Industrial IT Structures as a Means to Express Context Sensitive Information

Therapon Skotiniotis
College of Computer Science
Northeastern University
Boston, MA. 02115, USA
skotthe@ccs.neu.edu

Alban Frei
ABB Corporate Research
Segeholf, 5405 Baden-Dattwil
Switzerland
alban.frei@ch.abb.com

Karl Lieberherr
College of Computer Science
Northeastern University
Boston, MA, 02115, USA
lieber@ccs.neu.edu

Abstract

Industrial IT, from ABB, is a framework-based solution for integrating and operating many, possibly independent, industrial applications. Industrial IT provides a software architecture (embodied in a platform), along with supported tools, that allow integration both at the hardware and software level. In order to be able to still control and manage an industrial plant, it is essential to model and represent different types of information on the various domain objects through Industrial IT. The concepts of Aspect Objects and Aspects[1] along with their usage inside Structures provide the mechanisms by which one can represent and manipulate context specific information of domain objects inside Industrial IT. In this paper we introduce these concepts and show how they can be used today in Industrial IT applications to model but also operate on multiple views of the same domain objects.

1. Introduction

With the increased use of software to control and operate specialized equipment, organizations that use these types of equipment, have seen the need to integrate and operate a variety of software product lines together. The needs to efficiently monitor, control, and even automate the day-to-day operation of a plant (e.g. an oil drilling plant) become harder to accomplish when equipment requires different specialized software, and knowledge, in order to be operated.

Integrating all of these applications together requires an integration system that must handle many, possibly, unrelated systems and at different integration levels, hardware level integration as well as software level integration. For example, the flow of real-time information from low level devices (pumps, motors etc) to production level systems (maintenance systems, process engineering systems) should be possible.

The integrated system should also be able to handle the representation of different aspects (views) of domain objects. Production level systems have specific views of physically the same data/equipment along with their relationships. For example, a process engineering system views a pump (a domain object) according to whether it can participate in a batch production process and what its throughput can be in terms of gallons per minute. At the same time, a maintenance process views the same pump, according to its current temperature and oil pressure in order to check its current status and decide on possible hardware failures or necessary maintenance operations. Inside a process engineering system the pump will have a relation to other equipment with which it cooperates inside a batch process. In the maintenance system view however the pump will have a relation to the service station or to the closest replacement pump according to its physical location inside the plant.

All these different views of the same system need to coexist and seamlessly interoperate if necessary. Industrial IT provides both the mechanisms to integrate but also allow for these different system views, through one core architecture[5]. Modeling a plant, as well as operating on its different possible views, happens through the same interface providing a uniform operating environment.

Section 2 describes the core features of Industrial IT. In Section 3 Structures and Name Paths are described. Section 4 concludes the paper.

2. Industrial IT Architecture

The Industrial IT initiative was spawned by ABB to facilitate a better means of integrating both ABB's diverse, and independently developed, devices and their software but also third party software. At the same time, customers would like to continue to seamlessly operate and manage the different applications, and their data, through an integrated and stable environment.

[1]Aspect objects, as they are used in Industrial IT, are a special case of Aspects, as they are known in AOSD[3][4]. In AOSD, an aspect is a module whose ad-hoc implementation would cut across several modules. An Industrial IT Aspect Object qualifies as an AOSD Aspect because it is the composition of several objects. Industrial IT supports a simple structural form of aspect-oriented programming where new members may be introduced dynamically at run-time. Aspect-oriented programming languages typically support such introductions at the class level, a feature known as "open classes". Industrial IT supports "open objects".

Industrial IT is a Windows2000 based application. Following a client-server approach, scalability as well as different types of clients is supported. Due to the large size of the architecture and its components (over 1500 defined components) the following sections concentrate on the core ideas and tools found in Industrial IT which facilitate the usage of domain objects under multiple different views.

2.1 Aspect Object Model

In Industrial IT real world entities are viewed as software objects. These software objects are made up, entirely, of pointers to their Aspects. Aspects, as they are used in Industrial IT, should not be confused with Aspects, as they are known in AOSD[3][4]. By Aspects here we refer to a piece of information (a string or even a whole Word document) along with the operations to view and operate on. For example, a pump in a water plant can be represented as an Aspect Object that holds pointers to a documentation aspect, an error log aspect and a replacement availability aspect.

