XMI and the Many Metamodels of Enterprise Metadata

Joram **Borenstein**

Joshua **Fox**

Abstract

Enterprise metadata appears in many languages and formats. XML provides a standard and consistent language for metadata, simplifying both interchange and parsing. But simply storing metadata as an XML file (be it XSD, BPEL, WSDL, J2EE EJB descriptors files, or any of dozens of proprietary formats) does not automatically and formally capture the full richness of the given metadata language. Even if XSDs are used to constrain syntax, they cannot define all possible structures and relationships, nor can they express the meaning of metadata in its business context.

The XML Metadata Interchange standard (XMI) was developed by the Object Management Group (OMG) to answer the need for exchanging and storing metadata in a variety of different metadata languages. In XMI, the metamodel of the given language is described according to the Meta-Object Facility (MOF), a metamodeling standard closely based on Unified Modeling Language (UML). Each metamodel is codified in a special XSD or DTD which shows the structure that a given metadata artifact can take. For example, the relational database metamodel include tables, columns and foreign keys, while a business process metamodel includes inputs, outputs, events, and conditions.

Using XMI metamodels, development and runtime tools share metadata in the form of complex object models; these models can then be stored in a universal metadata repository that is open to any new metadata format that already exists or that may come into existence in the future. Indeed, the open-source Eclipse project has successfully integrated tools for Java, EJB, relational databases, and other formats through the medium of XMI. Many other UML modeling tools have also adopted XMI as their interchange format.

But XMI goes beyond metadata interchange and storage to achieve the business goals of enterprise architecture, by enabling the metamodel-driven automation of metadata processing. With an XMI metadata system based on flexibly-defined metamodels, all enterprise metadata is understood in the same way across an organization and across industries. The formally-captured metadata lends itself to having its semantics captured as well, by mapping to a central ontology model, which encodes the organization's real-life business concepts.

XMI and semantic mappings can be used to comprehensively support enterprise architecture in the use of business processes. With XMI and semantic mapping, business processes across an organization's many departments and business lines can reuse and share metadata. For example, if IT managers want to decommission a legacy data store, they can turn to their universal metadata repository to discover the interactions of the data store with other IT assets. In the course of this process, they may find that a legacy COBOL application draws data from the data store, with a J2EE application in turn relying on the COBOL application. By reference to the semantic mappings, the managers then use the repository to precisely determine the business meaning of the data. This gives IT managers

the information they need to decide whether to decommission or not, and if so, what other data stores can provide the relevant data.

XML and XSD provide standards for structuring data and syntax respectively. XMI goes beyond them in allowing the interchange of metadata in any of the many existing or future metadata languages. With XMI-based metadata in place, semantics can be added through ontological mappings, rounding out the picture of automated Enterprise Metadata Management.

This presentation will outline the standards and architecture for Enterprise Metadata Management based on XMI, illustrating with case studies from enterprise deployments.

Table of Contents

1. Introduction

Enterprise IT Departments operate the systems that support the business goals of the organization. The data of IT systems is called "Metadata," defined as the structures and descriptions which characterize business information and business processes. To process, store, and exchange metadata, IT departments need to find a convenient and standard format. And XML provides this format; XSD, BPEL, WSDL, J2EE EJB descriptors files, and others are all XML-based standards for storing and exchanging metadata (in addition, metadata is encoded in legacy non-XML legacy formats, such as COBOL Copybooks, Data Definition Language for RDB, and Interface Definition Language for software components).

