualidade e de projeto identificar deficiências e pontos de melhorias em seus processos e, baseadas nos processos organizacionais, determinar as ações que poderão ser tomadas.

Um aspecto interessante é que, anteriormente à adoção do ambiente de *Data Warehousing*, a organização já identificava possíveis problemas e soluções em seus processos. Contudo, a incapacidade de quantificar seu desempenho a impossibilitava de planejar estratégias efetivas de melhorias. Assim, a visibilidade provida pelo ambiente possibilitou ao nível estratégico organizacional traçar estas estratégias baseado em medidas que representam com exatidão o desempenho do processo atual. Como exemplo de uma estratégia de melhoria, uma grande quantidade de retrabalho foi identificada em um dado projeto da organização, quantificado através das análises em até 70%. Após, algumas ações tomadas houve uma redução deste percentual para 33% e atualmente trabalha-se neste projeto com a meta de alcançar um índice de 15% em 4 meses.

Através da utilização de gráficos e análises que demonstram a precisão das estimativas de projetos, informações quantitativas sobre o PDS e ainda a qualidade dos produtos desenvolvidos, obteve-se um aumentou da credibilidade da organização junto a seus clientes.

Cabe salientar que este trabalho foi base para a especificação do DW em produção bem como dos novos recursos de análises propostos na customização da ferramenta de BI (e.g. indicadores e dashoards) que a tornaram acessível a um número maior de usuários.

O DW também mostrou benefícios quanto à geração de estimativas de projetos, no sentido de se obter maior precisão em estimativas, baseado em variações de estimativas passadas. Como exemplo, em um determinado projeto, a variação de cronograma no baseline revisado caiu de 8,2% para 2,7%. Contudo, deve-se considerar que o ambiente de *Data Warehousing* não é o responsável pelas melhorias providas ao processo. Este provê apoio à organização na identificação dos problemas, mas as ações que levarão à melhoria propriamente dita é responsabilidade da organização. Assim, considerando o projeto do

exemplo anteriormente citado, pode se concluir que alta variação de cronograma se justificaria pela alta volatilidade de requisitos existente no projeto e, então ações neste sentido foram tomadas tendo como resultado a diminuição também da variação de cronograma.

Outro exemplo foi a identificação de um elevado índice de densidade de defeitos externos em um determinado projeto. Esta constatação resultou em uma ação corretiva que visava o aumento das atividades de *Peer Review*. As conseqüências observadas através desta ação foram: a) aumento na eficiência de remoção de defeitos (de 82 para 93%), b) aumento da densidade de defeitos internos (de 1.11 para 2.43 defeitos por ponto de função) e c) redução da densidade de defeitos externos (de 0.21 para 0.15 defeitos por ponto de função). Concluindo, mais defeitos foram detectados durante o desenvolvimento do produto e menos defeitos foram detectados pelo cliente. Neste sentido, os gastos quanto a defeitos foram reduzidos uma vez que defeitos encontrados no cliente são mais custosos à organização. Através das limitações identificadas no processo de revisão da organização, um próximo passo a ser dado no sentido de aumentar a efetividade das revisões realizadas seria a adoção de métodos formais.

A adoção do ambiente de *Data Warehousing* ainda é recente. Nem todos os projetos da organização conseguem detectar os benifícios de sua utilização. Contudo, observa-se ter havido uma diminuição das variâncias de densidades de defeitos internos e externos, produtividade e variações de cronograma e esforço.

Apesar dos diversos benefícios alcançados até o momento através da utilização do ambiente de *Data Warehousing*, observa-se que ainda não existe uma cultura de acesso espontâneo a este para o acompanhamento de projetos. Grande parte dos usuários organizacionais somente sentem-se confortáveis em acessar os indicadores disponibilizados através dos *dashboards*. Poucos usuários da organização acessam diretamente o ambiente para o preparo de material para as atividades formais (reuniões de acompanhamento) e para visualização de relatórios pré-definidos de um projeto específico. Normalmente, somente usuários que trabalham na equipe de qualidade, possuem conhecimento técnico necessário para utilizar as consultas disponibilizadas através dos cubos OLAP.

Neste sentido, muitas das análises que podem ser realizadas sobre os dados armazenados no DW acabam sendo subestimadas devido à dificuldade de acesso encontrada por usuários sem muito conhecimento técnico. Os recursos da camada de apresentação proposta neste trabalho (Capítulo 8) são embasados na compilação de sugestões e críticas emitidas sobre a interface atual, no estudo de bibliografia atualizada sobre novas tendências

de ferramentas de BI (referenciadas ao longo do texto) e ainda em métodos de como analisar quantitativamente processos, estes adaptados à realidade do DW desenvolvido.

A camada proposta visa sanar as deficiências de ferramentas OLAP como a agregação de dados aditivos e ainda a dificuldade encontrada entre os usuários organizacionais em acessar os dados. Neste sentido, a camada de apresentação proposta disponibiliza recursos analíticos que provêem maior semântica aos dados disponibilizados, facilitando e agilizando as análises realizadas sobre estes.

10 CONCLUSÕES E TRABALHOS FUTUROS

O presente trabalho centra-se na disponibilização de dados para análise do PDS providos através de um modelo multidimensional e da especificação de recursos analíticos fornecidos através de uma camada de apresentação. Organizações de desenvolvimento de software necessitam cada vez mais que o desempenho de seus PDS seja mensurado de forma quantitativa. Considerando esta necessidade, o modelo analítico proveu a padronização da forma como o PM pode ser adotado, disponibilizando análises capazes de representar o desempenho de diversos aspectos do PDS (custo, esforço, satisfação do cliente, entre outros).

O modelo multidimensional do DW, juntamente com os recursos de análise especificados e disponibilizados através da camada de apresentação, possui como adicional o fato de considerar as especificidades do ambiente de PDS. Ferramentas que implementam recursos OLAP e/ou BI voltadas a análises de processos mostram-se insuficientes diante desta realidade. O trabalho desenvolvido considera as características específicas de métricas voltadas à análise de PDS, as necessidades de informações de cada perfil de usuário encontrado em uma organização de software, como também suas diferentes expectativas de análise.

Grande parte das métricas que visam a análise do PDS é representada por razões. Esse fato dificulta a utilização da tecnologia OLAP que somente trabalha com agregações. Assim, o modelo analítico somente armazena as métricas bases e quando necessário, o ambiente calcula as métricas derivadas através de um redirecionador de consultas. Esse aspecto do modelo analítico é transparente ao usuário, uma vez que através do redirecionamento de consultas o usuário tem liberdade de realizar qualquer consulta, apenas respeitando a granularidade do modelo.

