Lessons Learned Building an Interface A Host System Using a Commercial Application Platform

Alan Weber
Alan Weber & Associates
Tim Sowell
Invensys Wonderware

Outline

• Presentation objectives
• Background
  – Industry context
  – APC/EES application requirements
• Implementation approach
  – Development/test environment
  – Application platform basics
• Lessons learned
• Benefits summary
  – Developer perspective
  – End user operational impact
• Acknowledgements
Presentation Objectives

- Explain rationale for choosing an application platform as the Interface A host implementation environment
- Describe the impact of this choice on the software development process
- Share the implementation lessons learned
- Highlight the benefits of this approach from the development and end user operational perspectives
Background

Industry Context for Interface A

- Standardization status
  - Detailed specifications complete and balloted, but still a moving target
  - Adoption underway
- The "other end of the wire"
  - Most effort to date has been on the tool side
  - Very little focus on the host side
  - "What are we going to do with all this data??"
- Will likely spawn an entirely new application market
  - Hard to predict what process engineers will want next after a few months of unprecedented tool data access!

Background

Interface A Adoption Challenge

- Bad news
  - It is difficult to validate an implementation of a communications interface when there's nobody to talk to…
  - It can be expensive to be first
- Good news
  - There are bridging/migration technologies for breaking this "chicken-and-egg" cycle
  - Use of commercial communications packages and applications platforms offer attractive alternatives to in-house development
Background

APC/EES Application Requirements

What will fab users need to fully deploy Interface A?

- Connectivity
  - Devices
  - Applications
  - People
- Data storage and management
  - Real-time, granular, voluminous
  - Data quality
- Security and reliability
- High performance and scalability
- Agility to support changing requirements

APC/EES Environment

Application Platform Architecture
Interface A Host Implementation

Minimal Product Requirements

- Provide process and operational data to APC/EES applications
- Comply with E134 – no proprietary shortcuts
- Support all types of tool data request
  - Trace data
  - Event data
  - Exception data
  - Ad hoc data (on-demand data)
- Insulate users from details and dynamics of E134
  - Application developers
  - Process engineers

E134 Data Collection Plan (DCP)

- Event Request: Data is sampled when a tool event occurs
- Trace Request: Data is sampled on a periodic time basis
- Exception Request: Data is sampled when a tool exception occurs

Each data collection plan can include any number of each type of data request
E134 Report Structure

Data values are delivered in the order requested in the DCP (not tagged)

Event Report Returned when a tool event occurs

Exception Report Returned when a tool exception occurs

Value Data item list (all report types)

Trace Report Returned on a periodic basis

Implementation Approach
Implementation Approach

- Used Wonderware ArchestrA application platform as Interface A host system
  - Complete development, integration, deployment, execution and support environment for mission-critical manufacturing systems
  - Configurable client applications suite for common visualization, reporting, and analysis functions
- Used Cimetrix CIMPortal product as Interface A equipment-side test vehicle
  - Includes Equipment Modeler and Equipment Simulator for assembling complex behavior from E120 nodes

Development/Test Environment

- MES/APC Frameworks
- Existing CIM System
- GEM/SECS
- CIMPortal™
- Pre-EDA
- EDA Capable
- Semiconductor Mfg. Tools
- EDA Interface
- COM, SQL, MQ, etc
Lessons Learned (1)

- Development in an application platform environment requires a careful mapping between two (or more) domains
  - Interface A syntax and semantics
  - ArchestrA development and execution environment
- You must truly understand how a platform is used to do this mapping effectively
  - Development tools and processes; built-in services
  - End user application functions and interaction modes
  - A couple of design iterations are required to get this right
  - A good mapping results in a very natural end user "feel" (and rapid acceptance)
- Corollary: the job is a lot easier if you pick a platform that is well suited for the domain
Lessons Learned (2)

- Example mappings include
  - Data types across the various standards/systems
  - DCP structure and interface object configuration tools
  - Visualization/navigation of all DCPs
    - Equipment level
    - Fab level
  - Report structure and application variable naming
  - DCP life cycle and interface run-time behavior

