PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL

FACULDADE DE INFORMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Uma Abordagem Orientada a Serviços para Captura de Métricas de Processo de Desenvolvimento de Software

VIRGINIA SILVA DA CUNHA

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.

Orientador: Prof. Dr. Duncan D. A. Ruiz

Porto Alegre

2006

Pontifícia Universidade Católica do Rio Grande do Sul

Dados Internacionais de Catalogação na Publicação (CIP)

C972a Cunha, Virginia Silva da
Uma abordagem orientada a serviços para captura de
métricas de processo de desenvolvimento de software / Virginia Silva da Cunha. - Porto Alegre, 2006.
139 f.

Diss. (Mestrado) - Fac. de Informática, PUCRS
Orientador: Prof. Dr. Duncan D. A. Ruiz

1. Engenharia de Software. 2. Data Warehouse. 3. Banco de Dados. 4. Informática. I. Título.

CDD 005.1

Ficha Catalográfica elaborada pelo Setor de Processamento Técnico da BC-PUCRS

PUCRS
Campus Central
Av. Ipiranga, 6681 - prédio 16 - CEP 90619-900
Porto Alegre - RS - Brasil
Fone: +55 (51) 3320-3544 - Fax: +55 (51) 3320-3548
Email: bceadm@pucrs.br
www.pucrs.br/biblioteca

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 Orientada a Serviços para Captura de Métricas de Processo de Desenvolvimento de Software***", apresentada por Virginia Silva da Cunha, como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Sistemas de Informação, aprovada em 26/01/2006 pela Comissão Examinadora:

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

Prof. Dr. Jorge Luís Nicolas Audy – PPGCC/PUCRS

Profa. Dra. Karin Becker – PPGCC/PUCRS

Profa. Dra. Renata de Matos Galante – UFRGS

Homologada em 16/04/07, conforme Ata No. 009 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
http: //www.pucrs.br/inf/pos e-mail: ppgcc@inf.pucrs.br

AGRADECIMENTOS

Aos meus pais, Luiz Carlos, Solange, Carlos Alberto e Marcia, minhas irmãs Valéria e Verônica, e meu sobrinho Pedro pelo carinho, incentivo, compreensão, por sempre acreditarem em mim e compreenderem minha ausência. Saibam que vocês são responsáveis pelo que há de melhor em mim.

Ao meu orientador, professor Duncan Ruiz, por toda dedicação, ensinamentos, conselhos e incentivo. Obrigada também pelas críticas que contribuíram para o meu crescimento como pessoa. Saiba que esse trabalho em grande parte, também é mérito teu.

A minha grande amiga e parceira, Taisa Novello que durante esse tempo todo esteve ao meu lado e que sempre me relembrava que no final tudo daria certo. Obrigada!

Aos amigos de longos anos e da PUC, Daia, Tita, Jose, André, Fabio, Guilherme, Diego e Régis, com certeza vocês são parte disso. Com vocês dividi momentos de tristeza e tensão que acabaram se tornando cômicos.

Ao Convênio HP Brasil/ PUCRS por viabilizarem a bolsa de estudos bem como todo o aprendizado fornecido durante esse tempo.

A professora Karin Becker, pela bolsa, apoio e suas críticas construtiva que me fizeram crescer muito como pessoa.

Ao grupo GPIN (Grupo de Pesquisa em Inteligência de Negócio) pelo conhecimento trocado, apoio em diversos momentos, que provaram que na pesquisa se faz grandes amigos.

Ao Programa de Pós-Graduação em Ciência da Computação e a todos os professores dos quais pude conviver durante esse período.

E as demais pessoas que durante esse tempo colaboraram de alguma forma com esse trabalho.

“Somos o que repetidamente fazemos.
A excelência, portanto, não é um feito,
Mas um hábito.”

Aristóteles

RESUMO

As organizações de *software* trabalham com diversos projetos de *software* que se diferenciam tanto pelas ferramentas de gestão utilizadas quanto pela forma que armazenam e controlam suas métricas de acompanhamento. Sendo assim, a inexistência de um repositório centralizado de dados dificulta o acompanhamento dos Processos de Desenvolvimento de *Software* (PDSs) dessas organizações.

Uma das etapas mais cruciais do Processo de Descoberta de Conhecimento em Banco de Dados é o processo de Extração, Transformação e Carga (ETC), pois este tem como finalidade a transformação dos dados brutos, extraídos de diversas fontes, em informações consistentes e de qualidade. Considerando que os PDSs possuem suas especificidades, realizou-se um estudo em um ambiente real e verificou-se que, em termos de ferramentas, são utilizadas desde planilhas eletrônicas (e.g. *MS Excel*) até ferramentas para controle da execução de atividades de projetos (e.g. *MS Project Server, IBM Rational Clear Quest, Bugzilla*). Detectou-se ainda o uso de diferentes modelos de PDS, com ciclos de vida variados para projetos distintos, que se traduzem em formas totalmente diversas de registrar estes projetos, ainda que na mesma ferramenta. Outro problema é que cada uma dessas ferramentas possui um modelo de dados próprio, que não segue padronizações estabelecidas de representação de dados, dificultando assim a extração desses dados. Por conseqüência, o grau de complexidade do processo de ETC, para esta organização, é muito alto.

O modelo proposto neste trabalho tem por mérito tratar, de forma integrada, dois aspectos: 1) a coleta de dados dos projetos de forma não intrusiva, levando em consideração vários tipos de heterogeneidade, 2) a transformação e integração desses dados, proporcionando uma visão organizacional unificada e quantitativa dos projetos. Esses aspectos são tratados utilizando uma arquitetura orientada a serviços.

A abordagem orientada a serviços busca lidar com vários tipos de heterogeneidade, tanto do ponto de vista organizacional (e.g. especializações do Processo de *Software* Padrão da Organização (OSSP – *Organization's Standard Software Process*) que resultam em formas distintas de desenvolvimento e registro de fatos sobre projetos), quanto do ponto de vista técnico (e.g. diferentes ferramentas). Essa heterogeneidade é convenientemente tratada através de serviços que atuam como *wrappers* dos diferentes tipos de extratores, que suporta um ambiente distribuído de desenvolvimento.

Para avaliação da abordagem proposta, foram desenvolvidos três exemplos, que consideram todas essas questões de heterogeneidade: diferentes tipos de projetos, diferentes ciclos de vida, diferentes modelos de gerenciamento e diversas ferramentas de apoio ao acompanhamento.

Palavras Chave: *Data Warehousing*, Arquiteturas Orientadas a Serviços e Processos de Desenvolvimento de *Software*.

ABSTRACT

Software organizations work with several software projects that differ in terms of both the management tools used and the way tracking metrics are stored and controlled. Thus, the lack of a central data repository poses difficulties for the traking of Software Development Processes (SDPs) in these organizations.

One of the crucial steps of the Knowledge Discovery in Databases Process is the process of Extraction, Transformation and Load (ETL). ETL aims to transform the raw data extracted from different fonts into consistent, reliable information. Considering the SDPs specificities, this study was carried out in the real computional environment. It was observed that the tools used range from spreadsheets (e.g. MS Excel) to control tools for the execution of project activities (e.g. MS Project Server, IBM Rational Clear Quest, Bugzilla). Different SDP models with a distinct life cycle for each project are also used, which result in completely different ways to register these projects even when using the same tool. Another problem is that each of those tools has an own data model that does not follow defined data representation standards. Therefore, the extraction of those data becomes a challenging goal to achieve, raising the complexity of ETL processes.

The model proposed in this study introduces a two-integrated approach to deal with the problem: 1) a non intrusive way of data extraction, taking several types of heterogeneities into account, 2) the transformation and integration of these data, providing a unified and quantified organizational view of the projects. These aspects are treated using a service-oriented architecture.

This service oriented architecture tries to deal with several types of heterogeneity, from both the technical (e.g. different tools) and organizational standpoint (e.g. Organization's Standard Software Process Standard specializations that result in distinct ways to develop and register project facts). This heterogeneity is conveniently treated through services that work as wrappers of the different types of extractors and through the support of a distributed development environment.

For the evaluation of the proposed approach, three examples that consider all heterogeneity issues (different types of projects, different life cycles, different management models and several management support tools) were developed.

Keywords: Data Warehousing, Service Oriented Architecture and Software Development Processes.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

AOS	Arquitetura Orientada a Serviços
BO	Base Organizacional
BPMPM	*Business Process Monitoring & Performance Management*
CEBO	Custo *Baseline* Original
CEBR	Custo *Baseline* Revisado
CMM	*Capability Maturity Model*
CMMI	*Capability Maturity Model Integrated*
CNC	Custo de Não Conformidades
CP	Custo de Prevenção
CR	Custo Real
CTP	Custo total do projeto
CTR	Custo Total de Revisão
CTT	Custo Total de Teste
DE	Número defeitos encontrados no cliente
DFBO	Data Final *Baseline* Original
DFBR	Data Final *Baseline* Revisado
DFR	Data Final Real do Projeto
DI	Número defeitos encontrados internamente
DIBO	Data Inicial *Baseline* Original
DIBR	Data Inicial *Baseline* Revisado
DR	Número defeitos encontrados em revisões
DW	*Data Warehouse*
EEBO	Esforço Estimado *Baseline* Original
EEBR	Esforço Estimado *Baseline* Revisado
ER	Esforço Real
ETC	Extração, Transformação e Carga
ETR	Esforço total em revisão
GQM	*Goal/Question/Metric*
HTTP	*Hypertext Transfer Protocol*
IDL	*Interface Definition Language*
ISO	*International Organization for Standardization*
KDD	*Knowledge Discovery in Database*
KLOC	*Kilo Line of Codes*
OSSP	*Organization's Set of Standard Process*
PDS	Processo de Desenvolvimento de *Software*
PDSP	*Project's Defined Software Process*
PF	Ponto de Função
PMHP	Ponto Médio HP
QA	*Quality Assurance*
QIP	*Quality Improvement Paradigm*
RA	Número de Requisitos Adicionados
RM	Número de requisitos modificados
RO	Número de requisitos originais
RR	Número de requisitos removidos
SEI	*Software Engineering Institute*
SMWS	*Solution Management Web Services*
SOAP	*Simple Object Access Protocol*
SPICE	*Software Process Improvement and Capability Determination*
TEBO	Tamanho Estimado *Baseline* Original
TEBR	Tamanho Estimado *Baseline* Revisado
TR	Tamanho Real
TR_KLOC	Tamanho Real em KLOC
UCP	*Use Case Points*
UDDI	*Universal Description, Discovery and Integration*
W3C	*World Wide Web Consortium*
WSDL	*Web Services Description Language*
XML	*Extensible Markup Language*

1 INTRODUÇÃO

1.1 Contexto da Pesquisa

Qualidade de *software* é um processo sistemático que focaliza todas as etapas e artefatos produzidos com o objetivo de garantir a conformidade de processos e produtos, prevenindo e eliminando defeitos [ROC01]. A qualidade dos Processos de Desenvolvimento de *Software* (PDSs) pode ser quantificada através da utilização de um programa de métricas que auxilie as organizações a organizar, monitorar, detectar e prevenir falhas, tanto do processo quanto do produto final. As organizações de desenvolvimento de *software* demonstram grande preocupação com a definição de métricas que representem suas áreas de qualidade. Porém, a simples definição dessas métricas de qualidade não é a garantia de sua utilização, devido à dificuldade de se coletar e apresentar os dados relativos aos diferentes projetos, de forma unificada.

As organizações de *software* buscam formas de gerenciar e acompanhar seus PDSs, seguindo modelos de capacidade[1] de *software* (e.g. *Capability Maturity Model* (CMM), [PAU01], *Capability Maturity Model Integrated* (CMMI), [SEI96], *Software Process Improvement and Capability Determination* (SPICE), [SPI04]). Esses modelos possuem os elementos necessários para tornar um PDS mais eficiente e controlado, pois focam na proposta de sua melhoria contínua, trazendo disciplina e controle no desenvolvimento e manutenção de *software* [SOM01]. No entanto, seguir estes modelos nem sempre é uma tarefa simples. As organizações buscam gradativamente adquirir novas competências e incrementar seu nível de eficiência e maturidade em relação aos diversos processos críticos envolvidos em um desenvolvimento de *software*. Para o CMM e CMMI, cada organização define o seu próprio conjunto de processos padrão (*Organization's Set of Standard Process* (OSSP)), os quais devem ser especializados a fim de considerar as particularidades dos diferentes projetos de desenvolvimento de *software*.

[1] Apesar da tradução mais apropriada para *Capability* ser competência, será usado, neste trabalho a forma capacidade, mais em voga em Ciência da Computação.

Os modelos CMM e CMMI definem que a construção de uma Base Organizacional (BO) é um dos requisitos necessários ao amadurecimento das organizações de desenvolvimento de *software*. Esta base deve prover o armazenamento e a manipulação dos dados relativos aos diferentes projetos, segundo uma visão unificada da organização. A construção de tal base organizacional não é uma tarefa trivial. A grande maioria das organizações de *software* trabalham simultaneamente com projetos que possuem particularidades quanto ao processo de desenvolvimento, ferramentas de gestão adotadas, bem como a forma como geram, armazenam e controlam suas métricas de acompanhamento. Ainda, distintas organizações diferem entre as suas práticas e OSSP, considerando em particular o nível de maturidade em que se encontram. Atualmente, não existe uma infra-estrutura genérica de apoio aos programas de mensuração de organizações de *software* que considere todos estes pontos de variabilidade. Propostas específicas, discutidas no Capítulo 8, podem ser encontradas em [GRE03, CAS04 e GRI04].

Este trabalho propõe um modelo não intrusivo de Extração, Transformação e Carga (ETC) que suporte diversos fatores de heterogeneidade em PDS utilizando padrões orientados a serviços [ALO05], tais como *Extensible Markup Language* (XML), *Simple Object Access Protocol* (SOAP), *Web Services Description Language* (WSDL) *e Universal Description, Discovery and Integration* (UDDI), e utiliza como cenário uma organização de *software* certificada CMM nível 2 que busca certificação CMM nível 3. Este modelo trata a heterogeneidade dos projetos, e das ferramentas de captura e armazenamento dos dados referentes a cada projeto, através de serviços que atuam como *wrappers* dos diversos aplicativos presentes nos PDSs. O armazenamento de dados dos PDSs, pelo ponto de vista organizacional, é realizado através de um modelo analítico multidimensional voltado ao acompanhamento de processos e baseado em métricas da organização. Diversos aspectos apresentados neste trabalho são objetos do artigo aprovado no *1st International Workshop on Business Process Monitoring & Performance Management (*BPMPM'05) intitulado como *A Data Warehousing Environment to Monitor Metrics in Software Development Processes* [RUI05].

A organização de *software* alvo do estudo de caso (Capítulo 6), apresenta heterogeneidade em termos das ferramentas de gestão utilizadas no controle das atividades dos projetos, onde são utilizadas desde planilhas eletrônicas (*MS Excel*) até ferramentas dedicadas a este fim (e.g. *MS Project Server, IBM Rational Clear Quest, Bugzilla*). Verificou-se também que essa organização possui distintos modelos de PDS, os quais apresentam tipos

de projetos distintos e ciclos de vida variados, que resultam em diversas maneiras de registrar seus projetos, mesmo que na mesma ferramenta. Ainda, as ferramentas utilizadas muitas vezes não seguem paradigmas convencionais de representação de dados. Todos esses fatores descritos conduzem a um Processo de ETC com um alto grau de complexidade.

1.2 Questão de Pesquisa

O estudo de um ambiente real, juntamente com o estudo sobre o estado da arte em relação aos fatores que afetam a implementação do processo de ETC para PDS, conduziram a seguinte questão de pesquisa para estudo:

Como capturar dados de execução de processos de desenvolvimento de software tal que seja possível acompanhar tais processos?

Capturar, nesse contexto, significa prover mecanismos de extração não intrusivos e que considerem a heterogeneidade das fontes de dados. Essa captura deve prover uma menor latência dos dados e os processos de limpeza e transformação devem buscar garantir dados sem inconsistências. Estes dados, quando atualizados, consistentes e de qualidade, podem ser utilizados pelos gestores no acompanhamento de seus PDS. Para tratar a heterogeneidade das fontes, observa-se que a utilização de conceitos de computação orientada a serviços apresenta-se para essa proposta como uma solução atual e viável.

O foco desse trabalho está no tratamento adequado da heterogeneidade dos dados pois, após analisarmos a organização de *software* em questão, verifica-se que ela não trabalha com um volume muito grande de dados (na ordem de 5.000 linhas de dados por projeto, em média), mas sim com dados heterogêneos que são constantemente atualizados. Essa heterogeneidade nas fontes de dados influencia na sua captura, pois quanto maior a freqüência de atualização destes, mais extrações, transformações e cargas são necessárias para manter os dados do *Data Warehouse* (DW) atualizados. Uma discussão mais exaustiva desse aspecto é apresentada no Capítulo 6.

1.3 Objetivos da Pesquisa

O objetivo principal deste trabalho é propor uma abordagem para captura de métricas de PDS fazendo uso da computação orientada a serviços. Esse modelo visa a solucionar alguns problemas na obtenção de métricas organizacionais: (1) a heterogeneidade em relação

às ferramentas de gestão utilizadas e a (2) forma como as organizações de *software* controlam e armazenam suas métricas de acompanhamento.

Os objetivos específicos estão descritos como segue.

- Definir uma estrutura genérica para extração, transformação e carga de dados através de uma arquitetura orientada a serviços.
- Integrar as fontes de dados e ferramentas utilizadas durante o processo de desenvolvimento de *software* de forma não intrusiva, através da utilização de *wrappers*.
- Possibilitar que as informações contidas no DW sejam mantidas atualizadas, coesas e consistentes, para que os gestores as utilizem no controle e acompanhamento de seus PDS.

1.4 Metodologia de Pesquisa

Segundo [FAC02] "todo trabalho científico deve ser apreciado em procedimentos metodológicos". Quanto aos objetivos classifica-se esta pesquisa como exploratória, pois se tem como finalidade a descoberta de teorias e práticas que modificarão as práticas existentes e principalmente fornecerão as inovações tecnológicas necessárias. Quanto ao procedimento, inicialmente foi realizado um estudo de um ambiente real em uma organização de *software* certificada CMM Nível 2, onde foram identificadas limitações no controle e acompanhamento de seus processos devido à dificuldade de obter métricas em ambientes heterogêneos. Ainda, foi utilizado o procedimento experimental para avaliação da solução proposta.

A Figura 1 representa as principais atividades envolvidas no desenvolvimento deste trabalho. As atividades em negrito no diagrama de atividade representam as atividades macro: Estudar Fundamentação Teórica, Realizar Estudo de Caso, Desenvolver Protótipo e Avaliar Solução Proposta. Para cada uma dessas atividades macro são apresentadas as atividades que foram desenvolvidas. Inicialmente foram estudadas as principais fontes sobre o PDS, qualidade no PDS, processo de ETC e AOS (Arquiteturas Orientadas a Serviços), buscando um entendimento geral. Posteriormente foi realizado o estudo de um ambiente real, onde foi caracterizado o problema a ser tratado nesta pesquisa. Identificado o problema, foi possível definir a questão de pesquisa e objetivos deste trabalho, e direcionar a pesquisa às abordagens relacionadas ao problema. Primeiramente o estudo enfocou-se em propostas específicas relacionadas ao Processo de *Data Warehousing* para PDS. Posteriormente à definição dos

mecanismos que seriam utilizados, o passo seguinte consistiu no desenvolvimento de uma estrutura para capturar métricas de PDS utilizando uma abordagem orientada a serviços. Buscando avaliar este protótipo utilizou-se a experimentação e para tal, foram definidos os objetivos, questões e os critérios de experimentação.

1.5 Estrutura do Trabalho

Este trabalho está dividido em 10 capítulos organizados como segue. O Capítulo 2 discute o cenário atual do PDS, apresentando conceitos de processos padrão, bem como este pode ser especializado. O Capítulo 3 discorre sobre a Qualidade no PDS, focando questões sobre mensuração e maturidade dos processos. O Capítulo 4 descreve os componentes principais de uma AOS e como esta pode auxiliar no tratamento adequado da heterogeneidade encontrada em PDS, e ao final deste, uma comparação de AOS com outras tecnologias existentes. O Capítulo 5 discorre sobre o Processo de Descoberta de Conhecimento em Banco de Dados, focando principalmente no Processo de ETC e as implicações que ambientes heterogêneos acarretam ao processo de ETC, bem como possíveis soluções para este problema. O Capítulo 6 relata o estudo realizado em organização de *software* certificada CMM Nível 2, que busca a certificação CMM Nível 3. Já o Capítulo 7 descreve a solução proposta, uma abordagem orientada a serviços para captura de métricas de PDS. O Capítulo 8 discorre sobre a experimentação da solução utilizando três exemplos que possuem características heterogêneas entre si. O Capítulo 9 apresenta os trabalhos relacionados comparando-os com nossa abordagem. O Capítulo 10 discorre sobre as conclusões, limitações e trabalhos futuros. Posteriormente, encontram-se as referências bibliográficas pesquisadas e os demais anexos.

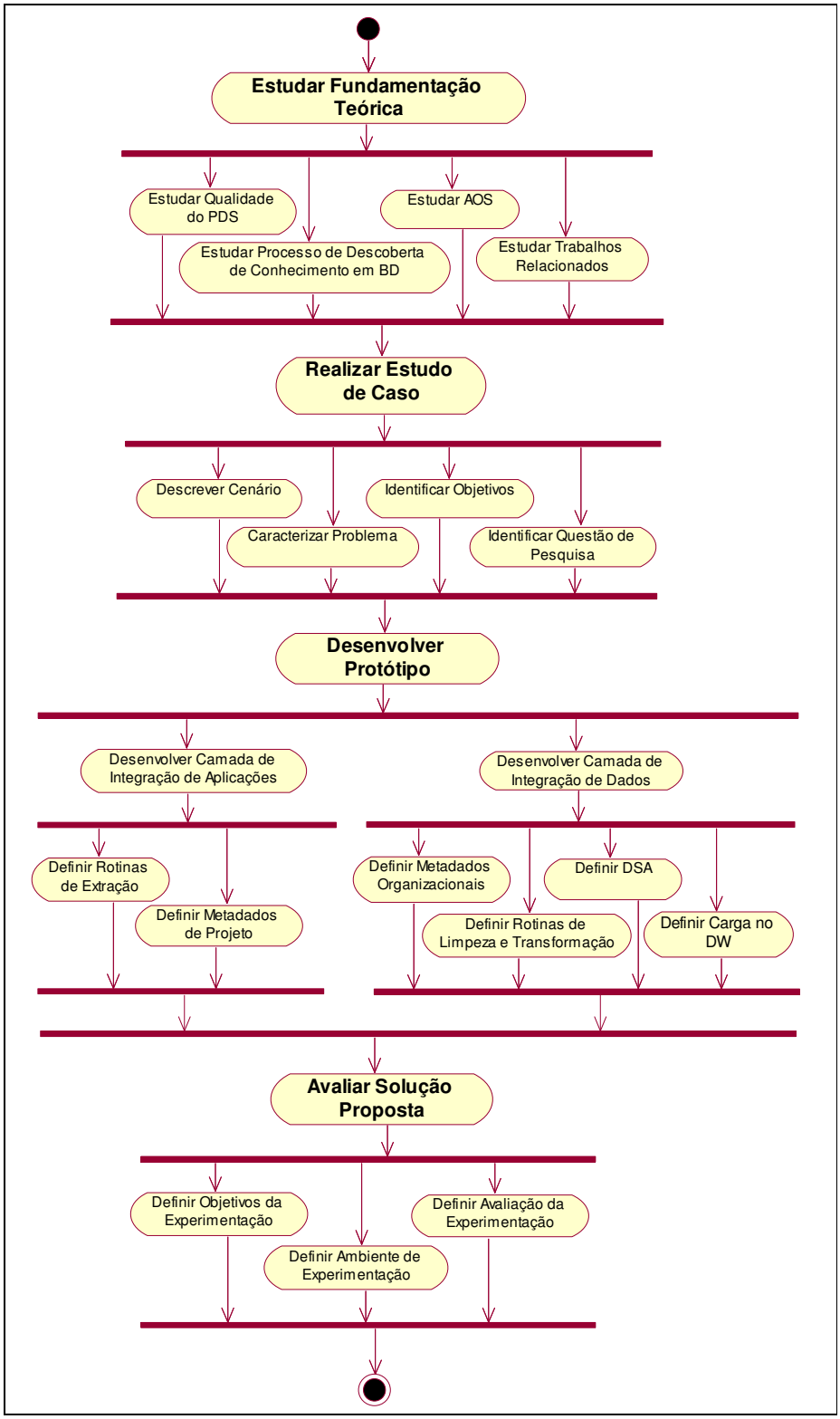

Figura 1 - Atividades conduzidas durante a pesquisa

2 PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

Este capítulo tem por objetivo relatar o cenário atual de desenvolvimento de software, e para tal apresenta o conceito de processo de desenvolvimento de software padrão, bem como seus elementos principais e relacionamento entre estes e ainda, como um processo padrão pode ser especializado, proporcionando sua usabilidade.

A IEEE [IEE98] define um processo como: "uma seqüência de passos realizados com um dado propósito". [SOM01] define um PDS como um conjunto completo de atividades necessárias para transformar os requisitos de usuários em produtos de *software*. Para [PRE01], este processo pode ser definido como uma estrutura para as tarefas que são necessárias à construção de um *software* de alta qualidade. Ainda, segundo [FUG00], o PDS é definido como um conjunto coerente de políticas, estruturas organizacionais, tecnologias, procedimentos e artefatos que são necessários para compreender, desenvolver e manter um produto de *software*.

Atualmente, as variáveis envolvidas no PDS têm um nível crescente de complexidade: reduzir custos e tempo, aumentar a produtividade e, ainda assim, garantir a qualidade do produto final. Sendo assim, para garantir a qualidade de um *software* deve-se estabelecer uma cultura de não-tolerância a erros, ou seja, buscar estruturar processos de forma que eles possuam mecanismos para inibir e impedir falhas, possibilitando que os diversos artefatos gerados durante o ciclo de desenvolvimento tenham procedimentos que avaliem sua qualidade e que possibilitem a identificação prematura de defeitos nesses artefatos.

Segundo [PAU93] definir um processo de *software* padrão a ser utilizado pela organização é fundamental na obtenção de altos níveis de maturidade. Atualmente, além da definição de um PDS, a implantação de um processo padrão de desenvolvimento de *software* nas organizações tem sido fortemente demandada pelas organizações de *software* e, neste sentido, vários processos têm sido sugeridos por organizações e companhias internacionais [YOO01].

2.1 Processo de Software Padrão da Organização

Um fundamental conceito na definição de processo, no CMM, é o Processo de *Software* Padrão da Organização (OSSP – *Organization's Standard Software Process*). Um OSSP é a definição operacional de procedimentos básicos que guiam o estabelecimento de um processo de *software* comum entre os projetos de *software* da organização. Ele descreve os elementos principais do processo de *software* que cada projeto de *software* deve incorporar, bem como os relacionamentos entre esses elementos de *software* da organização, e é essencial para a estabilidade e melhoramento do PDS [PAU93].

No nível gerencial, o OSSP necessita ser descrito, gerenciado, controlado e melhorado de uma maneira formal. Isso se torna possível através da utilização de práticas chave de definição de processos organizacionais (ver Figura 2). No nível de projeto, a ênfase está na usabilidade do OSSP e, dessa forma, é necessário definir um Processo de *Software* Definido do Projeto (PDSP – *Project's Defined Software Process*). O PDSP é uma especialização do OSSP ajustado às características do projeto. Este deve ser entendido e bem caracterizado e, para tal, deve ser descrito em termos de padrões, procedimentos, ferramentas e métodos de *software* [PAU93].

A parte superior da Figura 2, extraída de [PAU93], apresenta como as organizações estabelecem e mantêm um conjunto de processos de *software*. Esse conjunto de processos inclui:

1. **A base de dados do processo de *software* da organização, ou seja, um repositório que armazene as métricas dos projetos da organização (*Organization's Software Process Database*)**
    - A base de dados de processos da organização[2] é uma base de dados estabelecida para coletar e disponibilizar dados sobre o processo de *software*, bem como dos produtos de trabalho gerados. Exemplos de dados de processos e produtos de trabalho incluem esforço, custo e tamanho de *software* estimado, produtividade, eficiência de *Peer Review* e o número de defeitos encontrados e a severidade destes.
2. **A biblioteca com os documentos relacionados ao processo de *software* (*Library of Software Process-Related Documentation*)**

[2] Neste trabalho, denominada Base Organizacional (BO)

- Uma biblioteca de documentos relacionados ao processo de *software* é estabelecida para (1) armazenar documentos do processo que são potencialmente úteis para projetos correntes e futuros e (2) torná-los disponíveis para que sejam compartilhados entre os projetos.

3. **A descrição dos ciclos de vida aprovados para utilização (*Description of the Software Life Cycles*)**
    - Um ciclo de vida de *software* é o período de tempo que se inicia quando um produto de *software* é concebido e termina quando um produto de *software* não está distante de ser disponibilizado para uso [PAU93]. O ciclo de vida de *software* tipicamente inclui conceitos de estágios: estágio de requisitos, *design*, implementação, teste, instalação, operacional e manutenção. A Seção 2.1.1 descreve dois entre os ciclos de vida mais utilizados.
4. **As diretrizes e critérios para especialização do OSSP (*Guidelines and Criteria for Tailoring the Organization's Standard Software Process*)**
    - As diretrizes são estabelecidas para auxiliar os projetos a (1) selecionar o ciclo de vida de *software* adequado entre aqueles aprovados para o uso e (2) especializar o OSSP e o ciclo de vida de *software* selecionado para ajustarem-se às características específicas do projeto. Esses *guidelines* e critérios ajudam a garantir que exista uma base comum entre todos os projetos de *software* para planejamento, implementação, mensuração, análise e melhoramento do PDSP.
5. **A descrição do OSSP (*Description of Organization's Standard Software Process*), incluindo a arquitetura e a descrição dos elementos do processo de *software*:**
    - A arquitetura do processo de *software* é uma descrição de alto nível da OSSP. Ela descreve a ordem, interfaces, interdependências e outros relacionamentos entre os elementos do processo de *software* do OSSP. Cada elemento do processo abriga um bem definido e limitado conjunto de tarefas (e.g. elementos relacionados a estimativas de *software*, projeto de *software*, codificação e *Peer Review*).

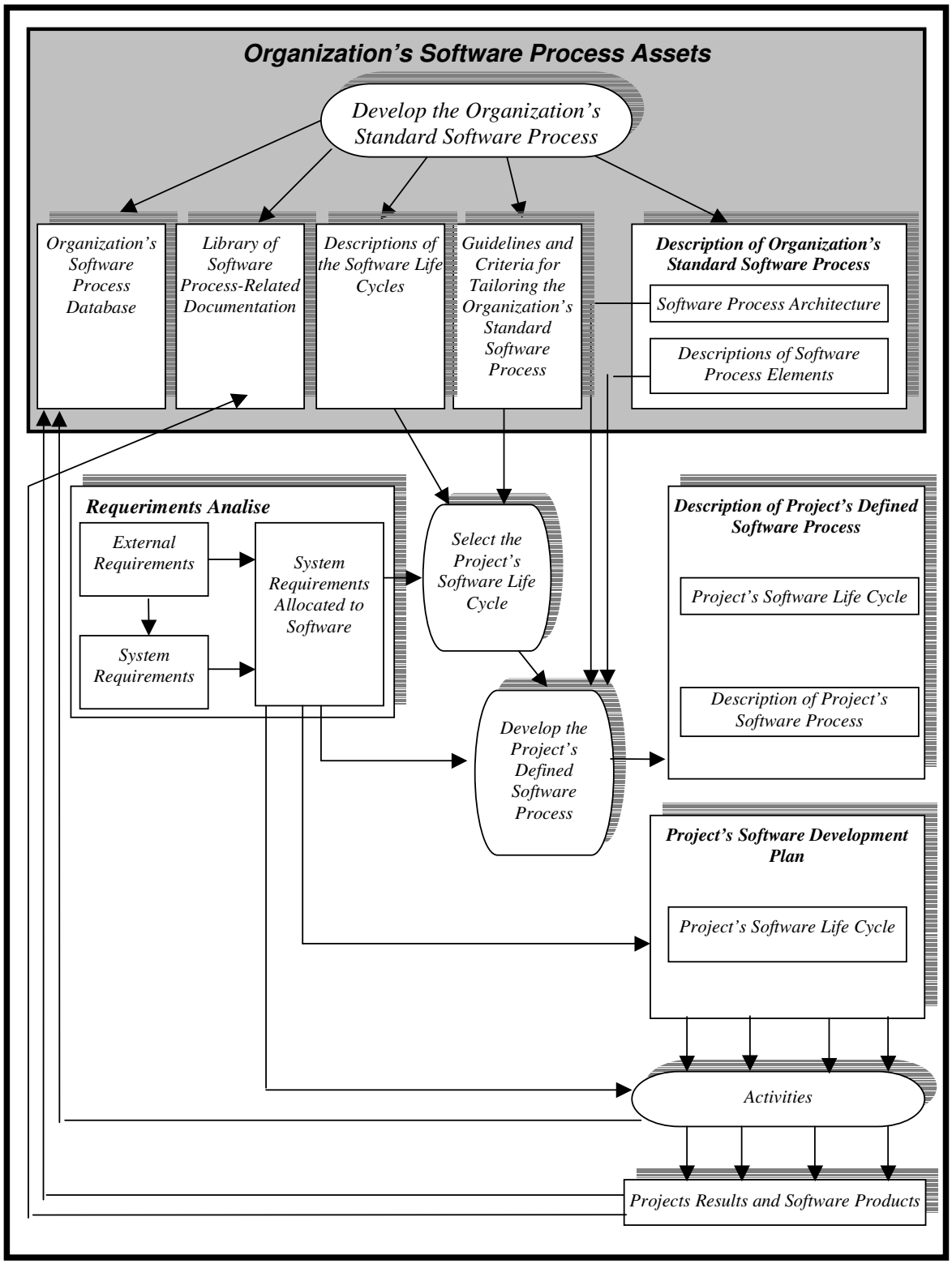

Figura 2 - ***Framework*** **do Processo de** ***Software*** **usado no CMM, extraída de [PAU93]**

A parte inferior da Figura 2, ilustra a especialização do OSSP para adequar-se às características específicas de um projeto. Esta especialização inclui: (1) analisar os requisitos do projeto, (2) selecionar um ciclo de vida dentre os aprovados pela organização e (3) desenvolver seu próprio PDSP. O PDSP provê a base para o planejamento, execução e aprimoramento das atividades, tanto de gestão quanto operacionais. O Plano de Desenvolvimento de Projeto de *Software* (*Project's Software Development Plan*) define como um projeto deve ser executado. Por exemplo, define quais recursos irão executar quais atividades, baseado em um cronograma. As tarefas do projeto estão divididas em estágios. Esses estágios representam um significante e mensurável conjunto de atividades que são executadas pelo projeto [PAU93]. O resultado dessas atividades e tarefas consiste em produtos de trabalho (*Project's Results and Software Work Products*)

2.1.1 Ciclos de vida

A variedade de ciclos de vida que podem ser utilizados pelas organizações de desenvolvimento de *software* deve-se ao fato dessas produzirem *software* para uma variedade de clientes e usuários com realidades distintas. Entre os ciclos de vida mais utilizados podemos citar:

- **Ciclo de vida em Cascata:** Este ciclo de vida utiliza uma abordagem seqüencial para o desenvolvimento, iniciando pela Definição dos Requisitos, Projeto, Implementação e Teste Unitário, Integração e Teste de Sistema, e Operação e Manutenção (ver Figura 3). Durante a fase final, Operação e Manutenção, o *software* é colocado em uso, e erros e omissões nos requisitos originais do *software* podem ser descobertos, necessitando assim a correção dos erros e a implementação de novas funcionalidades. Essas mudanças podem ocasionar a repetição de alguns ou todos os estágios anteriores. Este ciclo de vida é recomendado para projetos grandes e complexos nos quais os requisitos são bem definidos [SOM01].