Os recursos analíticos e gráficos disponibilizados através da camada de apresentação consideram as necessidades de análise dos diferentes perfis de usuários organizacionais, provendo diferentes níveis de informação (organização, projeto, etc). A semântica aos dados disponibilizados é fornecida através da definição de taxonomias de categorias, baseada nos indicadores de métricas. *Dashboards* utilizam-se dos indicadores de qualidade definidos pela organização com o intuíto de facilitar e agilizar a análise através da rápida detecção de desvios de performance de acordo com a meta organizacional definida (*e.g.* indicador).

O redirecionador de consultas provê uma infra-estrutura que intermedia a consulta gerada na camada de apresentação e sua execução propriamente dita no DW. De forma

transparente ao usuário, este recebe os parâmetros oriundos dos componentes gráficos da interface de usuário e então, a partir de metadados definidos, monta as consultas SQL que serão executadas no DW sobre qualquer tipo de métrica e característica de projeto selecionada pelo usuário.

Quanto à avaliação do trabalho desenvolvido, uma versão modificada do modelo analítico foi implementada na DW na organização alvo do estudo de caso, encontrando-se atualmente em produção. O DW foi um quesito bem avaliado entre aqueles que resultaram na certificação CMM nível 3 no final de 2005, sendo ainda vislumbrado como um instrumento de grande importância para a análise de performance dos projetos organizacionais. Proveu à organização oportunidades de buscar ações de melhorias, lições aprendidas e adoção de boas práticas anteriormente executadas. A camada de apresentação do protótipo operacional demonstrou as limitações de uma interface baseada exclusivamente em recursos OLAP, o que reforçou os princípios utilizados na concepção da camada de apresentação proposta neste trabalho. Contudo, não houve tempo hábil para conclusão de sua implementação, gerando a ausência de avaliação da mesma face aos requisitos que esta busca atender, levando assim a uma limitação importante deste trabalho.

Quanto à estensibilidade do modelo analítico, este pode ser estendido no sentido de comportar uma gama maior de métricas, ou ainda um número maior de características sobre o PDS. Já a estensibilidade para comportar diferentes estruturas de ciclos de vida deve ser cuidadosamente analisada a fim de assegurar que as características da estrutura do processo inicialmente definida sejam mantidas. Quanto aos recursos de análises, diferentes gráficos podem ser agregados e disponibilizados pela camada de apresentação.

Quanto às limitações do modelo analítico, a principal delas refere-se à indisponibilidade de estimativas de esforço em nível de atividade. A existência dessa se faz necessária devido a não existência desta informação em fontes de dados do ambiente transacional. Quanto aos recursos analíticos e a camada de apresentação tem-se como limitação a indisponibilidade de informação segundo um perfil de usuário e a indisponibilidade de dados através da *web*.

Os trabalhos futuros centram-se na definição de um componente de monitoramento igualmente suportado pela camada de apresentação. Devido ao fato do componente de monitoramento ter como objetivo principal detectar desvios rapidamente, este deverá comportar novos recursos analíticos que contemplem essas características. Neste sentido, considerando urgência na detecção de desvios, o componente de monitoramento deve utilizar-

se de dados oriundos do DSA e métricas específicas voltadas ao acompanhamento da performance de projetos em andamento.

Quanto às limitações do modelo analítico, a principal delas refere-se a não disponibilização de estimativas de esforço em nível de atividade. A existência desta se faz necessária devido a não existência desta informação em fontes de dados do ambiente transacional. Já quanto aos recursos analíticos e a camada de apresentação tem-se como limitação a não disponibilização de informação segundo um perfil de usuário e a não disponibilização de dados através da *web*.

Os trabalhos futuros centram-se na definição de um componente de monitoramento igualmente suportado pela camada de apresentação. Devido ao fato do componente de monitoramente ter como objetivo principal detectar desvios rapidamente, este deverá comportar novos recursos analíticos que contemplem a estas características. Neste sentido, considerando urgência dos usuários em detectar desvios, o componente de monitoramento deve utilizar-se de dados oriundos do DSA e métricas específicas voltadas ao acompanhamento da performance de projetos em andamento.

REFERÊNCIAS

[ALO04] G. ALONSO, F. CASATI, H. KUNO, V. MACHIRAJU. **Web Services – Concepts, Architetuctures and Applications**. Berlin: Springer Verlag, 2004. 354p.

[BEC04] K. BECKER, D. RUIZ. An Aggregate-aware Retargeting Algorithm for Multiple Fact Data Warehouses. In: 6th International Conference on Data Warehousing and Knowledge Discovery (DAWAK), 2004, Zaragoza. **Proceedings…** Berlin: Springer - Lecture Notes in Computer Science 3181, 2004. pp.118-128.

[BRO04] M. BROWN.; D. GOLDENSON. Measurement and Analysis: What Can and Does Go Wrong?. In: 10th International Symposium on Software Metrics (METRICS'04), 2004, **Proceedings…** Los Alamitos: IEEE Computer Society Press, 2004. pp. 131-138.

[BUG03] L. BUGLIONE, A. ABRAN. Assessment of Measurement Indicators in Software Process Improvement. Frameworks. In: International Workshop on Software Measurement (IWSM), Montreal, April 2003, pp. 24.

[CAR03] D. CARD. Integrating Practical Software Measurement and the Balanced Scorecard. In: COMPSAC 2003: Design and Assessment of Trustworthy Software-Based Systems, 3-6 November 2003. **Proceedings…** 27th International Computer Software and Applications Conference, 3-6 Nov. 2003, pp. 362 – 363.

[CAS04] M. CASTELLANOS, F. CASATI, U. DAYAL, M-C SHAN. A comprehensive and automated approach to intelligent business processes execution analysis. **Distributed and Parallel Databases**. Heidelberg – Germany, V.16, N. 3, November 2004, pp. 239 – 273.

[CAS05] M. CASTELLANOS, F. CASATI, U. DAYAL, M-C SHAN. iBOM: A Platform for Intelligent Business Operation Management. In: ICDE, Tokyo: IEEE, 2005. Atti del convegno: "ICDE", Tokyo, April 2005. **Proceedings…** 21st International Conference on Data Engineering. Los Alamitos: IEEE Press 5-8 April 2005, pp. 1084-1095.

[CMM02] CMMI **Product TeamCMMI for Systems Engineering/Software Engineering, Version 1.1 Staged Representation** (CMU/SEI-2002-TR-029, ESC-TR-2002-029). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002.

[COS05] COSMOS CORPORATION. **Cosmos Products**. Capturado em: http://www.cosmosm.com/pages/products/products.html, Dezembro, 2005.

[CUN05] V. CUNHA, D. RUIZ. **Em Direção a um Modelo de Referência para o Controle e Acompanhamento de Processos de Desenvolvimento de Software em Arquiteturas Orientadas a Serviços.** Relatório Seminário de Andamento. Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre: PUCRS, 2005.

