Objectivity Case History

Customer Information

Customer:

Status:	(X) SOLD ( ) Prospect ( )Integrator
Industry:	Aviation Controls
Application Domain:	This is a "**Detailed**" Case History – See Below
Platform:	Windows NT
Compiler:	Microsoft Visual C++
Other Tools:	Rational Rose, Python

Buying Criteria

See below

Why Objectivity

See below

Contact Information

Objectivity Rep:	John Jarrell
Customer Contact:	Lenny Hoffman
Phone:	
Email:	

Objectivity Detailed Case History for Rockwell Collins Aviation
Jack Murray

Project Name:

Aviation Systems Design Database (ASDD)

Project Purpose:

To create a common design repository for all new aircraft cockpit designs. The repository promotes reuse of design specifications and has the intelligence to verify compatibility of components in new designs. The repository is an infrastructure component set up to be usable by several engineering applications (many as yet undefined). The same group is also doing the first implementation called SAGE (no particular meaning) ; it has a slick Web Browser-like interface to enter components, connect them for a complete cockpit design and make changes.

Who is our customer?

Our direct customer is the ASDD development group headed by Lenny Hoffman. In turn his customers are other project teams which will use the infrastructure and end users (engineers) who are using the first implementation, SAGE.

What does the customer consider value?

Lenny has always considered the value in Objectivity to be in its distribution, scalability, performance and ease of use. Ease of use includes not needing to write mapping code and Objectivity's automatic associations and related methods and enforcement of referential integrity. Lenny has always been a big fan of STL. He used it in the ASDD application before it was supported by Objectivity. He is now using our STL support and considers that to be valuable. Finally, he sees schema evolution to be a very valuable feature in this dynamic development environment.

Are we delivering value?

In some of the above we have not met expectations. He has implemented his own associations to avoid the time-consuming complexity it adds to DDL. He has also run into some quirkiness in our initial implementation of STL and Schema Evolution but feels we will work this out over time.

They now feel the real value we provide is a "Basic Engine" to hide the complexities of persistence, network (IP) handling, basic data integrity and transaction handling. Scalability because of the distributed client, distributed server nature of Objectivity is also of great value to them.

How has Objectivity helped differentiate their product in the marketplace.

Even though their customers are internal, they do have other choices. The biggest differentiation is the ability to do true configuration management (several levels deep). Other initiatives within Rockwell have been unable to do this due to the limitations of relational databases. With ASDD, if something changes only that object needs to be changed. With relational, all related rows must be changed and code complexity increases while performance drops. As Lenny says if you try to do with relational what we are doing in the area of Configuration Management "You might as well just shoot yourself".

Discuss the Selection/Evaluation process including competition and why Objectivity was selected.

They began their selection process in the summer of 1995. They looked at only object databases because the felt relational would be too slow and require writing too much code. After presentations from Objectivity, ODI and Versant, they chose Objectivity for further prototyping and benchmarking. After about a year of testing they made their first purchase and began developing ASDD.

On what platform(s) are they developing?

Originally a big part of the Objectivity decision was the fact that it was the only ODBMS to support VAX/VMS. Luckily they gradually moved to first Windows 95 and then Windows NT as we dropped VMS.

On what platform(s) will they deploy?

Windows NT.

How is Objectivity actually being used in the project?

It is being used to store component definitions, their properties and rules for connections among them. Objects represent boxes, wires, data transferred over the wires, the format, frequency and quality of the data transferred. They also store a complete Meta-Data model and all the information needed for "Introspection" (total knowledge of what all the pieces of the model mean and rules for relationships). Other objects are stored for their own Event Notification.

What Language(s) are being used?

Primarily C++ but they are also using Python which is used to indirectly manipulate Objectivity via the ASDD layer and its metamodel and introspection rules. One of the other groups within Rockwell is planning to use ASDD in its application, which will be written in JAVA.

Are any of the following technologies implemented:

Object modeling tools

Rational Rose is used to document their model, not for code generation. They are moving toward UML.

MFC/ Other Class Libraries

Yes, MFC is used for part of the GUI implementation of SAGE. They have invented their own WYSWYG HTML interface, which can not yet be done with MFC. STL has been used (without the Objectivity support) for (among other things) maps and Indexes.

Object Request Brokers

Considering for the future.

Web/Email integration

They can generate HTML for web access that looks identical to the native interface (read only).

Other of interest

Event Notification:

Again, they have built their own. It uses “simplified polling” at a less granular (container) level where a transaction can register an interest in a container (not an object). If that container changes, the transaction receives an event log that the transaction analyses to determine the impact. Because there is no central registration point there is no bottleneck and this becomes highly scalable. Lenny compares it to the Objectivity distributed architecture where each computer shares the load avoiding a central bottleneck.

Configuration Management

They have built in the ability to have multiple concurrent changes (versions) to the same design. This is true configuration management, several levels deep using policies which go beyond referential integrity (this is total data integrity) or simple version control.

Features used:

Indexes

No, they use STL.

Maps

No, they use STL.

Named Objects

They use one as an entry point. Most of the rest of the lookups are navigational.

Associations

They have implemented their own bi-directional associations with policies defined in the metadata model, which trigger creation of the proper relationships.

Versioning

No. They do their own Configuration Management.

Predicate Query

No, they use their own "Introspection" for navigational based, intelligent searching.

SQL

Considered this but decided to do reporting with Python.

ODBC

Considered this but decided to do reporting with Python.

FTO

Possibly in the future.

DRO

Possibly in the future.

Schema Evolution

They are creating their own dynamic schema using the metaschema built into ASDD.

Heterogeneity of O/S

Planned initially but now all Windows XX.

Heterogeneity of Language

Yes, initially C++ but JAVA will be used in the future.

Multithreading

Not needed in their design.

ODMG interface

Not believers.

STL

Yes, although mostly their own.

How are they using Containers?

Since their configuration management uses change packages where everything under change is put in its own temporary container, everything else is basically read-only. Since there are no concurrency issues, they put everything in one container until they have a need to break it out. This is done administratively. They can re-containerize anything, anyway they want at any time for future flexibility.

Describe how transactions are used (long vs. short, MROW vs. non-MROW) and describe GUI and its transaction semantics (short vs. long locks for update).

They use short transactions under their GUI. All read transactions are MROW. Every user action creates a transaction that accomplishes everything the metamodel's rules require. No transaction is left open while a user looks at the screen. They may need to add checking for changes since the data was displayed to avoid walking on updates since the display. This is a slight exposure because of the way their Configuration Management does changes in a separate container via a "Change Package". The exposure here is only when more than one engineer is working on the same change package at the same time. They believe this can be done with a comparison to the screen data before updating.