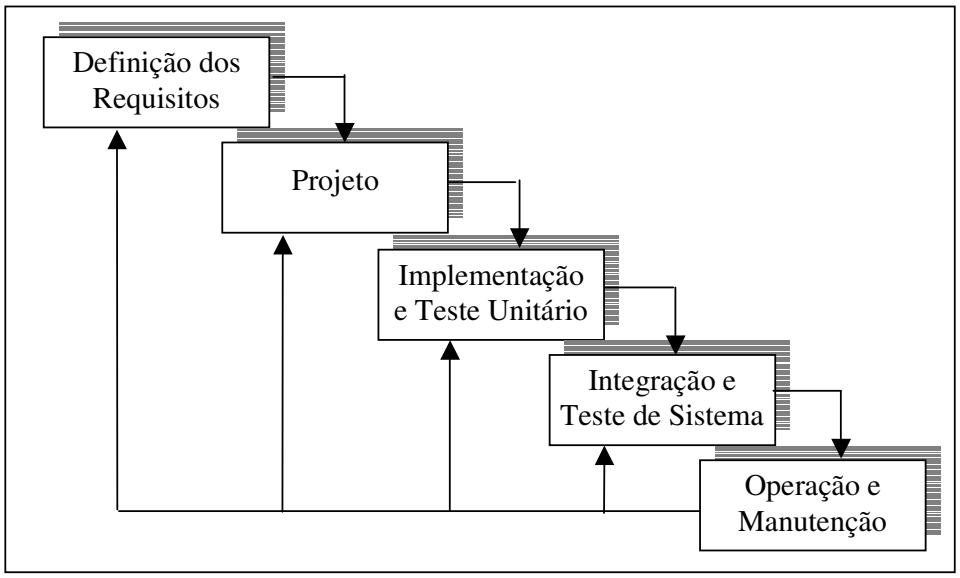

Figura 3 - Ciclo de Vida em Cascata, extraída de [SOM01]

- **Ciclo de vida *Evolutionary Development* ou Iterativo:** Iterações são pequenas etapas do ciclo de vida definidas por seqüências distintas de atividades com planejamento detalhado e critérios de validação, resultando sempre em algum produto, interno ou externo. Como o planejamento das iterações nunca é todo feito no início do projeto os riscos de mau planejamento reduzem-se, pois os elementos são integrados progressivamente. Segundo [SOM01], ver Figura 4, o ciclo de vida iterativo possui atividades de especificação, desenvolvimento e validação separadamente. Essas atividades são executadas de forma concorrente e entre elas existe um *feedback*.

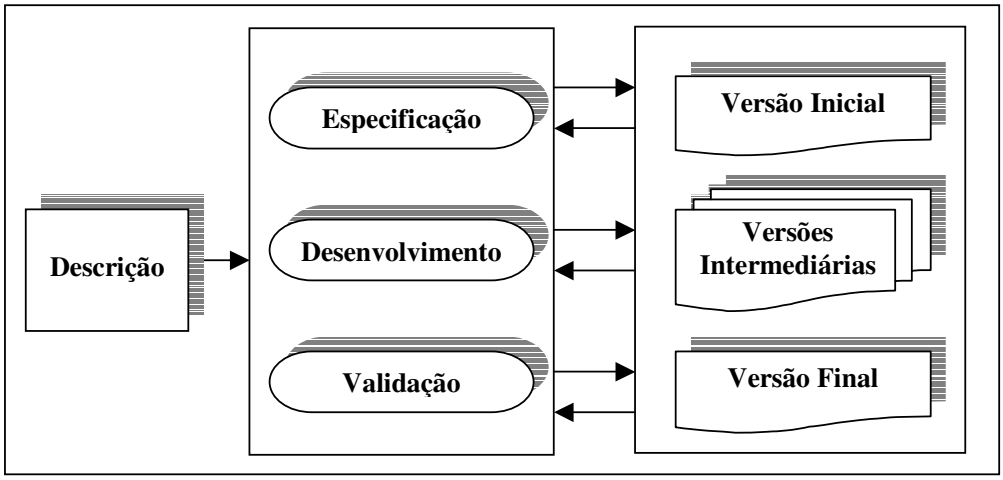

Figura 4 - Ciclo de Vida Evolutionary Development ou Iterativo, extraída de [SOM01]

2.2 Considerações Adicionais

Esse capítulo apresentou conceitos relacionados ao PDS, bem como a fundamental importância de definir um processo padrão, o qual deve ser especializado conforme necessidade dos projetos.

Ainda, a Seção 2.1 ressaltou que mesmo possuindo PDS padrão, existe heterogeneidade, ou seja, que um processo de *software* definido não é garantia de homogeneização. Sendo assim, os projetos partem de uma estrutura homogênea organizacional, OSSP, para uma estrutura heterogênea de projetos, PDSP, resultando em, diferentes tipos de projetos, diferentes ciclos de vida, diferentes modelos de gerenciamento, bem como diferentes ferramentas para o acompanhamento e gestão desses projetos.

O Capítulo 6, apresenta o estudo de caso realizado em uma organização de *software* que busca a certificação CMM3 e utiliza o conjunto de processos apresentado na Figura 2.

3 QUALIDADE NO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

Este capítulo tem por objetivo discorrer sobre a qualidade no PDS, e para tal descreve mensuração de PDS e maturidade organizacional através do modelo de capacidade de software CMM. Ainda, apresenta o conceito de métricas, bem como suas classificações. Essas métricas são utilizadas para mensurar a qualidade dos PDS auxiliando-os a organizar, monitorar e prevenir falhas.

Segundo [SOM01], as atividades do processo de QA (*Quality Assurance*) envolvem definir ou selecionar padrões que devam ser aplicados ao PDS e/ou ao produto de *software*. Esses padrões podem ser integrados a procedimentos ou processos que devam ser aplicados durante o desenvolvimento. Os processos podem ser apoiados pelo uso de ferramentas que integrem o conhecimento dos padrões de qualidade. Existem dois tipos de padrões que podem ser estabelecidos como parte do processo de garantia da qualidade:

a. **Padrões de produto:** são os padrões que se aplicam ao produto de *software* em desenvolvimento. Eles podem incluir padrões de documentos, como por exemplo, estrutura dos documentos dos requisitos que devem ser produzidos e manuais dos produtos produzidos.

b. **Padrões de processo:** são os padrões que definem os processos a serem seguidos durante o desenvolvimento de *software*. Eles podem incluir definições de especificação, processos de projeto e validação, e uma descrição dos documentos que devem ser gerados no decorrer do processo.

O controle de qualidade envolve supervisionar o PDS, a fim de assegurar que os procedimentos e os padrões de garantia da qualidade sejam seguidos.

3.1 Maturidade Organizacional

Com o passar dos anos, muitas organizações começaram a perceber que seu problema principal residia na inabilidade para gerenciar o processo de desenvolvimento de *software*.

À medida que uma organização vai se tornando madura e capaz, o processo de *software* vai ficando mais definido, possibilitando que o mesmo seja implementado de modo mais consistente em toda a organização. A capacidade do processo de *software* descreve a gama de resultados esperados que podem ser alcançados com a aplicação do processo de *software*, bem como, formas de se prever os resultados mais prováveis a serem esperados no próximo projeto a ser empreendido pela organização. Já a maturidade do processo de *software* é a extensão para a qual um processo específico é explicitamente definido, gerenciado, medido, controlado e efetivado. Em uma organização madura, o processo de *software* é compreendido – o que geralmente é feito por meio de documentação e treinamento – e está sendo continuamente monitorado e melhorado pelos seus usuários. A capacidade de um processo de *software* maduro é conhecida. A maturidade de um processo de *software* implica que a produtividade e a qualidade resultantes do processo possam ser continuamente melhoradas através de ganhos consistentes na disciplina alcançada com a sua utilização. Essa melhoria se deve à estrutura estabelecida pelo CMM, onde o projeto percorre um caminho evolutivo que incrementa, em níveis, a maturidade do processo de *software* [SOM01].

A estrutura da maturidade ordena esses níveis pré-estabelecidos (*Initial, Repeatable, Defined, Managed* e *Optimizing*), onde os resultados positivos alcançados em cada nível são utilizados como embasamento para o próximo, objetivando melhorias no processo em sua totalidade. Dessa forma, uma estratégia de melhoria projetada a partir de uma estrutura de maturidade de processo de *software* orienta quanto ao caminho a ser seguido para a contínua melhoria do processo de *software*.

O resultado desse trabalho foi o modelo CMM, que tem como foco o processo de *software* na proposta de melhoria contínua, trazendo disciplina e controle no desenvolvimento e manutenção do *software*. Essas são as chaves para aperfeiçoar a qualidade do desenvolvimento de produtos de *software*.

3.1.1 Capability Maturity Model

O CMM definido pelo SEI (*Software Engineering Institute*) descreve uma estrutura de trabalho que possui todos os elementos necessários para tornar um PDS mais eficiente e controlado. O CMM baseia-se em um modelo evolutivo de maturidade, no qual as organizações partem de uma total falta de controle e gerenciamento dos processos (maturidade organizacional) para gradativamente adquirir novas competências, incrementando seu nível de eficiência e maturidade em relação aos diversos processos críticos envolvidos em um desenvolvimento de *software* [ROC01]. Essa estrutura em níveis do CMM é baseada em princípios de qualidade do produto e esses princípios foram adaptados pelo SEI dentro da estrutura de maturidade que estabelece a gestão de projeto e os fundamentos de engenharia para o controle quantitativo do processo de *software*, que é a base para a contínua melhoria do processo.

A Figura 5[3], ilustra o modelo CMM que possui 5 níveis de maturidade: (1) *Initial*, (2) *Repeatable*, (3) *Defined*, (4) *Managed* e (5) *Optimizing*. Cada nível de maturidade (*Maturity Levels*) indica a capacidade de um processo e é composto de várias áreas-chave de processo (*Key Process Areas*), por exemplo, uma área chave de processo do nível 2 é a Gerência de Configuração de *Software* (*Software Configuration Management*). Cada área-chave de processo é organizada em cinco seções denominadas características comuns (*Common Features*), são elas: (1) comprometimento para executar, (2) habilidade para executar, (3) mensuração, (4) análise, e (5) verificação da implementação. Essas características comuns especificam as práticas-chave (*Key Practices*) que, quando tratadas coletivamente, atingem as metas previstas para a área-chave de processo.

Cada nível da maturidade fornece uma fundação para a melhoria contínua do processo. Cada área chave de processo compreende um conjunto de objetivos que, quando satisfeitos, estabilizam um componente importante do processo de *software*. Obtendo um nível de maturidade, institucionaliza-se um componente diferente no processo de *software*, que tem por finalidade um aumento total na potencialidade do processo da organização. Por exemplo, no nível 2 a capacidade de processo de uma organização foi elevada de uma situação "*ad-hoc*" para uma situação repetível, por meio do estabelecimento de sólidos controles de gestão de projeto.

[3] Essa figura é uma composição de duas figuras extraídas de [PAU93].

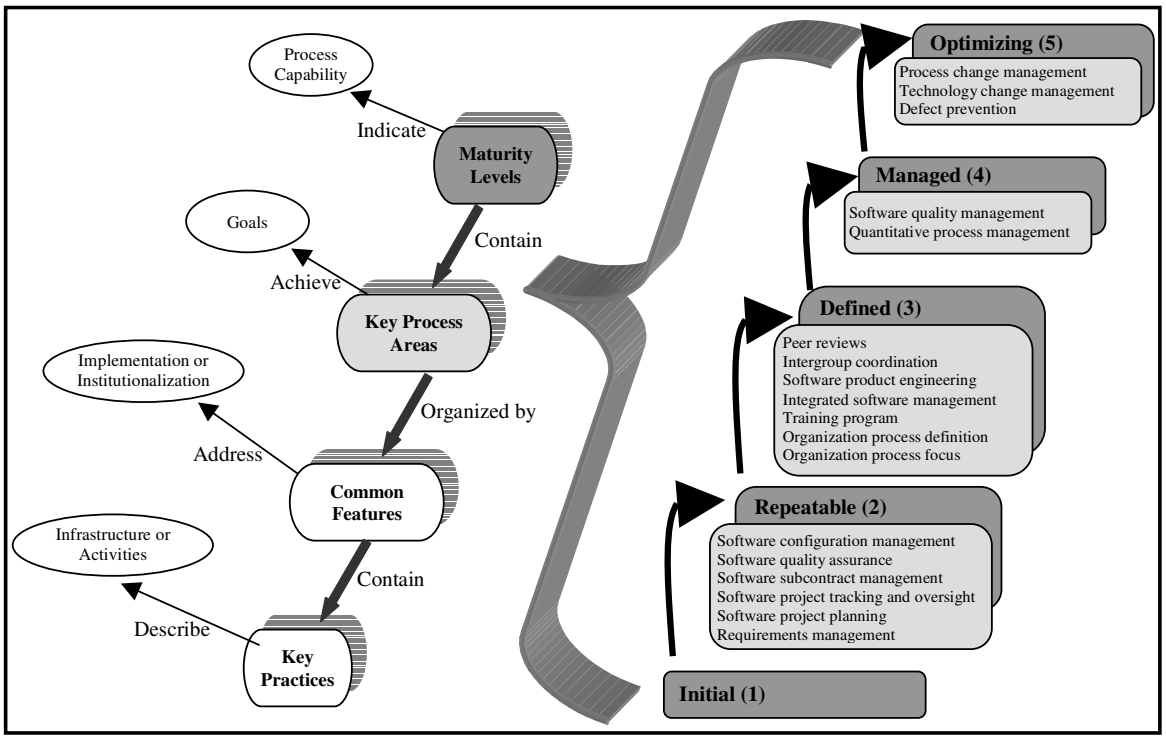

Figura 5 – Modelo CMM

Com exceção do nível 1, cada nível de maturidade é decomposto em várias áreas-chave de processo, que indicam as áreas nas quais uma organização deveria fixar seus esforços para a melhoria de seu processo de *software*. As áreas-chave de processo identificam os assuntos que devem ser tratados para se obter um determinado nível de maturidade.

Nenhuma empresa consegue sair do nível 1 e chegar ao nível 3 sem antes passar pelo nível 2. Cada nível é um pré-requisito para o outro e cada organização consegue enxergar somente o que a sua maturidade permite. Apesar disso, as organizações podem usar de maneira vantajosa processos descritos em níveis de maturidade superiores aos que se encontram.

3.2 Mensuração de PDS

Segundo [HAR91] *apud* [LIS04], a mensuração é muito importante. Se não é possível medir, não é possível controlar. Se não é possível controlar, não é possível gerenciar. E, se não é possível gerenciar, consequentemente, não é possível melhorar.

Segundo [PRE96], um *software* deve ser medido por diversas razões: (1) indicar a qualidade do produto; (2) avaliar a produtividade das pessoas que produzem o produto; (3) avaliar os benefícios (em termos de produtividade e qualidade) derivados de novos métodos e

ferramentas de *software*; (4) formar uma linha básica para estimativas; (5) ajudar a justificar os pedidos de novas ferramentas ou treinamento adicional. Em contradição a [PRES96], [SOM01] argumenta que o uso da medição e de métricas sistemáticas de *software* ainda é relativamente incomum. Ele afirma que existe uma relutância em introduzir a medição, porque os benefícios não são bem definidos. E afirma que uma razão para isso é que, em muitas empresas, os processos de *software* utilizados ainda são organizados de maneira precária e não estão suficientemente aperfeiçoados para fazer uso de medições. Outra razão é que não há padrões para as métricas, e daí decorre o suporte limitado para a coleta e análise de dados. O mesmo autor afirma, ainda, que a maioria das empresas não estará preparada para introduzir medições até que esses padrões e essas ferramentas estejam disponíveis.

[KAN04] afirma que as métricas de *software* podem ser particionadas em três domínios diferentes: métricas de produto, métricas de processo e métricas de projeto. Métricas de produto descrevem as características dos produtos tais como tamanho e complexidade. Métricas de processo podem ser usadas para melhorar a manutenção e desenvolvimento do *software*, por exemplo, a métricas eficiência de remoção de defeitos. Métricas de projeto descrevem as características e a execução do projeto. Exemplos incluem custo, cronograma e produtividade. Ainda, segundo [KAN04], métricas de qualidade de *software* são um subconjunto das métricas de *software* que focam aspectos de qualidade dos processos, produtos e projetos.

3.2.1 Definição de métricas

A IEEE [IEE98] define um atributo como "uma propriedade física ou abstrata mensurável de uma entidade". Um fator de qualidade é um tipo de atributo, ou seja, um atributo orientado a gestão de *software* que contribui para sua qualidade. Uma métrica é uma função mensurável e uma métrica de qualidade de *software* é "uma função cujas entradas sejam dados do *software* e cuja saída seja um único valor numérico que possa ser interpretado como o grau que o *software* possui, dado um atributo que afete sua qualidade". Neste trabalho, métrica e medida são consideradas sinônimos.

3.2.2 Classificação de métricas

A IEEE [IEE98] classifica as métricas de *software* em medidas diretas e medidas indiretas, e define que uma métrica direta é uma métrica que não depende da medida de nenhum outro atributo, nem necessita de validação. Já uma métrica indireta depende de outros

atributos e necessita ser validada. De acordo com a ISO [ISO93], as métricas podem ser divididas em métricas base e derivadas. Métricas base fornecem uma indicação quantitativa de um atributo. Métricas derivadas relacionam-se a métricas base com a finalidade de proporcionar uma informação quantitativa sobre um processo ou um produto. Elas são importantes para as atividades de gerenciamento de *software* e para o controle de desempenho dos processos, tanto no âmbito da organização como em cada projeto.

Em oposição à [IEE98] que diz que uma métrica direta é pressupostamente válida, [KAN04] afirma que todas as métricas devem ser validadas. Ainda, segundo [KAN04] a diferença entre métrica direta e indireta, ou métrica derivada, está em dizer que a métrica direta é uma função métrica cujo domínio é somente de um atributo e o domínio de uma métrica indireta é formado por uma *n-tupla*. Exemplos de métricas derivadas em engenharia de *software* são:

- Produtividade: número de linhas de código/ tempo.
- Densidade de defeitos: número de defeitos/ tamanho.
- Estabilidade dos requisitos: número de requisitos inicial / número total de requisitos.

Número de defeitos, tamanho e número de linhas de código são exemplos de métricas diretas.

3.2.3 *Framework* de métricas de qualidade de *software*

O uso de métricas de *software* reduz a subjetividade na estimativa e no controle da qualidade de *software* por prover uma base quantitativa para a tomada de decisão. Entretanto, a utilização de métricas de *software* não elimina a necessidade do julgamento humano em estimativas de *software*. O uso de métricas de *software,* nas organizações e em seus projetos, através da mensuração dos projetos, busca tornar a qualidade do *software* mais visível.

A Figura 6 ilustra o *framework* de métricas de qualidade de *software* proposto pela IEEE [IEE98]. Ele permite adicionar, excluir e modificar fatores e subfatores de qualidade, bem como métricas. Cada nível pode ser expandido para diversos subníveis. O *framework* pode assim ser aplicado para todos os sistemas e pode ser adaptado como apropriado sem mudar o seu conceito básico. O primeiro nível do *framework* de métricas de qualidade de *software* começa com o estabelecimento dos requisitos de qualidade, que descrevem a

qualidade do sistema (na Figura 6 representado por *Software quality system X*). Associada a cada fator de qualidade está uma métrica direta que serve como uma representação quantitativa desse fator. No segundo nível, estão os subfatores de qualidade que representam atributos orientados ao *software* que indicam a qualidade. Esses subfatores podem ser obtidos pela decomposição de cada fator de qualidade em atributos de *software* mensuráveis e podem corresponder a mais de um fator de qualidade, pois são atributos independentes do *software.* No terceiro nível, estão as métricas decompostas dos subfatores utilizadas para medir produtos e processos durante todo o ciclo de vida de desenvolvimento do *software*.

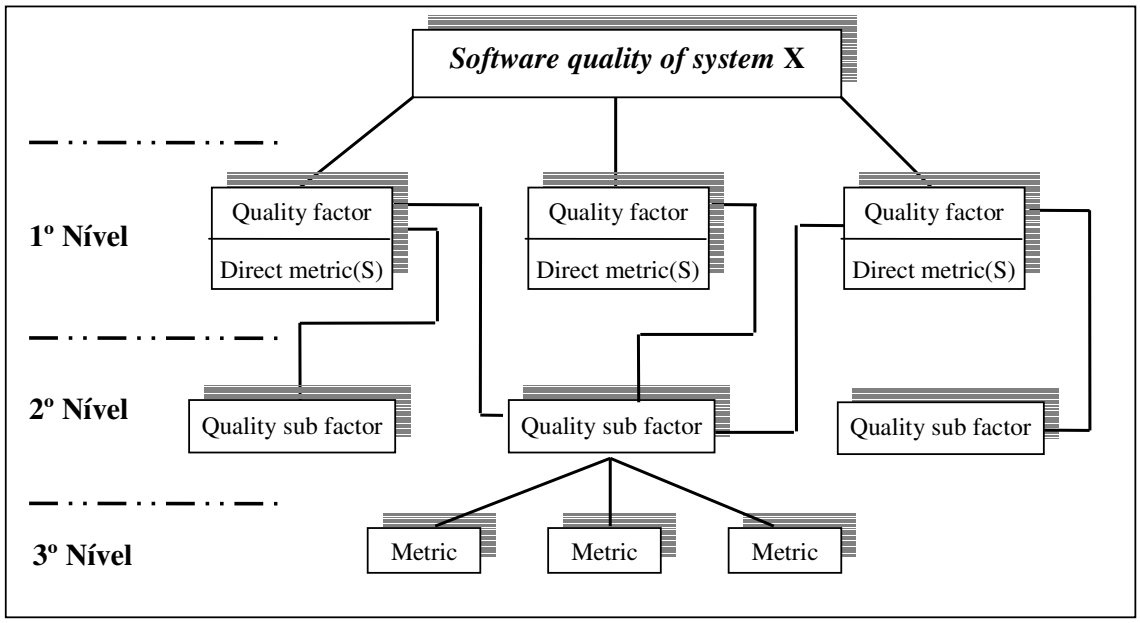

Figura 6 - *Framework* de métricas de qualidade de *software, extraída de [IEE98]*

Realizando uma análise *top-down* percebe-se que o *framework* facilita:

- O estabelecimento dos requisitos de qualidade em termos de fatores de qualidade e pela possibilidade de gerenciar o ciclo de vida do sistema.
- A identificação das métricas que estão relacionadas aos fatores e subfatores de qualidade.

Através de uma análise *bottom-up* percebe-se que o *framework* habilita a obtenção de uma:

- Avaliação dos produtos e processos de *software* através das métricas.

- Análise dos valores das métricas para estimar e avaliar os fatores de qualidade.

3.2.4 Critérios de Validação de Métricas de Qualidade de Software

Para [IEE98], o propósito da validação de métricas é identificar métricas de processos e de produtos que podem prever valores de fatores de qualidade. Esses fatores são representações quantitativas dos requisitos de qualidade e as métricas devem indicar se os requisitos de qualidade foram alcançados ou serão alcançados no futuro. Ainda, [IEE98] afirma que validação não significa uma validação universal de todas as métricas para todas as aplicações. Preferencialmente, refere-se à validação do relacionamento entre um conjunto de métricas e um fator de qualidade para uma dada aplicação.

A IEEE [IEE98] apresenta seis critérios de validação, os quais são expressos em termos de relacionamentos quantitativos entre o atributo sendo mensurado (o fator de qualidade) e a métrica. São eles:

1. **Correlação:** a métrica deve ser linearmente relacionada ao fator de qualidade como medido pela correlação estatística entre a métrica e o fator de qualidade correspondente.
2. **Consistência:** seja F, a variável fator de qualidade e Y a saída da função métrica, $M: F \rightarrow Y$. M deve ser uma função monotônica, isto é, se $f_1 > f_2 > f_3$, então devemos obter $y_1 > y_2 > y_3$.
3. **Rastreamento:** para funções métricas, $M: F \rightarrow Y$. Como F muda de f_1 para f_2 em tempo real, $M(f)$ deve mudar imediatamente de y_1 para y_2.
4. **Previsibilidade:** para funções métricas, $M: F \rightarrow Y$. Se o valor de Y é conhecido em algum ponto, pode-se predizer o valor de F.
5. **Poder discriminativo**: uma métrica será discriminativa quando estiver entre componentes de *software* de alta qualidade e componentes de *software* de baixa qualidade. O conjunto de valores de métricas associados com o anterior deve ser significativamente maior (ou menor) do que aqueles associados com o último. Esse critério avalia se uma métrica é capaz de separar um conjunto de componentes de *software* de alta qualidade de um conjunto de componentes de baixa qualidade.

6. **Confiança:** uma métrica deve demonstrar a propriedade de correlação, rastreamento, consistência, previsibilidade e poder discriminativo em pelo menos *P%* das aplicações das métricas, onde *P* é o percentual de sucesso que garante que uma métrica passou por testes de validação.

3.3 Comentários Adicionais

Esse capítulo apresentou conceitos relacionados à qualidade de *software* e como os processos de uma organização devem estar estruturados para garantir a qualidade durante o desenvolvimento e do produto final. Ainda na Seção 3.1, discorreu-se sobre que, à medida que uma organização vai se tornando madura e capaz através da adoção de modelos de maturidade, o processo de *software* vai se tornando mais definido, possibilitando que o mesmo seja implementado de modo mais consistente em toda a organização. A construção de uma BO é um dos requisitos para a obtenção da certificação CMM3. Essa base histórica de dados deve armazenar métricas da execução dos processos, possibilitando que essas métricas sejam utilizadas no planejamento de novos projetos, bem como na análise e melhoria dos projetos e processos já existentes.

Além disso, esse capítulo serviu de fundamentação teórica para o Capítulo 6, onde se discorre sobre o estudo de caso em um ambiente organizacional, apresentando para tal o nível de maturidade, programa de métricas, bem como a estrutura da BO dessa organização.

4 ARQUITETURAS ORIENTADAS A SERVIÇOS

Este capítulo apresenta os componentes principais de arquiteturas orientadas a serviços, focando-se principalmente em relatar como esses componentes interligados podem ser utilizados no tratamento adequado da heterogeneidade, bem como a comparação dessa arquitetura com outras tecnologias.

Segundo a *World Wide Web Consortium* (W3C) [W3C05], um "*Web Service* (WS) é um sistema de *software* projetado para suportar interações de máquina-à-máquina sobre uma rede. Este sistema tem uma interface descrita em um formato processável (especificamente WSDL). Outros sistemas interagem com o WS de uma maneira prescrita por sua descrição através de mensagens SOAP, tipicamente transportadas utilizando o HTTP em um formato XML, em conjunto com outros padrões *web* relacionados". O uso de tecnologias baseadas em XML para a construção de WS permite a utilização de serviços sem que haja a necessidade de se saber qual a plataforma ou linguagem de programação foi utilizada na sua construção. Esta definição detalha como os WS devem trabalhar, enfatizando que os WS devem ser capazes de serem definidos, descritos e descobertos. Esta definição, ainda, evidencia o significado de acessível e torna mais concreta a noção de orientada a internet e interfaces baseada em padrões.

Segundo [UDD03] *apud* [ALO04], a mais precisa definição de WS é provida pelo Consórcio UDDI que caracteriza o como "*self-contained, modular business application that have open, Internet-oriented, standards-based interfaces*". Essa definição é mais detalhada que a apresentada pela W3C, pois enfatiza a necessidade de ser complacente com os padrões de internet. Além disso, requer que o serviço seja aberto, o que essencialmente significa que ela tenha uma interface publicada que pode ser invocada através da internet.

Uma outra definição apresentada por [ALO04], é que um WS é definido como uma forma padronizada de integrar aplicações baseadas na *web* utilizando XML, SOAP, WSDL e

UDDI. O XML é usado para *tag* de dados, o SOAP é usado para transferir os dados, o WSDL é usado para descrever os serviços disponíveis e o UDDI é usado para listar quais serviços estão disponíveis.

Uma arquitetura orientada a serviços é essencialmente uma coleção de serviços. Esses serviços comunicam-se uns com os outros. A comunicação pode envolver uma simples troca de dados ou dois ou mais serviços que estejam coordenando alguma atividade. Essa arquitetura fornece o modelo teórico para todos os WS e contém três entidades e três operações de serviços (ver Figura 7). Quando o *Service Provider* deseja oferecer seus serviços, ele publica seus serviços colocando-os nas entradas apropriadas do *Service Directory* que contém todas as informações sobre todos os serviços que estão disponíveis. Um *Service Requestor* usa o *Service Directory* para encontrar um serviço apropriado, isto é, um serviço que combina com suas exigências. Em outras palavras, o *Service Requestor* é uma aplicação que deseja receber um serviço de outra aplicação. Ela não sabe onde está essa outra aplicação nem como localizá-la. Então, ela recorre ao *Service Directory* e requisita uma operação *Find ( )*. O *Service Directory* é uma aplicação conhecida que retorna informações sobre WS em resposta a um critério de pesquisa que tenha sido submetido por um *Service Requestor* na operação *Find( ).* Essas operações retornam as informações de contato para WS com base em suas classificações que correspondem ao critério de pesquisa. Ele também inclui informações sobre como descobrir detalhes de conexão. O *Service Provider* publica *(Publish( ))* esses detalhes de conexão, no *Service Directory*. Eles fazem isso pela mesma razão que uma empresa compraria um espaço nas páginas amarelas do catálogo telefônico local. O *Service Requestor* usa os detalhes de conexão para se vincular *(Bind( ))* ao *Service Provider*. Uma vez vinculados, eles podem enviar mensagens e receber respostas.

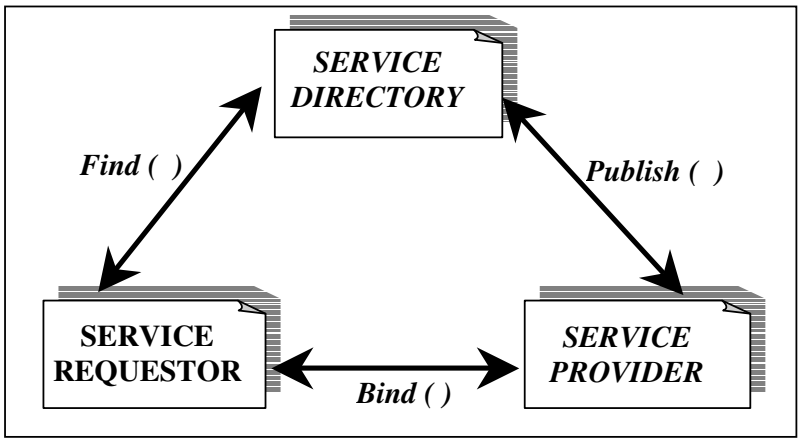

Figura 7 - Atores de uma arquitetura orientada a serviços, extraída de [LEY02]

Os componentes principais utilizados para a comunicação entre aplicações de WS, que inter-relacionados empacotam a requisição e a resposta entre um servidor e um cliente objetivando a integração de interações entre aplicações, são apresentados como segue.

4.1 Componentes Principais

4.1.1 Simple Object Access Protocol (SOAP)

SOAP, conforme [MIT04] é um protocolo projetado para invocar aplicações remotas através de RPC ou trocas de mensagens, em um ambiente independente de plataforma e linguagem de programação. O protocolo SOAP define o formato no qual as mensagens transportadas na rede devem ter para encaminhar requisições a serviços *web* e também define o formato das mensagens de resposta às requisições. SOAP é, portanto, um padrão normalmente aceito para utilizar-se com WS. Desta forma, pretende-se garantir a interoperabilidade e intercomunicação entre diferentes sistemas, através da utilização de uma linguagem XML. O mecanismo de transporte de mensagens é feito utilizando o HTTP, o que o torna um protocolo leve.

Esse mecanismo para interconexão de aplicações é formado por três diferentes partes:

1. **Especificação do Envelope SOAP:** na especificação são definidas as regras específicas para encapsular os dados que devem ser transferidos, incluindo dados da aplicação, tais como, métodos a invocar, parâmetros ou valores de

retorno. Pode também incluir informações sobre quem deve processar os conteúdos transferidos e, no caso de falha, como codificar mensagens de erro.

