Chapter XXXVI

Modeling Objects of Industrial Applications

Lenka Landryová
VŠB – Technical University Ostrava, Czech Republic

Marek Babiuch
VŠB – Technical University Ostrava, Czech Republic

ABSTRACT

This chapter presents the development and design of an industrial application based on new technologies and explores the technological dimension of data acquisition, storing, access, and use; the data structure and integration and aggregating values from data necessary for the control of production processes. The demo application presented here was designed according to the Enterprise-Control System Integration Standard (ISA-95) with the goal of maintaining a standard that defines the interface between control functions and other enterprise management and business functions. The component-based object-oriented development concept was implemented in order to utilize advantages provided by creating a complete plant equipment model. The value of semantics was rediscovered in applications, which communicate among their system modules on Simple Object Access Protocol (SOAP), the standard defined by W3C, whose initiative has provided standard semantics markup languages based on XML as well. The data format and the format of standard XML messages that are used in industrial applications are defined by Business To Manufacturing Markup Language (B2MML) as W3C XML Schema for implementation of ISA-95, while offering the framework for project integration, the separation of business processes from manufacturing processes and focusing on functions instead of systems, organizations or individuals themselves.
INTRODUCTION

Each production is organized in hierarchical levels and for its control and management uses control and information systems. Enterprises name their departments, activities and production functions differently. Communicated information varies depending on every company and its implemented control systems. Even within one company the terminology used for control systems and for management may vary, it is, however, very necessary to be able to communicate efficiently.

The progress and development of new technologies enable the information flow and make it more sophisticated and even easier. Interfaces between enterprise and production systems make important information accessible at the requested time and in a proper place to the appointed competent person. This is possible because of the tools, environments and standards introduced into the design and development of processes, which will be further described.

BACKGROUND: THE TECHNOLOGY WHICH IS HERE FOR US TO BE USED

XML Documents and Their Components

XML is the technology available here for us for sharing information easily based on a format of documents designed for reading over the Internet. XML is made for everybody and to be used by everybody and for almost anything by being easy to understand, easy to use, and easy to implement (Ortiz, 2002). This is one of the many reasons why it has become the universal standard and has faced and met the challenges of convincing us - the development and user community.

When the creators of XML were working out their design, their goals set for an XML document, among others factors, were defined by directions as to how XML is to be used:

- XML documents shall be easy to create, legible to read and reasonably clear, it should be easy to write programs that process XML documents,
- XML design should be prepared quickly, it should also be formal and concise,
- XML shall be straightforwardly usable over the Internet,
- XML shall support a variety of applications.

As referred to sources (Ortiz, 2002), XML offers a simple solution to a complex problem, a standard format for structuring data or information in a self-defined document format. This way, the data can be independent of the processes that will consume the data. But this concept behind XML is not new. It is a subset of a huge amount of specifications and conditions declared and developed by the World Wide Web Consortium (W3C) in 1986. The W3C began to develop the standard for XML in 1996. Since then, many software vendors have implemented various features of XML technologies.

An XML document contains a variety of constructs, also referred to as elements. Some of the frequently used ones include:

- Declaration: Each XML document can have the optional entry `<?xml version="1.0"?>`. This standard entry is used to identify the document as an XML document conforming to the W3C recommendation for version 1.0.
- Comment: An XML document can contain HTML-style comments such as `<!-- Equipment data -->`.
- Schema or Document Type Definition (DTD): In certain situations, a schema or DTD might precede the XML document. A schema or DTD contains the rules about
the elements of the document. For example, we can specify a rule like “An equipment element must have EquipmentName, but AliasName is optional.” .NET uses these schemas exclusively.

