Model-Driven Data Warehousing

Integrate.2003, Burlingame, CA

Wednesday, January 29, 16:30-18:00

John Poole

Hyperion Solutions Corporation

Why Model-Driven Data Warehousing?

Problem statement:

Data warehousing is one of most challenging areas for integration of diverse, multi-vendor tools and applications.

Data warehousing projects offer significantly enhanced return-on-investment (ROI) when successfully integrated.

Standards-Based Approach to Model-Driven Data Warehousing

Object Management Group standards for modeling and model interchange: MDA, UML, MOF, XMI, CWM.

Java Community Process interface/platform standards: J2EE, JMI, JOLAP, JDMAPI.

Other: W3C XML, XSD, Web services models.

Of course, the truly important question is…

How can these various standards be combined together to provide seamlessly integrated data warehousing, business intelligence, and information supply chain environments?.

Session Overview

This session provides an overview of how the complete evolution of a data warehouse might be realized using OMG MDA and Java platform standards:

- Conceptualization and design (logical and physical modeling)

- Configuration, generation and deployment

- Ongoing operations and maintenance

Session Overview (continued)

Particular emphasis is placed upon:

- CWM and *standard meta data patterns*
- Centralized, model repositories
- Automated schema generation and tool initialization
- Platform models based on Java™ 2 and XML standards

Common Warehouse Metamodel (CWM):

- A common metamodel of the data warehousing and business intelligence domain
- Consists of a platform-independent metamodel definition
- Includes an XML-based inter-change format for metadata
- Also includes a mapping to a platform-independent API specification (CORBA IDL)
- Tools that standardize on CWM can readily share metadata via CWM-compliant XML files

Java Metadata Interface (JMI):

- A Java Community Process (JCP) effort to develop a standard Java API for metadata access and management
- Provides a standard metadata management API for the Java 2 Platform, Enterprise Edition (J2EE)
- Defines a formal mapping from any OMG standard metamodel (such as CWM) to Java interfaces
- Supports advanced metadata services, such as reflection and dynamic programming
- Tools that standardize on OMG metamodels (such as CWM) and are deployed in the J2EE environment have a standard Java API to use for metadata access and management

Java OLAP Interface (JOLAP):

- A Java Community Process (JCP) effort to develop a standard Java API for access to OLAP servers and multidimensional databases
- Provides a standard OLAP API for the Java 2 Platform, Enterprise Edition (J2EE)
- Uses the CWM OLAP metamodel as the basis for defining OLAP metadata
- Leverages JMI for managing OLAP metadata
- Introduces comprehensive OLAP query and result set models as the basis for OLAP inquiry and data retrieval

Java Data Mining API (JDMAPI):

- A Java Community Process (JCP) effort to develop a standard Java API for access to data mining tools
- Provides a standard Java API for access to tools for building data mining models, scoring of data using models, as well as creation, storage, access and maintenance of data and metadata supporting data mining results and transformations
- Like JOLAP, JDMAPI defines a metamodel for data mining metadata that is based on the CWM Data Mining model
- The JDMAPI effort has also contributed significantly back to CWM (in CWM 1.1, which has adopted the JDMAPI metamodel as its own)

XML for Analysis (XMLA):

- Web-centric access mechanism for OLAP services
- Based on the W3C's eXtensible Markup Language (XML)
- OLAP API is formulated in XML
- Protocol is based on the Simple Object Access Protocol (SOAP)
- Eliminates the need for any "heavy" client-side components
- Supports MDX and any other provider-specific OLAP languages

Management of the Model-Driven Data Warehouse

Primary elements consist of:

- Meta data service initialization
- Model construction
- Tool initialization
- Metamodel interoperability
- Data warehouse information flow

Meta Data Service Initialization

Key Components:

- Centralized meta data is handled in terms of shared models based on a common metamodel

- The common metamodel is CWM

- A JMI-enabled meta data service provides the central meta data (model) store for the data warehouse

