Model-Driven Information Integration

"MetaMatrix is Enterprise Information Integration Software providing uniform access to disparate data sources"

metamatrix®

MetaMatrix presents the first solution to the EII problem. The MetaMatrix product suite, with its flagship MetaMatrix Server™ and MetaBase™ repository, provide a platform through which an enterprise can manage and provide uniform access to disparate information systems for timely application development in a hyper-competitive inter-networked world.

Model-Driven Information Integration

THE CHALLENGE OF INFORMATION INTEGRATION

Over the past twenty years, enterprises have created many diverse systems to manage their information and data. Individual systems combine a myriad of hardware configurations, operating systems, databases, and applications. Often, individual enterprises have found themselves with several disparate information systems among their divisions and departments, especially after mergers or acquisitions have broadened the scope and depth of the enterprise.

As the world, not to mention the enterprise, networks more completely, the enterprise needs to integrate its diverse systems to operate and analyze its resources more effectively. Numerous external sources, from partner information resources to real-time data feeds, have become available. The enterprise needs to marshal and integrate these disparate systems. At the heart of the systems integration challenge lies an information integration challenge.

Model-driven integration differs from the programmed integration. Programmed integration relies upon hard-coding a finite, and inextensible, solution to a particular challenge. Model-driven integration focuses on abstracting the information content into a model that describes the enterprise's information resources. This model captures the nature of the information the enterprise has within its systems and the way the enterprise uses data in its daily operations.

The data model does not rely on a particular hardware or operating system platform; instead, it contains standard constructs that show the data entities and operations. ***Once an enterprise captures its information resources in a model, it can easily integrate this information using a middleware server.***

Model-driven integration offers a complete solution because the data model can demonstrate:

- Platform independence, as the contents of the model and the systems modeled can represent one or more different types of actual physical systems.
- Singularity, as the model can contain as many information systems as needed.

Using model-driven integration middleware, such as the MetaMatrix Information Integration Server, also offers:

- Real-time access, because the model-driven solution does not manipulate copies of the data. Data consuming applications can report from or update the native systems.

- True integration. Data-consuming applications can access all data systems the enterprise has modeled.

Model-driven integration represents an evolution of the common and accepted practice of developing applications and representing systems using standard constructions in a model. The most common example of this practice, and the precedent for model-driven integration, is the use of Unified Modeling Language (UML).

THE HISTORICAL PRECEDENT: UML

As development languages and environments proliferated in enterprises, developers accustomed to varied, and often proprietary, systems and languages found themselves lacking a common way to express concepts such as workflow, relationships between entities, interaction, and other abstractions inherent in development.

In 1997, the Object Management Group (OMG) established a standard language for expressing these concepts. UML contains many types of models that represent standard presentation for a type of information. Hence, UML represents a metamodel, a language for constructing and relating diverse models.

The Unified Modeling Language (UML) gained rapid acceptance and set off a new trend in application development. Several companies, such as Rational and TogetherSoft, created software tools that implement the UML standard to create graphical tools that enterprises of all types can use to "design" applications. Figure 1 contains a UML class diagram depicting a portion of an application.

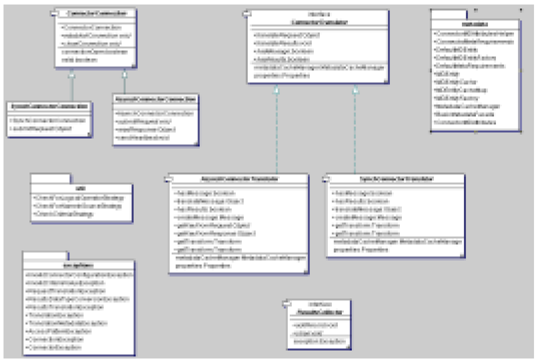

Figure 1: UML Class Diagram

This model contains standard constructs; anyone versed in UML can review the model and understand its contents. By modeling applications in UML before writing actual code, the enterprise garners many benefits:

- The Platform Independent Model (PIM) captures the application' s design.
- Certain tools can automatically generate code from a PIM.
- PIMs ease product migration from platform to platform.

The Platform Independent Model

Using UML, an enterprise can build a Platform Independent Model (PIM) that captures the design, business logic, and data requirements of each particular application. Independent of any platform, the model describes the goals of the application without referring to a specific operating system, hardware configuration, or even programming language.

The Platform Independent Model could describe an application written in Java, running on the Sun Solaris operating system on an Alpha or the PIM could refer to a VisualBasic application running on the Windows XP operating system on a quad Pentium IV server.

