UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE TECNOLOGIA E GEOCIÊNCIAS

PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Laura Jane Takahashi Monteiro

Uma Arquitetura de Integração de Metadados Governamentais - Abordagem baseada em CWM

RECIFE

Março de 2004

Laura Jane Takahashi Monteiro

Uma Arquitetura de Integração de Metadados Governamentais

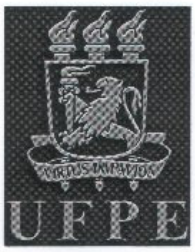
- Abordagem baseada em CWM

Dissertação apresentada como requisito parcial à obtenção do grau de mestre. Programa de Pós-Graduação em Engenharia Elétrica do Centro de Tecnologia e Geociências/Escola de Engenharia de Pernambuco, Universidade Federal de Pernambuco.

Orientador: Prof. Dr. Rafael Dueire Lins.

RECIFE

Março de 2004



Universidade Federal de Pernambuco

Pós-Graduação em Engenharia Elétrica

PARECER DA COMISSÃO EXAMINADORA DE DEFESA DE DISSERTAÇÃO DE MESTRADO PROFISSIONALIZANTE DE

Laura Jane Takahashi Monteiro

TÍTULO

"UMA ARQUITETURA DE INTEGRAÇÃO DE METADADOS GOVERNAMENTAIS – ABORDAGEM BASEADA EM CWM"

A comissão examinadora composta pelos professores: RAFAEL DUEIRE LINS, DES/UFPE, FERNANDA MARIA RIBEIRO DE ALENCAR, DES/UFPE E RICARDO MASSA FERREIRA LIMA, Escola Politécnica/UPE sob a presidência do primeiro, consideram a candidata **LAURA JANE TAKAHASHI MONTEIRO**

APROVADA

Recife, 02 de março de 2004.

RAFAEL DUEIRE LINS

FERNANDA MARIA RIBEIRO DE ALENCAR

RICARDO MASSA FERREIRA LIMA

Agradecimentos

A Deus que ilumina o meu caminho e me concede a luz da sabedoria.

Ao meu tesouro que é a minha família amada. Ao meu pai pelas longas conversas e pelos vários exemplos de vida. À minha mãe pelas palavras carinhosas e por sempre procurar nos mostrar o poder do grande amor de Deus, principalmente nas horas mais difíceis. À minha avó pela garra, determinação e grande ternura e delicadeza. E a meu irmão pelo forte incentivo e pelas palavras certas na hora do sufoco.

Aos amigos Alexandre Guedes, Jorge Abílio, Guilherme Moraes e Jorlene Marques pela companhia e animação nas incontáveis horas de estudo.

Às amigas Luíza Yanae, Andréa Mendonça e Vivian Lane pelas demonstrações de amizade e pelo grande apoio nos momentos mais difíceis.

Aos amigos José Pinheiro e Eudo de Lima por todas as tentativas de me mostrar a realidade da fase de escrita da dissertação, fazendo com que eu corresse cada vez mais rápido rumo ao produto final.

Aos amigos Marco Domingues, Jeisa e Cláudia pelos braços e coração abertos na calorosa recepção em Recife.

Ao prof. Rafael Dueire Lins pela coragem e ousadia ao encarar esse mestrado, e principalmente pela confiança depositada na fase de orientação desta dissertação.

À Secretaria de Economia e Finanças (SEMEF) por ter me proporcionado a materialização do mestrado.

Sumário

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

API – *Application Program Interface*

COM – *Component Object Model*

CORBA - *Common Object Request Broker Architecture*

CTI – Centro de Tecnologia da Informação

CWM – *Common Warehouse Metamodel*

CWMX – *CWM Extension*

DBA - *Database Administrator*

DTD - *Document Type Definition*

EJB – *Enterprise JavaBeans*

ER – Entidade-Relacionamento

ETL – *Extract, Transform and Load*

HTML - *HyperText Markup Language*

IDL – *Interface Definition Language*

J2EE - *Java 2 Platform, Enterprise Edition*

JCP - *Java Community Process*

JDBC - *Java Database Connectivity*

JDMAPI - *Java Data Mining API*

JMI - *Java Metadata Interface*

JOLAP - *Java OLAP*

LDO – Lei de Diretrizes Orçamentárias

LOA – Lei Orçamentária Anual

LRF – Lei de Responsabilidade Fiscal

MDA - *Model Driven Architecture*

MDC - *Meta Data Coalition*

MDIS - *Metadata Interchange Specification*

MDR - *Meta Data Repository*

MOF – *Metadata Object Facility*

ODBC - *Open Database Connectivity*

OCL - *Object Constraint Language*

OLAP - *Online Analytical Processing*

OMG – *Object Management Group*

PMM – Prefeitura Municipal de Manaus

PPA – Plano Plurianual

PRODAM – *Processamento de Dados Amazonas SA*

RPC – *Remote Procedure Call*

SAI – Sistema Administrativo Integrado

SEMEF – Secretaria Municipal de Economia e Finanças

SGBD – Sistema de Gerenciamento de Banco de Dados

SOAP – *Simple Object Access Protocol*

SQL - *Structured Query Language*

TI – Tecnologia da Informação

UML – *Unified Modeling Language*

W3C - *World Wide Web Consortium*

WSDL – *Web Services Definition Language*

XMI – *XML Metadata Interface*

XML – *Extensible Markup Language*

XSD - *Extensible Stylesheet Definition*

RESUMO

Cada aplicação utiliza diferentes estruturas de programação, sintaxe, e semântica para modelar seus metadados, gerando grandes incompatibilidades quando há a necessidade da interação com outras aplicações. O *Object Management Group* (OMG) vem trabalhando no estabelecimento do *Common Warehouse Metamodel* (CWM) como padrão para a integração de metadados nos domínios de *data warehouse* e *business intelligence*. Nesta dissertação primeiramente foram abordados os aspectos gerais dos metadados e de suas arquiteturas de integração, para em seguida se pesquisar a aplicação dos metamodelos que compõem o *Common Warehouse Metamodel* e suas tecnologias de fundamentação (UML, MOF, XMI). Desenvolveu-se a modelagem de um aspecto referente ao orçamento público existente em todas as instâncias governamentais utilizando-se alguns dos metamodelos CWM, como forma de se demonstrar a facilidade na troca de metadados entre os diversos recursos de dados.

Palavras-chave: Metadados, integração, *data warehouse*, CWM, UML.

ABSTRACT

Each application uses different programming, syntax, and semantic structures to model their metadata, generating big incompatibilities when there is necessity of interaction with other applications. The Object Management Group (OMG) is working at the establishment of the Common Warehouse Metamodel (CWM) as a standard for the metadata integration in the data warehouse and business intelligence domains. In this dissertation first was discussed the general aspects of metadata and its integration architectures, next was researched the application of the metamodels that belongs to the Common Warehouse Metamodel and its base technologies (UML, MOF, XMI). It was developed the modeling of an aspect concerning to the existing public budget at all the governmental instances using some of the CWM metamodels, as a manner of demonstrate the easiness of metadata exchange between the diverse data resources.

Keywords: Metadata, integration, data warehouse, CWM, UML.

1 INTRODUÇÃO

As empresas de hoje em dia possuem uma grande diversidade de sistemas computacionais para os mais variados fins, onde alguns chegam a ser utilizados como ferramentas de auxílio na tomada de decisões. Alguns desses sistemas encontram-se em meios legados, como *mainframes*, e outros, mais atuais, são desenvolvidos para micro. Também chegando a existir aqueles desenvolvidos para a *web*.

Além da diversidade já citada, o avanço da tecnologia vem a tornar tais sistemas cada vez mais distribuídos e complexos, dificultando, dessa forma, seu gerenciamento e o gerenciamento dos dados sobre os quais cada um trabalha.

Com a existência de tantos sistemas diferentes os dados vão ficando cada vez mais espalhados na empresa, o que pode levar à duplicidade de dados ou até mesmo a incompatibilidades para dados gerados em sistemas diferentes, mas que precisam ser trocados entre si. Os dados e os metadados[1] encontram-se espalhados em diferentes sistemas, levando à necessidade de se encontrar uma forma de coordenar e gerenciar esses metadados dispersos.

Em (POOLE, 2002) percebe-se que uma abordagem mais simplista seria a de que todos os fornecedores utilizassem as mesmas semânticas, paradigmas, etc. e colecionassem todo o metadado em um formato único e em uma localização única. Mas tal abordagem torna-se difícil de ser aplicada uma vez que as empresas possuem múltiplas plataformas de negócios, não permitindo a tão sonhada localização única da coleção de metadados.

---

[1] Um metadado é um dado sobre outro dado, mas que descreve apenas algumas características essenciais do dado. Para assegurar que o metadado é corretamente formado deve-se formulá-lo através de determinadas regras, o que levará os sistemas a interpretarem corretamente o metadado ao executarem qualquer operação no dado correspondente.

Através desta dissertação pretende-se, portanto, estudar a composição e formação dos metadados de um *data warehouse* e sua integração através do estudo e aplicação prática de uma abordagem de integração de metadados baseada em modelos. Para isto será utilizada a tecnologia denominada *Common Warehouse Metadata* (CWM), a qual é fundamentada em padrões como a *Unified Modeling Language* (UML) e a *Extensible Markup Language* (XML), ambas do *Object Management Group* (OMG).

1.1 Motivação

A falta de uma abordagem integrada para o gerenciamento de metadados pode facilmente levar à manipulação de informações incorretas, gerando metadados de baixa qualidade, e ocasionando, assim, situações problemáticas tais como perda de produtividade e dificuldade de atualização e do crescimento dos metadados.

O maior problema para a integração bem sucedida é a falta de padronização para a definição de metadados. De acordo com (MOSHER, 2002) cada aplicação utiliza diferentes estruturas de programação, sintaxe, e semântica para modelar seus metadados, gerando grandes incompatibilidades, as quais são justamente o maior empecilho para que uma aplicação consiga descobrir e interagir com os dados mantidos por outra aplicação. Percebe-se então a necessidade do estabelecimento de um padrão de metadados e de um mecanismo de troca destes metadados de aplicação para aplicação.

Na arquitetura de integração de metadados baseada em modelo a idéia básica é a de se utilizar um modelo padrão para que se possa definir o metadado. Para que tal modelo seja independente de plataforma, o mesmo deve ser expresso em uma linguagem formal, como a UML.

Segundo (POOLE, 2002): “nesta abordagem os metadados passam a poder existir fora e independentemente de qualquer plataforma específica ao serem expressos como um modelo formal e independente”.

Um produto que utilize tal modelo formal como base de seus próprios metadados pode executar um processo para mapeamento de importação a fim de traduzir este modelo formal na instância de seus próprios metadados, os quais são específicos do produto. E também pode executar um processo de mapeamento de exportação para transformar seus metadados internos em um modelo formal, independente de plataforma, e, desta maneira, exportá-los para outros produtos.

Ainda segundo (POOLE, 2002) pode-se descrever esta abordagem de integração e interoperabilidade em nível de metadados como sendo uma abordagem baseada em modelo para a integração de metadados ou ainda como uma arquitetura para metadado dirigida a modelo. Cada produto consegue mapear seu metadado compartilhado para a sua própria representação interna de metadados, e isto requer que o metamodelo[2] comum, o qual define o domínio como um todo, seja uma representação o mais completa possível, para que possa oferecer algum tipo de extensão.

A solução baseada em modelo deve definir uma interface comum que suporte o intercâmbio e o acesso aos metadados compartilhados. Uma das linguagens atualmente utilizadas para a troca de dados na *web* é a XML, podendo também ser utilizada em ambientes distribuídos, assíncronos e fracamente acoplados. Em um documento XML os elementos podem ser definidos através de *tags* configuráveis, as quais podem representar

---

[2] Um metamodelo é o modelo de um modelo. É uma definição precisa das regras e construções necessárias para a criação de modelos semânticos, e que, de acordo com (TANNENBAUM, 2002a), pode ser organizado por qualquer uma das seguintes perspectivas: pela origem, pela classificação ou pelo uso dos metadados.

conceitos definidos pelo metamodelo. O conteúdo do elemento passa a ser metadado, e as *tags* que definem os tipos possíveis de elementos passam a ser elementos do metamodelo.

Uma grande vantagem da interface comum é simplificar a construção de programas que devam acessar o metadado a partir de determinado produto, ferramenta, banco de dados ou um outro serviço qualquer. Os programas passam a ser codificados através de apenas um tipo de interface, não importando qual provedor de metadados implemente esta interface.

A arquitetura de integração de metadados baseada em modelos foi o fator motivador para a criação de um padrão para o intercâmbio de metadados, sendo esta uma solução baseada em padrões. *Common Warehouse Metamodel* (CWM) é a solução proposta, a qual é dirigida a modelo e, segundo (POOLE, 2002) "pode ser adotada tanto em ambientes de metadados ponto a ponto quanto para os baseados em repositórios". A adoção desta padronização pelos fornecedores permitirá o intercâmbio de metadados entre diversas ferramentas e de diversos fornecedores.

O CWM é uma padronização estabelecida em conjunto com a OMG e grandes fabricantes, tais como a IBM, Oracle e Unisys. Uma das vantagens de se estabelecer um padrão através do CWM é que o mesmo é baseado na utilização de dois outros importantes padrões da indústria: a *Unified Modeling Language* (UML, 2003) e a *Extensible Markup Language* (XML, 2003). Uma das conseqüências da adoção destes dois padrões é a de que o CWM pode levar a soluções de baixo custo e reutilizáveis.

O CWM, como pode ser visto na Figura 1, compõe-se de vários metamodelos diferentes, porém fortemente relacionados. E cada metamodelo corresponde a uma área funcional importante de uma corrente de fornecimento de informações.

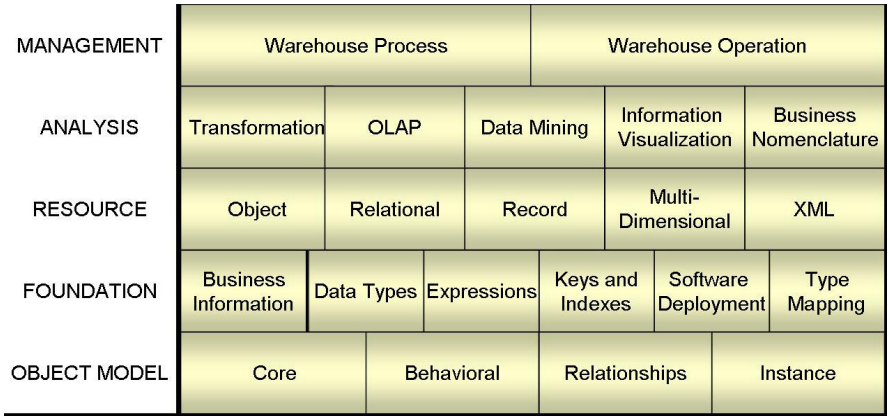


Figura 1 - Metamodelo CWM

Fonte: OMG, 2002a, pág. 58.

O CWM é expresso em UML, de onde também pode ser estendido um subconjunto com a finalidade de se incluir os conceitos de análises de negócios e de *data warehouse*. CWM utiliza a UML para definir as estruturas e os relacionamentos de metadados complexos.

O *Meta Object Facility* (MOF[3]) da OMG (MOF, 2003) disponibiliza uma linguagem abstrata utilizada para assegurar que todas as ferramentas irão interpretar o metamodelo comum da mesma forma. MOF serve de modelo comum para o CWM e para a UML. Sua semântica permite definir determinados serviços de repositórios de metadados que suportam operações tais como construção e atualização do modelo, por exemplo.

A especificação MOF permite o mapeamento de qualquer um de seus metamodelos, como o CWM por exemplo, para a IDL (notação de linguagem neutra para a especificação de interfaces) da OMG. Esse mapeamento define uma especificação de interface para a descoberta e gerenciamento de modelos, através de APIs programáticas (POOLE, 2002). Todos os modelos CWM são expressos em UML e implementam semântica MOF.

---

[3] MOF define um linguagem abstrata comum que pode ser utilizada para a especificação de metamodelos. Através desta define-se os elementos essenciais, sintaxe e estrutura de metamodelos, os quais são usados para construir modelos de sistemas, segundo (POOLE, 2002).

A abordagem para integração de metadado baseada em modelo preconiza um formato comum de intercâmbio para a troca de instâncias dos metadados compartilhados, e também uma interface comum para o acesso ao metadado, o que pode ser feito utilizando-se a XML. A XMI é a codificação de intercâmbio XML utilizada pelo CWM. XMI define o mapeamento formal de metamodelos MOF, definindo a forma como as *tags* XML podem ser utilizadas para armazenar instâncias de metamodelos CWM em documentos XML. Através do CWM pode-se definir uma coleção de *tags* XML, as quais são expressas na forma de um *Document Type Definition* (DTD) XML. Os metadados do CWM passam a ser então serializados em documentos XML, e cada instância do metadado é armazenada como um conteúdo XML, delimitado por *tags* apropriadas ao metamodelo.

Uma das vantagens da XMI é que tanto a *tag* quanto os itens descritos pela *tag* podem ser empacotados juntos no mesmo documento, e assim as aplicações podem entender rapidamente o conteúdo do documento.

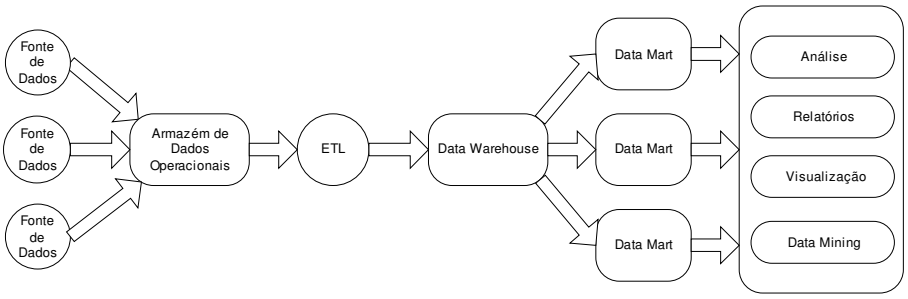


Figura 2 - A cadeia de fornecimento de informação

Fonte: POOLE, 2002, pág. 12.

Se os componentes de uma cadeia de fornecimento de informação (exemplificada na Figura 2), tais como ferramentas, aplicações e banco de dados, basearem-se no metamodelo CWM, então todos serão capazes de entender os mesmos modelos e metadados baseados em CWM.

1.2 Objetivo

Através desta dissertação pretende-se analisar o funcionamento e a utilização do CWM como ferramenta de padronização para a integração de metadados de *data warehouses*. Descrever seu funcionamento e comparar o CWM com outras tecnologias de integração de metadados. Propor uma arquitetura de integração de metadados governamentais através da modelagem de um determinado aspecto do orçamento público, utilizando-se o ambiente da Prefeitura Municipal de Manaus (PMM) como estudo de caso. Para tal, deverá ser estudado material referente ao orçamento público. Além disso, pretende-se também avaliar a real importância do CWM como ferramenta de padronização para integração de metadados governamentais.

1.3 Estrutura desta Dissertação

A presente dissertação é formada por seis capítulos, sendo o primeiro deles esta introdução. No Capítulo 2 é apresentado um estudo sobre as principais arquiteturas de integração de metadados encontradas na literatura. O Capítulo 3 aborda o modelo CWM, um metamodelo para *data warehouses*. No Capítulo 4 há a descrição do ambiente da Prefeitura Municipal de Manaus, o qual é utilizado como estudo de caso neste trabalho. No Capitulo 5 é proposta uma arquitetura para integração de metadados governamentais através da modelagem de uma questão orçamentária, tendo como alvo os dados orçamentários da PMM. E no Capítulo 6 são destacadas as contribuições principais desta dissertação e feitas sugestões de trabalhos futuros. Em seguida serão apresentadas as referências bibliográficas.

2 ARQUITETURA DE INTEGRAÇÃO DE METADADOS DE *DATA WAREHOUSES*

Os sistemas computacionais vêm sendo cada vez mais utilizados pelas empresas com a finalidade de auxiliar no seu gerenciamento e nas tomadas de decisões, entre outras atividades. Tais sistemas estão espalhados pela empresa e suas filiais, e podem ser desenvolvidos numa infinidade de linguagens, para as mais diversas plataformas e acessando os mais diferentes bancos de dados.

Uma empresa pode possuir sistemas desenvolvidos para plataformas avançadas e também possuir sistemas legados, implementados para funcionamento em plataformas mais antigas, não necessariamente sendo relacionais. Em algumas das vezes há a necessidade de se comparar dados de sistemas legados com dados de sistemas atuais para se tentar extrair algum padrão de comportamento que venha a auxiliar na tomada de decisão.

Atualmente nas empresas há tanta informação que a maioria das pessoas não consegue separar o que é bom do que é irrelevante. As informações existem em abundância, porém sem uma organização pré-determinada, e às vezes podem até existir em duplicidade nos mais diversos setores.

Essas informações coletadas e processadas geralmente tornam-se conhecimento. A Figura 3 mostra a pirâmide do conhecimento, chave para o gerenciamento do conhecimento. Conforme os sistemas vão amadurecendo, eles progridem da coleta e gerenciamento de dados para a coleta e gerenciamento de conhecimento.

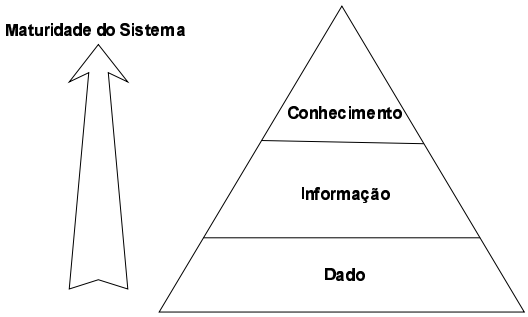


Figura 3 - Pirâmide do Conhecimento

Fonte: MARCO, 2000, pág. 317.

O dado, sozinho, tem pouco significado e propósito. Já a informação é dado que possui significado e propósito, mostrando a realidade de uma empresa e o funcionamento da mesma. O conhecimento está um passo a mais da informação, levando a entender o impacto que um negócio causa no mercado, a perceber a interação de uma empresa em relação a outras do mesmo ramo, e a entender como uma empresa é influenciada pelo mercado. Nesse contexto o papel dos metadados é o de tornar possível a tarefa de juntar de forma significativa os dados, a informação e o conhecimento. (MARCO, 2000)

Segundo (JEFFERY, 1998) talvez uma das primeiras aplicações de metadados foi em sistemas computadorizados para catálogos de bibliotecas baseados em técnicas de recuperação de informações, onde o registro do cartão do catálogo é um metadado descrevendo o dado real de um livro ou de outra publicação.

2.1 O Papel da Informação

A informação sem o metadado é útil somente para aqueles que entendem ou participaram de sua criação. O metadado tem a capacidade de levar à informação, além de servir para esclarecimento e definição. Mesmo que exista informação com as regras de

negócios e as fontes bem documentadas, será de pouca utilidade caso não esteja facilmente identificável, e acessível através de seu metadado.

Apesar de não ser uma regra completamente incorporada pela TI, é desejável que a informação seja sempre derivada a partir dos dados. A informação é a intersecção de dados e processos. As teias de informações são formadas a partir do momento em que se toma a informação e esta passa a ser revisada a fim de suportar processos totalmente diferentes ao de sua criação. Isto leva à dificuldade na identificação das fontes de várias informações, e cada uma dessas informações começará a perder as mesmas características que anteriormente as tornaram informação. (TANNENBAUM, 2002a)

2.2 Integrando os Dados

Inicialmente, os modelos de dados eram criados com o objetivo de refletir, de forma gráfica, os requisitos do negócio. Os padrões de modelagem de dados ditavam o formato desses diagramas. Em geral modelavam coisas (entidades) sobre as quais necessitava-se colecionar dados, os inter-relacionamentos entre as entidades, e também os itens de dados (atributos) que se necessitava colecionar. Tais modelos eram criados para novas aplicações e geralmente resultavam em um novo banco de dados para cada aplicação.

Um dos benefícios da modelagem para o ambiente relacional é o de fornecer uma perspectiva do funcionamento do negócio antes da sua tradução para o banco de dados. Os modelos lógicos oferecem às equipes de projetos um lugar para trabalhar metadados e dados sem a preocupação com as restrições de tecnologia física. Uma outra vantagem da modelagem lógica é a reutilização de dados e metadados previamente documentados.

Criar as definições para entidades, atributos e relacionamentos de forma padronizada, que sirva para todos os modelos da empresa, pode ser bastante vantajoso para a empresa.

Porém, esse processo de padronização envolve a resolução de conflitos entre modelos padronizados e o modelo lógico sob revisão. Este processo geralmente é político. Deve-se visar alcançar compromissos de tal forma que, por exemplo, uma entidade possa ser chamada "cliente de varejo" ou "cliente de atacado" – algo mais descritivo do que "cliente" com 20 definições, dependentes do modelo em questão.

Para que se consiga alcançar essa abordagem organizada, é necessário pessoal dedicado nesta tarefa de organização de gerenciamento de dado. A organização de gerenciamento de dado tem a ampla missão de analisar e desenvolver o modelo de negócios como um projeto para o futuro desenvolvimento do banco de dados. O modelo de negócios requer manutenção constante já que novas decisões de negócios são tomadas de forma a afetar os dados corporativos.

2.3 Data Warehouses e Metadados

O *data warehouse* caracteriza-se como ferramenta de apoio para suporte à decisão[4]. Projetos de *data warehouse* oferecem um grande retorno de investimento quando integrados com sucesso, sendo uma das áreas de maior desafio para a integração de dados dispersos, os quais normalmente se encontram em vários sistemas distribuídos pela empresa. Os *data warehouses* são projetados para a execução de consultas complexas e *ad hoc* com tempo de resposta otimizado.

No contexto de *data warehouse* os metadados respondem a questões sobre: dicionário de dados, fluxo de dados, transformação de dados, controle de versão, estatísticas de utilização de dados, informação do alias e segurança (JARKE, 2000). E, de acordo com

[4] O *data warehouse* permite juntar, estruturar, manipular, armazenar, acessar, apresentar e distribuir as informações.

(STAUDT, 1999), os metadados podem ser utilizados por outros componentes do *data warehouse* ou até mesmo diretamente pelas pessoas para que possam executar suas tarefas de forma bem sucedida.

O significado dos dados está nos Metadados, os quais são utilizados para descrever como a informação é estruturada (a sintaxe) e o seu significado (a semântica). Para os desenvolvedores o metadado é o fundamento de uma documentação consistente e integrada para o sistema de *data warehouse*.

Os metadados são utilizados por ferramentas, bancos de dados, aplicações e outros serviços de informação para definir a estrutura e o significado de seus objetos, serviços, e outros artefatos computacionais.

A maioria das aplicações define seus metadados de forma diferente, e para a modelagem destes cada uma pode vir a utilizar estruturas diferentes de programação, sintaxe, e semântica. Este esforço custoso e contínuo é o resultado da falta de um metadado comum – ou mais especificamente, um modelo comum para a criação de metadado (um metamodelo).

Existem vários metadados que podem ser extraídos das fases de um *data warehouse*, como pode ser visto em (COME, 1999). Durante a fase de *captura de dados* pode-se extrair metadados relativos às fontes de onde os dados são extraídos. Na fase de *integração de dados* pode haver a necessidade do mapeamento do novo dado até a sua fonte, então podem ser gerados metadados sobre as transformações realizadas e até sobre o próprio dado. A fase de *manipulação de dados* pode disponibilizar metadados com o histórico do que vem acontecendo com os dados, incluindo informações sobre as funções de agregação e desagregação. E ainda na fase de *gerência do data warehouse*, onde ocorrem monitoramentos de processos, podem ser gerados, por exemplo, metadados para uso administrativo, definindo o acesso de usuários aos dados.

Os metadados podem ser classificados em dois tipos:

- *Técnicos* – utilizados pelos administradores do banco de dados e pelos desenvolvedores de aplicações. São aqueles metadados críticos para a manutenção e o crescimento contínuo do *data warehouse*. Exemplos clássicos são os dicionários de dados dos fontes, *data warehouse*, *data marts*, e o código das regras de transformações.
- *Negócios* – ligação entre o *data warehouse* e os usuários de negócios. Fornecem um mapa para que os usuários possam acessar os dados. Mostram quais relatórios, consultas e dados estão no *data warehouse*, a localização dos dados, confiabilidade dos dados, o contexto dos dados, as regras de transformação que foram aplicadas e quais as origens desses dados. Os metadados de negócios (ou medatados de processos) são utilizados para capturar o conteúdo semântico sobre coisas relativas aos negócios, tais como regras, nomenclatura e terminologia.

O metadado é produzido, acessado e atualizado tanto por pessoas quanto por ferramentas de software. Os usuários finais e os usuários técnicos podem entrar com os metadados a serem armazenados, e utilizá-los posteriormente. Ferramentas de software ou alguns componentes do software também podem ser produtores de metadados, descrevendo dados e informações manuseadas por esses componentes. (STAUDT, 1999)

Vantagens da Padronização:

- padrões minimizam os esforços que cada Gerente de Projeto deve exercer para criar seu próprio material de trabalho.
- gerência passa a confiar na habilidade do Gerente de Projeto em entregar um produto com alta qualidade em um ambiente controlado.

Os riscos ocasionados pela falta de uma abordagem integrada para gerenciar, consolidar e manter os metadados atualizados são apontados por (TANNENBAUM, 2002a)

como sendo: informações incorretas, perda de qualidade dos metadados, perda de produtividade, desempenho inconstante e dificuldade para atualização e crescimento.

O intercâmbio de metadados era considerado um dos problemas mais críticos da indústria do *data warehouse*. Visando a eliminação de tal problema foi criado o grupo *Meta Data Coalition* (MDC), o qual é um grupo aberto a vendedores e usuários finais para a definição de padrões de metadados. Seus objetivos são criar mecanismo de acesso padrões e interfaces de programação padrões para metadados, permitir aos usuários controlar e gerenciar o acesso e manipulação de metadados em seus ambientes através da utilização de ferramentas concordantes com a especificação de intercâmbio e definir uma infraestrutura simples de implementação de intercâmbio que irá facilitar a conformidade minimizando a quantidade de modificação requerida para as ferramentas existentes (JARKE, 2000).

Tal grupo desenvolveu o *Metadata Interchange Specification* (MDIS), o qual é uma especificação de intercâmbio de metadados baseada em caracteres, sendo independente de plataforma. Uma das mais importantes características é a capacidade de extensão da especificação, porque o conteúdo do que é considerado metadado será desenvolvido (JARKE, 2000). Aborda assuntos referentes a troca, compartilhamento e gerenciamento de metadados. Define um conjunto mínimo de elementos de metadados e pontos mínimos de integração a serem incorporados em bancos de dados para compatibilidade.