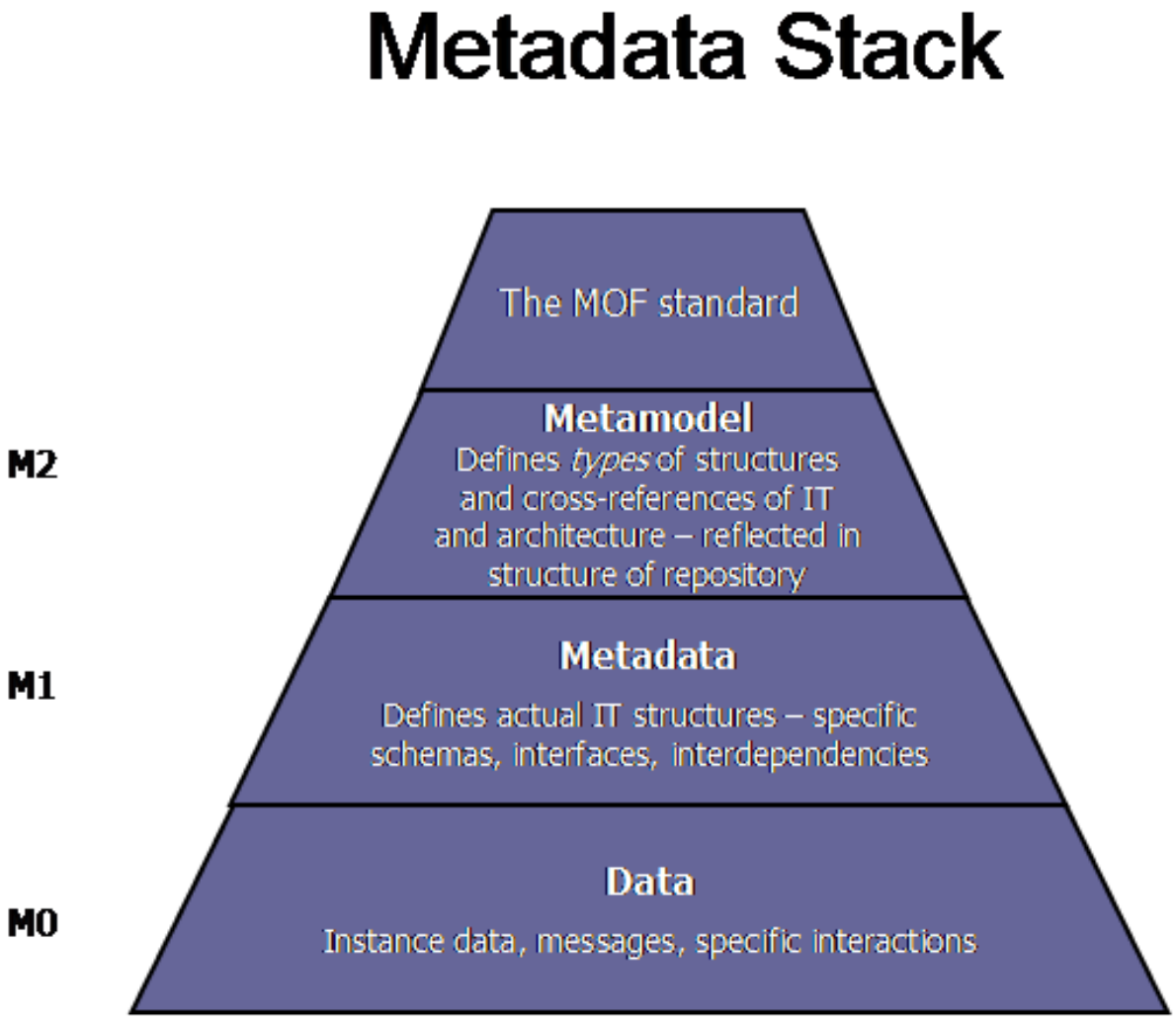
But just as ordinary XML files are best managed by using a schema to constrain their structure, so too the permissible structures for each "language" of metadata need to be defined and declared. Thus, an XSD schema for each language is essential. But this is insufficient. XSDs cannot automatically and formally capture the full richness of a given metadata language. XSDs can be used to constrain syntax, but they cannot define all possible structures and relationships. For example, metadata languages can designate that a certain item is an "ID." This means that no instance of this item occurs twice for different entities. But this does not express whether the designated ID is an industry-standard business identifier, or perhaps a "surrogate" key assigned automatically and unconnected to business realities.