See following slides....
Data Collection Plan Definition

EDATool Object Configuration

- DCP Name
- Trace Request ID
- Start & Stop Trigger Events
- Application object attribute name
- Data type expected (pull-down list)
- Tool parameter name & location

Tool/Fab-Level Visualization/Navigation

EDATool DCP Treeview Display

- Collapse/Expand Options
- TraceRequest1 for DCP3 on tool ETCH22
Hierarchical Variable Naming Scheme and Historization Configuration

DCP3.TR1.TC1.PV
Present Value of Thermocouple1 in TraceReport1 of DCP3

Automatically store in real-time database every 100ms

Lessons Learned (3)

• It is impossible to build [one end of] an interface without something robust to talk to
  – Must also be clear which version of the standard is implemented
• Debugging this kind of software is still very tricky
  – For example…. If the requested data doesn't come through, where’s the problem?
    • Client didn't request properly
    • Tool didn't understand the request
    • Tool didn't handle event properly
    • Client didn't process the response properly
  – Must anticipate/handle errors from many sources
  – But the problem is well bounded, and support tools exist [in an application platform environment]
Monitoring and Debug Tools

- Object Viewer can be used to monitor EDA Interface status and monitor statistics
- Diagnostic operations invoked by setting diagnostic triggers
- Note "Quality" column…. this item exists for all attributes

Lessons Learned (4)

- Built-in data quality support features are very useful
  - Separate field for every object attribute
  - Prevents making bad decisions from data
  - Values defined by OPC specifications
Lessons Learned (5)

- Requirements for a host implementation of an interface standard go well beyond the standard itself
  - Must walk in your customers’ shoes
  - This is the real opportunity for differentiation
- Example deltas to the Interface A specs include
  - Pre-defined ad hoc report request (query)
  - Auto-activate feature on DCPs
  - Import/export format (enables all sorts of fab-level capability)
  - Treeview navigation, collapse, expand, etc.
  - UUID assignment and management tools (à la Recipe Mgmt)

Lessons Learned (6)

- There were MANY things we did NOT have to build which are well covered by built-in application platform services
  - Logging, alarm generation and management
  - Schema generation, attribute naming and delivery to the applications, persistence, historization
  - Fault tolerance, load balancing/performance management
  - Security, change management/effectivity/propagation, packaging and deployment, installation
- The ISMI Scenarios document was a useful functional checklist and guide to test plan development
  - The Exception scenarios additions will help as well
Benefits Summary

Developer Perspective

• Use of an application platform allows developers to focus on domain issues rather than distributed computing infrastructure
  – Eliminates most difficult architecture decisions
  – Don't fall into the "I'm sure we can build what we need here for a fraction of the license fee" trap
  – Once you've built a couple of frameworks, you don't really want to do it again anyway….

• Reliable solutions can be developed and deployed more quickly
Benefits Summary

End User Impact

- Once data is collected by an application platform, a wide range of standard, familiar clients can use that information
  - Visualization, monitoring, analysis, control, reporting, etc.

Operational Impact

Development Cycle Time Reduction

- Application platforms affect almost every phase of manufacturing system development
  - Requirements cover unique value add, not system technologies
  - Design focuses on how to use existing features and services rather than basic application architecture
  - Implementation consists of configuration, specialization, scripting with minimal new code development
  - Unit test only required for completely new functions
  - Integration effort limited to new data sources and transactions
  - Production rollout and other scale-up processes are standard system administration tasks
### Operational Impact

#### Development Cycle Time Reduction

<table>
<thead>
<tr>
<th>Phase</th>
<th>Traditional Development</th>
<th>Platform-based Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>4 weeks</td>
<td>2 weeks</td>
</tr>
<tr>
<td>Design</td>
<td>8 weeks</td>
<td>2 weeks</td>
</tr>
<tr>
<td>Implementation</td>
<td>8 weeks</td>
<td>3 weeks</td>
</tr>
<tr>
<td>Unit Test</td>
<td>2 weeks</td>
<td>1 week</td>
</tr>
<tr>
<td>Integration</td>
<td>4 weeks</td>
<td>1 week</td>
</tr>
<tr>
<td>Pilot</td>
<td>2 weeks</td>
<td>2 weeks</td>
</tr>
<tr>
<td>Rollout</td>
<td>12 weeks</td>
<td>3 weeks</td>
</tr>
</tbody>
</table>

