Using MDA to Integrate Corporate Data into an SOA

Randall M. Hauch
Chief Architect

presented at

MDA, SOA and Web Services:
Delivering the Integrated Enterprise – Practice, not Promise

MetaMatrix Products at Work Today

Providing access to integrated information for …

- Government
- Financial services
- Telecommunications
- Life Sciences
- Manufacturing
- Pharmaceuticals

DISA

SAP®

NORTHROP GRUMMAN
Information Technology

BRITISH AMERICAN TOBACCO

Merrill Lynch

MOTOROLA

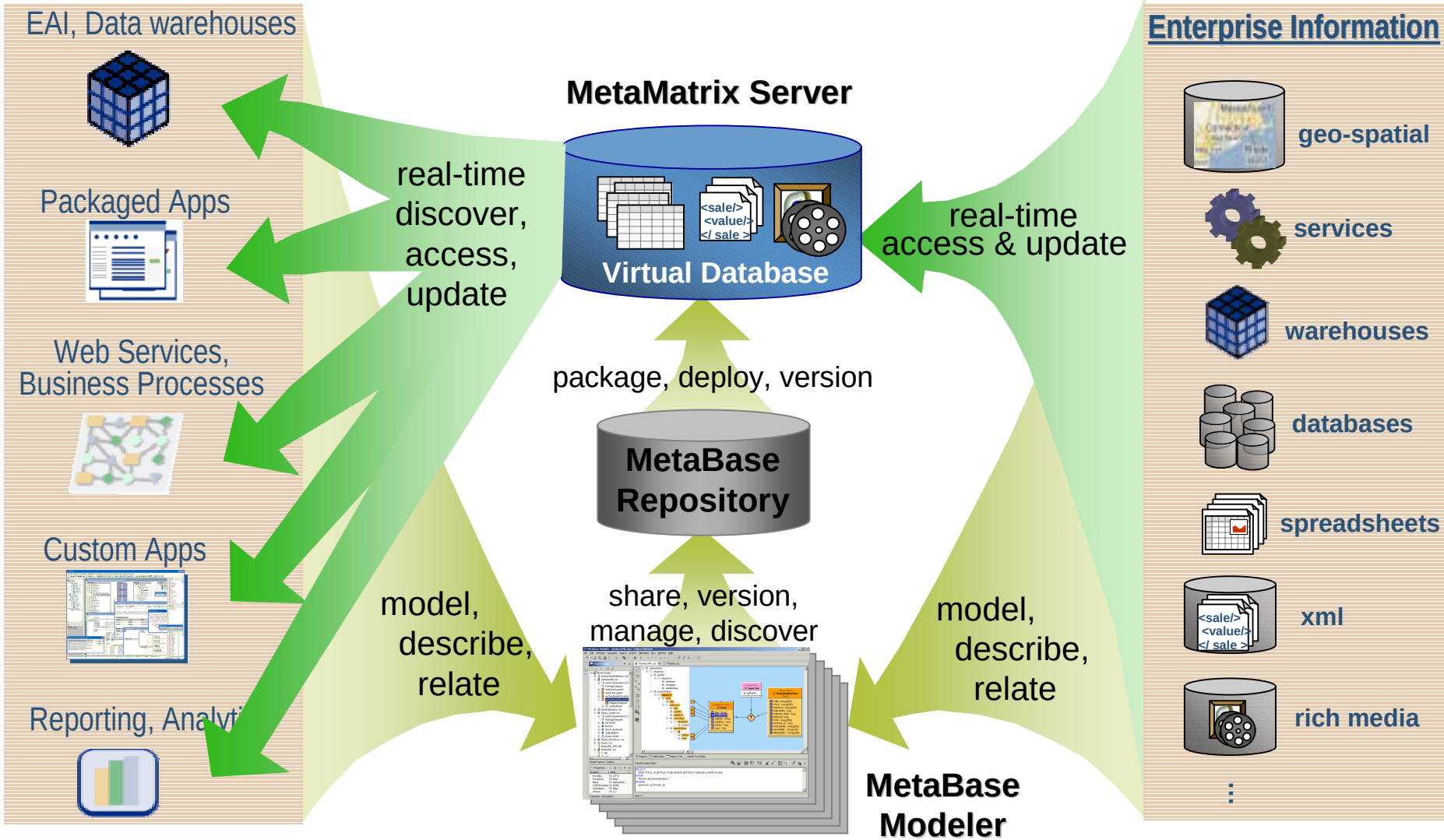
CREDIT SUISSE | FIRST BOSTON

Schroders | 2004
200 years of forward thinking

ING

MetaMatrix:Model-driven information integration

Understand, relate, harmonize, rationalize, and use



What the products do

- An integrated, Eclipse-based model development environment (MDE) to model & relate multiple kinds of enterprise information
  - Use multiple domain-specific modeling languages: Relational, UML, Web Services, XML, XML Schema, Taxonomies, Ontologies, etc.
  - Enterprise-class metadata management system with seamless access via JDBC, ODBC and SOAP
- Model-driven and real-time integration of enterprise information
  - Use the models to integrate and provide real-time access (including transactional updates) of disparate enterprise information
  - Supports nearly any kind of information source and stores
  - Highly efficient and optimized for high-volume, high availability
- Enable model-driving other applications and systems
  - Through rich metadata management, packaging and access mechanisms

Standards & Open Source Projects

- OMG standards
  - Meta Object Facility (MOF) with standard & custom metamodels
  - XMI, UML2, CWM, etc.
- W3C standards
  - XML, XML Schema, SOAP, UDDI, HTTP, XPath, XQuery, OWL, etc.
- Supports and uses Java standards
  - JDBC, J2EE, JMS, etc.
- Supports SQL-92 (plus)
- Committed to and built on top of open source projects
  - Eclipse, EMF, Apache, Doug Lea's Concurrency, and others

More on standards and open-source tomorrow afternoon

Service Oriented Architecture

"Policies, practices, principals and frameworks that enable functionality to be provided and consumed at a granularity relevant to the service consumer"[1]

- Concepts
  - Service: component that perform a task or functionality and that provide the natural building blocks for enterprise business applications
  - Provider: the owner of a service who is focused on the service architecture and component architecture, but not the application architecture
  - Consumer: the end-user of a service who is focused on the service architecture and application architecture, but not the component architecture
- This is nothing new
  - DCE, CORBA and DCOM systems were service-oriented
- So what's changed?
  - The **form** of communication is neutral of the implementation technology for the various components and services