More importantly, schemas for metadata languages do not express the relationships between different types of metadata - for example, that a given JavaBean field is fed from a given database column, or that the result of a business-logic calculation is placed into a given `simpleType` field in an XML message.

2. XMI and MOF

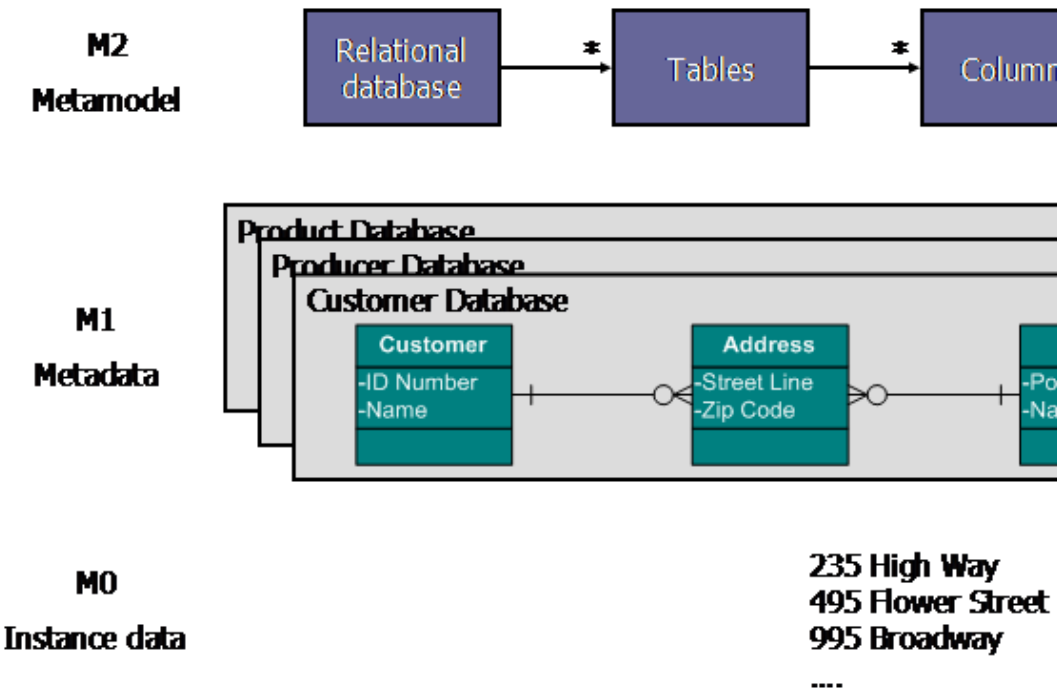
The XML Metadata Interchange standard (XMI) was developed by the Object Management Group (OMG) to answer the need for exchanging and storing metadata in a variety of different languages ([OMGXMI]). First, XMI gives the same general structure to metadata, regardless of the "language" used (e.g. relational databases, business processes, and application interfaces alike receive a consistent format). Second, the metamodel of the given metadata language (in other words, the model which constrains the metadata) is also described in a standardized format such as either DTD, XSD, or meta-XMI. For example, the relational database metamodel - the "language" or RDB - include tables, columns, primary keys, alternate keys, and foreign keys. Each table holds within it some columns, and each key is built of one or more columns. This is specified formally in a metamodel.

3. Applying XMI to Interchange



Data - Metadata (Model) - Metamodel - Meta-metamodel (MOF)

Figure 1. Metadata Stack



Example: Metadata Levels

Figure 2. Example: Metadata Levels (M0, M1, and M2)

With shared metamodels, development and runtime tools exchange metadata in the form of complex object models expressed in XMI; these are then stored in a universal metadata repository which is open to any new metadata format whose metamodel has been formalized. Indeed, the open-source Eclipse project has successfully integrated tools for Java, EJB, relational databases, and other formats through the medium of XMI; many other UML modeling tools have also adopted XMI as their interchange format.