O MDIS teve pouco suporte da indústria porque a cobertura dos tipos de metadados era limitada e também era limitado o leque de formatos abertos de intercâmbio.

O OMG vem trabalhando desde 1995 nos padrões de modelagem e metadados, que são os seguintes:

- *UML (Unified Modeling Language)* – 1997 (UML, 2003) – Oferece uma linguagem para a modelagem dos metadados.

- *MOF (Meta Object Facility)* – 1997 (MOF, 2003) – Oferece as APIs para a manipulação de Metadados.
- *XMI (XML Metadata Interchange)* – 1999 (XMI, 2003) – Fornece os mecanismos para o intercâmbio de metadados em XML.
- *CWM (Common Warehouse Metamodel)* – 2000 (CWM, 2002) – Utiliza UML, MOF e XMI para modelar, manipular e realizar o intercâmbio dos metadados do *data warehouse*, incluindo os metadados técnicos e de negócios.

2.3.1 Integração de Metadados

Cada ferramenta deve ter um entendimento da natureza do dado que ela esteja para consumir: de onde veio; o que significam seus vários campos; que transformações devem ser feitas nos dados; onde os resultados necessitam ser armazenados; ou que processos alvos requerem tais resultados; e daí em diante.

Para que um dado produto opere corretamente em seus dados, deve ter um entendimento completo da estrutura e do significado semântico de seus dados. Para que uma dada coleção de produtos de software participe efetivamente de uma cadeia de fornecimento de informações e interopere no nível de dados, eles devem ter um entendimento comum dos metadados que descrevem os dados. Cada um dos produtos de software e ferramentas que compreendem uma cadeia de informações deve ser integrado no nível de metadado antes que eles possam ser efetivamente integrados no nível de dados.

Integração de metadado também que lidar com o aspecto da heterogeneidade, como o caso de sistemas fontes substancialmente diferentes na forma de fornecer o metadado (DO, 2000). A maioria dos produtos comerciais armazena metadados em formatos amplamente diversificados. O fato de o metadado estar prontamente acessível não necessariamente

significa que ele seja universalmente entendível. A forma e semântica do metadado, como as interfaces que oferecem acesso a elas, são raramente uniformes entre os produtos. Geralmente o metadado é mantido independentemente de uma forma grandemente isolada e em formatos de representação específicos.

Geralmente, produtos de software e ferramentas são projetados e construídos em relativo isolamento um do outro. Os fabricantes têm estado relutantes em padronizar seus metadados com medo de perder mercado para os competidores.

Ferramentas com metadados diferentes são integradas através de complexas pontes de metadados. Cada ponte é equivalente a um pedaço de software capaz de traduzir metadados de um produto em um formato requerido por outro produto, oferecendo uma alta funcionalidade e compartilhamento de metadados.

2.3.2 Arquitetura de Metadados Ponto a Ponto

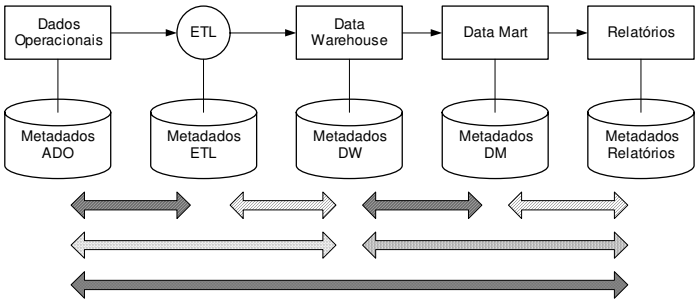


Figura 4 – Arquitetura de integração de metadados ponto-a-ponto

Fonte: POOLE, 2002, pág. 18.

Na arquitetura de integração de metadados ponto-a-ponto (ilustrada na Figura 4) é necessário que haja uma ponte em separado para cada par único de ferramentas sendo integradas (POOLE, 2002). Na figura cada ponte é identificada por uma seta, e geralmente precisa ser bidirecional, capaz de entender os mapeamentos de metadados em cada direção.

Cada mapeamento não necessariamente é inverso ao outro, a ferramenta pode ser consumidora e também publicar metadados ou ser somente consumidora, por exemplo. Normalmente ocorre uma certa quantidade de perda de informação ao traduzir de uma forma de metadado para outra, e a ponte de certa forma precisa considerar isso.

A construção de pontes é um processo muito difícil e caro. As pontes devem ter um conhecimento detalhado de modelos e interfaces de metadados proprietários, e também conhecimento de como os mapeamentos de diferentes modelos para outros devem ser inserido na ponte. A lógica de processamento que compõe uma determinada ponte não necessariamente é reutilizável na construção de outras pontes.

Manter essas pontes consome muitos recursos, principalmente se houver o crescimento no número de pontes. E geralmente esta estratégia requer que o usuário abandone as abordagens melhores e se sujeite aos produtos de um único fabricante. (IKEMATU, 2001)

Apesar desse custo na construção de pontes entre as ferramentas, a integração de metadados é um pré-requisito para a integração de dados efetiva. A cadeia de informações deve ser integrada totalmente, e não somente conexões entre ferramentas que compõem passos adjacentes ou relacionados na corrente. Uma descrição unificada de todo dado fluindo através da cadeia precisa definir os relacionamentos e as transformações entre os elementos de dados e traçar sua linhagem através do processo de refinamento. Este metadado global deve estar disponibilizado para, e deve ser prontamente entendido por, todos os componentes de software que compõem a cadeia de fornecimento de informações.

2.3.3 Arquitetura de Metadado Hub-and-spoke

Essa necessidade de uma definição de metadado disponível globalmente e entendida universalmente é alcançada parcialmente através da utilização de um repositório de metadados (Figura 5). Um repositório de metadados é um banco de dados de propósito especial que armazena, controla e disponibiliza ao restante do ambiente todos os componentes de metadados relevantes. Os vários produtos de software que compõem a cadeia de informações recuperam metadado global a partir do repositório central, ao invés de ter conexões ponto-a-ponto com outros produtos. O repositório contém uma única definição do metadado total, baseada em um metamodelo único específico ao próprio repositório do produto. Cada produto deve implementar sua própria camada de acesso ao repositório, a qual entende as estruturas de metadados específicas do repositório (ex.: interfaces e metamodelo) e sabe como mapear tais estruturas específicas de repositórios para estruturas de metadados específicas de produtos.

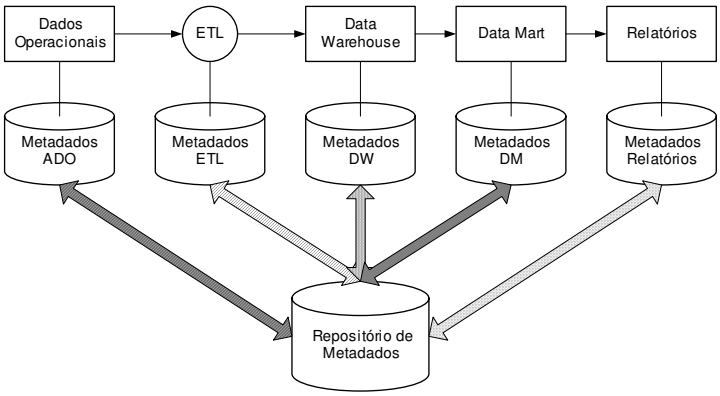


Figura 5 - Arquitetura de integração de metadados *hub-and-spoke*

Fonte: POOLE, 2002, pág. 19.

Apesar desta abordagem atenuar a necessidade da construção de muitas pontes ponto-a-ponto, o problema de se construir pontes ainda não foi totalmente eliminado. Uma camada

de acesso diferente deve ser desenvolvida para cada produto de software, e cada camada de acesso ainda é específica a um determinado produto de repositório. O repositório ainda é mais uma ferramenta que necessita ser integrada com outras ferramentas e produtos desenvolvidos pelo ambiente.

Precisa-se de uma arquitetura de integração de metadados única, de baixo custo que nos permita utilizar tanto a abordagem ponto-a-ponto ou repositório de metadados central como uma *estratégia de gerenciamento de metadados*, ao invés de serem utilizadas como arquiteturas de integração de metadados. O que se busca é um mecanismo de integração único e que funcione bem em qualquer situação, permitindo a escolha entre as abordagens ponto-a-ponto e de repositório tomando-se como base somente o nível de gerenciamento de metadados requerido, ao invés de se basear em custo e complexidade de integração.

2.3.4 Uma Abordagem Baseada em Modelo para Metadado

O pré-requisito para um mecanismo de intercâmbio padrão é um entendimento comum da sintaxe e semântica do metadado entre todos os componentes participantes. E uma solução possível para esse problema é introduzir um metamodelo comum o qual serve como base para sintaxe e semântica de metadados específicos de acordo com regras de comum concordância. (AUTH, 2002)

Grande parte dos produtos possui, disponibilizando seus metadados, metamodelos internos e interfaces proprietárias diferentes ou incompatíveis. Os metadados geralmente não descrevem todos os aspectos de seus dados, mas somente determinadas características essenciais. Em geral o metadado pode ser abstrato e conciso, não havendo a necessidade de descrever todas as características dos dados, mas somente a informação mínima requerida para que quaisquer operações possam ser executadas nos dados.

O metadado é essencialmente um modelo[5] formal de seu dado, sendo também uma descrição abstrata de dado que descreve dados precisamente. O metadado deve ser formulado de acordo com certas regras, assegurando que é corretamente formado. Isto garante que todo produto de software ciente das regras de descrição dos metadados, ao executar operações no dado correspondente, sempre será apto a interpretar corretamente o metadado.

Os metadados, quando expressos como um modelo formal e independente de plataforma, podem existir fora de, e independentemente de, qualquer plataforma determinada. Podendo ser traduzido para qualquer um dentre vários modelos diferentes de plataforma específica, cada um representando uma plataforma diferente (dados conjuntos apropriados de regras de mapeamento e implementações de tais regras).

Uma abordagem possível, de acordo com (POOLE, 2002), é desenvolver uma representação externa de metadados, a qual não é dependente de nenhum produto ou ferramenta. Tal representação é baseada em modelos de estruturas de informações formais e independentes de plataforma, descritos utilizando uma linguagem formal apropriada, tal como a UML.

Um produto utiliza tal modelo formal como base para seu próprio metadado invocando um processo de mapeamento de importação para traduzir o modelo formal em uma instância de seu próprio metadado, específico de produto. Similarmente, um produto pode expor seu metadado proprietário para outros produtos através de um processo de mapeamento de exportação que traduz seu metadado interno em um modelo formal, independente de plataforma.

Se o próprio metamodelo é disponibilizado e implementado independente de qualquer plataforma de implementação particular (produto) e os produtos que fazem o intercâmbio de

---

[5] Um modelo é uma descrição abstrata e precisa de algo, devendo ser construído de acordo com certas regras formais.

metadados concordam com este metamodelo externo e comum, então se acaba com o problema de tradução entre modelos de implementação proprietários.

À medida de que um metamodelo oferece uma descrição abstrata de objetos em um determinado domínio de interesses (ex.: esquemas de bancos de dados relacionais), deve-se assegurar que esse metamodelo é uma representação razoavelmente completa de seu domínio. Isto é, deve-se assegurar que ele é completo semanticamente ou que ao menos esteja razoavelmente próximo a ser. Para assegurar que o metamodelo é inteligível para todos os produtos de software, deve-se formá-lo de acordo com certas regras combinadas (isto é, de acordo com as semânticas de alguma linguagem formal).

Interoperabilidade de metadados é a habilidade de se intercambiar metadados ou proporcionar acesso a metadado entre os vários componentes de uma configuração de *data warehouse* (DO, 2000).

Esta abordagem de integração e interoperabilidade em nível de metadados pode ser descrita como uma *abordagem baseada em modelo para integração de metadados* ou como uma *arquitetura para metadado dirigida a modelo* (POOLE, 2002). Consiste fundamentalmente do intercâmbio de metadados compartilhados na forma de definições de metadados disponíveis expressos como modelos formais e independentes de produtos. Os softwares e ferramentas participantes deste ambiente devem concordar com um metamodelo comum que define o domínio como um todo, habilitando-os a rapidamente entender quaisquer instâncias do metamodelo (ex.: qualquer metadado compartilhado que possa ser intercambiado). Cada produto participante mapeia seu metadado compartilhado para sua própria representação interna de metadados. Isto requer que o metamodelo seja uma representação razoavelmente completa de seu domínio e que o metamodelo possa fornecer algum tipo de extensão, pela qual qualquer *gap* semântico no metamodelo possa ser compensado de uma forma padronizada e em concordância pelos produtos participantes.

Esta abordagem baseada em modelo elimina ou reduz consideravelmente os custos e as complexidades associadas com as arquiteturas de integração de metadados ponto-a-ponto tradicionais baseadas em pontes de metadados. Esta abordagem também oferece o mesmo benefício às arquiteturas de metadados baseadas em repositório central, no estilo *hub-and-spoke*.

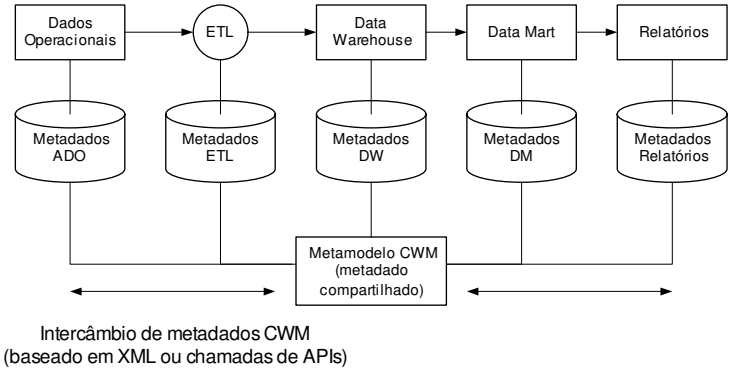


Figura 6 - Arquitetura de metadados ponto-a-ponto baseada em modelo

Fonte: POOLE, 2002, pág. 31.

Em arquiteturas ponto-a-ponto, como pode ser visto na Figura 6, a abordagem baseada em modelo elimina a necessidade de se construir múltiplas pontes de metadados. Ao invés disso, cada produto de software implementa um adaptador (uma camada da lógica do software) que entende o metamodelo comum e o metamodelo de implementação interno do produto. O adaptador é outra forma de uma ponte de metadado, mas uma que necessita ser escrita somente uma vez para um dado produto, porque todos os produtos concordam com o metamodelo comum. Isto vai de encontro à abordagem tradicional de pontes de metadados, onde as pontes são construídas para cada par de tipos de produtos que se pretende integrar.

A solução baseada em modelo claramente reduz os custos e complexidades da construção de arquiteturas de metadados ponto-a-ponto, aumentando o retorno do investimento na corrente de informações ou minimizando o esforço do *data warehouse*. A utilização de um metamodelo comum elimina a necessidade de se construir pontes de

metadados e permite uma equivalência semântica completa no nível de metadado entre as várias ferramentas e produtos que compõem a corrente de fornecimento de informações.

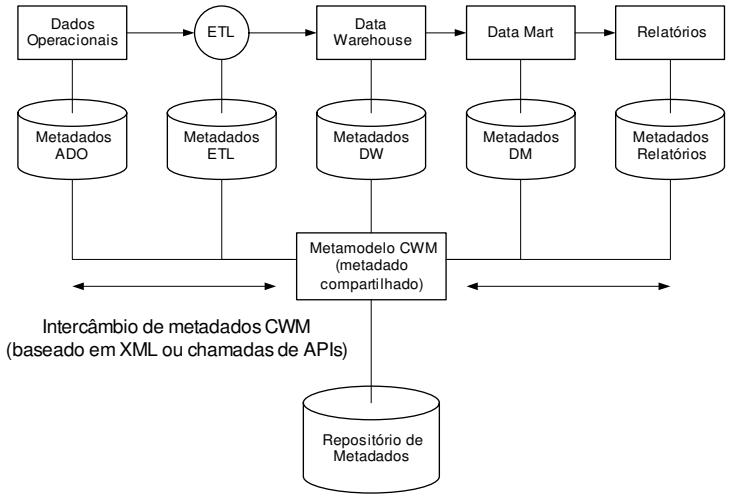


Figura 7 - Arquitetura de metadados *hub-and-spoke* baseada em modelo

Fonte: POOLE, 2002, pág. 32.

Nas arquiteturas de metadados *hub-and-spoke* baseadas em repositório central, o repositório geralmente emprega um novo significado como armazém central para as definições de metamodelo comuns e para todas as suas várias instâncias (modelos) utilizadas no ambiente completo. Conforme ilustrado na Figura 7, o repositório deve implementar uma camada com o adaptador independente de metamodelo como qualquer outro software ou ferramenta participante da arquitetura. O repositório poderia implementar seu próprio metamodelo interno e contar com sua camada de adaptação para traduzir entre instâncias compartilhadas do metamodelo comum e seu próprio metamodelo, apesar de ser preferível ao repositório implementar diretamente alguma representação interna do metamodelo comum. O adaptador independente de metamodelo inclui a camada de acesso ao repositório, tal camada geralmente é requerida neste tipo de ambiente de metadados, e além disso leva à redução de custos e complexidades de se construir uma solução baseada em repositório central.

Se a arquitetura física de metadados consistir de uma configuração ponto-a-ponto ou *hub-and-spoke*, o uso da abordagem baseada em modelo resulta na criação de uma arquitetura *hub-and-spoke* lógica. Neste caso, o metamodelo comum define o *hub* central, lógico. A decisão, no ambiente baseado em modelo, de que tipo de arquitetura de metadados implementar é menos determinada por questões de integração e influenciada mais diretamente por questões e estratégias de gerenciamento de metadados.

Uma solução de integração de metadados baseada em modelo completa consiste dos seguintes componentes (POOLE, 2002):

1. uma linguagem formal capaz de especificar metadados em termos de modelos compartilhados e independentes de plataforma.
2. um metamodelo comum que defina o domínio do problema.
3. um formato de intercâmbio comum para troca de metadados compartilhados.
4. uma interface de programação comum para acesso aos metadados.
5. mecanismos padrões para extensão dos modelos.
6. mecanismos padrões para extensão do metamodelo.
7. adaptadores de software facilitando importação e exportação de metadados relativos ao produto.
8. um repositório de metadados central (um componente opcional da arquitetura).
9. uma estratégia de gerenciamento de metadados abrangente.
10. uma arquitetura técnica de metadados abrangente.

Os componentes de 1 a 6 são os elementos primários que devem ser oferecidos por qualquer abordagem baseada em modelo para integração de metadado. Todos os componentes que participam da arquitetura de integração de metadados devem entender e concordar com o metamodelo comum. Modelos compartilhados intercambiados entre componentes são

instâncias do metamodelo comum. O metamodelo comum é independente de produto e de plataforma. O metamodelo deve possuir um nível suficiente de semântica (cobertura semântica ou completude semântica) para assegurar que ele pode modelar quase todos os aspectos do domínio do problema. Ele também deve ter sido formulado de acordo com um conjunto estabelecido de regras formais (linguagem abstrata) para assegurar que os vários softwares e ferramentas participantes na arquitetura sejam todos capazes de ter um entendimento comum do metamodelo.

Também devem ser definidas interfaces comuns que suportam intercâmbio e acesso a metadado compartilhado. Dois dos maiores inibidores à integração de metadados são a falta de um metamodelo padrão e de interfaces de metadados correspondentes. A solução baseada em modelo deveria definir um mapeamento do metamodelo comum tanto para um formato de intercâmbio baseado em fluxo quanto para uma interface expressa em alguma linguagem de programação.

A XML fornece uma linguagem de formato de dados simples e universal para o intercâmbio de todos os tipos de metadados do *warehouse*. E por ser simples e universal a XML é ideal para o intercâmbio de metadados ser bem sucedido e prevalecer, já que o formato de intercâmbio utilizado deve ser universal e barato de se implementar e manter.

XML é atualmente a linguagem de escolha para intercâmbio de dados na *web* e em qualquer ambiente fracamente acoplado, distribuído, assíncrono. Documentos XML são amplamente autodescritivos. XML permite a definição de *tags* configuráveis que são utilizadas para definir elementos em um documento XML. As *tags* representam conceitos definidos pelo metamodelo (mais uma vez, é empregado um conjunto de regras formais de mapeamento para associar elementos do metamodelo a *tags* específicas). O conteúdo do elemento é metadado, e as *tags* que definem os tipos possíveis de elementos são meta-metadados (elementos do metamodelo).

Ter uma interface programática comum (ou mapeamentos padrões de elementos de metadados para linguagens de programação amplamente utilizadas) simplifica grandemente a construção de clientes que devem acessar o metadado a partir de um determinado produto, ferramenta, banco de dados, ou outro serviço. Os clientes de metadados passam a ser codificados em apenas um formato de interface. Isto reduz custo e complexidade para os clientes de metadados.

Apesar de o metamodelo precisar ser uma descrição do domínio o mais completa possível, em algumas situações pode existir uma necessidade de intercambiar elementos de metadados que não constem no metamodelo. Neste caso o metamodelo pode oferecer construções padrões para a definição de novos elementos. Como estes elementos serão representados depende das ferramentas participantes do intercâmbio, já que seu significado semântico não pode ser inferido do metamodelo. Em outras situações, o metamodelo pode precisar ser estendido para incorporar novos conceitos de domínio. Estender um metamodelo existente com novos conceitos de domínios significa que a própria linguagem abstrata deve oferecer algum tipo de mecanismo que habilite essa operação de extensão.

Os adaptadores específicos de produtos e os repositórios de metadados (componentes 7 e 8) caem na área da implementação da arquitetura completa de integração de metadado baseada em modelo. Os adaptadores oferecem o mecanismo básico por onde um dado produto de software, aplicação, ferramenta ou banco de dados se conecta a arquitetura de metadado. Cada adaptador é específico de um determinado produto e contém a lógica para mapear o metamodelo comum para o metamodelo interno do produto. Um adaptador é a maneira pela qual um produto exporta seu metadado na forma de modelos padrões ao ambiente, como também a forma na qual um produto pode importar modelos compartilhados do ambiente.

Na arquitetura de integração de metadados baseada em modelo, o repositório serve como uma localização central para o armazenamento e gerenciamento de metamodelo comum

e de suas instâncias de modelo. O repositório também facilita a publicação dos vários modelos tornando-os disponíveis a todo o ambiente e oferecendo-os a quaisquer ferramentas ou produtos específicos que requisitá-los. Serviços críticos geralmente fornecidos pelo repositório: controle de acesso multiusuário, segurança e controle de versões. A utilização de um repositório de metadado pode ou não ser empregada pela arquitetura da solução, dependendo dos requisitos da estratégia completa de gerenciamento de metadado.

Uma estratégia de gerenciamento de metadado não é um componente da abordagem de integração de metadado baseada em modelo. A estratégia de gerenciamento de metadados determina o fluxo completo de metadado através do ambiente. Ex.: A estratégia de gerenciamento de metadado geralmente determina quais produtos ou ferramentas são responsáveis pela autoria/publicação de metadados; quais são responsáveis por publicação, declaração, controle, atualização, e gerenciamento de metadados existentes, e quais são consumidores somente de leitura de metadados compartilhados. Esta estratégia determina a arquitetura técnica completa de metadados, a qual resulta diretamente da estratégia de gerenciamento de metadados, e também as ferramentas selecionadas para implementar a arquitetura de integração de metadados.

A arquitetura técnica de metadado (componente 10) define como a estratégia de gerenciamento de metadados está fisicamente definida. Define a topologia específica de integração e interconexão de metadados, como a ponto-a-ponto, *hub-and-spoke*, ou rede federada. A arquitetura também define a distribuição de vários serviços de metadados a pontos específicos na topologia.

Se for decidido que o metamodelo comum deve ser publicado e gerenciado por um repositório, o repositório desenvolvido constitui um *hub* físico na arquitetura completa. Se um repositório central não precisar ser desenvolvido, a arquitetura deve de alguma forma disponibilizar os serviços tipo repositório e indicar onde eles são executados na arquitetura. O

metadado não deve apenas ser acessível globalmente e entendível universalmente através dessa arquitetura baseada em modelo, mas também deve ser localizável.

2.4 Considerações

Este capítulo procurou abordar os aspectos chave da integração de metadados a fim de esclarecer os conceitos básicos sobre o assunto, proporcionando o embasamento teórico.

Também foram destacadas as seguintes arquiteturas de integração de metadados: arquitetura ponto-a-ponto, arquitetura *hub-and-spoke* e arquitetura baseada em modelo. Nas arquiteturas ponto-a-ponto e *hub-and-spoke* foi verificado que o gerenciamento e implementação das pontes de integração são processos trabalhosos. E que, caso haja um crescimento no número de pontes, a manutenção pode vir a consumir muitos recursos.

Então se constatou que a associação da abordagem baseada em modelo com cada uma das outras duas arquiteturas leva a uma maior facilidade na implementação e manutenção das duas soluções. Verificando-se a grande vantagem em se expressar os metadados como modelos formais e independentes de produtos.

A solução da abordagem baseada em modelo é composta por uma série de componentes, e o *Common Warehouse Metamodel* é o padrão de intercâmbio de metadados que procura se adequar a essa solução. A finalidade do próximo capítulo é de se fazer uma explanação desse novo padrão, seus benefícios e suas tecnologias de fundamentação.

3 CWM – COMMON WAREHOUSE METAMODEL

O *Common Warehouse Metamodel* (CWM) surgiu da necessidade de se ter um padrão para o intercâmbio de metadados. De um modo geral, cada ferramenta possuía seu próprio formato de armazenamento de metadados, o qual geralmente era diferente do formato de outro fabricante.

Empresas que possuíam ferramentas de fabricantes diferentes dificilmente conseguiam fazer o intercâmbio de metadados entre uma ferramenta e outra, levando muitas vezes ao trabalho em dobro.

Percebeu-se então que para que todos pudessem sair ganhando, o ideal seria que esses metadados pudessem ser intercambiados entre as ferramentas dos diversos fabricantes interessados nesta nova abordagem. Foi quando um grupo de empresas se juntou à OMG para o estudo e desenvolvimento do padrão de intercâmbio de metadados CWM.

A razão fundamental para a utilização do CWM é gerenciar o ciclo de vida completo dos dados e conteúdo da empresa na Internet e intranet com melhor interoperabilidade. Com o CWM, o poder da modelagem de objetos e o gerenciamento de dados da empresa tornam-se agora disponíveis para modeladores de dados, projetistas de bancos de dados, administradores de *data warehouse*, e gerentes e desenvolvedores de portais corporativos.

O fato de se ter definições de metadados comuns facilita o intercâmbio de dados entre subsistemas, porque passa a existir um entendimento comum sobre o significado do dado (POOLE, 2001).

O grande problema é que o dado reside em muitos sistemas diferentes, em muitos formatos diferentes, e em muitas plataformas diferentes.

CWM é estruturado como uma coleção de metamodelos (ou sub-metamodelos) relacionados, cada metamodelo ocupando seu próprio pacote, e com um número mínimo de dependências entre pacotes (POOLE, 2000).

Os metadados compartilhados entre os produtos são formulados em termos de modelos de dados que são consistentes com um ou mais metamodelos CWM. Um produto exporta metadado através da formulação de um modelo de suas estruturas de metadados internas em um formato definido pelo CWM. Similarmente, um produto importa metadados através do consumo de um modelo em conformidade com o CWM e posterior mapeamento deste para seus metadados internos. (POOLE, 2001)

No contexto de e-Business, metadados e modelos são utilizados em conjunto para trazer um novo nível de abstração e flexibilidade ao software e à arquitetura do negócio. Sendo que ambos são fundamentais aos processos de configuração e personalização necessários para se gerenciar informações do negócio (conteúdo) e processos que utilizam este conteúdo.

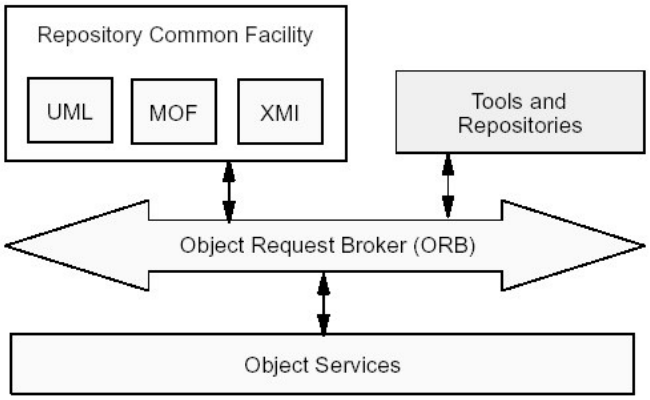


Figura 8 - Arquitetura de Repositório de Metadados da OMG

Fonte: OMG, 2002a, pág. 22.

Com a chegada da UML e a especificação MOF (*Meta Object Facility*), a qual define a arquitetura de metadados da OMG exibida na Figura 8, o aumento de padrões e *software* dirigidos a metamodelos cresceu rapidamente. A chegada da XML e da XMI (*XML Metadata Interchange*) unificou os mundos da modelagem, repositórios de metadados federados e a linguagem dos metadados de Internet.

Modelos de desenvolvimento de aplicações de e-Business, integração de aplicações e *business intelligence/data warehousing* (CWM) estão agora sendo desenvolvidos e padronizados como serviços de objetos de baixo nível (Transações, Segurança) e começaram a ser padronizados em 1990. Tais modelos permitem que os projetistas tirem vantagens das construções de modelagem mais ricas (semântica, relacionamentos, restrições) disponíveis em UML/MOF. O seguinte cenário de ferramentas (vide Tabela 1) cobre importantes ferramentas de *data warehousing* de alguns fornecedores que participam da cooperativa:

• **Dimension EDI:** Polyval XML Mediator, Polyval XML Questionnaire

• **Hyperion:** Hyperion Essbase OLAP Server, Hyperion Integration Server, Hyperion Application *Link*, Hyperion Analytical Reporting

• **IBM:** Visual Warehouse, DB2 Family, DB2 OLAP Server, IMS, Team Connection

• **NCR:** Teradata Warehouse Suite

• **Oracle:** Oracle Express, Oracle 8i, Oracle Discoverer, Oracle Warehouse Builder, Oracle Repository

• **Unisys:** Unisys Universal Repository (UREP)

Tabela 1 - Cenário de Ferramentas

Pacote CWM	Dimension EDI	Hyperion	IBM	NCR	Oracle	Unisys
CWM and Metadata Repository Facility		X	X		X	X
Software Deployment			X	X	X	
Relational	X	X	X	X	X	X
Record			X	X		X
Multidimensional		X			X	
XML	X		X		X	X
Transformation	X	X	X	X	X	
OLAP		X	X	X	X	
Data Mining		X	X	X	X	
Information Visualization	X	X	X		X	
Business Nomenclature	X	X	X		X	
Warehouse Process			X	X	X	
Warehouse Operation			X	X	X	

Fonte: OMG, 2002a, pág. 55.