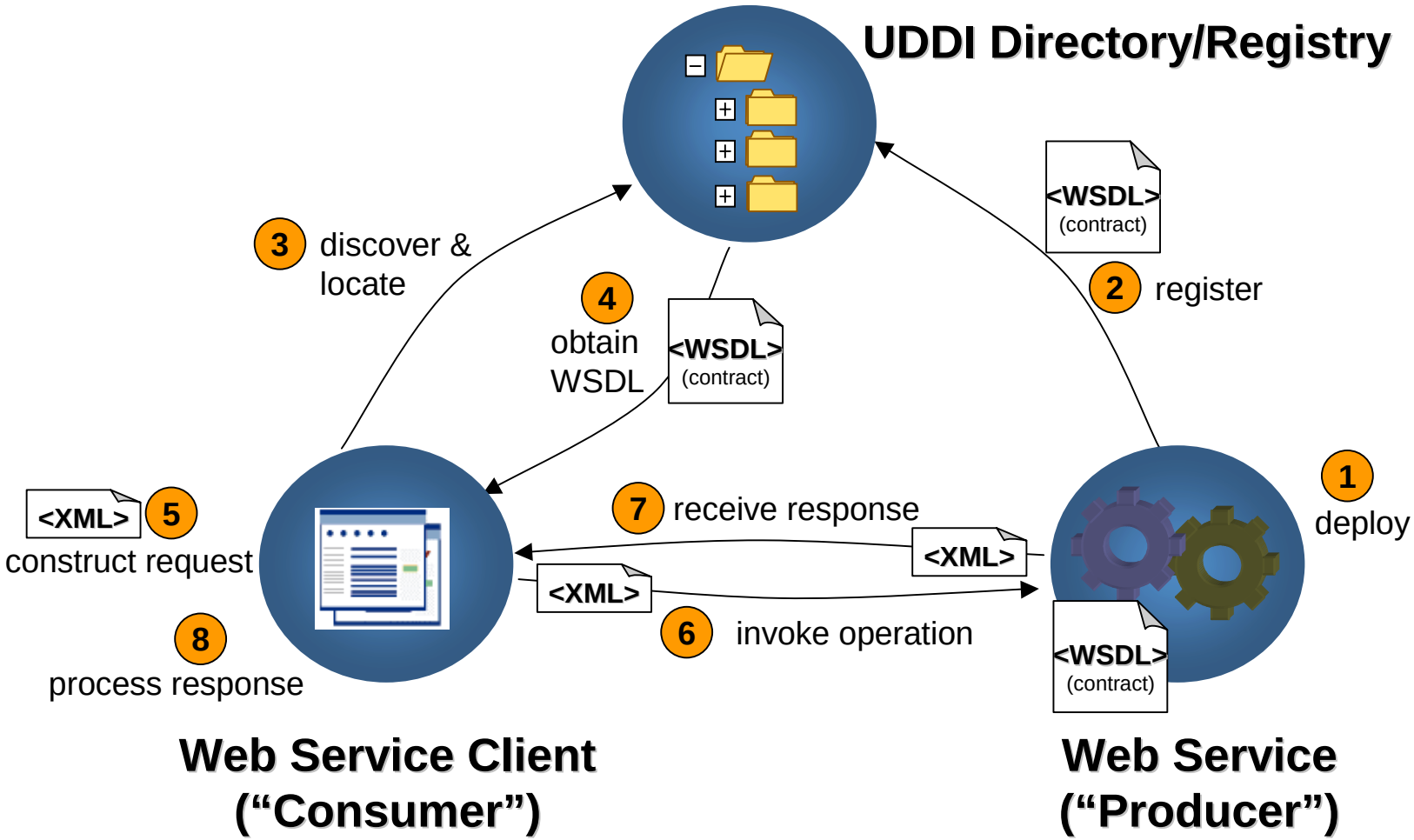
Web Services

"The set of protocols by which Services can be published, discovered and used in a technology neutral, standard form"[1]

- XML
  - The representation used for messages, descriptors, metadata, etc.
- SOAP
  - Provides the envelope for web service messages
- Web Service Description Language (WSDL)
  - Format for describing the operations, message structure, and instance (binding) information of a service
- Universal Description, Discovery and Integration (UDDI)
  - A set of standard "registry" services for publishing and discovering (e.g., searching) service descriptions and metadata
  - Searchable white, yellow and green pages
- Transport technologies: HTTP, SSL

Web Services

SOAP, WSDL and UDDI provide the building blocks



Web Services

Challenges

- Implementations may vary – either by extending the standard or by making a subset of the functionality available
- Standards may have multiple interpretations; somewhat mitigated by interoperability groups such as WS-I or SOAP Builders
- Different versions of the standard are supported
- XML isn't the "silver bullet of integration" – still need to agree upon vocabulary used in messages and interactions

and most importantly ….

- These standards do *not* (adequately) address the meaning of the data ***(semantics)***
- These standards do *not* address ***data quality***

Service Oriented Architecture

Guidelines for successful implementations

- You can't buy SOA
  - SOA is an architecture with best practices, while vendors provide tools and products that may provide some functionality required by an SOA
- You shouldn't build SOA internally
  - Simply not effective to build all of the components without taking advantage of products already on the market
- Practical Steps
  - Use a combination of external products and internal experts
  - Move an enterprise to SOA one step at a time
- Understand the benefits
  - Increases the enterprise's agility
  - Increases the visibility into operations
  - Reduces the complexity and cost of integrating systems; most effective when multiple fine-grained calls are replaced by coarser-grained and loosely couple services that are more flexible, more reusable, and *return all the information needed*

Service Oriented Architecture

Goals and objectives

- Web Service Principals
  - Technology neutral – independent of the consumer and provider platforms
  - Standardized – protocols that are based upon standards
  - Consumable – enable discovery and use
- SOA Principals
  - Reuse – using services, not copying code or implementation
  - Abstraction – all aspects of the service implementation are hidden
  - Published – the service is made available and it's interface is explicitly and precisely described
  - Formal – the provider and consumer understand and accept the rules of interaction
  - Relevance – functionality is presented at a granularity that is appropriate and useful for the consumer

Service Oriented Architecture

Achieving success

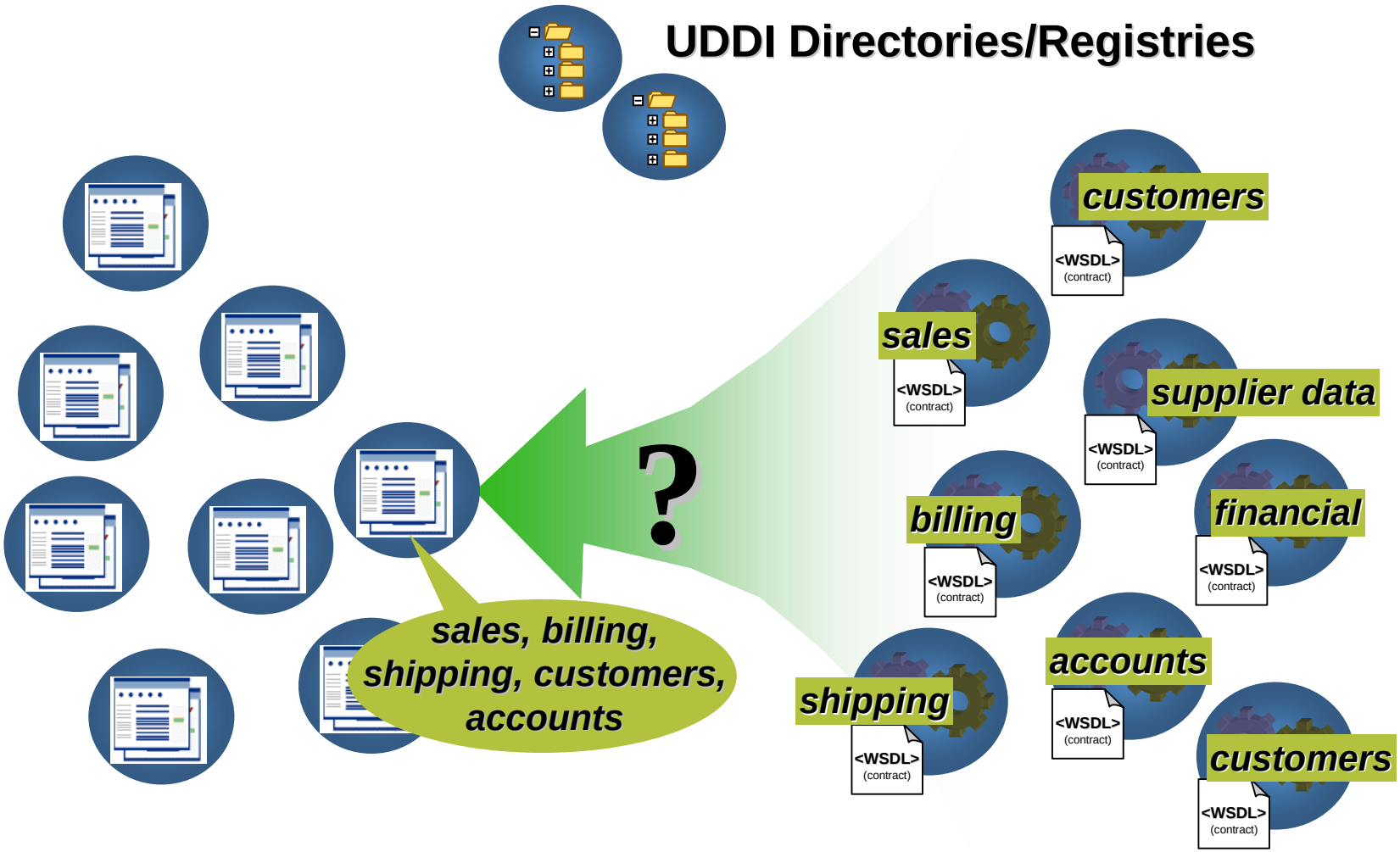
- Provisioning the services
  - Requires collaboration between consumers and provider
  - The needs of the consumers must be understood
  - The capabilities of the providers must be understood
- Services must have business-level functionality
  - Business-level abstractions and concepts can be understood
  - Facilitates reuse
  - Can be easily mapped to the business processes