- **40 Weeks**
- **60% reduction in time to validated pilot**
- **75% reduction in Production rollout phase**
- **65% reduction in overall project time**

### Operational Impact

#### Improving Overall IT Efficiency

For a piece of equipment, OEE is the time a tool is actually working on a product that will be sold for revenue.

The same model applies to an IT organization....
Operational Impact

*Improving Overall IT Efficiency*

- Application platforms can reduce much of the time spent in the non-productive activities
  - Unscheduled downtime – reacting to system excursions, crashes, corrupted databases, log file overflows, network system overloads
  - Scheduled downtime – maintenance activity, such as installing new patches or releases, backups, maintaining user profiles, virus scans, database integrity checks
  - Engineering time – evaluation of new system technologies, prototype development, experimental programs that may never make it to production
  - Standby time – waiting for required resources, such as capital/expense budget, specialized expertise, project approval

**Operational Benefits**

*Value of On-line Real-time Information*

- **Perceived Factory Uptime %**
  - Based on a true story....
Acknowledgements and Thanks

- My long-time partners, Jim Hollister and Paul McGuire
- My co-author, Tim Sowell, and other colleagues at Invensys Wonderware
- Dedicated volunteers and staff of SEMI Standards
- International SEMATECH

Backup Material
Industrial Application Platform

**Key Product Requirements**

- Design and development environment
- Event-based processing, scripting and calculation capabilities
- Data acquisition and field device integration
- Data visualization and monitoring
- Reporting and ad hoc query capability
- Alarm and event management, historization and security
- Support for industry standards such as SEMI E134, OPC, SQL
- Internationalization
- Inter-application communications and name service
- System diagnostics and system administration
- Version management
- License management and centralized deployment

**ArchestrA Application Platform**

**Integrated Development Environment (IDE)**

- Fab-wide application server IDE/GUI
- Serves as both development tool and online admin tool
ArchestrA IDE
Object Templates

- Templates define abstract objects for managing fab equipment and applications
- Also includes objects for computing environment
- Templates can be composed hierarchically

ArchestrA IDE
Hierarchical Model/Views

- Displays fab model as instances of templates
- Can be viewed by
  - Model hierarchy
  - Computing node deployment
  - Derivation dependencies
- Manages assignment and deployment of objects
ArchestrA IDE
Object Details and Properties

- Displays details for selected object
- User can customize objects
  - UDAs
  - Alarms
  - Scripts
- Changes managed via change control and managed deployment

Shared Services
Support Platform Longevity

Security Service (Flexible Model)  Deployment Service  Scripting Service
Alarm Service  Event Service  Messaging/Name Service
Historian/Storage Service  Visualization Service  Internationalization Service
Configuration Service  Configuration Service  External Data Integration Service (E134/OPC)
Diagnostics Service  Admin/Version Service  Licensing Service
Real-Time Database Requirements
Handling Tool/Process Data

- High performance and scalability
- Schema generation and management tools
- Direct/flexible integration with data collection system
- Universal access from applications and workstations
- Built-in standard functions for
  - Data quality verification
  - Limits checking
  - Alarm generation
  - Transformations for common queries
  - Historization
- Comprehensive self-diagnostics and system administration capabilities

IndustrialSQL Real-time Database
Granular Security Model

- OS
  - User
    - Role
      - Many
      - Many
      - Operational Permissions
        - Many
        - 1
        - Security Group
          - 1
          - Many
          - Object
            - 1
            - Many
            - Attribute
              - Many
              - Permission Requirement
                - Many
                - Free Access
                - Operate
                - Read Only
                - Secured Write
                - Verified Write
                - Tune
                - Configure