CWM é um superconjunto da UML, e permite que uma menor granularidade de detalhes seja incorporada aos modelos de objetos. Também suporta Modelagem ER e fornece integração destas duas técnicas de modelagem para acelerar a adoção da modelagem de objetos para gerenciamento de dados de formas estruturadas ou não estruturadas em bancos de dados e arquivos ou em formatos XML na Internet.

Basicamente o CWM consiste de:

1. uma linguagem padrão para a definição da estrutura e da semântica de metadados de uma maneira semi-formal, através do *Meta-Object Facility* (MOF) e da *Unified Modeling Language* (UML).
2. um mecanismo de intercâmbio padrão para compartilhamento de metadados definido na linguagem padrão, através da *eXtensible Markup Language* (XML) e da *XML Metadata Interchange* (XMI).
3. uma especificação padrão (interface) para acesso e análise dos metadados definidos na linguagem padrão, através da *CORBA Interface Definition Language* (IDL).

CWM permite a troca de metadados entre todas as tecnologias de *data warehouse*, *business intelligence*, gerenciamento do conhecimento e portal. Provê um modelo de objetos detalhado com um conjunto de APIs, formatos de trocas e serviços à cerca do ciclo de vida completo de metadados incluindo extração, transformação, transporte, carga, integração e até a fase de análise de um *data warehouse*. CWM também permite que os usuários estendam o modelo para que ele seja apropriado a suas necessidades específicas.

O CWM estende-se além do espaço do *data warehouse* para também gerenciar dados no banco de dados, sistemas *web*, sistemas legados, etc. em diferentes sistemas de arquivos. Provê o gerenciamento do ciclo de vida completo dos metadados do *data warehouse* e permite que se mova e compartilhe dados e metadados. A troca de informações entre

armazéns diferentes é também um forte benefício do CWM, especialmente porque as arquiteturas de dados variam grandemente nas companhias e muito mais ainda entre parceiros de negócios.

E, apesar de o propósito principal do CWM ser o de permitir a troca de metadados entre repositórios e ferramentas diferentes, ele também pode ser utilizado para construção de *modelos de objetos ativos*[6] para armazenamento e manutenção de metadados.

CWM é composto por cinco camadas (vide Figura 1):

- ***ObjectModel*** – Camada subconjunto da UML é utilizada pelo CWM como seu metamodelo base. Contém serviços gerais que são compartilhados por outros pacotes. É construída sobre o padrão da UML 1.3.
- ***Foundation*** – Suporta a especificação de vários elementos e serviços comuns, tais como tipos de dados, chaves abstratas e índices, expressões, informação do negócio, e implantação de software baseado em componente.
- ***Resource*** – Contém modelos de dados utilizados para fontes de dados operacionais e *data warehouse*. Esta camada endereça arquivos tradicionais, estruturas de banco de dados orientados a registros, estruturas relacionais primariamente utilizadas em aplicações de processamento de transações, e recursos de dados baseados em objetos e baseados em XML.
- ***Analysis*** – Provê metamodelos para transformação de dados, OLAP, relatório/visualização de informação, nomenclatura do negócio, e *data mining*. O metamodelo *Transformation*, por exemplo, suporta a definição de transformações entre fontes e destinos de *data warehouses*, e o metamodelo de

OLAP permitem que *data warehouses* armazenados em ferramentas relacionais ou multidimensionais possam ser vistos como dimensões e cubos.

- ***Management*** – Seus metamodelos representam processos e operações de *warehouse*. Isto permite a modelagem de eventos programados (como extrações e cargas diárias, por exemplo), e também o acompanhamento do status e completude de atividades, e manter um histórico das mudanças feitas aos elementos do *warehouse* (POOLE, 2000).

Sempre que possível, o CWM reutiliza diretamente classes e associações UML existentes ao invés de criar versões específicas de CWM. Esta escolha reduz o número de novas classes e associações CWM e potencializa as habilidades existentes de modeladores com conhecimento em UML.

O foco do CWM é permitir a visualização de dados em um formato aberto e que seja acessível por todos. Como CWM utiliza padrões abertos como o MOF, é possível utilizar informações do CWM e incrementar os investimentos em Java, C++, C, Cobol, COM+, e muitas outras tecnologias (via CORBA). O CWM possui tradução XML nativa, o que significa que ele foi definido utilizando formatações XML e utiliza tecnologias complementares como o XMI para a transferência de dados e metadados entre ferramentas, aplicações, *data warehouses* e portais de informações da empresa que estiverem em conformidade com o padrão. A especificação CWM inclui XML DTDs para cada um dos pacotes de metamodelos que compõe o CWM. Os esquemas também são diretamente expressos como documentos XML baseados no MOF DTD.

---

[6] Um Modelo de Objeto Ativo é um modelo baseado em instâncias e não em classes, representando classes, atributos e relacionamentos como metadados. A representação de objeto do domínio em estudo possui um modelo de objeto que é interpretado em tempo real, e que pode ser modificado com efeito imediato, porém controlado, no sistema que estiver interpretando-o e executando-o (YODER, 1998; YODER, 2001).

Uma empresa pode ter seus analistas de software treinados em Modelagem de Objetos e seus DBAs treinados em Modelagem E/R. O CWM permite que estas duas áreas trabalhem em conjunto utilizando os metadados do CWM para construir modelos E/R diretamente ou para transformar modelos de objetos UML em modelos E/R e vice-versa.

3.1 Benefícios

CWM permite que hajam melhores tomadas de decisão, já que a informação disponível existente em múltiplos recursos pode ser reunida e analisada em um formato único. O custo de se coletar estes dados é mais baixo já que não há necessidade de se construir e manter interfaces proprietárias específicas entre as fontes de dados.

O CWM permite elevar o investimento em UML e estender este investimento à parte de *data warehouse* do negócio. CWM facilita a sinergia e colaboração em áreas que tradicionalmente têm sido mantidas separadas.

O processamento ETL é simplificado por um modelo de metadado comum que coleciona métricas do processo ETL e entende melhor a utilização dos dados. Estas métricas são utilizadas então para melhorar o processo ETL. Capturando a linhagem de dados em CWM, usuários finais podem navegar nas informações utilizando termos dos negócios, entender o impacto de transformações de dados em valores atuais, e facilmente buscar as origens dos dados nos sistemas fontes.

Para administradores de *data warehouse*, que precisam ter acesso a toda informação necessária para controlar e monitorar o estado do *data warehouse*, o acesso rápido à fonte, atualização e derivação de informação significará recuperação mais rápida, e melhor integridade de dados.

Como a maioria dos clientes utiliza várias ferramentas diferentes e cada ferramenta possui seu próprio metadado, o gerenciamento através dessas ferramentas pode tornar-se um desafio. CWM ao disponibilizar um formato padrão para acesso aos metadados simplifica o suporte a múltiplas ferramentas, traduzindo-se em economia para a empresa.

Para fornecedores que implementem CWM, a nova abordagem arquitetural baixa o custo de se implementar ferramentas, pontes de tecnologia e formatos de intercâmbio específicos entre cada produto e todos os seus pares. Assim como CORBA tem aberto a interoperabilidade para objetos distribuídos na última década, CWM fornece um mecanismo central de troca e interoperabilidade para dados e conteúdos complexos. Essa troca é reforçada através da utilização de modelos de objetos precisamente especificados, modelos de dados e modelos de gerenciamento de *warehouses* para gerenciar as dependências de todos os dados relacionados em um *data warehouse* da empresa.

3.2 CWM Como Modelo para Integração de Metadados

O metadado específico da ferramenta ainda é controlado localmente pela ferramenta associada mas também é disponibilizado e acessado por outros componentes do *data warehouse* através de um mecanismo de intercâmbio padrão baseado em XMI. (AUTH, 2002)

CWM é um padrão que define um metamodelo comum de metadados e um formato de intercâmbio de metadados baseado em XML para os domínios de *data warehouse* e de análise de negócios. Como metamodelo, o CWM oferece a semântica e a sintaxe necessárias para a construção de metadados, descrevendo todos os componentes de uma cadeia de informações. CWM é composto de vários metamodelos diferentes porém fortemente relacionados, onde cada metamodelo representa algum subdomínio do ambiente da cadeia de informações.

Cada bloco da Figura 1 representa um metamodelo constituinte, ou pacote, do CWM e corresponde a uma área funcional importante da cadeia de informações.

Mesmo que o dado exista ou não no momento, ou que venha a existir em algum ponto no futuro, ainda assim pode ser descrito, pois os modelos derivados de CWM existem independentemente de determinado software, tecnologia ou plataforma.

CWM é expresso em UML, a qual fornece uma linguagem de modelagem única e poderosa para a modelagem de todos os tipos de metadados do *warehouse*. Ela permite a modelagem dos metadados, servindo como linguagem de modelagem única e disponibilizando uma abordagem dirigida a modelo para o intercâmbio de metadados.

Enquanto a UML é a base notacional para a definição do CWM, CWM também estende um subconjunto da linguagem UML para incluir conceitos de domínios de negócios e de *data warehouses*. CWM faz uso da expressividade e do poder da UML como formas de definir estruturas e relacionamentos de metadados complexos.

3.3 Os Metamodelos do CWM

O CWM é composto pelas camadas *ObjectModel, Foundation, Resource, Analysis* e *Management*, como pode ser visto na Figura 9. A seguir a descrição sucinta de cada metamodelo participante de cada uma das cinco camadas.

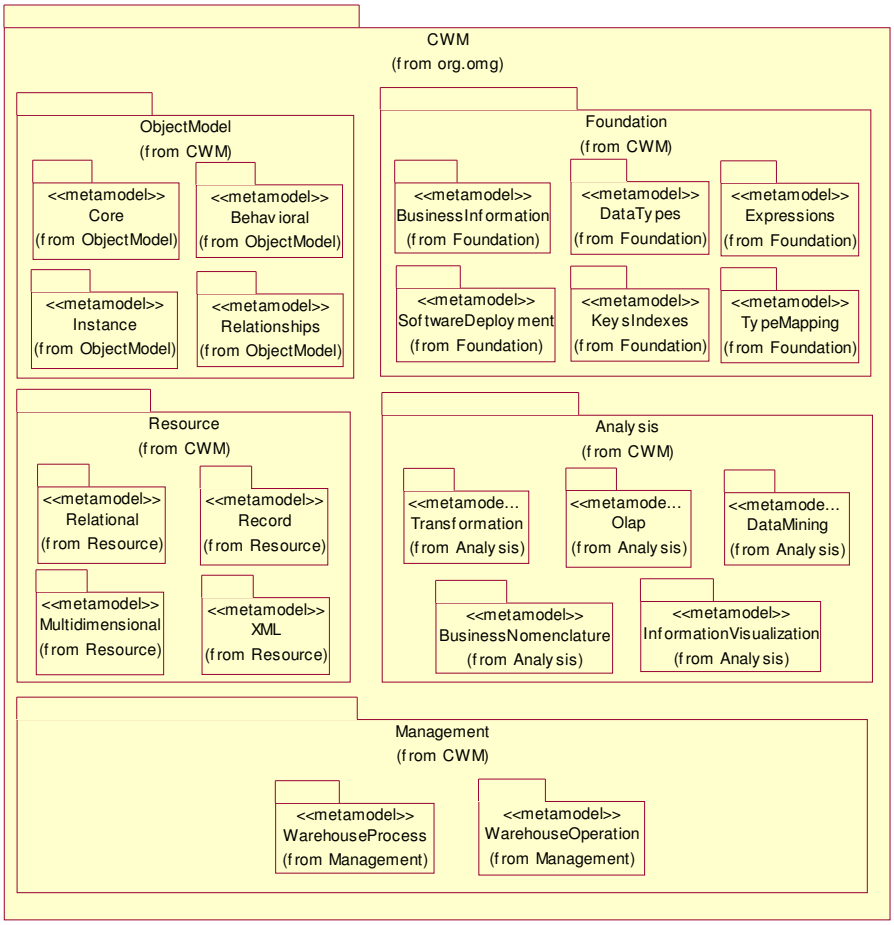


Figura 9 - Pacotes que compõem o CWM

3.3.1 Camada ObjectModel

Esta camada fornece as construções básicas para a criação e descrição de classes de metamodelos em todos os demais pacotes CWM. É uma subcamada da UML utilizada pelo CWM como base para seu metamodelo. Consiste de 4 metamodelos (Figura 10): *Core, Behavioral, Relationships* e *Instance*.

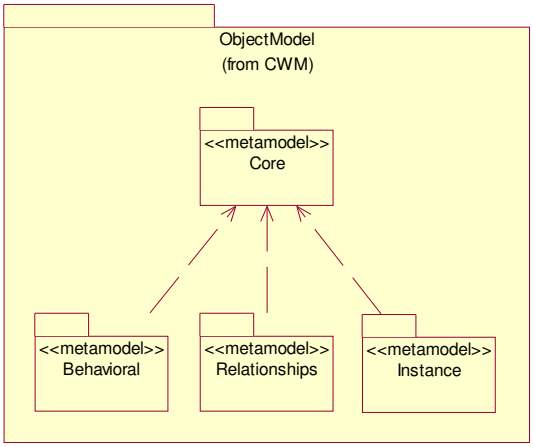


Figura 10 - Pacotes do metamodelo *ObjectModel*

Fonte: OMG, 2002a, pág. 72.

O **metamodelo *Core*** (Figura 11) não depende de nenhum outro pacote. Contém as classes e associações básicas utilizadas por todos os outros pacotes do CWM.

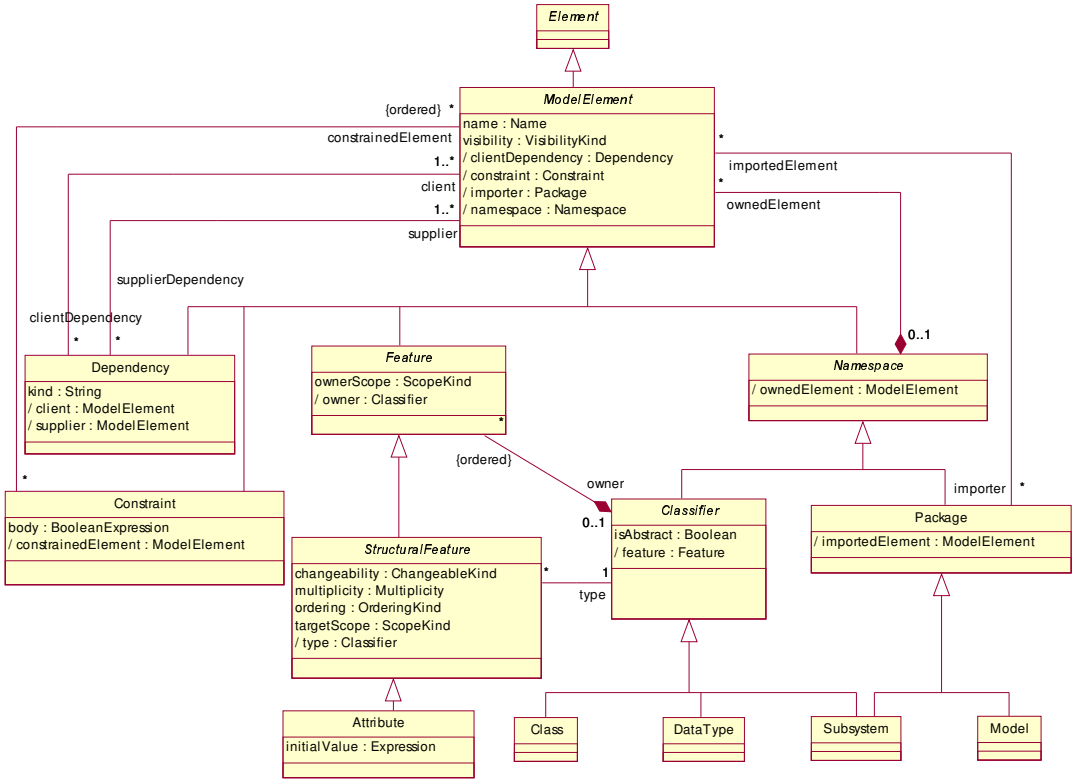


Figura 11 - Metamodelo *Core*

Fonte: OMG, 2002a, pág. 73.

O **metamodelo *Behavioral*** (Figura 12) contém classes e associações que descrevem o comportamento dos tipos CWM e fornece a base para o armazenamento das invocações de comportamentos definidos.

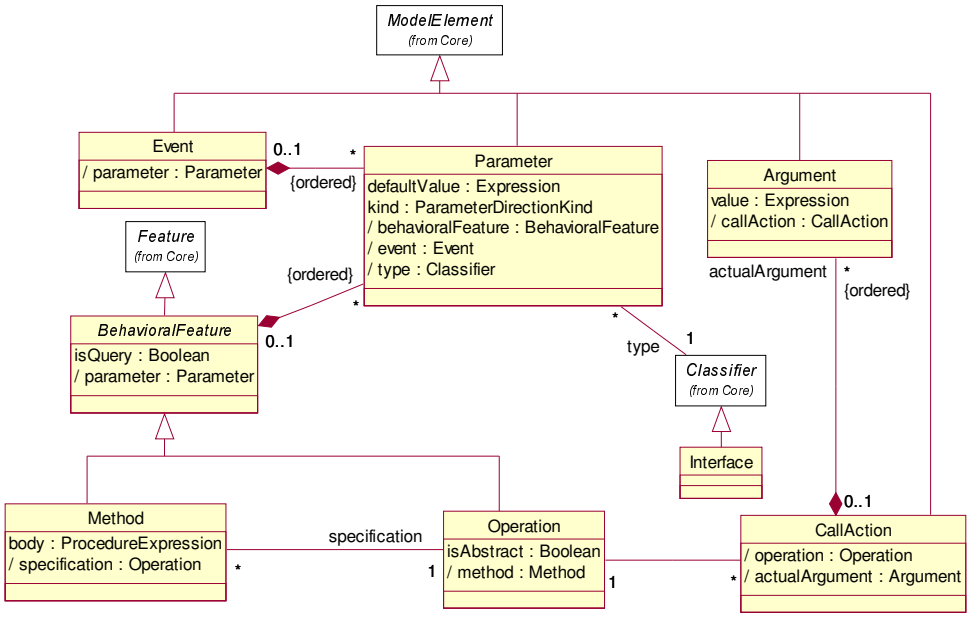


Figura 12 - Metamodelo *Behavioral*

Fonte: OMG, 2002a, pág. 112.

O **metamodelo *Relationships*** (Figura 13) contém classes e associações que descrevem os relacionamentos entre os objetos, permitindo descrever os relacionamentos de associação e generalização. As hierarquias de generalização CWM suportam herança múltipla.

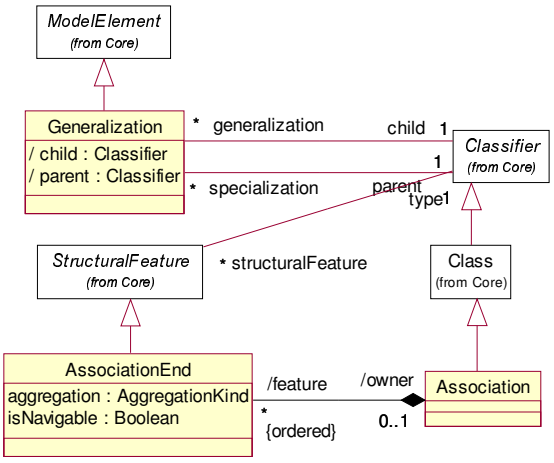


Figura 13 - Metamodelo *Relationship*

Fonte: OMG, 2002a, pág. 132.

Às vezes é necessário intercambiar, além dos metadados, determinadas instâncias de dados. O **metamodelo *Instance*** (Figura 14) permite a inclusão destas instâncias com o metadado.

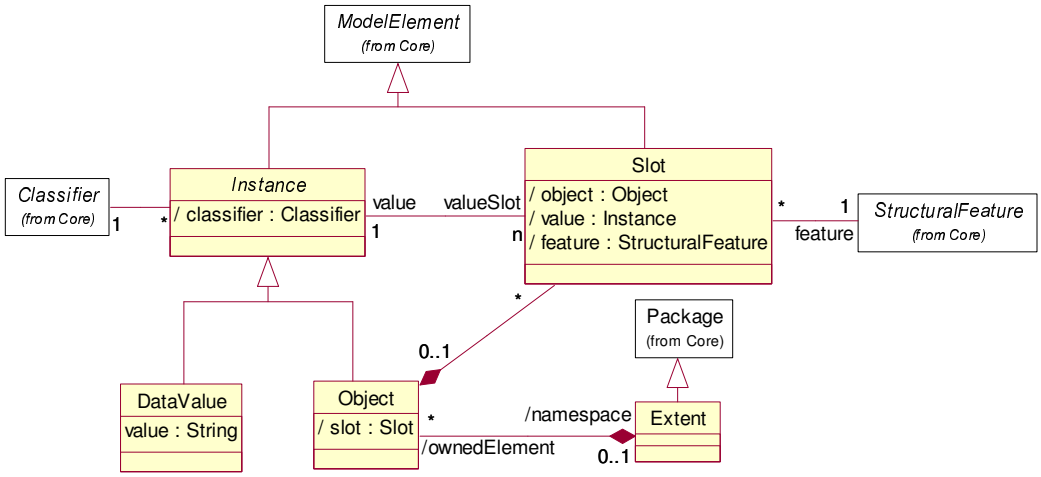


Figura 14 - Metamodelo *Instance*

Fonte: OMG, 2002a, pág. 142.

3.3.2 Camada Foundation

Esta camada consiste de uma coleção de pacotes que contêm elementos que representam conceitos e estruturas compartilhados por outros pacotes CWM (Figura 15). Seus elementos diferem dos elementos da camada *ObjectModel* porque são específicos aos propósitos e objetivos do CWM, enquanto que os elementos do *ObjectModel* são de natureza de propósito geral, sendo aplicáveis em diversas áreas.

O ponto chave desta camada é que , definindo-se tais conceitos em um nível bastante abstrato da arquitetura CWM, assegura-se que estes conceitos sejam definidos somente uma vez e que eles possam ser reutilizados em contextos mais específicos (POOLE, 2002).

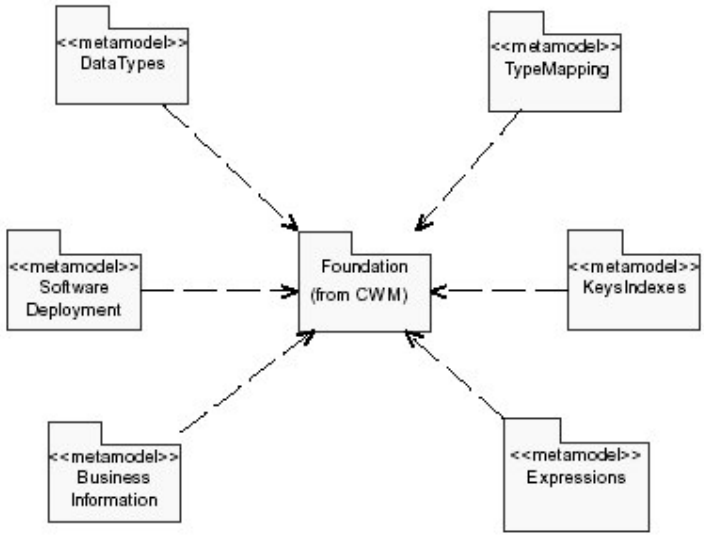


Figura 15 - Pacotes de nível mais alto da camada *Foundation*

Fonte: OMG, 2002a, pág. 156.

O **metamodelo *Business Information*** (figura 16) disponibiliza para todos os pacotes CWM serviços de propósito geral para a definição de informações orientadas ao negócio sobre os elementos do modelo. Tais serviços são projetados para suportar as necessidades dos sistemas de *data warehouse* e *business intelligence*. Estes serviços suportam as noções de partes responsáveis e informações sobre como contatá-los, identificação de documentação *off-line* e suporte para informações descritivas de propósito geral.

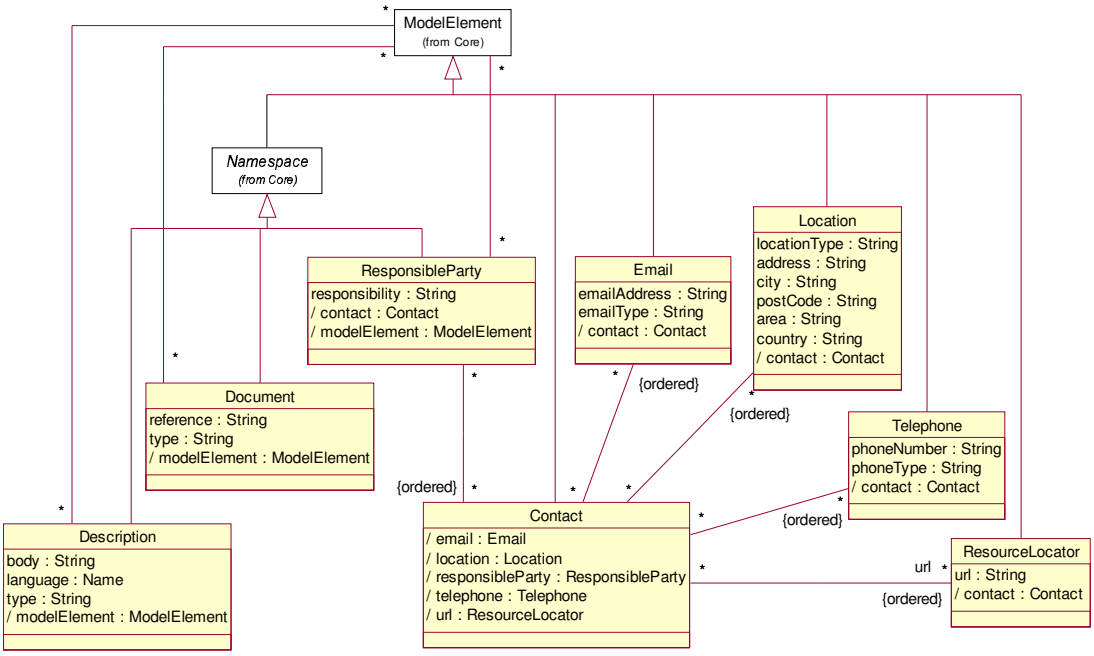


Figura 16 - Metamodelo *BusinessInformation*

Fonte: OMG, 2002a, pág. 157.

O **metamodelo *DataTypes*** (Figura 17) suporta a definição de construções no metamodelo que podem ser utilizadas para a criação de tipos de dados específicos que forem necessários. Apesar de o CWM *Foundation* não conter definições de tipos de dados específicos, são fornecidas várias definições de tipos de dados para ambientes utilizados amplamente como exemplos da utilização apropriada destas classes.

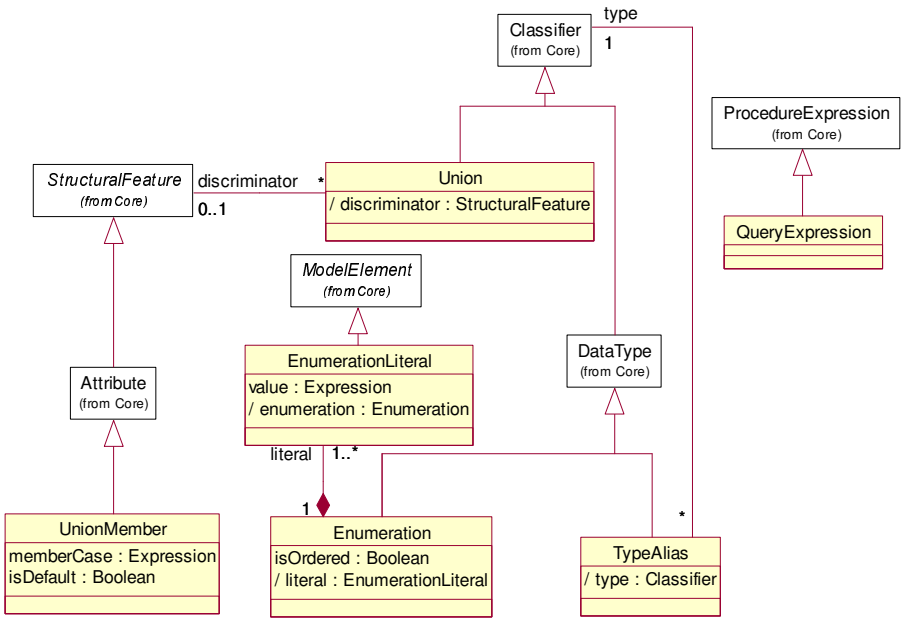


Figura 17 - Metamodelo *DataTypes*

Fonte: OMG, 2002a, pág. 181.

O **metamodelo *Expressions*** (Figura 18) fornece o suporte básico para a definição de árvores de expressão. A intenção é de se fornecer um lugar para que os demais pacotes CWM (como o *Transformation*) e para que as ferramentas concordantes com o CWM possam gravar as expressões compartilhadas em um formato comum que possa ser utilizado para intercâmbio.

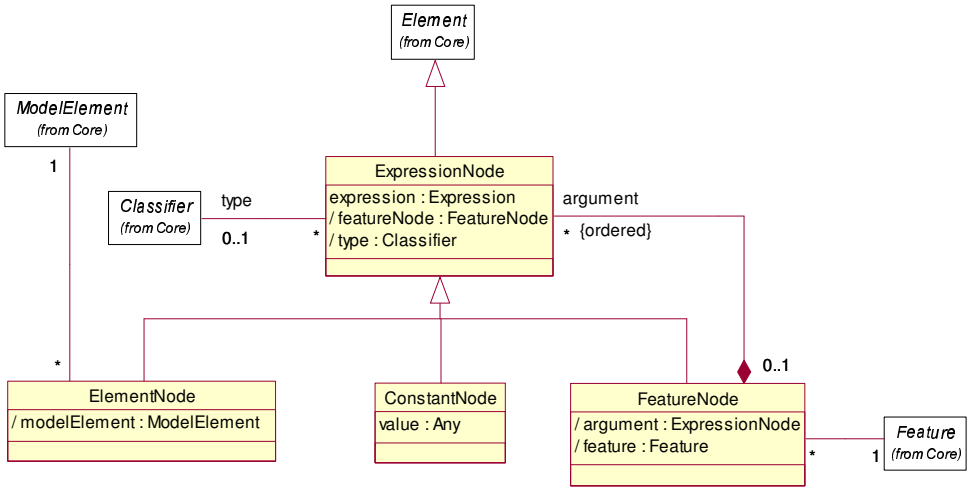


Figura 18 - Metamodelo *Expressions*

Fonte: OMG, 2002a, pág. 191.