2. **Regras de Codificação de Dados:** as regras de codificação possibilitam a troca de dados. Deve-se definir as regras de codificação de tipos de dados específicos transmitidos na mensagem. O SOAP inclui o seu próprio conjunto de convenções para codificar tipos de dados. A maior parte destas convenções são baseadas em XML *Schemas*.
3. **Convenções RPC:** as convenções RPC definem como modelar interações no estilo RPC.

Uma mensagem SOAP, ver Figura 8, consiste basicamente nos seguintes elementos:

- **Envelope:** o envelope define o documento XML como uma mensagem SOAP. Este pode conter declarações de *namespaces* e também atributos adicionais como o que define o estilo de codificação. Um *enconding style* define como os dados são representados no documento XML.

- ***Header***: o *header* é um cabeçalho opcional, que carrega informações adicionais como, por exemplo, se a mensagem deve ser processada por um determinado nó intermediário antes de chegar no destino final. Quando utilizado, o *Header* deve ser o primeiro elemento do Envelope.

- ***Body***: o *body* é um elemento obrigatório, e contém o documento com a informação a ser transportada para o seu destino final. O elemento *Body* pode conter um elemento opcional *Fault*, usado para carregar mensagens de *status* e erros retornadas pelos "nós" ao processarem a mensagem.

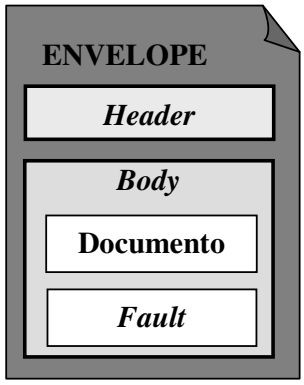

Figura 8 - Camadas da Estrutura SOAP, extraída de [ALO04]

4.1.2 Web Services Description Language (WSDL)

Especificações WSDL são documentos XML que descrevem os WS, bem como suas interfaces de serviços. Ela é o núcleo da estrutura dos WS e estabelece uma forma padronizada de representação dos tipos de dados passados nas mensagens, das operações realizadas sobre estas mensagens e do mapeamento das mensagens sobre os protocolos de transporte. Essas especificações são frequentemente caracterizadas por uma parte abstrata e uma parte concreta (Figura 9). Na parte abstrata são definidos os *portTypes*, *operations*, *messages* e *types*. Já na parte concreta são definidos os protocolos de ligação (*bindings*) e outras informações (*service and ports*).

- **Parte Abstrata**
  - ***Type***: o WSDL necessita que seja especificado o tipo, de forma que os dados a serem trocados possam ser corretamente interpretados durante a comunicação.
  - ***Messages***: cada mensagem é um documento tipado dividido em partes. Cada parte é caracterizada por um nome e por um tipo, o qual é definido no XML *Schema*.
  - ***Operation***: existem quatro operações básicas: *one-way, request-response, solicit-response* e *notification.* As operações *notification* e *one-way* envolve uma simples mensagem. Em interações *one-way*, o cliente invoca um serviço pelo envio de uma mensagem. Nas *notifications*, é o serviço que envia a mensagem. As operações *request-response* e *solicit-response* envolve a troca de duas mensagens.
  - ***PortTypes***: os *portTypes* definem o conjunto de todas as operações que são permitidas.
- **Parte Concreta**
  - ***Interface Bindings*:** essas *interfaces* servem como elemento de ligação entre os elementos abstratos e os concretos.
  - ***Ports*:** os *ports,* também conhecido como *EndPoint*, combinam a informação de *Interface Binding* com o endereço de rede (especificado pela URL) o qual pode ser acessado.

- ***Services*:** os serviços são agrupamentos lógicos de portas, tipicamente disponíveis no mesmo endereço.

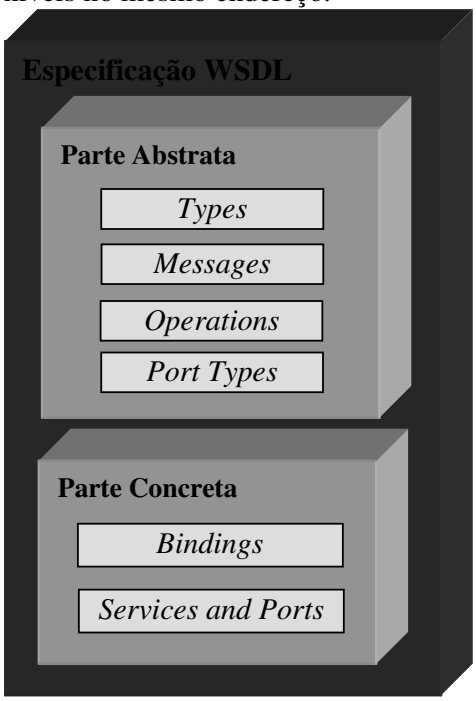

Figura 9 - Elementos do documento WSDL, extraída de [ALO04]

4.1.3 Universal Discovery, Description and Integration (UDDI)

Após a definição dos dados das mensagens (XML), da descrição dos serviços que receberão e processarão as mensagens (WSDL) e também dos meios de envio e recebimento das mesmas (SOAP), é necessário ter-se uma forma de publicar e divulgar o serviço que se oferece e de encontrar os serviços oferecidos por outros. Esta é a função do UDDI, que possui uma estrutura que define um modelo de dados em XML e também interfaces de programação SOAP para registrar, descobrir e integrar informações. Este componente habilita os usuários dos serviços a publicarem e descobrirem estes serviços na *web* [RYM03].

Portanto, pode-se fazer uma analogia do UDDI com as páginas amarelas de uma lista telefônica, onde a especificação UDDI é basicamente um diretório de WS que oferece uma maneira simples de registrar e localizar os serviços.

Em seu núcleo, UDDI consiste de duas partes:

- **API UDDI:** É uma especificação técnica para construir um diretório de WS. A informação UDDI é armazenada dentro de um formato específico XML,

definido por WSDL e XML *Schema*. A especificação inclui detalhes de uma API própria para buscar dados existentes ou publicar novos dados;

- **UDDI *Business Registry*:** também conhecido como "UDDI *cloud services*" é uma implementação operacional completa da especificação UDDI. Tal parte habilita qualquer um a buscar dados UDDI existentes, e também, a qualquer empresa registrar-se a si própria e seus respectivos serviços.

O processo de descoberta de um serviço (*discovery*) é feito através de *registries*. Estes, são repositórios contendo documentos que descrevem os negócios. Um *registry* pode localizar os serviços disponíveis publicamente que sejam adequados às necessidades de uma organização. Os *registries* do UDDI podem ser:

- **Públicos:** os registros públicos permitem que as informações publicadas no registro possam ser examinadas por todos;
- **Privados:** os registros privados são encobertos por um *firewall* de uma organização, sendo acessíveis apenas pelos seus membros;
- **Semiprivados:** os registros semiprivados, são aqueles que possuem um acesso restrito fora da organização.

4.2 Comparação entre Arquiteturas Orientadas a Serviços e outras tecnologias

Os WS são, na verdade, uma evolução das tecnologias CORBA (*Common Object Request Broker Architecture*), DCOM (*Distributed Common Object Model*) e RMI (*Remote Method Invocation*). A Tabela 1 compara as AOS a outras tecnologias, em relação a suas características principais, interlinguagem, interplataforma, tradução dos dados, vantagens e desvantagens. Percebe-se com essa comparação que AOS são superiores as demais tecnologias, tendo como principal característica a interoperabilidade. O primeiro aspecto apresentado é a característica principal dessas tecnologias. Posteriormente discute-se o suporte a diversas linguagens: o RMI não satisfaz esse quesito por limitar-se a utilização da Linguagem Java. No suporte a interplataforma, a tecnologia DCOM não provê suporte, pois somente funciona em plataformas *Microsoft*. Em relação a tradução dos dados, o CORBA considera esse aspecto através da IDL (*Interface Definition Language*), embora bem mais

complexo que o XML, solução adotada pelas AOS. Ao final são apresentadas as vantagens e desvantagens dessas tecnologias.

TABELA 1 – COMPARAÇÃO ENTRE AOS E DEMAIS TECNOLOGIAS

	AOS	CORBA	RMI	DCOM
Característica Principal	O uso de tecnologias baseadas em XML para a construção de WS permite a utilização de serviços sem que haja a necessidade de se saber qual a plataforma ou linguagem de programação foi utilizada na sua construção.	Mecanismo para construir aplicações cliente-servidor em ambientes heterogêneos	Mecanismo específico da Java para realizar chamadas de cliente-servidor, construída sobre a arquitetura de stub/estrutura	Mecanismo da *Microsoft* para realizar chamadas remotas
Interlinguagem	SIM	SIM	NÃO	SIM
Interplataforma	SIM	SIM	SIM	NÃO
Tradução de Dados	XML	IDL (*Interface Definition Language*)	-	-
Vantagens	Suporte a interoperabilidade. Utilização do XML	Podem manipular cargas de transação mais altas porque mantêm uma conexão persistente entre clientes e servidores à custa de servir menos clientes por servidor.	Podem manipular maiores cargas de transação porque a RMI mantém uma conexão persistente entre clientes e servidores à custa de servir menos clientes por servidor.	Possibilidade de ser construído utilizando diversas linguagens (*Visual Basic*, C++, etc).
Desvantagens	Desempenho: o HTTP não mantém conexões de longa duração e a necessidade de escolher linguagens eficientes de programação do lado servidor para construir seu serviço.	Complexidade do IDL	Limita-se a uma solução unicamente Java no cliente e no servidor.	Funciona somente em plataformas *Microsoft*.

4.3 Considerações Adicionais

Este capítulo apresenta conceitos relacionados a arquiteturas orientadas a serviços, discorrendo sobre seus principais componentes, e como esses componentes, quando interligados, podem auxiliar no tratamento adequado da heterogeneidade. Ainda, ao final deste capítulo compararam-se os WS com outras tecnologias.

O interesse deste trabalho em tratar a heterogeneidade, não se resume ao fato de diferentes plataformas, linguagens e ferramentas. Além desses aspectos, existe ainda a heterogeneidade resultante da especialização do OSSP em PDSP, o que resulta em diferentes tipos de projetos, com ciclos de vida diferenciados e formas de gerenciamento variadas (Capítulo2).

O capítulo seguinte, discorre sobre como a partir de fontes heterogêneas é possível extrair dados brutos e transformá-los em informações consistentes e de qualidade, e posteriormente armazená-las em uma fonte integrada de dados. Já no Capítulo 7, apresenta-se a solução proposta neste trabalho, onde conceitos de arquiteturas orientadas a serviços apresentados aqui são utilizados como uma solução viável e adequada.

5 PROCESSO DE DESCOBERTA DE CONHECIMENTO EM BANCO DE DADOS

Este capítulo discorre sobre o Processo de Descoberta de Conhecimento em Banco de Dados, destacando o Processo de Extração, Transformação e Carga. As etapas que compõem o processo de ETC são detalhadas, bem como a criticidade deste quando aplicado em ambientes heterogêneos.

Segundo [HAN01], o processo de Descoberta de Conhecimento em Banco de Dados, conhecido como KDD (*Knowledge Discovery in Databases*) consiste em uma seqüência iterativa dos seguintes passos (Figura 10):

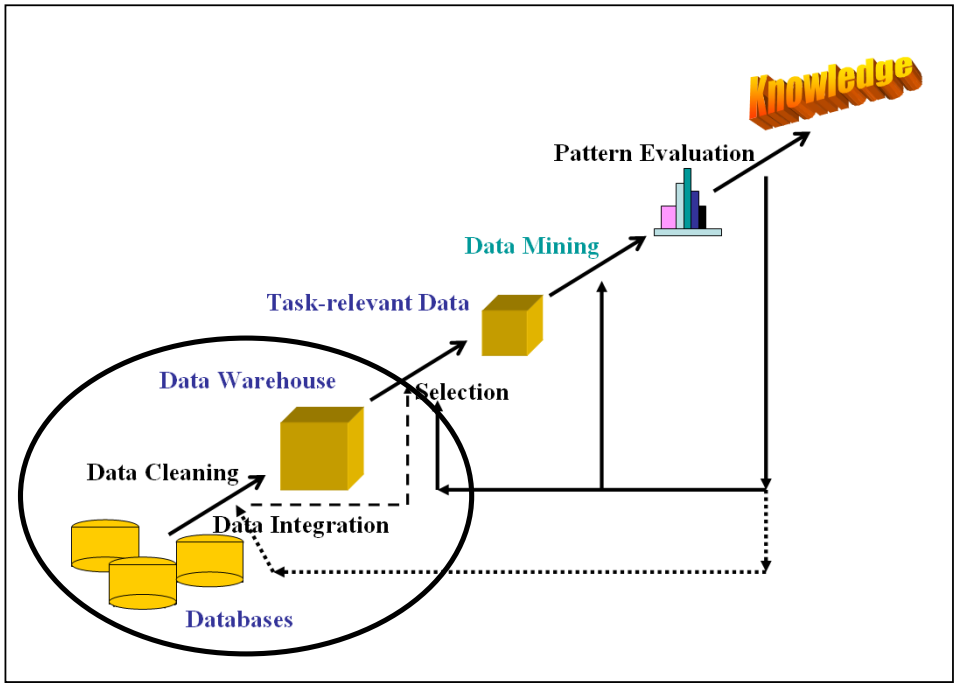

Figura 10 - Etapas do Processo de Descoberta de Conhecimento, extraída de [HAN01]

1. **Limpeza dos Dados:** etapa em que devem ser removidas as perturbações e as inconsistências dos dados.

2. **Integração dos Dados:** etapa na qual múltiplas fontes de dados devem ser combinadas em uma única.
3. **Seleção dos Dados:** etapa onde os dados relevantes para as tarefas de análise devem ser recuperados das fontes de dados.
4. **Transformação dos Dados:** etapa na qual os dados brutos devem ser transformados e consolidados em informações relevantes para o negócio.
5. **Mineração dos Dados:** etapa na qual métodos inteligentes são aplicados a fim de extrair padrões. Vários algoritmos podem ser utilizados nessa etapa, tais como, associação, classificação, segmentação, seqüência e sumarização.
6. **Avaliação de Padrões:** etapa na qual são identificados padrões interessantes, ou seja, que possam representar conhecimento baseado em alguma métrica.
7. **Apresentação do Conhecimento:** etapa na qual são utilizadas técnicas de representação do conhecimento e visualização para apresentar o conhecimento minerado para o usuário final.

Este trabalho considera somente as etapas referentes ao processo de ETC, do processo de KDD (círculo na Figura 10). O processo de ETC consiste na integração de dados de múltiplas fontes e sua limpeza e transformação em informações consistentes e de qualidade, e na inserção destes em um DW. [KIM98] utiliza a denominação de ferramentas de *Back End,* as quais são responsáveis pelo processo de ETC, bem como a restauração dos dados utilizados num sistema de DW. O processo de ETC é detalhado na seqüência.

5.1 Processo de Extração, Transformação e Carga

A Figura 11 ilustra um *framework* genérico para processos de ETC. Na parte superior descreve-se os dados armazenados que estão envolvidos em todo processo. Na parte esquerda, observa-se os dados brutos provenientes de diversas fontes de dados (e.g. banco de dados relacional e arquivos diversos). Os dados provenientes dessas fontes são extraídos através de rotinas de extração, e podem ser obtidos através de extrações completas, *snapshot* completo, ou extrações parciais, quando somente os dados que sofreram alguma alteração são extraídos. Dessa forma, esses dados devem ser propagados para o DSA (*Data Staging Area*) onde são limpos e transformados antes de serem carregados no DW. O DW engloba as tabelas fato e

dimensão. Já a parte inferior da Figura 11 ilustra todas as bases de dados utilizadas para extração, limpeza e carga dos dados, respectivamente.

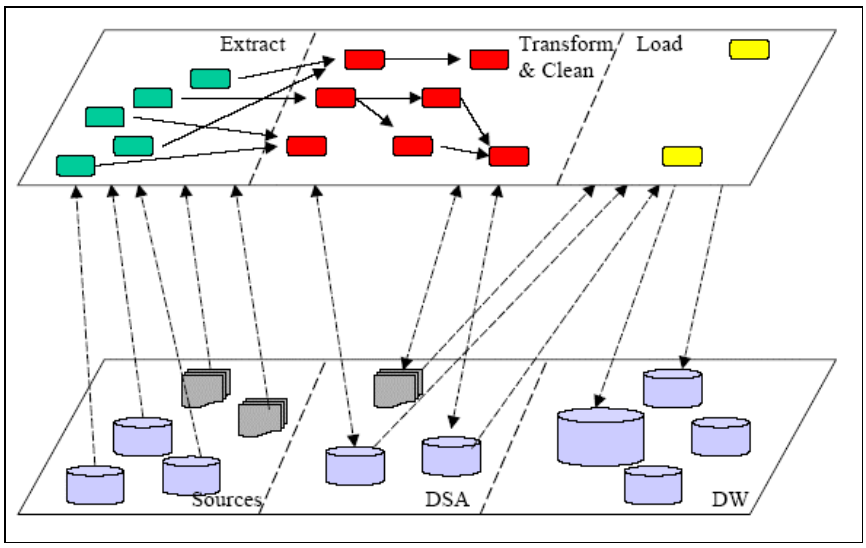

Figura 11 - Processo de Extração, Transformação e Carga, extraída de [VAS02]

Em um PDS, a ETC significa poder capturar informações da execução de processos completados e ainda em execução, e dos artefatos tratados e produzidos neste processo. A idéia é efetuar as transformações e tratamentos necessários para obter informações consistentes, úteis e de qualidade, a serem armazenadas em um DW. Estas informações devem possibilitar o monitoramento de processo, indicar problemas e também servir de fontes de dados para prover a descoberta de conhecimento em processos de desenvolvimento de *software*.

5.1.1 Extração

Os sistemas transacionais, também chamados de sistemas fontes ou sistemas legados, são fontes interessantes de informação para o negócio [KIM98]. Essas informações porém, podem ser encontradas em diversos locais e em diversos formatos. Apesar de estas fontes serem de grande interesse para o negócio, nem todas as informações contidas nelas podem fornecer algum valor para objetivo do negócio, e nem todas podem ser analisadas. Isso muitas vezes deve-se ao fato de que esses sistemas fontes mantêm pouca ou nenhuma informação histórica, e ainda possuem um enorme volume de dados.

Na etapa de extração, os dados a serem extraídos devem estar relacionados com o modelo de dados do DW alvo. Esses dados armazenados nos sistemas fontes, muitas vezes são incompletos, duplicados, inconsistentes e ruidosos, necessitando assim, de diversas limpezas e transformações, apresentadas a seguir.

5.1.2 Transformação e Limpeza

Rotinas de limpeza e transformação de dados buscam eliminar problemas dos dados, provenientes dos sistemas fonte que serão carregados no DW, através do preenchimento de valores nulos, eliminação de dados ruidosos e inconsistentes. Além disso, essas devem buscar unificar dados provenientes de múltiplas fontes de dados em dados integrados, consistentes e que posteriormente serão armazenados no DW. Essa etapa tem por objetivo fornecer ao usuário do sistema analítico, dados concisos e com uma qualidade que permita uma tomada de decisão baseada em valores mais próximos dos reais.

5.1.2.1 *Data Staging Area* (DSA)

Segundo [KIM98], o processo de *data staging*[4] é a parte mais crítica do projeto de DW. Sendo assim, busca-se minimizar essa dificuldade através da construção de uma área onde os dados limpos e transformados devem ser mantidos antes de serem carregados no DW. [KIM98] propõe um plano de 10 passos para a criação de um DSA, os quais foram seguidos por este trabalho.

1. Criar um plano de alto nível, através de um esquema de uma página ilustrando o fluxo da fonte para o destino.
2. Testar, escolher e implementar uma ferramenta de *data staging*.
3. Diminuir a granularidade da tabela alvo, esboçando graficamente qualquer transformação e reestruturação de dados complexa, e ilustrando, também, o processo de geração de chaves *surrogates*.
4. Construir e testar a carga de uma dimensão estática. O objetivo principal deste passo é desenvolver uma infra-estrutura que suporte conectividade, transferência de arquivos e problemas de segurança.
5. Construir e testar minunciosamente processos de mudanças para uma dimensão.

[4] Devido ao uso comum na área, este termo será mantido em inglês.

6. Construir e testar cargas nas dimensões restantes.
7. Construir e testar cargas nas tabelas fato históricas, incluindo substituição de chaves *surrogates*.
8. Construir e testar o processo de carga incremental.
9. Construir e testar a carga em tabelas agregadas e/ou carga em MOLAP.
10. Projetar, construir e testar a automação da aplicação de *staging*.

[KIM98] afirma que investir em um sólido DSA permite trabalhar com uma larga escala de requisitos de transformações evitando que cada processo de negócio necessite investir em decifrar sistemas fonte e executar os mesmos cálculos inúmeras vezes.

5.1.2.2 Rotinas de Limpeza e Transformação

Uma vez os dados tenham sido extraídos dos sistemas fonte, inúmeras ações devem ser executadas até transformar esses dados em informações que sejam apresentáveis para o usuário e, além disso, possuam algum valor para o negócio. [KIM98] apresenta 20 passos de transformações, citados a seguir:

1. Integração
2. Manutenção de mudanças nas dimensões
3. Desnormalização
4. Renormalização
5. Limpeza
6. Deduping
7. Merge
8. Purge
9. Conversão de tipo de dados
10. Cálculo
11. Derivação
12. Alocação

13. Agregação

14. Auditoria do conteúdo dos dados

15. Auditoria da linhagem dos dados

16. Análises específicas para transformação

17. Utilização de ferramentas específicas para transformação

18. Valores nulos

19/20. Pré e pós passos de saída

5.1.3 Carga

Após os dados extraídos serem limpos e transformados, eles devem ser carregados no DW. A carga dos dados é feita a partir das informações armazenadas no DSA que já passaram pelo processo de limpeza e transformação. As tabelas que serão atualizadas no sistema de DW devem ser montadas utilizando-se agregações, sumarizações e ordenações dos dados. A carga dos dados no DW dá-se inicialmente nas tabelas dimensão e posteriormente nas tabelas fato. Outro fator a ser considerado é o tipo de carga a ser realizada. Esta pode ser do tipo incremental ou por cima dos dados. A carga incremental normalmente é feita para tabelas fatos e a carga por cima dos dados é feita em tabelas dimensões onde o analista terá que excluir os dados existentes e incluí-los novamente. Em alguns casos poderá acontecer que as tabelas de dimensões tenham que manter o histórico. Então, o mesmo deverá ser mantido. O processo de carga, envolve grande complexidade e considera diversos aspectos, tais como integridade referencial e atualização de informações no DW.

Em DW, integridade referencial significa que para cada chave estrangeira da tabela fato existe uma entrada correspondente na tabela dimensão. Dessa forma, no momento da carga é necessário verificar os campos das tabelas fato que são chaves estrangeiras para certificar-se de que os dados existentes nas tabelas fato estão de acordo com a chave primária da tabela dimensão corresponde.

Ao final do processo de carga, os dados armazenados no DW estarão prontos para serem analisados pelos gestores. As seções seguintes descrevem o DW, propriamente dito, e os processos de atualizações, bem como o impacto dessas atualizações.

5.1.3.1 Data Warehouse (DW)

Um DW é uma base de dados semanticamente consistente que serve como uma implementação física do modelo de dados de suporte a decisão organizacional e armazena as informações que uma organização necessita para a tomada de decisão estratégica. Um DW é também visto como uma arquitetura construída através da integração de dados provenientes de múltiplas fontes para suportar consultas estruturadas e/ou *ad-hoc*, relatórios analíticos e tomada de decisões.

Segundo [INM96], um "DW é uma coleção de dados orientada a assuntos, integrada, variante no tempo e não volátil no suporte ao processo de tomada de decisão". Esta definição apresenta as maiores características de um DW, onde através dessas quatro palavras chaves consegue-se distinguir DW de qualquer outro sistema de repositório de dados, tais como sistemas de banco de dados relacional, sistemas de processamento transacional e sistemas de arquivos. Essas características são detalhadas como segue.

- **Orientado a assunto:** um DW é organizado em torno dos assuntos principais, tais como projeto, iteração, fase e atividade. Melhor do que se concentrar no processamento de operações e transações diárias de uma organização, um DW focaliza na análise dos dados para tomadores de decisões. Portando, o DW fornece tipicamente uma visão concisa através dos assuntos e pela exclusão dos dados que não são úteis no processo de suporte a decisão.
- **Integrado:** um DW é construído através da integração de múltiplas fontes heterogêneas, tais como sistemas de banco de dados relacional, sistemas de processamento transacional e sistemas de arquivos. As técnicas de integração e limpeza de dados são aplicadas para assegurar a consistência dos dados.
- **Variante no tempo:** os dados são armazenados para prover informações pela perspectiva histórica (e.g., 5 a 10 anos). Qualquer chave no DW contém, implícita ou explicitamente, um elemento de tempo.
- **Não volátil:** um DW é sempre um repositório fisicamente separado com dados transformados provenientes do ambiente transacional. Devido a esta separação, um DW não requer processamento de transação, recuperação e mecanismos do controle de concorrência. Isto geralmente requer somente duas operações no acesso aos dados: o carregamento inicial dos dados e o acesso aos dados.

Data warehousing é o processo de construção e utilização de DW. A construção de DW requer a integração, limpeza e consolidação de dados. A utilização de DW frequentemente necessita de diversas ferramentas de suporte a decisão. O processo de *data warehousing* é também útil pelo ponto de vista da integração de bases de dados heterogêneas. Muitas organizações tipicamente armazenam e mantêm diversos tipos de dados provenientes de grandes fontes de informações heterogêneas, autônomas e distribuídas.

O processo de *Data Warehousing* provê arquiteturas e ferramentas para executivos de negócio organizarem, entenderem e utilizarem seus dados na tomada de decisões estratégicas. Diversas organizações encontraram nos sistemas de DW valiosas ferramentas no mundo atual competitivo [HAN01].

5.1.3.2 Atualização de valores nas tabelas fato e dimensão

O processo de carga deve conter regras para determinar como lidar com valores de atributos que sofreram alguma alteração em relação ao valor já armazenado no DW. A atualização de valores no DW é um dos processos mais complexos, pois as informações carregadas no DW passaram pela etapa de limpeza e transformação desde sua extração das fontes de dados originais. Dessa forma, é necessário, por exemplo, manter um registro no DSA quando chaves substitutas forem utilizadas.

A aplicação de *data staging* deve conter as regras do negócio para determinar como lidar com valores de atributos que sofreram alguma modificação em relação ao valor armazenado no DW. [KIM98] apresenta técnicas que podem ser utilizadas quando mudanças dos valores armazenados nas tabelas dimensão e fato ocorrem:

- **Tipo 1: Sobrescrita:** quando os atributos das tabelas dimensão são atualizados, deve-se sobrescrevê-los.
- **Tipo 2: Criação de um novo registro na dimensão:** quando os atributos das tabelas dimensão são atualizados, deve-se criar um novo registro na tabela. Segundo [KIM98], esta técnica é a mais utilizada e mais difícil de ser gerenciada. Isto porque se deve criar uma nova linha na tabela e uma nova chave atualizada toda vez que se encontrar uma mudança em um registro da tabela.

5.2 Considerações Adicionais

Este capítulo relatou que a etapa de ETC do Processo de Descoberta de Conhecimento em BD é uma das mais importantes, pois essa etapa busca garantir que os dados armazenados no DW possuam um nível de qualidade adequado e possam ser utilizadas pelos gestores para a tomada de decisão.

Esse capítulo serviu como embasamento teórico para a solução proposta, Capítulo 7. A Seção 5.1.2.1 apresentou os passos para a criação de uma DSA, já a Seção 5.1.2.2 descreveu as rotinas de extração utilizadas e a Seção 5.1.3 apresentou o processo de carga, bem como a atualização de informações.

A criticidade do processo de ETL torna-se mais elevada ainda quando esta é aplicada em ambientes heterogêneos, devido à necessidade de um maior tratamento dos dados, pois estes dados podem encontrar-se espalhados em diversas fontes, em diversos formatos, com significados diferentes, inconsistentes entre outros. O Capítulo 6 discorre sobre um ambiente real, onde esses diversos fatores críticos para a heterogeneidade estão presentes.

6 ESTUDO DE AMBIENTE REAL

Este capítulo apresenta o estudo de um ambiente real realizado em uma organização de software. Para tal, são apresentadas a descrição do cenário e a caracterização do problema. Neste, foram identificadas limitações no acompanhamento dos seus PDS devido à dificuldade de se coletar e armazenar dados provenientes dos diversos projetos em um ambiente centralizado, que forneça uma visão organizacional unificada.

6.1 Descrição do Cenário

Esta seção tem por objetivo apresentar a estrutura da organização, discorrendo sobre o nível de maturidade, programa de métricas e modelo analítico adotado pela organização, bem como a caracterização do problema, descrevendo os três pontos principais a ser observados: heterogeneidade, intrusão e extração não incremental.

6.1.1 Estrutura da Organização

O estudo de um ambiente real foi realizado em uma organização de *software*, certificada CMM Nível 2. Esta tem como um de seus fundamentos a adoção de um modelo de gestão em qualidade a fim de utilizar um conjunto de processos e práticas definidas e específicas, capazes de satisfazer as necessidades do mercado, de modo a atingir os seguintes objetivos:

- Habilitar um desenvolvimento rápido e sustentável para projetos de desenvolvimento, manutenção e migração de *software*.
- Verificar e definir os processos de desenvolvimento de *software*, priorizando as ações de melhoria da qualidade ao longo de todo o ciclo de vida dos produtos.

- Fornecer, à Operação de *Software* vantagens competitivas no mercado de desenvolvimento de *software*.

Esta organização segue a analogia de desenvolvimento de produto, conforme ilustrado na Figura 2 (Capítulo 2), onde, a partir de um conjunto de processos de *software* padrão (OSSP), cada projeto pode-se especializar (PDSP), possuindo assim, características próprias, tais como, distintos tipos, distintos ciclos de vida, distintos modelos de gerenciamento e o uso de distintas ferramentas.

6.1.2 Nível de Maturidade

Partindo da premissa que a organização busca certificação CMM Nível 3 e que um dos requisitos necessários para obtenção desse nível de maturidade é a coleta e o armazenamento de métricas do ambiente transacional, foi necessária a construção de uma Base Organizacional (BO) que armazene dados históricos dos projetos da organização, ou seja, que nessa base sejam armazenadas as métricas referentes à execução desses projetos. Entre os objetivos da construção de uma BO para a organização alvo, destacam-se:

- **Planejamento de projeto:** os dados de projetos finalizados devem poder ser utilizados como base para o planejamento de novos projetos e no estabelecimento de metas realistas para os mesmos. Através da análise do desempenho de projetos similares já finalizados, as atividades de planejamento de recursos, estimativas de prazo, tamanho e custo tornam-se mais precisas.
- **Controle do desempenho dos processos dos projetos:** as métricas fornecem informações acuradas sobre o estado do projeto e, assim, podem ser utilizadas para tomada de ações corretivas em tempo hábil. Métricas são utilizadas para monitorar o desempenho do projeto e para verificar sua conformidade em relação aos planos de desenvolvimento do projeto como, por exemplo, os critérios de gatilho e replanejamento[5].
- **Análise e melhoria dos processos da organização:** somente através de métricas uma organização pode quantificar a qualidade e a produtividade de seus processos. Também, através de métricas, a identificação de potenciais áreas de melhoria torna-se mais visível.

[5] Os critérios de gatilho e replanejamento sinalizam situações nas quais são necessárias o replanejamento do projeto, como por exemplo, quando a variação de esforço for superior a 10% o projeto deve ser reestimado.

6.1.3 Programa de Métricas

A organização em questão definiu seu programa de métricas, apresentado na Tabela 2. Para cada métrica indireta, apresentam-se os seus objetivos, as métricas diretas que são utilizadas no seu cálculo, bem como sua equação de cálculo e unidade final. Ainda, para cada métrica direta apresentam-se as possíveis origens, já que essas podem ser provenientes de diversas fontes.