- **Elements**: An XML document is mostly comprised of elements. An element has a start-tag and an end-tag, for example `Equipment Requirement`. In between the start-tag and end-tag, we include the content of the element. An element might contain a piece of data, or it might contain other elements, such as `EQUIPMENT`. For example:

  ```xml
  <Equipment Requirement>
    <Property List/>
    <Segment Requirement>
      <EQUIPMENT>
        <Property List/>
      </EQUIPMENT>
    </Segment Requirement>
  </Equipment Requirement>
  ```

- **Root element**: In an XML document, one single main element must contain all other elements inside it. This specific element is often called the root element.

- **Attributes**: Since an element can contain other elements or data or both, an element can also contain zero or more so-called attributes. An attribute is just an additional way to attach a piece of data to an element. An attribute is always placed inside the start-tag of an element, and we specify its value using the “name=value” pair protocol. For example:

  ```xml
  <RollMill_Diameter ID="3501" EquipmentName="RM_Diameter" Alias="RM_Diameter" Description="Roll Diameter"/>
  ```

There is a more complete list of XML’s constructs at www.w3c.org/xml.

In an XML document, the data are stored in a hierarchical structure. This hierarchy is also referred to as a data structures tree. That suits very well to the purpose of using it by system clients in our system architectures (Babiuch, 2007).

### Available Tools to Create XML Documents

When XML has emerged as the web standard for representing and sending data over the Internet, the W3C worked out and established a series of standards for XML and related technologies, including XPath, XSL, and XML schemas. VS.NET provides a number of tools to work on XML documents (Viscom .NET Team, 2007):

- **XPath** is a query language for XML documents. XPath queries are executed on data items. Search results are returned as a list of items. Each XPath expression may specify both the location and a pattern to match. Boolean operators, string functions, and/or arithmetic operators can be applied to XPath expressions in order to build quite complex queries against an XML document. Furthermore, XPath provides functions to evaluate numeric expressions such as summations and rounding. The full W3C XPath specification can be found at www.w3.org/TR/xpath (Ortiz, 2002).

- **XSL** translates XML documents from one format to another. The Extensible Stylesheet Language Transformation (XSLT) is the transformation component of the XSL specification by the W3C (www.w3.org/Style/XSL). It is basically a template-based declarative language that can be used to transform an XML document to another XML document or to other types of documents, for example HTML and/or text. Various XSLT templates can be developed and applied to select, filter, and process various parts of an XML document.

- **XML schemas** define the structure and data types of the nodes in an XML document,
such as the 95 Equipment XML Schema, and other schemas developed and available from the (World Batch Forum, 2003).

These technologies are industry standards backed up by the W3C. All of these standards were taken and packaged into the .NET architecture which we are working with now.

Industrial Standard ISA-95 and Models Description

ISA-95 is the international standard for integration of control systems into management systems. It was written for common production environments and can be applied to any industrial area and for continuous, discrete and/or batch type of processes. The main objective was to create a framework for a project integration, to help to separate business processes from production, and to focus on production functions instead of functions of organizations or individuals.

ISA-95 standard divides companies from the control point of view into five levels, Level 0 – Level 4:

- Levels 0, 1, 2 – are process control levels, whose objective is equipment and process control (equipment, tools, units, cells) so that the production process is performed and a product is produced.
- Level 3 – is the MES (Manufacturing Execution System) control level. It consists of several activities, which must be performed for production preparation, monitoring and completion executed at lower levels of a plant, for example detailed planning, scheduling, maintenance and production monitoring.
- Level 4 – is the top control level. Systems at this level are the Enterprise Resource Planning systems (ERP). Logistics and financial issues and activities of a plant, which correlate with production directly or indirectly, are carried over, for example long term planning, marketing, sale, and supply.

Standard ISA-95 consists of five main parts:
- Describes terminology and object models.
- Deals with object attributes.
- Focuses on functionalities and activities of higher control levels.
- Deals with object models and attributes of production operation management.
- Describes transactions between sales and production.

We will describe ISA-95 standard implementation and the use of definitions for the Level 3 – MES. Furthermore, we will focus on object modeling for the component-based development of an application for a rolling mill supervisory control and visualization.

