

January 21, 2011

Topic: Dealing with the Complexity of developing an ISA-95/B2MML-based Operations Data Store and Application Platform

The Need for Event Driven Operations Management:

"There are enormous financial and strategic benefits to implementing event-driven business processes, because they suit the inherently event-driven nature of many aspects of the real world. Event-driven business processes are not just traditional processes made to run faster; rather, they have specific characteristics that distinguish them from business as usual."

Roy Schulte
Gartner

The ODS Requirement

- Bidirectional collaboration between the Supply Chain Planning and the plant work processes requires logic:
 - Event-driven
 - Concurrent
 - Complex
 - Distributable
 - Reusable
- Logic and connections must be easy to develop and change

The Obstacle to the ODS Requirement

- Development of concurrent, event-driven software is extremely complex
- Distributed systems exacerbate the complexity
- Systems are difficult to change and govern
- The underlying reasons
 - Today's software lack concurrency and easy programmability
 - Development tools (e.g. Visual Studio™) do not address the complexities of concurrent, event-driven software and are designed for craftsmen, not process engineers.

The ODS Solution

- An ISA-95 based System Development Platform simplifying development of event-driven, concurrent systems
- Combines three powerful concepts :
 - An ISA-95-based composite application development framework
 - Unifying service architecture
 - Distributed execution capability
- Objective: Provide a common platform for solutions enable integration and analysis of process data and implementation of rules-based logic and feedback
- KPI data is analyzed and processed in real-time as it is generated to capture excursions before significant impact to processing and operations
- Rules-based logic provides feedback to notification system and other critical system through ISA-95 standardized services for integration

The ODS system should contain the workflow management application and their work process instances as the ODS data model is not based on an asset model but is based on work process and resource data structures per the ISA-95 model. The ODS is one location for mapping of syntax, semantic and workflow definitions and governance.

The ODS explicitly handles the definitions and governance production, real-time inventory and maintenance operations management. However, the LIMS has the quality operations management (QOM) data and structure definitions AND the quality workflow definition and transaction engine. As such, the LIMS requires hard (not loose) coupling between the LIMS and ODS to make up the "operations platform". The LIMS is where these definitions will change first and most often so it retains QOM definition ownership and synchronization. The ODS will govern change for POM, IOM and MnOM definitions and synchronize them with other applications and work processes.

In developing the user requirement specification (URS) for the Operations data store (ODS) and the associated single canonical definition for operations integration, many manufacturer make the common mistake of only developing a URS with only high level businesses processes and supporting exchange interfaces. The ODS URS typically leaves out consistent business process for maintaining data integrity through strict governance of operations syntax and semantic definitions and structure in the ODS and in the other applications supporting ODS definitions such as the IOW, Operations Log book, and LIMS. Scheduling, Reporting framework, Process Control Performance Mgt System, Historians, Supply Chain Planning, and other applications should get their operations definitions and structures from the DEFINED resource definitions and management applications of ODS, IOW, Operations Log book, and LIMS through DEFINED governance business processes for change management and validation synchronization. Otherwise, the real-time data will be corrupted due to too much translation and not be able to be validated. Based on the described detailed URS, the functional requirement specification (FRS) defines this architecturally. Additionally, the URS for the KPIs system and Real Time Analysis Tools may be defined as an data definition owner and user but within an explicitly defined role with governance work processes between enterprise and operations data stores. An "Operations Data Definition" tracking database and governance workflow application should be defined and developed. The "SC/Planning Data Definition" ownership are to be architecturally defined where the business processes for data validation are synchronization with the "Operations Data Definition" application stack. Without these architectural elements, data integrity, reporting, and decision making will create large process and production losses. In the FDS, the business processes detail that Scheduling share business processes between SC/Planning and Operations Management; the Integration Matrix illustrates that there is an Operations definition synchronizing issue to handle with to-be-developed governance business processes and defined exchange interfaces.

The approach to B2MML is a rationalization of point-to-point (application-to-application (A2A)) semantics and structure between meta data of individual systems/application, not application-to-framework (A2F). Where this works as a mapping method, the method does not work as an Operation definition and data integrity validation method. Operation definition and data integrity validation requires the rationalization of each department's Syntax definitions to single definition in a canonical mapping at the Operations Management ODS framework.