[DOC05] HEWLETT-PACKARD. "**Documentação Interna – Guia de Métricas Organizacional**", Hewlett-Packard – Latin American Software Operation (LASO). Porto Alegre, Hewlett Packard, 2005.

[EMA97] K. EMAM, W. MELO, J. DROUIN. "**Spice: The Theory and Practice of Software Process Improvement and Capability Determination**", IEEE Computer Society Press, Los Alamitos, CA, vol. 1, pp. 450, 1997.

[FEN97] N. FENTON. "**Software Metrics a rigorous & practical approach**". In: 2nd Ed., PWS Publishing Company, 1997, 638p.

[GOL99] D. GOLDENSON, A. GOPAL, T. MUKHOPADHYAY. "Determinants of Success in Software Measurement Programs: Initial Results". In: Software Metrics Symposium, 1999. **Proceedings…** Sixth International 4-6, Boca Raton, FL, USA, November 1999, pp. 10 – 21.

[GOL03] D. GOLDENSON, J. JARZOMBEK, T. ROUT. "Measurement and Analysis in Capability Maturity Model Integration Models and Software Process Improvement". CrossTalk: **The Journal of Defense Software Engineering,** July 2003, pp. 20-24.

[GOL04] M. GOLFARELLI, S. RIZZI, I. CELLA. "Beyond Data Warehousing: what's next in Business Intelligence?" In: DOLAP 04. **Proceedings…** 7th ACM international workshop on Data warehousing and OLAP, Washington, DC, USA, November 12 - 13, 2004, pp.1-6.

[GRI04] D. GRIGORI, F. CASATI, M. CASTELLANOS, U. DAYAL, M. SAYAL, M. SHAN. "Business Process Intelligence". **Computers in Industry**, Amsterdan – The Netherlands, V.53, n.3, Elsevier Science Publishers April 2004, pp. 324-343.

[HAC03] R. HACKATHORN. "**Factors for Implementing Active Data Warehousing**". Capturado em: http://www.datawarehouse.com., Junho 2003.

[HAY03] S. HAYNES R. "Institutional Metrics for the United States Marine Corps." In: Conference on System Science (HICSS-36), 2003, 231p.

[HEL02] M. HELLINGER, S. FINGERHUT. Business Activity Monitoring: EAI meets Data Warehousing. **eAI Journal**, July 2002, pp.18-21.

[IEE91] ANSI/IEEE Std 610.12-1990, **IEEE Standard Glossary of Software Engineering Terminology**, February 1991.

[IEE98] ANSI/IEEE Std. 1061-1998, **IEEE Standard for a Software Quality Metrics Methodology,** May 1998.

[INM97] W. INMON H. **Como construir o Data Warehouse**. Rio de Janeiro, Ed. Campus, 1997, 404p.

[KAN04] C. KANER, W. P. BOND. *Software* Engineering Metrics: What do they measure and how do we know?. In: 10th International *Software* Metrics Symposium (Metrics 2004), Late Breaking Papers, 2004, Chicago – IL – USA. **Proceedings…** Los Alamitos: IEEE Computer Society Press, 2004. pp.1-12

[KIM98] R. KIMBALL. **Data Warehouse Toolkit**. São Paulo, Makron Books, 1998, 520p.

[LAZ95] S. LAZZARINI. Estudos de Caso: Aplicabilidade e Limitações do Método para Fins de Pesquisa. **Brasil: Economia & Empresa.** vol. 2 n 4, p 60, dezembro 1995.

[MAR03] W. MARTIN, R. NUßDORFER**. Pulse Check: Operational, Tactical and Strategic BPM.** CSA Consulting GmbH White Paper, 2003.

[MIC05] MICROSOFT CORPORATION. **Microsoft SQL Server.** Capturado em: http://www.microsoft.com/sql/solutions/bi/default.mspx, dezembro, 2005.

[NOV04] T. NOVELLO C.. **Documentação Interna – Proposta de um Modelo Multimensional para Análise de Dados de Projetos através de Métricas e Medidas.** Latin American Software Operation (LASO), Porto Alegre, janeiro 2004.

[NOV04a] T. NOVELLO C.. **Documentação Interna – Levantamento de Requisitos**. Latin American Software Operation (LASO), Porto Alegre, novembro 2004.

[NOV04b] T. NOVELLO C., V. CUNHA. **Documentação Interna – Comparação entre as Medidas de Processos Solicitadas pela Equipe CMM e o Projeto SIAPV**. – Latin American Software Operation (LASO), Porto Alegre, dezembro 2004.

[NOV05] T. NOVELLO C., K. BECKER. Data Warehouse voltado ao Monitoramento de Processos de Desenvolvimento de Software através de Métricas. In: I Escola Regional de Banco de Dados (ERBD), 2005, Porto Alegre. **Anais...** Porto Alegre: UFRGS, 2006. p. 58-63.

[OLI99] M. OLIVER. **A Framework for Automating Metrics Collection in a Software Development Organization**. Technical Report Integra TechSoft Pvt. Ltd., 1999.

[OTL01] D. OTLEY. Extending the Boundaries of Management Accounting Research: Developing Systems for Performance Measurement, **British Accounting Review**, vol. 33, p. 243-261, 2001.

[ORA05] ORACLE CORPORATION. **Oracle Express Server**: Delivering OLAP to the Enterprise. Capturado em: http://www.oracle.com/solutions/business_intelligence/olap.html, dezembro, 2005.

[PAL03] E. PALZA, C. ABRAN A. Establishing a Generic and Multidimensional Measurement Repository in CMMI context. 28th Annual IEEE/NASA Software Engineering Workshop, Greenbelt, MD, USA, 2003. **Proceedings** of the 28th Annual NASA Goddard Software Engineering Workshop (SEW'03), Greenbelt (Maryland, USA), December 3-4, 2003, p.12-20.

[PAU93] M. PAULK, B. CURTIS, M. CHRISSIS, C. WEBER. **Capability Maturity Model for Software.** Version.1.1 (CMU/SEI-93-TR-024, ADA 263403). Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute, 1993. http://www.sei.cmu.edu/publications /documents/93.reports/93.tr.024.html.

[PMB00] PMBOK. PMI Standards Committee. **A Guide to the Project Management Body of Knowledge,** PMI Publishing Division, Philadelphia, USA, 2000.

[PRE01] R. PRESSMAN. **Software Engineering:** a practitioner's approach. Singapore: McGraw-Hill International Editions, 2001, 888p.

[POE98] V. POE, P. KLAUER, S. BROBST. **Building a Data Warehouse for Decision Support**. New Jersey, Prentice Hall PTR, 1998, 285p.