ISA-95 Standard Implementation

ISA-95 standard implementation provides models and information in multiple levels of detail and abstraction. Each model increases the level of detail defined in the previous model, however, they all come from the ISA-95 major object model, which is structured into:

- The basic resources models – such as equipment and material used in a process;
- A model for a product definition and product production rules, such as the business view of production on how to make a product and what is needed to make a product.

Definition of an Object Model

An object model can be described in ISA-95 standard by the relationship of 95.01 and 95.02 parts of the standard:

- ISA-95.01 defines object models
Modeling Objects of Industrial Applications

- ISA-95.02 describes attributes for the object models.

An example of such a definition is demonstrated for models of equipment and equipment class in Figure 1, Figure 2, and Table 1.

Object-Based Modeling and Development of an Industrial Application

Software environments based on .NET platform have brought a new approach to SCADA/HMI.
Table 1. Definition of the equipment attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description of an attribute</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>A unique identification of a piece of equipment within the information exchanged</td>
<td>Motor 101, Rolls WR 2, Coil C1003, Panel 1</td>
</tr>
<tr>
<td>Description</td>
<td>Additional information about the equipment</td>
<td>a power unit ensuring movement of rolls and a coil together with material all rolls (work, intermediate, backup,…) appearing in a rolling mill and their hydraulics produced material through the rolling process represents the control and supervisory panel</td>
</tr>
<tr>
<td>Run</td>
<td>Boolean, direction of rotation, operated by the end-user-supervisor.</td>
<td>Run of a motor</td>
</tr>
<tr>
<td>Diameter</td>
<td>Analogue real value, actual or newly set by a supervisor</td>
<td>Diameter for roll</td>
</tr>
<tr>
<td>Bent</td>
<td>Analogue real value, bent of working rolls in axes against the strip, operated by a supervisor</td>
<td>Work roll bent</td>
</tr>
</tbody>
</table>

(Supervisory Control and Data Acquisition/Human Machine Interface) and MES application development enabling developers the use of development screens and end-users the use of the runtime screens to create and run an application for a complete plant model. The developers are able to concentrate on modeling how the production facility is designed and the end users are dealing with plant-wide supervisory control of processes which comprise the various manufacturing units and cells. After the plant model is ready, supervisory control functions (Landryova, 2004) can be implemented.

Tag- vs. Component-Based Development

A development environment of SCADA/HMI software has functions required for creating animations and touch sensitive objects displayed on runtime application screens (Zolotová, I. 2004). In this development environment the tools are available for introducing logics through common graphics containing scripts, and for monitoring the alarm states of variables and their trends with the help of created tags and their database, see Figure 3. This tag database connects each application screen directly to industrial controllers, I/O systems or other MS Windows applications while switching the tags in runtime. However, each change in the database or change of a tag in the system must be analyzed for the effects on the rest of the application.

In SCADA/HMI applications developed with a component-based concept, the application objects contain attributes or parameters, which are directly associated with equipment they are representing. For example, a work roll or a motor object contains its own event triggers, alarm message and alarm limit definitions, access rights and security attributes, other communications and scripting associated with the equipment. The concept of a component-based object-oriented development enables us to select objects already defined in software or to define our own template for a new object, see Figure 4. Then the object’s properties are shared and used with their attributes in efficient ways releasing the developers from
repetitive programming tasks and, at the same time, making use of parent-child relationships, re-application and change propagation features throughout the development of an application.

Creating Object Templates

As mentioned before, manufacturing operation or a process consists of resources, such as equipment, energy, and material used for production. These can represent the physical model view of a production and can be further developed as individual models, while similar equipment or material item properties can be shared, inherited, and defined into one of their classes.

The shared properties then may be defined for one template, on which the modeled object will be built in a development environment for the application. The model created in the development environment of one computer is then deployed to all computers that will work with the application (Garbrecht, 2006).

Configurable Objects

Each device or a component of the modeled production system is then configurable by the object template it has been derived from. The configuration process complies with the standards set for each supervisory application and for any future application as well. Once the application is developed, system maintenance is easy. Changes made to object templates during configuration are then being propagated to their child objects (Landryova, Osadnik, 2008).

