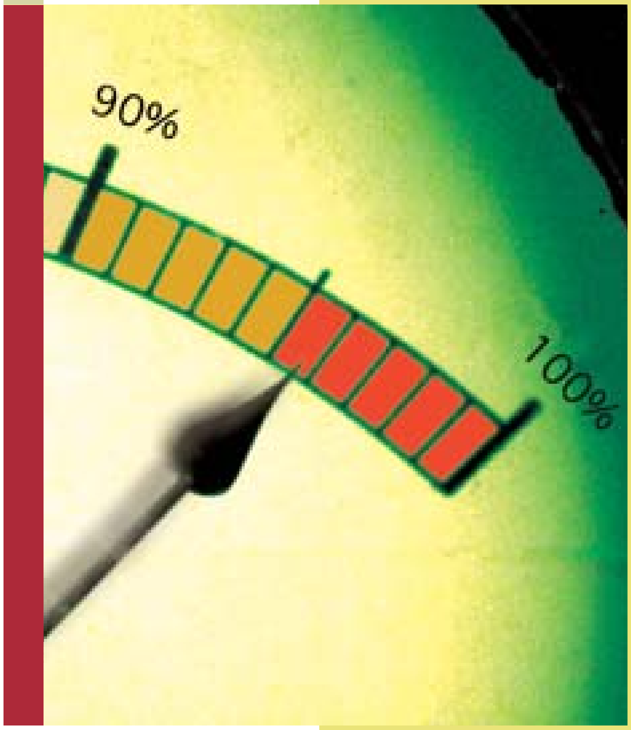
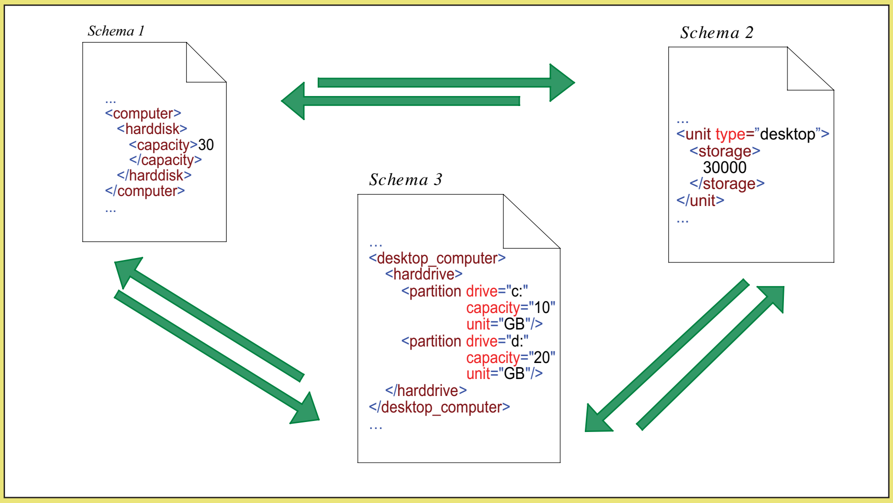
Active Information Models for Data Transformation

By Joshua Fox, Ph.D.

Regardless of industry sector or organizational size, enterprises face critical problems in data management and operational efficiency as a result of multiple, overlapping and distinct schemas in each domain. Corporate mergers often result in parallel applications that handle similar domains. Often, departments define their transmission schemas separately because of a lack of coordination or the use of different applications.