Service Oriented Architecture

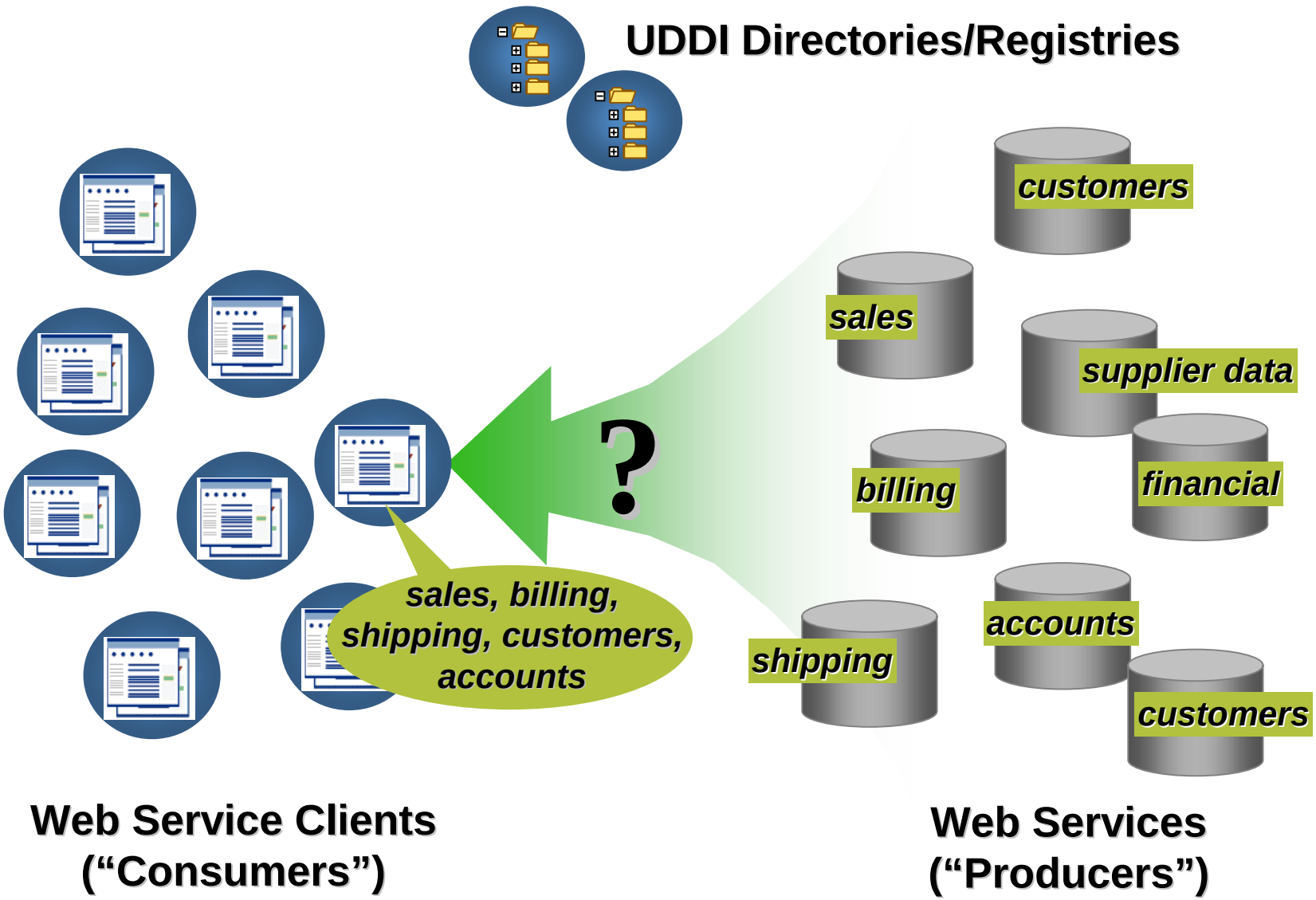
Using corporate data

- Information exists, is managed, and is persisted in numerous places
  - Database systems – relational, legacy, proprietary, warehouses, marts
  - Files – spreadsheets, character-separate value files, XML, etc.
  - Services and Applications – mainframes, web services, components
- Roadblocks to accessing information
  - Dislocated – the access technology varies
  - Security – often source-specific protocols and policies
  - Disparate – semantics, syntax and structure of information varies wildly
  - Siloed – information is not reused or shared, and often there is no complete knowledge of the information that is available within an enterprise
  - Data Quality – not all data is good data
  - *These roadblocks will always exist!*
- The information is critically valuable
  - In many ways, information is the currency of the enterprise
  - Therefore, a coherent data architecture is required