Automatic Code Generation through Tools

Certain new UML modeling tools can generate platform-specific code from a UML model. Using these tools and UML saves many enterprises a great deal of programming effort—and expense.

Simplified Application Migration

When an enterprise needs to migrate an application from one platform to another, a UML model makes the process straightforward. The UML model documents workflow and objects outside of the comments within the code and the memories of developers. Without the model, enterprises can find both occasionally unreliable for a complex task.

PIMs and System Integration

While enterprises have applied UML models successfully to application development, UML has not yet significantly affected the problem of systems integration. While Platform Independent Models offers excellent return on investment (ROI) for large enterprises, enterprises have not yet applied the lessons learned with PIMs and UML to the challenge of systems integration.

THE MODEL IN INFORMATION INTEGRATION

The success of the Platform Independent Model in application development leads to the use of PIMs for enterprise information sources. Enterprises use PIMs to ease their systems design and application design; in the same way, enterprises can use model-driven integration to solve their information integration challenges, which lies at the heart of their systems integration.

To integrate their information sources, enterprises can produce PIMs of diverse information stores. However, different enterprise information systems have different methods of storing information. A single metamodel, the abstraction that describes the structure of models, cannot accommodate all possible variations. With that, the meta-metamodel was born.

Differences in Metamodels

When constructing PIMs for software applications, enterprises needed only one metamodel, provided with the UML specification, to describe the models the enterprise created. However, because information sources, including relational databases, hierarchical databases, object databases, files, streaming information, and many others, can have radically different structures, enterprises will need more than one way to describe the models they need to create. These enterprises need more than one metamodel.

With that end in mind, the Object Management Group created the Meta Object Facility (MOF) standard, extending UML to apply it to modeling diverse information systems. The Meta Object Facility standard describes diverse metamodels, essentially abstracting a form and structure to describe metamodels. Figure 2 describes the Meta Object Facility's structure.

The Object, Relational, File, and XML information sources have individual structures, described in the model (M1 in the figure). Each information source has a metamodel, which determines not only the structure, but also the relationships between the entity types in a model (M2 in the figure). The meta-metamodel (M3 in the figure), then, is a metamodel that describes the contents of metamodels, in this case the types of entities shared by information systems of all types. The MOF standard describes the methods of describing all information system types, and can extend to include systems beyond the four described above.