Maintaining the variety of applications and integrating between them is a challenge. It's difficult to track the schemas and business entities each represents. Currently, to integrate the applications that use these schemas, transformation logic must be custom-developed at great cost, reflecting semantic and syntactic differences between schemas. Any effort to manage these metadata and data-transformation assets manually is doomed to failure since changing requirements quickly invalidate analyses and interschema transformations.



```
...
<computer>
  <harddisk>
    <capacity>30
    </capacity>
  </harddisk>
</computer>
...
```

```
...
<unit type="desktop">
  <storage>
    30000
  </storage>
</unit>
...
```

```
...
<desktop_computer>
  <harddrive>
    <partition drive="c:"
               capacity="10"
               unit="GB"/>
    <partition drive="d:"
               capacity="20"
               unit="GB"/>
  </harddrive>
</desktop_computer>
...
```

Figure 1 — Transformed Documents (Sample Fragments)

The Problem

Enterprise data resources can take many different forms: Relational Database (RDB) tables, XML documents, Electronic Data Interchange (EDI) messages, COBOL records and others. Independent Software Vendor (ISV) applications — such as Enterprise Resource Planning (ERP), Customer Relationship Management (CRM) software, or home-brew software — define their own input and output schemas. Often, the different schemas hold similar information, though the structure of the information may differ.

Enterprise Application Integration (EAI) systems must overcome the problem of heterogeneous schemas. To do this, EAI implementers must fully understand the semantics of each schema, since the schemas often represent the decisions of a single application developer working from specifications of a single step in a process. Ultimately, new schemas multiply as requirements change, and the investment in analyzing requirements, as captured in older schemas, is lost.

Today, overcoming this heterogeneity requires significant time, with little return. Developers must create transformation code manually, whether by coding in textual languages, such as eXtensible Stylesheet Language Transformation (XSLT), or by using graphical transformation design tools. While this is tenable with a few schemas, a larger number results in a skyrocketing number of such transformations. The number of transformations goes up as the square of the number of schemas.

Even if applications are manually integrated at great expense, the solution isn't maintainable. When schemas change, as they often do in a quickly changing environment, existing code for transforming data from one schema to several others becomes unusable. In the worst-case scenario, when a single application's output or input schema changes, all code for integrating that application must be rewritten from scratch, since XSLT code and Structured Query Language (SQL) queries don't lend themselves to reuse. Since the semantics of the original schema frequently aren't captured formally, re-developing transformation logic means re-analyzing schema semantics. Finding the appropriate domain expert and dedicating the time required for this additional development cycle are two of the obvious drawbacks.

To illustrate with a simple example (see Figure 1), desktop computers with their associated storage capacities may be represented with a variety of XML structures. Each schema conveys fundamentally the same information (here, a desktop computer with 30 gigabytes of hard-disk capacity) but the tag names and structure differ completely. Each of these applications must send messages to the others on the bus. To convert from one to the other, elements must be taken out of the source document and arranged in a different way in the target, with some application of rules along the way — such as converting the megabytes recorded in schema 2 to the gigabytes of schema 1. If the three applications here are to be fully integrated, six XSLT transformations will be required.

Although the example here is built with XML, the same principles would apply to other structures, such as EDI messages, or even to disparate database tables, transformed via SQL.

Today's EAI Systems

Today's hub-and-spoke messaging bus architectures have revolutionized application integration, addressing some of the problems of point-to-point heterogeneous enterprise information systems (see Figure 2). However, they still require point-to-point effort for schema integration.

The transport hub-and-spoke architecture, as represented by asynchronous messaging buses, has tremendously simplified integration efforts. No longer do integrators have to develop a messaging link for each pair of source and target applications; the source application can now send a message on the bus, while the target application simply listens to an agreed-on queue or "topic."

The format hub-and-spoke architecture deals with the wide variety of data formats found in the enterprise, such as comma-separated textual fields, application-specific binary protocols, and RDBs. A standards-based approach uses XML as a hub format. EAI technologies can take data from most common formats and convert it to the hub format. Nonetheless, this solution doesn't solve the problem of disparate schemas. Existing tools can convert all messages to XML, but the schemas may differ. Similar information may be carried in different elements or attributes, as in our example (Figure 1).