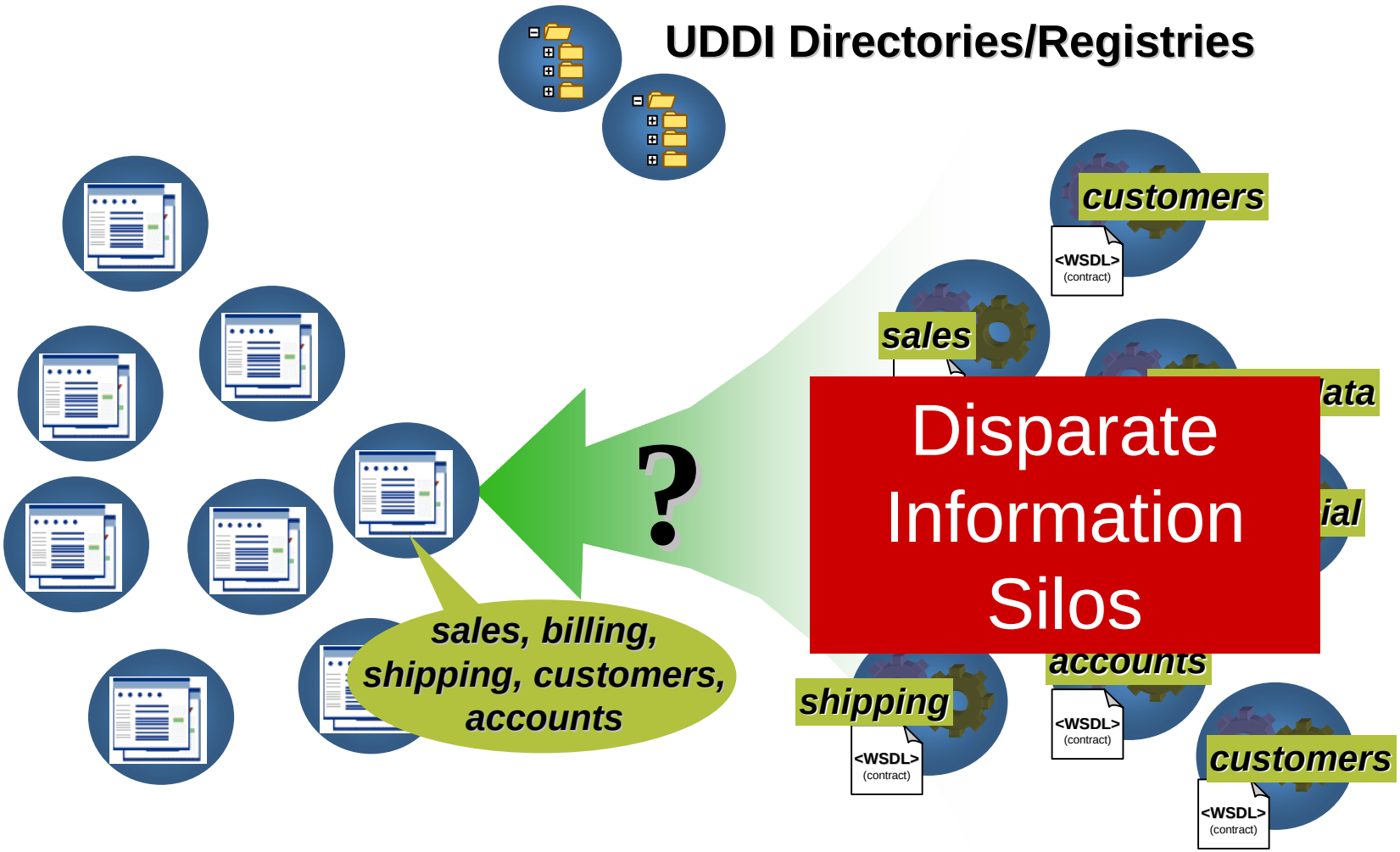
Enterprise Information



Enterprise Information



Enterprise Information



Making Corporate Data Relevant

Provide consumers the information they need

- People will never agree on a single representation of the data
  - This is true for storage and for exchanging information
  - Even if the structure is the same, the semantics are probably not
- Each consumer (or domain of consumers) will likely require the information structured to fit their needs
  - Volume and breadth
  - Ad hoc
- Challenging when integrating information from multiple sources (XML files, web services, relational databases, flat files, etc.)
  - Schema (Syntactic) mismatch
  - Semantic mismatch
  - Data mismatch

Challenges of Integrating Information

Schema (syntactic) mismatch

XML Document "A"

```
<cellPhone>123-555-1212</cellPhone>
```

XML Document "B"

```
<phone>(123) 555-1212 Ext 1000</phone>
```

XML Document "C"

```
<phone number="1235551212" type="cell" /phone>
```

XML Document "D"

```
<phone type="mobile">1.123.555.1212</phone>
```

- Differences in the names and structure of elements and attributes
- Differences in the datatypes (type, pattern, ranges, etc.)
- Differences in the domain of values for elements and attributes
- Often arises when getting information from 2+ sources

Challenges of Integrating Information

Semantic mismatch

XML Document "A"

```
<customer>
    <phone>123-555-1212</phone>
    <mobilePhone>123-555-1213</mobilePhone>
    <faxPhone xsi:nil/>
</customer>
```

XML Document "B"

```
<customer>
    <homePhone>123-555-1212</homePhone>
    <workPhone>123-555-1214</workPhone>
    <faxPhone>123-555-1215</faxPhone>
</customer>
```

- Does this represent the same information?
- Does this represent the same type of information
- Does this represent information about the same customer?
- Often arises when getting information from 2+ different sources

Challenges of Integrating Information

Data mismatch

XML Document "A"

```
<customer>
    <name>Jon A. Smith</name>
    <streetAddress>12 N. Main St.</streetAddress>
    <city>Springfield</city>
</customer>
```

XML Document "B"

```
<customer>
    <name>John Smith</name>
    <streetAddress>12 Main Street</streetAddress>
    <city>Springfield</city>
</customer>
```

- Supposed to represent semantically the same type of information
- Is this information about the same person?
- Often indicative of a data quality problem (1 or more sources)