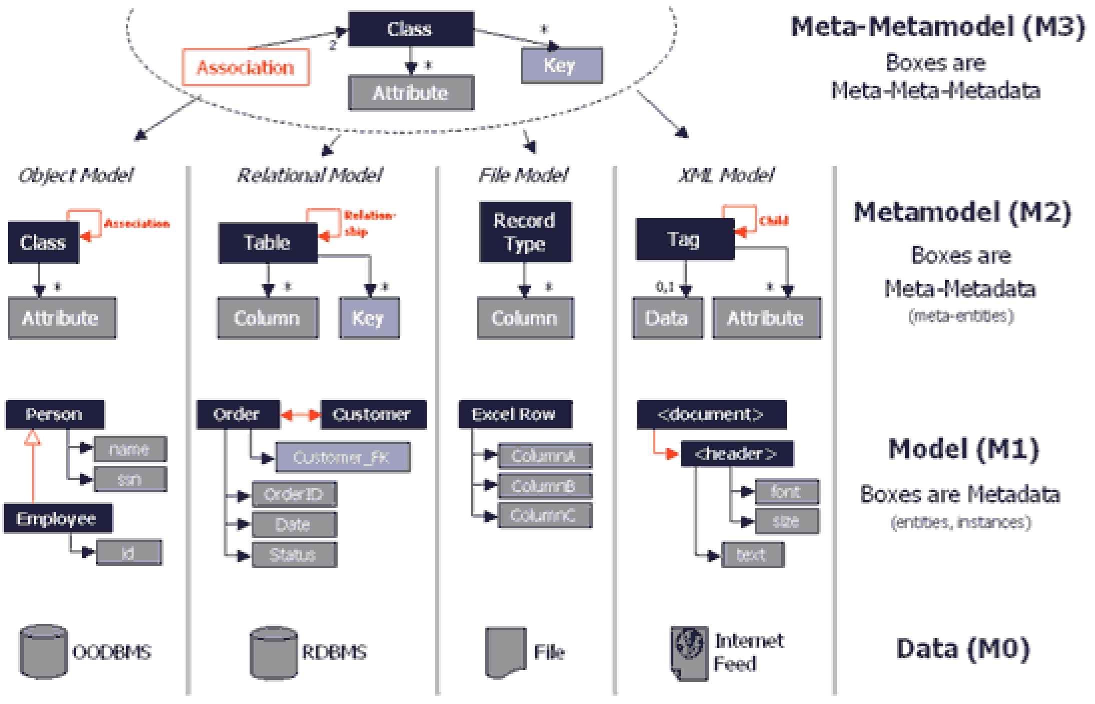

Figure 2: The MOF Standard

Modeling Data with Metadata

When modeling an information system, the enterprise captures the essence of the information within its systems—including technical aspects of the data, which describe the structure of the data, and business aspects of the data, which describe the way the enterprises uses the data. This captured essence is called metadata, data about the data. The Platform Independent Models include both the technical and business metadata, but remain platform independent because their contents remain descriptive in nature.

The Platform Independent Models contain metadata that describe the data in the physical sources. For example, in the relational metamodel, this includes table and column names, data types, keys, and foreign keys. This metadata is called physical metadata.

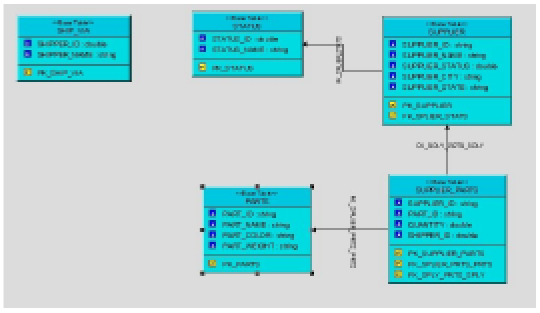
Figure 3: Physical Metadata Model

To simplify application development and to achieve information integration, enterprises need to describe, in a common way, the metadata in the disparate physical sources. For instance, system 1 has a column named **cust_id** with the data type of string. System 2 has a column of equivalent information named **cust_num** with the data of integer. The metadata model that can represent a column named **CustomerNumber**, which transforms and maps the relationship of **cust_id** and **cust_num**, would go a long way to solving the information integration problem. This metadata, which describes the data as the enterprise, or its data-consuming applications, use it, is called virtual metadata. Figure 4 displays a portion of a Platform Independent Model containing virtual metadata.

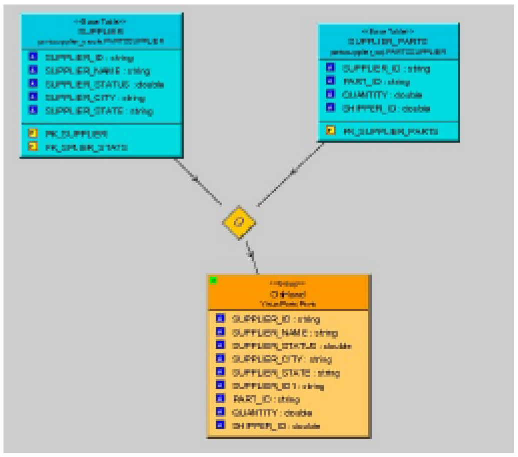
Figure 4: Virtual Metadata Model

The Tool for Modeling Metadata

A graphical tool for modeling diverse information sources through different metamodels should:

- Use the MOF standard to integrate Platform Independent Models.
- Support importing metadata from information systems and assembling structure from the information systems into a model.
- Recognize multiple metadata models.
- Support creation of relationships between models using the abstractions created within the MOF standard. For example, the tool can create relationships between columns in a spreadsheet and attributes in an object-oriented database.
- Have a repository that can recognize multiple metamodels. Within this repository, the enterprise can store its models.
- Support XML Metadata Interchange, the OMG standard format for interchanging metadata, models, and metamodels.

The MetaMatrix MetaBase™

MetaMatrix offers such a tool, MetaBase™. MetaBase contains three components as described in Figure 5.

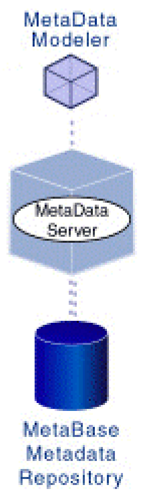

Figure 5: MetaBase™

MetaBase™ contains the following components:

- The MetaData Modeler™, a Java client that can model and import metadata from diverse sources. The MetaData Modeler also package metadata into models and displays these models as a hierarchy or as a UML/MOF diagram.
- The MetaData Server, a server that streams the XMI models between the MetaData Modeler and MetaBase Metadata Repository and can import/export XMI to any other XMI-compliant tool.
- The MetaBase Metadata Repository, an MOF-based repository that can instantiate multiple metamodels as well as store and interrelate models.

