Developing a Common Information Model

Aligning SOA Information Architecture with Data Architecture

Industry has put a lot of effort into developing advanced data architectures to govern how data is represented in their various applications. Data dictionaries have been created, Entity-Relationship diagrams drawn and management/governance processes have been established. These processes have been applied to how applications have been developed in the past and Service Oriented Architecture is *just* a new way to design applications, isn't it? So why not just use existing data architecture based processes for SOA?

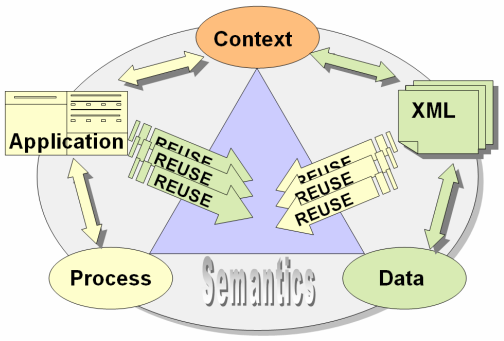
There are several significant reasons why a data architecture processes are not suitable for directly developing the XML information architecture that is needed for SOA. The primary reasons are:

- Scope. SOA architecture needs to incorporate message-based data from not only the various systems in the landscape, but also other services, often from outside the scope of the IT data architecture. These external message models include those from partners, customers, suppliers and are often based on standards such as IFX, MISMO, OAGIS and UBL.
- SOA is based on the foundation of interfaces to systems, not how systems are created. Interfaces describe the messages exchanged between systems, the actions to perform on these messages and the details on how to pass the messages. SOA messages are encoded in XML. SOA message metadata must be fully capable of describing and understanding XML.
- XML and relational databases fundamentally view data differently. XML messages combine data and context for processing in many applications. Applications combine processes and context to be applied to multiple data sets. Databases are relational, XML is hierarchical. Databases have many views on data whereas XML message is a single view of data.

While the SOA and data-driven applications have different requirements, there is a lot to be gained if the XML information architecture is aligned with the data architecture. Because of the differences between messages and database tables, the alignment can not be complete, but with appropriate design the information architecture can share and leverage many aspects with the data architecture.

The alignment should include:

- Fundamental datatypes – the technical specification of data elements should match database requirements
- Dictionary terms and definitions – a *customer, product, or part number* should mean the same in either architecture. They may be structured differently and even contain different data, but their business definition should be the same.
- Process Context – applications apply processes to XML messages to create business results. These processes should be used to guide development of message architectures.

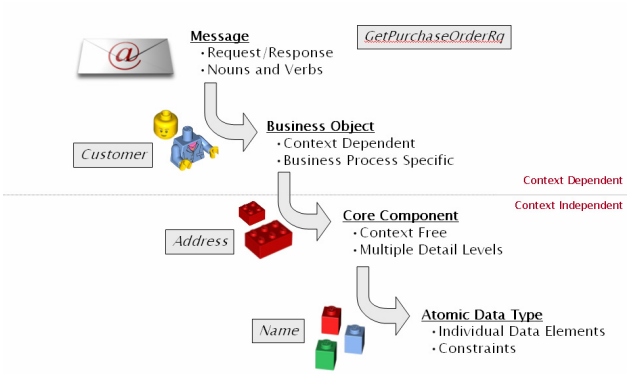


Done correctly, aligned data and XML information architectures can achieve high levels of reuse—reuse of the XML messages in various applications and reuse of applications across multiple data sets. This level reuse eludes most SOA programs because they are not fundamentally aligned with the way that applications view data and context.