TABELA 2 - PROGRAMAS DE MÉTRICAS DA ORGANIZAÇÃO ALVO

Métricas Indiretas	Objetivos	Métricas Diretas	Origem	Equação	Unidade
Variação de Cronograma no Baseline Original (VcrBO)	Dar visibilidade da aderência do projeto em relação aos compromissos do projeto, possibilitando que o mesmo seja entregue no prazo.	Data Final *Baseline* Original (DFBO)	Project	$VCrBO = \frac{(DFR - DFBO)}{(DIBO - DFBO)} \times 100$	%
		Data Inicial *Baseline* Original (DIBO)	Project		
		Data Final Real do Projeto (DFR)	Project, Excel		
Variação de Cronograma no Baseline Revisado (VCrBR)		Data Final *Baseline* Revisado (DFBR)	Project	$VCrBR = \frac{(DFR - DFBR)}{(DIBR - DFBR)} \times 100$	%
		Data Inicial *Baseline* Revisado (DIBR)	Project		
		Data Final Real do Projeto (DFR)	Project, Excel		
Variação de Esforço no Baseline Original (VEBO)	Dar visibilidade da diferença entre o esforço estimado para o projeto e o esforço realizado.	Esforço Estimado *Baseline* Original (EEBO)	Project	$VEBO = \frac{(ER - EEBO)}{EEBO} \times 100$	%
		Esforço Real (ER)	Project, Excel		
Variação de Esforço Baseline Revisado (VEBR)		Esforço Estimado *Baseline* Revisado (EEBR)	Project	$VEBR = \frac{(ER - EEBR)}{EEBR} \times 100$	%
		Esforço Real (ER)	Project, Excel		
Variação de Tamanho no Baseline Original (VTBO)	Fornecer visibilidade da diferença entre o tamanho previsto para o projeto e do tamanho realizado, buscando dimensionar a magnitude do produto ou serviço a ser desenvolvido.	Tamanho Estimado *Baseline* Original (TEBO)	Project, Clear Quest (CQ)	$VTBO = \frac{(TR - TEBO)}{TEBO} \times 100$	%
		Tamanho Real (TR)	Project, CQ		
Variação de Tamanho no Baseline Revisado (VTBR)		Tamanho Estimado *Baseline* Revisado (TEBR)	Project, CQ	$VTBR = \frac{(TR - TEBR)}{TEBR} \times 100$	%
		Tamanho Real (TR)	Project, CQ		

Métricas Indiretas	Objetivos	Métricas Diretas	Origem	Equação	Unidade
Variação de Custo no Baseline Original (VCBO)	Fornecer visibilidade da diferença entre o custo realizado de um projeto frente ao custo previsto.	Custo *Baseline* Original (CEBO)	Project	$VCBO = \frac{(CR - CEBO)}{CEBO} \times 100$	%
		Custo Real (CR)	Project		
Variação de Custo no Baseline Revisado (VCBR)		Custo *Baseline* Revisado (CEBR)	Project	$VCBR = \frac{(CR - CEBR)}{CEBR} \times 100$	%
		Custo Real (CR)	Project		
Custo da Qualidade	Determinar os custos incorridos para atingir qualidade no projeto, relacionados a garantia de qualidade, controle da qualidade, planejamento da qualidade e retrabalho.	Custo Real (CR)	Project, Excel	$CQ = \frac{(CR + CTT + CP + CNC)}{CTR}$	%
		Custo total de revisão (CTR)	Project, Excel		
		Custo total de teste (CTT)	Project, Excel		
		Custo de prevenção (CP)	Project, Excel		
		Custo de não conformidades (CNC)	Project, Excel		
Volatilidade de Requisitos	Fornecer a relação entre o n° de requisitos modificados pelo cliente (adicionados, removidos, alterados) e o n° de requisitos aprovados inicialmente pelo cliente.	Nº de requisitos adicionados (RA)	Excel	$VR = \frac{(RA + RR + RM)}{RO}$	%
		Nº de requisitos removidos (RR)	Excel		
		Nº de requisitos modificados (RM)	Excel		
		Nº de requisitos originais (RO)	Excel		
Densidade de Defeitos Entregues (DDE)	Prover visibilidade da relação entre o número de defeitos encontrados pelo cliente e o tamanho atual do produto.	Nº defeitos encontrados no cliente (DE)	CQ	$DDE = \frac{DE}{TR\_KLOC}$	$\frac{Defeitos}{KLOC}$
		Tamanho Real em KLOC (TR_KLOC)	Project, CQ		
Densidade de Defeitos (DDI)	Prover visibilidade da relação entre o n° de defeitos encontrados durante todo o ciclo de vida de desenvolvimento do produto e o tamanho atual do produto.	Nº defeitos encontrados internamente (DI)	CQ	$DDI = \frac{DI}{TR\_KLOC}$	$\frac{Defeitos}{KLOC}$
		Tamanho Real em KLOC (TR_KLOC)	Project, CQ		

Métricas Indiretas	Objetivos	Métricas Diretas	Origem	Equação	Unidade
Eficiência de Remoção de Defeitos (ERD)	Fornecer visibilidade da efetividade das atividades de verificação e validação executadas no projeto.	Nº defeitos encontrados internamente (DI)	CQ	$ERD = \frac{DI}{(DI + DE)} \times 100$	%
		Nº defeitos encontrados no cliente (DE)	CQ		
Eficiência de Revisão (ERE)	Prover visibilidade dos defeitos encontrados em revisões do tipo *Peer Review*.	Nº defeitos encontrados em revisões (DTR)	CQ	$ERE = \frac{DTR}{ETR} \times 100$	%
		Esforço total em revisão (ETR)	Project, Excel		
Produtividade (P)	Fornecer visibilidade da taxa de trabalho despendido para desenvolvimento de um determinado projeto.	Esforço Real (ER)	Project, Excel	$P = \frac{ER}{TR\_KLOC}$	$\frac{Horas}{KLOC}$
		Tamanho Real em KLOC (TR_KLOC)	Project, CQ		

6.1.4 Modelo Analítico

6.1.4.1 Proposta Inicial

O DW, como apresentado no Capítulo 5, é uma base de dados unificada e centralizada. Esta permite análises dos PDS que suportam o programa de métricas da organização de acordo com diferentes perspectivas de análise, níveis de sumarização e papéis organizacionais. Supõe-se que os projetos de *software* estão organizados como descritos na Figura 12, através de um diagrama de classe UML. Os atributos das tabelas fato referem-se às métricas diretas que devem ser coletadas. O modelo analítico proposto comporta a execução de projetos organizados em fases e atividades, segundo os ciclos de vida adotados pela organização, dos quais resultam em um ou mais produtos de *software*. A modelagem do modelo analítico não faz parte do escopo deste trabalho [RUI05, NOV05].

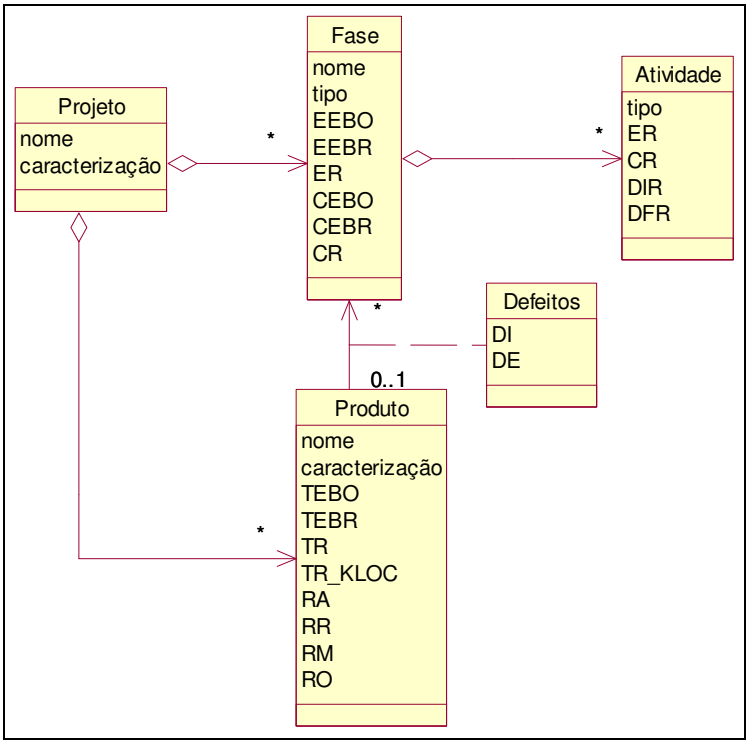

Figura 12 - Estruturas dos Projetos

A caracterização dos projetos é realizada através de suas fases, divisão destas em atividades e dos produtos resultantes. Sobre estes projetos são capturadas métricas diretas necessárias para os cálculos das métricas derivadas definidas pela organização. Neste sentido, as tabelas dimensão descrevem projeto, produto, fase, atividade, defeito, *status* do projeto e *status* da fase (concluído ou em desenvolvimento), e os tipos de fatos (reais e previstos). A Tabela 3 nomina e descreve sucintamente cada dimensão.

As tabelas do modelo analítico representam um esquema do tipo constelação de fatos, ilustrado na Figura 13. As tabelas fato, descritas sucintamente na Tabela 4, representam as medidas de projetos em diferentes níveis de granularidade, sendo que a informação de maior granularidade, prevista pelo modelo, é representada pela Fato_Atividade e a de menor granularidade pela Fato_Produto. Considerando as medidas diretas definidas na Tabela 2, tem-se que: tamanhos (TEBO, TEBR, TR, RA, RR, RM e RO) estão presentes em Fato_Produto; esforços (EEBR, EEBO e ER) e custos (CEBO, CEBR e CR) estão em Fato_Atividade e Fato_Fase; e defeitos (DI e DE) estão em Fato_Produto/Fase. Cada métrica direta é caracterizada pelas associações que tem com as dimensões. Por exemplo, em um mesmo projeto, diferencia-se o tamanho estimado (TEBO e TEBR) do tamanho real (TR) pela

ligação destes últimos para diferentes entradas em Dim_Tipo_Fato: *baseline* original, *baseline* revisado ou realizado.

TABELA 3 - DIMENSÕES DO MODELO ANALÍTICO

Nome da Dimensão	Descrição da Dimensão
Dim_Projeto	Armazena informações que a organização julga importante referente a algum projeto. (Exemplo: porte, tecnologia empregada, etc)
Dim_Produto	Armazena informações referentes a produtos de projetos.
Dim_Fase	Armazena informações referentes a fases de projetos.
Dim_Atividade	Define atividades de uma fase de projeto, juntamente com seu tipo (trabalho, retrabalho e revisão).
Dim_Defeito	Caracteriza os defeitos encontrados por categoria (interno ou externo) e severidade (baixa, média ou alta).
Dim_Status	Define o produto ou fase de um projeto como: em desenvolvimento ou concluído.
Dim_Tempo	Armazena datas (data, ano, mês, dia e semestre).
Dim_Tipo_Fato	Define se o fato é uma estimativa (*baseline* original, *baseline* revisado) ou registro de uma realização.

Ainda, analisando as métricas utilizadas no cálculo das métricas indiretas percebe-se que essas equações nem sempre são somente formadas por métricas diretas. Algumas vezes, métricas indiretas são combinadas com métricas diretas na produção de outras métricas indiretas. Por exemplo, a métrica "Custo da Qualidade" é formada pela razão do CTR, CTT, CP e CNC pelo CR, onde CTR é o custo total das atividades do tipo revisão de uma *release*, CTT é o custo total das atividades do tipo trabalho da fase de teste de uma *release*, CP é o custo total das atividades do tipo qualidade e CNC é o custo das atividades do tipo retrabalho. Essas métricas, CTR, CTT, CP e CNC, são exemplos de métricas indiretas utilizadas no cálculo de outra métrica indireta.

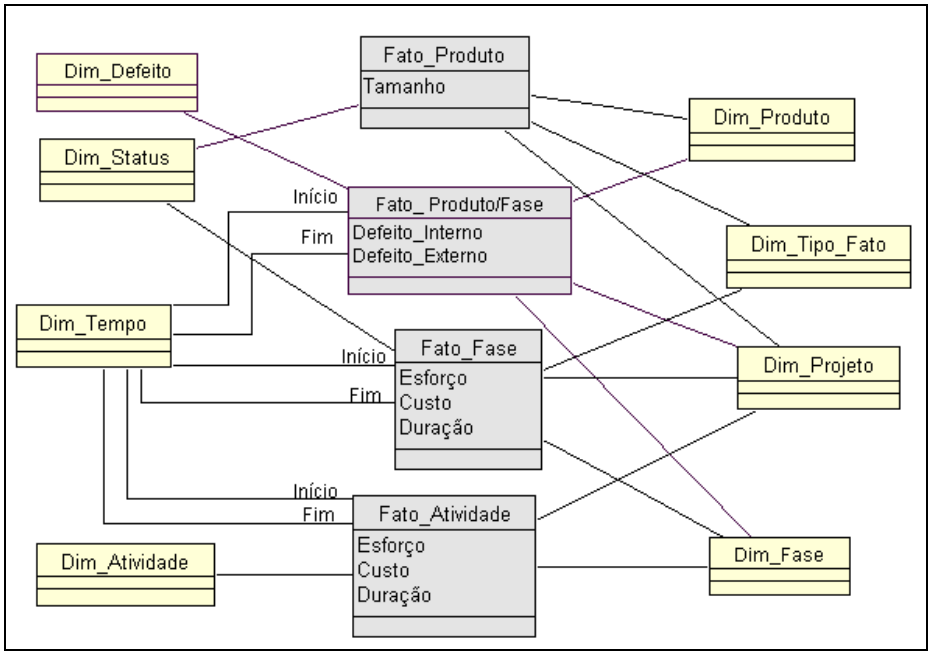

Figura 13 - Modelo Analítico

TABELA 4 - FATOS DO MODELO ANALÍTICO

Nome do Fato	Descrição do Fato
Fato_Atividade	Armazena as medidas de esforço, custo e duração de uma atividade específica.
Fato_Fase	Armazena as medidas estimadas e reais de uma fase. Estas são: esforço, custo e duração.
Fato_Produto/Fase	Armazena os defeitos (internos e externo) de um produto com sua respectiva fase.
Fato_Produto	Armazena as medidas de tamanho de um produto.

6.1.4.2 Modelo Final

Devido às necessidades organizacionais, o modelo apresentado evoluiu. Esta nova estruturação deve-se ao fato da organização ter adquirido uma ferramenta de BI (Business Intelligence), a qual necessita de uma determinada estruturação das tabelas do DW possibilitando o detalhamento da informação, através de operações *drill-up* e *drill-down*.

Neste modelo analítico, as tabelas dimensão descrevem os atributos que caracterizam os projetos: Dim_Projeto, Dim_Indústria, Dim_Cliente, Dim_Porte_Projeto, Dim_Tipo_Projeto, Dim_Tecnologia, Dim_Iteração, Dim_Fase, Dim_Atividade,

Dim_Defeito, Dim_Status, Dim_Unidade, Dim_Tempo_Ini e Dim_Tempo_Fim. A Tabela 5 nomina e descreve sucintamente cada dimensão.

TABELA 5 - DIMENSÕES DO MODELO ANALÍTICO

Dimensão	Descrição
Dim_Projeto	Armazena informações referentes aos projetos.
Dim_Industria	Armazena informações referentes aos tipos de indústria (e.g. Governo, Finanças)
Dim_Cliente	Armazena informações referentes ao nome dos clientes de cada projeto.
Dim_Porte_Projeto	Armazena informações referentes ao porte dos projetos (e.g. grande, médio e pequeno).
Dim_Tipo_Projeto	Armazena informações referentes ao tipo do projeto (e.g. manutenção, desenvolvimento).
Dim_Tecnologia	Armazena informações referentes à tecnologia empregada nos projetos (e.g. Java, C++, VB).
Dim_Iteração	Armazena informações referentes a iterações de projetos (e.g. Iteração 01, Iteração 02).
Dim_Fase	Armazena informações referentes a fases de projetos (e.g. Planejamento e Análise de Requisitos, Design).
Dim_Atividade	Armazena as atividades de uma fase de projeto, juntamente com seu tipo (e.g. trabalho, retrabalho e revisão).
Dim_Defeito	Armazena os defeitos encontrados por categoria (e.g. interno ou externo), tipo (classifica se defeitos internos são de *peer review* ou não) e severidade (e.g. baixa, média ou alta).
Dim_Status	Armazena informações sobre o *status* referente a um projeto uma *release* ou fase de um projeto (Concluído ou Em desenvolvimento).
Dim_Unidade	Armazena informações sobre a unidade utilizada por cada projeto (e.g. PF (Ponto de Função), PMHP (Ponto Médio *Hewlett*-Packard) ou UCP (*Use Case Points*))
Dim_Tempo_Ini	Armazena as datas inicias em relação ao BO, BR e Real.
Dim_Tempo_Fim	Armazena as datas finais em relação ao BO, BR e Real.

Esse modelo foi adotado pela organização e para a pesquisa em questão não interessa sua construção, e sim, como obter as métricas que o compõe, bem como, estas devem ser armazenadas e atualizadas. As tabelas do modelo analítico (Figura 14, Figura *15*, Figura *16*, Figura *17* e Figura *18*) representam um esquema do tipo constelação de fatos. Essas evoluções do modelo são devido à heterogeneidade dos projetos em relação a seu ciclo de vida, cascata e iterativo. No modelo inicial não é possível armazenar informações relativas a projetos com ciclo iterativo.

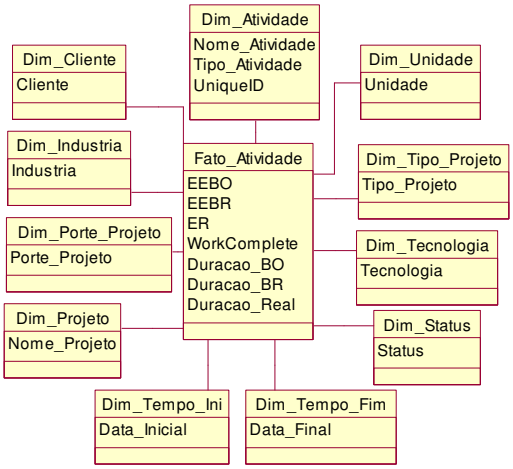

Figura 14 - Fato Atividade

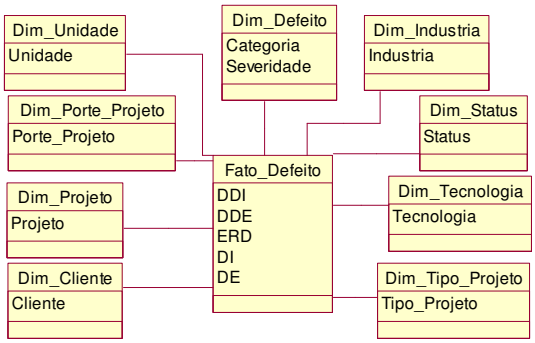

Figura 17 - Fato Defeito

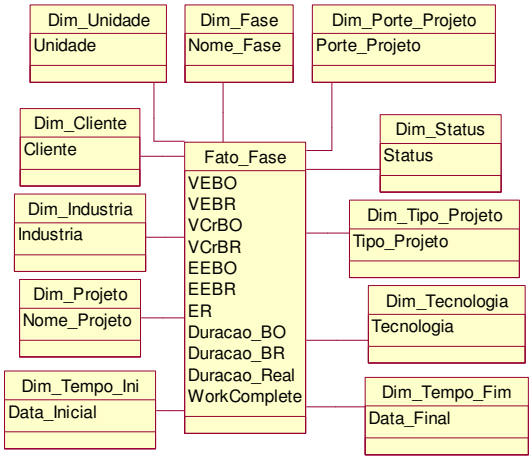

Figura 15 - Fato Fase

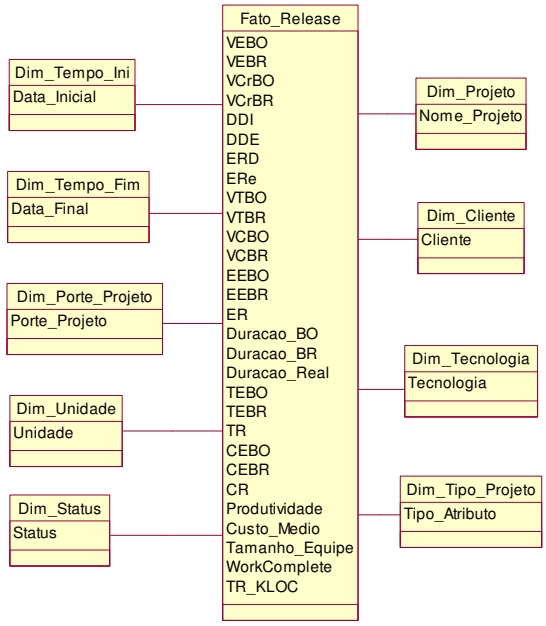

Figura 18 - Fato Release

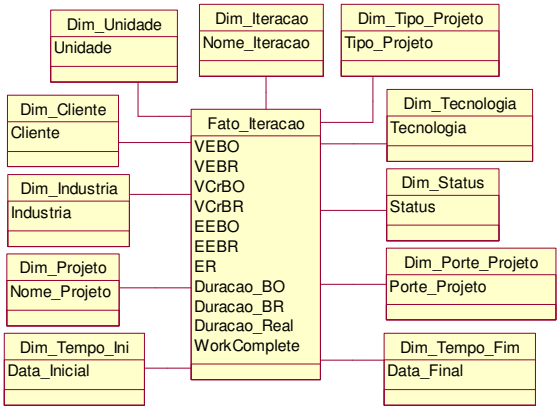

Figura 16 - Fato Iteração

Na Tabela 2, apresenta-se as seguintes métricas: DIBO, DFBO, DIBR, DFBR, DIR e DFR. Já nas Figura 14 a Figura 18, essas métricas não aparecem nas tabelas Dim_Tempo_Ini e Dim_Tempo_Fim. Isso acontece porque para cada métrica existe uma data inicial e uma data final. Por exemplo, para métrica EEBO, têm-se uma DIBO e uma DFBO; para a métrica EEBR, têm-se uma DIBR e uma DFBR e para a métrica ER, têm-se uma DIR e uma DFR.

TABELA 6 - FATOS DO MODELO ANALÍTICO

Dimensão	Descrição
Fato_Atividade	Armazena as métricas base de esforço, estimadas e reais, de uma atividade.
Fato_Fase	Armazena as métricas base estimadas e reais de uma fase, assim como as métricas derivadas de variação referentes a estas métricas. Estas são: esforço e duração
Fato_Iteração	Armazena as métricas base estimadas e reais de uma iteração, assim como as métricas derivadas de variação referentes a estas métricas base. Estas são: esforço e duração
Fato_Defeito	Armazena os defeitos (internos e externos) de um produto, bem como o cálculo das métricas respectivas a eles.
Fato_Release	Armazena as métricas bases respectivas a tamanho, custo, esforço e duração, bem como as métricas derivadas referentes a essas métricas em uma release.

6.2 Caracterização do Problema

As organizações de *software* trabalham com diversos projetos de *software* que se diferenciam tanto pelas ferramentas de gestão utilizadas quanto pela forma que armazenam e controlam suas métricas de análise e acompanhamento.

Esse estudo de um ambiente real serviu para identificar limitações em uma organização de desenvolvimento de *software* no controle e acompanhamento de seus PDS por não possuírem informações integradas de seus processos. Ainda, permitiu constatar a criticidade na obtenção dessas informações dada a heterogeneidade das fontes do ambiente em questão. Diversos aspectos são descritos como segue:

- A extração das informações é efetuada de forma manual, um ator do PDS deve ser deslocado de suas atividades normais para efetuar a extração das informações a serem carregadas na BO.

- Em termos de ferramentas para controle da execução de atividades de projetos, são utilizadas desde planilhas eletrônicas (*MS Excel*) até ferramentas dedicadas a este fim (e.g. *MS Project Server, IBM Rational ClearQuest, Bugzilla*).
- Detectou-se ainda o uso de diferentes modelos de PDS, com ciclos de vida variados para projetos distintos, que se traduzem em formas bastante diversas de registrar os projetos, ainda que na mesma ferramenta.
- Outro problema é que cada uma destas ferramentas possui um modelo de dados próprio, que não segue paradigmas convencionais de representação de dados, dificultando assim a extração desses dados. Por conseqüência, o grau de complexidade do processo de ETC para esta organização é muito alto, ver Figura 19.

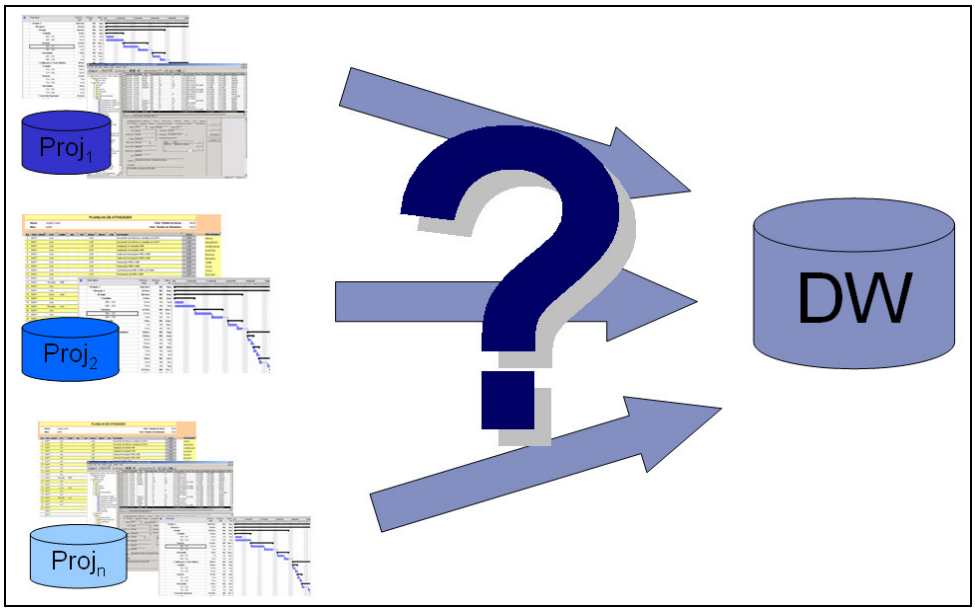

Figura 19 - Heterogeneidade da organização alvo

A seguir são discutidos três pontos principais observados. São eles: heterogeneidade, intrusão e extração não incremental.

6.2.1 Heterogeneidade

Os tipos de heterogeneidade encontrados na organização alvo foram: tipos de projetos, ciclos de vida, modelos de gerenciamento e tipos de ferramentas. Esta heterogeneidade é

ocasionada devido à especialização dos projetos em relação ao OSSP. Esses tipos de heterogeneidade encontrados na organização são descritos como segue.

6.2.1.1 Tipos de Projetos

A organização apresenta dois tipos de projeto: desenvolvimento e manutenção. Os projetos de desenvolvimento possuem seus ciclos de vida completos, onde todas as fases devem ser executadas. Já os projetos de manutenção têm seu ciclo de vida iniciado na fase de análise ou projeto.

6.2.1.2 Ciclos de Vida

Como a organização produz diversos projetos para uma variedade de clientes e usuários, um determinado ciclo de vida de *software* pode não ser apropriado para todas as situações. Dentre os ciclos de vida aprovados para utilização no OSSP estão: cascata, V, *staged delivery* e iterativo. Atualmente somente são utilizados os ciclos cascata e iterativo.

6.2.1.3 Modelos de Gerenciamento

O modelo de gerenciamento é a forma como os projetos gerenciam seus processos. A ferramenta utilizada para o controle da execução das atividades é *MS Project Server*. Atualmente são utilizadas duas formas de cronogramas para o controle das atividades de um projeto: cronogramas orientados a fase e cronogramas orientados a entregáveis.

Nos cronogramas orientados a fase tem-se que um *release* possui várias fases e estas várias atividades. Já nos cronogramas orientados a entregáveis, tem-se que um *release* possui vários entregáveis e estes várias fases, que por sua vez, possuem várias atividades.

6.2.1.4 Tipos de Ferramentas

Assim como os projetos possuem autonomia para escolherem seus tipos, ciclos e modelos de gerenciamento, eles também podem escolher suas ferramentas. A organização disponibiliza um conjunto de ferramentas com a finalidade de acelerar o processo de desenvolvimento e entrega de produtos, e diminuir o tempo de resolução de problemas e tomada de decisão. As ferramentas disponibilizadas dividem-se em gerenciadoras de cronogramas, acompanhamento de defeitos ou melhorias, métricas de dados de projetos e gerência de configuração.

- Gerenciadoras de cronograma: *MS Project Server, Gantt Project*.
- Acompanhamento de Defeitos ou Melhorias: *IBM Rational Clear Quest, Bugzilla*.
- Métricas de Projeto e Repositório de Dados do Projeto: *Organization Database*.
- Gerência de Configuração: *IBM Rational Clear Case, MS Visual Source Safe*.

6.2.2 Intrusão

Atualmente a organização possui um processo intrusivo de obtenção das métricas organizacionais, ou seja algum ator do PDS deve ser deslocado das suas atividades normais para executar todo o processo de obtenção das métricas e transformação destas métricas para o padrão da organização.

O processo de obtenção das métricas organizacionais consiste em realizar diversas consultas nas ferramentas de acompanhamento dos projetos para obter informações sobre esforço (nome do projeto; *release*; tipo da atividade: trabalho, retrabalho, revisão ou qualidade; fase da atividade e total de horas), defeitos (nome do projeto; *release*; tipo do defeito: defeito externo ou interno; fase de origem, *peer review*: se o defeito foi encontrado em PR ou em teste e severidade do defeito) e requisitos (nome do projeto; *release*; número de requisitos originais; número de requisitos adicionados; número de requisitos modificados e ; número de requisitos removidos). Essas informações são separadas em arquivos diferentes e encaminhadas para a pessoa responsável pela carga na BO.

6.2.3 Extração não incremental

A cada carga das métricas na BO, os dados são todos recarregados, mesmo após o projeto estar finalizado. Atualmente, não existe uma forma incremental de carga dos dados extraídos, ou seja, a cada nova carga os dados são todos novamente carregados, não havendo uma carga apenas daqueles dados que sofreram alguma alteração.

6.3 Comentários Adicionais

Este capítulo buscou apresentar o estudo de um ambiente real da organização alvo e a caracterização do problema a serem tratados nessa pesquisa, no Capítulo 7.

Não faz parte do escopo desse trabalho discutir a qualidade do modelo analítico utilizado pela organização alvo e sim, como este modelo deve ser alimentado. Este trabalho define como, a partir de fontes distintas, os dados necessários devem ser extraídos, transformados e carregados no modelo analítico, considerando ainda, diversos aspectos levantados neste capítulo como heterogeneidade, intrusão e extração incremental.

Constatou-se nesse estudo, que grande parte da heterogeneidade é proveniente da especialização dos projetos em relação ao processo organização padrão, conhecido como OSSP. É importante ressaltar também, que apesar do modelo analítico sofrer alteração, o programa de métricas organizacionais não foi modificado.

7 SOLUÇÃO PROPOSTA

Este capítulo apresenta a solução proposta, para o problema apresentado no capítulo anterior, que está inserida em uma arquitetura para o ambiente de Data Warehousing. Esta proposta abrange as camadas de Integração de Aplicações e Integração de Dados, onde o Processo de Extração, Transformação e Carga segue uma abordagem orientada a serviços.

A arquitetura desenvolvida para o ambiente de *Data Warehousing* é composta de três camadas: integração de aplicações, integração de dados e apresentação (ver Figura 20). A camada de integração de aplicações é responsável pela extração dos dados brutos dos projetos considerando as especificidades das ferramentas. A camada de integração de dados compreende a *Data Staging Area* (DSA) e o DW propriamente dito. O processo de ETC, que atua sobre as camadas de integração de aplicações e de integração de dados, segue uma abordagem orientada a serviços. A camada de apresentação e a modelagem do DW, apresentados em [NOV05, RUI05] e que não fazem parte do escopo deste trabalho, facilitam o acompanhamento segundo o programa de métricas definido pela organização.

Tendo por base a arquitetura apresentada na Figura 20, o processo de ETC consiste na captura de dados provenientes de múltiplas fontes de informação, na sua transformação em informações consistentes e de qualidade, e na inserção destes em um DW. Os dados devem ser extraídos segundo o modelo analítico alvo. A limpeza e a transformação dos dados extraídos têm por objetivo corrigir imperfeições contidas nas fontes de dados originais (e.g. ausência de dados, duplicações, homônimos, sinônimos), integrá-los e modificá-los para sua posterior carga no repositório de dados analítico. O objetivo é fornecer ao usuário do sistema analítico, informações (concisas, relevantes e de qualidade) que possibilitem tomadas de decisão de qualidade.

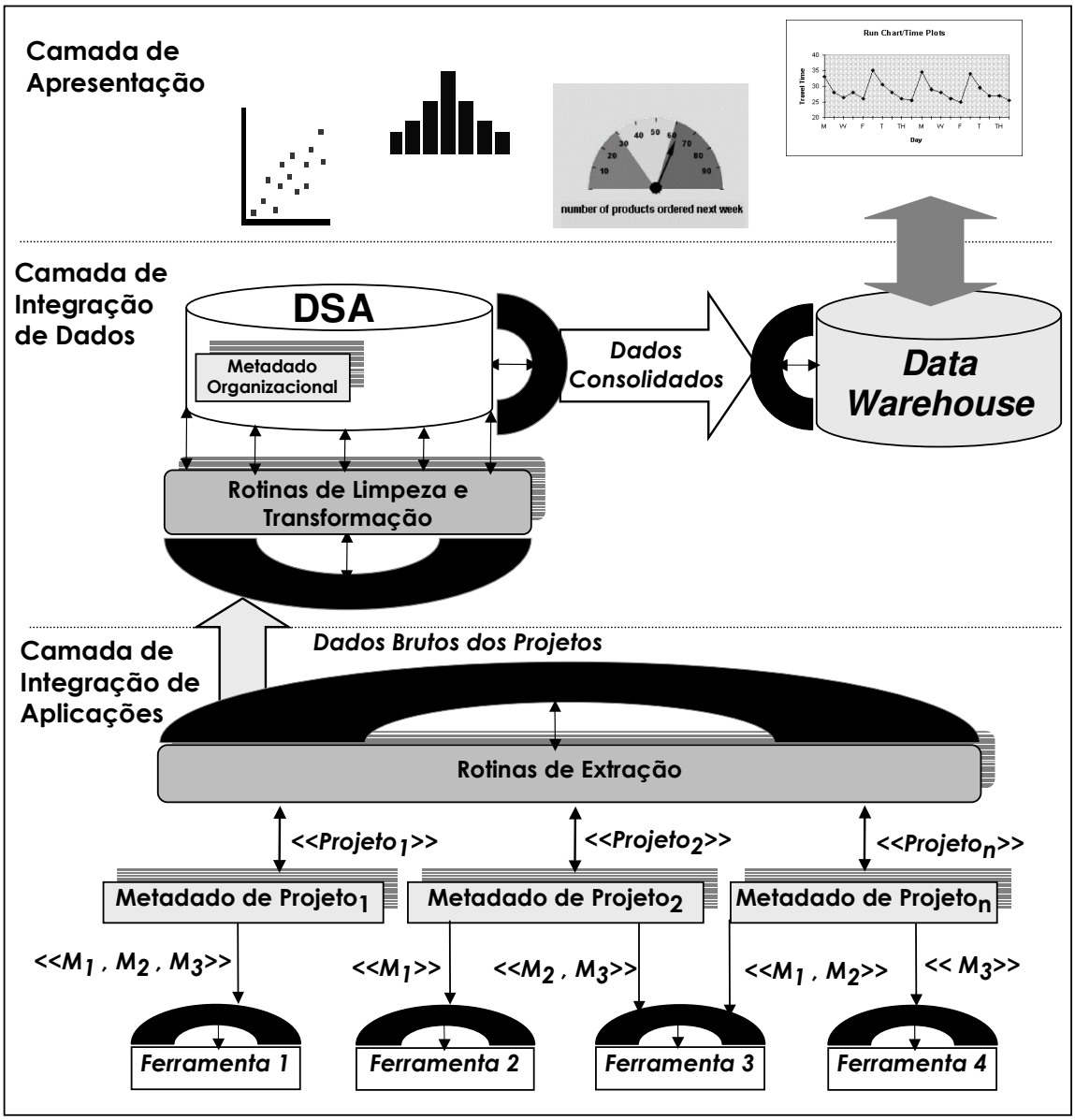

Figura 20 - Arquitetura de Data Warehousing

O processo de ETC, em PDS, visa capturar dados sobre a execução de processos concluídos ou em execução, e sobre artefatos manipulados e produzidos por cada processo. O repositório de dados analítico (DW) tem por objetivo estabelecer uma visão organizacional homogênea dos projetos, abstraindo suas especificidades. Portanto, é responsabilidade do ETC lidar com as especificidades dos projetos, que se traduzem, pelo ponto de vista de captura de dados, na necessidade de tratar as seguintes heterogeneidades: 1) diferentes ferramentas utilizadas para registrar dados dos projetos; 2) práticas diversas de registros de

dados ainda que em uma mesma ferramenta e 3) modelos de PDSs especializados em relação a OSSP.