O **metamodelo *KeyIndexes*** (Figura 19) permite definir os conceitos abstratos de chaves únicas e estrangeiras, e também as restrições de ordenação impostas aos conjuntos de dados. Os conceitos de chave e índice foram disponibilizados na camada *Foundation* porque são utilizados em muitos tipos de recursos de dados (modelos relacionais, modelos de objetos, *data warehouse* etc).

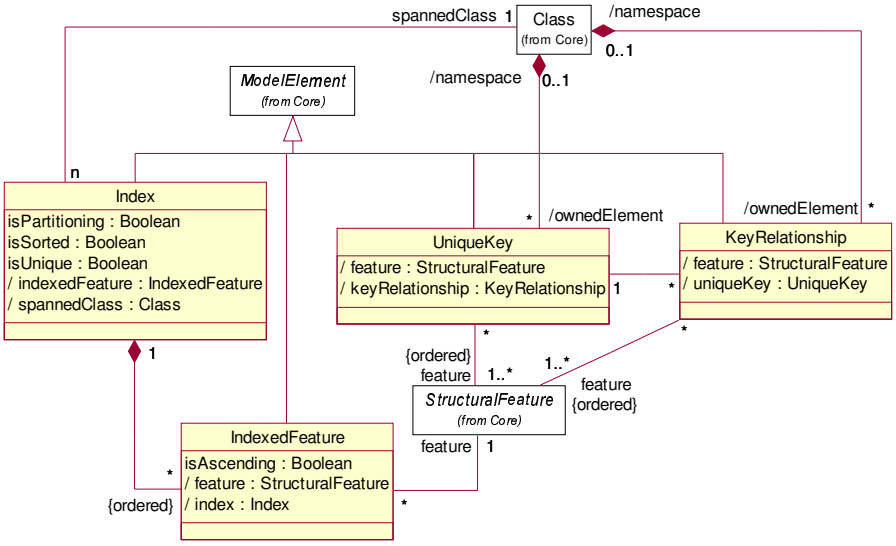


Figura 19 - Metamodelo *KeyIndexes*

Fonte: OMG, 2002a, pág. 202.

O **metamodelo *SoftwareDeployment*** (Figura 20) contém classes para armazenar a forma de utilização de *software* e *hardware* no *data warehouse*. Desta forma as ferramentas podem utilizar as informações encontradas aqui para localizar componentes de *software* e *hardware* do *warehouse*. O pacote tenta capturar somente as configurações operacionais necessárias para servir aos outros pacotes CWM, não tentando ser um modelo completo ou de propósito geral.

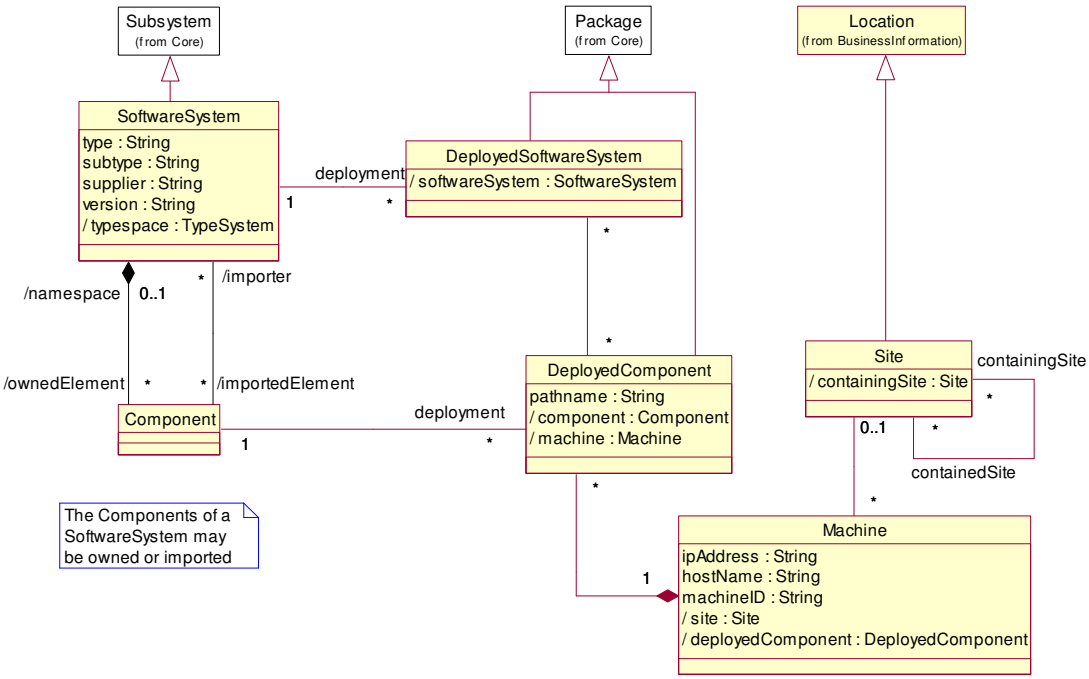


Figura 20 - Metamodelo *SoftwareDeployment*

Fonte: OMG, 2002a, pág. 216.

E a seguir a inclusão de algumas classes (Figura 21) no **metamodelo *SoftwareDeployment*** para identificação de mecanismos que fornecem acesso aos dados (SGBD ou sistema de gerenciamento de arquivos, por exemplo), de tipo de container de dados (esquema, catálogo relacional, arquivos, por exemplo), e de mecanismos que agem como cliente para fornecer acesso aos dados (ODBC, JDBC).

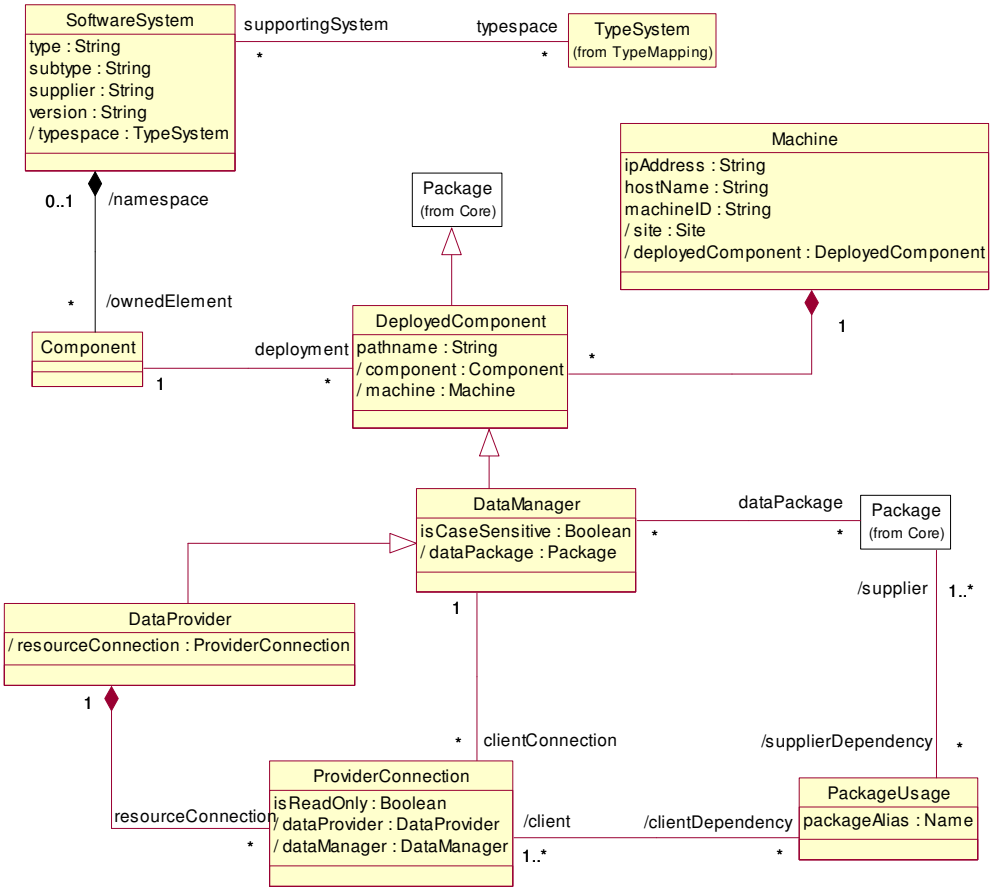


Figura 21 – Metamodelo *SoftwareDeployment: DataManager* e *DataProvider*

Fonte: OMG, 2002a, pág. 217.

O **metamodelo *TypeMapping*** (Figura 22) suporta o mapeamento de tipos de dados entre sistemas diferentes, com o objetivo de indicar tipos de dados em sistemas diferentes que sejam compatíveis de forma que os valores de dados possam ser intercambiados entre tais sistemas. São permitidos mapeamentos múltiplos entre quaisquer pares de tipos e também é fornecida uma forma de identificação do mapeamento preferido.

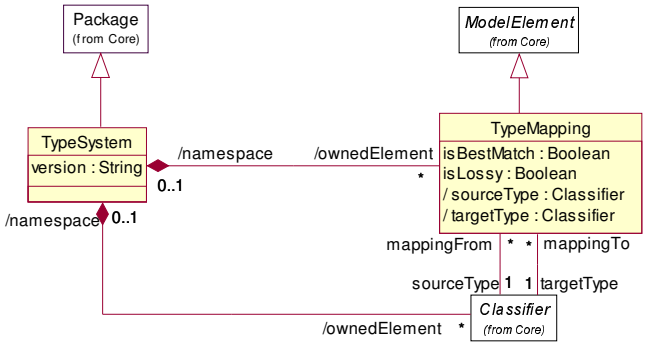


Figura 22 - Metamodelo *TypeMapping*

Fonte: OMG, 2002a, pág. 239.

3.3.3 Camada Resource

Esta camada define metamodelos dos vários tipos de recursos de dados que podem participar de uma cadeia de informações. Tais metamodelos estendem as camadas *ObjectModel* e *Foundation* para definir novos elementos de modelagem utilizados para construir metadados que definem bancos de dados relacionais, orientados a registro, orientados a objeto, servidores multidimensionais, e recursos de dados baseados em documentos XML.

Os pacotes da camada *ObjectModel* podem ser utilizados para descrever a estrutura de bancos de dados orientados a objetos e de aplicações orientadas a objetos que agem como fontes de dados (objetos COM e EJB, por exemplo).

O **metamodelo *Relational*** descreve dados que podem ser acessados através de uma interface relacional (um SGBD nativo ou JDBC, por exemplo). Pretende-se que o container *Catalog* cubra, através de uma única declaração, todas as tabelas que um usuário possa utilizar. Um catálogo contém esquemas (Figura 23), os quais por sua vez contêm tabelas (Figura 24). Tabelas são feitas de colunas, as quais possuem um tipo de dado associado.

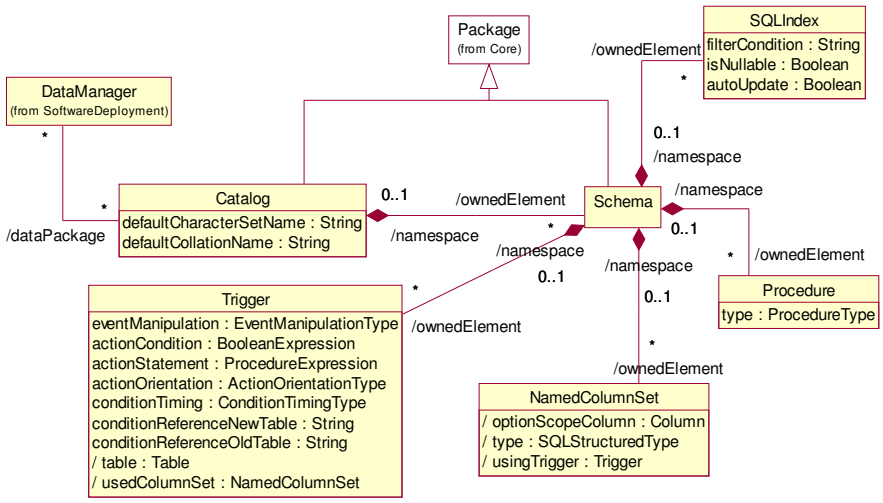


Figura 23 – *Metamodelo Relational*: Esquemas e objetos

Fonte: OMG, 2002a, pág. 247.

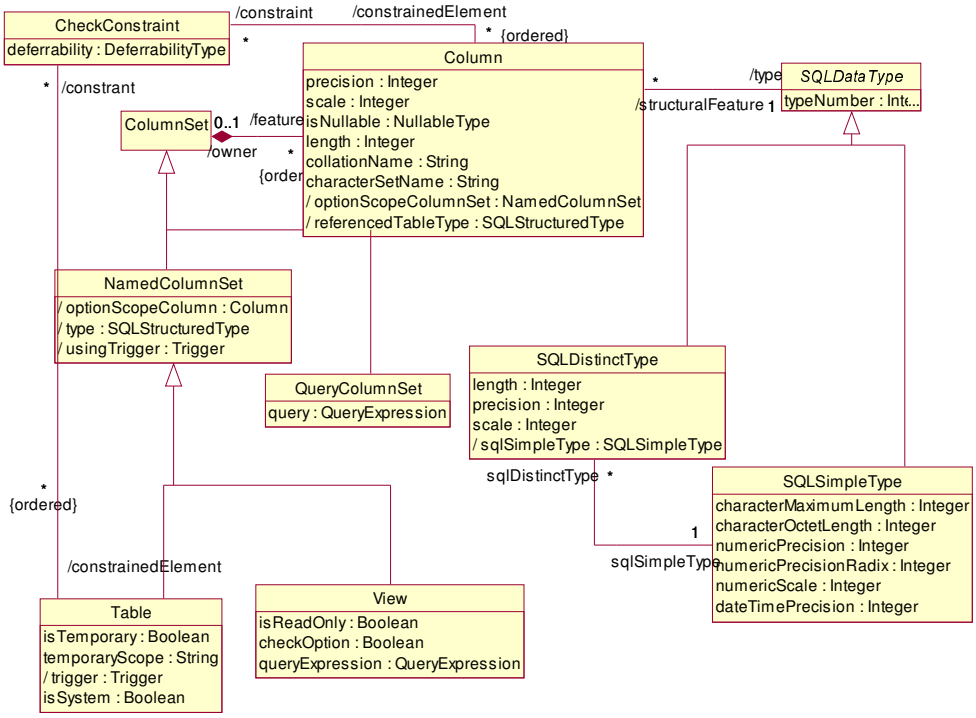


Figura 24 - *Metamodelo Relational*: Tabelas, colunas e tipos de dados

Fonte: OMG, 2002a, pág. 248.

O padrão SQL adiciona noções orientadas a objetos ao SQL com tipos estruturados (Figura 25). Este tipo estruturado é definido em termos de colunas.

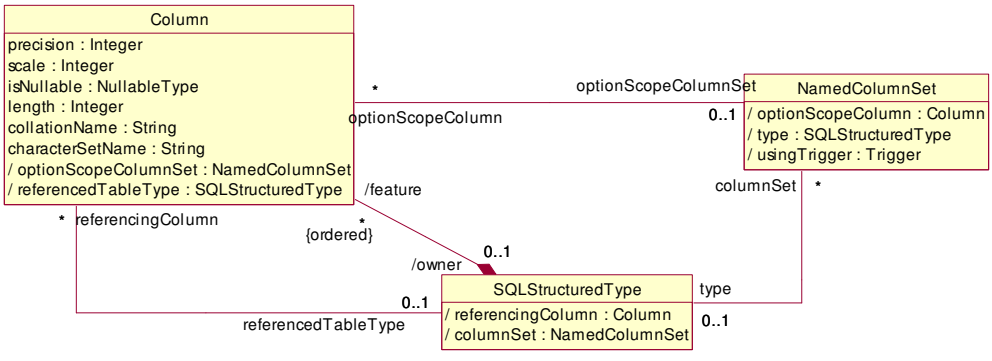


Figura 25 - *Metamodelo Relational*: *SQLStructuredType* e suas associações

Fonte: OMG, 2002a, pág. 251.

A seguir, na Figura 26, um exemplo de dois tipos de dados estruturados:

```
CREATE TYPE Credor_Fornecedor AS (nome varchar(80), endereço varchar(80))
CREATE TYPE Consignatario UNDER Credor_Fornecedor AS (codFolhaPagamento integer, descFolhaPagamento varchar(80))
```

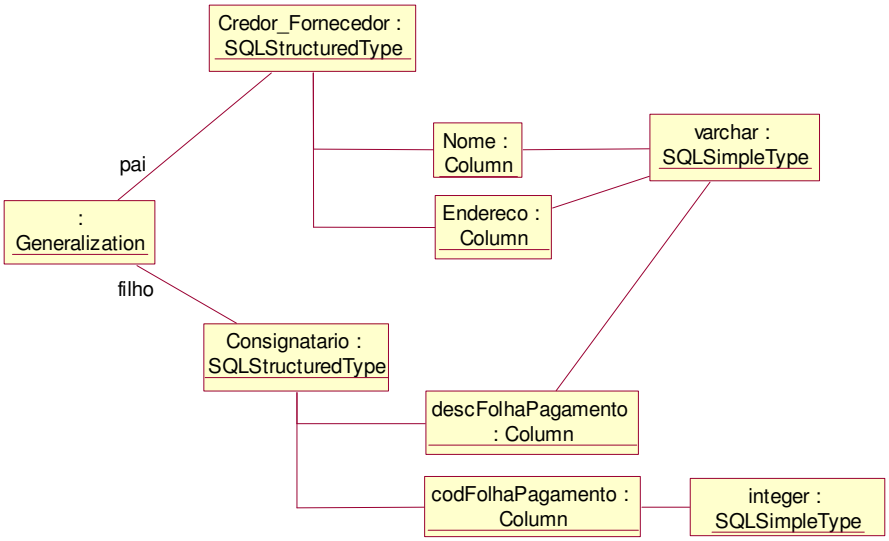


Figura 26 - Diagrama de Instância para dois tipos estruturados

Este metamodelo também permite a descrição de indexação (figura 27), chaves primárias e estrangeiras (Figura 28) através de conceitos correspondentes encontrados na camada *Foundation,* do metamodelo *KeyIndexes.*

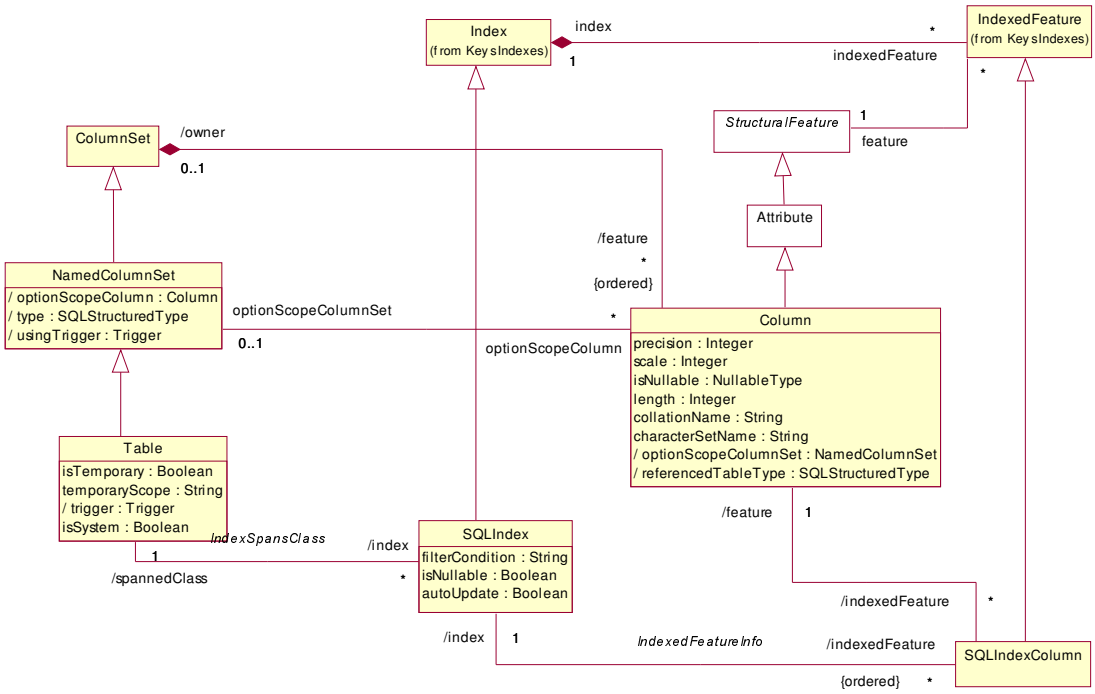


Figura 27 - *Metamodelo Relational*: Indexação

Fonte: OMG, 2002a, pág. 253.

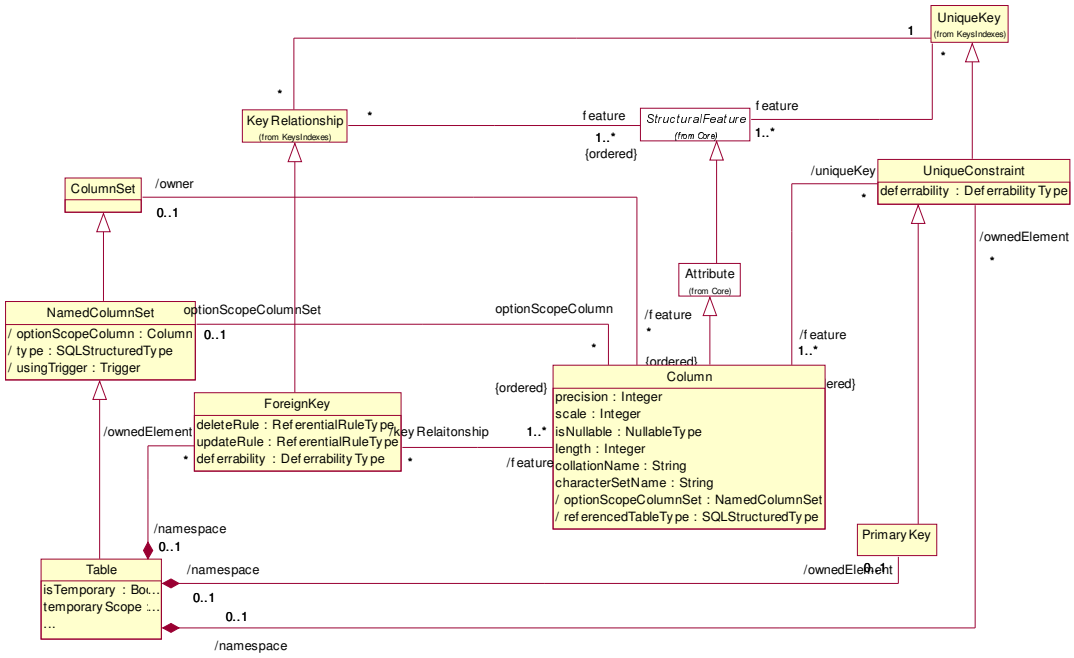


Figura 28 - *Metamodelo Relational*: Chaves primária e estrangeira

Fonte: OMG, 2002a, pág. 252.

As *triggers* (Figura 29) representam uma ação a ser executada quando ocorre alguma mudança em determinada tabela, e também podem ser representadas através deste metamodelo.

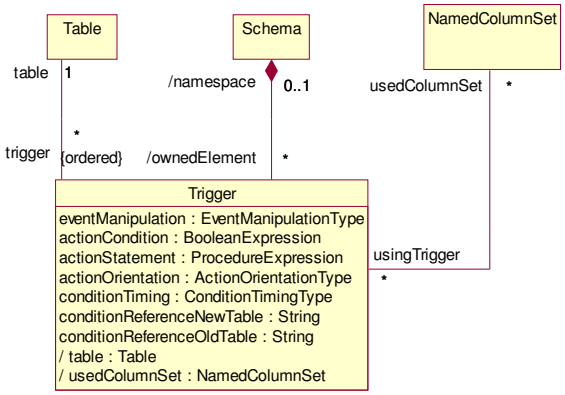


Figura 29 - *Metamodelo Relational*: *Triggers*

Fonte: OMG, 2002a, pág. 254.

As *procedures* (Figura 30) estendem a classe *Method*, do *ObjectModel*, e fazem parte de um esquema.

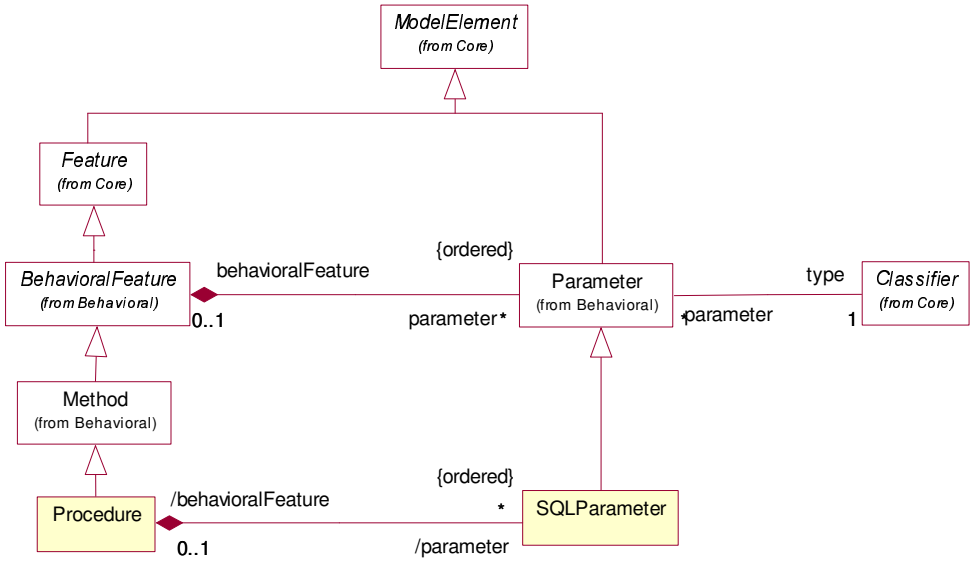


Figura 30 - *Metamodelo Relational*: *Stored Procedures*

Fonte: OMG, 2002a, pág. 254.

Além de todas essas funcionalidades, às vezes é necessário fornecer uma cópia ou um exemplo do dado (instâncias – Figura 31) como parte do metadado. Esta funcionalidade é similar à utilização de diagramas de Colaboração na UML.

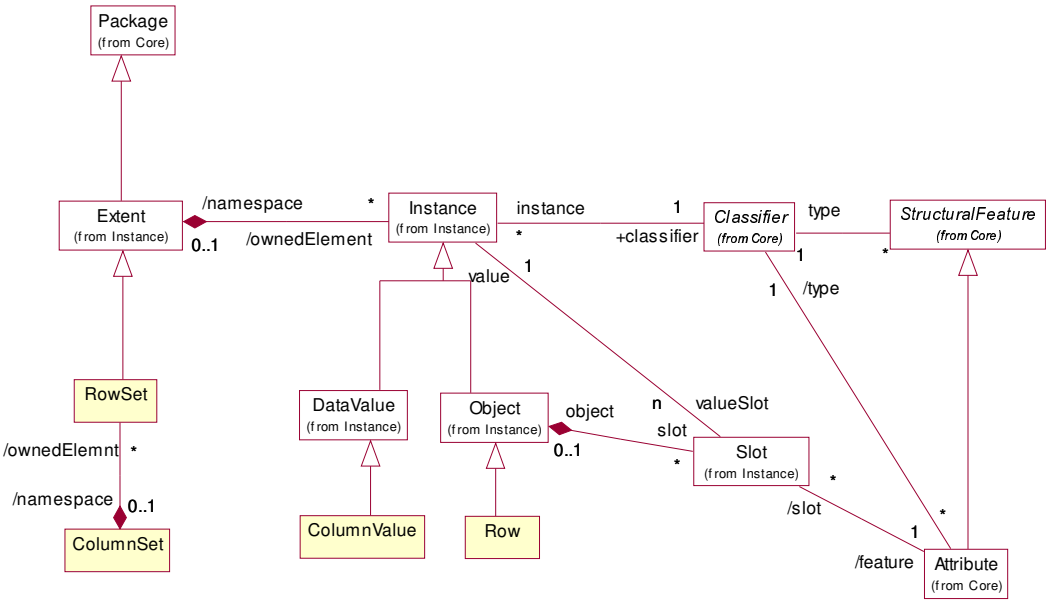


Figura 31 - *Metamodelo Relational*: Classes *Instance*

Fonte: OMG, 2002a, pág. 255.

O **metamodelo *Record*** (Figura 32) cobre o conceito básico de um registro e sua estrutura. O pacote dá uma visão ampla da noção de registro, incluindo tanto os registros que são armazenados em arquivos e bancos de dados quanto os tipos de dados estruturados das linguagens de programação. Estes conceitos podem ser utilizados como base para pacotes de extensão, descrevendo quaisquer estruturas hierárquicas, ou aninhadas, por natureza tais como documentos, questionários, e estruturas organizacionais.

Como existem muitos modelos antigos baseados em registro, com características nem sempre padrões em algumas linguagens ou implementações, tais características foram deixadas fora do metamodelo *Record*, podendo ser embutidas nos pacotes de extensão. No pacote CWMX (OMG, 2002b), estão disponibilizadas os seguintes pacotes de extensões: COBOL Data Division, DMS II e IMS.