The configurable attributes of an object template are represented with real I/O variables available in the PLC or a control system, linked to the I/O through device-integration objects.

Components of a system, which were modeled in an application, are assigned to groups according to their security rights as well. The application is visualized in the SCADA/HMI software, see Figure 5.
Modeling Objects of Industrial Applications

Figure 4. An object being defined in a component-based development environment

Figure 5. Runtime screens of the supervisory control system application using object model
Event Notifications and Messages on Runtime Screens

Industrial applications collect huge amounts of data, which is required by more and more users at different levels of production or administration (Levine, 1996). Some of these users need effective means that are easy to operate in order to monitor and control processes. Others do not require data continuously, but are interested in data summaries, production results, the quality and history of data. These users only need simple interfaces to access, inspect and analyze the data, such as internet browsers. The internet has changed communication within control systems into an interactive and graphically very powerful tool, which is now closer to visualization systems than that of the human-machine interface category, because it is able to bring data to the end users. Data is transferred between web clients and servers, where the client is the computer requesting documents, and the server is a computer that is able to send them, because it is equipped with specific software. In the case the server sends files written in some of the standard supported formats, like HTML or XML, the client only needs a browser to view them, and is then called a thin client. The web server is able to send data in other formats as well, but the client then needs specific software, in order to be able to read, process and display the data. In such case the client is called a fat client.

Browser-Based Interface

Industries require the capability to handle events and display them to the end-users, supervisors, with the help of control system’s thin-clients. Messages are defined for data being updated in the background of a manufacturing process. But the HTML protocol is request-response based. The thin-client browser must trigger “a new page” request. There is no default mechanism for event notifications without a trigger. So, the behavior of browser-based user interfaces of thin-client applications is that they can not automatically react to external events.

Services provided via web service interface allow industrial applications to access available data from the process. Web services here represent a layer between an industrial application and a thin client. They are the interface to the industrial application (API) developed according to a concept based on standard internet technologies (Avery J. (2004), Babiuch, M. 2004).

Simple Object Access Protocol (SOAP) used for this communication refers to the XML-based, extensible message envelope format enabling message transfer. SOAP consists of an envelope, a set of rule codes, and convention for remote procedure calls (RPC).

Web Service Description Language (WSDL) refers to the XML format allowing service interfaces to be described. WSDL file with service interface definition is an XML document using interface methods and their parameters.

Event Notification Services

Event notification is a type of service, which provides the mechanism to dispatch events to one or several supervisory stations. Events can be generated within the industrial application through:

Business logic (BL) executed in an application server (AS), which provides services such as data consolidation, user authorization, client data access and scheduled report generation. The services are implemented as a set of objects and server processes.

Custom scripting written in computer language, such as C#. The scripting environment provides an application programming interface (API) (Turtschi, Werry, Hack, Albahari, Nandu, Lee, 2003).

By communication with external applications.
The requests for data update can be generated by different event sources:

- Messages are the input for updating requests data, which can be generated upon, for example, a manual data correction by the end-user, supervisor.
- The communication between the different tiers of a system may also be based on the asynchronous publish-subscribe message pattern.
- The messages must include a topic. The receivers subscribe only for the topics they are interested in. The sender then publishes these messages.

Presenting Runtime Information

With the legacy GUI and information technology, meanings and relationships are predefined and written into data formats and the application program code during their development phase. This means that when two programs need to interact in a new way or when something changes during runtime and previously not communicated information needs to be exchanged, the developers are back to their work and must get involved. Then off-line, they must define and communicate the knowledge needed to make the change, recode the data structures and program logic to accommodate it, then apply these changes to the database and the application. And then, the end-users can implement the changes into a process (Lee, 2004).