A abordagem proposta neste trabalho tem as seguintes vantagens. Primeiro, permite reduzir a complexidade de implementação e manutenção dos procedimentos de extração. Segundo, os projetos podem evoluir com o tempo (adotar novas ferramentas ou mudar o modelo de gerenciamento). Terceiro, habilita lidar com a heterogeneidade utilizando os seguintes protocolos padrões: XML, SOAP, WSDL e UDDI. Esses protocolos são usados para definir o formato das mensagens, especificar a interface para onde estas são enviadas, descrever as convenções para o mapeamento do conteúdo das mensagens de entrada e saída responsáveis pelo serviço, e definir mecanismos para publicar e encontrar interfaces de serviços [ALO04].

Na implementação, foi utilizado o *framework Apache Axis*, que permite a construção de *Web Services* utilizando o protocolo SOAP. Considerando que os metadados estão descritos em XML *Schema* foi necessário utilizar alguma forma que fosse possível manipulá-los, o que se resume em interpretar, extrair dados e tratar eventos. Sendo assim, utiliza-se a *Simple* API *for* XML (SAX), onde através da linguagem de programação Java pode-se instanciar os *parsers* utilizando a Java API for XML *Parsing* (JAXP 1.1) implementada pelo *Axis*. O modelo SAX permite que esses *parsers* leiam documentos e respondam a eventos produzidos durante a leitura. Para implementação da DSA foi utilizado o *MS SQL Server*.

A seguir, são apresentadas, respectivamente, as camadas de integração de aplicações e integração de dados, descrevendo para tal, seus componentes principais e o relacionamento entre estes.

7.1 Camada de Integração de Aplicações

A camada de integração de aplicações é responsável pela extração de dados brutos dos projetos provenientes das diversas ferramentas e carregá-los na DSA. Para esta solução, adota-se uma abordagem de baixa intrusão seguindo um padrão arquitetural orientado a serviços, onde os serviços atuam como *wrappers*. Cada *wrapper* endereça a extração de dados considerando uma ferramenta em particular, e é focado no modelo de dados proprietário da mesma. Além disso, cada projeto 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 (métricas e atributos das dimensões) são

armazenados nessas ferramentas, de acordo com: o ciclo de vida do projeto (iterativo, cascata), tipo (desenvolvimento, manutenção) e modelo de gerenciamento (cronogramas orientados a fases, cronogramas orientados a entregáveis). As rotinas de extração exploram o metadado de projeto para localizar o *wrapper* correto e guiar a extração baseada no mapeamento estabelecido entre o dado bruto e o dado requerido. A seguir são descritos os três principais elementos da camada de integração de aplicações: os metadados de projeto, os *wrappers* e as rotinas de extração.

7.1.1 Metadados

Os metadados são definidos como dados sobre os dados. Para [KIM98], os metadados representam um conjunto de informações que descrevem o DW e atuam com um papel ativo na sua criação, uso e manutenção. Ainda, segundo o mesmo autor, uma das mais importantes características dos sistemas que suportam o processo de ETC é que eles possuem metadados dirigidos. Esses metadados podem assumir papéis ativos ou passivos, servindo apenas como uma documentação do processo de DW, quando passivos, e podem servir diretamente como um conjunto de instruções para esses processos quando é uma parte ativa do processo.

Segundo [INM96] os metadados podem ser caracterizados "como um diretório que auxilia os analistas a localizarem os componentes do DW". Ele ainda define quais informações os metadados mantêm, tais como:

- A estrutura dos dados segundo a visão do programador.
- A estrutura dos dados segundo a visão dos analistas.
- A fonte de dados que alimenta o DW.
- A transformação sofrida pelos dados no momento de sua migração para o DW.
- O modelo de dados do DW.
- O histórico das extrações de dados.
- Os dados referentes aos relatórios que são gerados pelas ferramentas OLAP, assim como os que são gerados nas camadas semânticas.

Considerando a necessidade dessas informações, dois tipos de metadados foram criados: os metadados de projeto e os metadados organizacionais. Os metadados de projeto referem-se à etapa de extração e, os organizacionais, às etapas de limpeza e transformação.

Como o processo de ETC segue uma abordagem orientada a serviços, optou-se neste trabalho por desenvolver os metadados em XML *Schema*[6]. A seguir, são apresentados os metadados de projeto. Já os metadados organizacionais são apresentados na Seção 7.2.2.

7.1.1.1 Metadados de projetos

Os metadados de projeto definem para cada projeto, o relacionamento entre o dado bruto e o dado requerido, considerando seu tipo, ciclo de vida, modelo de gerenciamento e ferramenta utilizada. A Figura 21 ilustra, através de um Modelo de *Features*, a heterogeneidade das fontes de dados. Analisando de forma *bottom-up* o modelo apresentado na Figura 21, verifica-se que em termos das ferramentas de gestão utilizadas no controle das atividades dos projetos são utilizadas desde planilhas eletrônicas (*MS Excel*) até ferramentas dedicadas a este fim (e.g. *MS Project Server, IBM Rational Clear Quest*). Ainda, essa organização possui distintos modelos de PDS (orientado a fases e orientado a entregáveis), os quais apresentam ciclos de vida variados (iterativo e cascata) e tipos de projetos distintos (manutenção e desenvolvimento), que resultam em diversas maneiras de registrar seus projetos, mesmo que na mesma ferramenta. Em hachurado na Figura 21 estão os projetos (Projetos 3, 5 e 6) utilizados na experimentação (ver Capítulo 8).

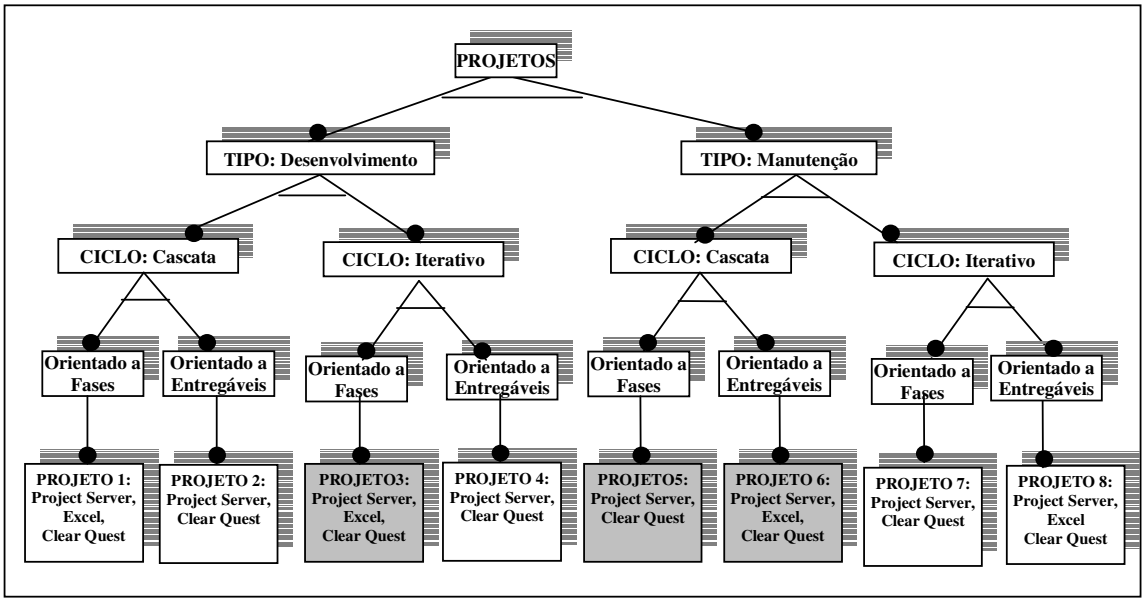

Figura 21 - Heterogeneidade dos Projetos

[6] Um documento em XML *Schema* é um documento XML utilizado para determinar que tipos de dados um documento XML está transportando.

A Figura 22, ilustra um metadado de projeto, onde o elemento inicial do XML *Schema* indica o nome do projeto e os posteriores indicam as ferramentas utilizadas pelo projeto. Para cada ferramenta tem-se a descrição de como os dados estão estruturados. Por exemplo, o projeto 3 do tipo desenvolvimento, com ciclo de vida do tipo iteração e com cronogramas orientados a fases possui a seguinte estruturação no *Project Server*: nome iteração, nome da fase, tipo das atividades e nomes das atividades. A diferenciação está na hierarquia apresentada no cronograma que é dependente das características do projeto. O relacionamento entre o dado bruto e o dado requerido, para cada informação a ser extraída, é especificado através dos elementos tabela, campo, tipo e seu tamanho na fonte de dado original. O Anexo I ilustra os metadados de projetos construídos.

```
<?xml version="1.0" encoding="UTF-8" ?>
   <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
      <xs:element name="Projeto3">
         <xs:element name="ProjectServer">
            <xs:element name="Nome_Projeto">
               <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
               <xs:element name="Campo" value="ProjectEnterpriseText3" />
               <xs:element name="Tipo" value="ntext" />
               <xs:element name="Tamanho" value="16" />
            </xs:element>
            <xs:element name="Versao"> . . . </xs:element>
            <xs:element name="Cliente"> . . . </xs:element>
            <xs:element name="Tamanho_Time"> . . . </xs:element>
            <xs:element name="Custo_Total"> . . . </xs:element>
            <xs:element name="Industria"> . . . </xs:element>
            <xs:element name="Tecnologia"> . . . </xs:element>
            <xs:element name="Tipo_Projeto"> . . . </xs:element>
            <xs:element name="Porte_Projeto"> . . . </xs:element>
            <xs:element name="Status"> . . . </xs:element>
            <xs:element name="Unidade_Tamanho"> . . . </xs:element>
            <xs:element name="Iteracoes">
            <xs:element name="NomeIteracao"> . . . </xs:element>
            <xs:element name="Fases">
               <xs:element name="NomeFase"> . . . </xs:element>
               <xs:element name="Tipos">
                  <xs:element name="Tipo_Atividade"> . . . </xs:element>
                     <xs:element name="Atividades">
                        <xs:element name="NomeAtividade"> . . . </xs:element>
                        <xs:element name="Custo_BO"> . . . </xs:element>
                        <xs:element name="Custo_BR"> . . . </xs:element>
                        <xs:element name="Esforco_BO"> . . . </xs:element>
                        <xs:element name="Esforco_BR"> . . . </xs:element>
                        <xs:element name="Data_Inicial_BO"> . . . </xs:element>
                        <xs:element name="Data_Inicial_BR"> . . . </xs:element>
                        <xs:element name="Data_Inicial_Real"> . . . </xs:element>
                        <xs:element name="Data_Final_BO"> . . . </xs:element>
                        <xs:element name="Data_Final_BR"> . . . </xs:element>
                        <xs:element name="Data_Final_Real"> . . . </xs:element>
                     </xs:element>
                  </xs:element>
               </xs:element>
            </xs:element>
         </xs:element>
         <xs:element name="Excel"> . . . </xs:element>
         <xs:element name="ClearQuest"> . . . </xs:element>
      </xs:element>
   </xs:schema>
```

Figura 22 - Metadado de Projeto

7.1.2 Wrappers

Wrappers são empacotadores que encapsulam uma ou mais aplicações provendo assim uma única interface [ALO05]. Como esta solução segue uma abordagem orientada a serviços, para cada fonte de dados existe um WSDL referente, onde os serviços atuam como *wrappers* que consideram a extração de uma ferramenta em particular, considerando suas especificidades. A Figura 23 ilustra o WSDL gerado a partir da classe de extração de dados do *MS Project Server* (*ProjectServer.java*). O arquivo XML (Figura 23) descreve o acesso ao documento que realiza o serviço solicitado. Ele possui uma série de parâmetros e definições para o acesso do solicitante e para a resposta esperada.

Nas primeiras linhas são definidas a versão do XML, qual o arquivo que será usado como base para gerar o WSDL e o local onde ele se encontra. Os *xmlns* informam de onde serão retirados os padrões de formatação para criar o WSDL e de qual implementação '*.jws*' serão retirados os dados. A *tag* <*schema*> define o tipo de resultado que será gerado no final da execução do serviço, ou seja, o que é retornado para quem realizou a requisição. Logo após, são criadas as *tags* <*message*>, de *request* e de *response*, que informam os métodos de chamada e retorno que compõe o serviço. Em cada função são integradas '*tags*' <*part name*>, nas quais constam os parâmetros de entrada que serão informados na chamada do serviço. Após a definição das funções, parâmetros e formatos, então são descritas as operações do serviço. A *tag* <*portType*> é composta inicialmente pelo nome do arquivo de execução do serviço, seguido pelas *tags* <*operation*> que definem o *input* e o *output* de cada método que o arquivo de serviço utiliza. Após as definições deve-se criar o processo de comunicação SOAP. Esse processo é iniciado com a *tag* <*binding*>, com a criação do esquema XML SOAP, definindo uma ação de comunicação de *input* e de *output* para cada operação que o serviço pode realizar. Para finalizar o processo ele gera, através da *tag* <*service*>, a definição do nome do serviço descrito no <*binding*> que será executado, e ainda, qual o local que ele se encontra.

```
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions targetNamespace="http://localhost:8080/axis/ETL/Extracao/ProjectServer/ProjectServer.jws"
    xmlns:apachesoap="http://xml.apache.org/xml-soap"
    xmlns:impl="http://localhost:8080/axis/ETL/Extracao/ProjectServer/ProjectServer.jws"
    xmlns:intf="http://localhost:8080/axis/ETL/Extracao/ProjectServer/ProjectServer.jws"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<wsdl:types>
<schema targetNamespace="http://localhost:8080/axis/ETL/Extracao/ProjectServer/ProjectServer.jws"
    xmlns="http://www.w3.org/2001/XMLSchema">
<import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
<complexType name="ArrayOf_soapenc_string">
<complexContent>
<restriction base="soapenc:Array">
<attribute ref="soapenc:arrayType" wsdl:arrayType="soapenc:string[]" />
    </restriction>
    </complexContent>
    </complexType>
    </schema>
    </wsdl:types>
<wsdl:message name="mainResponse" />
<wsdl:message name="mainRequest">
<wsdl:part name="args" type="impl:ArrayOf_soapenc_string" />
    </wsdl:message>
<wsdl:portType name="ProjectServer">
<wsdl:operation name="main" parameterOrder="args">
<wsdl:input message="impl:mainRequest" name="mainRequest" />
<wsdl:output message="impl:mainResponse" name="mainResponse" />
    </wsdl:operation>
    </wsdl:portType>
<wsdl:binding name="ProjectServerSoapBinding" type="impl:ProjectServer">
<wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
<wsdl:operation name="main">
<wsdlsoap:operation soapAction="" />
<wsdl:input name="mainRequest">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    namespace="http://DefaultNamespace" use="encoded" />
    </wsdl:input>
<wsdl:output name="mainResponse">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    namespace="http://localhost:8080/axis/ETL/Extracao/ProjectServer/ProjectServer.jws" use="encoded" />
    </wsdl:output>
    </wsdl:operation>
    </wsdl:binding>
<wsdl:service name="ProjectServerService">
<wsdl:port binding="impl:ProjectServerSoapBinding" name="ProjectServer">
<wsdlsoap:address location="http://localhost:8080/axis/ETL/Extracao/ProjectServer/ProjectServer.jws" />
    </wsdl:port>
    </wsdl:service>
    </wsdl:definitions>
```

Figura 23 - *Wrapper* do *Project Server* (WSDL)

7.1.3 Rotinas de extração

Quando as rotinas de extração necessitam buscar as informações nas fontes de dados, elas percorrem os metadados de projeto (apresentados na Figura 22), os quais fornecem informações sobre o mapeamento entre o dado requerido para extração e o dado bruto sendo extraído. Esses metadados parametrizam as rotinas de extração objetivando a localização do *wrapper* correto (ver Figura 24).

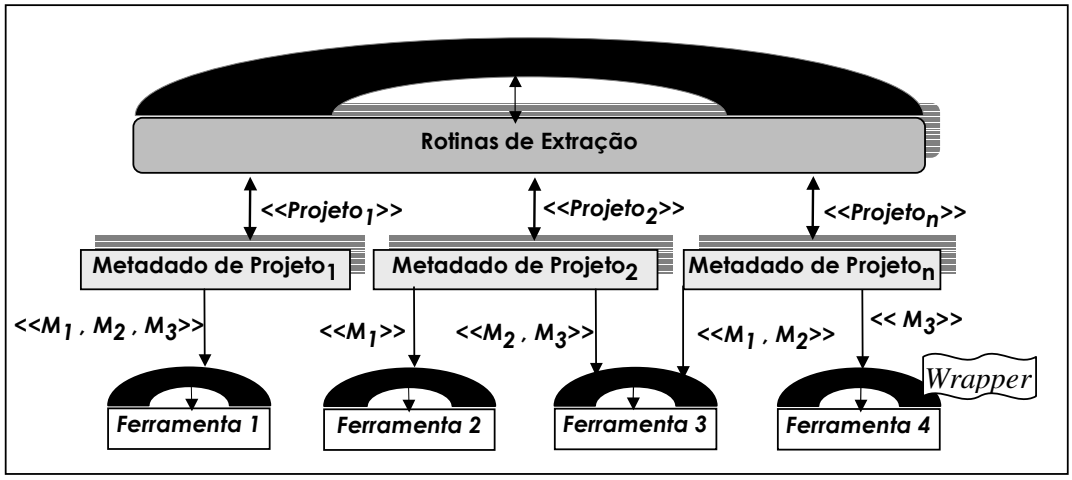

Figura 24 – Camada de Integração de Aplicações

Como citado anteriormente, nesta solução é utilizado o Apache *Axis* para a construção dos WSs. Ele trabalha internamente com SOAP, fornecendo todo o suporte para a comunicação baseada em XML. Para funcionar como um WS, os arquivos Java são transformados em arquivos com extensão '.jws'. Quando transformados em arquivos '.jws', o *Axis* gera WSDL, ou seja, arquivo XML que descreve o WS. Este arquivo Java é desenvolvido de forma a apresentar uma chamada para a classe *Service*, ou seja, criação de um serviço. Este serviço chama o método *createCall()*, que por sua vez, utiliza uma chamada para o método *setOperation()*. Este método possui como parâmetro o nome da função do WS que deverá ser executada. Após isso, é utilizada uma chamada ao método *invoke()* para, de fato, executar as funções definidas, via WSDL, que retornarão o resultado. Com os resultados recebidos pode-se realizar qualquer função ou operação como, por exemplo, efetuar a integração dos dados ou enviar as informações para a camada de integração de dados. Na Figura 25 é apresentado o trecho de código da classe *RotinaExtracao.java* que, conforme parametrização obtida através dos metadados de projeto, executa a chamada ao WS *ProjectServer.jws, ClearQuest.jws* ou *Excel.jws*. Onde o *wrapper* de um WS é responsável

pela extração de uma ferramenta em particular, considerando suas especificidades. Na Figura 25 visualiza-se mais facilmente a utilização dos métodos e chamadas descritos acima.

```
String urlWS = "http://localhost:8080/axis/ETL/Extracao/ProjectServer.jws";
Object [ ] params = ();
Service service = new Service ();
Call call = (Call) service.createCall ();
call.setTargetEndpointAddress (urlWS);
call.setOperationName ("BuscaDados");
ResultSet= (String []) call.invoke (params);
```

Figura 25 – Localização do *wrapper* do *Project Server*

Neste trabalho, o UDDI não foi utilizado, pois se sabe qual o endereço a ser acessado, ou seja, o local onde se podem encontrar os dados a serem extraídos (ferramentas).

Apesar das rotinas de limpeza e transformação serem responsáveis pela execução das técnicas de transformação, duas técnicas apresentadas por [KIM98], (ver Seção 5.1.2.2), são executadas nas rotinas de extração, são elas:

- **Integração**: dados provenientes de diversas fontes devem integrados para que possam ser enviados para a DSA. Por exemplo, quando uma atividade tem seus valores referentes ao *baseline* original e revisado proveniente do *MS Project Server* e seus valores referentes ao realizado proveniente do *MS Excel*, devem ser geradas novas chaves de registro, evitando o uso das chaves de registro originais dos sistemas legados, proporcionando uma integridade referencial entre as tabelas do modelo dimensional.
- **Combinação**: através dessa técnica é possível combinar atributos ou registros. Por exemplo, os campos customizados em nível de projeto no *MS Project Server* somente são obtidos pela combinação de outros dois atributos.

```
SELECT DISTINCT
        CAST(dbo.MSP_OUTLINE_CODES.OC_NAME AS VARCHAR(20)) AS Industria,
        ...
        FROM
                dbo.MSP_VIEW_PROJ_PROJECTS_ENT INNER JOIN
                dbo.MSP_OUTLINE_CODES ON
                dbo.MSP_VIEW_PROJ_PROJECTS_ENT.ProjectEnterpriseOutlineCode1ID
                = dbo.MSP_OUTLINE_CODES.CODE_UID
                …
```

7.2 Camada de Integração de Dados

Na camada de integração de dados (ver Figura 26), os dados extraídos das ferramentas do ambiente transacional são limpos e transformados no DSA, através das rotinas de limpeza e transformação, auxiliadas pelos metadados organizacionais. Após todo esse processo, os dados consolidados são carregados para o ambiente de DW. A seguir, são descritos os principais componentes da camada de integração de dados: as rotinas de limpeza e transformação, os metadados organizacionais, a DSA e a carga no DW.

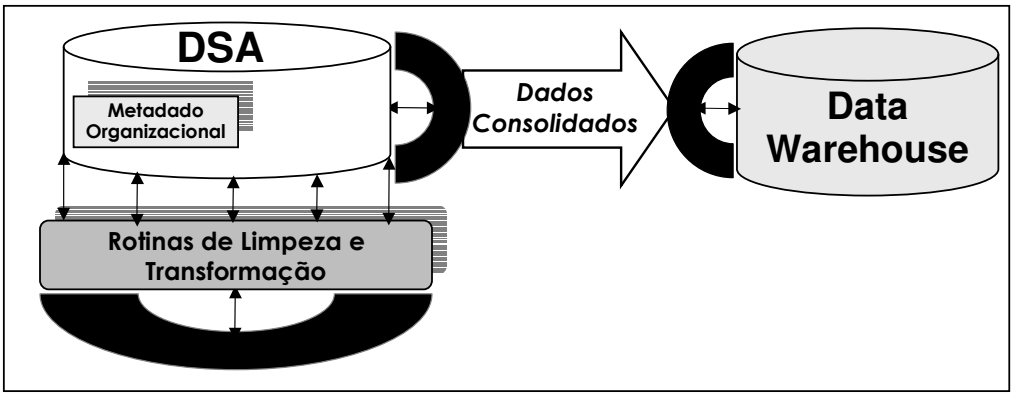

Figura 26 – Camada de Integração de Dados

7.2.1 Rotinas de Limpeza e Transformação

A execução das rotinas de limpeza e transformação sobre os dados brutos provenientes da camada de integração de aplicações tem por objetivo transformá-los em informações úteis, consistentes e de qualidade. [KIM98] apresenta 20 técnicas, das quais nessa etapa foram implementadas as seguintes[7]:

- **Limpeza:** como os dados brutos provenientes do sistema fonte muitas vezes são informados manualmente por seus usuários, estes podem apresentar diversos problemas. Esta técnica busca eliminar a utilização incorreta de caracteres, códigos e valores duplicados que, se não eliminados, causariam inconsistências no ambiente de DW.
- **Conversão de tipos de dados:** os dados brutos provenientes de um sistema fonte muitas vezes não possuem atributos com tipos semelhantes aos do

[7] Dentre as técnicas apresentadas por [KIM98], duas delas são executadas na camada de integração de aplicações: a integração dos dados provenientes das diversas fontes e a combinação de atributos ou registros.

modelo analítico. Por exemplo, `CAST(dbo.MSP_VIEW_PROJ_TASKS_STD.TaskName AS VARCHAR(20)) AS Nome_Fase`, onde o atributo Nome_Fase, armazenada no DW, deve ser do tipo "*varchar*", enquanto que nas fontes de origem pode ser tanto *ntext*, quanto *text*, dependendo da fonte de origem.

- **Valores nulos:** quando os sistemas fonte possuem dados muito antigos, podem existir dados com valores nulos. Por exemplo, a severidade dos defeitos em determinados projetos não possui classificação. Buscando não interferir nos valores produzidos pelos demais projetos, classificam-se valores nulos como "NA".
- **Verificação da integridade referencial:** esta última etapa consiste em verificar se um determinado atributo de uma tabela fato corresponde ao mesmo atributo armazenado na dimensão.

O processo de integração de dados pode ser obtido através do pré-processamento dos dados para padronização de nomes e valores, buscando resolver discrepâncias na representação de dados, fusão de valores em comum e valores equivalentes de dados provenientes de diversas fontes.

Após a execução das rotinas de limpeza e transformação, os dados já podem ser carregados para o DW. Apesar de [KIM98] afirmar que, para o DW ser auto-suficiente e a fonte única de informações, a DSA deva ser limpa. Neste trabalho, nós apenas excluímos da DSA as atividades completadas, isto se deve à necessidade de manter o registro do mapeamento entre os dados inseridos no DW e os dados sendo extraído das fontes de dados, para que em posteriores cargas, as informações armazenadas no DW sejam atualizadas corretamente.

O escopo dessa pesquisa busca encontrar padrões de transformação possibilitando afirmar que dados de um determinado tipo nas fontes de dados originais, e que no DW sejam de outro tipo, devam sofrer determinadas transformações. Essa definição tem por objetivo facilitar o processo de ETC no caso de novas métricas de análise serem inseridas. Por exemplo, dados do tipo *ntext* ou *text* nas fontes de dados originais e que no DW são do tipo *varchar*, devem sofrer a seguinte transformação: `CAST(METRICA_x AS VARCHAR (20))`. Outro exemplo de transformação, são dados do tipo *datetime* no sistema fonte e no DW sejam do tipo *date*, para isso é necessário a seguinte transformação

`CONVERT(VARCHAR(10), METRICA_y, 103)`, a qual converte a data atual para caracter no formato dd/mm/aaaa.

7.2.2 Metadados Organizacionais

Os metadados organizacionais (ver Figura 27) estabelecem um conjunto de regras de transformação da origem para o destino. Esses metadados são interpretados através de *parsers* que lêem cada metadado e produzem eventos durante a leitura. Por exemplo, o projeto 1 classifica a severidade de defeitos encontrados em testes como {1,2,3,4}. Já os defeitos encontrados em revisões são classificados como {A,B,C}. Esses metadados definem as regras de conversão dessas escalas específicas de projeto em escalas organizacionais, por exemplo, {Alta, Média, Baixa}. Outra transformação importante é em relação ao Tamanho_Total. Cada projeto tem a liberdade de escolher sua unidade de tamanho. Esta deve ser convertida para KLOC (*Kilo Line of Code*) e tal conversão depende da linguagem de programação utilizada. Por exemplo, se o Projeto 1 mede seu tamanho por PF (Ponto de Função) e utiliza a linguagem C++, têm-se 0,053 KLOC por PF. O Anexo II ilustra o Metadado Organizacional por completo.

```
<?xml version="1.0" encoding="UTF-8" ?>
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:element name="Organizacional">
                <xs:element name="Transformação">
                        <xs:element name="Projeto1">
                                <xs:element name="Severidade_Revisao">
                                        <xs:element name="A" value="Alta" />
                                        <xs:element name="M" value="Media" />
                                        <xs:element name="B" value="Baixa" />
                                </xs:element>
                                <xs:element name="Severidade_Teste">
                                        <xs:element name="1" value=" Baixa " />
                                        <xs:element name="2" value="Media" />
                                        <xs:element name="3" value="Media" />
                                        <xs:element name="4" value=" Alta " />
                                </xs:element>
                                <xs:element name="Tamanho_Total">
                                    <xs:element name="Linguagem">
                                            <xs:element name="C++" value="53" />
                                            <xs:element name="Cobol" value="107" />
                                            <xs:element name="Delphi_5" value="18" />
                                            <xs:element name="HTML_4" value="14" />
                                            <xs:element name="Visual_Basic_6" value="24" />
                                            <xs:element name="SQL" value="13" />
                                            <xs:element name="Java" value="46" />
                                    </xs:element>
                                </xs:element>
                            </xs:element>
                        <xs:element name="Projeto2"> . . .  </xs:element>
                        <xs:element name="Projeto3"> . . . </xs:element>
                </xs:element>
        </xs:element>
</xs:schema>
```

Figura 27 - Metadado Organizacional

7.2.3 Data Staging Area (DSA)

Buscando minimizar a intrusão do processo de ETC, utiliza-se uma área temporária[8] de dados, denominada DSA, que contém um conjunto de tabelas onde os dados brutos extraídos do sistema fonte devem ser armazenados para serem pré-processados através das rotinas de limpeza e transformação, considerando o modelo analítico alvo (descrito na Seção 6.1.4). O DSA foi implementado utilizando *MS SQL Server* e as rotinas de limpeza e transformação através da linguagem de programação Java. Essas rotinas baseiam-se nos metadados organizacionais que provêm uma visão unificada de todos os projetos pela perspectiva organizacional.

7.2.3.1 Estrutura do DSA

[KIM98] sugere uma série de passos para a construção de uma DSA. São eles: 1) Inicialmente, deve-se criar um plano de alto nível com o fluxo das informações da fonte para o destino. 2) Posteriormente, deve-se construir um plano detalhado para cada fluxo descrito no passo 1. A Figura 28 ilustra o plano de alto nível, que consiste em um esquema simples que ilustra as fontes e o destino. Segundo [KIM98], esse esquema de alto nível possibilita visualizar os três grandes passos de um DW: a obtenção dos dados das fontes, as trasnformações e a carga nas tabelas alvo. Com esse esquema deve ser possível visualizar:

- **Volume de dados de cada fonte:** cada projeto possui várias *releases* e cada uma dessas possui um cronograma no *MS Project Server* associado. O volume de dados de um cronograma fica em torno das 1000 atividades, o que não pode ser considerado um grande volume de dados.
- **Freqüência de atualização das fontes:** a freqüência de atualização varia de fonte para fonte; nas planilhas Excel verifica-se que a freqüência de atualização é diária, pois esta armazena as informações sobre as atividades realizadas. Sendo assim, a volatilidade dos dados disponíveis por esta ferramenta é alta.
- **Estimado x Realizado:** indica se as informações referem-se aos valores estimados ou realizados. O *MS Project Server* fornece tanto valores estimados

[8] O termo temporário está diretamente ligado ao tempo em que uma atividade de um projeto demora para ser completada, ou seja, enquanto ela não estiver completa ela será mantida na DSA.

quanto realizados, já as Planilhas Excel e o IBM *Rational Clear Quest* fornecem somente valores realizados.

- **Nível de transformação:** o nível de transformação pode ser alto, médio ou baixo. As informações provenientes das Planilhas Excel possuem um alto nível de transformação, já as provenientes do *IBM Rational Clear Quest* possuem um baixo nível.

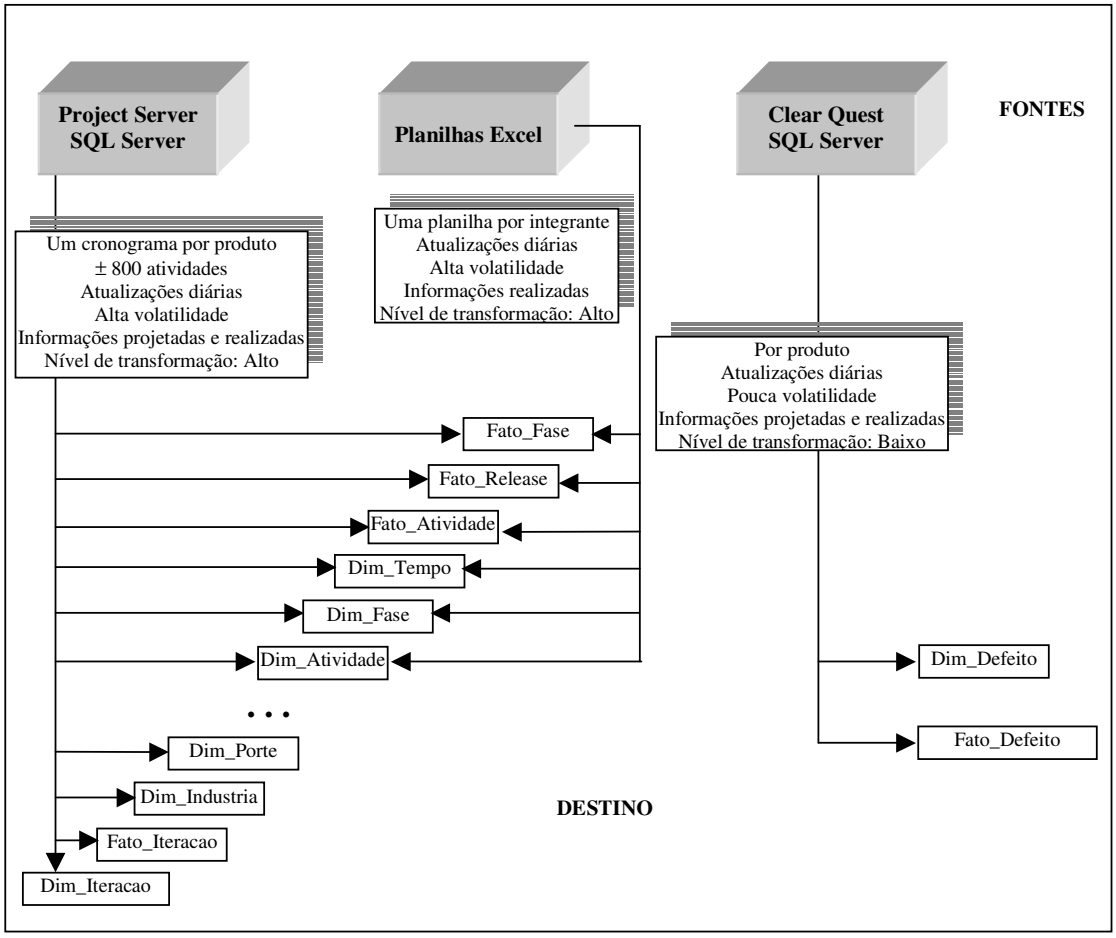

Figura 28 - Plano de Alto Nível

Segundo [KIM98] um plano detalhado deve, para cada fonte ilustrada no plano de alto nível (ver Figura 28), detalhar cada um dos seus fluxos. A Figura 29 apresenta o plano detalhado do *MS Project Server*, ilustrando para essa fonte de dados, quais as tabelas são trabalhadas, quais os dados são necessários, a ordem de coleta e quais as transformações são necessárias para cada conjunto de dados. Por exemplo, para as informações requeridas (Indústria, Tecnologia, Tipo, Porte e Unidade, ilustradas no segundo bloco da Figura 29) deve-se combinar dois atributos de duas tabelas diferentes para obter seu valor, e ainda deve-

se efetuar uma conversão de tipo de dados (*ntext* para *varchar*). Essas transformações são auxiliadas pelos metadados de projeto.