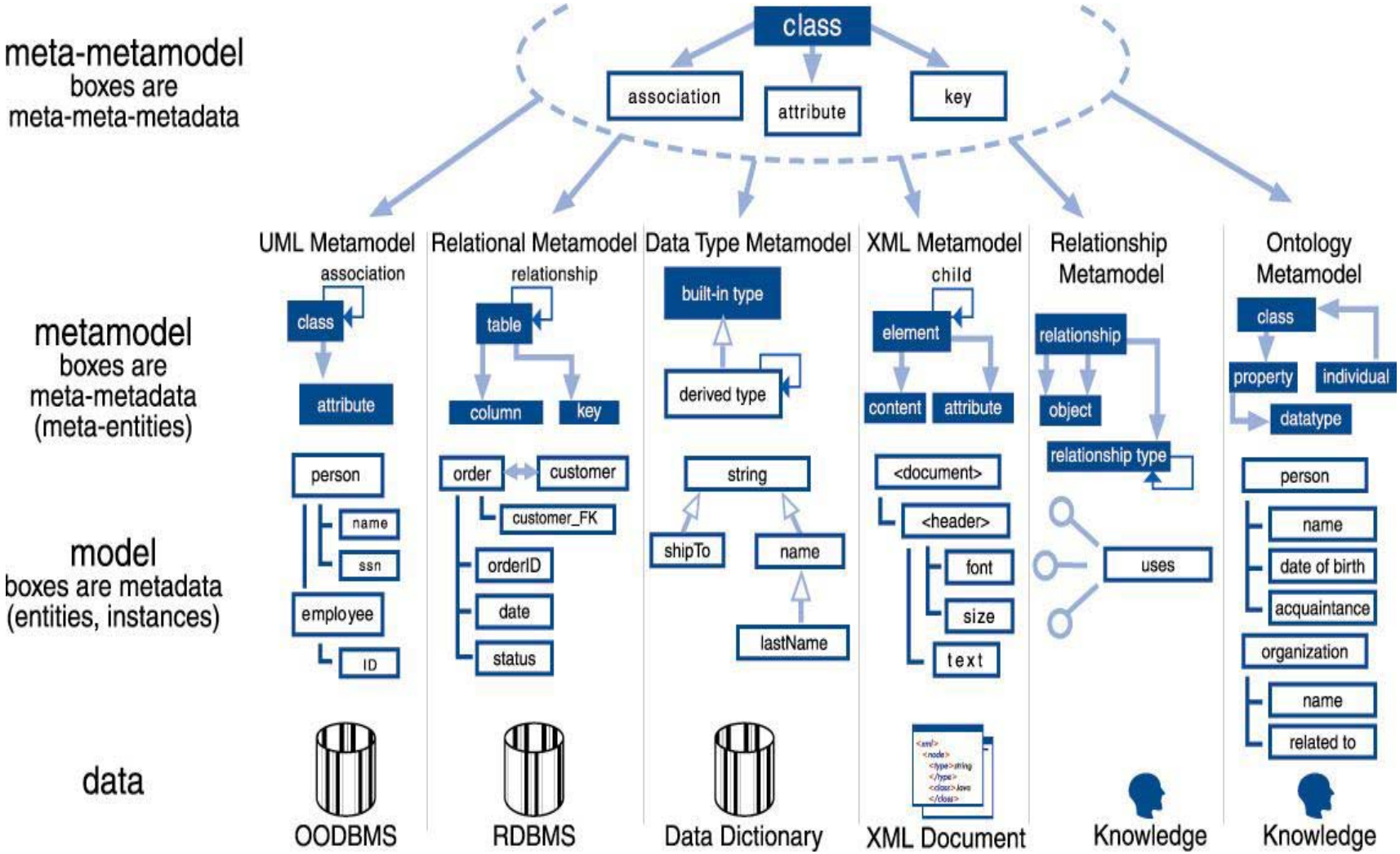
Using a Model-Based Approach

To model disparate information …

- Information takes many forms
  - Relational tables, views, procedures
  - XML elements, attributes, namespaces
  - Object-oriented classes, properties, operations, associations
  - Simple and complex datatypes
  - Services, components, messages
- Domain-specific languages are best
  - Use the modeling language best suited for the system being described
  - Leverage native artifact structures (e.g., DDL, WSDL, XSD, OWL, etc.)
  - One modeling environment to view, discover, relate, map, and report across all of the different types of models

Using a Model-Based Approach

With domain-specific languages (metamodels)



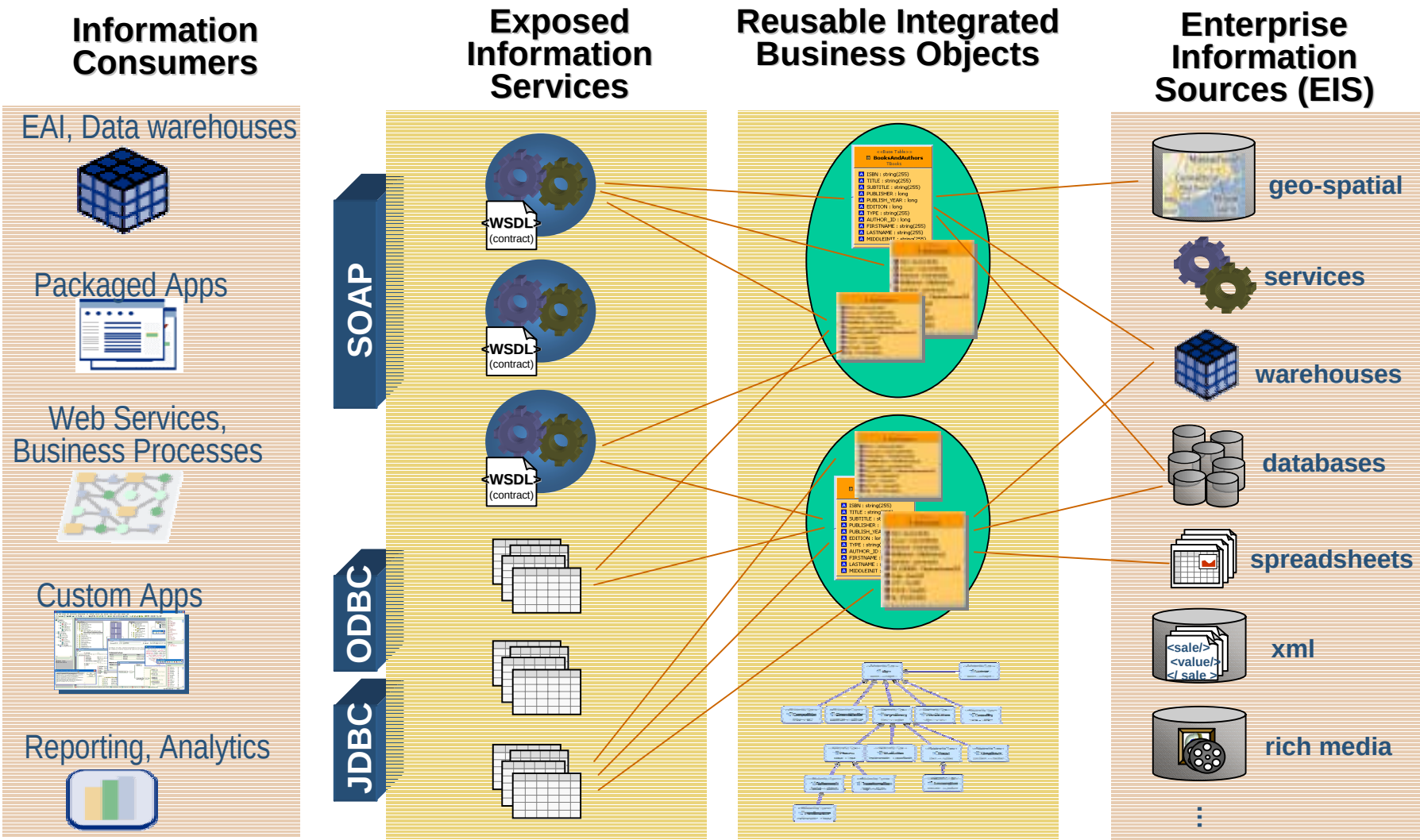
Using a Model-Based Approach

To integrate disparate information

- Understand what is available
  - The existing sources
- Relate to enterprise business concepts
  - Identify how the various information structures related to common or business-domain concepts
  - Reusable business objects, mapped to underlying sources
- Expose consumer-specific views of common business objects
  - The expectations, structures and form of the desired information
- Use these models! Not just documentation