The theoretical underpinnings of XMI are defined by the Meta Object Facility (MOF) ([OMGMOF]), an OMG metamodeling standard closely based on Unified Modeling Language (UML). MOF is itself a *meta-metamodel*, an English-language specification describing how one may build metamodels. Thus, for example, MOF provides a specification for how to model the fact that an interface has methods, or that a Web service has service endpoints. This is done using the familiar "boxes and arrows" notation of UML, with some minor constraints necessitated by the nature of metamodeling.

XMI is a mature technology: Version 1.4 was released in 1999, and the greatly improved version 2.0 came out in 2002. Likewise, MOF has been in version 1.4 since 2002 and is now about to transition to 2.0.

It is important to correctly understand the relationship between the basic data (often denoted M0), the models or metadata (M1), metamodel (M2), and meta-metamodel (M3); the figures below show the stack of meta-levels, and illustrate them with an example (Figure 1, “Metadata Stack” and Figure 2, “Example: Metadata Levels (M0, M1, and M2)”). M0 is the basic data, the lifeblood of the business, for example the address "233 High Way" or the price "$291.70". M1 is the metadata: schemas and interfaces describing the structure of this data, for example "the Customer table" or "the Zip Code column." M1 is at the heart of IT; it is the information describing how the enterprise shares and stores its data. M2 is the metamodel, or the "IT language" (in this example, a MySQL relational database). M3 is the MOF specification itself, which allows us to draw the boxes-and-arrows of UML which say "a MySQL database has tables, and each table has zero or more columns."

Though we distinguish the different meta-levels, they actually have a lot in common. Just as metadata describes data, so too do metamodels describe the permitted configurations of metadata; likewise, the MOF standard itself (the meta-metamodel), gives a language for describing and interchanging metamodels. Still, the different meta-levels have quite different use cases: Data is used by the business, metadata is used by IT, and metamodels are used by metadata repositories (particularly metadata repositories that allow metamodels to be configured rather than hard-coded). Fortunately for IT, there is generally less metadata than data, and much less variety in metadata languages (metamodels) than in metadata. A given enterprise, for example, may have millions of database rows, hundreds of schemas, but only a few different varieties of data bases are installed.

4. Example of XMI

For an example of XMI with accompanying metamodel, let's look at this following example, adapted from the OMG XMI Standards document [XMISPEC]. It is a simple illustration of the metamodel and some sample data for an IT system.

In considering this sample, we begin with a look at the metamodel. (See below.) This metamodel tells us how Applications and Datastores are related to each other. Specifically, it says that each Application persists its data in one or more Datastores, and that each Datastore persists data for one or more Applications. It also states that RDBs and XML Documents are different types of Datastores. The metamodel in this case is itself serialized in XMI--the dialect of XMI used for UML. In fact, the metamodel is more usually serialized in XSD or DTD for easier validation of the metadata XMI document. In this example, we use XMI/UML for the metamodel, highlighting the fact that MOF, the language for specifying metamodels, is itself a subset of UML.

```
<XMI version="1.1" xmlns:UML="org.omg/UML1.3">
 <XMI.header>
  <XMI.model xmi.name="Application" href="Application.xml"/>
  <XMI.metamodel xmi.name="UML" href="UML.xml"/>
 </XMI.header>
 <XMI.content>
  <UML:Class name="Application" xmi.id="Application"/>
  <UML:Class name="Datastore" xmi.id="Datastore"/>
  <UML:Class name="RDB" xmi.id="RDB" generalization="Datastore"/>
  <UML:Class name="XMLDoc" xmi.id="XMLDoc" generalization="Datastore"/>
  <UML:Association>
   <UML:Association.connection>
   <UML:AssociationEnd name="persistsIn" type="Datastore"/>
   <UML:AssociationEnd name="persistsFor" type="Application"/>
   </UML:Association.connection>
  </UML:Association>
 </XMI.content>
</XMI>
```