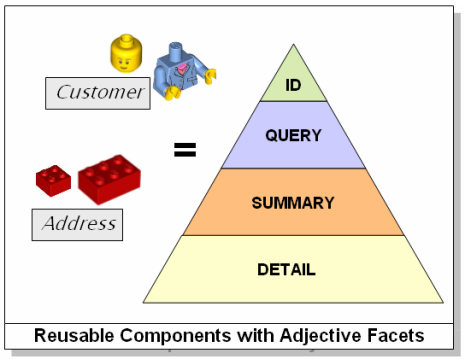
How to align XML Information with a Data Architecture

Definitions, datatypes and context are keys to alignment of XML information with data architecture. If the information architecture can separate the context dependent from the context independent elements, then the context independent elements can be tightly aligned with their data architecture counterparts.

First, the context-free atomic elements can leverage the alignment between XML and SQL to share definitions and constraints on datatypes. Or, perhaps more likely, the XML datatypes can be used by the data architecture team to define their desired "to-be" specifications for these foundational data elements. Core components can also be aligned to reflect the tables and entity definitions in the data architecture.



Second, the components can be aligned with the data architecture reference model. Standard forms for the components can be designed for systems of record and reference, master data structures, and standardized data queries. Such an alignment will minimize the need for custom SOA data services to align data requirements with data resources. The number of service accesses will be reduced while improving the interoperability of messages across multiple applications.



Finally, the information model underlying the architecture can provide a layer of abstraction to allow the information architecture to adopt the dictionary definitions from the data architecture to create a shared vocabulary.

With these three alignments, the information architecture can then focus on its SOA requirements. These are fundamentally to define service interfaces using XML Schemas embedded in WSDL as the format of the SOA metadata. To be successful, SOA metadata must be XML-, infrastructure-, and team-aware.

Metadata is at the heart of the service oriented architecture. Technically, a service is a computer program that responds to input documents, performs a function on the input data contained in documents, and delivers output documents. What makes SOA unique is that it defines these computer programs in terms of *interfaces*, not what goes on inside the program but just the behavior it exposes to the outside world. Service oriented metadata must be able to define these interfaces and the details of their input and output documents.

XML-Aware: Document Oriented Metadata

The first metadata consideration for service interface definitions is to clearly and precisely define the input and output documents as a core part of a service interface definition. In SOA, input and output documents are primarily XML documents therefore SOA metadata should be fully *XML-aware*—that is it should be able to describe all of the structure and nuance found in XML documents. XML is a document oriented approach to architecture. Since services work by exchanging XML documents, *service orientation is also document orientation*. The distinction between document orientation and system orientation is subtle but has a significant impact on the metadata.

Most organizations have expertise in using metadata in the form of Entity-Relationship (ER) modeling for databases and Unified Modeling Language (UML) for applications and systems. These types of metadata serve us well in designing systems and enterprise data models. However they do not capture the detail of the document structures nor the relationships between them as needed to implement scalable service orientated architecture. Metadata used to describe XML documents is unique from these types of classic metadata.

Infrastructure-Aware: System Level Metadata

Metadata for service orientation needs to precisely describe the types and structure of XML documents. The W3C *XML Schema* language is used for service interface definitions. At the system level, XML Schemas are well suited as metadata for services. They are precise and industry support has been widely implemented. Wide industry support ensures that service oriented systems can directly process metadata.

Unfortunately, XML Schemas are complex and difficult to master. In order for a team to mainstream SOA into the team development processes, they need a common methodology for designing the metadata and business oriented tools that easily represent relationships between schemas used in the services integration.

Team-Aware: Human Level Metadata

The last consideration for SOA metadata is that it must be readily usable at the team level. In order to scale SOA teams must be able to capture and manage the structure and semantics in the documents exchanged between the services in addition to supporting XML Schemas. This team function is critical. While SOA standardizes connectivity and interface definitions, it does not standardize document formats or meaning of the data in the documents.

The document orientation of services enables the invention of service vocabularies to overcome the usability limitations of schemas. In the human world of spoken and written language, linguists use *vocabularies* to describe the words and context in which those words have specific meanings. We can use vocabularies as SOA metadata to describe XML data and context in which it has meaning.