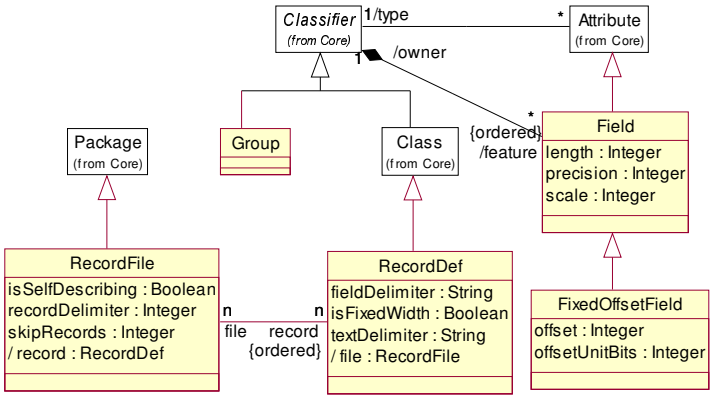


Figura 32 - Metamodelo *Record*

Fonte: OMG, 2002a, pág. 286.

O **metamodelo *Multidimensional*** (Figura 33) é uma representação genérica de um banco de dados multidimensional. Estes são bancos de dados OLAP que são implementados diretamente pelos sistemas de banco de dados multidimensionais.

Este metamodelo não pretende fornecer uma representação completa de todos os aspectos dos bancos de dados multidimensionais disponíveis comercialmente, podendo ser feitas extensões ao metamodelo, como as disponibilizadas no pacote CWMX (OMG, 2002b): Essbase e Express.

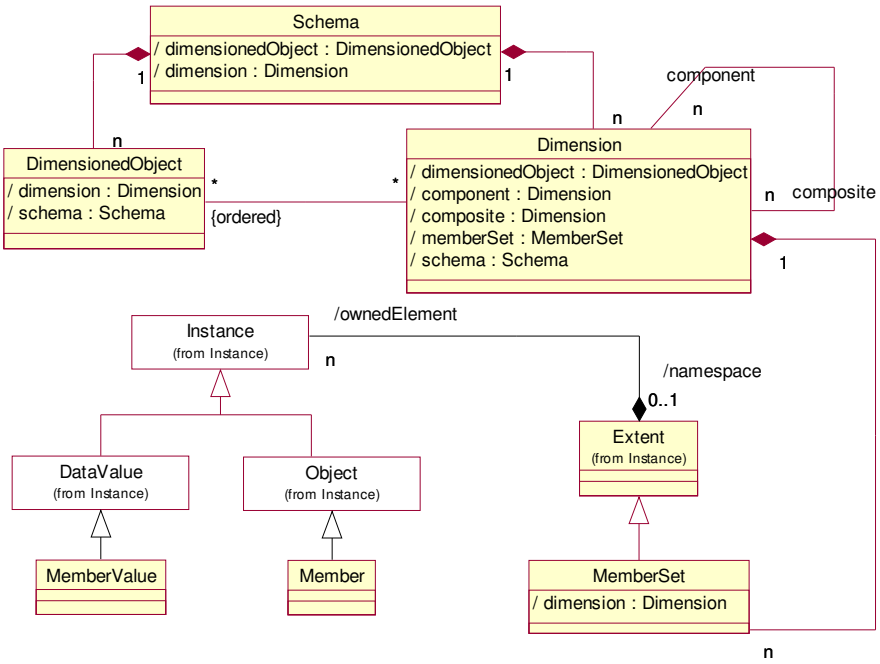


Figura 33 - Metamodelo *Multidimensional*

Fonte: OMG, 2002a, pág. 298.

A Figura 34 ilustra a herança das classes *Multidimensional* a partir das metaclasses do *ObjectModel.*

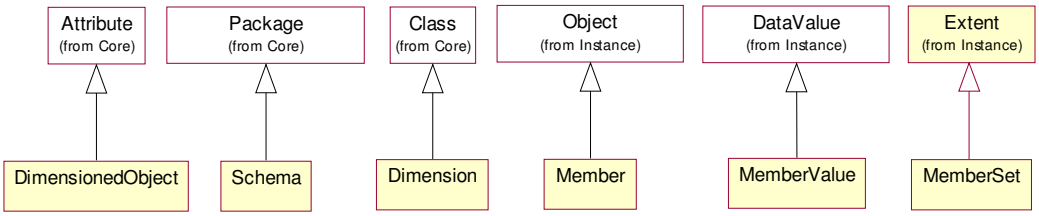


Figura 34 - Metamodelo *Multidimensional*: herança do *ObjectModel*

Fonte: OMG, 2002a, pág. 299.

O metamodelo XML (figura 35) não contém um documento XML com dados a serem intercambiados, ao invés disso, contém o XML *Document Type Definition* que descreve a estrutura de documentos XML que podem ser intercambiados. É baseado no XML 1.0.

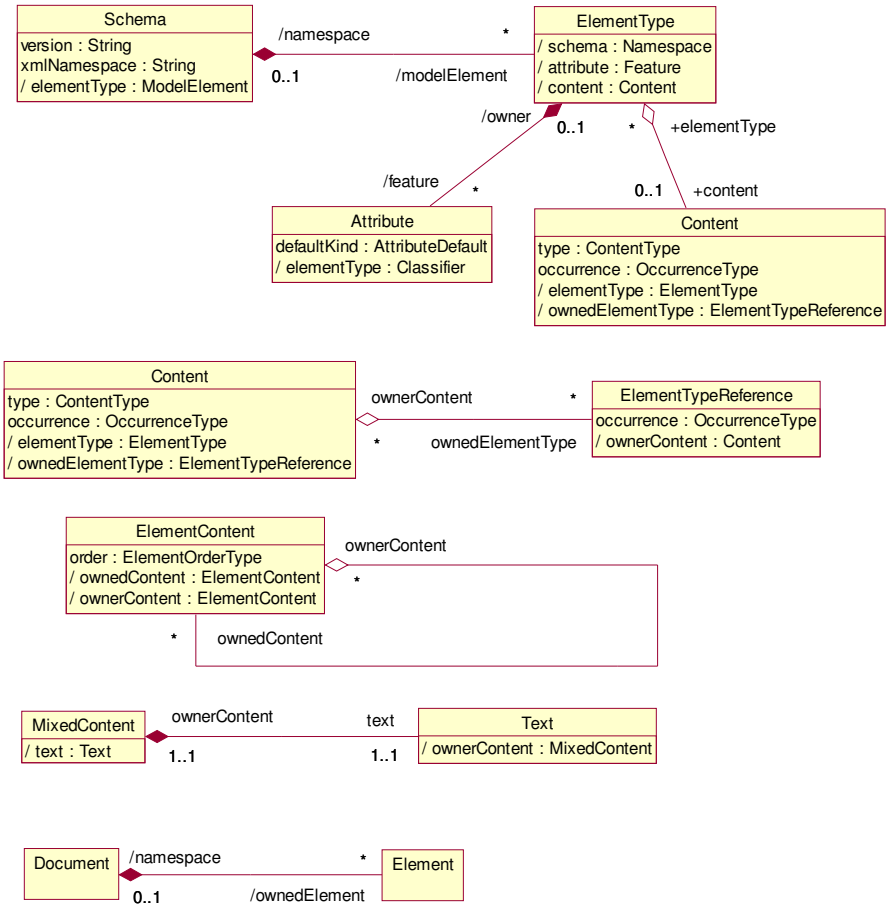


Figura 35 - Metamodelo XML

Fonte: OMG, 2002a, pág. 311.

3.3.4 Camada Analysis

Conceitos da análise do negócio são representados por metamodelos que fazem parte desta camada.

O **metamodelo *Transformation*** (figuras 36 e 37) contém classes e associações que representam metadados de transformação comuns utilizados em *data warehouse*. Ele cobre transformações básicas entre todos os tipos de fontes e destinos de dados: orientado a objeto, relacional, registro, multidimensional, XML, OLAP, e *data mining*.

Por exemplo, o tipo de fonte de uma transformação pode ser um XML *Schema* enquanto que o tipo do destino pode ser uma coluna, caso a transformação tenha que lidar com o armazenamento de um documento XML em uma coluna de um banco de dados relacional.

Este metamodelo contém elementos que suportam as seguintes funções:

- transformação e linhagem de dados – Estas classes e associações lidam com transformações e suas fontes, destinos, restrições e operações.
- agrupamento e execução de Transformações – Lidam com agrupamento de transformações para formar unidades lógicas e para definir seqüências de execução.
- transformações especializadas – Estas classes e associações definem transformações especializadas que são comumente utilizadas em *data warehouse*.

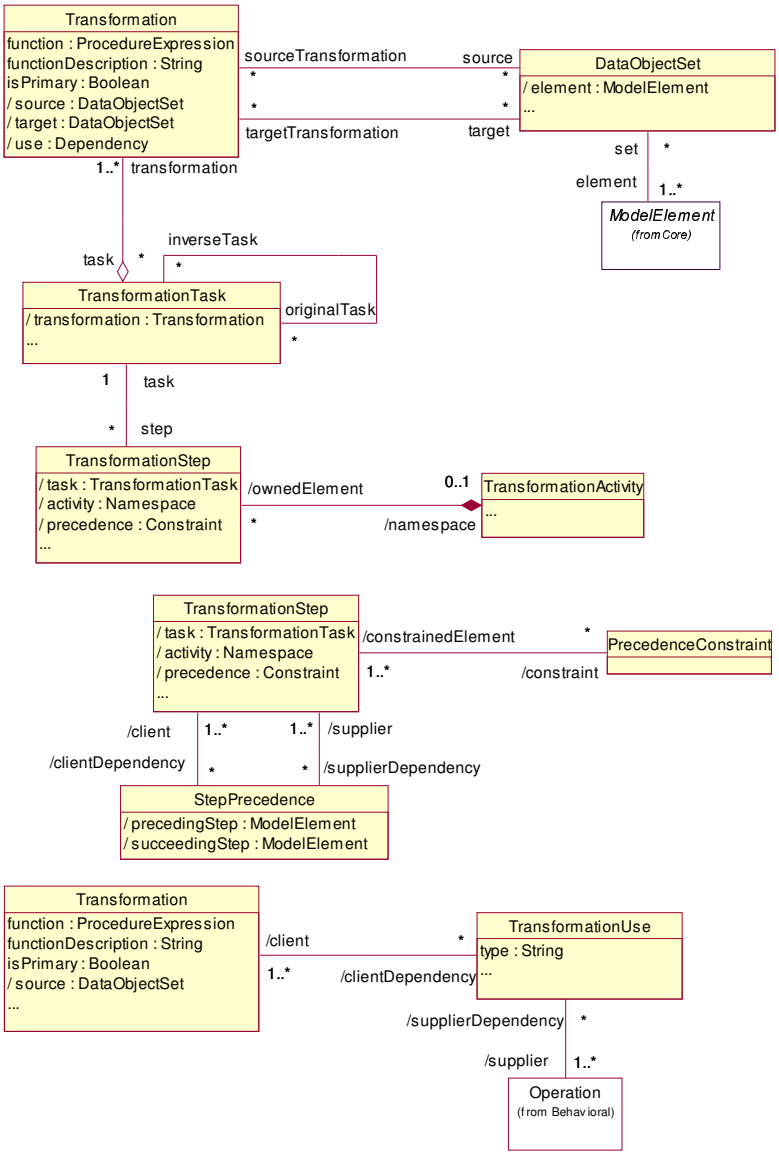


Figura 36 - Metamodelo *Transformation*: Relacionamentos – 1

Fonte: OMG, 2002a, pág. 330.

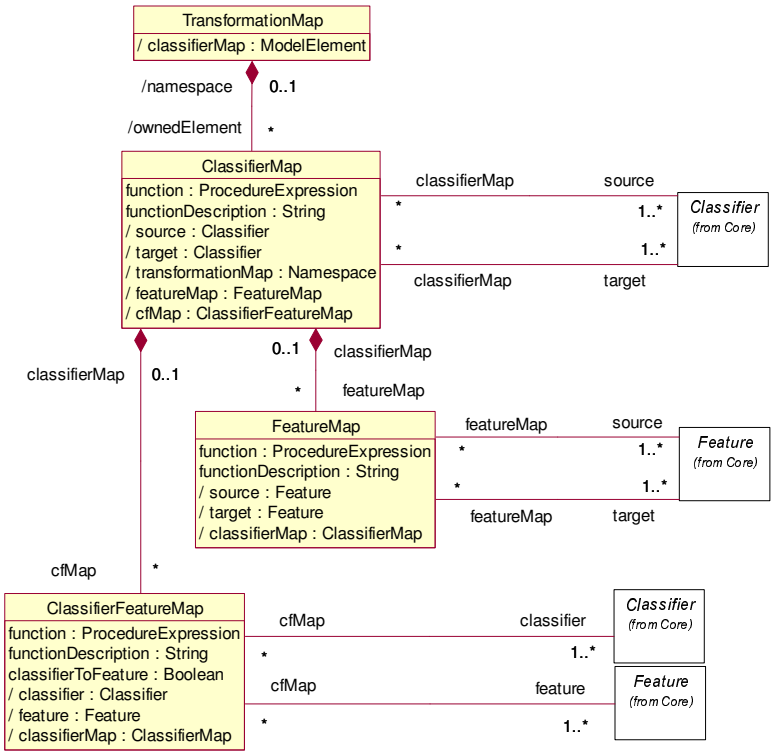


Figura 37 - Metamodelo *Transformation*: Relacionamentos - 2

Fonte: OMG, 2002a, pág. 331.

O **metamodelo *OLAP*** (Figuras 38 e 39) possui os seguintes objetivos:

- definir um metamodelo de conceitos OLAP essenciais.
- fornecer uma facilidade pela qual instâncias do metamodelo OLAP sejam mapeadas para estruturas de implantação (por exemplo modelos de recursos de dados físicos como os pacotes CWM *Relational* e *Multidimensional*).
- assegurar que a navegação através da hierarquia do modelo OLAP lógico e seus vários modelos de recursos seja sempre executada de uma forma uniforme.
- alavancar os serviços fornecidos por outros pacotes CWM, onde for apropriado.

Este metamodelo permite a definição de objetos como esquemas, cubos, dimensões, seleção de membros da coleção de uma dimensão, entre outros.

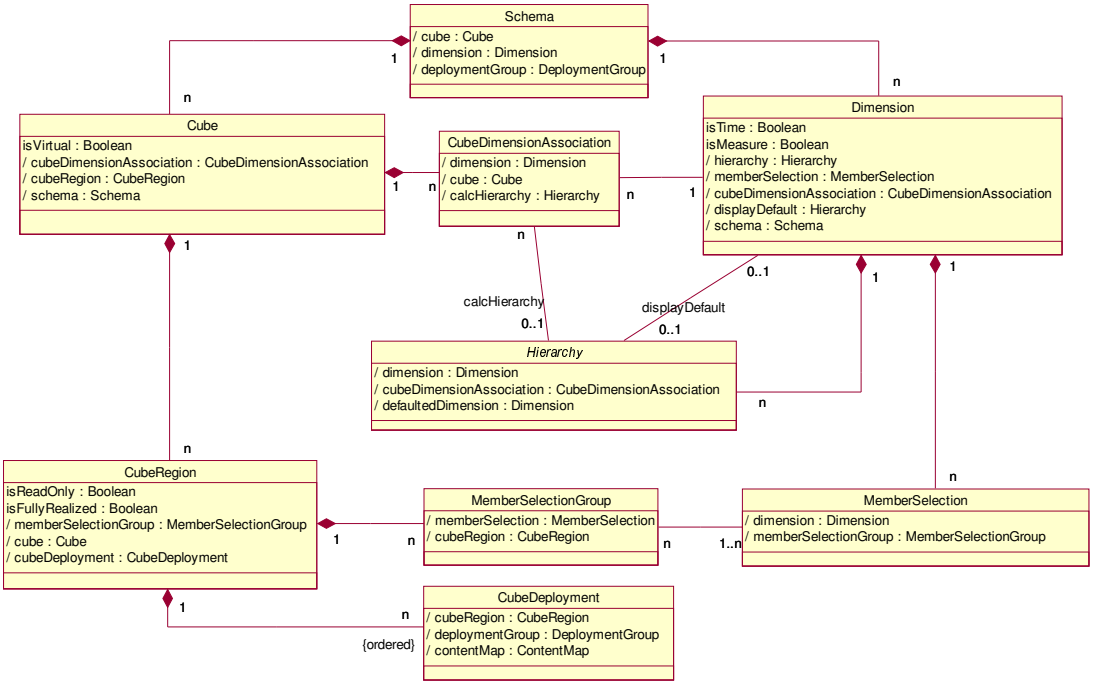


Figura 38 - Metamodelo *OLAP*: Classes e associações principais

Fonte: OMG, 2002a, pág. 361.

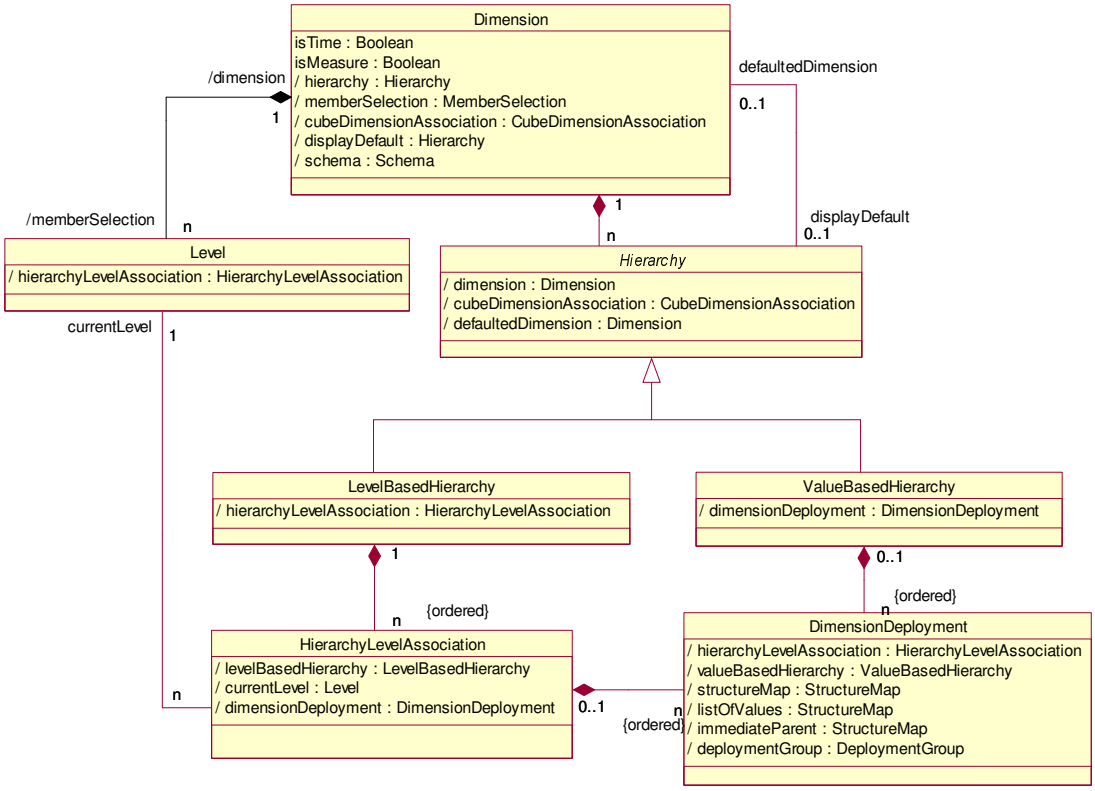


Figura 39 - Metamodelo *OLAP*: Dimensão e Hierarquia

Fonte: OMG, 2002a, pág. 363.

O metamodelo OLAP descreve os modelos lógicos dos sistemas OLAP, mas não especifica diretamente como um sistema OLAP é implantado fisicamente. A modelagem da implantação de um sistema OLAP requer o mapeamento (vide Figura 40) de instâncias de metaclasses OLAP para instâncias de outras metaclasses CWM que representem os recursos físicos (por exemplo o mapeamento de uma Dimensão OLAP para uma Tabela Relacional).

O pacote CWM *Transformation* é utilizado como a forma principal de se descrever tais mapeamentos. Um desenvolvedor construindo um modelo OLAP baseado em CWM irá definir as instâncias da metaclasse *TransformationMap* para conectar os objetos OLAP

lógicos, e para conectar tais objetos lógicos a outros objetos que estiverem representando suas fontes físicas de dados.

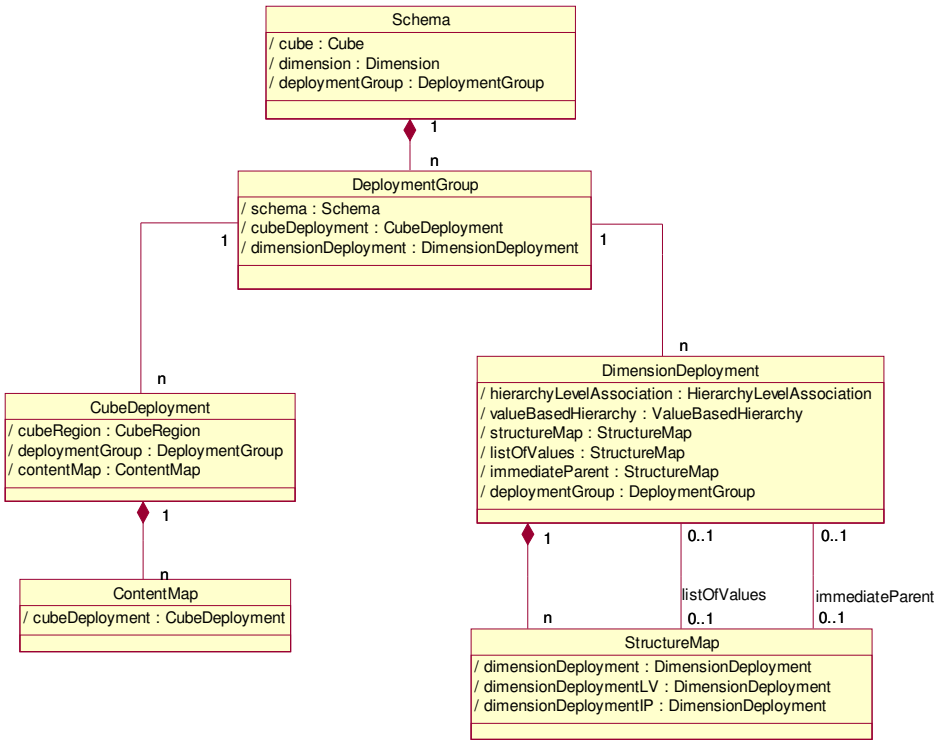


Figura 40 - Metamodelo *OLAP*: Estruturas de mapeamento de implantação

Fonte: OMG, 2002a, pág. 367.

O **metamodelo *Data mining*** contém descrições dos resultados das atividades de *data mining* representando os modelos que eles descobrem e os valores dos atributos que tiverem sido utilizados na exploração. O modelo (Figura 41) permite a representação do resultado de uma operação de *data mining*, isto é, o modelo matemático de algum aspecto da informação descrita no CWM.

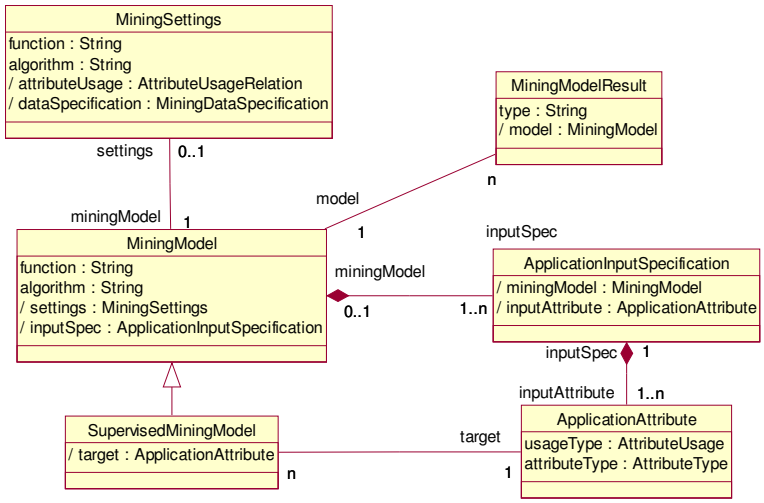


Figura 41 - Metamodelo *Data Mining*

Fonte: OMG, 2002a.

A área *settings* (figura 42) descreve os parâmetros e valores de entrada para atributos que foram ou serão utilizados para construir um modelo *mining*.

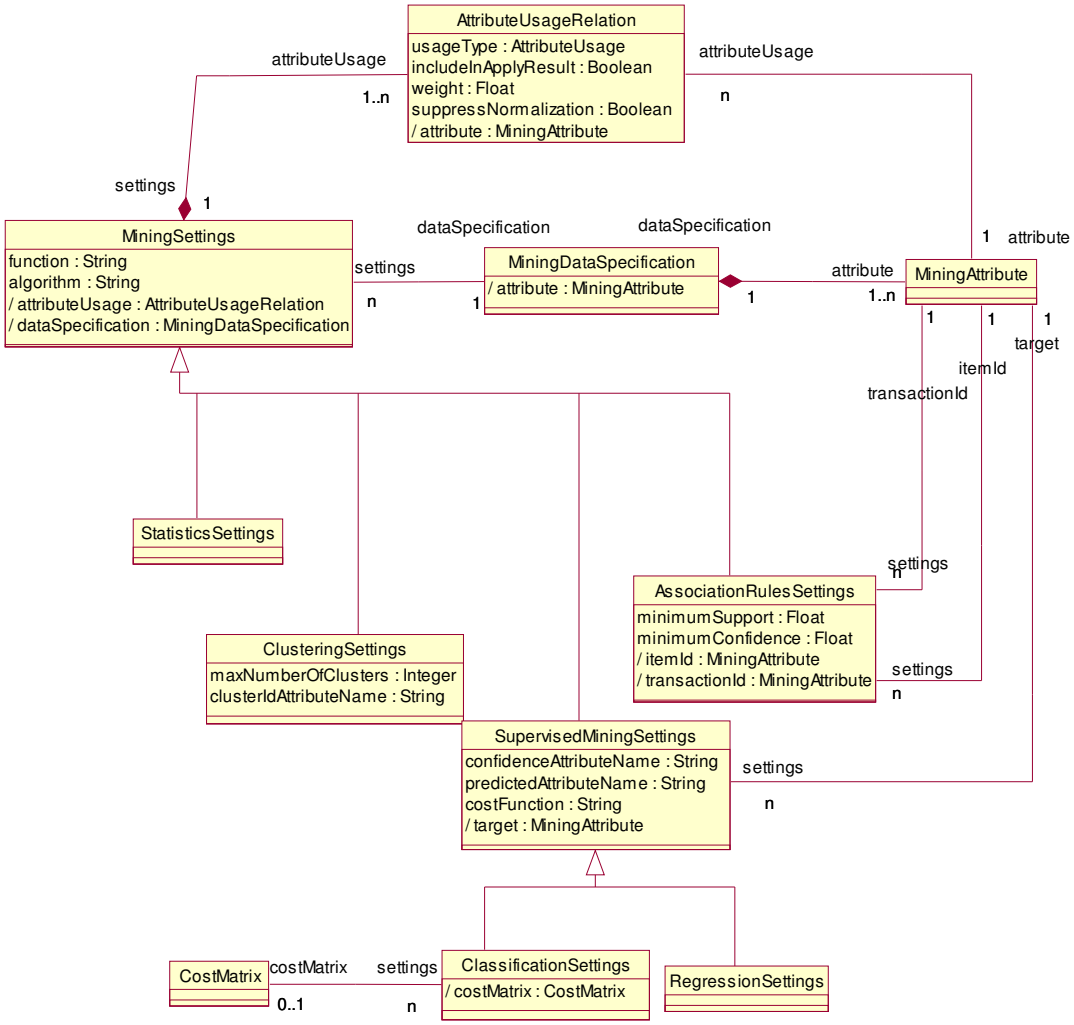


Figura 42 - Metamodelo *Data Mining – Settings*

Fonte: OMG, 2002a.

A área *attributes* (figura 43) permite a inclusão de atributos numéricos e categóricos.

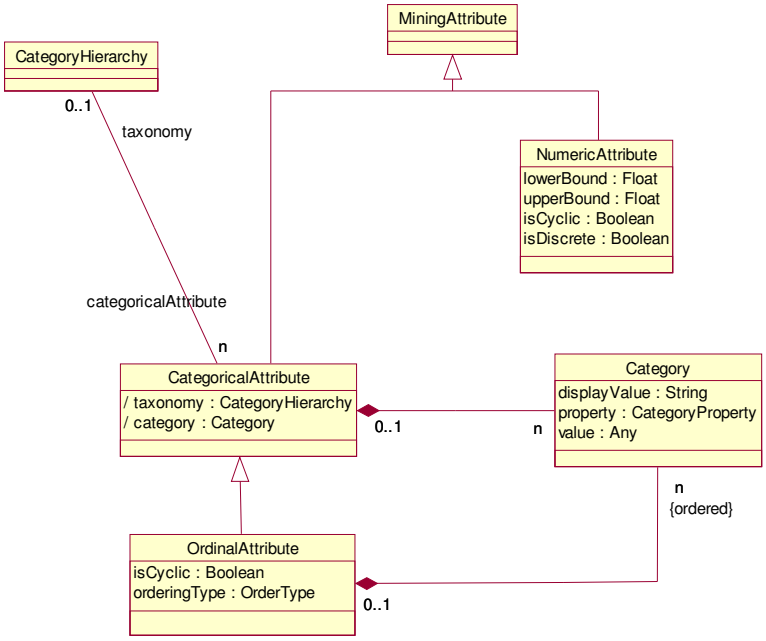


Figura 43 - Metamodelo *Data Mining - Attributes*

Fonte: OMG, 2002a.

O **metamodelo *Information Visualization*** (Figura 44) define metadados que suportem a publicação da informação ou, de forma mais geral, a visualização da informação. Define construções de metadados como containers, bastante genéricos, que possam conter ou referenciar mecanismos de visualização mais complexos.

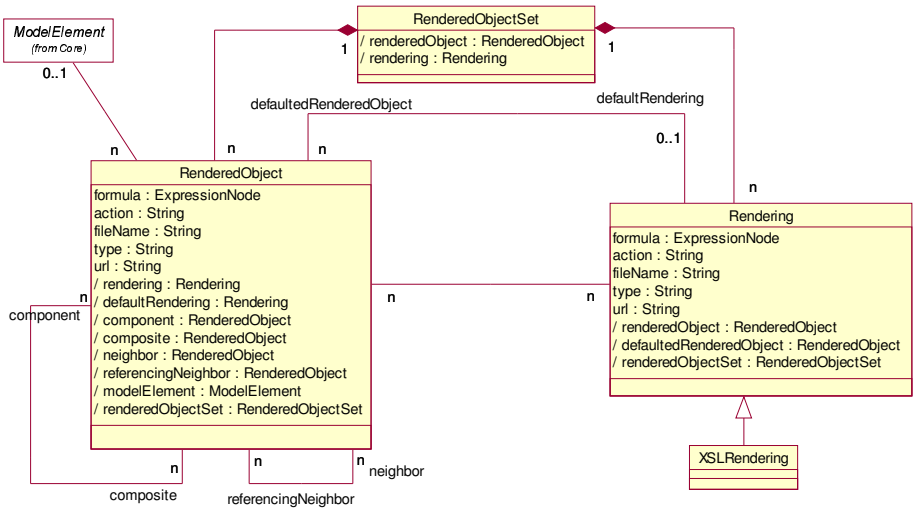


Figura 44 - Metamodelo *Information Visualization*

Fonte: OMG, 2002a, pág. 538.