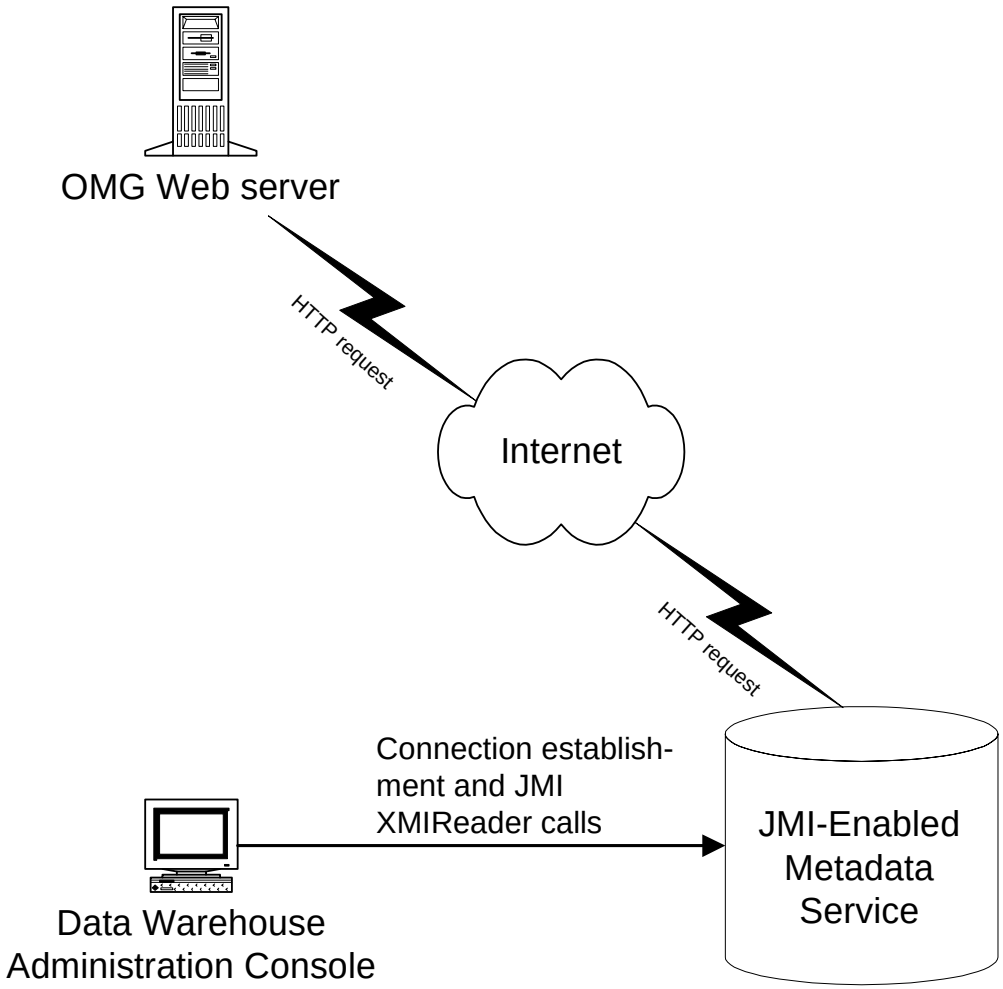
- Access to JMI service via the J2EE ™ Connector Architecture

Meta Data Service Initialization (cont)

Key Events:

- Connect to JMI service
- Download the common metamodel from publishing source (i.e., CWM rendered in XMI from OMG Web site)
- Initialize the model repository (load the metamodel)
- Generate the repository (i.e., create and launch a meta data server)

Meta Data Service Initialization (cont)