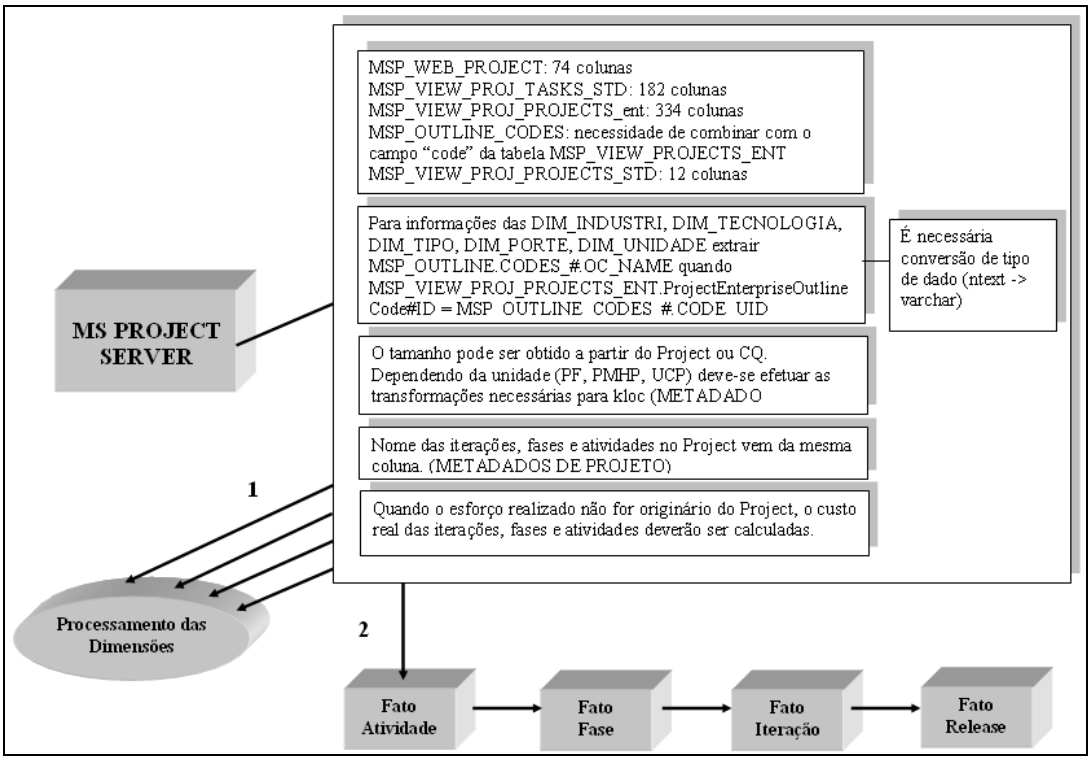

Figura 29 - Plano Detalhado (*MS Project Server*)

Ao final desses passos e conhecendo o modelo de dados alvo, consegue-se verificar quais tabelas são necessárias para a criação de uma DSA. A Figura 30 ilustra a DSA, a qual possui cinco tabelas que correspondem às cinco tabelas fato do DW: DSA_Atividade, DSA_Fase, DSA_Iteracao, DSA_Release, DSA_Defeito. Por exemplo, a tabela DSA_Atividade contém as informações correspondentes a tabela fato Fato_Atividade e a tabela dimensão Dim_Atividade do DW. Já tabela DSA_Fase possui as informações referentes as tabelas Fato_Fase e Dim_Fase. A tabela DSA_Iteração armazena as informações correspondentes as tabelas Fato_Iteracao e Dim_Iteracao. A tabela DSA_Defeito contém as informações referentes as tabelas Fato_Defeito e Dim_Defeito. Já a tabela DSA_Release armazena as informações referente a tabela fato Fato_Release, bem como as informações correspondentes às tabelas dimensão Dim_Cliente, Dim_Industria, Dim_Porte_Projeto, Dim_Projeto, Dim_Tecnologia, Dim_Tipo_Projeto e Dim_Unidade. O Anexo III apresenta os scripts para criação do DSA.

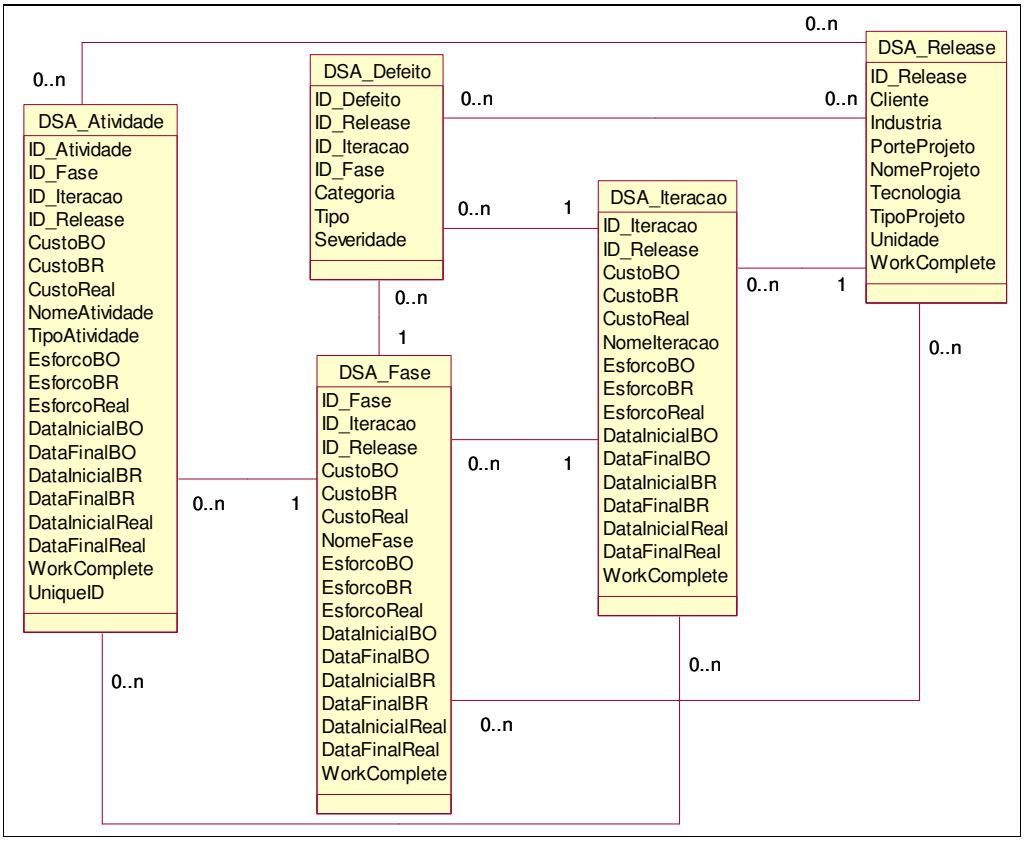

Figura 30 - ***Data Staging Area***

7.2.4 Carga no Data Warehouse

A última etapa do processo de ETC consiste em carregar os dados brutos extraídos do ambiente transacional, limpos e transformados no DSA através das rotinas de limpeza e transformação, no DW.

O objetivo do modelo analítico proposto para o DW é prover uma base de dados unificada e centralizada, que permita a análise dos dados resultantes, de todos os projetos de desenvolvimento da organização, através de métricas organizacionais. O modelo analítico comporta a execução de projetos organizados em fases e atividades, segundo os diferentes ciclos de vida, dos quais resultam um ou mais produtos de *software* (ver Seção 6.1.4).

A carga dos dados no DW dá-se inicialmente nas tabelas dimensão e posteriormente nas tabelas fato. Essa carga, envolve grande complexidade e considera diversos aspectos, tais como integridade referencial e atualização de informações no DW. Em DW, integridade referencial significa que para cada chave estrangeira da tabela fato existe uma entrada correspondente na tabela dimensão. Dessa forma, no momento da carga é necessário verificar

os campos das tabelas fato que são chaves estrangeiras para certificar-se de que os dados existentes nas tabelas fato estão de acordo com a chave primária da tabela dimensão corresponde.

O processo de carga deve conter regras para determinar como lidar com valores de atributos que sofreram alguma alteração em relação ao valor já armazenado no DW (ver Seção 5.1.3.2). Ainda, deve-se manter o registro no DSA do mapeamento entre as informações de atividades não completadas inseridas no DW e os dados brutos das fontes de origem. Esse mapeamento permite que os dados do DW sejam atualizados corretamente, evitando assim, que as métricas armazenadas nas tabelas fato sejam incrementadas de forma errônea.

Seguindo o padrão arquitetural orientado a serviços proposto neste trabalho, os dados armazenados no DSA devem ser encapsulados numa mensagem SOAP e enviados para o DW. Para que esses dados possam ser enviados do DSA para o DW através de mensagens SOAP, foi necessário especificar os serviços que receberão e processarão as mensagens, através do WSDL.

Ao final desta etapa, os dados armazenados no DW estarão prontos para serem analisados pelos gestores. Um dos pontos que ainda necessitam ser discutidos são o impacto das atualizações das informações no DW já que a solução proposta propõe ser evolutiva. Os fatores envolvidos na atualização de informações e os impactos dessas, são discutidos como segue.

7.2.4.1 Impacto das Atualizações das Informações

Diversos autores [KIM98, HAN01, IMN96] convergem em suas discussões sobre a complexidade de atualização das informações, ainda mais se esta for efetuada de forma evolutiva. Segundo [KIM98], quando os valores são atualizados, deve-se criar um novo registro na tabela fato ou dimensão. Ainda, o mesmo autor afirma que esta técnica é a mais utilizada e mais difícil de ser gerenciada. Apesar disso, esta é utilizada neste trabalho e consiste na inserção de uma nova linha na tabela e a criação de uma nova chave atualizada toda vez que ocorrer alguma mudança em um registro da tabela.

Estudos foram realizados sobre o impacto das atualizações, tanto nos atributos armazenados nas tabelas dimensões, quanto nas métricas armazenadas nas tabelas fato. Para os atributos das tabelas cria-se um novo registro na dimensão com o valor atualizado. Para as

métricas armazenadas nas tabelas fato também cria-se um novo registro na fato com o valor atualizado. A diferença está no tipo da métrica (direta ou indireta): se a métrica for direta, além da criação de um novo registro deve-se verificar o impacto dessa atualização nas métricas indiretas, ou seja, toda vez que uma métrica direta, que compõem uma métrica indireta for atualizada, esta deve ser recalculada. Além disso, como apresentado na Seção 6.1.4.1, verifica-se que algumas métricas indiretas são formadas a partir de outras métricas indiretas, o que resulta não somente do cálculo de métrica indireta final, mas também nas métricas indiretas que a compõem, o que ocasiona numa atualização de maior complexidade.

Outro ponto a ser discutido é como manter um relacionamento entre o dado armazenado no DW que deverá ser atualizado e o dado bruto que deverá ser extraído. Primeiramente, deve-se discutir a diferença entre atividades completadas e em andamento. Uma atividade é considerada completada quando o campo MSP_VIEW_PROJ_TASKS_STD.WorkComplete for igual 100, ou seja, o percentual de esforço completo dessa atividade no *MS Project Server* é 100%. Caso contrário, essa atividade é considerada em andamento. Enquanto a atividade não for completada ela deverá ser atualizada no DW. Para evitar que atividades sejam atualizadas erroneamente deve-se manter um relacionamento entre esta informação e o dado bruto sendo extraído. A solução para isso foi manter as atividades que não possuem o campo "DSA_ATIVIDADE.WorkComplete = 100" na DSA. Para cada atividade armazenada na DSA deve-se carregar a sua identificação única (DSA_ATIVIDADE.UniqueID), a qual mantêm o ID inicial de cada atividade. Essa informação permite que se mantenha o relacionamento entre o dado carregado no DW, após a integração das informações e mudanças dos registros, e o dado bruto sendo extraído.

7.3 Considerações Adicionais

Este capítulo apresentou a solução proposta para o problema apresentado no capítulo 6 utilizando os conceitos apresentados nos capítulos 2, 3, 4 e 5. Essa solução está inserida em uma arquitetura de *Data Warehousing* e abrange as camadas de integração de aplicações e de integração de dados.

Na camada de integração de aplicações apresentou-se os componentes principais desta: (1) os metadados de projeto que descrevem o mapeamento entre o dado requerido para extração e o dado bruto sendo extraído. (2) os *wrappers* endereçam a extração de dados considerando as especificidades das ferramentas. Esta solução segue uma abordagem

orientada a serviços, onde os serviços atuam como *wrappers*. (3) já as rotinas de extração exploram o metadado de projeto para localizar o *wrapper* correto e conduzir a extração baseada no mapeamento entre o dado bruto e o dado requerido.

Na camada de integração de dados, apresentou-se como os dados brutos dos projetos provenientes da camada de integração de aplicações são limpos e transformados e carregados na DAS. Ainda descreveu-se como os metadados organizacionais auxiliam essas rotinas e, ao final, discorreu-se sobre o processo de carga e atualização dados, bem como o impacto dessas atualizações.

O capítulo seguinte propõe através de experimentações demonstrar que a solução proposta neste trabalho é evolutiva, de baixa intrusão e trata a heterogeneidade adequadamente, considerando assim, as especificidades dos projetos.

8 EXPERIMENTAÇÃO DA SOLUÇÃO

Este capítulo descreve a experimentação da solução proposta e, para tal, utilizam-se três exemplos. Esses exemplos buscam mostrar que a solução proposta é evolutiva, de baixa intrusão e que trata a heterogeneidade das fontes. Para isso, são descritos os elementos do experimento: o estabelecimento dos objetivos, a formulação das questões, a elaboração das métricas, e o relato do experimento. Ao final, apresenta-se uma análise e interpretação dos resultados.

Segundo [TRA02], a experimentação é o "centro do processo científico" e tem seus objetivos principais relacionados à execução de experimentos em Engenharia de *Software*: caracterização, avaliação, previsão, controle e melhoria a respeito de produtos, processos, recursos, modelos e teorias, entre outros. Ainda, segundo o mesmo autor, a experimentação pode ajudar a construir uma base de conhecimento confiável e reduzir assim a incerteza sobre quais teorias, ferramentas, e metodologias são adequadas.

Os elementos principais de um experimento são as variáveis, os objetos, os participantes, o contexto do experimento, hipóteses e o tipo de projeto do experimento. Esses conceitos são descritos como segue.

- **Variáveis**: as variáveis podem ser de dois tipos, independentes e dependentes. As variáveis independentes, também denominadas fatores, apresentam a causa que afeta o resultado do processo de experimentação (tratamento). Já as dependentes, referem-se à saída do experimento, ou seja, ao efeito causado pelos fatores do experimento (resultado).
- **Objetos**: os objetos são utilizados para verificar o relacionamento causa-efeito em uma teoria. Durante a execução do experimento os tratamentos são aplicados ao conjunto dos objetos e assim o resultado é avaliado.

- **Participantes**: os participantes são os indivíduos selecionados para conduzir o experimento.
- **Contexto do experimento**: o contexto do experimento contém as condições em que o experimento está sendo executado, por exemplo, in-vitro vs. in-vivo, alunos vs. profissionais, problema de sala de aula vs. problema real e específico vs. geral.
- **Hipóteses**: normalmente um experimento é formulado de hipóteses., o qual é formado por uma hipótese nula e tem por objetivo rejeitar a hipótese nula a favor de uma ou algumas hipóteses alternativas.
- **Tipo de projeto do experimento:** o projeto do experimento determina a maneira como um experimento será conduzido, ou seja, é o momento no qual é realizada a alocação dos objetos e dos participantes, bem como a quantidade e a seqüência dos testes experimentais.

A Figura 31 ilustra os relacionamentos entre os elementos principais de um experimento, descritos acima.

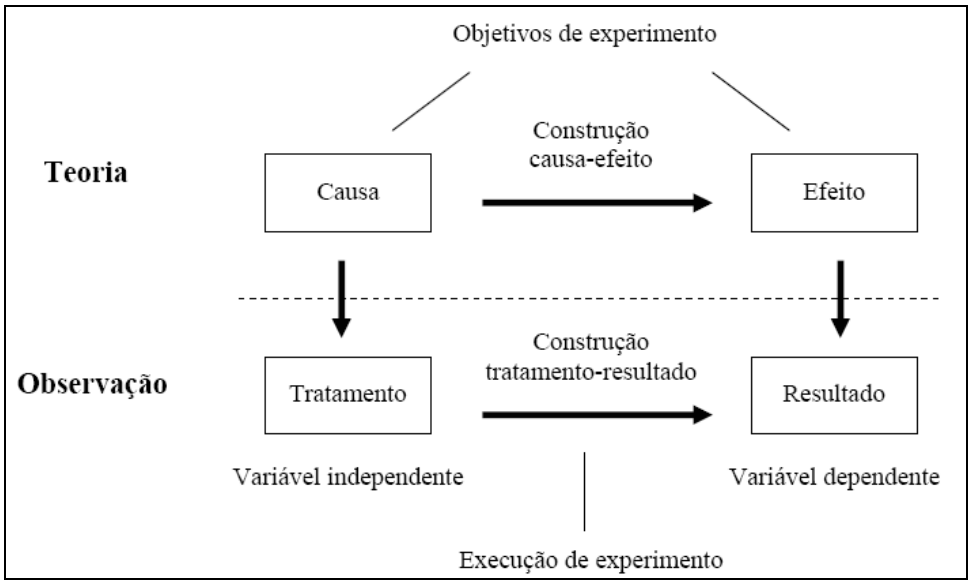

Figura 31 - Conceitos de um experimento, extraída de [TRA02]

[TRA02] discorre sobre duas metodologias de experimentação: (1) QIP (*Quality Improvement Paradigm*) e a (2) GQM (*Goal/Qestions/Metric*). A primeira metodologia, QIP, define seis passos que ao final resultam em um ciclo de melhoria do processo por completo.

Já a segunda metodologia, GQM, utiliza para definição uma abordagem *top-down* para o estabelecimento dos objetivos, a formulação das questões e a elaboração das métricas. Já para a interpretação usa uma abordagem *bottop-down*: a medição para receber os dados experimentais, a formulação das respostas para as questões baseadas nos dados experimentais, e o agrupamento das respostas para demonstrar o grau de sucesso dos objetivos estabelecidos. Todos esses passos, que fazem parte do processo principal da abordagem GQM, são ilustrados na Figura 32.

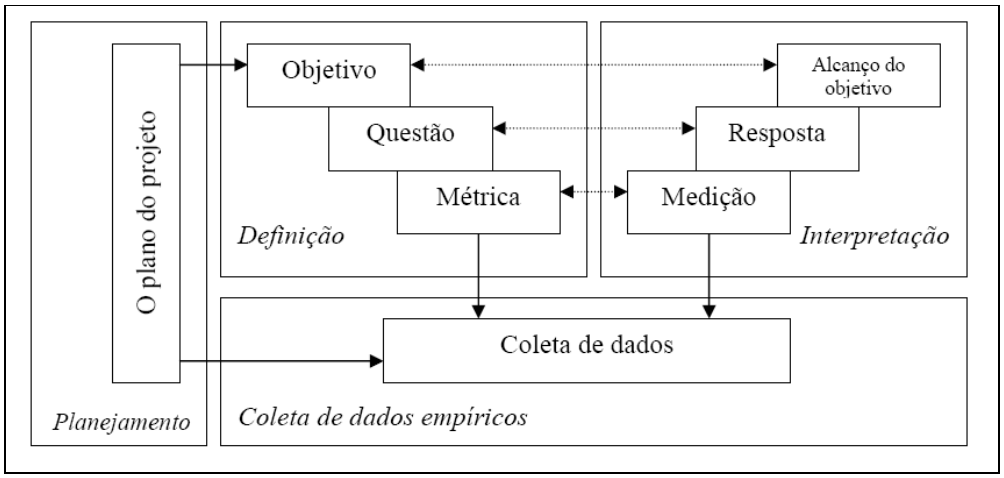

Figura 32 - O processo principal da abordagem GQM, extraída de [TRA02]

A abordagem GQM é composta de quatro fases: (1) a fase de *Planejamento*, quando o projeto da medição está selecionado, definido, caracterizado e planejado, resultando no plano de projeto; (2) a fase de *Definição*, quando o programa de medição é conceitualmente preparado, ou seja, os objetivos, as questões, as métricas e as hipóteses são estabelecidos; (3) a fase de *Coleta de dados empíricos*, quando a coleta de dados experimentais é efetivamente feita resultando em um conjunto de dados prontos para a interpretação; e (4) a fase de *Interpretação*, quando os dados são processados a respeito das métricas, questões e objetivos definidos.

A metodologia de experimentação utilizada neste trabalho foi a GQM, isso deve-se ao fato de que esta permite que os resultados obtidos durante o experimento sejam utilizados na melhoria da solução proposta. A seguir, são descritos os passos seguidos na experimentação: inicialmente discorre-se sobre a fase de definição dos dados, através da definição dos objetivos, as questões e as métricas e posteriormente sobre o relato do experimento.

8.1 Definição dos Objetivos

- **Objetivo do Estudo:** Analisar a solução proposta para verificar se ela atende aos objetivos específicos propostos incialmente (ver Seção 1.3).
- **Objetivo do Experimento:** A realização do experimento tem o propósito de avaliar a solução proposta de uma abordagem orientada a serviços para captura de métricas de PDS, buscando caracterizá-la como evolutiva, de baixa intrusão e adequado tratamento da heterogeneidade.

8.2 Questões

Questão 1 (Q1): A solução proposta para uma abordagem orientada a serviços para captura de métricas de PDS é evolutiva?

Métrica: Somente são carregados no DW os dados que sofreram alguma modificação, sendo o fator de atualização inferior a 100%. Quando não for a primeira carga.

Questão 2 (Q2): A solução proposta para uma abordagem orientada a serviços para captura de métricas de PDS é de baixa intrusão?

Métrica: Não é preciso deslocar nenhum ator do PDS para realizar as atividades de extração, transformação e carga, ou seja, quando o indicador de número de atores deslocados for igual a zero.

Questão 3 (Q3): A solução proposta para uma abordagem orientada a serviços para captura de métricas de PDS trata adequadamente a heterogeneidade?

Métrica: Tratamento das especificidades dos PDS, considerando diferentes tipos de projetos, diferentes ciclos de vida, diferentes modelos de gerenciamento e diferentes ferramentas utilizadas, ou seja, quando o indicador de tratamento adequando da heterogeneidade for igual a 4/9, 5/9 ou 6/9[9]. Quando esse fator for 1/9, 2/9 ou 3/9 o tratamento é considerado parcial. Através da Figura 33 verifica-se que esse fator seja igual a 4/9 a solução proposta deve considerar para cada projeto seu tipo, ciclo,

[9] Nove corresponde ao conjunto total de elementos: Ferramentas (*MS Project Server, IBM Rational Clear Quest, MS Excel*), Modelo de Gerenciamento (Orientado a Fases e Orientado a Entregáveis), Ciclos de Vida (Cascata e Iterativo) e Tipos de Projeto (Desenvolvimento e Manutenção).

modelo de gerenciamento e uma ferramenta. Para que este fator seja igual a 5/9 ele devera considerar duas ferramentas e igual a 6/9 considerar três ferramentas.

8.3 Relato do Experimento

8.3.1 Bancada de Teste

Com o propósito de validar a implementação que está sendo desenvolvida, foram construídos três exemplos de teste conforme a heterogeneidade apresentada na Figura 33. Esses três projetos cobrem todo o espectro de aspectos a serem testados. O modelo apresentado na Figura 33 mostra que quando um determinado projeto é do tipo desenvolvimento, ele não pode ser também do tipo manutenção, ou seja, a partir do topo do modelo deve-se percorrer os caminhos permitidos pelo modelo. Por exemplo, um projeto pode ser do tipo desenvolvimento, com ciclo iterativo, modelo orientado a fases e utiliza as ferramentas *MS Project Server, MS Excel* e *IBM Rational Clear Quest* (Projeto 1). A seguir, apresenta-se uma descrição dos três projetos utilizados como exemplo, salientando suas diferenças e semelhanças.

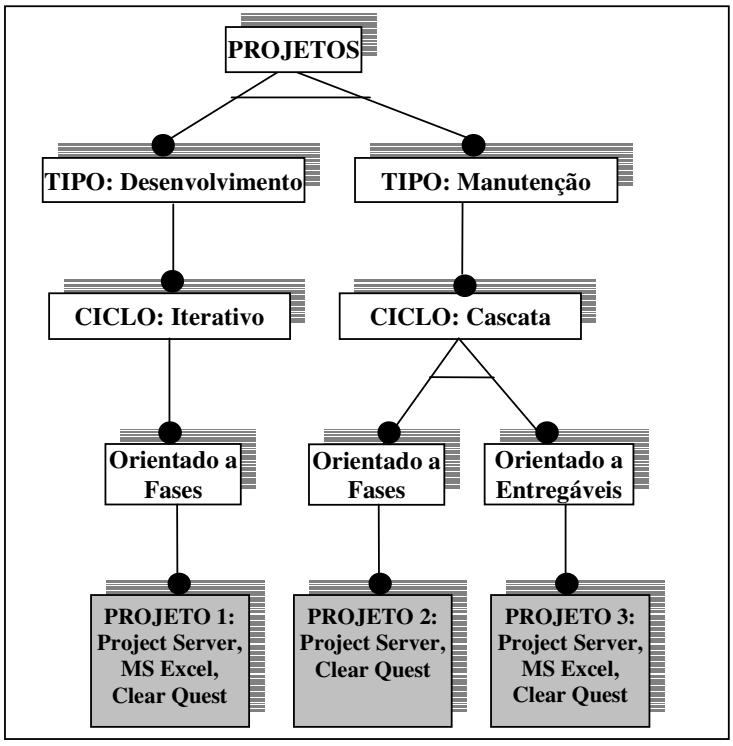

Figura 33 - Características dos exemplos desenvolvidos para experimentação

8.3.1.1 Projeto 1

O projeto 1 é do tipo desenvolvimento, com ciclo iterativo e modelo de gerenciamento orientado a fases, e utiliza três ferramentas: *MS Project Server*, *MS Excel* e IBM *Rational Clear Quest*. Para o gerenciamento dos cronogramas desse projeto é utilizado o *MS Project Server* que fornece os valores estimados nas *baselines* para as atividades, bem como o tamanho (estimado e realizado) de cada *release*. Para o registro dos valores das atividades realizadas é utilizado o *MS Excel*. As informações referentes a defeitos são armazenadas no IBM *Rational Clear Quest*. A diferenciação desse projeto para os demais está na hierarquia apresentada no cronograma (*MS Project Server*) que é dependente das características do projeto e obedece a hierarquia apresentada na Figura 34. Observa-se nessa figura, que existem cinco níveis de informação a serem coletadas: o nome do projeto, o nome da iteração, o nome da fase, o tipo e o nome da atividade. O nome da iteração deve-se ao fato do projeto 1 possuir um ciclo iterativo. Já o nome da atividade ser pertencente ao seu tipo deve-se ao fato do cronograma ser orientado a fases.

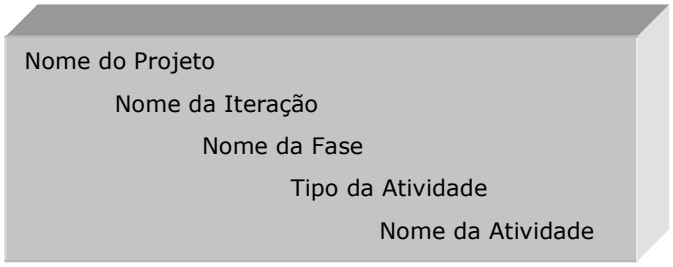

Figura 34 - Hierarquia do Projeto 1

8.3.1.2 Projeto 2

O projeto 2 é do tipo manutenção, com ciclo cascata e modelo de gerenciamento orientado a fases e utiliza duas ferramentas: *MS Project Server* e IBM *Rational Clear Quest*. Nesse projeto para gerenciamento de seus cronogramas utiliza-se o *MS Project Server*, tanto para registro dos valores estimados nas *baselines* para as atividades, como para os valores das atividades realizadas. Ainda são armazenados os tamanhos de cada *release*, tanto valores estimados nas *baselines* quanto realizados. Já no IBM *Rational Clear Quest* são armazenadas as informações referentes a defeitos. A diferenciação desse projeto para os demais está na hierarquia apresentada no cronograma (*MS Project Server*) que é dependente das características do projeto e obedece a hierarquia mostrada na Figura 35. Analisando essa figura, verifica-se que existem quatro níveis de informação a serem coletadas: o nome do

projeto, o nome da fase, o tipo e o nome da atividade. O motivo pelo qual o nome da atividade pertence ao seu tipo é devido ao fato do cronograma ser orientado a fases.

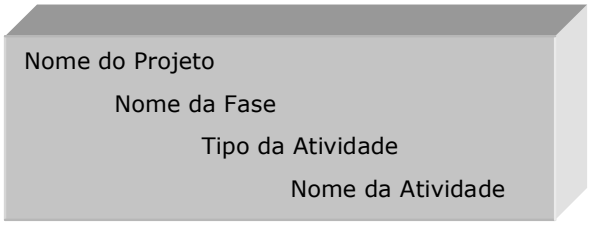

Figura 35 – Hierarquia do Projeto 2

8.3.1.3 Projeto 3

O projeto 3 é do tipo manutenção, com ciclo cascata e modelo de gerenciamento orientado a entregáveis e utiliza três ferramentas: *MS Project Server*, *MS Excel* e IBM *Rational Clear Quest*. Para o gerenciamento dos cronogramas desse projeto é utilizado o *MS Project Server* que armazena os valores estimados nas *baselines* para as atividades. No registro dos valores referente às atividades realizadas é utilizado o *MS Excel*. Para armazenar as informações referentes a defeitos e tamanhos (valores estimados nas *baselines* e realizados) é utilizado o IBM *Rational Clear Quest*. A diferenciação desse projeto para os demais está na hierarquia apresentada no cronograma (*MS Project Server*) que é dependente das características do projeto e obedece hierarquia ilustrada na Figura 36. Observando essa figura, verifica-se que existem quatro níveis de informação a serem coletadas: o nome do projeto, o nome da fase, o nome e o tipo da atividade. O motivo pelo qual o tipo da atividade pertence ao seu nome é devido ao fato do cronograma ser orientado a entregáveis.

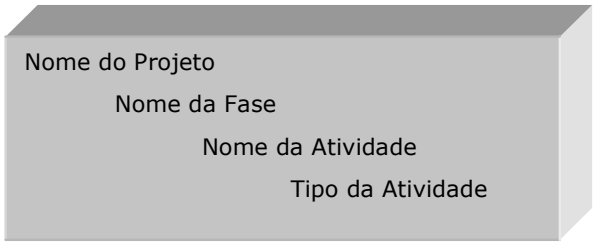

Figura 36 - Hierarquia do Projeto 3

8.3.2 Descrição da Instrumentalização

A Tabela 7 ilustra os aspectos considerados na experimentação, determinando quando a solução proposta pode ser considerada, ou não, evolutiva, de baixa intrusão e que efetua o tratamento adequado da heterogeneidade.

TABELA 7 - ASPECTOS CONSIDERADOS NA EXPERIMENTAÇÃO

Evolutiva (E)	Baixa Intrusão (I)	Tratamento da heterogeneidade (H)
1. Não é evolutiva, quando recarrega todas as informações a cada atualização do DW. Ou seja, caso já tenha uma população de fatos no DW, a renovação desses fatos deve ser igual a 100%. 2. É evolutiva, quando mantêm os valores armazenados no DW e somente carrega os valores que sofreram alguma alteração. Ou seja, caso já tenha uma população de fatos no DW, a renovação desses fatos deve ser inferior a 100%.	1. Não é de baixa intrusão, quando é necessário deslocar um ou mais atores do PDS para executar o processo de ETC, ou seja, quando o indicador de número de atores deslocados for maior que zero. 2. É de baixa intrusão, quando o processo de ETC é executado automaticamente, não necessitando que se desloque um ator do PDS, ou seja, quando o indicador de número de atores deslocados for igual a zero.	1. Não é heterogênea quando não considera nenhuma das especificidades dos PDS, ou seja, quando o indicador de tratamento adequado da heterogeneidade for igual a zero. 2. É heterogênea quando considera as especificidades dos PDS. Ou seja, quando o indicador de tratamento adequando da heterogeneidade for igual ou superior 4/9. 3. Quando esse fator estiver entre 1/9 e 3/9 o tratamento da heterogeneidade é considerado parcial.

Ainda, a Tabela 8 relaciona as questões (Seção 8.2) à combinação de aspectos considerados na experimentação. O resultado desse relacionamento foram oito combinações de aspectos. Dentre esses, o oitavo aspecto (em negrito) que envolve as métricas Q1, Q2 e Q3 é o que demonstra que a solução proposta atinge os objetivos esperados, sendo evolutiva, de baixa intrusão e que trata adequadamente a heterogeneidade. Para cada aspecto aplicam-se testes para definir:

- Se a solução pode ser considerada evolutiva.
- Se a solução pode ser considerada de baixa intrusão.