We continue our example with a sample of XMI metadata (below), which is structured in conformance to the metamodel presented above. The XMI tells us that our corporation has an Application called Shipping. The Shipping application persists its data into a Relational Database called ShippingDatabase, and also into an XML Document called InvoiceMessageFormat. Note that this XMI document specifies *metadata*. The actual data (for example data indicating that "Widget XYZ was shipped to customer #1234 on 14-Jan.-2005") is stored on the *data*level as an RDB row or an XML element, and is not within the use cases addressed by the XMI standard.

```
<XMI version="1.1" xmlns:Application="com.corporation/Application">
 <XMI.header>
  <XMI.model xmi.name="Shipping" href="Shipping.xml"/>
  <XMI.metamodel xmi.name="Application" href="Application.xml"/>
 </XMI.header>
 <XMI.content>
  <Application:Application name="Shipping">
   <Application:Application.persistsIn>
    <Application:RDB name="ShippingDatabase"
     xmi.id="ShippingDatabase"/>
    <Application:XMLDoc name="InvoiceMessageFormat"
     xmi.id="InvoiceMessageFormat"/>
   </Application:Application.persistsIn>
  </Application:Application>
 </XMI.content>
</XMI>
```

5. Understanding the Business

XMI helps IT achieve the goals of enterprise architecture, both with metadata exchange and storage, but also by enabling the metamodel-driven automation of metadata processing. With an XMI metadata system based on flexibly defined metamodels, all enterprise metadata is understood in the same way across an organization and across an industry. The formally-captured metadata lends itself to having its semantics captured as well, by mapping to a central ontology model of the organization's business concepts.

6. Metadata, Information Models, and Semantic Mapping

XML and XSD provide standards for structuring data and syntax respectively. XMI goes beyond them in allowing the interchange of metadata in any of the many existing or future metadata languages. With XMI-based metadata in place, semantics can be added through ontological mappings, rounding out the picture of automated Enterprise Metadata Management.

Ontology is a modeling system designed for describing the real world and can be used for metamodeling the varieties of IT systems used in the business. As discussed in previous XML Conference presentations ([JF02], [JF03]), ontology allows the formal expression of classes of entities, the relationship between these entities, and the constraints on the relationships. Ontology is used for building metamodels which express the IT domain formally and precisely, and so can be used for automated functionality in understanding and processing metadata.

The fundamental unit in ontology is the class, a set of real-world entities that have certain defined relationships to other entities in common. Classes can be related in an inheritance relationship, and a class can have properties (also known as associations, characteristics, or relationships) that relate it to other classes. Constraints can be defined to indicate restrictions on the possible values for properties of a given class. For example, a constraint may express a mathematical expression, e.g. that `lengthInMeters= lengthInMillimeters/1000`.

