A Component-Based Development Case Study

Jim Q. Ning

Andersen Consulting
3773 Willow Road
Northbrook, Illinois 60062-6212, U.S.A.
+1 847 714 2537
jim.q.ning@ac.com

Project Background

The client referenced in this paper is a global machinery manufacturer who currently goes to market through a global network of dealers. The client supplies these dealers with many disparate systems which include:

- Custom-built, AS/400 "green screen" core systems
- PC-based systems
- Corporate 3270-based systems

The client wants to create a stronger extended enterprise by linking customers, dealers, and the corporation through the use of technology. To deliver this vision of an extended enterprise, the systems listed above need to be integrated, modified and enhanced.

Component Technology Objectives

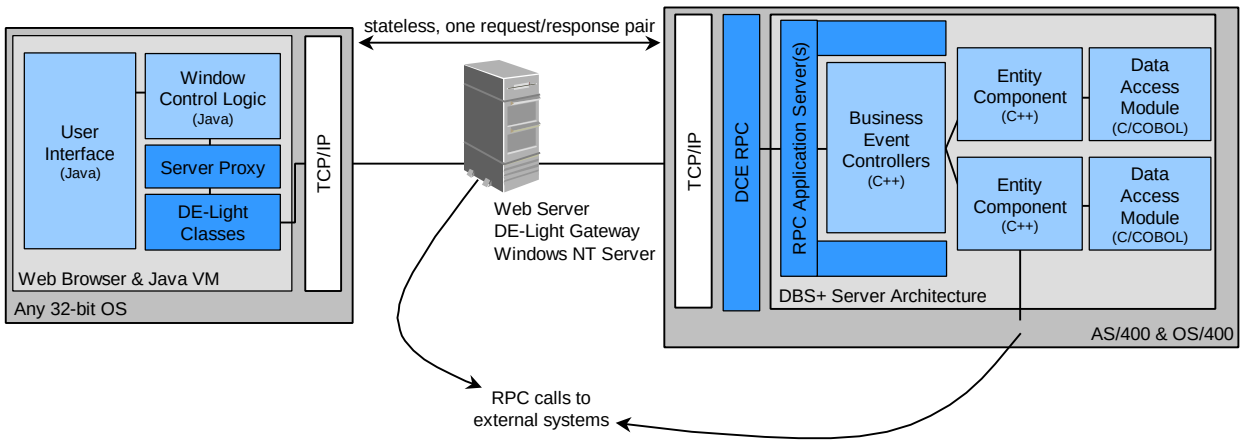
To build the systems of the extended enterprise, the client selected component technology as a key enabler. The client believes that components will help to deliver:

- An open architecture
- Better systems engineering and easier maintenance
- Ability to transparently re-locate processing in the future
- Business that is developed once and reused across applications
- Universal access

Architecture Overview

The project featured in this case study is currently delivering the first in a series of a component-based applications. This application is being built on a 3-tier client/server architecture that uses the AS/400 for the server platform. The overall architecture is shown in the diagram below.

The Java-based operationally thin-client, which is hosted in Internet Explorer, communicates with server-based business logic through the DE-Light (an IBM middleware product that supports DCE) RPC gateway. The client invokes operations on Business Event Controllers. The Business Event Controllers collaborate with Entity Components to complete an operation. The Entity Components interact with the database through Data Access Modules.



Lessons Learned

Development Lessons

- Use most experienced resources on design
- Design by business functions, not by architectural layers
- Code shells improve productivity, but they do not help to develop programming skills
- Mapping to an existing data model is difficult
- Use variability analysis effectively to forecast future requirements
- Component packaging is complicated by dependencies between components
- Component development tools are immature
- Vendor-provided training (on C++, Java, etc.) is not enough
- Coordinating changes is more difficult

Architecture Lessons

- Budget more (and resist to cut down) on architecture work
- Define architecture early
- Prototype a vertical slide of the application
- Java-based thin clients are difficult to implement
- Integrating Java applets with other desktop applications is difficult

Performance Management Lessons

- Performance is impacted the most by application design
- Excessive data conversions when integrating various technologies and platforms can lead to performance problems
- Code shells can lead to performance problems
- Excessive encapsulation can lead to performance problems

Testing Lessons

- Testing is harder if development was done by architectural layers
- Use automated testing as much as possible
- Debugging is difficult and tools must be used
- Testing efforts should happen earlier in the development lifecycle
- Component dependencies impact testing

Project Management and Estimating Lessons

- Extra contingency must be planned
- Iterative development must be well managed
- Project tracking can be more granular
- There are more artifacts to manage and impact analysis is harder
- Plan on spend more time to refine the build process

Client and Team Organization Lessons

- Managing client expectations is more difficult since visible results come later than usual in component projects
- Component technology impacts client's IS organization
- Organizational communication overhead is significant
- Plan to overstaff the development team
- Greater collaboration between development team members is necessary
- Use a matrixed team organization
- Separate programming lead and project management roles
- Designate full-time resources to legacy integration
- Use outside contractors for programming skills only