[RUI05] D. RUIZ, K. BECKER, T. NOVELLO C., V. CUNHA. A data warehousing environment to monitor metrics in software development processes. In: International Workshop on Business Process Monitoring & Performance Management (BPMPM 2005), 2005, Copenhagen. **Proceedings…** of the 16th Workshop on Database and Expert Systems Applications (DEXA05). Los Alamitos : IEEE Press, 2005. p. 936-940.

[SAY02] M. SAYAL, F. CASATI, U. DAYAL, M. SHAN C. Business Process Cockpit, In: **Proceedings…**of the VLDB'02, Hong Kong, China, August 2002, pp. 880-883, 2002.

[SEI96] SEI, Software Engineering Institute. CMMI for Software Engineering (CMMI-SW, V1.1), Staged Representation. Pittsburgh: Carnegie Mellon University and Software Engineering Institute, 2002. Capturado em: http://www.sei.cmu.edu/pub/documents/02.reports/pdf/02tr029.pdf.

[SEL04] R. SELBY, W. Software Requirements Metrics Provide Leading Indicators in Measurement-Driven Dashboards for Large-Scale Systems. 19th International Forum on COCOMO and Software Cost Modeling, Los Angeles, CA, October 26-29, 2004.

[SOL01] P. SOLOMON. Practical Software Measurement, Performance-Based Earned Value. Cross Talk, **The Journal of Defense Software Engineering,** September, 2001, p. 25-29,.

[SOM04] I. SOMMERVILLE. **Software Engineering.**, 6th Edition, Addison-Wesley, 2000.

[SUB99] V. SUBRAMANYAM, S. SHARMA. HPD - **Query tool on Projects Historical Database**. Hewlett-Packard – Latin American Software Operation (LASO). Porto Alegre, Hewlett Packard, 1999.

[YIN01] R. YIN, K. **Estudo de Caso – Planejamento e Métodos.** Trad. de Daniel Grassi. Brasil: Bookman Companhia Editora. 2001.

[WOU99] M. WOUTERS, K. KEES, J. THEEUWES, K. DONSELAAR. Identification of critical operational performance measures - a research note on a benchmarking study in the transportation and distribution sector. **Management Accounting Research**, vol. 10, p. 439-452, 1999.

APÊNDICE I

Modelo Analítico

1. Dim_Projeto

```
CREATE TABLE [Dim_Projeto] (
    [ID_Projeto] [int] NOT NULL ,
    [Nome] [char] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
    CONSTRAINT [PK_Dim_Projeto] PRIMARY KEY  CLUSTERED
    (
        [ID_Projeto]
    )  ON [PRIMARY]
) ON [PRIMARY]
GO
```

2. Dim_Porte_Projeto

```
CREATE TABLE [Dim_Porte_Projeto] (
    [ID_Porte_Projeto] [int] IDENTITY (1, 1) NOT NULL ,
    [Nome] [char] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
    CONSTRAINT [PK_Dim_Porte_Projeto] PRIMARY KEY  CLUSTERED
    (
        [ID_Porte_Projeto]
    )  ON [PRIMARY]
) ON [PRIMARY]
GO
```

3. Dim_Tecnologia

```
CREATE TABLE [Dim_Tecnologia] (
    [ID_Tecnologia] [int] IDENTITY (1, 1) NOT NULL ,
    [Nome] [char] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
    CONSTRAINT [PK_Dim_Tecnologia] PRIMARY KEY  CLUSTERED
    (
        [ID_Tecnologia]
    )  ON [PRIMARY]
) ON [PRIMARY]
GO
```

4. Dim_Industria

```
CREATE TABLE [Dim_Industria] (
    [ID_Industria] [int] IDENTITY (1, 1) NOT NULL ,
    [Nome] [char] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
    CONSTRAINT [PK_Dim_Industria] PRIMARY KEY  CLUSTERED
    (
        [ID_Industria]
    )  ON [PRIMARY]
) ON [PRIMARY]
GO
```

5. Dim_Tipo_Projeto

```
CREATE TABLE [Dim_Tipo_Projeto] (
    [ID_Tipo_Projeto] [int] IDENTITY (1, 1) NOT NULL ,
    [Nome] [char] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
    CONSTRAINT [PK_Dim_Tipo] PRIMARY KEY  CLUSTERED
    (
        [ID_Tipo_Projeto]
    )  ON [PRIMARY]
) ON [PRIMARY]
GO
```

6. Dim_Cliente

```
CREATE TABLE [Dim_Cliente] (
    [ID_Cliente] [int] IDENTITY (1, 1) NOT NULL ,
    [Nome] [char] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
    CONSTRAINT [PK_Dim_Cliente] PRIMARY KEY  CLUSTERED
    (
        [ID_Cliente]
    )  ON [PRIMARY]
) ON [PRIMARY]
GO
```

7. Dim_Release

```
CREATE TABLE [Dim_Release] (
    [ID_Release] [int] IDENTITY (1, 1) NOT NULL ,
    [Nome] [char] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
    CONSTRAINT [PK_Dim_Produto] PRIMARY KEY  CLUSTERED
    (
        [ID_Release]
    )  ON [PRIMARY]
) ON [PRIMARY]
GO
```

8. Dim_Iteração

```
CREATE TABLE [Dim_Iteracao] (
    [ID_Iteracao] [int] IDENTITY (1, 1) NOT NULL ,
    [Nome] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
    CONSTRAINT [PK_Dim_Iteracao] PRIMARY KEY  CLUSTERED
    (
        [ID_Iteracao]
    )  ON [PRIMARY]
) ON [PRIMARY]
GO
```

9. Dim_Fase

```
CREATE TABLE [Dim_Fase] (
    [ID_Fase] [int] IDENTITY (1, 1) NOT NULL ,
    [Nome] [char] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
    [Sigla] [char] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
    CONSTRAINT [PK_Dim_Fase] PRIMARY KEY  CLUSTERED
    (
        [ID_Fase]
    )  ON [PRIMARY]
) ON [PRIMARY]
GO
```

10. Dim_Atividade

```
CREATE TABLE [Dim_Atividade] (
    [ID_Atividade] [int] IDENTITY (1, 1) NOT NULL ,
    [Nome] [char] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
    [Tipo] [char] (20) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
    CONSTRAINT [PK_Dim_Atividade] PRIMARY KEY  CLUSTERED
    (
        [ID_Atividade]
    )  ON [PRIMARY]
) ON [PRIMARY]
GO
```