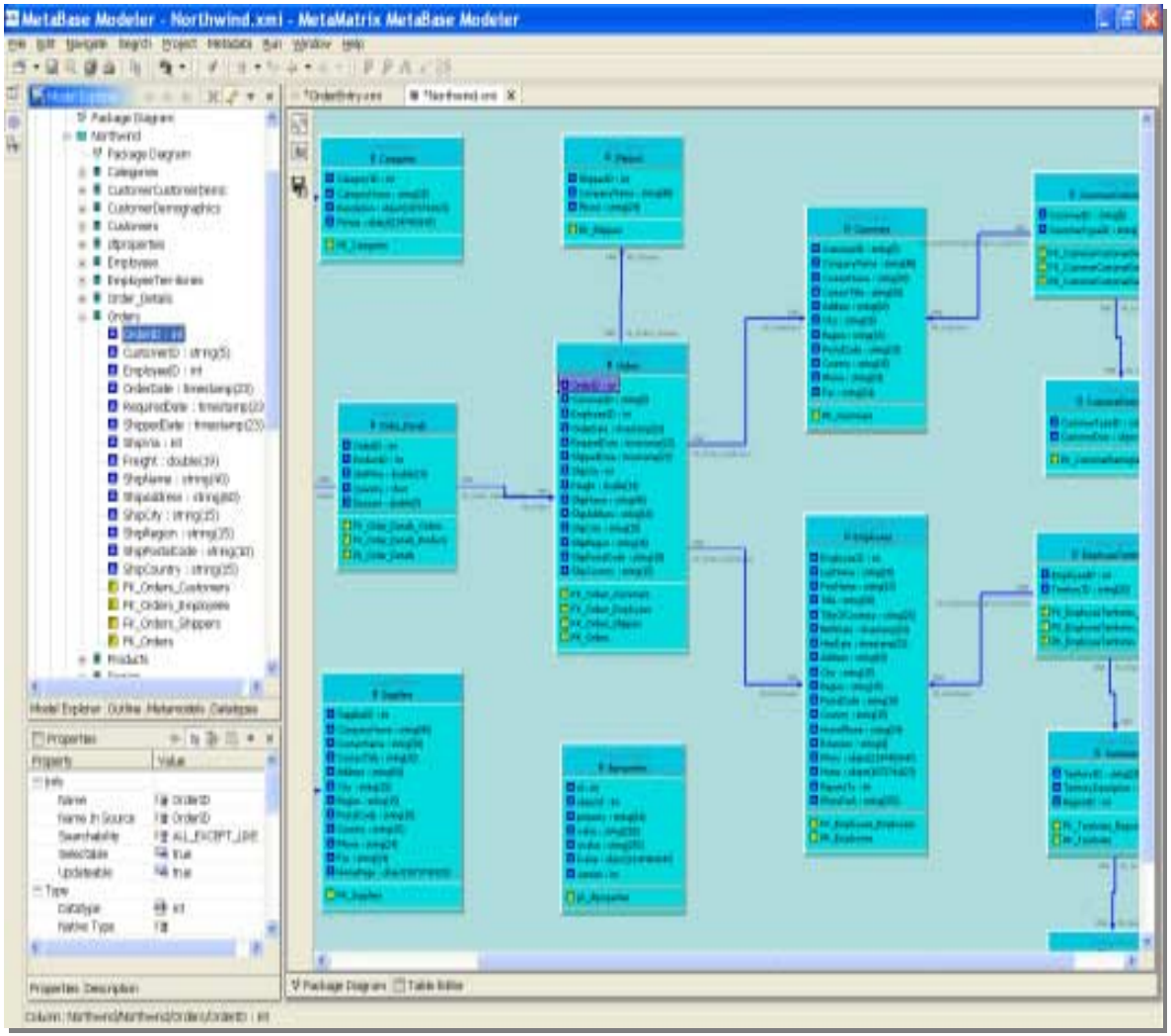
Using a Model-Based Approach

Modeling information services as layers



Using a Model-Based Approach

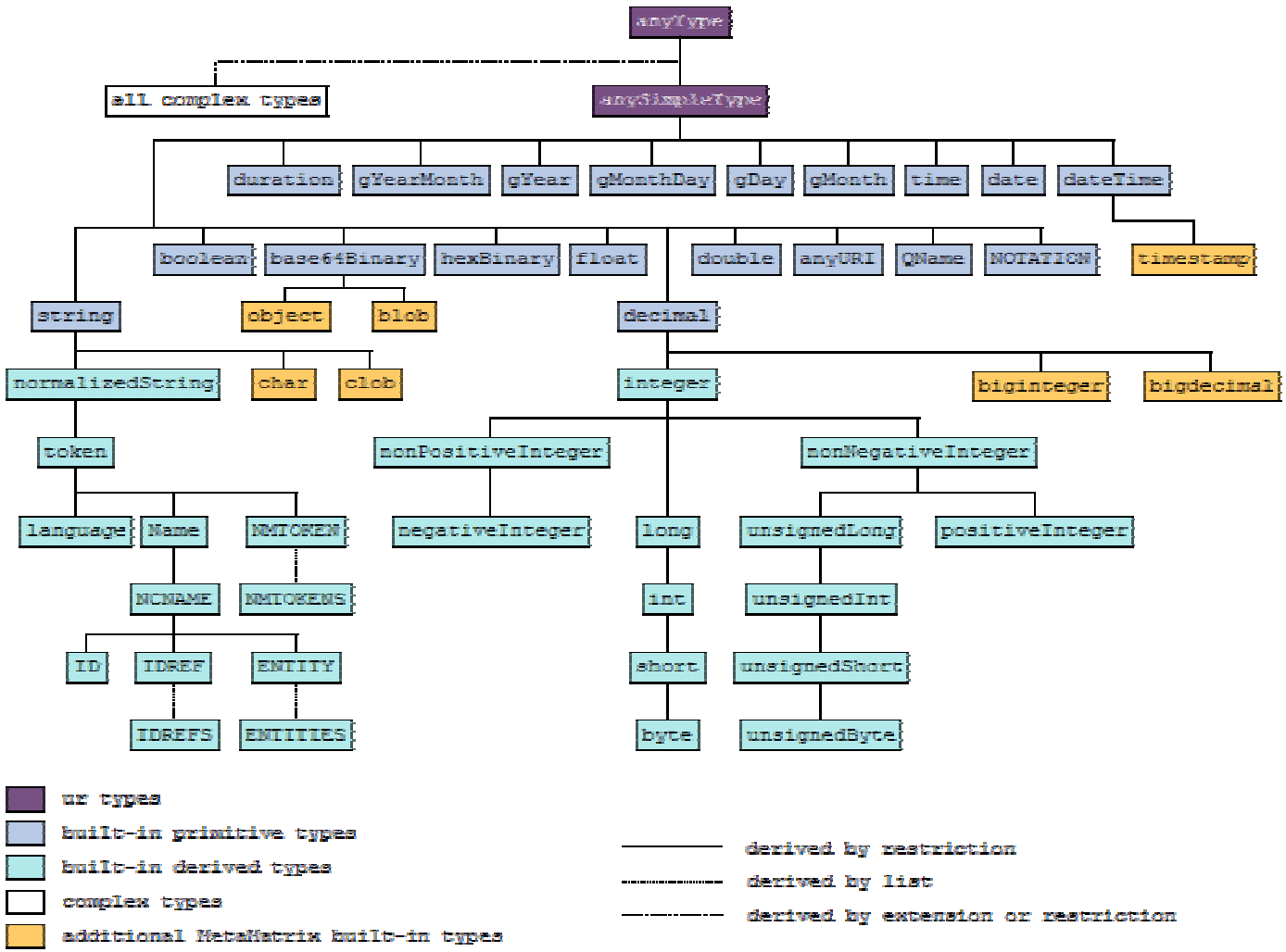
Modeling the information sources



- Model the disparate information sources
  - Relational DBs
  - Content Management Systems
  - Files
  - Services
  - Applications
- Uses and retains domain-specific modeling languages
  - Relational models have “Tables” and “Columns”
  - UML models have “Packages”, “Classes”, and “Attributes”