EAI vendors include message broker components (also known as "integrators") that can transform data from one schema to another, once transformation logic is deployed. These message brokers sit as a peer application on the bus. Each source application sends messages to a messaging-bus address for the message broker. The message broker receives the message and, depending on

Integration Layer	Hub
Transport	EAI Bus
Format	XML
Schema	Central Information Model

Figure 2 — Hub-and-Spoke on Architectural Layers

workflow rules, transforms the message into a schema suitable for a target application, then re-sends the output to the target. These message-broker transformation engines are an essential part of modern EAI systems.

However, the message brokers still require the transformation logic to be developed manually. Thus, a schema hub-and-spoke architecture is still needed to avoid the design-time effort of application-to-application transformation.

Transformation Development Techniques

Enterprises today have various approaches to the problem of multiple incompatible schemas. The typical solution is to hand-code distributed services that transform one schema to another, using a general-purpose language, such as Java, or a transformation-specific language, such as XSLT or IBM's ESQL.

Another approach is to develop transformation logic using graphical development tools, often bundled with EAI message broker products. Although these tools simplify the development of any given transformation, the user must still manually create transformation logic for a given source schema and target schema. This includes analyzing the schema requirements, designing the transformations and building them. The effort required still goes up rapidly as the number of schemas increases, and it's still impossible to maintain the transformation logic, given application requirements. This is parallel to the well-known problem with application-to-application transport connections that was solved by messaging buses: an integration effort that increases rapidly as the enterprise's needs inevitably grow.

The Unifying Information Model

The semantic hub is at the heart of a new enterprise data integration architecture in which a central information model dynamically generates data transformations between and among schemas, obviating the need to develop schema-to-schema transformation code for any pair of schemas.

This new generation of software creates a new hub-and-spoke system for schemas, naturally extending the benefits of the hub-and-spoke system for transport and for format found in today's EAI systems. This semantic hub automatically generates all the needed transformation code.

The information model is central in that it represents a neutral semantic view of the enterprise.

The semantic hub is centered on a rich, central, active information model. This information model is based on ontology, a modeling technique that has been under development in academia for decades. In recent years, Tim Berners-Lee, inventor of the World Wide Web, has put forward his vision of the Semantic Web, the next stage of information sharing, this time between applications. Ontology is at the core of the Semantic Web. The World Wide Web Consortium (W3C), which has created standards for the Web and for XML, is now finalizing a standard approach to ontology as part of its Web Ontology Working Group, guaranteeing widespread standardization for semantic systems. The semantic hub, as used in EAI systems, is an application of Semantic Web concepts to particular enterprise needs, which include robustness, maintainability and scalability.

Older modeling techniques have some superficial resemblance to the ontological method. Ontology resembles the Entity Relationship (ER) model in the linking between classes/entities by means of properties/relationships. It resembles Object-Oriented (OO) design in the use of inheritance as a powerful class-building principle. However, ER is primarily intended for modeling RDBs, while OO is aimed at software design. Ontology, on the other hand, models enterprise semantics, rationalizing schemas written in various languages, including XML Document Type Definitions (DTDs) or schemas, COBOL copybooks, RDB data dictionaries and others.

Rich Information Model

This information model is rich in that it describes real-world business entities, using techniques such as inheritance and interclass links to indicate specializations among entity types. It includes rules that show the relationships between entities. The model presents a coherent view of the complex web of meaning, relating entities in the enterprise. It uses vocabulary from the business domain rather than the cryptic identifiers found in code.

Besides code generation, the rich information model maximizes the value of schemas by serving as a central semantic repository for enterprise schemas. Centralizing and clarifying the semantics of a schema in a unified, coherent way, the information model lets managers catalog and assess their metadata assets. Legacy schemas are often difficult to understand, maintain, and modify. When schemas are rationalized to the central information model, which uses intuitive terminology rather than the cryptic identifiers of schema languages, the meaning of the schemas and their subelements is clearly indicated.