11. Dim_Defeito

```
CREATE TABLE [Dim_Defeito] (
    [ID_Defeito] [int] IDENTITY (1, 1) NOT NULL ,
    [Categoria] [char] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
    [Peer_Review] [char] (20) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
    [Severidade] [char] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
    CONSTRAINT [PK_Dim_Defeito] PRIMARY KEY  CLUSTERED
    (
        [ID_Defeito]
    )  ON [PRIMARY]
) ON [PRIMARY]
GO
```

12. Dim_Status

```
CREATE TABLE [Dim_Status_Projeto] (
    [ID_Status_Projeto] [int] NOT NULL ,
    [Status_Projeto] [char] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
    CONSTRAINT [PK_Dim_Status_Projeto] PRIMARY KEY  CLUSTERED
    (
        [ID_Status_Projeto]
    )  ON [PRIMARY]
) ON [PRIMARY]
GO
```

13. Dim_Tempo

```
CREATE TABLE [Dim_Tempo] (
    [ID_Tempo] [int] IDENTITY (1, 1) NOT NULL ,
    [Inicio] [char] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
    [Fim] [char] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
    [Data] [datetime] NOT NULL ,
    CONSTRAINT [PK_Dim_Tempo] PRIMARY KEY  CLUSTERED
    (
        [ID_Tempo]
    )  ON [PRIMARY]
) ON [PRIMARY]
GO
```

14. Dim_Tipo_Fato

```
CREATE TABLE [Dim_Tipo_Fato] (
    [ID_Tipo_Fato] [int] IDENTITY (1, 1) NOT NULL ,
    [Tipo_Fato] [char] (20) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
    CONSTRAINT [PK_Dim_Tipo_Fato] PRIMARY KEY  CLUSTERED
    (
        [ID_Tipo]
    )  ON [PRIMARY]
) ON [PRIMARY]
GO
```

15. Fato_Release

```
CREATE TABLE [Fato_Release] (
    [ID_Projeto] [int] NOT NULL ,
    [ID_Cliente] [int] NOT NULL ,
    [ID_Industria] [int] NOT NULL ,
    [ID_Tipo_Projeto] [int] NOT NULL ,
    [ID_Porte_Projeto] [int] NOT NULL ,
    [ID_Tecnologia] [int] NOT NULL ,
    [ID_Release] [int] NOT NULL ,
    [ID_Status_Projeto] [int] NOT NULL ,
    [ID_Tempo] [int] NOT NULL ,
    [ID_Tipo_Fato] [int] NOT NULL ,
    [Esforco] [real] NULL ,
    [Duracao] [real] NULL ,
    [Requisitos_Aprovado] [real] NULL CONSTRAINT [DF_Fato_Release_Requisitos_ Aprovado] DEFAULT (0),
    [Requisitos_Adicionado] [real] NULL CONSTRAINT [DF_Fato_Release_Requisitos_Adicionado] DEFAULT (0),
    [Requisitos_Modificado] [real] NULL CONSTRAINT [DF_Fato_Release_Requisitos_Modificado] DEFAULT (0),
    [Requisitos_Eliminado] [real] NULL CONSTRAINT [DF_Fato_Release_Requisitos_Eliminado] DEFAULT (0),
    [Tamanho] [real] NULL CONSTRAINT [DF_Fato_Release_Tamanho] DEFAULT (0),
    [Satistacao_Cliente] [real] NULL CONSTRAINT [DF_Fato_Release_Satistacao_Cliente] DEFAULT (0),
    CONSTRAINT [PK_Fato_Release] PRIMARY KEY  CLUSTERED
    (
        [ID_Projeto],
        [ID_Cliente],
        [ID_Industria],
        [ID_Tipo_Projeto],
        [ID_Porte_Projeto],
        [ID_Tecnologia],
        [ID_Release],
        [ID_Status_Projeto],
        [ID_Tempo],
        [ID_Tipo_Fato]
    )  ON [PRIMARY] ,
) ON [PRIMARY]
GO
```

16. Fato_Iteracao

```
CREATE TABLE [Fato_Iteracao] (
    [ID_Projeto] [int] NOT NULL ,
    [ID_Cliente] [int] NOT NULL ,
    [ID_Industria] [int] NOT NULL ,
    [ID_Tipo_Projeto] [int] NOT NULL ,
    [ID_Porte] [int] NOT NULL ,
    [ID_Tecnologia] [int] NOT NULL ,
    [ID_Iteracao] [int] NOT NULL ,
    [ID_Status_Projeto] [int] NOT NULL ,
    [ID_Tempo] [int] NOT NULL ,
    [ID_Tipo_Fato] [int] NOT NULL ,
    [ID_Release] [int] NOT NULL ,
    [Duracao] [real] NULL ,
    [Esforco] [real] NULL ,
    CONSTRAINT [PK_Fato_Iteracao] PRIMARY KEY  CLUSTERED
    (
        [ID_Projeto],
        [ID_Cliente],
        [ID_Industria],
        [ID_Tipo_Projeto],
        [ID_Porte],
        [ID_Tecnologia],
        [ID_Iteracao],
        [ID_Status_Projeto],
        [ID_Tempo],
        [ID_Tipo_Fato]
        [ID_Release],
    )  ON [PRIMARY]
) ON [PRIMARY]
GO
```

17. Fato_Defeito

```
CREATE TABLE [Fato_Defeito] (
    [ID_Projeto] [int] NOT NULL ,
    [ID_Cliente] [int] NOT NULL ,
    [ID_Industria] [int] NOT NULL ,
    [ID_Tipo_Projeto] [int] NOT NULL ,
    [ID_Porte_Projeto] [int] NOT NULL ,
    [ID_Tecnologia] [int] NOT NULL ,
    [ID_Release] [int] NOT NULL ,
    [ID_Iteracao] [int] NOT NULL ,
    [ID_Fase] [int] NOT NULL ,
    [ID_Defeito] [int] NOT NULL ,
    [ID_Status_Projeto] [int] NULL ,
    CONSTRAINT [PK_ Fato_Defeito] PRIMARY KEY  CLUSTERED
    (
        [ID_Projeto],
        [ID_Cliente],
        [ID_Industria],
        [ID_Tipo_Projeto],
        [ID_Porte_Projeto],
        [ID_Tecnologia],
        [ID_Release],
        [ID_Iteracao],
        [ID_Fase],
        [ID_Defeito],
        [ID_Status_Projeto] [int] NULL ,
    )  ON [PRIMARY]
) ON [PRIMARY]
GO
```