It is possible now to modify the display presentation of some GUI controls based on runtime information. We are now working in environments and with software, which encodes meanings separately from data and content files, and separately from the application code. This enables end-users, operators to understand the runtime screens, share knowledge and reason experience at execution time. With this type of software, adding, changing and implementing new relationships or interconnecting programs in a different way can be done as simply as changing and deploying the external model that these programs share. This gives the end-users the required flexibility in data manipulation before display in the GUI (Landryova, Valas, & Winkler, 2008).

To see the idea and how it works, simple examples will be presented, such as style modifications of text and numerical messages depending on runtime information presented to end-users and operators on runtime screen panels, and the manual entry fields requiring supervision of operators when entering set points and command, and data format changes made independently from the required database data type.

Scripting for Runtime Screens

Using XPath expressions (previously described in the section Available tools to create XML documents), simple calculations can be made in a runtime software environment. Therefore, the read-only types of properties whose values come from an associated calculation and not from the database are allowed and the calculation is configurable for runtime screens. The property, whose value is calculated, is then associated to an object. The object may, for example, change the color of

![Figure 6. Alarm and event display in the supervisory control system application](image-url)
Figure 7. Information and manual entry fields presented to the end-users on a legacy runtime screen panel

Figure 8. Conditions for displays required by the end-user and the result from XPath implications on displayed values

<Format Condition>
Low pressure $\geq$ 300 display field: red
High pressure < 700 display row: orange

<table>
<thead>
<tr>
<th>Media</th>
<th>Low Pressure</th>
<th>High Pressure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hydraulics</td>
<td>100</td>
<td>600</td>
</tr>
<tr>
<td>Gear</td>
<td>200</td>
<td>800</td>
</tr>
<tr>
<td>Rolling Oil Pumps</td>
<td>300</td>
<td>1000</td>
</tr>
</tbody>
</table>

Some issues coming from such end-user requirements, however, must be dealt with. For example, the value of a property can be set as the result of a calculation by scripting, but it must also be saved in a property database. The calculations are done only when the property’s value is accessed by the end-user. It is also a good idea to set a time-out for all scripting, if a calculation script is hanging in an infinite loop and has to be stopped. The calculation itself may have to throw out an exception, if it cannot be done, so the problem can be logged and presented in order to notify the end-user, and similar cases.

Table 2 describes the idea behind the runtime data manipulation by the end-user, if it is more
Table 2. Flexibility given to the end-user for a comfortable data manipulation during process runtime

<table>
<thead>
<tr>
<th>Describing the runtime manipulation step</th>
<th>Result in the runtime environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>There is a “numeric” data type set in a database for the Priority property of a displayed object:</td>
<td><img src="image1.png" alt="Table 2.1" /></td>
</tr>
<tr>
<td>The scripting code can change the displayed data type from “numeric” to “text”, for example, if the end-user prefers semantic information to figures:</td>
<td><img src="image2.png" alt="Table 2.2" /></td>
</tr>
<tr>
<td>The result after applying scripting for changing data type from “numeric” to “text” is:</td>
<td><img src="image3.png" alt="Table 2.3" /></td>
</tr>
</tbody>
</table>

comfortable to work with a different data type than was defined for a database.

**FUTURE TRENDS**

Industrial automation in process industries has increased its significance in recent years. Process automation serves to enhance product quality, improves functionalities of the whole range of products, improves process safety and plant availability, and efficiently utilizes resources and lowers emissions. The greatest requirement for process automation is the fastest growing demand for hardware, standard software and services of process automation. The traditional barrier between information and automation technology is gone. The latest technologies, including XPath, XSL, and XML schemas, help to improve the communication of process system modules.
Another scripting then allows to decode the text values and save them in a “numeric” data type into a database even if values were previously set in the “text” data type by the end-user:

```java
// define user readable texts
string[] texts = new string[]("net set", "low", "medium", "high", "urgent");
if ((formatter.EncodedValue == null) ||
    String.IsNullOrEmpty(formatter.EncodedValue.ToString()))
{
    formatter.Value = null;
    list
    // set unknown text
    formatter.Value = null;
    for (int i = 1; i <= 4; i++)
    {
        if (formatter.EncodedValue.ToString().ToLower().Equals(texts[i]))
        {
            current.Value = i;
            break;
        }
    }
}
```

Check Syntax... OK

But, if the supervisor changed a value of the Priority property in a manual entry field from “High” to “Low” (on the first row of a runtime display), the Priority value is saved in a database in the “numeric” data type although it was written by the end-user as a text.