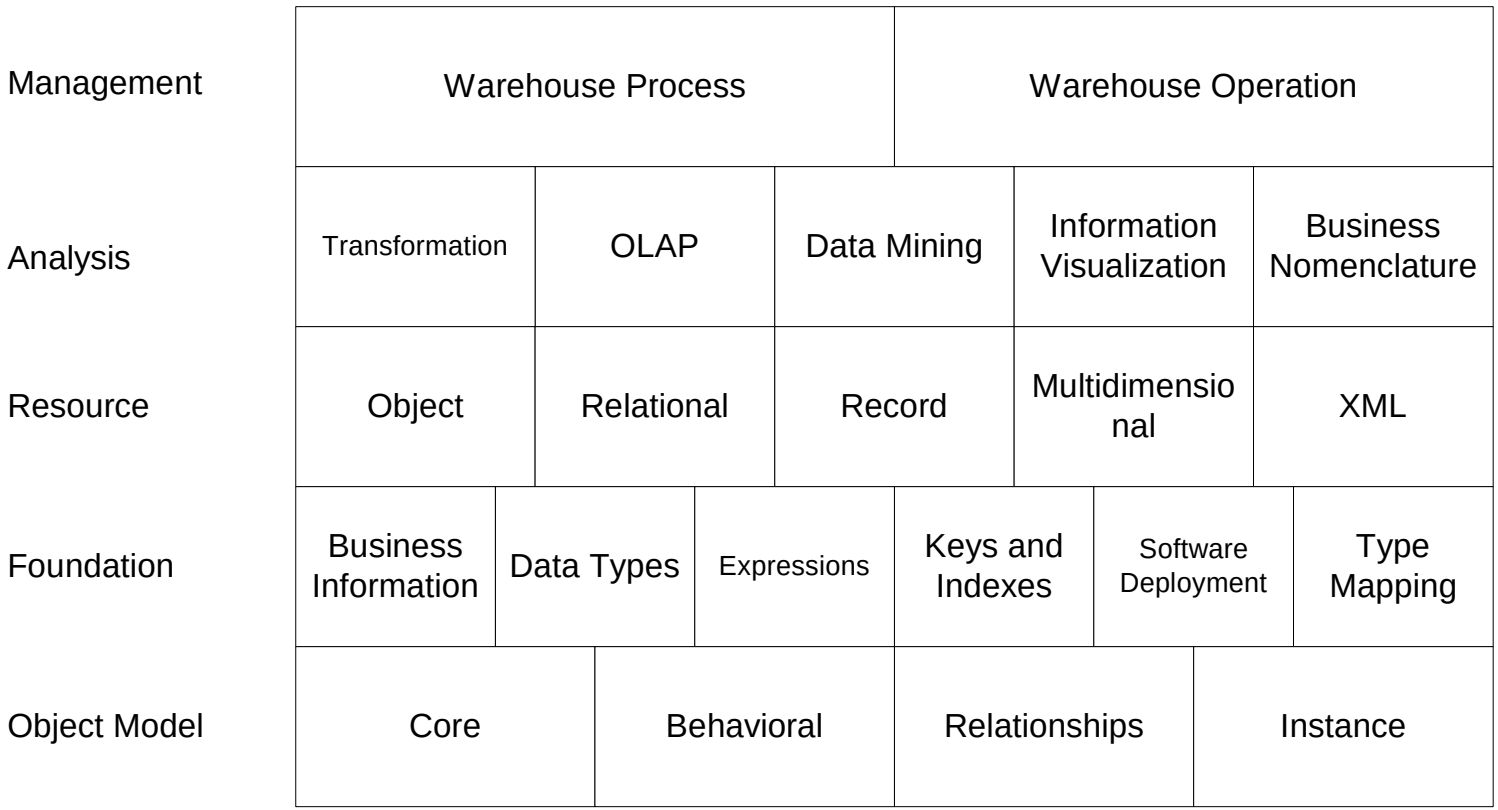
Meta Data Service Initialization (cont)

```
public interface com.mdservice.Connection {

    public void close() throws com.mdservice.ResourceException;

    public javax.resource.cci.ConnectionMetaData getMetaData()
        throws com.mdservice.ResourceException;

    public javax.jmi.reflect.RefPackage getTopLevelPackage()
        throws com.mdservice.ResourceException;

    public javax.jmi.xmi.XmiReader getXmiReader()
        throws com.mdservice.ResourceException;

    public javax.jmi.xmi.XmiWriter getXmiWriter()
        throws com.mdservice.ResourceException;
}
```

Meta Data Service Initialization (cont)

```
ConnectionFactory cxf = new ConnectionFactory();
Connection cx = cxf.getConnection( properties );
RefPackage msTlp = Connection.getTopLevelPackage();
XmiReader xmiReader = Connection.getXmiReader();
xmiReader.read( "http://cgi.omg.org/docs/ad/01-02-03.txt", msTlp );
```


Meta Data Service Initialization (cont)



Model construction

Key Components/Events:

- CWM-aware visual modeling tool

- Modeler constructs a complete data warehouse model by selecting and connecting modeling elements from various CWM packages