18. Fato_Fase

```
CREATE TABLE [Fato_Fase] (
    [ID_Projeto] [int] NOT NULL ,
    [ID_Cliente] [int] NOT NULL ,
    [ID_Industria] [int] NOT NULL ,
    [ID_Tipo_Projeto] [int] NOT NULL ,
    [ID_Porte] [int] NOT NULL ,
    [ID_Tecnologia] [int] NOT NULL ,
    [ID_Fase] [int] NOT NULL ,
    [ID_Status_Projeto] [int] NOT NULL ,
    [ID_Tempo] [int] NOT NULL ,
    [ID_Tipo_Fato] [int] NOT NULL ,
    [ID_Release] [int] NOT NULL ,
    [Duracao] [real] NULL CONSTRAINT [DF_Fato_Fase_Duracao] DEFAULT (0),
    [Esforco] [real] NULL CONSTRAINT [DF_Fato_Fase_Esforco] DEFAULT (0),
    CONSTRAINT [PK_Fato_Fase] PRIMARY KEY  CLUSTERED
    (
        [ID_Projeto],
        [ID_Cliente],
        [ID_Industria],
        [ID_Tipo_Projeto],
        [ID_Porte],
        [ID_Tecnologia],
        [ID_Fase],
        [ID_Status_Projeto],
        [ID_Tempo],
        [ID_Tipo_Fato],
        [ID_Release],
    )  ON [PRIMARY] ,
) ON [PRIMARY]
GO
```

19. Fato_Atividade

```
CREATE TABLE [Fato_Atividade] (
    [ID_Projeto] [int] NOT NULL ,
    [ID_Cliente] [int] NOT NULL ,
    [ID_Industria] [int] NOT NULL ,
    [ID_Tipo_Projeto] [int] NOT NULL ,
    [ID_Porte] [int] NOT NULL ,
    [ID_Tecnologia] [int] NOT NULL ,
    [ID_Fase] [int] NOT NULL ,
    [ID_Iteracao] [int] NOT NULL ,
    [ID_TipoAtividade] [int] NOT NULL ,
    [ID_Status_Projeto] [int] NOT NULL ,
    [ID_Release] [int] NOT NULL ,
    [Esforco] [real] NULL ,
    CONSTRAINT [PK_Fato_Atividade] PRIMARY KEY  CLUSTERED
    (
        [ID_Projeto],
        [ID_Cliente],
        [ID_Industria],
        [ID_Tipo_Projeto],
        [ID_Porte],
        [ID_Tecnologia],
        [ID_Fase],
        [ID_Iteracao],
        [ID_TipoAtividade],
        [ID_Status_Projeto],
        [ID_Release],
    )  ON [PRIMARY] ,
) ON [PRIMARY]
GO
```

APÊNDICE II

Metadados Instanciados

Meta_Qual

Qual_Str	Dim_User	Atrib_User	Val_User
Organização	Dim_Projeto		
Projeto Nome do Projeto	Dim_Projeto Dim_Projeto	Dim_Projeto.Nome Dim_Projeto.Nome	
Tipo	Dim_Tipo_Projeto	Dim_Tipo_Projeto.Nome	
Porte	Dim_Porte_Projeto	Dim_Porte_Projeto.Nome	
Tecnologia	Dim_Tecnologia	Dim_Tecnologia.Nome	
Cliente	Dim_Cliente	Dim_Cliente.Nome	
Unidade Tamanho	Dim_Unidade_Tamanho	Dim_Unidade_Tamanho.Nome	
Indústria	Dim_Industria	Dim_Industria.Nome	
Versão	Dim_Release	Dim_Release.Nome	
Nome da Versão	Dim_Release	Dim_Release.Nome	
Iteração	Dim_Iteracao	Dim_Iteracao.Nome	
Nome da Iteração	Dim_Iteracao	Dim_Iteracao.Nome	
Fase	Dim_Fase	Dim_Fase.Nome	
Nome da Fase	Dim_Fase	Dim_Fase.Nome	
Defeito	Dim_Defeito	Dim_Defeito.Categoria	
Categoria	Dim_Defeito	Dim_Defeito.Categoria	
Severidade	Dim_Defeito	Dim_Defeito.Severidade	
Peer Review Atividade	Dim_Defeito Dim_Atividade	Dim_Defeito.Peer_Review Dim_Atividade.Nome	
Tipo Atividade	Dim_Atividade	Dim_Atividade.Nome	
Tempo Inicial	Dim_Tempo	Dim_Tempo.Data	
Tempo Final	Dim_Tempo	Dim_Tempo.Data	
Nome do Projeto=	Dim_Projeto		
Tipo=	Dim_Tipo_Projeto		
Porte=	Dim_Porte_Projeto		
Tecnologia=	Dim_Tecnologia		
Cliente=	Dim_Cliente		
Unidade Tamanho=	Dim_Unidade_Tamanho		
Indústria=	Dim_Industria		
Nome da Versão=	Dim_Release		
Nome da Iteração=	Dim_Iteracao		
Nome da Fase=	Dim_Fase		
Categoria=	Dim_Defeito		
Severidade= Peer Review=	Dim_Defeito Dim_Defeito		
Tipo Atividade=	Dim_Atividade		
Tempo Inicial=	Dim_Tempo		
Tempo Final=	Dim_Tempo		

*** O metadado Meta_Qual apresenta apenas algumas intâncias afim de exemplificação.

Meta_Metrica

Metrica_Str	Metr_Base_text	Metrica_Base	Dim_Metr_Base	Atrib_Metr_Base	Val_Metr_Base	Tipo	Dim_Tipo_Fato	Fato	Val_Tipo_Fato
Variação de esforço BO	ER	Fato_Atividade.Esforco				Int	Dim_Tipo_Fato	Fato_Atividade	Dim_Tipo_Fato.Tipo= '1'
Variação de esforço BO	EBO	Fato_Fase.Esforco				Int	Dim_Tipo_Fato	Fato_Fase	Dim_Tipo_Fato.Tipo= '2'
Variação de esforço BR	EBR	Fato_Fase.Esforco				Int	Dim_Tipo_Fato	Fato_Fase	Dim_Tipo_Fato.Tipo= '3'
Variação de esforço BR	ER	Fato_Atividade.Esforco				Int	Dim_Tipo_Fato	Fato_ Atividade	Dim_Tipo_Fato.Tipo= '1'
Variação de Cronograma-BO	DTR	*busca só dimensao*	Dim.Tempo	Dim.Tempo.Data		Date	Dim_Tipo_Fato	Fato_Fase	Dim_Tipo_Fato.Tipo= '1'
Variação de Cronograma-BO	DTBO	*busca só dimensao*	Dim.Tempo	Dim.Tempo.Data		Date	Dim_Tipo_Fato	Fato_Fase	Dim_Tipo_Fato.Tipo= '2'
Variação de Cronograma-BO	Duracao_BO	Fato_Fase.Duracao				Int	Dim_Tipo_Fato	Fato_Fase	Dim_Tipo_Fato.Tipo= '2'
Variação de tamanho-BO	TR	Fato_Release.Tamanho				Int	Dim_Tipo_Fato	Fato_Release	Dim_Tipo_Fato.Tipo= '1'
Variação de tamanho-BO	TBO	Fato_Release.Tamanho_Original				Int	Dim_Tipo_Fato	Fato_Release	Dim_Tipo_Fato.Tipo= '2'
Eficiência de Remoção de defeitos	DE	Fato_ Defeito.Defeitos	dbo.Dim_Defeito		dbo.Dim_Defeito.Categoria = “DE”	Int	Dim_Tipo_Fato	Fato_Defeito	Dim_Tipo_Fato.Tipo= '1'
Eficiência de Remoção de defeitos	DI	Fato_ Defeito.Defeitos	dbo.Dim_Defeito		dbo.Dim_Defeito.Categoria = “DI”	Int	Dim_Tipo_Fato	Fato_Defeito	Dim_Tipo_Fato.Tipo= '1'
Custo da Qualidade	Custo_Tot_Proj	Fato_Release.Custo				Int	Dim_Tipo_Fato	Fato_Release	Dim_Tipo_Fato.Tipo= '1'
Custo da Qualidade	Esf_Testes	Fato_Atividade.Esforco	dbo.Dim_Atividade		dbo.Dim_Atividade.Nome= “Testes”	Int	Dim_Tipo_Fato	Fato_Atividade	Dim_Tipo_Fato.Tipo= '1'