O **metamodelo *BusinessNomenclature*** (Figura 45) contém classes e associações que podem ser utilizadas para representar metadados de negócios. O acesso fácil a tais metadados de negócios permite que os usuários de negócios explorem o valor da informação em um *data warehouse*. E também pode ajudar os usuários técnicos em determinadas tarefas. Este pacote abrange somente os domínios de *data warehouse* e *business intelligence*.

São fornecidas duas construções principais para representar os termos e conceitos de negócios e as semânticas relacionadas:

- ***Taxonomia*** – é uma coleção de conceitos que fornecem o contexto para o significado de um determinado termo.
- ***Glossário*** **–** é uma coleção de termos e várias formas relacionadas do termo.

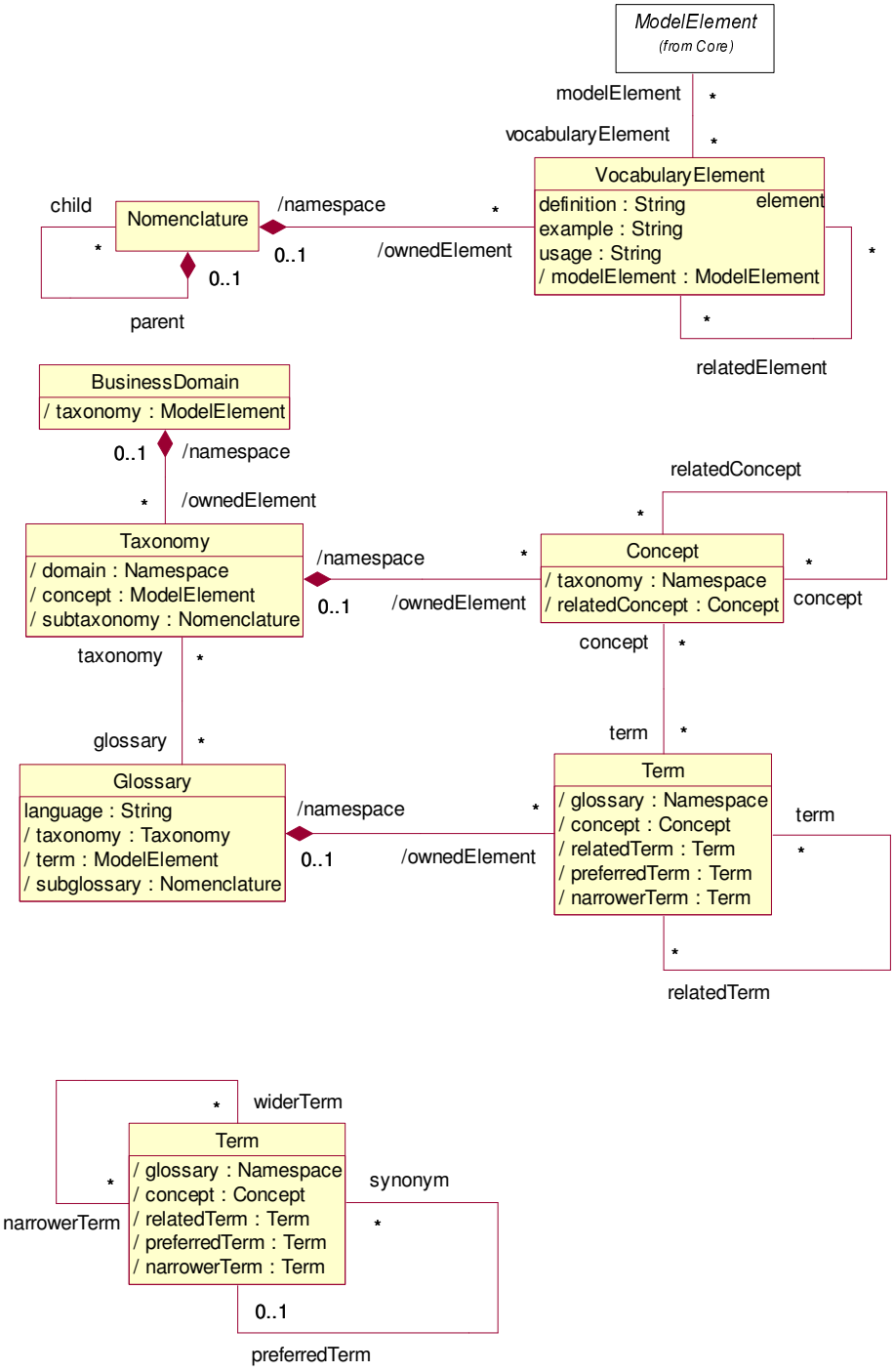


Figura 45 - Metamodelo *BusinessNomenclature*

Fonte: OMG, 2002a, pág. 555.

3.3.5 Camada Management

Esta camada define dois metamodelos que são críticos para a definição de metadados e que descrevem o processamento da cadeia de informações como um todo.

O **metamodelo *Warehouse Process*** (Figura 46) documenta o fluxo de processo utilizado para executar transformações. Tais fluxos podem ser documentados no nível de uma atividade de transformação completa ou de seus passos de transformação individuais.

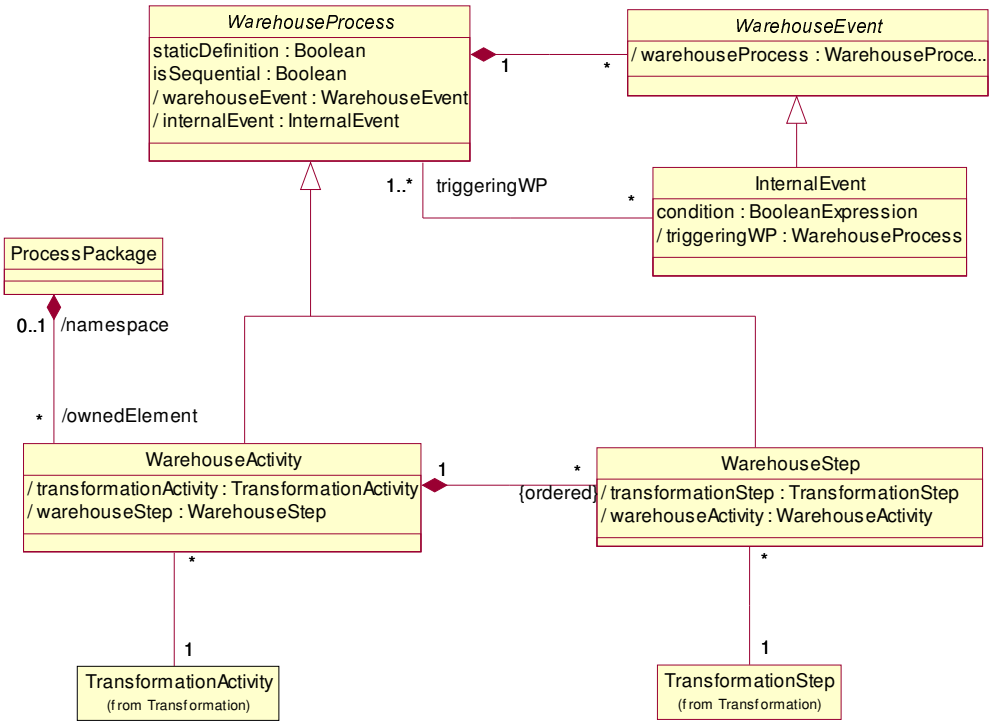


Figura 46 - Metadado *Warehouse Process*

Fonte: OMG, 2002a, pág. 578.

O objeto *WarehouseProcess* associa uma transformação a um conjunto de eventos que serão utilizados para disparar a execução da transformação. Este objeto pode ser associado a um ou mais *WarehouseEvents* (Figura 47), e tais eventos podem ser divididos em:

- ***Agendado*** – são eventos que podem ser definidos como um ponto no tempo ou por intervalos. Um evento tipo ponto no tempo pode ser definido como um calendário configurável que contém um conjunto de datas, e estas podem sejr reutilizadas entre vários processos.
- ***Externo*** – os eventos são disparados por algo que aconteça fora do *data warehouse*.
- ***Interno -*** são eventos disparados pelo término de um processo. Eles podem ser eventos de tentativas ou em cascata.

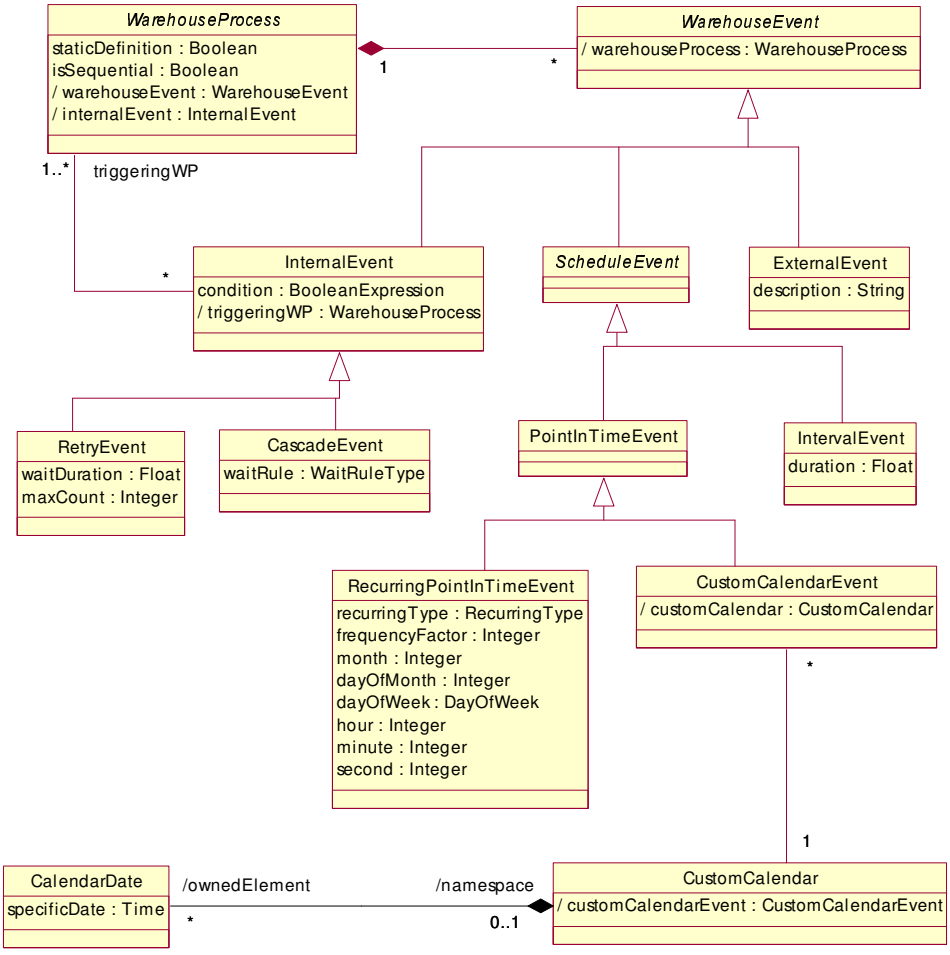


Figura 47 - Metamodelo *Warehouse Process* - Eventos e Calendário

Fonte: OMG, 2002a, pág. 579.

O **metamodelo *Warehouse Operation*** contém classes que armazenam as operações do dia-a-dia dos processos de *warehouse*, e cobre as áreas a seguir:

- ***Execuções de Transformações* –** podem ser gravados detalhes das mais recentes execuções de transformações, identificando quando foram executadas e se foram bem sucedidas (Figura 48). A linhagem do dado em um *data warehouse* pode ser preservada, armazenando-se quando e como ele foi derivado, e de onde ele veio.
- ***Medidas* –** Permitem que sejam mantidas métricas para quaisquer elementos do modelo (Figura 49). Por exemplo, pode-se utilizá-las para armazenar o valor atual, o estimado e o planejado para o tamanho de uma tabela.
- ***Requisições de Mudança* –** Permite que sejam gravados detalhes de mudanças propostas que afetem qualquer elemento do modelo (Figura 50). Também podem ser utilizadas para manter um registro histórico das mudanças implementadas ou rejeitadas.

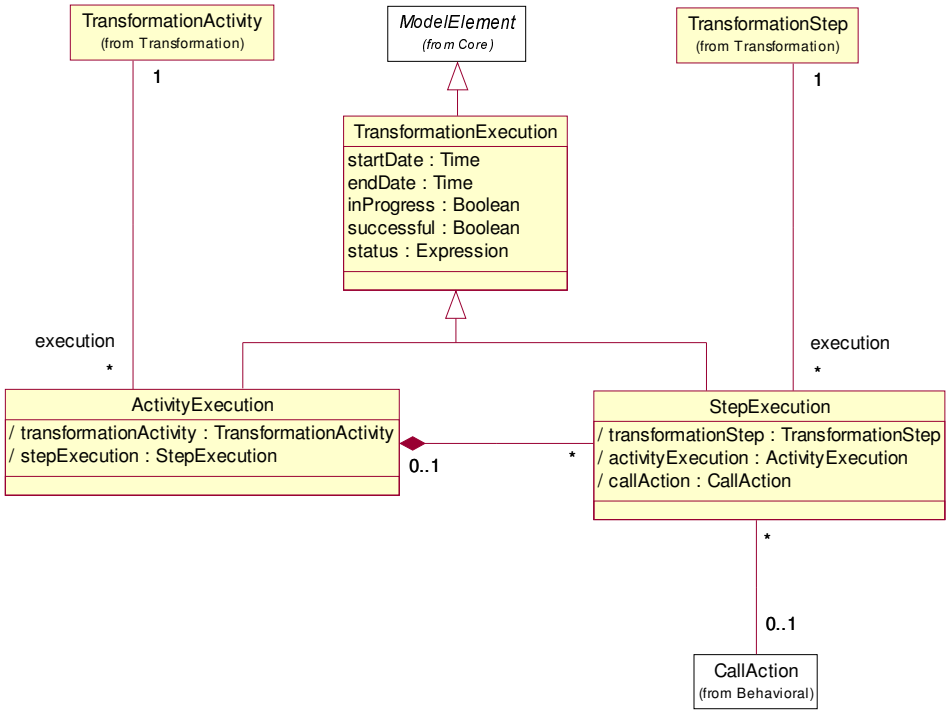


Figura 48 - Metamodelo *Warehouse Operation* - Execuções de Transformações

Fonte: OMG, 2002a, pág. 600.

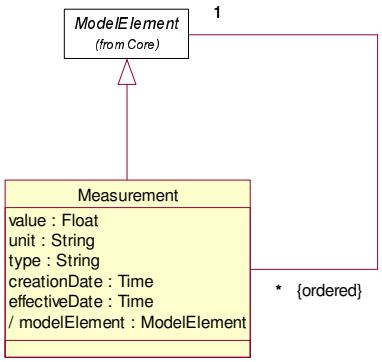


Figura 49 - Metamodelo *Warehouse Operation* – Medidas

Fonte: OMG, 2002a, pág. 601.

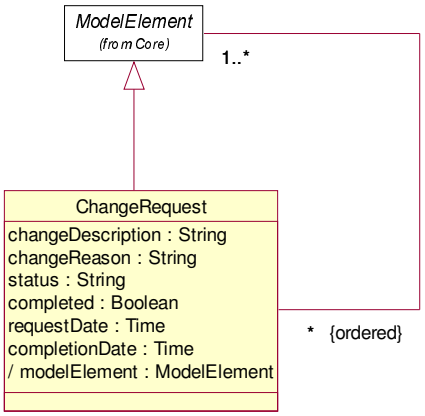


Figura 50 - Metamodelo *Warehouse Operation* - Requisições de mudanças

Fonte: OMG, 2002a, pág. 601.

3.4 Tecnologias para Fundamentação

O objetivo do CWM é padronizar o formato de intercâmbio de metadados compartilhados para *data warehouse* e *business intelligence* e padronizar a API de linguagem de programação para o acesso de tal metadado. O CWM, cuja abordagem é dirigida a modelo, consiste dos seguintes passos:

1. define em UML o modelo de metadado compartilhado para CWM.
2. gera em XML a especificação do formato de intercâmbio para metadado CWM.
3. gera em CORBA IDL a especificação da API da linguagem de programação para o acesso do metadado CWM.

Uma ferramenta não precisa entender o formato de metadado proprietário de cada ferramenta com a qual ela opera. Ela somente precisa entender os modelos padrões e as interfaces do armazém compartilhado. Isto reduz consideravelmente os custos de se desenvolver e implantar ferramentas de *data warehouse* (POOLE, 2002).

3.4.1 Unified Modeling Language (UML)

A especificação UML 1.3 consiste das seguintes partes:

- ***Semântica UML –*** Define a semântica do metamodelo UML. O metamodelo UML possui camadas arquiteturais e é organizado por pacotes. Em cada pacote, os elementos do modelo são definidos em termos de sintaxe abstrata (utilizando diagramas de classes), regras bem formadas (em OCL), e semânticas (em inglês).
- ***Guia de Notação UML –*** Especifica a sintaxe gráfica (por exemplo, diagrama de classe) para expressar a semântica do metamodelo UML.
- ***Especificação de Linguagem de Restrição de Objeto –*** Define a sintaxe, semântica e gramática da *Object Constraint Language* (OCL), a qual é uma linguagem formal para expressão de restrições.
- ***Especificação XMI DTD UML –*** Utiliza XML DTD para definir um mecanismo para intercâmbio de modelos UML que estejam em conformidade com o metamodelo UML.
- ***Definição da Interface CORBAfacility UML –*** Utiliza IDL para especificar um repositório que permita a criação, armazenamento e manipulação de modelos UML.
- ***Perfis Padrões UML –*** Define o Perfil UML para Processos de Desenvolvimento de Software e o Perfil UML para Modelagem de Negócios.

CWM depende somente das três primeiras partes da especificação UML 1.3. Os blocos básicos de construção da UML são os seguintes:

- ***Elementos do Modelo –*** Conceitos comuns orientados a objetos como classes, objetos, interfaces, componentes, *use cases*, mensagens, estados etc.
- ***Relacionamentos –*** Conexões entre elementos do modelo tais como associações, generalização, dependências, etc.
- ***Diagramas –*** Grupos de símbolos gráficos que podem ser utilizados para representar elementos do modelo e seus relacionamentos, tais como diagramas de classes, diagramas de objetos, diagramas de *use case*, etc.

A UML pode ser utilizada para modelar aspectos diferentes de um sistema incluindo os seguintes:

- ***Modelagem Estrutural –*** Enfatiza a estrutura dos objetos do sistema, incluindo suas classes, relacionamentos, atributos, e operações. Modelagem estrutural consiste de modelagem da estrutura estática utilizando-se diagramas de classes e diagramas de objetos, e modelagem da implementação utilizando-se diagramas de componentes e diagramas de *deployment*.
- ***Modelagem Use case –*** Enfatiza a funcionalidade do sistema como ele aparece para usuários finais. Um modelo de *use case* particiona a funcionalidade do sistema em transações (*use cases*) que são significantes para os usuários (atores).
- ***Modelagem Comportamental –*** Enfatiza o comportamento dos objetos do sistema, incluindo suas interações, eventos e fluxo de dados e de controle. Modelagem Comportamental consiste de modelagens de interação utilizando-se diagramas de seqüência e diagramas de colaboração, modelagem de eventos utilizando-se diagramas de estado, e modelagem de fluxo de dados e de controle utilizando-se diagramas de atividades.

Sendo que dentre estes, somente o aspecto da modelagem da estrutura estática é necessário para a modelagem de metadados.

3.4.2 Extensible Markup Language (XML)

XML (um padrão W3C[7]) disponibiliza o formato universal para intercâmbio de metadado CWM. A especificação XML define as regras de "bem formação" para documentos XML, as regras gramaticais para DTDs, e as regras de validação para a validação de documentos XML com seus DTDs. Já a especificação XML *Namespace* define o mecanismo para prover suporte ao *Namespace* e documentos e DTDs XML.

Recentemente o W3C adotou o XML *Schema* como um padrão, cujo objetivo é disponibilizar uma forma alternativa ao DTD para definição das estruturas dos documentos XML. Há muitas vantagens do XML *Schema* sobre o DTD, incluindo as seguintes:

- DTD é baseado em uma sintaxe especializada; XML *Schema* é baseado em XML.
- DTD trata todos os dados (com poucas exceções) com *strings* ou *strings* enumeradas; XML *Schema* suporta um conjunto rico de tipos de dados, comparáveis àqueles em SQL e Java.
- DTD utiliza um modelo de conteúdo fechado que permite muito pouca extensibilidade; XML *Schema* utiliza um modelo de conteúdo aberto, o qual permite a extensão de vocabulários e estabelece relacionamentos de substituição entre elementos.
- DTD permite somente uma associação entre um documento e seu DTD através da declaração do tipo de documento; XML *Schema* suporta integração de *Namespace*,

[7] http://www.w3.org/

o que permite a associação de nós individuais de um documento com declarações de tipo em um esquema.

Devido a estas vantagens, para o futuro espera-se que o XML *Schema* seja utilizado, ao invés do DTD, para a definição das estruturas dos documentos XML, principalmente naqueles utilizados para troca de dados ou metadados. XMI 2.0 irá definir regras que podem ser utilizadas para gerar automaticamente o CWM XSD[8], que seria a definição XML *Schema* para CWM.

XML é uma linguagem para definição de linguagens de marcação (por exemplo, HTML). XML permite que marcadores de metadados sejam embutidos em documentos web (STAUDT, 1999). Descreve uma classe de objetos de dados, documentos XML, que são escritos utilizando-se essas linguagens de marcação e descreve o comportamento de programas de computador que os processa. XML é projetada para utilização na Internet. Documentos XML são formais e concisos (isto é, processáveis através de máquinas) e razoavelmente claros (legíveis por humanos). Também são fáceis de criar e processar. Além do que, desde sua criação em 1998, XML tem rapidamente se tornado o formato universal para intercâmbio de dados e intercâmbio de aplicações.

CWM é baseado em XML, padrão adotado pelo W3C em fevereiro de 1999. Sua especificação consiste das seguintes partes:

- ***Definição do Documento XML –*** Define as estruturas físicas e lógicas de um documento XML.
- ***Definição Document Type Definition (DTD) –*** Define as regras gramaticais que podem ser utilizadas para definir marcações de documentos.

Um documento XML pode conter elementos e atributos que são definidos para e utilizados por múltiplos sistemas de software. Tais documentos, contendo múltiplos vocabulários de marcação, podem sofrer problemas de identificação e colisão. Sistemas de software devem ser capazes de reconhecer elementos e atributos para os quais eles são projetados a processar, mesmo em face de colisões ocorrendo quando marcadores planejados para utilização de outros sistemas utilizem o mesmo tipo de elemento ou nome de atributo. *Namespaces* XML fornece o mecanismo para que tipos de elementos e atributos tenham nomes universais, cujo escopo se estende além do seu documento.

3.4.3 Meta Object Framework (MOF)

MOF é um metametamodelo com semântica suficiente para descrever metamodelos em vários domínios. Seu objetivo principal é o de fornecer um conjunto de interfaces CORBA que possam ser utilizadas para definir e manipular um conjunto de metamodelos interoperáveis. (STAUDT, 1999)

MOF é um *framework* de objetos distribuído, dirigido a modelo, para especificação, construção, gerenciamento, intercâmbio, e integração de metadados em sistemas de software. O objetivo do *framework* é suportar qualquer tipo de metadado e permitir que novos tipos de metadados sejam adicionados. Para isso MOF utiliza uma arquitetura de metadados de quatro camadas, denominada Arquitetura de Metadados OMG (vide Tabela 2).

---

[8] XML *Schema Specification* (XMI, 2003)

Tabela 2 - Arquitetura de Metadados OMG

META-NÍVEL	TERMOS MOF	EXEMPLOS
M3	Meta-metamodelo	Modelo MOF
M2	Metamodelo, meta-metadado	Metamodelo UML, Metamodelo CWM
M1	Modelo	modelos UML
M0	Objeto, dados	Sistemas modelados, dados de *warehouse*

Fonte: OMG, 2002a, pág. 38.

Esta arquitetura trata metadados (M1) como dados (M0) e formalmente modela cada tipo distinto de metadado. Estes modelos formais, os chamados metamodelos (M2), são expressos utilizando as construções de meta-modelagem fornecida por um único meta-metamodelo (M3), o qual é denominado Modelo MOF.

O *Meta Object Facility* (MOF) da OMG fornece uma linguagem abstrata que assegura que todas as ferramentas de software irão interpretar o metamodelo comum da mesma forma. MOF é um padrão OMG que define uma linguagem abstrata comum para a especificação de metamodelos. Define os elementos essenciais, sintaxe e estrutura de metamodelos que são utilizados para construir modelos de sistemas discretos. MOF serve como o modelo comum do CWM e da UML. O poder do MOF é que ele permite que metamodelos diferentes, representando domínios diferentes, sejam utilizados de forma interoperável. Aplicações MOF podem não ter nenhum conhecimento das interfaces específicas de domínio de alguma instância do modelo mas ainda pode ler e atualizar tal modelo utilizando as operações genéricas das interfaces refletivas.

A semântica MOF geralmente define certos serviços de repositórios de metadados que suportam construção, descoberta, passagem e atualização do modelo. MOF define semântica do *ciclo de vida do modelo,* definindo funções de autoria e publicação efetivos de metadados, especificamente se combinados com suporte à modelagem visual.

Metamodelos recentemente desenvolvidos podem ser persistidos no repositório MOF e combinados com outros metamodelos existentes. Um repositório MOF fornece vários serviços importantes de metadados que vão além da construção e apresentação de metadado (ex.: persistência, versionamento, e serviços de diretório).

CWM é baseado em MOF, o qual foi adotado pela OMG em Setembro de 1999. A especificação MOF consiste das seguintes partes:

- ***Modelo MOF –*** Define os elementos de modelagem, incluindo as regras para sua utilização, as quais podem ser utilizadas para a construção de metamodelos.
- ***Interfaces Refletivas MOF –*** Permitem a um programa criar, atualizar, acessar, navegar, e invocar operações nos metadados sem a utilização de interfaces específicas do metamodelo.
- ***Mapeamento MOF para IDL –*** Define o mapeamento padrão de um metamodelo definido utilizando o Modelo MOF em CORBA IDL, permitindo assim a geração automática de interfaces específicas de metamodelos para acesso e manipulação de metadados.

O Modelo MOF é baseado nos conceitos e construções da UML, particularmente seu modelo de estrutura estática e gerenciamento de modelo. O Modelo MOF não define sua própria notação gráfica ou linguagem de restrição mas utiliza a notação UML e OCL para tais propósitos, respectivamente. Como o metamodelo UML, o Modelo MOF é arquiteturalmente estendido em camadas e organizado em pacotes. Em cada pacote, os elementos do modelo são definidos em termos de sintaxe abstrata (utilizando digramas de classe), regras bem formadas (em OCL), e semântica (em Inglês).

Os principais elementos do Modelo MOF são: classes, objetos, atributos e operações. Tais conceitos e construções são idênticos àqueles do modelo de estrutura estática da UML porém em um meta nível mais alto. Isto é, classes, objetos, atributos, e operações MOF estão no nível M3 e são utilizados para definir metamodelos (M2); ao passo que classes, objetos, atributos e operações UML estão no nível M2 e são utilizados para definir modelos (M1).

Os principais relacionamentos do Modelo MOF são: associações, agregação e generalização. Tais conceitos e construções são novamente idênticos àqueles do modelo de estrutura estática da UML porém em um meta nível mais alto. Isto é, associações, agregações e generalizações MOF são utilizadas para definir metamodelos (M2); ao passo que associações, agregações e generalizações UML estão no nível M2 e são utilizadas para definir modelos (M1).

Apesar de o Modelo MOF ser idêntico ao modelo de estrutura estática UML e gerenciamento de modelo na maioria dos conceitos e estruturas, existem algumas diferenças importantes. Primeiro, uma associação MOF deve ser binária, isto é, deve ser definida entre duas classes e possuir dois finais de associação. Cada fim de associação possui um nome, tipo, e multiplicidade. Segundo, uma associação MOF não pode ter uma classe anexada (a classe de associação na UML). Terceiro, uma classe MOF pode ter referências. Uma referência tem um nome, e define o conhecimento da classe de, e o acesso a, *links* que são instâncias de uma associação.

Cada metamodelo MOF tem uma representação como um XML DTD (de acordo com as regras do XMI), e também uma definição IDL. Esses DTDs são relevantes quando os modelos CWM são serializados e intercambiados entre as ferramentas como documentos XMI. A IDL é relevante na construção de modelos de objetos CWM em memória ou no armazenamento destes em repositórios, já que ela define as interfaces, assinaturas de métodos e estrutura de pacotes que o modelo deve suportar (POOLE, 2002).

O mapeamento MOF para IDL define o mapeamento padrão de um metamodelo que tiver sido definido utilizando-se o Modelo MOF em CORBA IDL. As interfaces resultantes permitem ao usuário criar, acessar, e atualizar instâncias do metamodelo utilizando programas clientes CORBA.

3.4.4 XML Meta Data Interchange (XMI)

A abordagem baseada em modelo para integração de metadado requer um formato de intercâmbio comum para troca de instâncias de metadados compartilhados, e também de uma interface comum de programação para acesso ao metadado. A codificação para intercâmbio utilizada pela CWM é a XMI, um padrão OMG que define um mapeamento formal de metamodelos MOF. XMI integra as três tecnologias de fundamentação discutidas anteriormente: UML, MOF e XML. XMI permite ao metadado MOF, ser trocado como *streams* ou arquivos com um formato padrão baseado em XML. XMI define precisamente como *tags* XML podem ser utilizadas para armazenar instâncias de metamodelos CWM em documentos XML.