Our data validation & reconciliation (DVR) approach should be differentiated by a bottom assessment of manual data collection methods across operations TOWARDS minimizing transcription cycles, collection lag times (real-time manual DC), and paper-based forms. This is done with real-time workflow application on wireless industrial tablets and PDAs (with e-signature and GPS tracking). Risk reducer: Increase DVR for automated DCs from Level 2 systems.

Commercial-off-the-shelf (COTS) solution for ODS System and B2MML Interfaces

Applying **Ninety-Five's Easy95** Application Framework and Platform for the ODS System, B2MML Interfaces, and reporting configuration to bring greater Value in the short term through accelerated development applications and integration framework and in the mid and long term as a foundation for operations data management and data validation & reconciliation (DVR)

Although manufacturers, software vendors and integrators tend to believe that an ISA-95 based framework is relatively simple to build, many initiatives fail and get stopped due a general lack of understanding of the actual ISA-95 models and their interrelationships. Even people having a good track record in understanding ISA-95, application development is typically under estimated. **Ninety-Five** took a new approach with **Easy95** towards MES/MOM applications, operations master data, and operations data storage by basing the approach on today's web-based technology. The **Easy95** approach has been proven in real projects that are operating live.

The challenge of building an ISA-95-based application framework and platform, like **Easy95**, is really related to the fact that ISA-95 is a set of interrelated resource and function/task objects provided in the form of UML models. The objects are not just isolated objects but have pre-defined relationships with each other. When a plant environment requires an Operational Data Store (ODS), the name already implies the need to store real-time, historic, and exchanged data. This of course requires some kind of database application. Although there are some object oriented databases available on the market, almost everyone works with relational databases. This means the ODS database requirement is to build an object to relational mapping, which is established by using a persistence layer like Hibernate*. Persistence means that our application's data outlives the application run process. Relational Database Management Systems (RDBMSs) represent data in tabular format (like a spreadsheet) whereas object-oriented languages represent it as an interconnected graph of objects. Hibernate is an object-relational mapping (ORM) library for the Java language, providing a framework for mapping an object-oriented domain model to a traditional relational database. Hibernate solves object-relational impedance mismatch problems by replacing direct persistence-related database accesses with high-level object handling functions.

In real-time manufacturing operations, very high quantities of data are collected and retrieved which require the database design to focus on performance as a high priority. Where data storage is fairly straight forward, correct design for retrieval is much more critical when balancing high response with the rules governing interrelated resource and function/task objects. The database application also must store two types of data:

- 1) definition data
- 2) operations data.