The active information model maintains full synchronization with enterprise schemas. Mapping each new schema into the model immediately relates it semantically to every other schema. When changes are needed, the user can introduce mappings of the new schemas, or the information model can actually generate new schemas.

Central Information Model

The information model is central in that it represents a neutral semantic view of the enterprise, not related to any one schema. This model is linked to metadata, such as XML schemas, Java Application Program Interfaces (APIs), COBOL copybooks, and RDB data dictionaries. This is in contrast to other types of models, such as Entity-Relational (E-R) diagrams or Unified Modeling Language (UML), which are related only to the databases or software that they served to design; E-R and UML models are often created during development and then filed away. A central information model gives coherence to the enterprise view, providing a live picture of the current state of the wide variety of enterprise data resources. Each of the resources is rationalized through mapping to the central model.

Active Information Model

The information model is active in that it can generate transformation code to transform from any one schema to any other, logically unifying all schemas. This is essential in application integration, since when two applications must communicate, the output of one differs in its structure from the expected input of the other. The semantic software can generate the transformation logic, seamlessly integrating the two applications without further intervention.

Creation of this rich, central, active model gives IT managers new power. It converts data, comprehensible only to the applications that read and write it, into information that includes a semantic description that lets all enterprise applications use it.

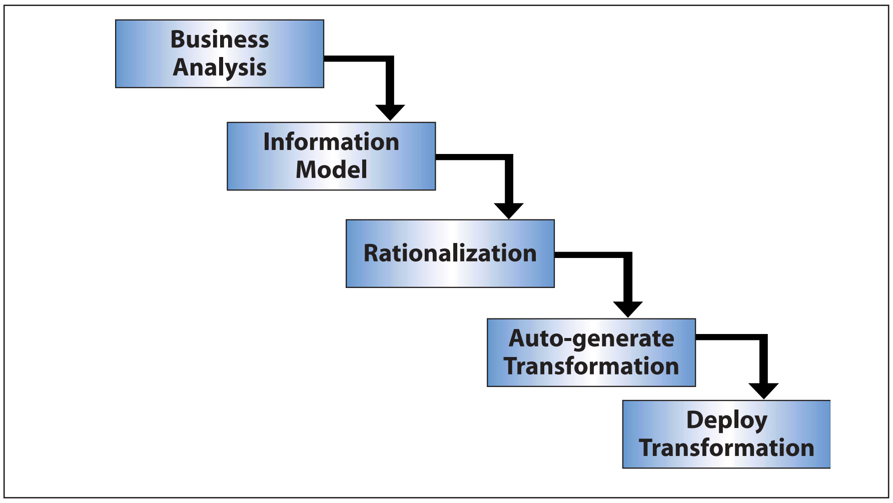


Figure 3 — Project Methodology

The information model is active in that it can generate transformation code to transform from any one schema to any other.

Data Integration Project Methodology

The development process for creating transformations through an active information model differs from the process for developing them manually. By focusing on schema rationalization to a central information model, the investment in developer time isn't lost with creating each new transformation. Rather, a single per-schema effort translates into a rich, central model that rationalizes and clarifies enterprise data. Transformations can then be produced with no additional developer effort.

This structured process, centered on the information model, minimizes human errors, facilitates application integration, and reduces lag time in responding to changing needs. As shown in Figure 3, process stages include:

- Business analysis
- Creating an information model
- Rationalization of the schemas through mapping to the model
- Automated generation of transformations
- Deployment.

Business Analysis

Developing transformations with a semantic hub begins when domain analysts examine data schemas. The analysts determine the semantic value of the schema elements — the real-world entities they represent — and the rules governing schemas. In our example, analysts determine the computer types being sold, the computer properties most relevant to the manufacturing and sales process, the units of measurement for these properties, and the rules relating to the properties. For example, computers are one of the products being manufactured, drives can be divided into partitions, and so on.