Metrica_Str	Metr_Base_text	Metrica_Base	Dim_Metr_Base	Atrib_Metr_Base	Val_Metr_Base	Tipo	Dim_Tipo_Fato	Fato	Val_Tipo_Fato
Custo da Qualidade	Esf_Rev	Fato_Atividade.Esforco	dbo.Dim_Atividade		dbo.Dim_Atividade.Nome= “Revisão”	Int	Dim_Tipo_Fato	Fato_Atividade	Dim_Tipo_Fato.Tipo= '1'
Custo da Qualidade	Esf_Retrab	Fato_Atividade.Esforco	dbo.Dim_Atividade		dbo.Dim_Atividade.Nome= “Retrabalho”	Int	Dim_Tipo_Fato	Fato_Atividade	Dim_Tipo_Fato.Tipo= '1'
Custo da Qualidade	Esf_Qual	Fato_Atividade.Esforco	dbo.Dim_Atividade		dbo.Dim_Atividade.Nome= “Qualidade”	Int	Dim_Tipo_Fato	Fato_Atividade	Dim_Tipo_Fato.Tipo= '1'
Eficiência de Revisão	Esf_Rev	Fato_Atividade.Esforco	dbo.Dim_Atividade		dbo.Dim_Atividade.Nome = “Revisão”	Int	Dim_Tipo_Fato	Fato_Atividade	Dim_Tipo_Fato.Tipo= '1'
Eficiência de Revisão	Defeitos_PR	Fato_Defeito.Defeitos			dbo.Dim_Defeito.Peer_Review =PR	Int	Dim_Tipo_Fato	Fato_Defeito	Dim_Tipo_Fato.Tipo= '1'

***O metadao Meta_Métrica apresentado representa apenas algumas intâncias que compõem o PM definido a fim de exemplificação deste.

Meta_Juncao

Dimensao	Fato	Juncao
dbo.Dim_Projeto dbo.Dim_Projeto	Fato_Release Fato_Iteracao	Fato_Release.ID_Projeto= Dim_Projeto.ID_Projeto Fato_Iteracao.ID_Projeto= Dim_Projeto.ID_Projeto
dbo.Dim_Projeto	Fato_Defeito	Fato_ Defeito.ID_Projeto=Dim_Projeto.ID_Projeto
dbo.Dim_Projeto	Fato_Fase	Fato_Fase.ID_Projeto=Dim_Projeto.ID_Projeto
dbo.Dim_Projeto	Fato_Atividade	Fato_Atividade.ID_Projeto= Dim_Projeto.ID_Projeto
dbo.Dim_Tipo_Projeto	Fato_Release	Fato_Release.ID_Tipo_Projeto= Dim_Tipo_Projeto.ID_Tipo_Projeto
dbo.Dim_Tipo_Projeto	Fato_Iteracao	Fato_Iteracao.ID_Tipo_Projeto= Dim_Tipo_Projeto.ID_Tipo_Projeto
dbo.Dim_Tipo_Projeto	Fato_Defeito	Fato_ Defeito.ID_Tipo_Projeto= Dim_Tipo_Projeto.ID_Tipo_Projeto
dbo.Dim_Tipo_Projeto	Fato_Fase	Fato_Fase.ID_Tipo_Projeto= Dim_Tipo_Projeto.ID_Tipo_Projeto
dbo.Dim_Tipo_Projeto	Fato_Atividade	Fato_Atividade.ID_Tipo_Projeto= Dim_Tipo_Projeto.ID_Tipo_Projeto
dbo.Dim_Porte_Projeto	Fato_Release	Fato_Release.ID_Porte_Projeto= Dim_Porte_Projeto.ID_Porte_Projeto
dbo.Dim_Porte_Projeto	Fato_Iteracao	Fato_Iteracao.ID_Porte_Projeto= Dim_Porte_Projeto.ID_Porte_Projeto
dbo.Dim_Porte_Projeto	Fato_ Defeito	Fato_ Defeito.ID_Porte_Projeto= Dim_Porte_Projeto.ID_Porte_Projeto
dbo.Dim_Porte_Projeto	Fato_Fase	Fato_Fase.ID_Porte_Projeto= Dim_Porte_Projeto.ID_Porte_Projeto
dbo.Dim_Porte_Projeto	Fato_Atividade	Fato_Atividade.ID_Porte_Projeto= Dim_Porte_Projeto.ID_Porte_Projeto
dbo.Dim_Tecnologia	Fato_Release	Fato_Release.ID_Tecnologia= Dim_Tecnologia.ID_ Tecnologia
dbo.Dim_Tecnologia	Fato_Iteracao	Fato_Iteracao.ID_Tecnologia= Dim_Tecnologia.ID_ Tecnologia
dbo.Dim_Tecnologia	Fato_ Defeito	Fato_ Defeito.ID_Tecnologia= Dim_Tecnologia.ID_ Tecnologia
dbo.Dim_Tecnologia	Fato_Fase	Fato_Fase.ID_Tecnologia= Dim_Tecnologia.ID_ Tecnologia
dbo.Dim_Tecnologia	Fato_Atividade	Fato_Atividade.ID_Tecnologia= Dim_Tecnologia.ID_ Tecnologia
dbo.Dim_Cliente	Fato_Release	Fato_Release.ID_Cliente= Dim_Cliente.ID_Cliente
dbo.Dim_Cliente	Fato_Iteracao	Fato_Iteracao.ID_Cliente= Dim_Cliente.ID_Cliente
dbo.Dim_Cliente	Fato_ Defeito	Fato_ Defeito.ID_Cliente= Dim_Cliente.ID_Cliente
dbo.Dim_Cliente	Fato_Fase	Fato_Fase.ID_Cliente= Dim_Cliente.ID_Cliente
dbo.Dim_Cliente	Fato_Atividade	Fato_Atividade.ID_Cliente= Dim_Cliente.ID_Cliente
dbo.Dim_Unidade_Tamanho	Fato_Release	Fato_Release.ID_Unidade_Tamanho= Dim_Unidade_Tamanho.ID_Unidade_Tamanho
dbo.Dim_Unidade_Tamanho	Fato_Iteracao	Fato_Iteracao.ID_Unidade_Tamanho= Dim_Unidade_Tamanho.ID_Unidade_Tamanho
dbo.Dim_Unidade_Tamanho	Fato_Defeito	Fato_Defeito.ID_Unidade_Tamanho= Dim_Unidade_Tamanho.ID_Unidade_Tamanho
dbo.Dim_Unidade_Tamanho	Fato_Fase	Fato_Fase.ID_Unidade_Tamanho= Dim_ Unidade_Tamanho.ID_ Unidade_Tamanho
dbo.Dim_Unidade_Tamanho	Fato_Atividade	Fato_Atividade.ID_Unidade_Tamanho= Dim_Unidade_Tamanho.ID_Unidade_Tamanho
dbo.Dim_Industria	Fato_Release	Fato_Release.ID_Industria= Dim_Industria.ID_Industria
dbo.Dim_Industria	Fato_Iteracao	Fato_Iteracao.ID_Industria= Dim_Industria.ID_Industria
dbo.Dim_Industria	Fato_ Defeito	Fato_ Defeito.ID_Industria= Dim_Industria.ID_Industria
dbo.Dim_Industria	Fato_Fase	Fato_Fase.ID_Industria= Dim_Industria.ID_Industria
dbo.Dim_Industria	Fato_Atividade	Fato_Atividade.ID_Industria= Dim_Industria.ID_Industria
dbo.Dim_Release	Fato_Release	Fato_Release.ID_Release= Dim_Release.ID_Release
dbo.Dim_Release	Fato_Iteracao	Fato_Iteracao.ID_Release= Dim_Release.ID_Release
dbo.Dim_Release	Fato_Defeito	Fato_Defeito.ID_Release= Dim_Release.ID_Release
dbo.Dim_Release	Fato_Fase	Fato_Fase.ID_Release= Dim_Release.ID_Release