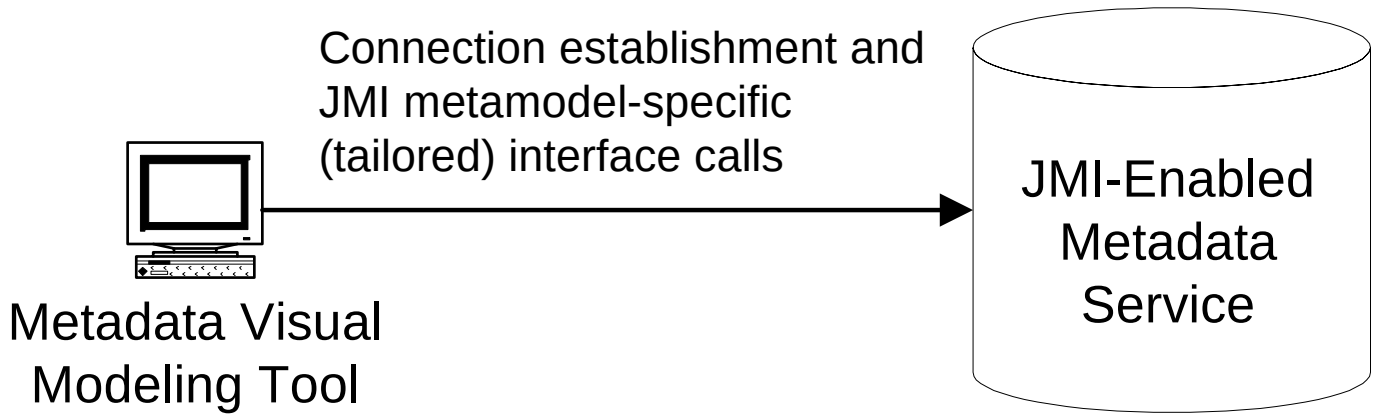
- Completed model is published to environment via the meta data service

Model construction (cont)

CWM packages used:

- Record package: Model the raw data feeds
- Relational package: Model ODS and dimensional store
- Transformation package: Model data transformations and source-target mappings
- OLAP package: Model multidimensional abstractions for analysis and reporting
- Warehouse Process and Warehouse Operation packages: Model ETL and other warehouse management and event recording processes

Model construction (cont)



Model construction (cont)

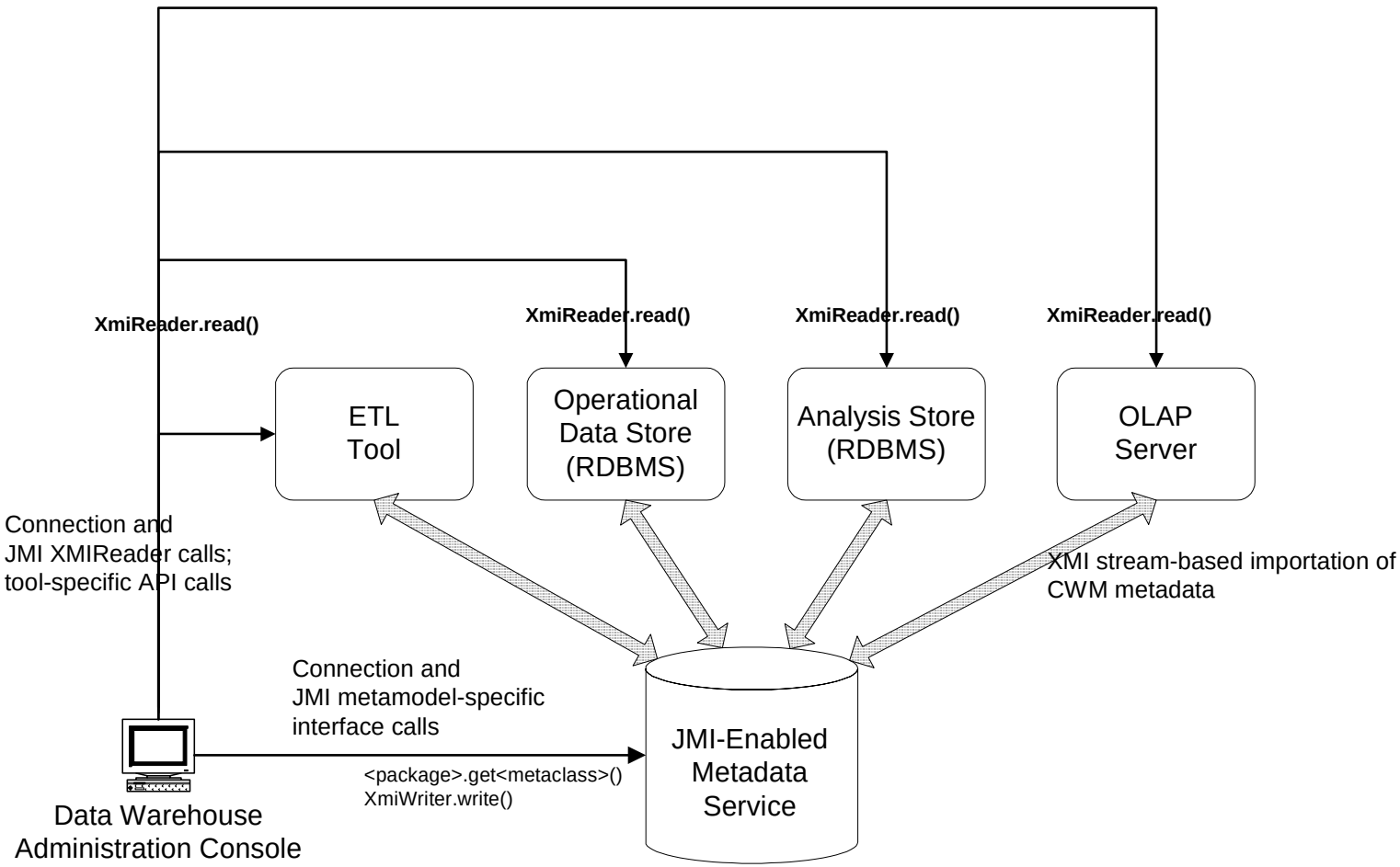
```
// Get the DimensionClass proxy
org.omg.java.cwm.analysis.olap.DimensionClass
   dc = olapPkg.getDimension();

// Create a Time Dimension

org.omg.java.cwm.analysis.olap.Dimension
  timeDim = dc.createDimension();
timeDim.setName( "Time" );
timeDim.setTime( true );
```