- Se a solução trata adequadamente a heterogeneidade.

Resultado: N aspectos com valores (E; I; H), onde E – evolutiva {0 – não é evolutiva, 1 – é evolutiva}; I – baixa intrusão {0 – não é de baixa intrusão, 1 – é de baixa intrusão}; H – tratamento adequado da heterogeneidade {0 – não heterogênea, 1 - heterogênea}.

TABELA 8 - MÉTRICAS E QUESTÕES ESTABELECIDAS PARA CADA ASPECTO

Nº	E	I	H	Descrição dos Aspectos	Questões
1	0	0	0	Não é evolutiva, não é de baixa intrusiva, não é heterogênea	N/A
2	0	0	1	Não é evolutiva, não é de baixa intrusiva, é heterogênea	Q3
3	0	1	0	Não é evolutiva, é de baixa intrusiva, não é heterogênea	Q2
4	0	1	1	Não é evolutiva, é de baixa intrusiva, é heterogênea	Q2, Q3
5	1	0	0	É evolutiva, não é de baixa intrusiva, não é heterogênea	Q1
6	1	0	1	É evolutiva, não é de baixa intrusiva, é heterogênea	Q1, Q3
7	1	1	0	É evolutiva, é de baixa intrusiva, não é heterogênea	Q1, Q2
8	**1**	**1**	**1**	**É evolutiva, é de baixa intrusiva, é heterogênea**	**Q1, Q2, Q3**

8.4 Análise e Interpretação dos Resultados

Os resultados a serem analisados são baseados em sete cargas executadas na seqüência. Para cada uma dessas cargas foram analisadas as questões a serem respondidas, bem como suas métricas. A seguir, discorre-se sobre cada uma dessas cargas buscando analisar e interpretar os resultados encontrados.

1. Carga total, onde as informações referentes aos três projetos foram extraídas, transformadas e carregadas pela primeira vez no DW.

TABELA 9 - PRIMEIRA CARGA

	M1	M2	M3
Projeto 1	Não pode ser avaliada, pois é a primeira carga a ser efetuada, logo M1 = NA.	Utiliza procedimentos automatizados para a extração, transformação e carga, logo, M2 = 0.	Considerou o tipo, ciclo, modelo de gerenciamento e três ferramentas distintas, logo, M3 = 6/9.
Projeto 2			Considerou o tipo, ciclo, modelo de gerenciamento e três ferramentas distintas, logo, M3 = 5/9.
Projeto 3			Considerou o tipo, ciclo, modelo de gerenciamento e três ferramentas distintas, logo, M3 = 6/9.

2. Carga parcial, somente incremento de valores das atividades sem a finalização das atividades.

TABELA 10 - SEGUNDA CARGA

	M1	M2	M3
Projeto 1	Possui 147 atividades carregadas no DW, das quais 15 sofreram atualizações, logo M1 = 10,2%	Utiliza procedimentos automatizados para a extração, transformação e carga, logo, M2 = 0.	Considerou o tipo, ciclo, modelo de gerenciamento e três ferramentas distintas, logo, M3 = 6/9.
Projeto 2	Possui 101 atividades carregadas no DW, das quais 23 sofreram atualizações, logo M1 =22,77%		Considerou o tipo, ciclo, modelo de gerenciamento e duas ferramentas distintas, logo, M3 = 5/9.
Projeto 3	Possui 183 atividades carregadas no DW, das quais 42 sofreram atualizações, logo M1 = 22,95 %		Considerou o tipo, ciclo, modelo de gerenciamento e três ferramentas distintas, logo, M3 = 6/9.

3. Carga parcial, com a inclusão de novas iterações e fases nos cronogramas, sem alterar os indicadores das atividades já carregadas.

TABELA 11 - TERCEIRA CARGA

	M1	M2	M3
Projeto 1	Possui 187 atividades carregadas no DW, das quais 30 foram incluídas, logo M1 = 16,04%	Utiliza procedimentos automatizados para a extração, transformação e carga, logo, M2 = 0.	Considerou o tipo, ciclo, modelo de gerenciamento e três ferramentas distintas, logo, M3 = 6/9.
Projeto 2	Possui 126 atividades carregadas no DW, das quais 25 foram incluídas, logo M1 =19,84%		Considerou o tipo, ciclo, modelo de gerenciamento e duas ferramentas distintas, logo, M3 = 5/9.
Projeto 3	Possui 222 atividades carregadas no DW, das quais 39 foram incluídas, logo M1 = 17,57 %		Considerou o tipo, ciclo, modelo de gerenciamento e três ferramentas distintas, logo, M3 = 6/9.

4. Carga parcial, somente incremento de valores das atividades com a finalização de algumas atividades.

TABELA 12 - QUARTA CARGA

	M1	M2	M3
Projeto 1	Possui 187 atividades carregadas no DW, das quais 49 sofreram atualizações, logo M1 = 26,2%	Utiliza procedimentos automatizados para a extração, transformação e carga, logo, M2 = 0.	Considerou o tipo, ciclo, modelo de gerenciamento e três ferramentas distintas, logo, M3 = 6/9.
Projeto 2	Possui 126 atividades carregadas no DW, das quais 72 sofreram atualizações, logo M1 = 57,14%		Considerou o tipo, ciclo, modelo de gerenciamento e duas ferramentas distintas, logo, M3 = 5/9.
Projeto 3	Possui 183 atividades carregadas no DW, das quais 124 sofreram atualizações, logo M1 = 67,76 %		Considerou o tipo, ciclo, modelo de gerenciamento e três ferramentas distintas, logo, M3 = 6/9.

5. Carga parcial, com a inclusão de atividades nos cronogramas ocasionando na alteração da identificação das atividades.

TABELA 13 - QUINTA CARGA

	M1[10]	M2	M3
Projeto 1	Possui 205 atividades carregadas no DW, das quais 18 foram incluídas, logo M1 = 8,78%	Utiliza procedimentos automatizados para a extração, transformação e carga, porém devido a erros na execução ajustes manuais foram necessários.	Considerou o tipo, ciclo, modelo de gerenciamento e três ferramentas distintas, logo, M3 = 6/9.
Projeto 2	Possui 144 atividades carregadas no DW, das quais 18 foram incluídas, logo M1 = 12,5%		Considerou o tipo, ciclo, modelo de gerenciamento e duas ferramentas distintas, logo, M3 = 5/9.
Projeto 3	Possui 201 atividades carregadas no DW, das quais 18 foram incluídas, logo M1 = 8,95 %		Considerou o tipo, ciclo, modelo de gerenciamento e três ferramentas distintas, logo, M3 = 6/9.

6. Carga parcial, somente incremento de valores das atividades com a finalização de todas as atividades de um projeto 1.

TABELA 14 - SEXTA CARGA

	M1	M2	M3
Projeto 1	Possui 205 atividades carregadas no DW, das quais 37 sofreram atualizações, logo M1 = 18,05%	Utiliza procedimentos automatizados para a extração, transformação e carga, logo, M2 = 0.	Considerou o tipo, ciclo, modelo de gerenciamento e três ferramentas distintas, logo, M3 = 6/9.
Projeto 2[11]	Possui 144 atividades carregadas no DW, das quais 56 sofreram atualizações, logo M1 = 38,89%.		Considerou o tipo, ciclo, modelo de gerenciamento e três ferramentas distintas, logo, M3 = 5/9.
Projeto 3	Possui 201 atividades carregadas no DW, das quais 93 sofreram atualizações, logo M1 = 46,27 %		Considerou o tipo, ciclo, modelo de gerenciamento e três ferramentas distintas, logo, M3 = 6/9.

[10] Na primeira execução da quinta carga não houve renovação dos fatos devido a problemas na carga. O problema estava na falta de um relacionamento entre a informação carregada no DW e o dado bruto sendo extraído. A solução para tal, foi armazenar a identificação única de cada informação extraída, evitando assim, que cargas posteriores sejam efetuadas erroneamente.

[11] Na execução da sexta carga, o projeto 2 foi considerado finalizado, pois todas as suas atividades foram concluídas. Isso siginifica que na próxima carga não devem ser extraídas informações referentes a este projeto.

7. Carga parcial, somente incremento de valores das atividades com a finalização de algumas atividades.

TABELA 15 - SÉTIMA CARGA

	M1	M2	M3
Projeto 1	Possui 205 atividades carregadas no DW, das quais 93 sofreram atualizações, logo M1 = 45,37%	Utiliza procedimentos automatizados para a extração, transformação e carga, logo, M2 = 0	Considerou o tipo, ciclo, modelo de gerenciamento e três ferramentas distintas, logo, M3 = 6/9.
Projeto 2	Projeto Finalizado.	Projeto Finalizado.	Projeto Finalizado.
Projeto 3	Possui 201 atividades carregadas no DW, das quais 115 sofreram atualizações, logo M1 = 57,21 %	Utiliza procedimentos automatizados para a extração, transformação e carga, logo, M2 = 0	Considerou o tipo, ciclo, modelo de gerenciamento e três ferramentas distintas, logo, M3 = 6/9.

8.4.1 Análise Quantitativa dos Resultados

Após a execução das sete cargas, obtiveram-se os seguintes resultados ilustrados nas Figuras 37, 38 e 39. A Figura 37 ilustra os resultados obtidos para o aspecto que analisava se a solução é evolutiva. Desconsiderando a primeira execução, pois o DW estava vazio, somente a quinta carga apresentou problemas. A Figura 38 apresenta os resultados obtidos em relação ao aspecto baixa intrusão, onde exceto a quinta carga que apresentou problemas e foram necessários procedimentos manuais, as demais utilizaram procedimentos automatizados para extração, transformação e carga. A Figura 39 demonstra os resultados obtidos para o aspecto que buscava analisar se a proposta possui um tratamento adequado da heterogeneidade, onde em todas as execuções de carga os resultados foram satisfatórios.

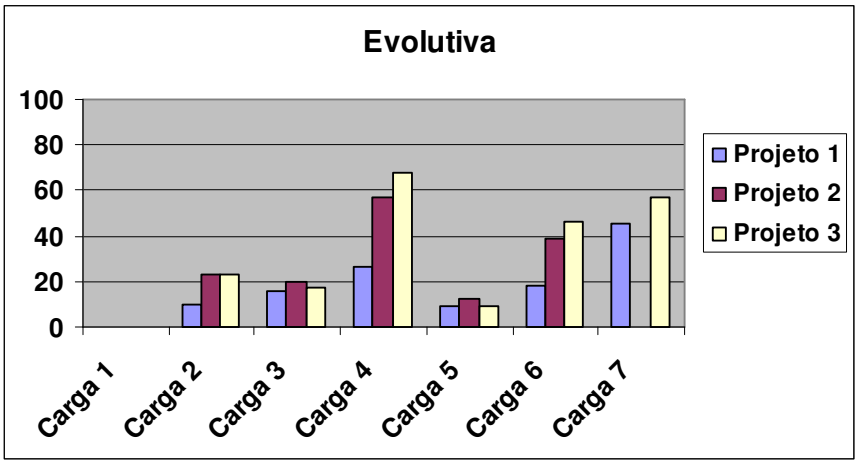

Figura 37 - Resultados do aspecto evolutiva

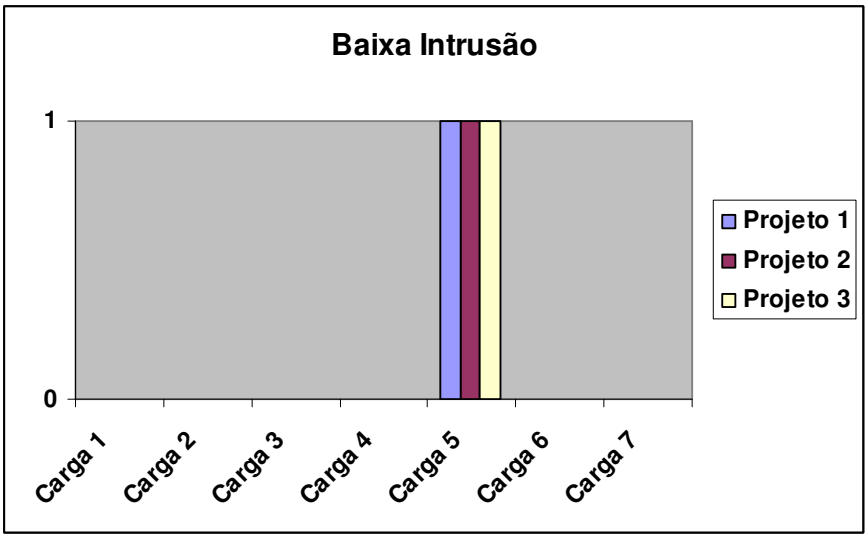

Figura 38 - Resultados do aspecto baixa intrusão

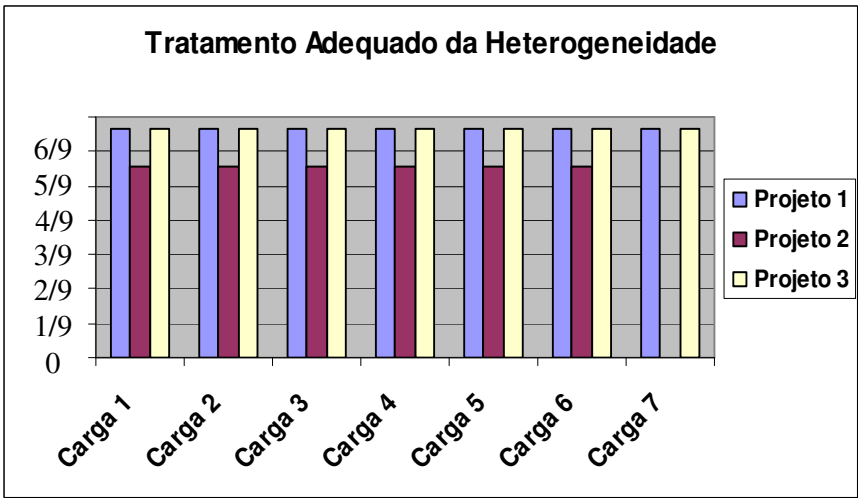

Figura 39 - Resultados do aspecto tratamento adequado da heterogeneidade

8.5 Considerações Adicionais

Esse capítulo serviu para verificar se a solução proposta abrange seus aspectos principais: evolutiva, de baixa intrusão e adequado tratamento da heterogeneidade. Inicialmente estabeleceram-se os objetivos, posteriormente para cada aspecto definiram-se as questões a serem respondidas e elaboraram-se as respectivas métricas que foram utilizadas no experimento. Posteriormente, descreveram-se as características dos três projetos utilizados para teste, salientando suas principais semelhanças e diferenças.

Na seqüência foram executas sete cargas buscando analisar e interpretar os resultados dessas execuções. Nessa interpretação observaram-se primeiramente as métricas obtidas, depois as respostas obtidas para as questões levantadas para o experimento e ao final agruparam-se esses os valores obtidos buscando demonstrar que os resultados obtidos durante o experimento foram considerados satisfatórios, tanto pelo ponto de vista do atingimento dos objetivos propostos neste trabalho, quanto pela possibilidade de melhoria da solução, pois com este experimento foram encontrados problemas que puderam ser solicionados.

O capítulo seguinte apresenta os trabalhos relacionados à solução proposta, apresentada neste trabalho e, ao final os compara a esta solução considerando os três aspectos discutidos neste capítulo.

9 TRABALHOS RELACIONADOS

Este capítulo discorre sobre propostas específicas relacionadas ao Processo de Data Warehousing para PDS. Primeiramente, são apresentadas essas propostas, posteriormente é feita uma comparação desses trabalhos com a solução proposta, considerando os seguintes fatores: baixa intrusão, extração não incremental e heterogeneidade.

Embora durante a pesquisa não tenham sido encontradas propostas específicas que abordem o processo de ETC para PDS, foram encontrados dois trabalhos direcionados a processos de negócios, os quais são examinados como segue.

9.1 A Framework for Analyzing and Measuring Business Performance with Web Services

A abordagem proposta por [GRE03] apresenta um *framework* que combina conceitos da arquitetura orientada a serviços, princípios de sistemas de suporte a decisão e uma abordagem multi-agentes. O mesmo tem por objetivo o monitoramento e análise de processos de negócios, fornecendo informações focadas no *status* e na performance desses processos, independentemente da aplicação utilizada. A Figura 40 ilustra esse *framework* denominado *Solution Manager Service* que possui quatro elementos principais: (1) *Web Services Interfaces* (WSI), (2) *Agent Server* (AS), (3) *Event Processing Container* (EPC) *e* (4) *Data Warehouse* (DW). O WSI é responsável pela extração de *logs* de dados. O EPC é responsável pela integração dos *logs* de dados provenientes dos processos de negócios que serão posteriormente carregados nas tabelas *Audit Log Data* e *Audit Summary Data* no DW. A tabela *Definition Data* do DW armazena os dados produzidos pelo AS resultantes do WS.

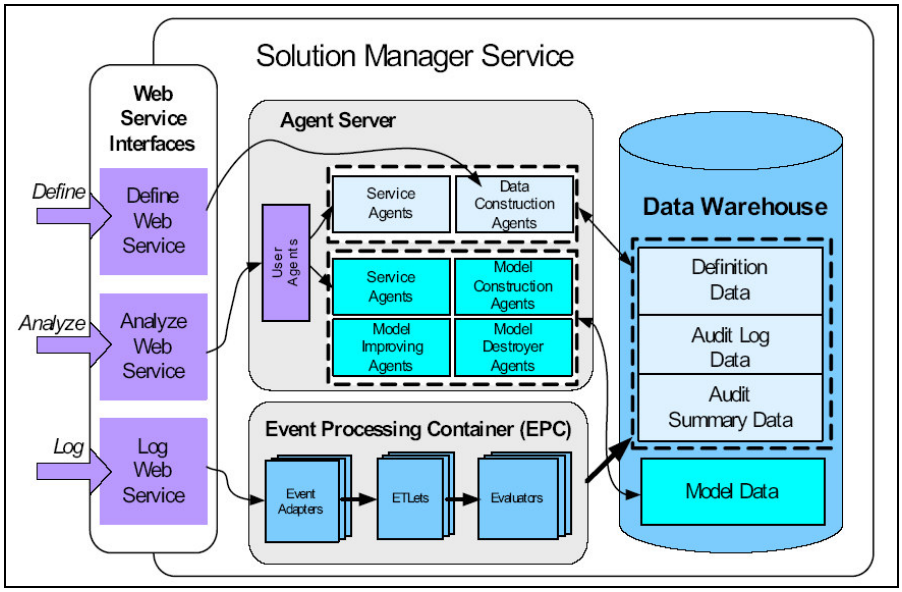

Figura 40 - Arquitetura SMWS, extraída de [GRE03]

9.2 Business Process Intelligence (BPI)

[CAS04 e GRI04] apresentam um conjunto de conceitos e uma arquitetura de implementação denominada *Business Process Intelligence* (BPI), que provê diversas características para a gestão de processos. São elas: análise, predição, monitoramento, controle e otimização. Um dos principais componentes do BPI (ver Figura 41) é o *Process Data Warehouse Loader* (*PDWLoader*) cuja funcionalidade é extrair dados de *logs* de execução de processos, verificar a consistência dos dados, limpar, calcular métricas e carregar no *Process Data Warehouse* (PDW). O PDW possui uma estrutura que habilita diversas funcionalidades analíticas, pois os dados são organizados de acordo com um esquema constelação de fatos (ver Figura 42). As mudanças nos estados dos processos, serviços e nodos são os fatos a serem analisados, enquanto que as definições de processos, as definições de nodos, definições de serviços, definições de dados, recursos, tempo e comportamento são as dimensões sobre as quais os dados fato podem ser analisados. Através de técnicas de DW e de mineração de dados, o BPI oferece mecanismos para análise e predição sobre execuções de quaisquer tipos de processos de negócio. Essas análises são realizadas através de um conjunto de interfaces gráficas, denominado *Cockpit*, focado no acompanhamento dos processos através de métricas e indicadores.

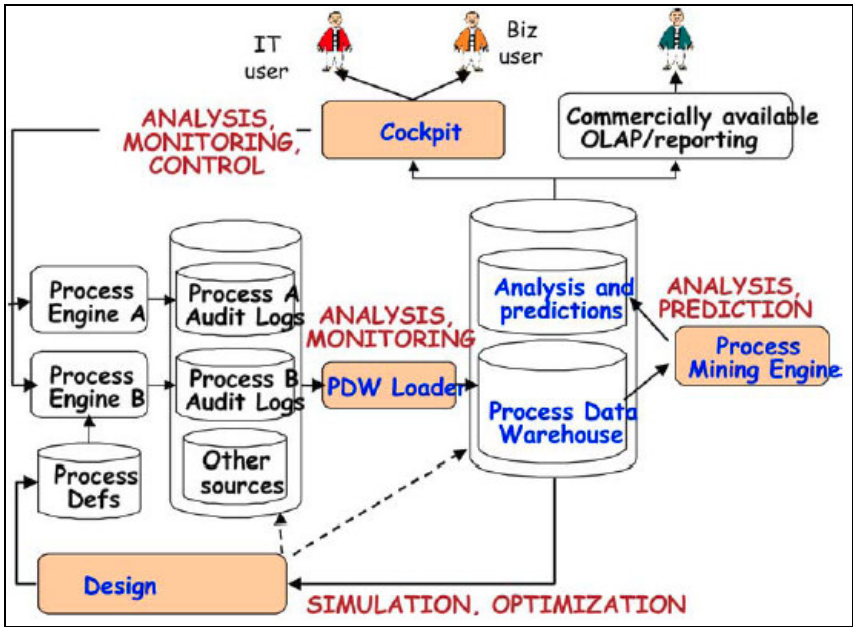

Figura 41 - Arquitetura do BPI, extraída de [CAS04 e GRI04]

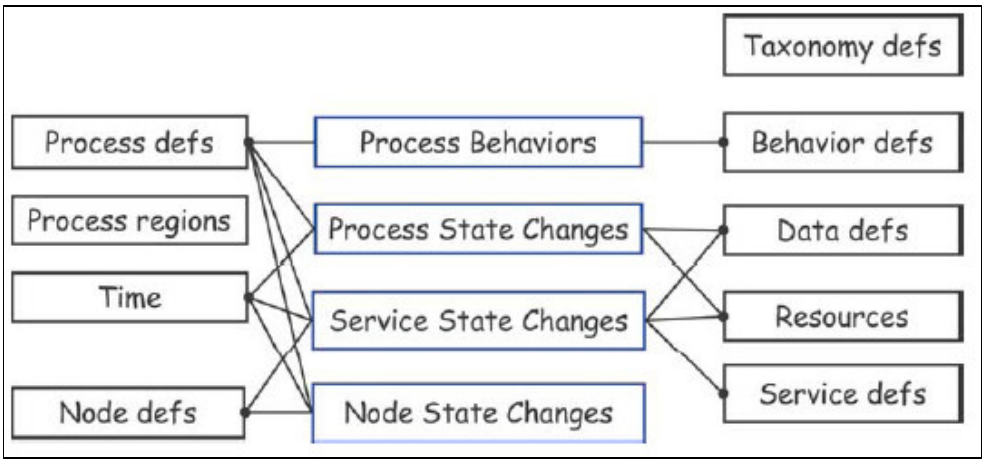

Figura 42 - Esquema constelação de fatos do PDW, extraída de extraída de [CAS04 e GRI04]

9.3 Comparação entre os trabalhos relacionados e a solução proposta

Similarmente ao nosso trabalho, a abordagem proposta por [GRE03] é baseada em uma arquitetura orientada a serviços e suporta a integração de dados provenientes de diversas fontes, mas restringe-se a tratar apenas a questão de extração, do processo de ETC, considerando alguns aspectos de heterogeneidade, tais como, independência da plataforma e linguagem de programação utilizada pelo sistema de origem. Já nossa abordagem, endereça além da extração, a transformação e carga utilizando uma arquitetura orientada a serviços.

O BPI [CAS04 e GRI04] é semelhante a nossa abordagem, pois propõe uma arquitetura abrangente. Porém não considera especificidades de PDSs, tais como heterogeneidade e métricas específicas dos processos e produtos de *software*, essenciais para um controle e acompanhamento efetivo, bem como para processos de certificação da maturidade.

A Tabela 9 tem por objetivo comparar os trabalhos relacionados a solução proposta considerando os seguintes aspectos: objetivos, heterogeneidade, baixa intrusão, extração evolutiva, especificidades dos PDS, métricas e desvantagens.

TABELA 16 - TRABALHOS RELACIONADOS X SOLUÇÃO PROPOSTA

	SMWS	**BPI**	**Solução Proposta**
Objetivos	Monitorar e analisar processos de negócios através de uma arquitetura orientada a serviços.	Prover diversas características para a gestão dos processos de negócios: análise, predição, monitoramento, controle e otimização.	Prover, através de uma abordagem orientada a serviços, informações consistentes e de qualidade para a gestão de PDSs.
Heterogeneidade	Sim	Não	Sim
Baixa Intrusão	Sim	Sim	Sim
Extração Evolutiva	Sim	Sim	Sim
Especificidades dos PDS	Não	Não	Sim
Métricas	Não	Sim	Sim
Desvantagens	Não considera as especificidades dos PDS	Não considera as especificidades dos PDS	Conjunto especifíco de métricas

9.4 Considerações Adicionais

Esse capítulo buscou comparar trabalhos concorrentes a nossa proposta. O primeiro trabalho apresentado (Seção 9.1) utiliza uma arquitetura orientada a serviços para extração, considerando assim, a heterogeneidade das fontes de dados. O segundo, (Seção 9.2) apresenta-se como o trabalho mais semelhante a solução proposta, apesar de não tratar as especificidades dos PDS.

Ao final é apresentada uma tabela comparativa desses trabalhos à solução proposta, considerando os seguintes pontos: objetivos, heterogeneidade, baixa intrusão, extração evolutiva, especificidades dos PDS, métricas, bem como as desvantagens de cada abordagem. Através desta tabela foi possível mostrar que os trabalhos relacionados não consideram os todos os aspectos considerados na solução proposta, tais como heterogeneidade, baixa intrusão e evolutiva. Em especial, nenhum dos trabalhos correlatos considera as especificidades de PDS.

10 CONSIDERAÇÕES FINAIS

Este trabalho propôs um ambiente de ETC como infra-estrutura de apoio à adoção de um programa de métricas em uma operação de *software* de nível CMM2. O ambiente proposto tem por mérito tratar de forma integrada dois aspectos cruciais do processo de *data warehousing*: 1) a coleta de dados dos projetos, sujeita a vários tipos de heterogeneidade e 2) sua transformação e integração, para proporcionar uma visão organizacional unificada e quantitativa dos projetos.

Visando caracterizar o problema, os objetivos e a questão de pesquisa a ser tratada, foi realizado um estudo de um ambiente real em uma operação de *software* CMM2[12].

A solução proposta neste trabalho está inserida em uma arquitetura para o ambiente de *Data Warehousing*, composta de três camadas: integração de aplicações, integração de dados e apresentação [NOV05, RUI05]. A camada de integração de aplicações é responsável pela extração dos dados brutos dos projetos considerando as especificidades das ferramentas. A camada de integração de dados compreende a DSA e o DW propriamente dito. O processo de ETC atua sobre as camadas de integração de aplicações e de integração de dados, e segue uma abordagem orientada a serviços.

No tocante à ETC, são tratados vários tipos de heterogeneidade, tanto do ponto de vista organizacional (e.g. especializações da OSSP que resultam em formas distintas de desenvolvimento e registro de fatos sobre projetos), quanto do ponto de vista técnico (e.g. diferentes ferramentas). A abordagem orientada a serviços para integração das aplicações tem como vantagens: 1) tratar dos vários tipos de heterogeneidade através de serviços que atuam como *wrappers* dos diferentes tipos de extratores, 2) permitir um mecanismo não intrusivo para extração das métricas necessárias à organização, e 3) suportar um ambiente distribuído

[12] Cabe ressaltar que durante a execução dessa pesquisa, a organização de *software* em questão obteve certificação CMM Nível 3. Além disso, o presente trabalho contribuiu para essa certificação, a medida que foram realizados estudos de como obter dados brutos de fontes heterogêneas, bem como as transformações necessárias para tornar esses dados brutos em informações consistentes e de qualidade.

de desenvolvimento. As primeiras duas vantagens são fundamentais para aumentar a aceitação do programa de métricas implantado pela organização. Ao tratar da heterogeneidade, permite-se que os diferentes processos especializem os elementos constantes da OSSP (Ver Seção 2.1) e adotem as ferramentas de apoio mais adequadas às suas especificidades e graus de produtividade desejados. Outro benefício é permitir que esses projetos acompanhem as evoluções tecnológicas, incorporando novas ferramentas, caso conveniente. A extração não intrusiva e automatizada não gera sobrecarga para as equipes, permite definir a periodicidade desejada para coleta, e aumenta a acurácia dos dados coletados. A última vantagem citada vai ao encontro das tendências de desenvolvimento distribuído de grandes organizações (e.g. *outsourcing*).

Para avaliar a solução proposta, foi definido um ambiente de experimentação considerando os seus três aspectos principais (ver Capítulo 8). Essa experimentação visou demonstrar que essa solução, para uma abordagem orientada a serviços para captura de métricas de PDS, caracteriza-se como evolutiva, de baixa intrusão e adequado tratamento da heterogeneidade.

Quanto à extensibilidade da abordagem proposta, cabe ressaltar que a mesma foi desenvolvida visando que novas especificidades de PDS possam ser consideradas, bem como novas ferramentas possam ser adicionadas.

10.1 Trabalhos Futuros

Para esta proposta discorre-se sobre os trabalhos futuros, os quais visam tornar a solução proposta a mais genérica possível. Estes trabalhos, estão descritos como segue.

1. Analisar como esta solução se comportaria em organizações com níveis de maturidade 4 e 5, já que a solução é focada para organizações de *software* certificadas nível 2 e 3. A cada nível de maturidade alcançado são agregadas novas características de medição e análise, onde essas análises devem prover melhorias e a customização dos PDS provendo assim, as inovações tecnológicas necessárias. Acredita-se que a abordagem proposta seja adequada para a evolução do programa de métricas para níveis superiores, pois uma das vantagens desta solução proposta é a possibilidade de evoluir com o tempo, tanto do ponto de vista da adoção de novas ferramentas, quanto às mudanças em relação ao ciclo de vida, modelo de gerenciamento, e etc. O que

provavelmente deverá ser trabalhado é a forma como essas informações serão apresentadas para o usuário final.

2. Examinar a viabilidade de extrair informações de uma maior diversidade de PDS, ampliando assim o espectro de informações que serão agregadas ao DW.
3. Pesquisar a possibilidade de efetuar a carga em diferentes modelos analíticos através da definição de metadados de carga.
4. Estudar o suporte a evolução dos programas de métricas, possibilitando assim aplicar essa abordagem para organizações que não possuam um programa fixo de métricas.
5. Buscar encontrar uma padronização do processo de ETC para organizações de *software*, buscando determinar que métricas armazenadas no modelo analítico podem possuir um comportamento padrão de extração, transfomação e carga.

REFERÊNCIAS BIBLIOGRÁFICAS

[ALO04] G. Alonso, F. Casati, H. Kuno, V. Machiraju. “Web Services – Concepts, Architetuctures and Applications”. Berlin: Springer, 2004, 354p.

[BAR02] A. Bartié. “Garantia da Qualidade de Software: adquirindo maturidade organizacional”. Rio de Janeiro: Elsevier, 2002, 3ª Edição, 291p.

[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, pp. 239 – 273. Novembro 2004,

[FAC02] O. Fachin. “Fundamentos de Metodologia”. São Paulo: Editora Saraiva, 2002, 3ª Edição, 200p.

[FUG00] Fuggetta, A. “Software Process: A Roadmap”, In: The future of Software engineering. Limerick Ireland: ACM Press, 2000. Capturado em: http://store.acm.org/acmstore, Março 2005. 8p.

[GRE03] C. McGregor, J. Schiefer. “A Framework for Analyzing and Measuring Business Performance with Web Services”. In: IEEE International Conference on E-Commerce (CEC’03), 2003, Newport Beach – CA – USA. **Proceedings…** Los Alamitos – CA – USA: IEEE Computer Society, Junho 2003, pp.405-412.

[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, pp. 324-343. Abril 2004.

[HAN01] J. Han, M. Kamber. “Data Mining: Concepts and Techniques”. San Diego – CA - USA: Academic Press, 2001, 550p.

[HAR91] J. Harrington. “Business Process Improvement – The breakthrough strategy for total quality, productivity, and competitiveness". New York: Mc-Graw-Hill, 1991, 274p.

[IEE98] IEEE, "IEEE Std. 1061-1998, IEEE Standard for a Software Quality Metrics Methodology" Piscataway, NJ: IEEE Standards Dept., 1998. Capturado em: http://ieeexplore.ieee.org/, Março 2005, 94p.

[INM96] W. Inmon. "Building the Data Warehouse". New York: John Wiley &

Sons, Inc., 1996, 2ª Edição, 401p.

[ISO93] ISO, ISO: International Vocabulary of Basic and Generated Terms in Metrology. Geneva: International Organization for Standardization, 1993.

[JUP05] Jupitermedia Corporation. Webopedia: Online Dictionary for Computer and Internet Terms. Capturado em: http:// www.webopedia.com/, Abril 2005.

[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". New York: John Wiley & Sons, Inc., 1998, 771p.

[LEY02] F. Leymann, D. Roller, M. Schmidt. "Web services and business process management". In: New Developments in Web Services and E-commerce. IBM Systems Journal, vol. 41-2, 2002, pp 198-211.

[LIS04] B. List, K. Machaczek. "Towards a Corporate Performance Measurement System". In: Proceedings of the 19th Annual ACM Symposium on Applied Computing (SAC´04), 2004, Nicosia – Cyprus. **Proceedings…** New York – USA: ACM Press, 2004, pp. 1344-1350.

[MIT04] Mitra N., SOAP Version 1.2 Part 0: Primer, Jun 2003. Capturado em http://www.w3.org/TR/soap12-part0/ , Maio 2005.