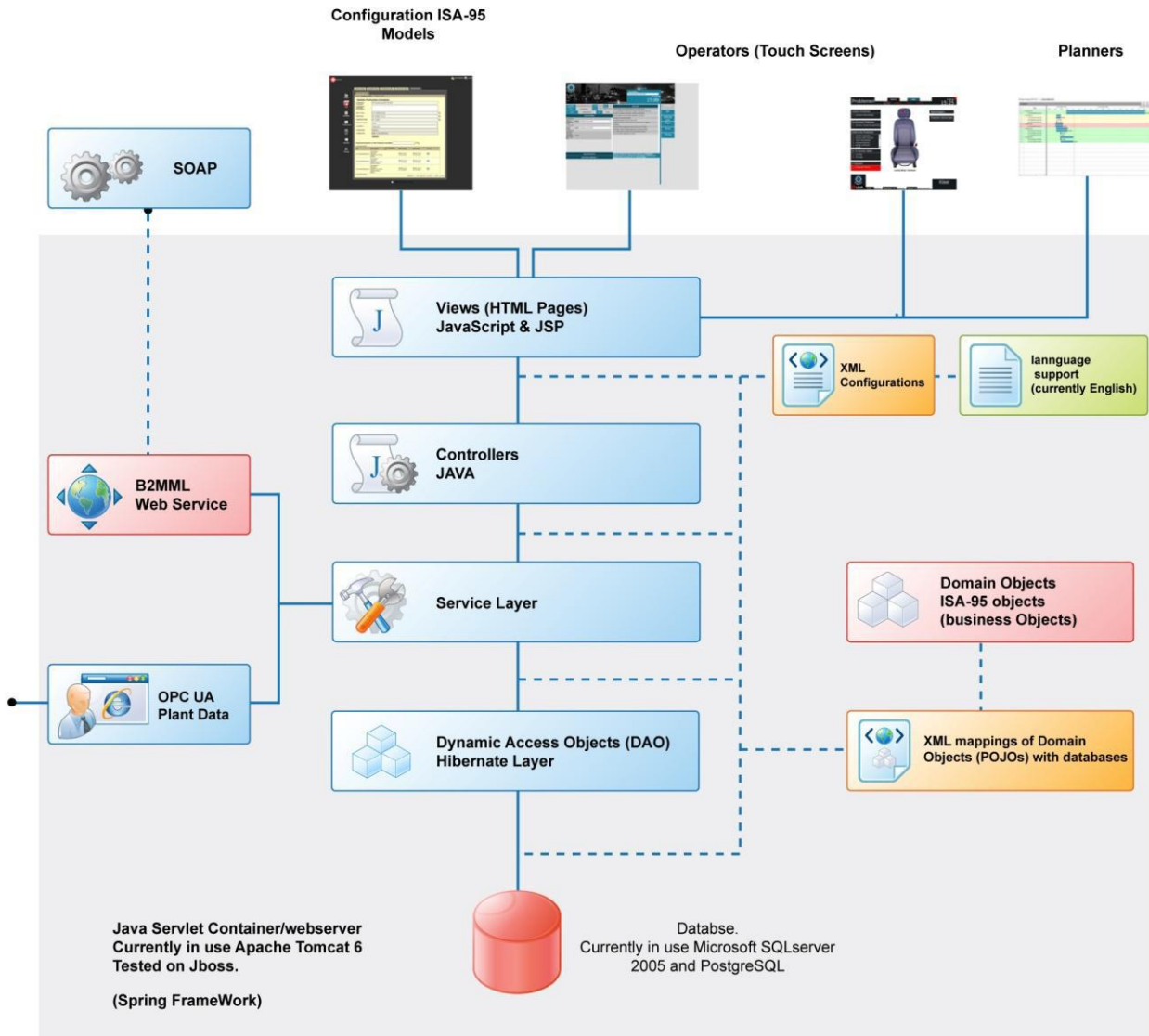
In the UML structures and rules of ISA-95 models, these data types are combined. In an application framework and platform approach such as **Easy95**, the two data types are split into different user interfaces for clarity while keeping the underlying logic applied. Another aspect of **Easy95** or similar platforms (not defined by ISA-95) is the critical functionality of lifecycle management of every piece of data such as creation date, author and last-modified-by information. A critical ISA-95 aspect of the platform design that may seem obvious but many users struggle to implement correctly are the properties for object classes and object instances for resources. ISA-95/B2MML requires only that a value be applied. In **Easy95**, this property value can be any type of data (time, numeric, string, document, etc.) where the value can be one value or multiple values or needs to be selected from predefined lists. Another critical functional component provided by **Easy95** is the basis to define the source of the value (OPC or B2MML). This functionality is currently be extended. The whole **Easy95** database consists of about 200 tables.

To build applications on the **Easy95** framework and platform, the engineer or designer is also able to easily view, add, delete or edit the data in the defined ISA-95 structure. Given the amount of options in the UML rules in each of the nine ISA-95 models, the design time to build all these accurately in a custom application is a few man years. All these views are available as a web application in **Easy95**.

The original fourteen plus the new four B2MML schemas are really data definition in transport mechanisms by representing ISA-95 models, interrelationships, and UML rules in XML. XML is a hierarchical structure. This means that the ISA-95 objects also need to be converted into a hierarchical structure. This in itself is not that complex, but B2MML is provided in the form of XSD files, so the mapping is important since it is a structure definition that must be accurately applied. But before you do this mapping, the above mentioned persistence layer is required in an ISA-95 framework and platform to build applications.

Since B2MML schemas only provides the XSD containing both the data methods and the data definition, the web services and methods supporting the ISA-95 interrelationships must also be designed into the platform as in **Easy95** to build applications and integrate them. **Ninety-Five** (company owning **Easy95**) designed a logical

structure by splitting the services and by grouping the methods logically. These services make it easy to connect the Operation Data Store part of **Easy95** to any Service Bus and integrate the system in the background to existing infrastructures. The **Easy95** methods are used by **Cordys** to embed it into their BPM solution to make it fit for manufacturing. Cordys markets the **Easy95/Cordys** BPM solution under their brand: **Cordys Plant Operations Management**.