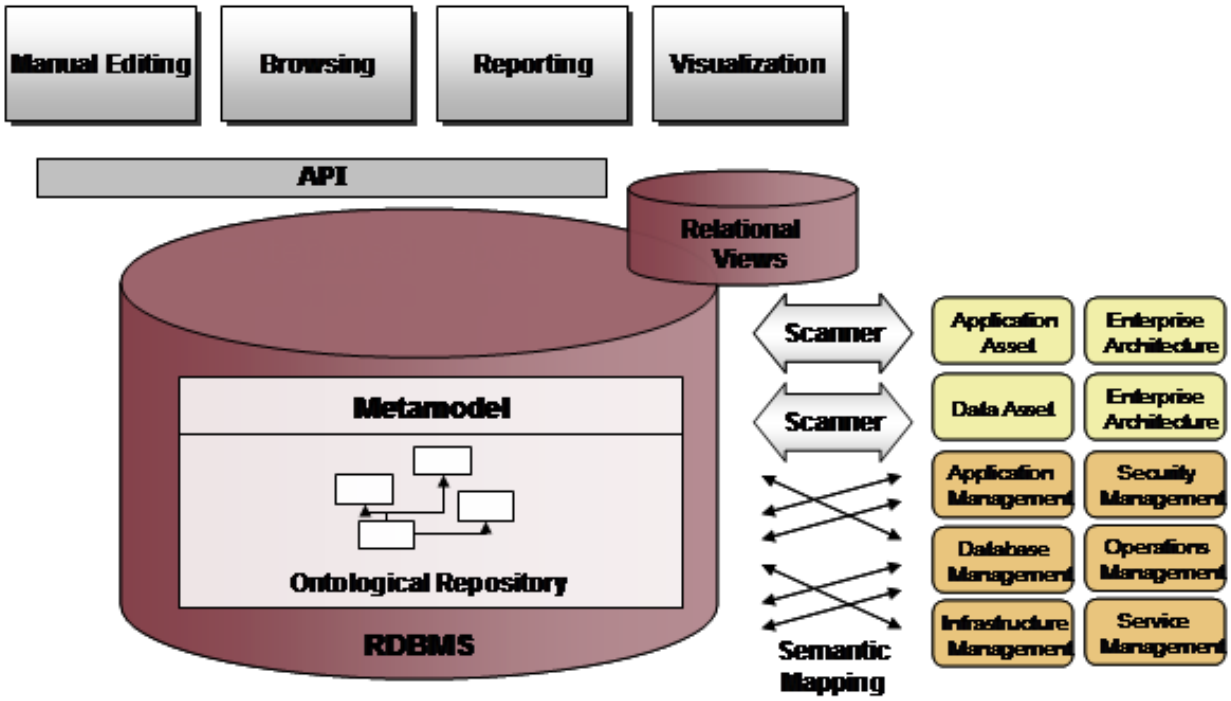
These previous XML Conference papers ([JF02],[JF03]) described semantic mapping in the context of applying semantics for XSLT generation or XSD schema discovery. This same mapping process can be used in the management of metamodels. In this context, semantic mapping is the process of expressing the semantics of the various IT metamodels by associating their elements with the concepts in the central ontological metamodel. Semantic mapping does require

up-front effort. However, there are generally many fewer metamodels (IT languages) than schemas, and the effort in a single metamodel bears fruit for the large amount of metadata expressed in each language. Even without an ontological metamodel, this sort of analysis must be done - but doing it with an ontological metamodel allows the work to be captured formally and then reused automatically.

Thus, by presenting the metamodel in accordance with the MOF standard, we achieve flexibility in supporting a variety of metamodels. Without MOF, metamodels are specified in various ways: as an XSD schema, an SQL/DDL for an RDB Schema, or in software code. But when metamodels are expressed ad hoc, there is no way to relate different metamodels, to formally capture the similarities and differences, and to transform from one metamodel to the other. Unless metamodels (metadata languages) are encoded consistently and their semantics captured, he well-known problem of disconnected and meaningless data or metadata can also appear at the metamodel level.

On the other hand, with XMI and semantic mapping, business processes across an organization's many departments and business lines can reuse and share metadata. For example, if managers want to decommission a legacy data store, they can turn to the universal repository to discover that it plays a role in a complex interaction. Perhaps they find that a legacy COBOL application draws data from this data store, with a J2EE application in turn relying on the COBOL application. The managers then use the system to precisely determine the business meaning of the data, by reference to the semantic mappings. This gives them the information they need to decide whether to decommission or not, and if so, what other data stores can provide the relevant data.

This architecture (Figure 3, “Metadata Management (Overall Architectural View)”) enables a wide variety of functionalities unachievable without formally captured metamodels. The system allows the user to browse the metadata and data and to generate reports on metadata in the enterprise, whatever form it may take. Such an architecture enables automatic transformation of metadata from one form to another, e.g., from an RDB schema to an XSD. When one IT system impacts another - for example, the decommissioning of a database will cause an application to crash - the MOF-based metadata system will warn of such an impact.