Central Information Model

The second stage is developing a central information model that encodes the semantic information in the schemas. Figure 4 shows a small part of a central information model for our computer-manufacturing example. The class, ElectronicDevice, is among those that inherit directly from the universal base class, Being. Computer and HardDisk are specializations (subclasses) of ElectronicDevice; HardDisk is also a subclass of Storage.

Examining the class Computer, each computer has a RandomAccessStorage device of type HardDisk. HardDisk has the property partitions, showing that the disk has multiple partitions. The information model also has rules; for example, a rule indicating that the capacity in

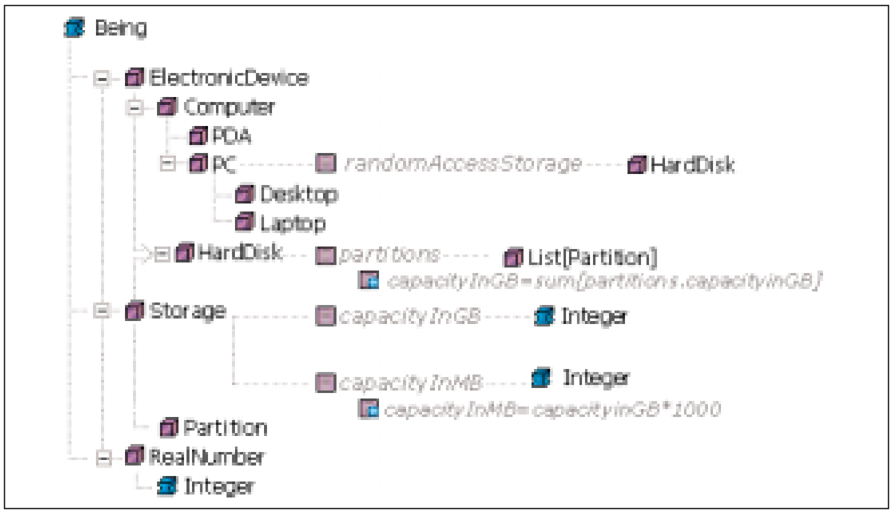


Figure 4 — (Partial) Information Model

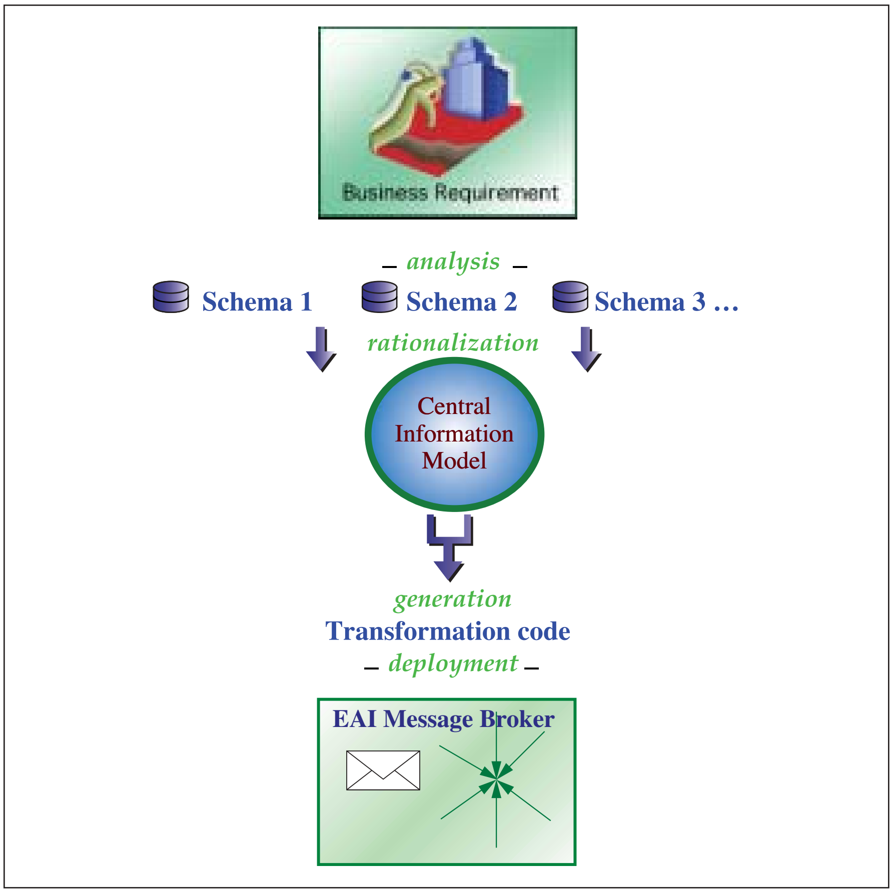


Figure 5 — Overview of Development Process

megabytes equals the capacity in gigabytes multiplied by 1,000.

Rationalizing Schemas

The third stage is rationalizing the schemas by mapping them to the ontology. Here, each complex (structured) element of the schemas is mapped to a class, while each simple (atomic) element is mapped to a class property.

The structured elements representing a desktop computer in the three schemas are mapped to class DesktopComputer; the simple elements indicating capacity in megabytes are mapped to the property, CapacityInMB.

The advantage of the semantic hub becomes clear. The schema-to-schema approach requires developing transformation code for each pair of schemas, while the semantic hub approach requires each schema to be mapped only once to the central information model.

Auto-Generating Transformation Code

The fourth stage is the automated generation of transformation code. The semantic hub software searches for shared semantics of the source and target schemas, where both source and target are mapped to the same concepts in the model. More advanced semantic hubs can transform schemas even if they're not mapped to the same concepts in the model, by applying rules encoded in the semantic hub.

The semantic hub encodes the fact that different elements are mapped to the same class: computer, unit, and desktop computer elements are all mapped to the Desktop class. When converting XML from one schema to another, the respective subdocuments are transferred to the new tag names. Likewise, the multiple elements called partition in schema 3 are transformed into a single harddisk element in schema 1. The semantic hub further generates the code needed to convert megabytes into gigabytes under the encoded rule.