Modeling Data Presents a Partial Solution

However, modeling data as metadata only presents a partial solution to information integration by offering the conceptual integration. Ultimately, the enterprise needs a common way to access all of its systems using the information contained in the Platform Independent Models.

TRUE MODEL-DRIVEN INTEGRATION

To achieve integration at the information level, the enterprise must take the abstract and make it real, much as it takes the UML diagram and actualizes it into an application, created and compiled in a particular programming language. The enterprise needs to use the information within the metadata Platform Independent Models and create a specific, platform-based means of data access.

From PIM to PSM

Remember, the Platform Independent Models describe the information sources. Platform Specific Models (PSMs) must couple the design-time metadata constructions with platform-specific information that contains actual parameters and connection information for the enterprise information systems described in the PIMs.

Platform Specific Models contain technical information from the PIMs coupled with actual connection details for a particular data source and its system. ***Deployed within an integration platform, these PSMs provide the key for information integration through a Model-Driven Architecture.***

The Tool for Integrating Information

The ideal information integration server requires several features:

- A security model to limit or grant access, flexibly, to the enterprise information systems.
- A standard interface language, such as SQL, to hasten application development. The enterprise does not need to train its developers in a new Application Programming Interface.
- Extensible connector technology to enable the enterprise to connect to any and all possible enterprise information systems. The enterprise can develop its own connectors or use third-party connectors.
- Standard client interface options, such as XML and JDBC, to offer easy access for

the enterprise's information-consuming applications.

- Query planning and optimizing algorithms to ensure peak data access performance.
- Support for synchronous and asynchronous sources to ensure the enterprise can use the information it wants.
- Auditing to enable the enterprise to track usage patterns.
- Scalablity and fault tolerance to ensure continued operation.

The MetaMatrix Information Integration Server

MetaMatrix provides such a tool, the MetaMatrix Information Integration Server. Used in conjunction with MetaBase, the MetaMatrix Information Server provides data access capability using Platform Specific Models, called the virtual database. Figure 6 shows the complete MetaMatrix System, including MetaBase.

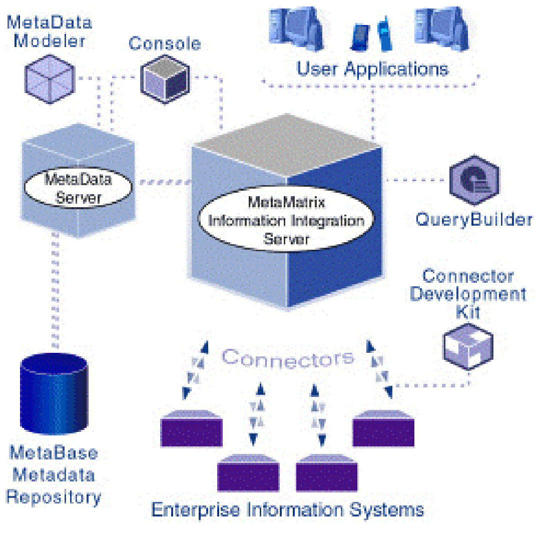
MetaData Modeler
Console
User Applications
MetaData Server
MetaMatrix Information Integration Server
QueryBuilder
Connector Development Kit
Connectors
MetaBase Metadata Repository
Enterprise Information Systems

Figure 6: The MetaMatrix System

In addition to MetaBase, the MetaMatrix System includes:

- The MetaMatrix Information Integration Server, a server that handles the communication and integration of the enterprise's information systems and presents the integrated view to the data-consuming user applications.
- The MetaMatrix Console, a Java application for configuring and managing the MetaMatrix System, including user accounts and security.
- The MetaMatrix QueryBuilder™, a Java application that enables the enterprise to create and test queries against the enterprise's information systems.
- Connectors, Java classes that translate standard requests into the enterprise information systems' native APIs and handle connections to those enterprise information systems. MetaMatrix offers a number of ready-made connectors and a framework and documentation for enterprises to create their own.
- The MetaMatrix Connector Development Kit, a Java application that simulates a live Information Integration Server to enable an enterprise to test a connector against a native enterprise information system without affecting the production server.

MODEL-DRIVEN INTEGRATION WITH METAMATRIX

The Process

Figure 7 describes the process of modeling metadata and creating Platform Specific Models using the MetaMatrix System's Model-Driven Architecture.

Information Integration, using MetaMatrix's Model-Driven Architecture, follows these steps:

1. An enterprise models its existing systems into Platform Independent Models that describe the nature and structure of those existing systems. These PIMs contain the physical metadata models.