Dimensao	Fato	Juncao
dbo.Dim_Release	Fato_Atividade	Fato_Atividade.ID_Release= Dim_Release.ID_Release
dbo.Dim_Iteracao	Fato_Iteracao	Fato_Iteracao.ID_Iteracao= Dim_Iteracao.ID_Iteracao
dbo.Dim_Iteracao	Fato_ Defeito	Fato_ Defeito.ID_Iteracao= Dim_Iteracao.ID_Iteracao
dbo.Dim_Iteracao	Fato_Fase	Fato_Fase.ID_Iteracao= Dim_Iteracao.ID_Iteracao
dbo.Dim_Iteracao	Fato_Atividade	Fato_Atividade.ID_Iteracao= Dim_Iteracao.ID_Iteracao
dbo.Dim_Defeito	Fato_Defeito	Fato_Defeito.ID_Defeito= Dim_Defeito.ID_Defeito
dbo.Dim_Defeito	Fato_Fase	Fato_Fase.ID_Defeito= Dim_Defeito.ID_Defeito
dbo.Dim_Defeito	Fato_Atividade	Fato_Atividade.ID_Defeito= Dim_Defeito.ID_Defeito
dbo.Dim_Fase	Fato_Fase	Fato_Fase.ID_Fase= Dim_Fase.ID_Fase
dbo.Dim_Fase	Fato_Atividade	Fato_Atividade.ID_Fase= Dim_Fase.ID_Fase
dbo.Dim_Atividade	Fato_Atividade	Fato_Atividade.ID_Atividade= Dim_Atividade.ID_Atividade
dbo.Dim_Tempo	Fato_Release	Fato_Release.ID_TempoTermino = Dim_Tempo.Data.ID_Tempo
dbo.Dim_Tempo	Fato_Iteracao	Fato_Iteracao.ID_TempoTermino = Dim_Tempo.Data.ID_Tempo
dbo.Dim_Tempo	Fato_ Defeito	Fato_ Defeito.ID_TempoTermino = Dim_Tempo.Data.ID_Tempo
dbo.Dim_Tempo	Fato_Fase	Fato_Fase.ID_TempoTermino = Dim_Tempo.Data.ID_Tempo
dbo.Dim_Tempo	Fato_Atividade	Fato_Atividade.ID_TempoTermino = Dim_Tempo.Data.ID_Tempo
dbo.Dim_Tipo_Fato	Fato_Release	Fato_Release.ID_Tipo_Fato= dbo.Dim_Tipo_Fato.ID_Tipo_Fato.ID_Tipo_Fato
dbo.Dim_Tipo_Fato	Fato_Iteracao	Fato_Iteracao.ID_Tipo_Fato= dbo.Dim_Tipo_Fato.ID_Tipo_Fato.ID_Tipo_Fato
dbo.Dim_Tipo_Fato	Fato_Fase	Fato_Fase.ID_Tipo_Fato= dbo.Dim_Tipo_Fato.ID_Tipo_Fato.ID_Tipo_Fato

Meta_Funcao_Metrica

Metrica_Str	Função
Variação de esforço BO	(ER- EBO)*100/(EBO)
Variação de esforço BR	(ER- EBR)*100/(EBR)
Variação de Cronograma. BO	(DATEDIFF([day]; DTR; DTBO))*100/Duracao_BO
Variação de Cronograma BR	(DATEDIFF([day]; DTR; DTBR))*100/Duracao_BR
Variação de tamanho BO	(TR-TBO)*100/(TBO)
Variação de tamanho BR	(TR-TBR)*100/(TBR)
Eficiência de Remoção de defeitos	(DI/ (DI + DE))
Custo da Qualidade	((CARV + CFT + CAQ+ CART) / CR)*100
Eficiência de Revisão	Defeitos_PR / Esf_Rev
Produtividade	ER/TR
Variação de Custo BO	(CR- CBO)*100/(CBO)
Variação de Custo BR	(CR- CBR)*100/(CBR)
Volatilidade de Requisitos	((RA+RR + RM)/RAC)
Satisfação do Cliente	AVG(SC)