O metamodelo CWM é utilizado para definir uma coleção de *tags* XML expressas na forma de um DTD XML. Os metadados CWM são serializados em documentos XML. Cada instância do metadado é armazenada como conteúdo de um elemento XML, delimitado por *tags* apropriadas de metamodelo.

XMI consegue resolver muitos dos problemas encontrados ao se tentar utilizar uma linguagem baseada em marcação para a representação de objetos e de suas associações. O fato de XMI ser uma forma da utilização de XML significa que tanto a *tag* quanto os itens descritos pela *tag* (conteúdo do elemento) podem ser empacotadas no mesmo documento, permitindo que as aplicações entendam rapidamente o conteúdo do documento. A

comunicação do conteúdo é tanto autodescritiva como assíncrona por natureza, que são os principais motivos do XMI e do intercâmbio baseado em XML serem tão importantes em ambientes distribuídos e heterogêneos.

XMI é um padrão OMG que mapeia o MOF para o XML. Define como as *tags* XML são utilizadas para representar modelos MOF serializados. Os metamodelos baseados em MOF são traduzidos para XML DTD e os modelos são traduzidos em Documentos XML que são consistentes com seus DTDs correspondentes. (POOLE, 2001)

XMI suporta o intercâmbio de metadado completo ou apenas de fragmentos de metadados. Cada documento XML que contém metadados XMI pode conter: elementos XML que são requeridos por XMI, elementos XML que contêm metadados em conformidade com o MOF, e, opcionalmente, elementos XML que contêm metadados que representam extensões do metamodelo. O mecanismo *Namespace XML* permite ao XMI utilizar vários metamodelos ao mesmo tempo em um documento XML.

CWM é baseado em XMI, que foi adotado pela OMG em Outubro de 1999. A especificação XMI consiste das seguintes partes:

- ***Produção DTD XML –*** Regras para transformação de metamodelos MOF em DTDs XML.
- ***Princípios de Projeto DTD XML –*** Para produção de DTDs XML.
- ***Produção de Documentos XML –*** Regras para codificar e descodificar Metamodelos MOF em documentos XML.
- ***Princípios de Geração XML –*** Para produção de documentos XML.

Um DTD XML fornece a forma pela qual um processador XML pode validar a sintaxe e semântica estrutural de um documento XML. XMI fornece regras pelas quais um DTD pode ser gerado por qualquer metamodelo MOF. Entretanto, a utilização do DTD é opcional; um

documento XML não precisa referenciar um DTD, mesmo que exista um. Executar validação XML no documento XML contendo metadado MOF pode ser vantajoso. Se um DTD é referenciado no documento XML, qualquer processador XML pode executar qualquer verificação, aliviando as ferramentas de importação e exportação de metadados da carga de executar essas checagens.

3.5 CWM IDL

O acesso a recursos de metadados CWM é definido através de mapeamentos padronizados de metamodelos MOF para várias linguagens de programação. A especificação MOF, em particular, define um mapeamento de qualquer metamodelo MOF, como CWM, para a IDL (uma notação de linguagem neutra para especificação de interfaces) da OMG. A especificação CWM contempla a definição completa da IDL para CWM. Definir uma interface em alguma linguagem de programação de escolha (ex.: Java ou C++) requer a compilação da CWM IDL em definições de interface expressas na sintaxe da linguagem alvo, utilizando um compilador IDL apropriado para a linguagem alvo.

Porém somente UML e XML não são suficientes para o sucesso do CWM. A UML é uma linguagem de modelagem para metadado; XML é um formato de intercâmbio para metadado. Eles são entidades disjuntas. Para juntá-los de tal forma que alguém possa começar com a modelagem de metadados utilizando UML e terminar com intercâmbio de metadado modelado em XML, CWM utiliza uma abordagem dirigida a modelo. O desenvolvimento da especificação CWM inicia com o metamodelo CWM, o qual é representado em notação UML utilizando uma ferramenta tal como a Rational Rose. Quando o metamodelo CWM é desenvolvido, é utilizado para desenvolve a especificação CWM em fomato texto adicionando texto Inglês e restrições OCL quando apropriado. O metamodelo CWM é assim utilizado para

gerar automaticamente um CWM XML, um CWM DTD, e um CWM IDL, de acordo com XMI e MOF, respectivamente. O documento CWM XML é gerado pelo MOF DTD e representa uma definição formal, independente de ferramenta do metamodelo CWM. O CWM DTD pode ser utilizado para validar documentos XML utilizados para intercâmbio de metadado CWM. O CWM IDL define uma API independente de linguagem para acesso de metadado CWM.

3.6 Resumo dos Padrões OMG

Os seguintes padrões OMG são tecnologias de base para CWM: UML, IDL, MOF e XMI, e em conjunto elas formam o núcleo da padronização de metadados adotada pela OMG. UML e MOF, juntamente com CWM, também formam os componentes núcleos da OMG MDA[9].

UML disponibiliza a notação UML e OCL como parte da linguagem de metamodelagem para CWM. Adicionalmente, o metamodelo UML, ou um subconjunto dela, foi utilizado como base de projeto para o pacote CWM *ObjectModel*, o qual serve tanto como metamodelo base como metamodelo de objeto para o CWM.

MOF disponibiliza a arquitetura de metamodelagem para CWM. Especificamente, o modelo MOF provê a semântica da linguagem de metamodelagem para CWM; as interfaces refletivas MOF provêm APIs genéricas para acesso aos metadados CWM; e o mapeamento MOF para IDL provê o mecanismo pra gerar interfaces CWM IDL para acesso aos metadados CWM.

---

[9] *Model Driven Architecture* (MDA) é uma abordagem que define uma técnica de especificação de sistemas que separa a especificação da funcionalidade do sistema da especificação da implementação e da estrutura tecnológica.

XMI disponibiliza o mecanismo de intercâmbio de metadados para CWM. Especificamente, XMI define regras que podem ser utilizadas para gerar automaticamente o CWM DTD, o qual pode ser utilizado para validar quaisquer documentos XML que contenham metadados CWM. XMI também define regras que podem ser utilizadas para gerar documentos XML que contenham metadados CWM e que sejam validados pelo CWM DTD.

A IDL provê uma linguagem de definição de interface para utilização no ambiente CORBA. É independente de linguagem de programação. Qualquer linguagem de programação que tiver um mapeamento IDL definido e suporte a CORBA pode ser utilizada para implementar interfaces IDL.

Os relacionamentos entre CWM e UML, IDL, MOF, e também XMI são resumidos na Tabela 3.

Tabela 3 - Padrões CWM e OMG

CATEGORIA	PADRÃO	TECNOLOGIA BASE PARA CWM
OMG	UML	Notação UML OCL Metamodelo UML
	IDL	Definição CORBA IDL
	MOF	Modelo MOF Refletivo MOF Mapeamento MOF-para-IDL
	XMI	Produção XML DTD Produção de documento XML Produção de XML *Schema* (futuro)

Fonte: POOLE, 2002, pág. 247.

3.7 Extensões ao CWM

Metadados que não sejam compatíveis com o formato CWM (ex.: metadado altamente específico de uma determinada ferramenta) são manuseados através de mecanismos de extensão padrões, fornecidos pelo CWM através de extensões ao metamodelo núcleo do

CWM, ou através da utilização de padrões específicos de produtos, entradas de usuários, ou alguma outra lógica de implantação definida. (POOLE, 2001)

Uma solução de integração de metadados baseada em modelo também deve fornecer alguma forma padrão para a extensão de modelos. Esse mecanismo de extensão é necessário para a definição de metadados específicos de produtos não considerados pelo CWM. Também é necessário poder estender o metamodelo para permitir a inclusão de novos metamodelos representando subdomínios adicionais que se possa querer incluir na solução da cadeia de informações. Devido ao CWM ser baseado em UML, pode-se contar com mecanismos padrões de extensão UML. Tais mecanismos padrões de extensão consistem dos elementos de modelagem *Tagged Value, Stereotype* e *Constraint*. Esses elementos são definidos com o metamodelo *Core* (núcleo) da camada CWM *ObjectModel*.

3.8 CWM e Outros Padrões

Para que CWM continue seu sucesso inicial e prospere, deve envolver com outros padrões relevantes, ambos existentes e emergentes. Mesmo que *data warehouse* e *business intelligence* sejam partes principais de uma infra-estrutura de TI da empresa, eles são somente partes daquela infra-estrutura e não ela inteira. Portanto, CWM, sendo o padrão para intercâmbio de metadado para *data warehouse* e *business intelligence*, deve ser consistente com e complementar outros padrões relevantes na infra-estrutura de TI das empresas.

3.8.1 OMG MDA

Em parte como um resultado do sucesso da abordagem dirigida a modelo CWM, o OMG tem embarcado recentemente em uma nova direção estratégica em *Model-Driven Architecture* (MDA), como mostrado na Figura 51.

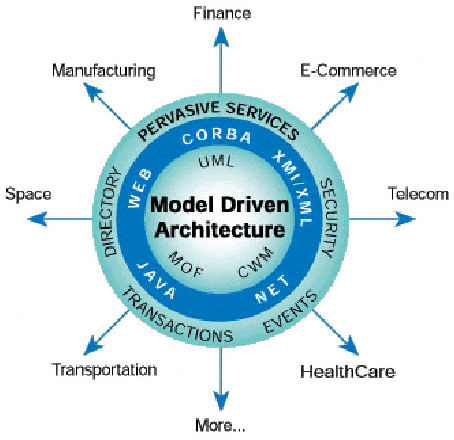


Figura 51 - OMG MDA

Fonte: MDA, 2003.

MDA é uma abordagem para especificação de sistema e interoperabilidade baseada na utilização de modelos formais (MDA, 2003). Esta abordagem leva a integração e interoperabilidade incrementando o ciclo de vida de um sistema de software de modelagem e projeto, para construção de componente, integração, implantação, gerenciamento, e evolução. As definições de modelos e seus subtipos em MDA são as seguintes:

- **Modelo** – Uma especificação formal da função, estrutura, e comportamento de um sistema de software.
- **Modelo independente de plataforma** – É um modelo que abstrai detalhes tecnológicos e de engenharia.
- **Modelo específico de plataforma** – É um modelo que contém detalhes tecnológicos e de engenharia.

Apesar de os modelos específicos de plataforma serem necessários para implementar tais modelos, os modelos independentes de plataforma são úteis por duas razões básicas. Primeiro, por abstrair a estrutura fundamental e o significado, eles tornam mais fáceis de validar a corretude do modelo. Segundo, mantendo a estrutura essencial e o significado do sistema sem variações, eles facilitam a produção de implementações em plataformas diferentes.

Existem três formas básicas de se construir modelos específicos de plataforma a partir de um modelo independente de plataforma:

- pode uma pessoa estudar o modelo independente de plataforma e construir manualmente um modelo específico de plataforma.
- pode-se aplicar um algoritmo ao modelo independente de plataforma resultando no esqueleto de um modelo específico de plataforma que é alterado manualmente pelo programador.
- pode-se aplicar um algoritmo ao modelo independente de plataforma, gerando-se um modelo específico de plataforma completo.

Transformações completamente automatizadas a partir de modelos independentes de plataforma para modelos específicos de plataforma são viáveis somente em certos ambientes restritos.

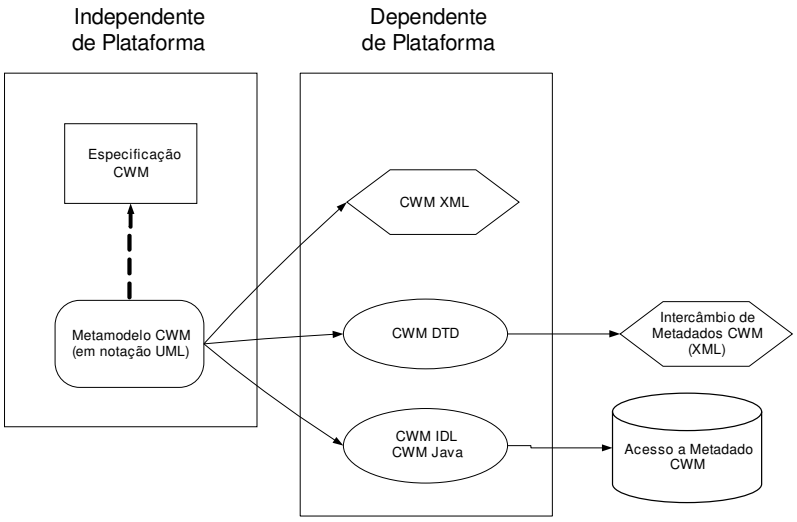


Figura 52 - CWM e MDA: perspectiva de plataforma tecnológica

Fonte: POOLE, 2003, pág. 673.

O metamodelo, a especificação, e os artefatos gerados pelo CWM cabem no MDA muito bem e podem ser observados sob duas perspectivas diferentes. Sob a perspectiva de plataforma tecnológica (isto é, se pensarmos em XML, CORBA, e Java como plataformas tecnológicas diferentes), os relacionamentos entre o modelo independente de plataforma (o metamodelo e especificação CWM) e os modelos específicos de plataforma (CWM XML, CWM DTD, CWM IDL, e, no futuro, CWM Java) são mostrados na Fig. 52. Todos os modelos específicos de plataforma neste caso são gerados completamente e automaticamente a partir do modelo independente de plataforma.

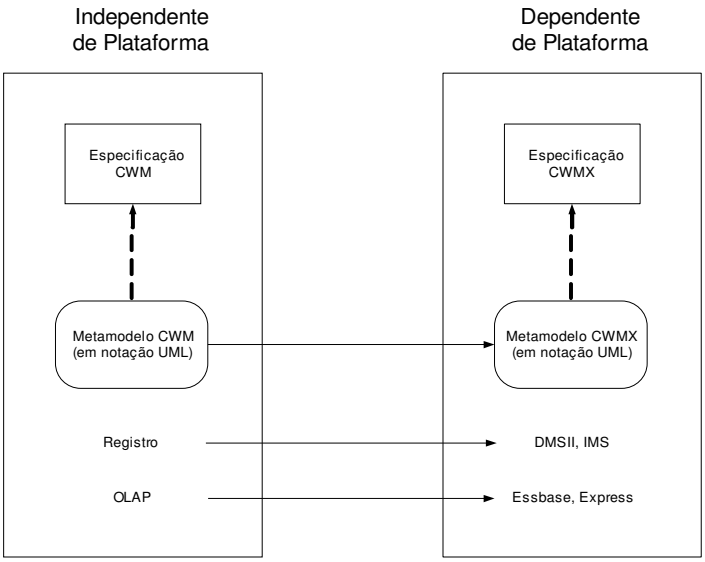


Figura 53 - CWM e MDA: perspectiva de plataforma de produto

Fonte: POOLE, 2003, pág. 674.

Sob a perspectiva de plataforma de produto (isto é, considerando-se DMSII, IMS, Essbase, Express, e assim por diante como plataformas de produtos diferentes), os relacionamentos entre o modelo independente de plataforma (o metamodelo e especificação CWM) e os modelos específicos de plataforma (DMSII, IMS, Essbase, Express, e assim por diante) são mostrados na Fig. 53. Todos os modelos específicos de plataforma neste caso são construídos pelos próprios desenvolvedores a partir dos modelos independentes de plataforma.

3.8.2 Extensão para Java

Ultimamente vêm sendo feitos esforços junto ao *Java Community Process* (JCP) com o objetivo de desenvolver modelos de programação Java traduzindo os padrões OMG na forma de APIs padrões J2EE (ex.: JMI, JOLAP e JDMAPI) e também visando melhorar a interoperabilidade baseada em metadados de aplicações distribuídas (POOLE, 2001).

Java Metadata Interface (JMI) define um mapeamento formal a partir de qualquer metamodelo OMG padrão (como o CWM) para as interfaces Java. E suporta serviços avançados de metadados, como reflexão e programação dinâmica.

Java OLAP Interface (JOLAP) é uma API Java desenvolvida para o acesso a servidores OLAP e a bancos de dados multidimensionais. Utiliza o metamodelo CWM OLAP como base para definição de metadados OLAP, assegurando que recursos compatíveis com o JOLAP sejam capazes de interoperabilidade e intercâmbio de metadados de forma completa através do padrão CWM. JOLAP também define interfaces de consultas que suportam a formação e execução de consultas OLAP, juntamente com o gerenciamento e manipulação de *result sets* multidimensionais . (POOLE, 2003b)

Assim como o JOLAP, *Java Data Mining API* (JDMAPI) define um metamodelo para metadados de *data mining* que é baseado no modelo CWM *Data Mining*. JDMAPI proporciona uma API Java para aplicações de *business intelligence* empregando técnicas de *data mining* para descoberta e análise de conhecimento. (POOLE, 2001)

Na figura 54 segue são mostrados os relacionamentos existentes entre os padrões OMG e os padrões Java.

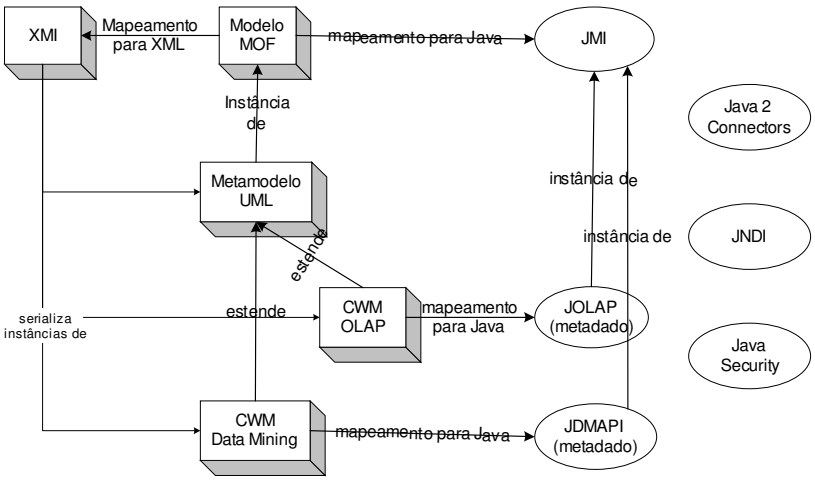


Figura 54 - Relacionamentos entre padrões

Fonte: POOLE, 2001, pág. 13.

3.9 O que CWM não oferece

Em (POOLE, 2002) pode-se ter uma noção clara dos aspectos ficaram de fora da especificação CWM.

CWM não prescreve nenhuma arquitetura de implementação, pois é um modelo de utilização, e não um modelo de implementação. Como um modelo de utilização, são utilizados os vários elementos de modelagem CWM para definir interfaces padrões (tanto em termos de XML como na forma de interfaces programáticas) para o acesso a instâncias de metadados que eles representam. O projeto e implementação dos adaptadores de software, que facilitam a importação e exportação de metadados, estão além do escopo do CWM.

CWM não define nenhuma arquitetura de repositório. A derivação do CWM a partir do MOF não implica que um repositório de metadado CWM deva suportar certos serviços de metadados requeridos por todos os repositórios MOF, em adição às interfaces padrões (XML e APIs programáticas) utilizadas para acessar metadados CWM. Vários serviços importantes, orientados a repositórios, tais como versionamento de modelos, controle de persistência e acesso não são endereçados especificamente na plataforma de tecnologia suportada, e CWM tenta ser estritamente de tecnologia neutra.

CWM não define uma estratégia para gerenciamento de metadados. Uma estratégia coerente de metadados é chave para a aplicação bem sucedida de qualquer solução de integração de metadado. Uma estratégia completa de gerenciamento de metadado, entretanto, não é definida pelo CWM. (O uso de CWM terá certas implicações para a estratégia)

CWM não define uma arquitetura de integração de metadados. A arquitetura de metadado geralmente segue diretamente da estratégia de gerenciamento de metadados. (é uma realização da estratégia completa). CWM simplifica a aplicação das topologias ponto a ponto e *hub-and-spoke*. A seleção de determinada topologia de interconexão é agora mais dirigida

por considerações de gerenciamento de metadados, do que por questões de integração de metadados.

3.10 Considerações

O CWM é o padrão que permite o intercâmbio de metadados entre diversos produtos de diversos fornecedores, e atualmente já se encontra implementado em várias ferramentas de tais fornecedores.

Um ponto importante é que este padrão, além de disponibilizar os metadados através de arquivos XMI, também tem alguns de seus metamodelos implementados em Java na forma de APIs padrões J2EE: JMI, JOLAP e JDMAPI. Em algumas ferramentas, como no MDR[10], a interface JMI passou a ser uma alternativa à interface XMI.

O problema da falta de integração entre os sistemas governamentais e a criação de Câmaras técnicas que visam justamente tal integração são alguns pontos a serem abordados no próximo capítulo. E também serão tratadas as leis que servem de base para a administração orçamentária dos diversos entes governamentais.

---

[10] MDR (*Meta Data Repository*) é um projeto da NetBeans que implementa o padrão baseado em repositório de metadados MOF, da OMG, e que é integrado à plataforma de ferramentas NetBeans (MDR, 2004).

4 O AMBIENTE DA PREFEITURA MUNICIPAL DE MANAUS (PMM)

Os entes da Federação[11] precisam dispor de meios modernos e confiáveis para fornecer à sociedade respostas ágeis e de qualidade a seus questionamentos. E também há a necessidade de se ter formas seguras de mensurar despesas e receitas, obedecendo à Lei Complementar nº 101, de 04/05/2000, também conhecida como Lei de Responsabilidade Fiscal (LRF).

> "A responsabilidade na gestão fiscal pressupõe a ação planejada e transparente, em que se previnem riscos e corrigem desvios capazes de afetar o equilíbrio das contas públicas, mediante o cumprimento de metas de resultados entre receitas e despesas e a obediência a limites e condições no que tange a renúncia de receita, geração de despesas com pessoal, da seguridade social e outras, dívidas consolidada e mobiliária, operações de crédito, inclusive por antecipação de receita, concessão de garantia e inscrição em Restos a Pagar." (FIGUEIREDO, 2001)[12]

A sanções pelo descumprimento da LRF estão estabelecidas na Lei nº 10.028, de 19/10/2000, denominada Lei de Crimes Fiscais. Qualquer crime sendo considerado contra as finanças públicas pode vir a gerar a reclusão ou detenção do ordenador.

Pelo exposto acima, observa-se o quão primordial é o papel da informática como ferramenta de auxílio no controle das finanças públicas e, conseqüentemente, nas tomadas de decisões das entidades da Federação.

Hoje em dia as organizações governamentais podem lançar mão de importantes inovações tecnológicas tais como a Internet, redes de computadores, ferramentas como *data warehouse* e OLAP, e ferramentas de geoprocessamento.

---

[11] Entendam-se como *Entes da Federação* a União, o Distrito Federal, cada Estado e cada Município.

[12] Lei Complementar 101, de 4 de maio de 2000. Capítulo 1, Art. 1º, §. 1º.

Porém promover grandes mudanças tecnológicas nem sempre é fácil, mesmo tendo-se noção de que tais mudanças podem levar a um aumento da qualidade das informações que serão disponibilizadas. Para que as mudanças possam acontecer, geralmente há a necessidade de uma profunda transformação a nível cultural na execução de determinados processos de trabalho, sendo necessária a incorporação de novas práticas para se adequar às novas ferramentas.

Uma das opções de gerenciamento da máquina pública é a utilização dos sistemas de informação. As prefeituras podem ser dividas entre aquelas que utilizam sistemas de informação e aquelas que ainda estão exercendo o controle somente de forma manual. Todavia dentre essas possuidoras de ferramentas que as auxiliem nas tomadas de decisões, poucas vezes tais sistemas encontram-se integrados, levando à duplicidade de dados nos diversos sistemas, não assegurando a veracidade e segurança nos dados demonstrados. Geralmente o motivo da falta de integração é o surgimento dos grandes sistemas de gestão em diferentes períodos, utilizando tecnologias diversas, e tendo sido construídos com foco no cumprimento da função específica do órgão que o gerencia.

O Comitê Executivo do Governo Eletrônico, na sua primeira reunião do Governo Lula, decidiu criar oito Câmaras técnicas para a coordenação das iniciativas de governo eletrônico. São elas: Implementação de Software Livre, Inclusão Digital, Integração de Sistemas, Sistemas Legados e Licenças, Gestão de *Sites* e Serviços *On-line*, Infra-Estrutura de Rede, Governo para Governo e Gestão de Conhecimento e Informação Estratégica (EGOV, 2003). A seguir uma breve descrição da atividade-fim de cada uma delas:

- **Câmara de Implementação do Software Livre –** Estabelece a estratégia de implementação de migração e adoção de soluções baseadas em software livre.
- **Câmara de Inclusão Digital –** Estabelece diretrizes e coordena a estratégia das ações públicas de inclusão digital.

- **Câmara de Integração de Sistemas –** Estabelece normas e critérios para integração de sistemas estruturadores do governo.
- **Câmara de Sistemas Legados e Licenças** – Estabelece critérios para a evolução dos sistemas legados e cria normas para a renegociação de contratos com grandes fornecedores, visando a redução de custos e diminuição da dependência.
- **Câmara de Gestão de *Sites* e Serviços *On-line*** – Estabelece normas e políticas para integração e otimização dos serviços e informações on-line prestados pelo governo.
- **Câmara de Infra-Estrutura de Redes –** Estabelece normas e políticas visando a integração das diversas redes (voz, dados e imagem) do governo.
- **Câmara Governo para Governo –** Cria políticas e normas para a integração de aplicações com estados, municípios e demais poderes.
- **Câmara de Gestão do Conhecimento e Informação Estratégica -** Estabelece normas e políticas para a geração e gestão de base de conhecimento estratégico.

Com isso pode-se notar que o problema de integração e compartilhamento de dados públicos é de tamanha importância no contexto nacional que, dentre as Câmaras citadas acima, duas terão como foco principal a integração de sistemas, sendo a Câmara de Integração de Sistemas e a Câmara Governo para Governo.

E um dos pontos importantes e que deve ser levado em consideração para que possa haver essa tão sonhada integração é justamente o da integração de metadados dos diversificados sistemas, pois, como em tantas outras áreas, há a carência na padronização de metadados.

4.1 A Prefeitura Municipal de Manaus

A organização em questão para aplicação do objeto de estudo desta dissertação é a Prefeitura Municipal de Manaus (PMM), a qual é composta por diversos *órgãos*[13] tanto da administração direta quanto da administração indireta[14]. Cada órgão pode ser classificado como sendo do Poder Legislativo ou Executivo.

Tais órgãos atuam em áreas como educação, transporte, finanças, obras públicas, saúde, entre outras. Porém cada uma na atividade principal que lhe compete. Cada órgão é responsável pelo controle dos gastos de suas atividades principais, além de gastos mais gerais como telefone, combustível, energia, água, etc.

A Constituição de 1988 (SENADO, 2003), institui os planos plurianuais (PPA), os quais servem de instrumento para orientar a elaboração da lei de diretrizes orçamentárias (LDO) e da lei orçamentária anual (LOA), e também a apresentação de emendas por parte dos legisladores.

O PPA tem a duração de 4 anos, a mesma duração do mandato do Chefe do Poder Executivo, porém cobrindo o período compreendido entre o início do segundo ano do mandato presidencial e o final do primeiro exercício do mandato subseqüente.

A LDO é feita anualmente, e tem os propósitos de orientar a estruturação da proposta orçamentária anual, incluindo as despesas para o exercício financeiro subseqüente. Ela contém as instruções e regras a serem cumpridas na execução do orçamento. A LDO para o

---

[13] O órgão tem o sentido de órgão de Governo ou unidade administrativa, e também compreende uma repartição do órgão ou um agrupamento de serviços subordinados a determinado órgão, sendo a unidade executora do projeto ou atividade. Os recursos orçamentários (dotações) são consignados a ele.

[14] Um órgão da *Administração Indireta* é aquele tipo de entidade que possui personalidade jurídica própria, abrangendo autarquias, empresas públicas, sociedades de economia mista e fundações públicas.

próximo exercício deve ser aprovada pelo Legislativo até o dia 30 de junho do exercício corrente.

A LOA é feita anualmente, com base no disposto na LDO, e tem a finalidade de estimar a receita e fixar a despesa. Contém, além do texto regulamentar, uma variedade de quadros demonstrativos da receita e da despesa, os quais são detalhados de acordo com critérios como por função, por fonte, entre outros. A LOA para o próximo exercício deve ser aprovada pelo Legislativo até o dia 15 de dezembro do exercício corrente.

O orçamento é regido pelo disposto na Lei nº 4.320/64 e suas portarias. Em linhas gerais o seu controle ocorre da seguinte forma: durante o exercício fiscal corrente, o Legislativo, na figura da Câmara Municipal, aprova em um primeiro momento a LDO, e mais tarde a LOA do próximo exercício. E são as despesas fixadas e as receitas estimadas na LOA que devem ser controladas por cada órgão no exercício vindouro.

Durante a vigência do orçamento todas as despesas que precisam ser realizadas precisam ser previamente autorizadas e empenhadas. Para isso reserva-se o valor desejado indicando na célula orçamentária onde haverá a movimentação. Após o empenho e após a entrega do serviço ou início do serviço é que será feita uma análise deste serviço e autorizado o pagamento, sendo gerada primeiramente a Liquidação de Despesa e logo em seguida o pagamento apropriado. E essas informações sobre valor liquidado e valor pago são atualizadas na célula orçamentária com o objetivo de se permitir o acompanhamento detalhado da execução orçamentária.

A Secretaria Municipal de Economia e Finanças (SEMEF) é responsável pela coordenação da elaboração orçamentária e agregação das propostas necessárias à unificação

da Lei Orçamentária[15]. Sendo responsável ainda pela apresentação de balancetes e outros relatórios contábeis e orçamentários ao Tribunal de Contas para o devido acompanhamento.

A Informática exerce um papel fundamental na área fazendária, participando ativamente de todo o ciclo do processo. Abrange a elaboração e execução do orçamento com todo o controle da movimentação financeira e contábil, e também engloba a inscrição no cadastro de contribuintes, a arrecadação dos tributos (contemplando também a cobrança, dívida ativa e execuções judiciais), a fiscalização e autuação.

Atualmente o Centro de Tecnologia de Informação (CTI) da PMM tem utilizado o SGBD Oracle para armazenar os dados operacionais de seus sistemas, e ainda não possui nenhuma ferramenta de *data warehouse* ou OLAP. Anteriormente ao surgimento do CWM a tendência seria adquirir tais ferramentas do mesmo fabricante, pois geralmente cada uma tinha seu próprio sistema de gerenciamento de metadados, vindo a dificultar e muito o intercâmbio de seus metadados para produtos de outros fabricantes.