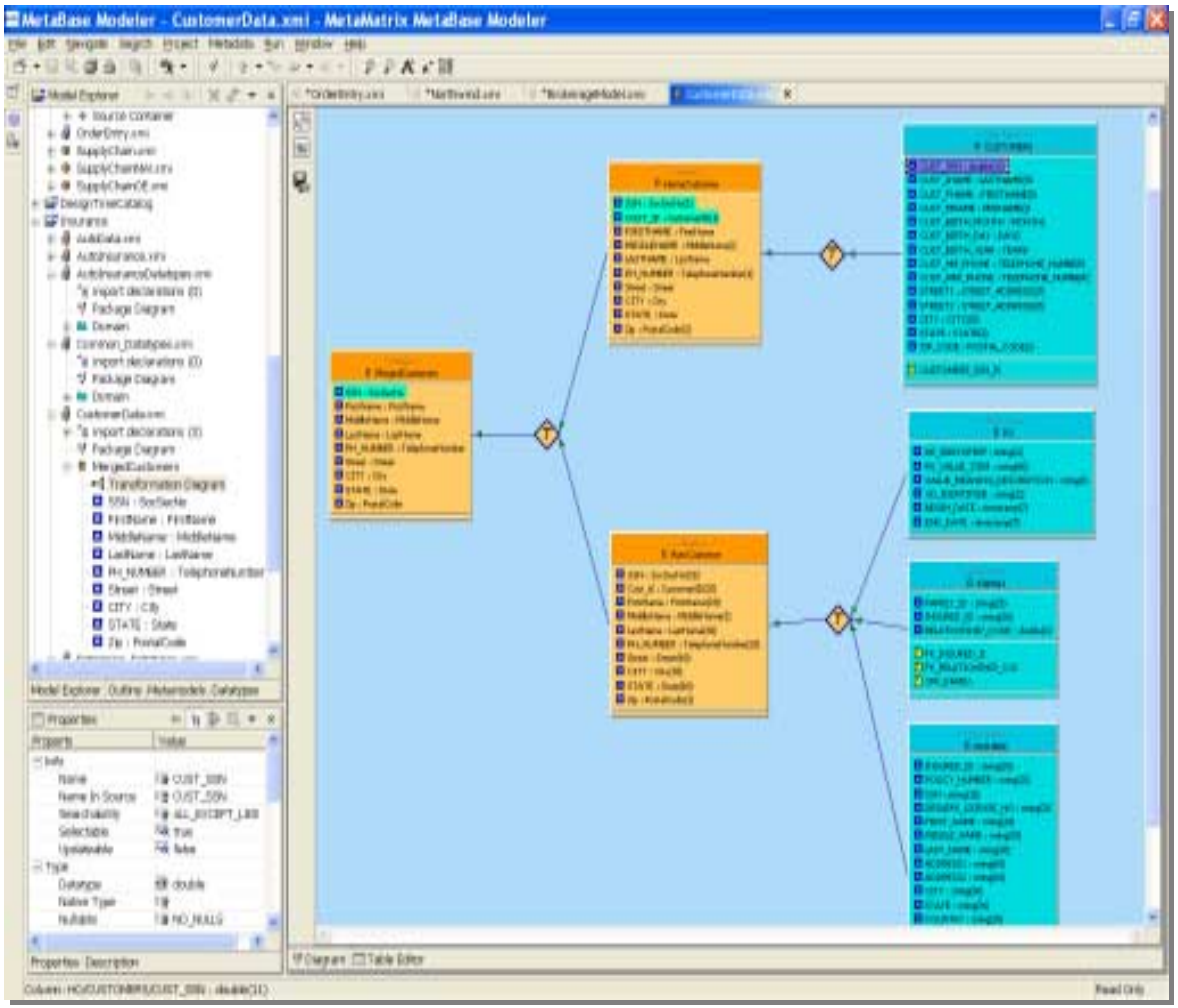
Using a Model-Based Approach

Modeling enterprise datatypes



Using a Model-Based Approach

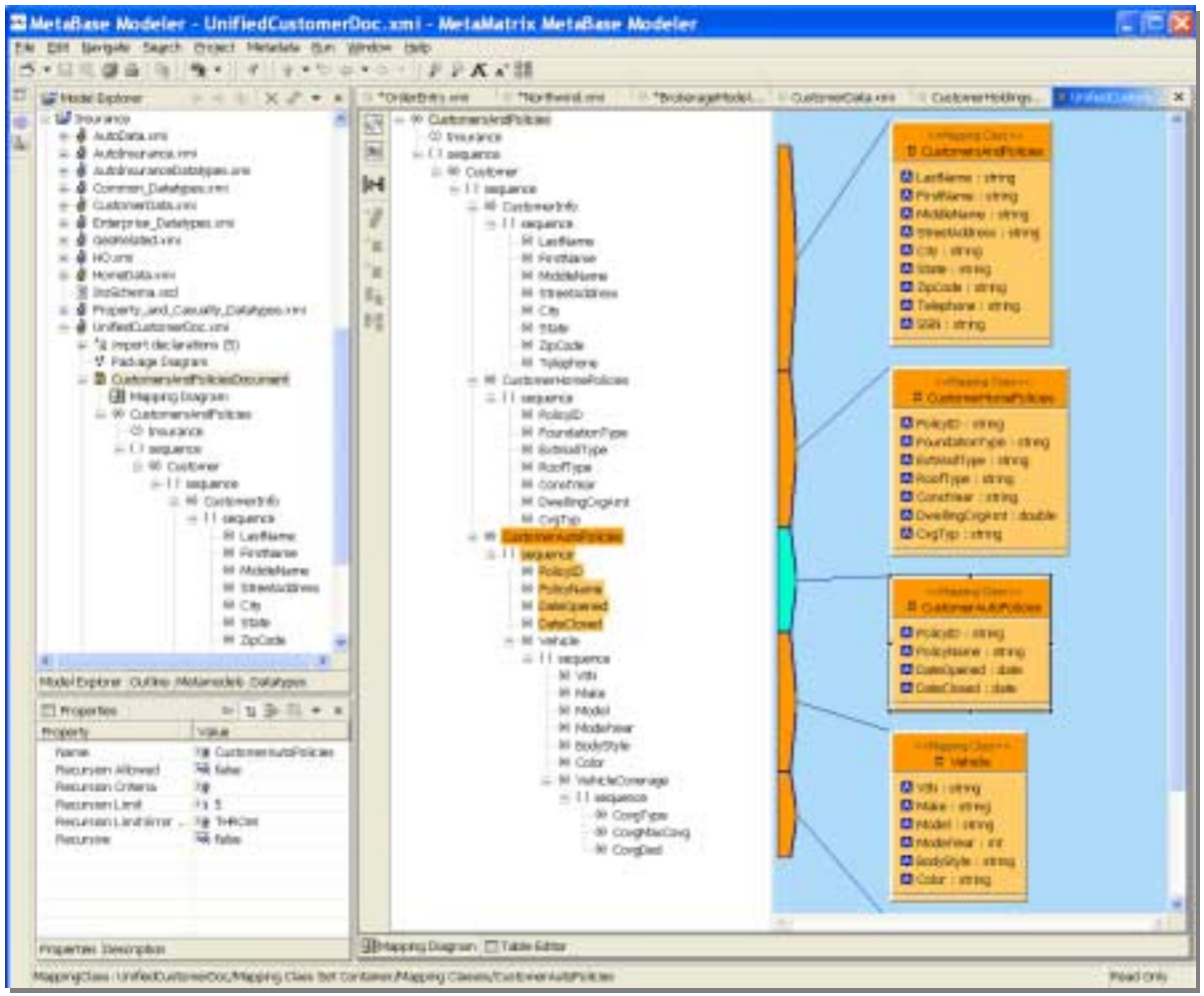
Modeling the reusable business objects



- Define reusable business objects
- Uses enterprise datatypes
- Map to other business objects or to EISs and integrate
  - Join
  - Union
  - Functions
  - Procedures
  - Criteria
- Perform schema and semantic matching