Using the development environment of the industrial application server helped us to demonstrate a scalable and flexible supervisory control system process development based on a distributed and object-oriented technology. During the application development we used “roll”, “coil”, “motor” object templates representing the technology, equipment, their I/O references, scripts, alarm state definitions, security and data history implemented within. From objects prepared in such a way other “child” objects could be derived, deployed and repeatedly used within our system to built a hierarchical model of a production control technology. Further development of the application is therefore expected. The equipment model and equipment class model will be extended, models for material (consumed, produced) will be added in order to demonstrate the complexity of a production process and interaction with supervisory functions being modeled based on industrial standards applied.
CONCLUSION

A demo application was designed using new technologies to demonstrate an object-based modeling enabling easier automation of information flow, given the interfaces available for an enterprise and a production system.

The application developers create these new manufacturing execution system solutions (MES solutions) using services for communicating via messages from different sources on the web.

Web services as a new class of applications were developed to serve multiple computer architectures, operating systems, and languages, enabling documents to be readable and to be exposed in a standard way. ISA-95 standard, which was produced in cooperation of ISA and WBF, now provides the tools for implementation. B2MML provides the XML schema implementation of ISA-95. ISA-95 standard is in use today. It is gaining more acceptance driven by operating companies, and provides the framework for further information and process integration. This assures planning for material, equipment and human resources to be ready for production processes controlled by a manufacturing execution system based on new approaches towards collaborative production management.

ACKNOWLEDGMENT

This work was concluded in our laboratories with the software and licenses provided by “Factory Suite Educational Demo Consignment” from Pantek (CS) s.r.o. Furthermore, the developed model and the created application are a part of our research conducted under the project Future Industrial Model for SMEs (Future SME) of FP7-NMP-2007-LARGE-1 supported by EU.

REFERENCES


**KEY TERMS AND DEFINITIONS**

**Application Programming Interface (API):** A set of declarations of the functions (or procedures) that an operating system, library or service provides to support requests made by computer programs.

**Application Server (AS):** Executes applications for end users and controls networking, is usually between a database and a client application presenting data, application logics, controls data saving into a database and provides data to client applications.

**Business Logic (BL):** A term generally used to describe the functional algorithms, which handle information exchange between a database and a user interface. It is distinguished from input/output data validation and product logic.

**Component:** In object-oriented programming and distributed-object technology, a component is a re-usable program building block that can be combined with other components in the same or other computers in a distributed network to form an application. Components can be deployed on different servers within a network and can communicate with each other to perform services. Examples include a single button in a graphical user interface, a small interest calculator and an interface to a database. [Garbrecht, S. D., 2006]

**Data Acquisition (DA):** The detection and collection of data from processes and systems external to the computer system.

**Graphical User Interface (GUI):** The display presentation of some GUI controls based on runtime information, for example, to change the style of data presented when entered into a software and data format changes independent from the required database data type.

**Manufacturing Execution System (MES):** A system that companies use to measure and control critical performance and production activities.

**Service Oriented Architecture (SOA):** The software architecture for production control and information management systems that uses software services independent of the underlying platform and programming language.

**Supervisory Control (SC):** Control where one or more human operators are continually programming and receiving information from a computer that interconnects through sensors to the controlled process or task environment.
**XML document:** A textual representation of hierarchical data structures. XML documents are case sensitive. XML schema specifies the XML structure of the document made up of XML elements.

**XPath:** A query language for XML documents. Search results in these documents are returned as a list of items.