Excelon Corporation is an XML solutions provider focused on using XML for large-scale middleware solutions. The company's foundations are in object-oriented solutions for databases. Excelon has been prescribing XML technologies for 3 years.

Pitch:

Excelon sells a product called ***Portal Server*** that is coupled with their *xConnect* tool. *Portal Server* has 4 sets of functionality that are important to our business; *Aggregation*, Management, *Transformation*, *Delivery*. They describe each as follows:

- **Aggregation**: The process of compiling many data sources into a single XML document that exists within the *Portal Server* environment
- **Management**: The maintenance of accurate, and up to date data within the aggregate XML document
- **Transformation**: As in the world of XML, the conversion of one XML document into another well formed document (often into HTML or another mark-up language.)
- **Delivery**: The management and reporting of a transformed XML document, or of a section of XML from the Aggregate. Deliveries may be persisted and linked to a user or session ID (the notion of Caching of an XML document /or result set)

  More info on portal server can be found at Exceloncorp.Com.

xConnect is a solution for multiple data source referencing. It serves to maintain the database connection information and sources for the aggregation component of *Portal Server*. This tool acts much like a general connection object for multiple data sources.

Theory:

Excelon's model requires the use of XML as a primary data source. They champion the benefit of aggregating multiple data sets into one master XML document. This super XML document would be based on a *common information model*, or XML schema that reflects the full collection of data used by a department or organization.

For parametric search for example, Excelon would suggest aggregating all Parametric Search and Meta-data SQL tables into one common information model. Instead of executing queries against SQL tables, they suggest executing queries against a large XML document.

The arguments for a common information model are simple:

- XML is designed for extensibility, making it easier to maintain and more flexible than a database.
- XML was designed for interoperability and is exceptionally platform neutral.

The *common information model* would enable us to worry much less about ADO connection strings, creating and manipulating SQL stored procedures, or creating, maintaining and installing COM components.

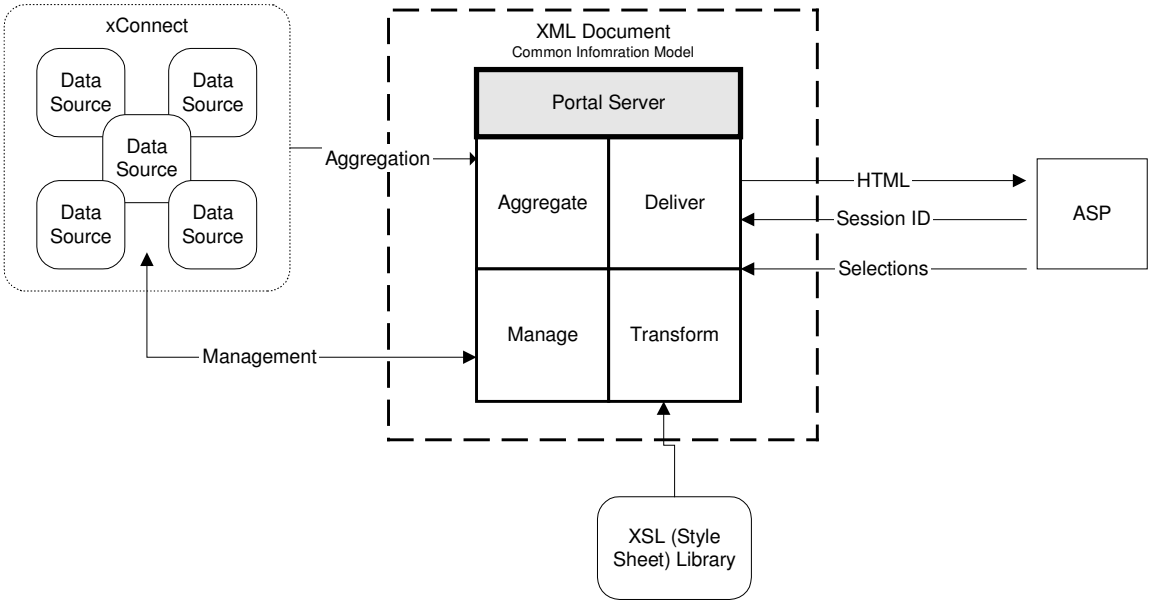
The *Transformation* and *Delivery* portion of the *Portal Server* would replace much of the component object logic we currently have. A transformation would extract

information from the persisted XML document, while the delivery component would associate that transformation with a user session or common result set.

By moving business logic into XML transformations (more commonly known as XSLT) we could combine the speed of compiled code and the flexibility of scripted code in one tier. According to Excelon, XSLT would execute logic as quickly if not faster than our current, ADO dependent COM architecture.

The architecture also affords use of COM objects as extensions to the transformation process. If for example we have compiled code that we wish to continue using, we can simply add that COM process to the transformation of our XML.

The following diagram describes parametric search in the Excelon *Common Information Model* architecture:



Evaluation

As interesting as these theories may be they would have an immense effect on our current designs.

The theory behind a common information model is clearly enticing but may also prove to be an over-adoption of technology. Just as it may be possible to over-compenentize our business processes, it may be possible to rely too heavily on XML technologies.

Our experience has taught us that XML documents in DOM form may be up to 7 times the size of the same document in text file format. While Excelon boasts compression that maintains only a 2 – 3x ratio (the industry average is a 5x ratio) per document the overhead associated with the common information model would be unforgivably large.

Even though XML documents by nature should not have as much data scarcity as a relational database (the absence of data can be indicated by the absence of an XML element) the load on *Portal Server* hardware would be tremendous.

This architecture also relies heavily on the *Portal Server* software and would force us into a long-term relationship with a company with whom we have had limited experience.

Overall it seems that only the *xConnect* would be a useful tool in our current environment. More research will prove if *xConnect* is capable of working in our server environment.