[NEW02] E. Newcomer. "Understanding Web Services: XML, SOAP, and UDDI". New York: Addison Wesley, 2002, 332p.

[NOV05] T. Novello, K. Becker. "Recursos Analíticos para Análise e Monitoramento de Processos de Desenvolvimento de Software". Relatório de Seminário de Andamento. Pontifícia Universidade Católica do Rio Grande do Sul, 2005.

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

[PAL03] E. Palza, C. Fuhrman, A. Abran. "Establishing a Generic and Multidimensional Measurement Repository in CMMI context". IN: 28th

Annual IEEE/NASA *Software* Engineering Workshop, 2003, Greenbelt, MD, USA. **Proceedings…** Los Alamitos – CA – USA: IEEE Computer Society Press, 2003. pp.12-20.

[PAU93] M. C. Paulk., B. Curtis, M. Chrissis, C. Weber. "Capability Maturity Model for Software", Version.1.1 Technical Report. Software Engineering Institute, Carnegie-Mellon University. Pittsburgh, PA: 1993. Capturado em: http://www.sei.cmu.edu/publications/documents/93.reports/93.tr.024.html.

[PRE95] R. Pressman. "Engenharia de Software". São Paulo: Mackron Books, 1996, 1056p.

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

[ROC01] A. Rocha, J. Maldonado, K, Weber. "Qualidade de Software Teoria e Prática. São Paulo: Prentice Hall, 2001, 303p.

[RUI05] D. Ruiz, K. Becker, T. Novello, 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…** Los Alamitos – CA – USA : IEEE Computer Society Press, 2005. pp. 936-940.

[RYM04] A. Ryman. "Understanding Web Services". Capturado em: http://www-106.ibm.com/developerworks/websphere/library/techarticles/0307ryman/ryman.htm, Junho 2005.

[SEI96] SEI, Software Engineering Institute, CMMI for Software Engineering (CMMI-SW, V1.1), Staged Representation. 2002, Carnegie Mellon University, Software Engineering Institute: Pittsburgh Henderson-Sellers B., Object Oriented Metrics: Measures of Complexity, Prentice Hall, 1996.

[SOM04] I. Sommerville. "Software Engineering". Harlow: Addison-Wesley, 2004, 6ª Edição, 592p.

[SPI04] SPICE, "Software Process Improvement and Capability dEtermination". Capturado em: http://www.sqi.gu.edu.au/spice, Julho 2004.

[TRA02] G. Travassos, D. Gurov, E. Amaral. “Introdução à Engenharia de Software Experimental”. Relatório Técnico, Rio de Janeiro: COPPE/UFRJ, 2002.

[UDD03] UDDI Consortium. UDDI Executive White Paper, Nov. 2001. Capturado em: http:// uddi.org/pubs/ UDDI_Executive_White_ Paper.pdf, Junho 2004.

[VAS02] P. Vassiliadis, A. Simitsis, S. Skiadopoulos. “Modeling ETL activities as graphs”. In: Proceedings of Design and Management of Data Warehouses (DMDW 2002), 2002, Toronto – CA. **Proceedings…** Aachen – Germany: CEUR-WS.org Workshop Procedings, 2002. pp. 52-61.

[YIN05] R. Yin. “Estudo de Caso: Planejamento e Métodos”, Porto Alegre: Bookman, 2005, 3ª Edição, 212p.

[YOO01] I. Yoon, S. Min, D. Bae. “Tailoring and Verifying Software Process”, In: Eighth Asia-Pacific Software Engineering Conference (APSEC´01), 2001, Macau – China. **Proceedings…** Los Alamitos – CA – USA: IEEE Computer Society Press, 2001. pp.202-209.

ANEXOS

ANEXO A

Metadados de Projeto

PROJETO 1

```
<?xml version="1.0" encoding="UTF-8" ?>
- <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
- <xs:element name="Projeto1">
- <xs:element name="ProjectServer">
- <xs:element name="Nome_Projeto">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseText3" />
  <xs:element name="Tipo" value="ntext" />
  <xs:element name="Tamanho" value="16" />
    </xs:element>
- <xs:element name="Versao">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseText2" />
  <xs:element name="Tipo" value="ntext" />
  <xs:element name="Tamanho" value="16" />
    </xs:element>
- <xs:element name="Cliente">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseText1" />
  <xs:element name="Tipo" value="ntext" />
  <xs:element name="Tamanho" value="16" />
    </xs:element>
- <xs:element name="Tamanho_Time">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseNumber4" />
  <xs:element name="Tipo" value="int" />
  <xs:element name="Tamanho" value="4" />
    </xs:element>
- <xs:element name="Custo_Total">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseNumber6" />
  <xs:element name="Tipo" value="int" />
  <xs:element name="Tamanho" value="4" />
    </xs:element>
- <xs:element name="Industria">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseOutlineCode1ID" />
  <xs:element name="Tipo" value="int" />
  <xs:element name="Tamanho" value="4" />
    </xs:element>
- <xs:element name="Tecnologia">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseOutlineCode2ID" />
  <xs:element name="Tipo" value="int" />
  <xs:element name="Tamanho" value="4" />
    </xs:element>
- <xs:element name="Tipo_Projeto">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseOutlineCode3ID" />
  <xs:element name="Tipo" value="int" />
  <xs:element name="Tamanho" value="4" />
    </xs:element>
- <xs:element name="Porte_Projeto">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseOutlineCode4ID" />
  <xs:element name="Tipo" value="int" />
  <xs:element name="Tamanho" value="4" />
```

```
    </xs:element>
- <xs:element name="Status">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskPercentWorkComplete" />
  <xs:element name="Tipo" value="smallint" />
  <xs:element name="Tamanho" value="2" />
    </xs:element>
- <xs:element name="Unidade_Tamanho">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseOutlineCode5ID" />
  <xs:element name="Tipo" value="int" />
  <xs:element name="Tamanho" value="4" />
    </xs:element>
- <xs:element name="Tamanho_BO">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseText4" />
  <xs:element name="Tipo" value="ntext" />
  <xs:element name="Tamanho" value="16" />
    </xs:element>
- <xs:element name="Tamanho_BR">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseText5" />
  <xs:element name="Tipo" value="ntext" />
  <xs:element name="Tamanho" value="16" />
    </xs:element>
- <xs:element name="Tamanho_Real">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseText6" />
  <xs:element name="Tipo" value="ntext" />
  <xs:element name="Tamanho" value="16" />
    </xs:element>
- <xs:element name="Iteracoes">
- <xs:element name="NomeIteracao">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskName" />
  <xs:element name="Tipo" value="nvarchar" />
  <xs:element name="Tamanho" value="256" />
    </xs:element>
- <xs:element name="Fases">
- <xs:element name="NomeFase">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskName" />
  <xs:element name="Tipo" value="nvarchar" />
  <xs:element name="Tamanho" value="256" />
    </xs:element>
- <xs:element name="Tipos">
- <xs:element name="Tipo_Atividade">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskName" />
  <xs:element name="Tipo" value="nvarchar" />
  <xs:element name="Tamanho" value="256" />
    </xs:element>
- <xs:element name="Atividades">
- <xs:element name="NomeAtividade">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskName" />
  <xs:element name="Tipo" value="nvarchar" />
  <xs:element name="Tamanho" value="256" />
    </xs:element>
```

```
- <xs:element name="Custo_BO">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskBaselineCost" />
  <xs:element name="Tipo" value="decimal" />
  <xs:element name="Tamanho" value="13" />
    </xs:element>
- <xs:element name="Custo_BR">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskBaseline1Cost" />
  <xs:element name="Tipo" value="decimal" />
  <xs:element name="Tamanho" value="13" />
    </xs:element>
- <xs:element name="Esforco_BO">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskBaselineWork" />
  <xs:element name="Tipo" value="decimal" />
  <xs:element name="Tamanho" value="13" />
    </xs:element>
- <xs:element name="Esforco_BR">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskBaseline1Work" />
  <xs:element name="Tipo" value="decimal" />
  <xs:element name="Tamanho" value="13" />
    </xs:element>
- <xs:element name="Data_Inicial_BO">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskBaseline1Start" />
  <xs:element name="Tipo" value="datetime" />
  <xs:element name="Tamanho" value="8" />
    </xs:element>
- <xs:element name="Data_Inicial_BR">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskBaselineStart" />
  <xs:element name="Tipo" value="datetime" />
  <xs:element name="Tamanho" value="8" />
    </xs:element>
- <xs:element name="Data_Final_BO">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskBaseline1Finish" />
  <xs:element name="Tipo" value="datetime" />
  <xs:element name="Tamanho" value="8" />
    </xs:element>
- <xs:element name="Data_Final_BR">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskBaselineFinish" />
  <xs:element name="Tipo" value="datetime" />
  <xs:element name="Tamanho" value="8" />
    </xs:element>
    </xs:element>
    </xs:element>
    </xs:element>
    </xs:element>
    </xs:element>
    </xs:element>
- <xs:element name="Excel">
- <xs:element name="Atividades">
- <xs:element name="Data_Inicial_Real">
  <xs:element name="Tabela" value="Atividades" />
  <xs:element name="Campo" value="Data" />
```

```
<xs:element name="Tipo" value="date" />
<xs:element name="Tamanho" value="6" />
</xs:element>
- <xs:element name="Data_Final_Real">
<xs:element name="Tabela" value="Atividades" />
<xs:element name="Campo" value="Data" />
<xs:element name="Tipo" value="date" />
<xs:element name="Tamanho" value="6" />
</xs:element>
- <xs:element name="Nome_Fase">
<xs:element name="Tabela" value="Atividades" />
<xs:element name="Campo" value="Equipe" />
<xs:element name="Tipo" value="varchar" />
<xs:element name="Tamanho" value="50" />
</xs:element>
- <xs:element name="Nome_Atividade">
<xs:element name="Tabela" value="Atividades" />
<xs:element name="Campo" value="Atividade" />
<xs:element name="Tipo" value="varchar" />
<xs:element name="Tamanho" value="50" />
</xs:element>
- <xs:element name="Tipo_Atividade">
<xs:element name="Tabela" value="Atividades" />
<xs:element name="Campo" value="Complemento" />
<xs:element name="Tipo" value="varchar" />
<xs:element name="Tamanho" value="50" />
</xs:element>
- <xs:element name="Esforco_Real">
<xs:element name="Tabela" value="Atividades" />
<xs:element name="Campo" value="Total" />
<xs:element name="Tipo" value="decimal" />
<xs:element name="Tamanho" value="10" />
</xs:element>
</xs:element>
</xs:element>
- <xs:element name="ClearQuest">
- <xs:element name="Defeitos">
- <xs:element name="Categoria">
<xs:element name="Tabela" value="Defect" />
<xs:element name="Campo" value="Type" />
<xs:element name="Tipo" value="varchar" />
<xs:element name="Tamanho" value="50" />
</xs:element>
- <xs:element name="Severidade">
<xs:element name="Tabela" value="Defect" />
<xs:element name="Campo" value="Severity" />
<xs:element name="Tipo" value="varchar" />
<xs:element name="Tamanho" value="50" />
</xs:element>
- <xs:element name="Tipo">
<xs:element name="Tabela" value="Defect" />
<xs:element name="Campo" value="PeerReview" />
<xs:element name="Tipo" value="varchar" />
<xs:element name="Tamanho" value="50" />
</xs:element>
</xs:element>
</xs:element>
</xs:schema>
```

PROJETO 2

```
<?xml version="1.0" encoding="UTF-8" ?>
- <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
- <xs:element name="Projeto2">
- <xs:element name="ProjectServer">
- <xs:element name="Nome_Projeto">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseText3" />
  <xs:element name="Tipo" value="ntext" />
  <xs:element name="Tamanho" value="16" />
    </xs:element>
- <xs:element name="Versao">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseText2" />
  <xs:element name="Tipo" value="ntext" />
  <xs:element name="Tamanho" value="16" />
    </xs:element>
- <xs:element name="Cliente">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseText1" />
  <xs:element name="Tipo" value="ntext" />
  <xs:element name="Tamanho" value="16" />
    </xs:element>
- <xs:element name="Tamanho_Time">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseNumber4" />
  <xs:element name="Tipo" value="int" />
  <xs:element name="Tamanho" value="4" />
    </xs:element>
- <xs:element name="Custo_Total">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseNumber6" />
  <xs:element name="Tipo" value="int" />
  <xs:element name="Tamanho" value="4" />
    </xs:element>
- <xs:element name="Industria">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseOutlineCode1ID" />
  <xs:element name="Tipo" value="int" />
  <xs:element name="Tamanho" value="4" />
    </xs:element>
- <xs:element name="Tecnologia">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseOutlineCode2ID" />
  <xs:element name="Tipo" value="int" />
  <xs:element name="Tamanho" value="4" />
    </xs:element>
- <xs:element name="Tipo_Projeto">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseOutlineCode3ID" />
  <xs:element name="Tipo" value="int" />
  <xs:element name="Tamanho" value="4" />
    </xs:element>
- <xs:element name="Porte_Projeto">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseOutlineCode4ID" />
  <xs:element name="Tipo" value="int" />
  <xs:element name="Tamanho" value="4" />
```

```
    </xs:element>
- <xs:element name="Status">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskPercentWorkComplete" />
  <xs:element name="Tipo" value="smallint" />
  <xs:element name="Tamanho" value="2" />
    </xs:element>
- <xs:element name="Unidade_Tamanho">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseOutlineCode5ID" />
  <xs:element name="Tipo" value="int" />
  <xs:element name="Tamanho" value="4" />
    </xs:element>
- <xs:element name="Tamanho_BO">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseText4" />
  <xs:element name="Tipo" value="ntext" />
  <xs:element name="Tamanho" value="16" />
    </xs:element>
- <xs:element name="Tamanho_BR">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseText5" />
  <xs:element name="Tipo" value="ntext" />
  <xs:element name="Tamanho" value="16" />
    </xs:element>
- <xs:element name="Tamanho_Real">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseText6" />
  <xs:element name="Tipo" value="ntext" />
  <xs:element name="Tamanho" value="16" />
    </xs:element>
- <xs:element name="Fases">
- <xs:element name="NomeFase">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskName" />
  <xs:element name="Tipo" value="nvarchar" />
  <xs:element name="Tamanho" value="256" />
    </xs:element>
- <xs:element name="Tipos">
- <xs:element name="Tipo_Atividade">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskName" />
  <xs:element name="Tipo" value="nvarchar" />
  <xs:element name="Tamanho" value="256" />
    </xs:element>
- <xs:element name="Atividades">
- <xs:element name="NomeAtividade">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskName" />
  <xs:element name="Tipo" value="nvarchar" />
  <xs:element name="Tamanho" value="256" />
    </xs:element>
- <xs:element name="Custo_BO">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskBaselineCost" />
  <xs:element name="Tipo" value="decimal" />
  <xs:element name="Tamanho" value="13" />
    </xs:element>
- <xs:element name="Custo_BR">
```

```
<xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
<xs:element name="Campo" value="TaskBaseline1Cost" />
<xs:element name="Tipo" value="decimal" />
<xs:element name="Tamanho" value="13" />
</xs:element>
- <xs:element name="Custo_Real">
<xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
<xs:element name="Campo" value="TaskCost" />
<xs:element name="Tipo" value="decimal" />
<xs:element name="Tamanho" value="13" />
</xs:element>
- <xs:element name="Esforco_BO">
<xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
<xs:element name="Campo" value="TaskBaselineWork" />
<xs:element name="Tipo" value="decimal" />
<xs:element name="Tamanho" value="13" />
</xs:element>
- <xs:element name="Esforco_BR">
<xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
<xs:element name="Campo" value="TaskBaseline1Work" />
<xs:element name="Tipo" value="decimal" />
<xs:element name="Tamanho" value="13" />
</xs:element>
- <xs:element name="Esforco_Real">
<xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
<xs:element name="Campo" value="TaskWork" />
<xs:element name="Tipo" value="decimal" />
<xs:element name="Tamanho" value="13" />
</xs:element>
- <xs:element name="Data_Inicial_BO">
<xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
<xs:element name="Campo" value="TaskBaseline1Start" />
<xs:element name="Tipo" value="datetime" />
<xs:element name="Tamanho" value="8" />
</xs:element>
- <xs:element name="Data_Inicial_BR">
<xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
<xs:element name="Campo" value="TaskBaselineStart" />
<xs:element name="Tipo" value="datetime" />
<xs:element name="Tamanho" value="8" />
</xs:element>
- <xs:element name="Data_Inicial_Real">
<xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
<xs:element name="Campo" value="TaskStart" />
<xs:element name="Tipo" value="datetime" />
<xs:element name="Tamanho" value="8" />
</xs:element>
- <xs:element name="Data_Final_BO">
<xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
<xs:element name="Campo" value="TaskBaseline1Finish" />
<xs:element name="Tipo" value="datetime" />
<xs:element name="Tamanho" value="8" />
</xs:element>
- <xs:element name="Data_Final_BR">
<xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
<xs:element name="Campo" value="TaskBaselineFinish" />
<xs:element name="Tipo" value="datetime" />
<xs:element name="Tamanho" value="8" />
</xs:element>
```

```
- <xs:element name="Data_Final_Real">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskFinish" />
  <xs:element name="Tipo" value="datetime" />
  <xs:element name="Tamanho" value="8" />
    </xs:element>
    </xs:element>
    </xs:element>
    </xs:element>
    </xs:element>
- <xs:element name="ClearQuest">
- <xs:element name="Defeitos">
- <xs:element name="Categoria">
  <xs:element name="Tabela" value="Defect" />
  <xs:element name="Campo" value="Type" />
  <xs:element name="Tipo" value="varchar" />
  <xs:element name="Tamanho" value="50" />
    </xs:element>
- <xs:element name="Severidade">
  <xs:element name="Tabela" value="Defect" />
  <xs:element name="Campo" value="Severity" />
  <xs:element name="Tipo" value="varchar" />
  <xs:element name="Tamanho" value="50" />
    </xs:element>
- <xs:element name="Tipo">
  <xs:element name="Tabela" value="Defect" />
  <xs:element name="Campo" value="PeerReview" />
  <xs:element name="Tipo" value="varchar" />
  <xs:element name="Tamanho" value="50" />
    </xs:element>
    </xs:element>
    </xs:element>
    </xs:element>
    </xs:schema>
```

PROJETO 3

```
<?xml version="1.0" encoding="UTF-8" ?>
- <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
- <xs:element name="Projeto3">
- <xs:element name="ProjectServer">
- <xs:element name="Nome_Projeto">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseText3" />
  <xs:element name="Tipo" value="ntext" />
  <xs:element name="Tamanho" value="16" />
    </xs:element>
- <xs:element name="Versao">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseText2" />
  <xs:element name="Tipo" value="ntext" />
  <xs:element name="Tamanho" value="16" />
    </xs:element>
- <xs:element name="Cliente">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseText1" />
  <xs:element name="Tipo" value="ntext" />
  <xs:element name="Tamanho" value="16" />
    </xs:element>
- <xs:element name="Tamanho_Time">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseNumber4" />
  <xs:element name="Tipo" value="int" />
  <xs:element name="Tamanho" value="4" />
    </xs:element>
- <xs:element name="Custo_Total">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseNumber6" />
  <xs:element name="Tipo" value="int" />
  <xs:element name="Tamanho" value="4" />
    </xs:element>
- <xs:element name="Industria">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseOutlineCode1ID" />
  <xs:element name="Tipo" value="int" />
  <xs:element name="Tamanho" value="4" />
    </xs:element>
- <xs:element name="Tecnologia">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseOutlineCode2ID" />
  <xs:element name="Tipo" value="int" />
  <xs:element name="Tamanho" value="4" />
    </xs:element>
- <xs:element name="Tipo_Projeto">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseOutlineCode3ID" />
  <xs:element name="Tipo" value="int" />
  <xs:element name="Tamanho" value="4" />
    </xs:element>
- <xs:element name="Porte_Projeto">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseOutlineCode4ID" />
  <xs:element name="Tipo" value="int" />
  <xs:element name="Tamanho" value="4" />
```

```
    </xs:element>
- <xs:element name="Status">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskPercentWorkComplete" />
  <xs:element name="Tipo" value="smallint" />
  <xs:element name="Tamanho" value="2" />
    </xs:element>
- <xs:element name="Unidade_Tamanho">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseOutlineCode5ID" />
  <xs:element name="Tipo" value="int" />
  <xs:element name="Tamanho" value="4" />
    </xs:element>
- <xs:element name="Tamanho_Estimado">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseNumber1" />
  <xs:element name="Tipo" value="int" />
  <xs:element name="Tamanho" value="4" />
    </xs:element>
- <xs:element name="Tamanho_Revisado">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseNumber3" />
  <xs:element name="Tipo" value="int" />
  <xs:element name="Tamanho" value="4" />
    </xs:element>
- <xs:element name="Tamanho_Atual">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_PROJECTS_ENT" />
  <xs:element name="Campo" value="ProjectEnterpriseNumber2" />
  <xs:element name="Tipo" value="int" />
  <xs:element name="Tamanho" value="4" />
    </xs:element>
- <xs:element name="Entregavel">
- <xs:element name="NomeEntregavel">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskName" />
  <xs:element name="Tipo" value="nvarchar" />
  <xs:element name="Tamanho" value="256" />
    </xs:element>
- <xs:element name="Fases">
- <xs:element name="NomeFase">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskName" />
  <xs:element name="Tipo" value="nvarchar" />
  <xs:element name="Tamanho" value="256" />
    </xs:element>
- <xs:element name="Atividades">
- <xs:element name="NomeAtividade">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskName" />
  <xs:element name="Tipo" value="nvarchar" />
  <xs:element name="Tamanho" value="256" />
    </xs:element>
- <xs:element name="Tipos">
- <xs:element name="Tipo_Atividade">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskName" />
  <xs:element name="Tipo" value="nvarchar" />
  <xs:element name="Tamanho" value="256" />
    </xs:element>
```

```
- <xs:element name="Custo_BO">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskBaselineCost" />
  <xs:element name="Tipo" value="decimal" />
  <xs:element name="Tamanho" value="13" />
    </xs:element>
- <xs:element name="Custo_BR">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskBaseline1Cost" />
  <xs:element name="Tipo" value="decimal" />
  <xs:element name="Tamanho" value="13" />
    </xs:element>
- <xs:element name="Esforco_BO">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskBaselineWork" />
  <xs:element name="Tipo" value="decimal" />
  <xs:element name="Tamanho" value="13" />
    </xs:element>
- <xs:element name="Esforco_BR">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskBaseline1Work" />
  <xs:element name="Tipo" value="decimal" />
  <xs:element name="Tamanho" value="13" />
    </xs:element>
- <xs:element name="Data_Inicial_BO">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskBaseline1Start" />
  <xs:element name="Tipo" value="datetime" />
  <xs:element name="Tamanho" value="8" />
    </xs:element>
- <xs:element name="Data_Inicial_BR">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskBaselineStart" />
  <xs:element name="Tipo" value="datetime" />
  <xs:element name="Tamanho" value="8" />
    </xs:element>
- <xs:element name="Data_Final_BO">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskBaseline1Finish" />
  <xs:element name="Tipo" value="datetime" />
  <xs:element name="Tamanho" value="8" />
    </xs:element>
- <xs:element name="Data_Final_BR">
  <xs:element name="Tabela" value="MSP_VIEW_PROJ_TASKS_STD" />
  <xs:element name="Campo" value="TaskBaselineFinish" />
  <xs:element name="Tipo" value="datetime" />
  <xs:element name="Tamanho" value="8" />
    </xs:element>
    </xs:element>
    </xs:element>
    </xs:element>
    </xs:element>
    </xs:element>
- <xs:element name="Excel">
- <xs:element name="Atividades">
- <xs:element name="Data_Inicial_Real">
  <xs:element name="Tabela" value="Atividades" />
  <xs:element name="Campo" value="Data" />
  <xs:element name="Tipo" value="date" />
```

```
    <xs:element name="Tamanho" value="6" />
      </xs:element>
  - <xs:element name="Data_Final_Real">
    <xs:element name="Tabela" value="Atividades" />
    <xs:element name="Campo" value="Data" />
    <xs:element name="Tipo" value="date" />
    <xs:element name="Tamanho" value="6" />
      </xs:element>
  - <xs:element name="Nome_Fase">
    <xs:element name="Tabela" value="Atividades" />
    <xs:element name="Campo" value="Equipe" />
    <xs:element name="Tipo" value="varchar" />
    <xs:element name="Tamanho" value="50" />
      </xs:element>
  - <xs:element name="Nome_Atividade">
    <xs:element name="Tabela" value="Atividades" />
    <xs:element name="Campo" value="Atividade" />
    <xs:element name="Tipo" value="varchar" />
    <xs:element name="Tamanho" value="50" />
      </xs:element>
  - <xs:element name="Tipo_Atividade">
    <xs:element name="Tabela" value="Atividades" />
    <xs:element name="Campo" value="Complemento" />
    <xs:element name="Tipo" value="varchar" />
    <xs:element name="Tamanho" value="50" />
      </xs:element>
  - <xs:element name="Esforco_Real">
    <xs:element name="Tabela" value="Atividades" />
    <xs:element name="Campo" value="Total" />
    <xs:element name="Tipo" value="decimal" />
    <xs:element name="Tamanho" value="10" />
      </xs:element>
      </xs:element>
      </xs:element>
  - <xs:element name="ClearQuest">
  - <xs:element name="Defeitos">
  - <xs:element name="Categoria">
    <xs:element name="Tabela" value="Defect" />
    <xs:element name="Campo" value="Type" />
    <xs:element name="Tipo" value="varchar" />
    <xs:element name="Tamanho" value="50" />
      </xs:element>
  - <xs:element name="Severidade">
    <xs:element name="Tabela" value="Defect" />
    <xs:element name="Campo" value="Severity" />
    <xs:element name="Tipo" value="varchar" />
    <xs:element name="Tamanho" value="50" />
      </xs:element>
  - <xs:element name="Tipo">
    <xs:element name="Tabela" value="Defect" />
    <xs:element name="Campo" value="PeerReview" />
    <xs:element name="Tipo" value="varchar" />
    <xs:element name="Tamanho" value="50" />
      </xs:element>
      </xs:element>
  - <xs:element name="Tamanho">
  - <xs:element name="Tamanho_BO">
    <xs:element name="Tabela" value="Size" />
    <xs:element name="Campo" value="OriginalSize" />
```

```
<xs:element name="Tipo" value="varchar" />
<xs:element name="Tamanho" value="50" />
  </xs:element>
- <xs:element name="Tamanho_BR">
<xs:element name="Tabela" value="Size" />
<xs:element name="Campo" value="RevisedSize" />
<xs:element name="Tipo" value="varchar" />
<xs:element name="Tamanho" value="50" />
  </xs:element>
- <xs:element name="Tamanho_Real">
<xs:element name="Tabela" value="Size" />
<xs:element name="Campo" value="RealSize" />
<xs:element name="Tipo" value="varchar" />
<xs:element name="Tamanho" value="50" />
  </xs:element>
  </xs:element>
  </xs:element>
  </xs:element>
  </xs:schema>
```

ANEXO B

Metadados Organizacionais

METADADO ORGANIZACIONAL

```
<?xml version="1.0" encoding="UTF-8" ?>
- <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
- <xs:element name="Organizacional">
- <xs:element name="Projeto1">
- <xs:element name="Severidade_Revisao">
  <xs:element name="A" value="Alta" />
  <xs:element name="M" value="Media" />
  <xs:element name="B" value="Baixa" />
    </xs:element>
- <xs:element name="Severidade_Teste">
  <xs:element name="1" value="Baixa" />
  <xs:element name="2" value="Media" />
  <xs:element name="3" value="Media" />
  <xs:element name="4" value="Alta" />
    </xs:element>
- <xs:element name="Tamanho_Total">
- <xs:element name="Linguagem">
  <xs:element name="C++" value="53" />
  <xs:element name="Cobol" value="107" />
  <xs:element name="Delphi_5" value="18" />
  <xs:element name="HTML_4" value="14" />
  <xs:element name="Visual_Basic_6" value="24" />
  <xs:element name="SQL" value="13" />
  <xs:element name="Java" value="46" />
    </xs:element>
    </xs:element>
    </xs:element>
- <xs:element name="Projeto2">
- <xs:element name="Severidade_Revisao">
  <xs:element name="A" value="Alta" />
  <xs:element name="M" value="Media" />
  <xs:element name="B" value="Baixa" />
    </xs:element>
- <xs:element name="Severidade_Teste">
  <xs:element name="1" value="Alta" />
  <xs:element name="2" value="Media" />
  <xs:element name="3" value="Baixa" />
    </xs:element>
- <xs:element name="Tamanho_Total">
- <xs:element name="Linguagem">
  <xs:element name="C++" value="53" />
  <xs:element name="Cobol" value="107" />
  <xs:element name="Delphi_5" value="18" />
  <xs:element name="HTML_4" value="14" />
  <xs:element name="Visual_Basic_6" value="24" />
  <xs:element name="SQL" value="13" />
  <xs:element name="Java" value="46" />
    </xs:element>
    </xs:element>
    </xs:element>
- <xs:element name="Projeto3">
```

```
- <xs:element name="Severidade_Revisao">
  <xs:element name="H" value="Alta" />
  <xs:element name="M" value="Media" />
  <xs:element name="L" value="Baixa" />
    </xs:element>
- <xs:element name="Severidade_Teste">
  <xs:element name="1" value="Baixa" />
  <xs:element name="2" value="Media" />
  <xs:element name="3" value="Media" />
  <xs:element name="4" value="Alta" />
    </xs:element>
- <xs:element name="Tamanho_Total">
- <xs:element name="Linguagem">
  <xs:element name="C++" value="53" />
  <xs:element name="Cobol" value="107" />
  <xs:element name="Delphi_5" value="18" />
  <xs:element name="HTML_4" value="14" />
  <xs:element name="Visual_Basic_6" value="24" />
  <xs:element name="SQL" value="13" />
  <xs:element name="Java" value="46" />
    </xs:element>
    </xs:element>
    </xs:element>
    </xs:element>
    </xs:schema>
```

ANEXO C

Estrutura do DSA

1. TABELA DSA_RELEASE

```
create table DSA_Release (

     ID_Release          varchar(50),
     CustoBO             decimal(10),
     CustoBR             decimal(10),
     CustoReal           decimal(10),
     NomeFase            varchar(50),
     EsforcoBO           decimal(10),
     EsforcoBR           decimal(10),
     EsforcoReal         decimal(10),
     DataInicialBO       date,
     DataInicialBR       date,
     DataInicialReal     date,
     DataFinalBO         date,
     DataFinalBR         date,
     DataFinalReal       date,
     WorkComplete        decimal(10),

     constraint PK_DSA_Release primary key (ID_Release)

)
```

2. TABELA DSA_ITERAÇÃO

```
create table DSA_Iteracao (

     ID_Iteracao         varchar(50),
     ID_Release          varchar(50),
     CustoBO             decimal(10),
     CustoBR             decimal(10),
     CustoReal           decimal(10),
     NomeFase            varchar(50),
     EsforcoBO           decimal(10),
     EsforcoBR           decimal(10),
     EsforcoReal         decimal(10),
     DataInicialBO       date,
     DataInicialBR       date,
     DataInicialReal     date,
     DataFinalBO         date,
     DataFinalBR         date,
     DataFinalReal       date,
     WorkComplete        decimal(10),

     constraint PK_DSA_Iteracao primary key (ID_Iteracao),
     constraint FK_DSA_Release foreign key (ID_Release)
          references DSA_Release

)
```

3. TABELA DSA_FASE

```
create table DSA_Fase (

     ID_Fase             varchar(50),
```

```
     ID_Iteracao         varchar(50),
     ID_Release          varchar(50),
     CustoBO             decimal(10),
     CustoBR             decimal(10),
     CustoReal           decimal(10),
     NomeFase            varchar(50),
     EsforcoBO           decimal(10),
     EsforcoBR           decimal(10),
     EsforcoReal         decimal(10),
     DataInicialBO       date,
     DataInicialBR       date,
     DataInicialReal     date,
     DataFinalBO         date,
     DataFinalBR         date,
     DataFinalReal       date,
     WorkComplete        decimal(10),

     constraint PK_DSA_Fase primary key (ID_Fase),
     constraint FK_DSA_Iteracao foreign key (ID_Iteracao)
          references DSA_Iteracao,
     constraint FK_DSA_Release foreign key (ID_Release)
          references DSA_Release

)
```

4. TABELA DSA_ATIVIDADE

```
create table DSA_Atividade (

     ID_Atividade        varchar(50),
     ID_Fase             varchar(50),
     ID_Iteracao         varchar(50),
     ID_Release          varchar(50),
     CustoBO             decimal(10),
     CustoBR             decimal(10),
     CustoReal           decimal(10),
     NomeAtividade       varchar(50),
     TipoAtividade       varchar(50),
     EsforcoBO           decimal(10),
     EsforcoBR           decimal(10),
     EsforcoReal         decimal(10),
     DataInicialBO       date,
     DataInicialBR       date,
     DataInicialReal     date,
     DataFinalBO         date,
     DataFinalBR         date,
     DataFinalReal       date,
     WorkComplete        decimal(10),
     UniqueID            decimal(10),


     constraint PK_DSA_Atividade primary key (ID_Atividade),
     constraint FK_DSA_Fase foreign key (ID_Fase)
          references DSA_Fase,
     constraint FK_DSA_Iteracao foreign key (ID_Iteracao)
          references DSA_Iteracao,
```

```
     constraint FK_DSA_Release foreign key (ID_Release)
          references DSA_Release

)
```

5. TABELA DSA_DEFEITO

```
create table DSA_Defeito (

ID_Defeito              varchar(50),
     ID_Fase            varchar(50),
     ID_Iteracao        varchar(50),
     ID_Release         varchar(50),
     Categoria          varchar(50),
     Tipo               varchar(50),
     Severidade         varchar(50),

     constraint PK_DSA_Defeito primary key (ID_Defeito),
     constraint FK_DSA_Fase foreign key (ID_Fase)
          references DSA_Fase,
     constraint FK_DSA_Iteracao foreign key (ID_Iteracao)
          references DSA_Iteracao,
     constraint FK_DSA_Release foreign key (ID_Release)
          references DSA_Release

)
```