Tool Initialization

Key Components/Events:

- Physical generation of data warehouse
- Meta data initialization of JMI-enabled data warehousing tools
- Data warehouse administration tool interfaces central with meta data service
- Either programmatic (JMI) access to shared meta data or bulk interchange (XMI)

Tool Initialization (cont)



Metamodel Interoperability

Key Components:

- Advanced functionality based on MOF/JMI Reflection
- Metamodel-level interoperability
- OLAP Server (based on CWM OLAP metamodel)
- Advanced multidimensional visualization/reporting tool (MOF-compliant metamodel other than CWM)

Metamodel Interoperability (cont)

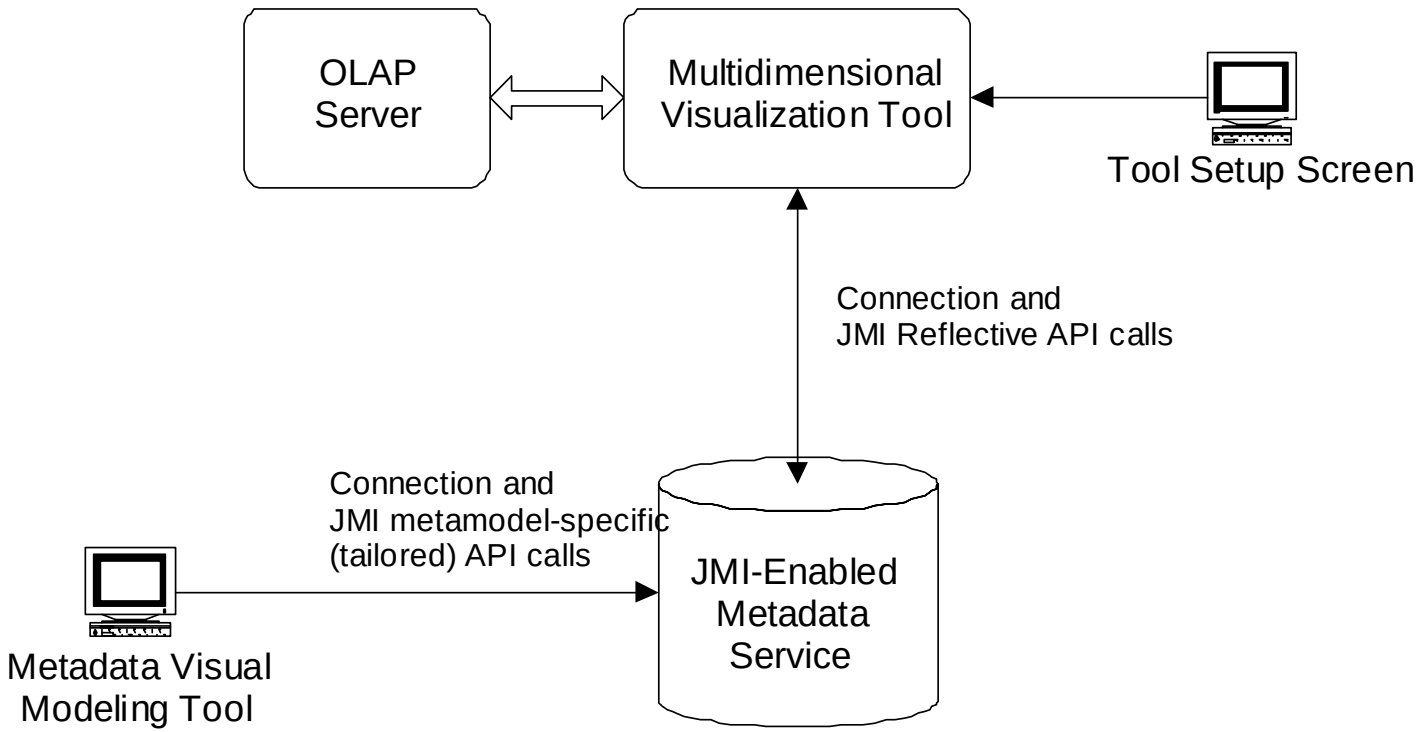
Key Events:

- Modeler connects to central meta data service

- Defines an instance of CWM Visualization metamodel and connects to OLAP model

- Develops dynamic programmatic script (Java) that uses JMI Reflective calls to initialize the non-CWM visualization tool based on the CWM OLAP+Visualization models

Metamodel Interoperability (cont)

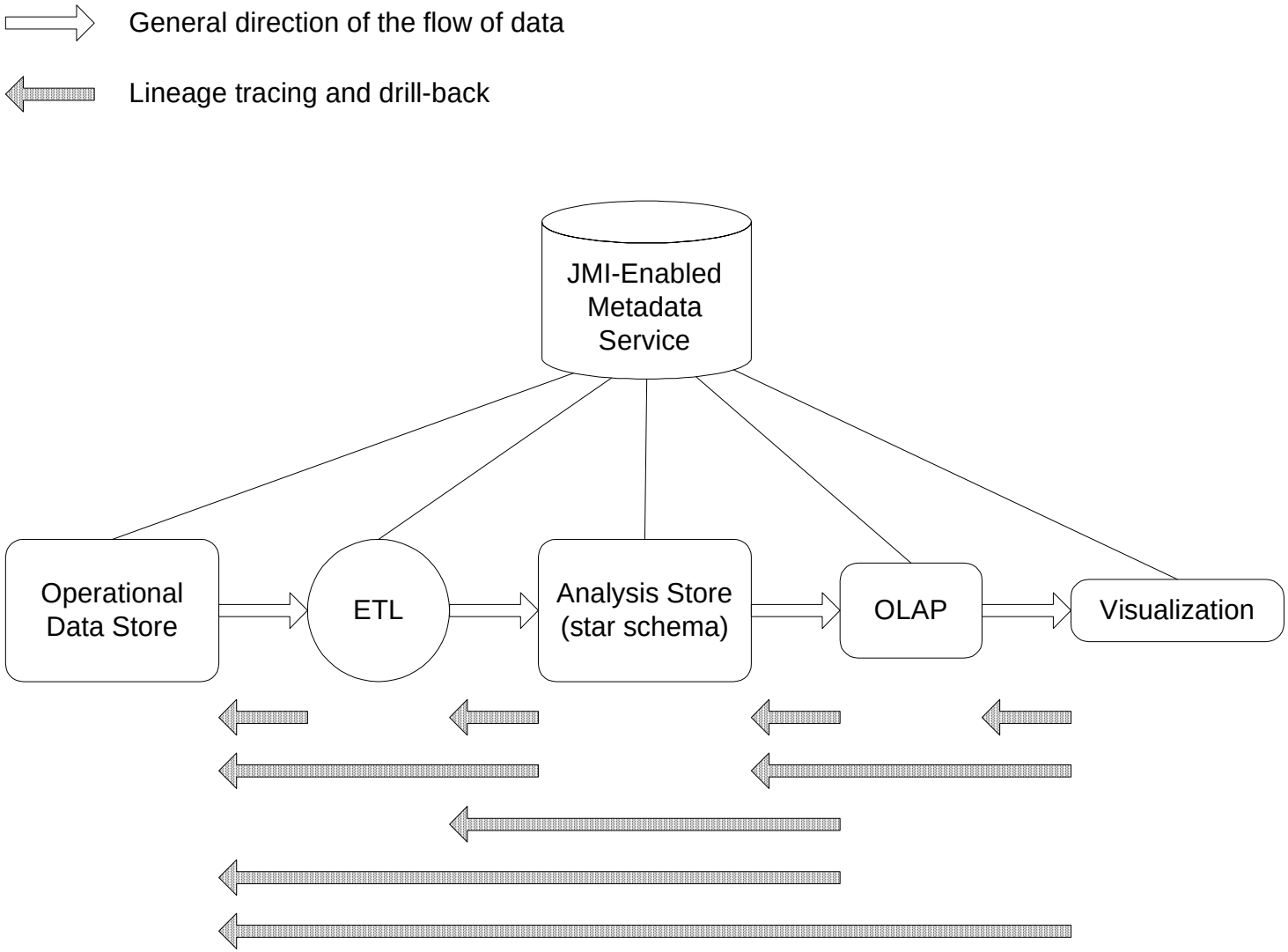


Data Warehouse Information Flow

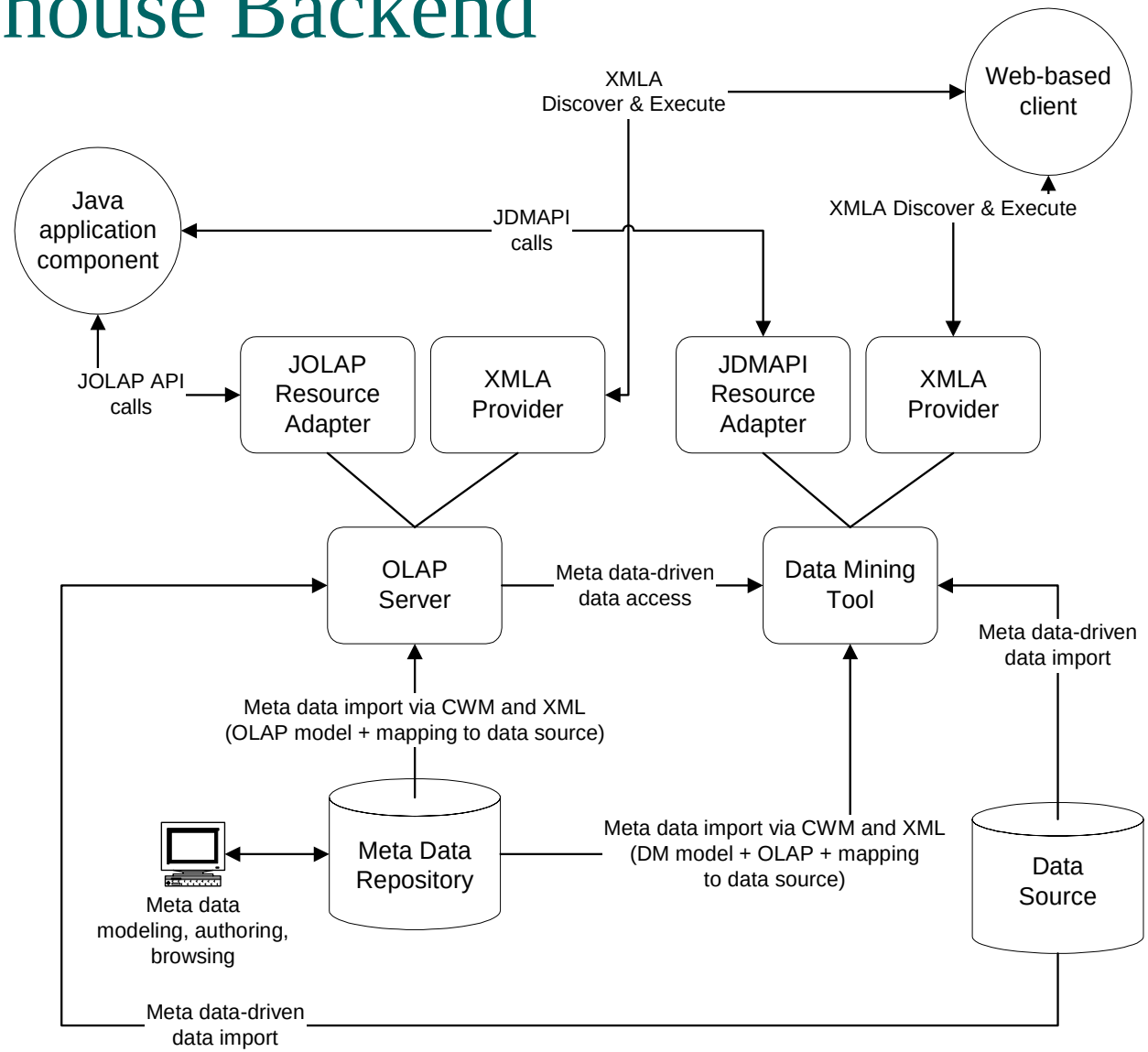
Key Components:

- Fully-integrated information supply chain
- Integration via centralized shared meta data and meta data –driven tools
- Meta data communication via JMI programmatic API and bulk interchange (XMI)
- MOF/JMI Reflection used to reconcile meta data integration between tools based on dissimilar metamodels

Data Warehouse Information Flow (cont)



Advanced Analytics Scenario: Data Warehouse Backend



Why Metadata Interchange Patterns?

CWM is a language for describing meta data to be interchanged.

CWM is highly flexible and expressive.

CWM does not, however, provide a means of expressing the *intent*, or *intended semantics*, of the content of an XMI file (this is beyond the scope of CWM).

Analog: English language sentences relative to context of a conversation; sense versus senselessness.

Why Metadata Interchange Patterns?

It might be reasonable to assume that an CWM XMI file contains a syntactically valid instance of the CWM metamodel (i.e., can always validate against the CWM DTD before exploring the model).

But is it reasonable to assume that an XMI file (whose content is currently *unknown*) contains a CWM instance that is *meaningful* to me?

For example: An instance of the CWM Relational metamodel organized as a Star or Snowflake schema.

Defining Metadata Interchange Patterns

1. *Projection* (or *semantic context*): Portions (sub-graphs) of the CWM metamodel that are semantically meaningful for some given interchange scenario.

2. *Restriction*: Boundaries on the number of instances of certain model element instances (or constraints on their values).

3. *Anchor*: Starting element(s) for search of projected/restricted instances.

Defining Metadata Interchange Patterns

Definition:

A *CWM Metadata Interchange Pattern* is an identified projection of the CWM metamodel, optionally with restrictions on instances of that projection, and possibly with one or more specified anchor elements.

Specifying Metadata Interchange Patterns

Mechanisms for pattern specification:

Human readable formal specification – Enumerates classes and associations of the projection as literals, along with OCL expressions restricting instances of the projection. (see *CWM Developer's Guide* – John Wiley & Sons, 2003)

Machine readable (and interchangeable) formal metamodel – References (directly or indirectly) classes and associations of the projection, along with OCL constraints restricting instances of the projection. (OMG CWM MIP RFP)

Summary

Model-driven data warehouse management based on CWM, UML, XMI, MOF, JMI

Model-driven analysis based on CWM, JMI, JOLAP, JDMAPI, and XMLA

Emphasis on model-based specification of the data warehouse and automated generation of the physical warehouse using meta data-driven tools

Bi-directional information supply chain

Enhanced ROI by reducing overall integration and life cycle costs

Summary (cont)

Use of standardized meta data interchange to enhance overall interoperability and simplify tool construction

Additional Sources of Information

CWM Forum Web site: http://www.cwmforum.org

CWM Book Series Web site:
http://www.wiley.com/compbooks/poole

OMG CWM Resource page:
http://www.omg.org/technology/cwm

OMG MDA page: http://www.omg.org/mda

Java Community Process home page: http://jcp.org

XMLA home page: http://xmla.org

Additional Sources of Information

http://www.wiley.com/legacy/compbooks/poole/CWM_Guide/patterns.html

http://www.wiley.com/legacy/compbooks/poole/patterns/StarJoin.pdf