Easy95 addresses the challenges of:

- Real-time data collection, contextualization and analysis
- Systems integration
- Critical event detection and response

Easy95 produces highly configurable composite applications built on a common, distributable event-driven platform. This provides manufacturers with the ability to easily add and adapt functionality to meet the changing needs of business and unify manufacturing systems. **Easy95** is built on a **Common, Distributable Event-Driven Platform** unified by a **Common Service Architecture**.

Easy95 provides:

- **Open Connectivity** to systems and hardware and we pride ourselves in our communications and connectivity capabilities
- **Ability to Easily Add and Adapt Functionality** to meet the changing needs of plant commissioning, optimization, and unpredictable events of ongoing operations

Easy95 is an application development platform that dramatically simplifies software and system development for manufacturers, system integrators and OEMs/ISVs, allowing them to quickly and easily integrate highly disparate manufacturing systems and address a variety of application needs within the manufacturing environment. **Easy95** combines three powerful concepts into one software product:

- 1) a composite application development framework
- 2) unifying service architecture
- 3) distributed execution capability

The Easy95 Composite Application Development Framework is built on:

1. Unifying Service Architecture

ISA-95 Services form the foundation of **Easy95** application framework and platform. This promotes a service-based approach to application development and integration allowing for the modular development and modular connectivity of functionality. ISA-95 Services formalize and generalize the interactions between functional components of an integrated system. They also help isolate change and allow for the controlled extension or reconfiguration of systems. This dramatically reduces system complexity while increasing maintainability and flexibility. Implementing standardized services provides a common approach to integration between manufacturing operations and enterprise systems. It also addresses the high level of diversity found in manufacturing operations as well as the demands of unique implementations.

2. Distributed Execution Capability

Manufacturing operations generate significant event and data volume, simply moving this information to a database for future analysis is an ineffective strategy. **Easy95** allows for the distribution of execution capability to the source of event and data generation where ISA-95 data models contextualize this information in near real-time at the ODS. Contextualized information is provided through services to manufacturing operations concurrently. It also enables peer-to-peer interaction, creating a much more responsive and intelligent manufacturing operation.

3. Definition of ISA-95-based Services:

- Services are the contractual relationships between elements of a system, a strictly defined interface
- Services take the form of events, commands or request/reply transactions between elements
- Services allow for the modular development and deployment of highly distributed systems
- Standardized services provide a common framework for integration between the plant floor and the enterprise

***Hibernate** is an object-relational mapping (ORM) library for the Java language, providing a framework for mapping an object-oriented domain model to a traditional relational database. Hibernate solves object-relational impedance mismatch problems by replacing direct persistence-related database accesses with high-level object handling functions. Hibernate provides transparent persistence for Plain Old Java Objects (pojos). The only strict requirement for a persistent class is a no-argument constructor, not necessarily *public*. Proper behavior in some applications also requires special attention to the *equals()* and *hashCode()* methods.

Collections of data objects are typically stored in Java collection objects such as Set and List. Java generics, introduced in Java 5, are supported. Hibernate can be configured to lazy load associated collections. Lazy loading is the default as of Hibernate 3.

Related objects can be configured to *cascade* operations from one to the other. For example, a parent such as an Album object can be configured to cascade its save and/or delete operation to its child Track objects. This can reduce development time and ensure referential integrity. A *dirty checking* feature avoids unnecessary database write actions by performing SQL updates only on the modified fields of persistent objects.

Mapping java classes to database tables is accomplished through the configuration of an **XML** file or by using **JAVA ANNOTATIONS**. When using an xml file, hibernate can **GENERATE** skeletal **SOURCE CODE** for the persistence classes. This is unnecessary when annotation is used. Hibernate can use the xml file or the annotation to maintain the **DATABASE SCHEMA**. Facilities to arrange **ONE-TO-MANY** and **MANY-TO-MANY** relationships between classes are provided. In addition to managing association between objects, hibernate can also manage **REFLEXIVE** associations where an object has a one-to-many relationship with other instances of its own **TYPE**. Hibernate supports the mapping of custom value types. This makes the following scenarios possible:

- Overriding the default SQL type that Hibernate chooses when mapping a column to a property.
- Mapping Java Enum to columns as if they were regular properties.
- Mapping a single property to multiple columns.