Metadata Management Architecture

Figure 3. Metadata Management (Overall Architectural View)

7. Case Study

In a case study typical of enterprise IT, one of America's largest public power companies had a wide variety of interconnected metadata assets, including databases, business process models, and messaging formats.

Like many enterprises, this power company has a highly complex and poorly understood IT environment that results in wasted IT spending, compromised business information quality, and slow IT response times. The company attempted to track its metadata with an inflexible metadata repository, and found that the repository itself had become another silo. It needed a way to view *all* metadata, from whatever source, in whatever format, and in whatever metadata persistence system, using a uniform and easily accessible view. The company migrated its metadata from the legacy repository to a flexibly-definable metadata management system, transforming the metadata from one metamodel to another.

By storing business and technical metadata in one flexibly-metamodeled repository, in which the metamodel could be adjusted to meet their needs, and in which different metadata elements could be mapped to each other and to a semantic business model, the data managers achieved the ability to browse their metadata in a uniform way and also to understand the relationships and dependencies between IT infrastructure, personnel, and business processes.

The company now has a methodology for coping with its ever-growing range of metadata languages and metadata sources by expanding the metamodel to suit the dynamic nature of the IT infrastructure.

8. Conclusion

Enterprises are realizing the importance of metadata in characterizing their data, but they are now encountering a complex variety of metadata formats and languages in their IT systems. The MOF metamodeling standard, together with the XMI interchange format, enables the formal capture of metadata in these different languages. Metadata can now therefore be analyzed, interrelated, and compared regardless of its "language." Most importantly, metadata can be related to the real-life needs of enterprise IT, as expressed in a central ontological metamodel, so that any differences between capabilities and needs can be identified and resolved.

Bibliography

[JBJF05] *Enterprise Semantics: Aligning Service Oriented Architecture with the Business* Joshua Fox and Joram Borenstein Web Services Journal, May 2005 Available at http://webservices.sys-con.com/read/79276.htm .

[JBJF03] *Semantic Discovery for Web Services* Joram Borenstein and Joshua Fox Web Services Journal, April 2003 Available at http://webservices.sys-con.com/read/39718.htm .

[JF03] *Know What Your Schemas Mean: Semantic Information Management for XML Assets* Joshua Fox XML Conference, Dec. 2003 Available at http://www.idealliance.org/xmlusa/03/call/xmlpapers/04-03-04.403/.04-03-04.html .

[JF02] *Generating XSLT with a Semantic Hub Transformations for the Semantic Web* Joshua Fox XML Conference, Dec. 2002 Available at http://www.idealliance.org/papers/xml02/dx_xml02/papers/04-04-05/04-04-05.pdf .

[OMGXMI] *CORBA, XML and XMI Resource Page [XMI Homepage]* OMG, Available at http://www.omg.org/technology/xml/

[OMGMOF] *Data Warehousing, CWM and MOF Resource Page [MOF Homepage]* OMG, Available at http://www.omg.org/technology/cwm/

[XMISPEC] *OMG XML Metadata Interchange (XMI) Specification V1.2* OMG, Available at http://www.omg.org/docs/ptc/01-08-27.pdf

Biography

Joram **Borenstein**
Director of Marketing
Unicorn Solutions [http://www.unicorn.com]
New York
New York
United States of America

Joram Borenstein serves as Director of Marketing at Unicorn Solutions (www.unicorn.com). His previous experience includes managing the rollout of content management software platforms. He has written and lectured extensively semantics, ontologies, Web services, and grid technologies. He can be reached at info@unicorn.com [mailto:info@unicorn.com]

Joshua **Fox**
Chief Software Architect
Unicorn Solutions [http://www.unicorn.com]
New York
New York
United States of America

As Chief Software Architect of Unicorn Solutions [http://www.unicorn.com], Joshua Fox worked on the design and development of the Unicorn system for integrated IT management. Fox's previous experience includes the design and development of large-scale distributed Internet systems for collaboration over the Internet. He has published and lectured extensively in the fields of software engineering and data management. For links see joshuafox.com [http://www.joshuafox.com].