2. The enterprise models its information using PIMs that describe how that enterprise uses the data. These PIMs contain the virtual metadata models.

3. The enterprise creates the transformations and mappings that describe the methods by which it derives the data it uses from the data stored in its systems.

4. The enterprise transforms the PIM(s) into PSMs for use in runtime information access and integration. This transformation process adds platform-specific information that the MetaMatrix Information Integration Server.

   An enterprise can transform a PIM into many PSMs, which means the enterprise only has to manage one PIM set but can convert/transform that metadata into PSM metadata for different uses.

5. The enterprise distills the technical metadata needed for actual communication with the enterprise information systems and combines it with connection properties to create the Platform Specific Models. Deployed on an integration platform, these PSMs show a direct route to access data in its physical sources based on the business needs modeled in the metadata.

6. The enterprise's information-consuming applications request information from the integration server. This integration server uses these PSMs at runtime to access the data sources when the enterprise's information-consuming applications request it.

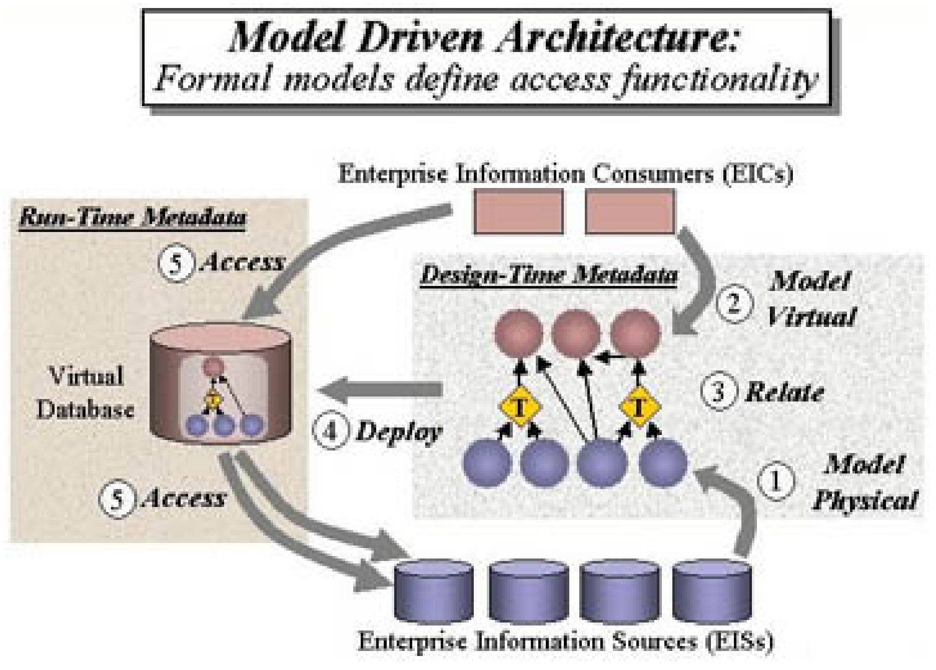
Model Driven Architecture:
Formal models define access functionality
Enterprise Information Consumers (EICs)
Run-Time Metadata
5 Access
Virtual Database
T
5 Access
Design-Time Metadata
2 Model Virtual
3 Relate
T
T
4 Deploy
1 Model Physical
Enterprise Information Sources (EISs)

Figure 7: Creating PIMs and PSMs

The Benefits

The MetaMatrix Model-Driven Architecture for information integration offers a number of benefits for the enterprise that impact the enterprise' s bottom line. These benefits include:

- Adherence to standards, which assure the enterprise that MetaMatrix works with other standards-compliant tools and technology.
- Platform Independent Models (PIM) that capture the enterprise' s information assets and can present this information in an organized, graphic fashion.
- Design-time models independent of runtime models enable the enterprise to modify and model more information without impeding data access through the integration server.
- Extensibility that enables the enterprise to model, integrate, and access legacy, existing, and even future data storage technologies.

The MetaMatrix System, by using a modeling and implementation process that many enterprises already use for application design, extends a familiar methodology in a new direction. Enterprises can integrate their existing information systems easily without costly or confusing conversions and can access the information in those systems in real-time.

The MetaMatrix System looks something like a database and something like middleware. As a platform independent systems integration technology, it represents the first of its kind and the leader in what will become the dominant trend in systems, and information, integration.

This and other MetaMatrix White Papers are available for download at www.metamatrix.com/whitepapers.html

To contact MetaMatrix email solutions@metamatrix.com