The generated code can be in various languages: for XML, as used in EAI messaging systems, XSLT is used. When the goal is to directly transform between RDBs, the code is SQL that SELECTs data from the source RDB, then INSERTs the transformed data into the target.

Auto-generated code gives the additional advantage of easing maintenance. As needs change, a simple update to the active information model supports new code generation.

Deployment

The final stage is deployment. Just like manually developed schema-to-schema transformations, the auto-generated transformations can be deployed into the message broker component of an EAI system, then executed in the message broker's transformation engine (see Figure 5).

Conclusion

Enterprises are faced with too much data, too many data schemas, and not enough information. Semantics — meaning — is what turns data into information, but often, the semantics are expressed only implicitly in application code, so application data interchange requires human input.

Enterprises are now undergoing a slow evolution from managing data to managing information. For data to go beyond the silo walls of a single application to full cross-application sharing in a smooth, automatic manner, semantics must be formally expressed, then linked to the data. A rich information model makes data comprehensible to external applications designed to read it. EAI systems are providing a central hub for message transport, eliminating the need for custom transport code for each application. Up to now, however, EAI hub-and-spoke messaging has required a spaghetti of point-to-point analysis and mapping. The central information model works with EAI systems to provide coherence for data semantics, allowing data to become information.

The author acknowledges Joram Borenstein for his assistance with this article.

About the Author

Joshua Fox, Ph.D., is a software architect at Unicorn Solutions, working on the Unicorn Coherence platform for unification of information in the enterprise. Unicorn is a member of the W3C Web Ontology Working Group, which is standardizing ontology languages for the Web. His previous experience includes the design and development of large-scale distributed Internet systems. Voice: 866-286-4267, x115; e-Mail: joshua@unicorn.com; Website: www.unicorn.com.