Using a Model-Based Approach

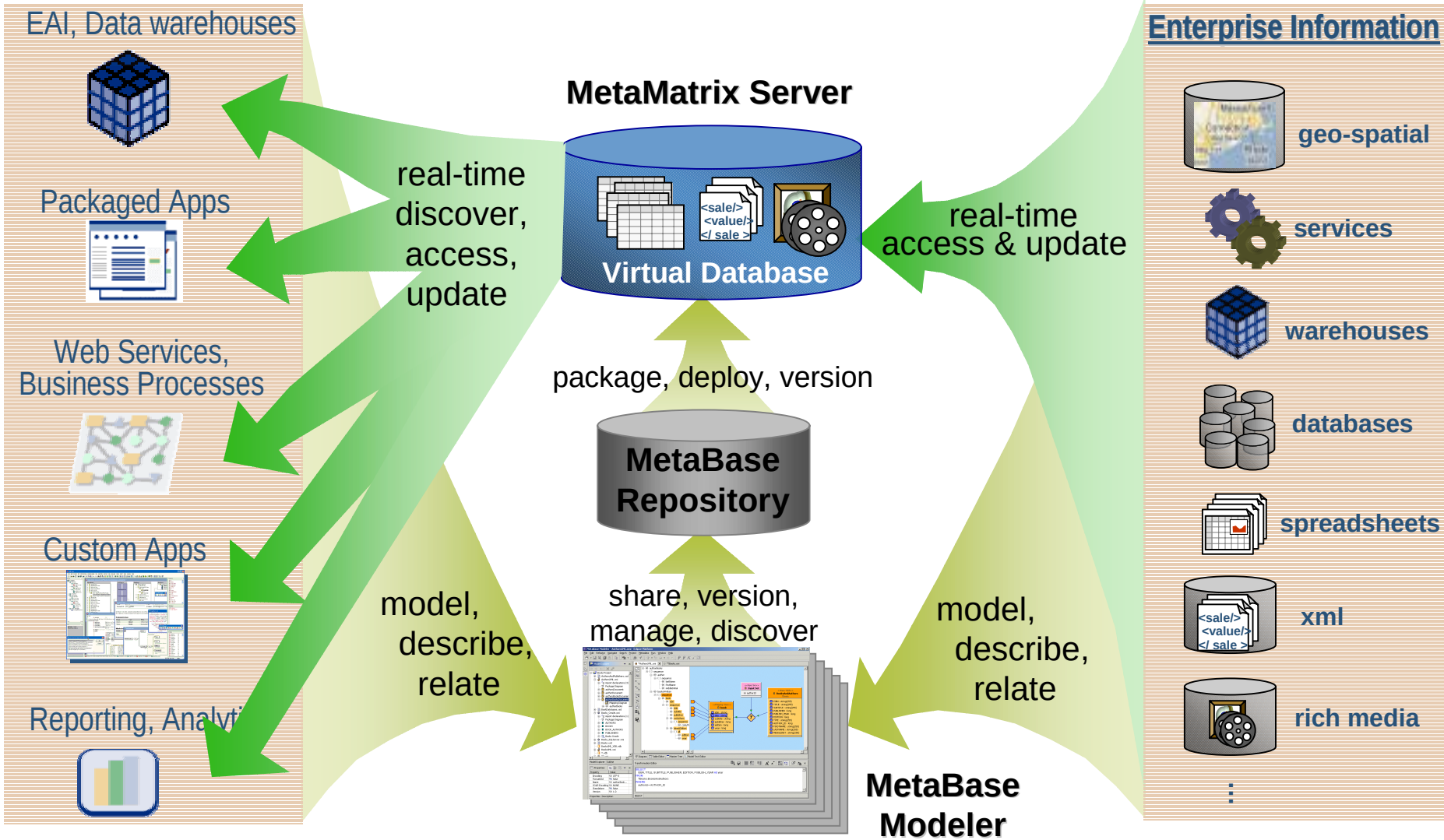
Modeling the exposed information services



- Model Web Service operations
- Model XML messages (based upon XML Schema)
- Map reusable business objects to any XML schema

Using a Model-Based Approach

Deploy models to enable real-time integration



Using a Model-Based Approach

Layered information services

- Enterprise Information Sources (EISs)
  - Use existing sources (legacy applications, databases, services, etc.)
  - Use data warehouses, data marts when EISs can't be hit in real-time
- Reusable Integrated Business Objects
  - Build subsets that share common data dictionary and semantic intent for each business (or business unit)
  - Represent the logical concepts of the business (or business unit)
  - Maps to underlying EIS systems
  - Reusable components
- Exposed Information Services
  - Designed to provide information in format needed each consumer
  - May be reusable, but probably tailored for different consumers
  - Exposed

Using a Model-Based Approach

Abstracts intention from execution

- Models declare what the intent is and what should happen
  - The “how” is determine later
- Interpretation, optimization and execution is left to the runtime engine
  - Access – providing multiple APIs
  - Push-down – do as much work as close to the data as possible
  - Orchestration – coordinating the interaction with multiple sources, including transactional updates
  - Caching – enable reuse of already integrated information
  - Safety – guard against unexpected and unacceptable uses
  - Scalability – handling massive volumes of information and large numbers of concurrent consumer requests
  - Performance – choosing appropriately and using the optimized algorithms for integrating information
  - Security – policies applied at runtime applied heterogeneously
  - Traceability – expose and ensure compliance
  - Improvements – engines can be improved without changes in the models

Loosely-Coupled Services

How can we move toward loosely-coupled systems that use dynamic discovery & invocation?

- Machines must be able to understand, compare and map the syntax and the semantics
  - Precise and accurate modeling of syntax is possible today
  - Precise and accurate modeling of the semantics is currently insufficient
  - Correctly comparing and using the semantics is in research stages (at best)

- Ontologies
  - Have potential to capture the semantic intent and meaning
  - Technologies still maturing
  - Currently labor intensive to build the ontologies, relate them to each other, and map them to the enterprise information
  - Will help *understand* the information in each silo, but will not help *access and integrate* the information in the silos

Model-Driving Information Integration & SOA

Summary

- Successful information services
  - Provide information at the granularity appropriate for the information consumer
  - Abstract the information consumer from the implementation
  - Expose valuable corporate data

- Model-driving the access to integrated information
  - Models provide understanding of how information is used in enterprise, facilitating discovery, reporting, harmonization, rationalization
  - Views of required information are mapped to available information through common business objects and concepts
  - Models simply deployed to enterprise-class integration engine
  - Intelligent information is the fusion of corporate data with corporate metadata, allowing multiple classes of applications to adapt to continuously-changing information landscape

References

1. “Understanding Service-Oriented Architecture” by D. Sprott and L. Wilkes, CBDI Forum, January 2004
http://msdn.microsoft.com/architecture/soa/default.aspx?pull=/library/en-us/dnmaj/html/aj1soa.asp#aj1soa_topic2

Abstract

One challenge of implementing a Service Oriented Architecture (SOA) is to have services that provide unified, integrated and real-time access to an enterprise's existing corporate data, even when the data are stored in many legacy, relational and other data sources. Model-driven Enterprise Information Integration (EII) technologies make it possible to provide such services by modeling the available information, the desired information, and the relationships between them. Thus, model-driven EII is a critical part of any SOA where the goals are decoupling (or loose coupling), abstraction, integration, flexibility, speed and agility.

About the Presenter

Randall Hauch is the Chief Architect at MetaMatrix, the leading provider of Enterprise Information Integration (EII) middleware to access and integrate in real time disparate information sources. He has been working with Java technologies for the past 8 years, and with application development for over 10 years. After receiving a BS and MS in Aerospace Engineering, he developed engineering models and engineering applications for aerospace systems. He has been at MetaMatrix for 5 years, and has helped define and build the company's modeling infrastructure and metadata management capabilities, including the Eclipse-based modeling tool. Randall is also the MetaMatrix representative to the Object Management Group (OMG), where he has participated on various task forces, including MOF, XMI, CWM, MOF2 and UML2 Infrastructure.