A idéia principal é a de se integrar os dados desses vários ambientes diferenciados sem necessariamente ter que se migrar dados de um sistema a outro, fato que geralmente leva à duplicidade de informações, dificultando um maior controle gerencial.

Quando não havia essa preocupação de se integrar os metadados das diversas ferramentas, a empresa ao adquirir uma ferramenta de determinado fabricante, ficava presa às demais ferramentas deste, pois o padrão de metadados implementado por ele geralmente não era disponibilizado para que outros fabricantes pudessem ter acesso. Exemplificando, caso uma determinada empresa adquirisse o SGBD Oracle, e se resolvesse adquirir alguma ferramenta de *data warehouse*, tenderia a adquiri-la da própria Oracle.

Em caso de órgãos públicos tal situação é um pouco complicada, pois nesse caso durante o processo de Licitação haveria a necessidade de se explicitar o fabricante do produto.

---

[15] Lei nº 704, de 03/07/2004, Capítulo VII, Art. 35. LDO para o exercício de 2004.

E, atualmente, com a existência do CWM, bastaria solicitar a Licitação de uma ferramenta de *Data Warehouse* que utilize a abordagem CWM para integração de metadados. Todos os fabricantes implementando a tecnologia CWM poderiam participar do processo licitatório, já que suas ferramentas poderiam interagir tranqüilamente com ferramentas compatíveis com o CWM já anteriormente adquiridas, ainda que de outro fabricante.

4.2 Considerações

Este capítulo procurou dar uma visão geral da problemática da falta de integração entre os sistemas dos diversos âmbitos governamentais. Verificou-se que a fim de minimizar tais problemas dentre as oito câmaras criadas pelo pelo Comitê Executivo do Governo Eletrônico, duas visam tratar exclusivamente do assunto integração.

Um outro tópico abordado foi o das leis que regem o funcionamento da PMM, enfocando principalmente o aspecto orçamentário. Para isso procurou-se enfatizar a aplicação do PPA, LDO e LOA, os quais foram instituídos pelo Senado Federal na Constituição de 1988. Os aspectos orçamentários regidos pela Lei nº 4.320/64 foram destacados com a finalidade de se proporcionar o embasamento para o modelo de integração das informações orçamentárias a ser proposto no próximo capítulo.

5 DESENVOLVIMENTO DE UMA ARQUITETURA DE INTEGRAÇÃO DE METADADOS GOVERNAMENTAIS

Como será visto a seguir, para que se possa acompanhar a execução orçamentária ao longo dos anos, há a necessidade de uma padronização na classificação dos elementos a sofrerem a análise. Porém, devido às variações geradas pelas atualizações na lei que rege o orçamento, esta padronização de tempos em tempos sofre alguma alteração, dificultando a comparação entre os exercícios.

A seguir será mostrada uma proposta de arquitetura de integração de metadados orçamentários através da disponibilização de um esquema em estrela dos dados orçamentários para um ambiente de *data warehouse*. A modelagem deste esquema foi baseada nas partes orçamentárias do modelo ER do Sistema Administrativo Integrado (SAI[16]), o qual engloba os subsistemas responsáveis pela Elaboração do Orçamento, Execução do Orçamento, Execução Financeira e Contabilidade de todos os órgãos que compõem a PMM.

5.1 O Orçamento

A necessidade da padronização de orçamentos públicos para os diversos níveis de governo ocorreu em 1932, quando ao tentar consolidar a dívida externa brasileira o governo federal encontrou grandes dificuldades devido às diferenças de nomenclaturas e títulos.

Então em 1938 foi aprovada resolução que atribuía ao então Conselho Técnico de Economia e Finanças do Ministério da Fazenda a elaboração de estudo que conduzisse à

[16] O SAI foi desenvolvido para o ambiente Cliente-Servidor, com base de dados Relacional Oracle e linguagem de programação Delphi.

padronização das normas e à classificação dos orçamentos dos três níveis, a entrar em funcionamento em 1939.

Em 24/11/1939 o Governo Federal baixou o Decreto-lei nº 1.804, o qual padronizava os orçamentos dos Estados e Municípios, ainda não enquadrando o orçamento da União. E finalmente em 17/03/1964 o Congresso Nacional aprovou a Lei nº 4.320, disponibilizando o modelo orçamentário-padrão para os três níveis de governo, e instituindo a adoção de plano de contas único para as três esferas.

Esta técnica padronizadora também permitiu a atualização dos anexos da lei mediante atos administrativos, como a Portaria nº 9, de 28/01/1974, a qual introduziu a classificação funcional-programática da despesa orçamentária.

Com a constante evolução através da atualização dos anexos da lei, a modernização orçamentária levou à substituição da classificação funcional-programática pelas classificações funcional e por programas, através do Decreto nº 2, de 29/10/1998, e da Portaria nº 42, de 14/04/1999, sancionados pelo Ministério do Orçamento e Gestão.

O elemento básico de expressão do orçamento é a conta, ou célula orçamentária. Esta célula pode ser utilizada tanto para análise quanto para síntese. Para que se possa alcançar determinado objetivo deve-se estabelecer algum critério na classificação da célula.

Exemplificando, a compra de luvas cirúrgicas pode ser classificada:

- segundo a data de aquisição;
- segundo o item da despesa (ex.: material hospitalar);
- no programa que utilizará as luvas (ex.: Médico ou Odontologia); ou
- no programa, segundo o tipo de realização (ex.: Atendimento no Programa Médico da Família ou Pronto Atendimento Odontológico).

A Lei nº 4.320/64 instituía a classificação funcional, onde a célula orçamentária era identificada principalmente através das *funções* e *subfunções*. Esta classificação compreendia 10 *funções*, com cada uma subdividida em 10 *subfunções.*

Ex.: Para a *função* 6 - Educação e Cultura existiam as seguintes *subfunções:*

6.0 – Administração

6.1 – Ensino Primário

6.2 – Ensino Secundário e Normal

6.3 – Ensino Técnico-Profissional

6.4 – Ensino Superior

... (até o 6.9 - Diversos)

Porém em 1974, através da Portaria nº 9, foi introduzida a classificação funcional-programática. A categoria *função* teve seu número ampliado de 10 itens para 16, e a categoria *subfunção* desapareceu, sendo criados em seu lugar os *programas*, os quais se subdividem em *sub-programas* e estes em *projetos* e *atividades*.

Ex.: Função: 08 – Educação e Cultura

Programa: 44 – Ensino Superior

Subprograma: 205 – Ensino de Graduação

Subprograma: 206 – Ensino de Pós-graduação

Subprograma: 207 – Extensão Universitária

Esta classificação por programas vigorou, para os orçamentos municipais, até o exercício de 2001, e foi quando se voltou a adotar a classificação por *funções* e *subfunções*,

separada da classificação por *programas*. Esta nova classificação compreende 28 *funções* e 109 *subfunções*.

Este fato de a célula orçamentária poder sofrer alterações conforme ocorram modificações nas leis que regem o Orçamento público gera alguns problemas para os órgãos executores do Orçamento. Exemplificando, um PPA compreende 4 exercícios, como acompanhar a execução de determinada célula se houver mudanças entre os exercícios?

5.2 O Modelo Orçamentário Proposto

Até o término do exercício de 2001 a PMM seguia a classificação funcional-programática, e a célula orçamentária era expressa da seguinte forma (Tabela 4):

Tabela 4 - Exemplo de célula da classificação funcional-programática

CAMPO	CÓD	DESCRIÇÃO
ÓRGÃO	18100	SECRETARIA MUNICIPAL DE EDUCAÇÃO
FUNÇÃO	08	EDUCAÇÃO E CULTURA
PROGRAMA	42	ENSINO FUNDAMENTAL
SUBPROGRAMA	217	TREINAMENTO DE RECURSOS HUMANOS
PROJETO ATIVIDADE	2181.110	PROGRAMA DE FORMAÇÃO E VALORIZAÇÃO DOS PROFISSIONAIS DA EDUCAÇÃO
NATUREZA DA DESPESA	312000	MATERIAL DE CONSUMO
FONTE DE RECURSO	07	TRANSFERENCIAS DO ESTADO

A partir do exercício de 2002, após a instituição da classificação por *função* e *subfunção*, a célula orçamentária passou a ser representada da seguinte forma (Tabela 5):

Tabela 5 - Exemplo de célula da classificação por *função* e *subfunção*

CAMPO	CÓD	DESCRIÇÃO
ÓRGÃO	18100	SECRETARIA MUNICIPAL DE EDUCAÇÃO
FUNÇÃO	12	EDUCAÇÃO
SUBFUNÇÃO	361	ENSINO FUNDAMENTAL
PROGRAMA	1002	FORMAÇÃO E VALORIZAÇÃO DOS PROFISSIONAIS DA EDUCAÇÃO
AÇÃO	01035	FORMAÇÃO DE DOCENTES EM NÍVEL SUPERIOR
NATUREZA DA DESPESA	33903900	OUTROS SERVIÇOS DE TERCEIROS – PESSOA JURÍDICA
FONTE DE RECURSO	75	COTA-PARTE DO FUNDO DE EDUCAÇÃO – FUNDEF

Ao associar cada item da classificação funcional-programática (anterior) com cada item da classificação por *função* e *subfunção* (corrente) deve-se prestar atenção a detalhes como: aquela que na classificação funcional-programática era uma *função* agrupada na classificação por *função* e *subfunção* foi transformada em duas *funções*. A comparação por *função* ocorrerá somente ao reagrupar as *funções* da classificação atual, de acordo com o que estava estabelecido na classificação anterior.

Além disso, observando-se os dois exemplos de células orçamentárias listados acima tem-se PROGRAMA 42 – ENSINO FUNDAMENTAL e na célula da classificação atual SUBFUNÇÃO 361 – ENSINO FUNDAMENTAL. A tendência é a de se associar o *Programa* da classificação anterior com a *Subfunção* da classificação atual. Porém, de acordo com o informado pelos colaboradores da área orçamentária, a comparação deve ser feita entre o item da classificação anterior com o item da classificação atual. As associações passam a ficar da seguinte forma (Tabela 6):

Tabela 6 - Associação proposta

CLASSIFICAÇÃO FUNCIONAL-PROGRAMÁTICA	CLASSIFICAÇÃO POR FUNÇÃO E SUBFUNÇÃO
FUNÇÃO	FUNÇAO
PROGRAMA	PROGRAMA
PROJETO/ATIVIDADE	AÇÃO
NATUREZA DA DESPESA	NATUREZA DA DESPESA
FONTE DE RECURSO	FONTE DE RECURSO

Os itens *Subprograma*, da classificação funcional-programática, e *Subfunção*, da classificação por *função* e *subfunção*, não fazem parte da solução, pois semanticamente não têm nenhuma associação entre si.

Para que o modelo proposto consiga alcançar o objetivo de permitir as comparações dos dados orçamentários da classificação anterior com a nova classificação o formato proposto é o que está definido na Figura 55.

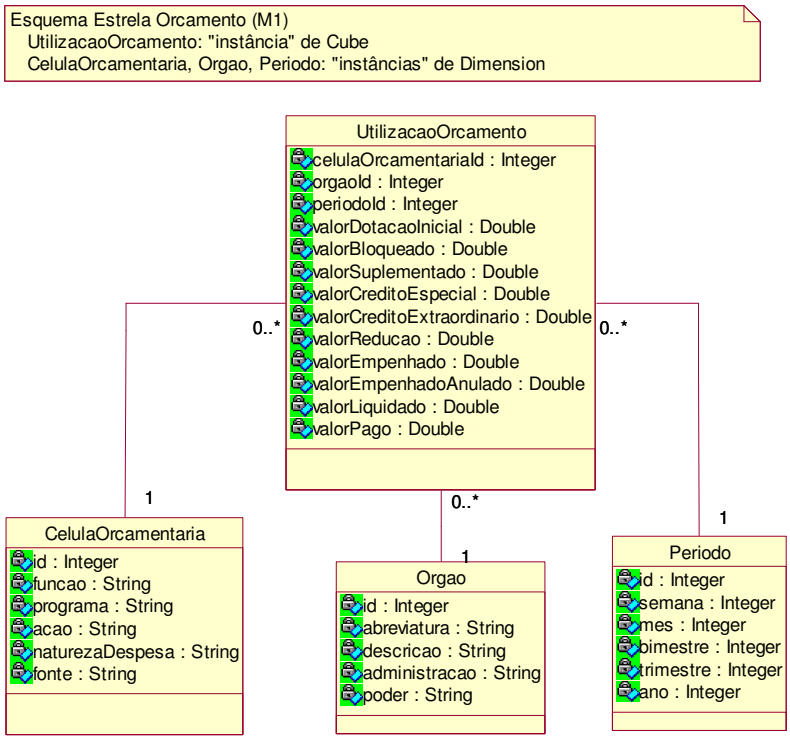


Figura 55 - Esquema Estrela - Orçamento

É proposto um esquema estrela pelo fato de a modelagem dimensional possibilitar a otimização de consultas, e também por este tipo de modelo ser menos complexo ao se apresentar ao usuário, e além disso, no modelo em floco de neve a união entre as diversas partes do floco de neve irá comprometer o desempenho do sistema como um todo.

A dimensão *CelulaOrcamentaria* contém os itens das células orçamentárias que podem ser utilizados para a comparação, e que são aqueles relacionados na Tabela 6. Como para a classificação atual a função foi desdobrada, o processo de ETL deve unir novamente as duas funções que na classificação anterior eram relativas a somente uma função, e esta informação deverá ser atribuída ao campo *funcao*. Como os itens ação, da classificação atual, e projeto/atividade, da classificação anterior, podem ser comparados, apesar das descrições serem diferentes, os mesmos devem ser atribuídos ao campo *acao*.

É através das soluções para esses dois campos citados acima que o modelo proposto pretende eliminar o problema de se realizar associações e comparações entre células orçamentárias da classificação funcional-programática com as da classificação por *função* e *subfunção.*

Os valores definidos em *UtilizacaoOrçamento* aqueles constantes nos sistemas transacionais existentes (SIADAM e SAI), e são os que podem despertar o interesse da análise. As informações disponibilizadas por semana, mês, bimestre, trimestre e ano podem permitir a análise comparativa dependendo do período escolhido.

5.3 Vantagens do CWM para a PMM

Os dados orçamentários dos órgãos que compõem a PMM encontram-se atualmente em dois ambientes computacionais diferentes: grande parte dos dados (compreendendo dados do exercício de 1995 até o exercício atual) encontra-se em mainframe, no sistema SIADAM (desenvolvido pela PRODAM - Centro de Processamento de Dados do Amazonas). E dados (desde o exercício de 2002) de alguns órgãos da administração indireta encontram-se em ambiente cliente-servidor, no SAI.

Pretende-se com o CWM minimizar o impacto da diferença entre os ambientes computacionais, bancos de dados e modos de acesso, já que um dos objetivos do CWM é justamente o de padronizar os formatos de intercâmbio de metadados para ambientes de *business intelligence* e *Data Warehouse*, além de padronizar a API da linguagem de programação para o acesso desses metadados.

As vantagens para o ambiente governamental da utilização de ferramentas que sejam implementadas de acordo com o padrão CWM são inúmeras. Na aquisição de novas ferramentas não haverá a necessidade de se fixar em um determinado fornecedor. Devido à

padronização estabelecida, os metadados são reconhecidos em cada ferramenta que esteja em conformidade com o CWM. Exemplificando, no caso da necessidade de aquisição de uma ferramenta OLAP, bastaria indicar na Licitação a aquisição de ferramenta OLAP que esteja em conformidade com o CWM.

A PMM possui a base de dados Oracle 9i, a qual já está em conformidade com o CWM. Caso pretenda adquirir uma ferramenta para *Data Warehouse* e para OLAP, poderá adquirir qualquer uma que também esteja em conformidade com o CWM. O intercâmbio dos metadados que já se encontram no banco de dados relacional para a ferramenta de *data warehouse* poderá ser feito através da geração e posterior leitura em XMI.

E, além disso, como foi visto anteriormente, com a criação da Câmara de Integração de Sistemas, parece existir no governo atual um forte interesse para que haja um esforço para que se alcance a verdadeira integração dos sistemas de informação existentes. Isto porque no momento cada ente da Federação tem o seu próprio mecanismo de controle, sendo uma ilha de informações, do ponto de vista gerencial total. Porém, com a integração haverá um controle mais apurado sobre os gastos e a receita pública, levando a um melhor gerenciamento da vida do País.

5.4 Considerações

Como pôde ser visto no início deste capítulo, a célula orçamentária é o elemento básico para expressão do orçamento. E o orçamento, por sua vez, é a ferramenta de gestão utilizada para a administração do erário público. O orçamento não é estático no tempo, e sim pode vir a sofrer determinadas alterações naturais com o passar do tempo. Alterações estas que sempre visam à melhoria do controle orçamentário.

Relatada como não existente pelo diretor do orçamento e sua equipe, a abordagem proposta, que permite o acompanhamento das ações orçamentárias através da comparação de itens da célula orçamentária de classificações diferentes, não foi encontrada semelhante na literatura.

O modelo proposto surgiu da necessidade de se permitir à equipe orçamentária efetuar análises comparativas entre dados orçamentários da classificação funcional-programática e os dados da classificação por *função* e *subfunção*.A solução encontrada foi a modelagem em um esquema estrela, porém não pôde ser implementada devido a entidade em questão não possuir as ferramentas de *data warehouse* e OLAP necessárias para isto, e também por seu alto custo de aquisição.

Porém já praticamente próximo à conclusão deste trabalho, foi encontrada uma solução em software livre, disponibilizada pela Sun Microsystems. Tal solução faz parte do projeto de software livre NetBeans, através do módulo denominado MDR (*Meta Data Repository*). Este módulo implementa o CWM, permite a importação do XMI gerado, e também permite a exportação de metadados XMI ou interfaces JMI.

No capítulo a seguir serão disponibilizadas as considerações finais e as sugestões de trabalhos futuros.

6 CONCLUSÕES FINAIS E TRABALHOS FUTUROS

Nesta dissertação inicialmente foram abordadas as definições de metadados, destacando-se a importância dos metadados de *data warehouses*, e também das arquiteturas provedoras da integração de metadados (ponto-a-ponto, *hub-and-spoke* e abordagem baseada em modelo).

E foi através da pesquisa da abordagem baseada em modelo que se chegou ao CWM, o qual permite a integração de metados a partir da utilização de um modelo comum de integração. Foi dada uma breve explanação sobre cada tecnologia que serve de base para o CWM (UML, MOF, XML e XMI). Chegou-se à nova especificação da OMG, o MDA, e também às especificações Java que implementam o CWM: JMI, JOLAP e JDMAPI.

Ainda em relação ao CWM foram abordadas todas as camadas que o compõem (camadas *Object Model, Foundation, Resource, Analysis* e *Management*), juntamente com seus respectivos metamodelos.

Um aspecto interessante que se verificou foi a capacidade de se estender o metamodelo CWM. Através desse mecanismo, podem-se desenvolver outros aspectos faltantes em determinados metamodelos, justamente por serem específicos de determinadas ferramentas.

Após a pesquisa da fundamentação foi feita então a aplicação prática do estudo através da modelagem de um esquema em estrela para o Orçamento Público. Esse modelo contém as informações chaves para se realizar o acompanhamento e controle das informações orçamentárias de uma entidade pública, podendo-se também confrontar tais dados com as informações realizadas, isto é, as informações financeiras.

Para a área governamental a grande vantagem na aquisição de ferramentas que implementem o CWM será justamente a economia do erário público, já que a utilização do CWM leva à democratização no aspecto licitatório do processo de aquisição de ferramentas.

Já não há a necessidade de se fixar em um determinado fabricante só porque já se adquiriu uma ferramenta anterior deste mesmo fabricante. Os outros fabricantes implementando o CWM levam à interoperabilidade entre as diversas ferramentas de diversos fabricantes.

E para a área tecnológica a grande vantagem do CWM é que não há mais a necessidade de se criar diferentes pontes de integração para cada ferramenta com a qual se quiser integrar. O fabricante já não precisará se preocupar a cada vez que uma nova ferramenta de outro fabricante seja lançada. Se todos estiverem implementando o CWM, a troca de metadados será feita de acordo o padrão estabelecido. Então a partir de agora cada ferramenta deverá se preocupar em desenvolver e melhorar somente os seus aspectos que a tornem o diferencial dentre outras ferramentas similares.

A partir do que foi alcançado nesta dissertação, sugerem-se as seguintes propostas de trabalhos futuros:

- implementar o modelo proposto em uma ferramenta relacional compatível com o CWM. Exportar seus metadados através da XMI ou IDL para alguma outra ferramenta (de preferência de um fabricante diferente) que implemente o CWM, podendo ser uma ferramenta de *data warehouse* ou OLAP, por exemplo.
- implementar um modelo em ambiente de Registros e fazer a troca de metadados com um ambiente Relacional. Na área governamental existem muitos sistemas robustos cuja plataforma é o mainframe, e os novos sistemas geralmente vêm sendo desenvolvidos para as novas tecnologias de micro, e às vezes surge a necessidade de se comparar dados existentes nos dois ambientes.
- estudar como o Java está implementando o CWM, verificando-se se estão sendo abordados todos os aspectos do metamodelo escolhido, já que Java é

uma linguagem que vem sendo bastante difundida e utilizada em vários ambientes.

- Utilização do MDR como ferramenta de repositório para a importação e exportação de arquivos XMI, e também para geração de DTDs e interfaces JMI.

REFERÊNCIAS BIBLIOGRÁFICAS

AUTH, Gunnar, MAUR, Eitel von, HELFERT, Markus. **A Model-based Software Architecture for Metadata Management in Data Warehouse Systems.** BIS '02, 5th International Conference on Business Information Systems, Poznan, Poland, 2002.

BISKUP, Joachim, CONVENT, Bernhard. **A Formal View Integration Method.** ACM SIGMOD Record, 15(2), 398-407. Junho, 1986.

BNDES – Banco Nacional de Desenvolvimento Econômico e Social. Disponível em: http://www.bndes.gov.br. Acessado em: dezembro, 2002.

CA – Computer Associates. **Putting Metadata to Work in the Warehouse:** A White Paper. 2000.

COME, Gilberto de. **Os Metadados no Ambiente de Data Warehouse.** IV SEMEAD, Universidade de São Paulo. Outubro, 1999.

COREY, Michael, ABBEY, Michael, ABRAMSON, Ian, TAUB, Ben. **Oracle 8i Data Warehouse**. Ed. Campus, Rio de Janeiro. 2001.

CWM – Common Warehouse Metamodel. Disponível em: http://www.omg.org/cwm/. Acessado em: 2002.

DO, Hong Hai, RAHM, Erhard. **On Metadata Interoperability in Data Warehouses.** ISSN 1430-3701. Março, 2000.

EGOV – Governo Eletrônico. **Novas Câmaras do Governo Eletrônico.** Disponível em: http://www.governoeletronico.e.gov.br/. Acessado em agosto de 2003.

FIGUEIREDO, Maurício Carlos, FERREIRA, Cláudio, RAPOSO, Fernando, BRAGA, Henrique, NÓBREGA, Marcos. **Comentários à Lei de Responsabilidade Fiscal.** Ed. Revista dos Tribunais, 2ª ed. rev., atual. e ampl., São Paulo, 2001.

GIACOMONI, James. **Orçamento Público.** Ed. Atlas, 9ª edição revista e atualizada, São Paulo, 2000.

HOLZNER, Steven. **Desvendando o XML**. Ed. Campus, Rio de Janeiro, 2001.

HULL, Richard. **Managing Semantic Heterogeneity in Databases: A Theoretical Perspective.** Proceedings of the ACM SIGA CT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), pages 51-61. ACM Press, 1997.

IKEMATU, Ricardo Shoiti. **Gestão de Metadados: Sua Evolução na Tecnologia da Informação.** DataGramaZero - Revista de Ciência da Informação, v.2, n.6. Dezembro, 2001.

INMON, W.H., WELCH, J.D., GLASSEY, Katherine L. **Gerenciando Data Warehouse**. Makron Books, São Paulo, 1999.

IYENGAR, Sridhar. **CWM Audio Briefing:** The Key to Integrating Business Intelligence. OMG. http://www.omg.org/news/releases/pr2000/cwm/whitepaper.htm. Acessado em: maio, 2002.

JARKE, Matthias, LENZERINI, Maurizio, VASSILIOU, Yannis, VASSILIADIS, Ranos. **Fundamentals of Data Warehouses**. Ed. Springer, Germany. Págs. 123 – 158. 2000.

JEFFERY, Keith G. **Metadata: An Overview and Some Issues.** ERCIM News, ed. 35. Outubro, 1998.

KIMBALL, Ralph. **Data Warehouse Toolkit –** Técnicas para Construção de Data Warehouses Dimensionais. São Paulo: Makron Books, 1998.

KIMBALL, Ralph, ROSS, Margy. **The Data Warehouse Toolkit – Guia Completo para Modelagem Dimensional**. Ed. Campus, Rio de Janeiro, 2002.

MACHADO JR., J. Teixeira, REIS, Heraldo da Costa. **A Lei 4.320 Comentada –** Com a Introdução de Comentários à Lei de Responsabilidade Fiscal. Ed. IBAM, 30ª ed. rev. e atual., Rio de Janeiro, 2001.

MARCO, David. **Top 10 Questions do Ask/Mistakes to Avoid When Building a Data Warehouse or a Meta Data Repository.** White Paper. Disponível em: http://www.dmreview.com. Acessado em: 2001.

MARCO, David. **Bulding and Managing the Meta Data Repository** – A Full Lyfecycle Guide. Ed. John Wiley & Sons, Inc., 2002.

MCLAUGHLIN, Brett. **Integrating Java Objects and XML Data.** Disponível em: http://www-106.ibm.com/developerworks/library/x-quick/. Acessado em: agosto, 2002.

MDA – Model Driven Architecture. Disponível em: http://www.omg.org/mda/. Acessado em: 2003.

MDR – Meta Data Repository. Disponível em: http://mdr.netbeans.org/index.html. Acessado em: fevereiro, 2004.

MOF – Meta Object Facility. Disponível em: http://www.omg.org/technology/documents/formal/mof.htm. Acessado em: 2003.

MOSHER, Chuck. **A New Specification for Managing Metadata.** Disponível em: http://developer.java.sun.com/developer/technicalArticles/J2EE/JMI/. Acessado em: abril, 2002.

NETO, Jorge Abílio A. **Aplicações de Data Warehouse e Operational Data Storage na Administração Pública**. Monografia da disciplina de Banco de Dados. Manaus, abril, 2001.

OMG – Object Management Group. **Common Warehouse Metamodel (CWM) Specification.** Versão 1.1. Disponível em:

http://www.omg.org/technology/documents/modeling_spec_catalog.htm. Fevereiro, 2002a.

OMG – Object Management Group. **Common Warehouse Metamodel (CWM) Specification, Volumes 2 - Extensions.** Versão 1.1. Disponível em: http://www.omg.org/technology/documents/modeling_spec_catalog.htm. Fevereiro, 2002b.

PAN, Jeff Z., HORROCKS, Ian. **Metamodeling Architecture of Web Ontology Languages.** Proceedings of the Semantic Web Working Symposium. 2001.

PMM – PREFEITURA MUNICIPAL DE MANAUS. Disponível em: http://www.pmm.am.gov.br. Acessado em: julho, 2002.

POOLE, John. **The Common Warehouse Metamodel as a Foundation for Active Object Models in the Data Warehouse Environment.** Disponível em: http://www.cwmforum.org. 2000.

POOLE, John D. **Model-Driven Architecture: Vision, Standards and Emerging Technologies.** Position Paper. Workshop on Metamodeling and Adaptative Object Models, ECOOP 2001. Abril, 2001.

POOLE, John, CHANG, Dan, TOLBERT, Douglas, MELLOR, David. **Common Warehouse Metamodel –** An Introduction to the Standard for Data Warehouse Integration. Ed. John Wiley & Sons, Inc., New York, 2002.

POOLE, John, CHANG, Dan, TOLBERT, Douglas, MELLOR, David. **Common Warehouse Metamodel –** Developer's Guide. Ed. John Wiley & Sons, Inc., Indianapolis, Indiana, 2003a.

POOLE, John. **Model-Driven Data Warehousing.** Integrate.2003, Burlingame, CA. 2003b.

SANTOS, Hélio Lopes dos. **Uma Solução de Metadados Baseada nos Padrões MOF e XML.** Dissertação de Mestrado, UFPE. Recife. Março, 2003.

SANTOS, Hélio Lopes dos, BARROS, Roberto Souto Maior de, FONSECA, Décio. **Uma Proposta para Gerenciamento de Metadados nos Padrões XML e DTD em Repositórios MOF.** 18º Simpósio Brasileiro de Banco de Dados. Págs. 41 – 55. Manaus. Outubro, 2003.

SENADO – Senado Federal. **Constituição da República Federativa do Brasil**. Disponível em: http://legis.senado.gov.br/con1988/index.htm. Acessado em: 2003.

STAUDT, Martin, VADUVA, Anca, VETTERLI, Thomas. **Metadata Management and Data Warehousing.** Technical Report, Dept. of Information Technology (University of Zurich), 1999.

TANNENBAUM, Adrienne. **Metadata Solutions:** Using Metamodels, Repositories, XML, and Enterprise Portals to Generate Information on Demand. Ed. Addison Wesley, 2002a.

TANNENBAUM, Adrienne. **Meta Data Solutions and Their Return on Investment.** Disponível em: http://www.datawarehouse.com. 2002b.

THOMSEN, Erik. **OLAP:** Construindo sistemas de informações multidimensionais. Tradução da 2ª edição original. Ed. Campus, Rio de Janeiro, 2002.

UML – Unified Modeling Language. Disponível em: http://www.omg.org/uml/. Acessado em: 2003.

W3C - World Wide Web Consortium. Disponível em: http://www.w3c.org/. Acessado em 2003.

XMI – XML Metadata Interchange. Disponível em: http://www.omg.org/technology/documents/formal/xmi.htm. Acessado em: 2003.

XML – Extensible Markup Language. Disponível em: http://www.w3.org/XML/. Acessado em: 2003.

YODER, Joseph W., RAZAVI, Reza. **Adaptive Object-Models.** Technical Report #WUCS-98-25 (PLoP '98/EuroPLoP '98), *Department of Computer Science*, Washington University, Setembro, 1998.

YODER, Joseph W., BALAGUER, Federico, JOHNSON, Ralph. **Architecture and Design of Adaptive Object-Models.** ACM SIG PLAN Notices, vol. 36, nº 12, 50-60, 2001.