Each of these aspects holds information (data). The data belongs to the system as a whole and not to the aspect itself. Data is stored inside the Aspect Directory, which serves as a Database System to the whole architecture providing transactions mechanisms and data persistency. For example, the documentation aspect found in a pump holds a URL, and a Web Page aspect can use the URL to load a browser and direct it to that URL for easy retrieval of the online documentation for the pump.

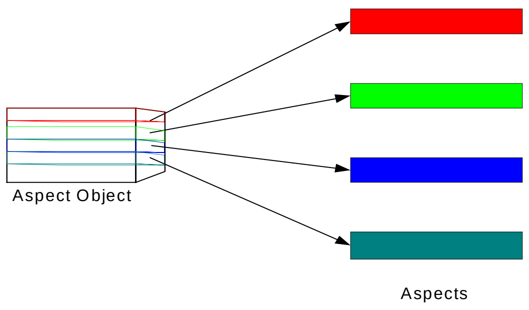

Figure 1 Conceptual View of an Aspect Object and its Aspects

The building blocks inside Industrial IT used to represent a real life object are Aspects and Aspect Objects. The platform also provides logical mechanisms to group and abstract upon the definitions of Aspects and Aspect Objects. Object Types can be viewed as stereotypes for the creation of Aspect Objects, which hold the common Aspects that are to be found in such a type of Aspect Objects.

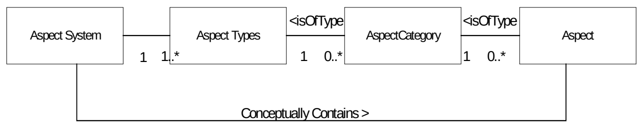

Figure 2 Conceptual relationships between Aspect Systems, Aspect Types, Aspect Categories and Aspects[9]

Aspect Systems are made up of one or more Aspect Types. Aspect Categories are then a sub categorization of Aspect Types, holding the definition of one, or more, Aspects.

The Industrial IT platform is build on top of the Windows 2000 platform and uses COM technology as its component model along with OPC (OLE for Process Control) for communication with hardware components. The platform reuses most of the Windows 2000 security policies, connectivity, installation/deinstallation features and support for multiple screens. Beyond these Window's services the platform also provides

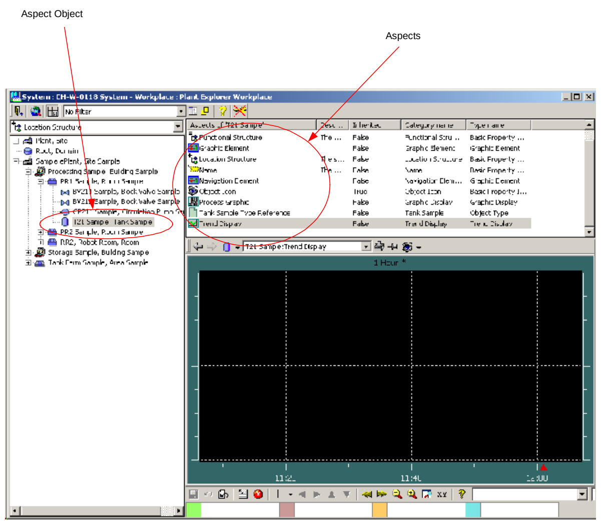

Figure 3 Plant Explorer view with Aspect Objects and Aspects

- Aspect Directory, used to provide basic database functionality and persistent data storage. Data belongs to the Aspect Directory and not to the individual Aspect Object.
- Name and Structure Server, which provides a Naming service to uniquely identify and retrieve Aspects within Structure definitions. These 2 services deal with complex queries of the form "retrieve all pumps found in Building A32"
- Time Service, used to provide time synchronization in a distributed environment.

At the application level, all access to the services and objects are performed as COM calls to a local client server library component, which handles communication with a corresponding component on the server computer. The creation of this library component was necessary in order to facilitate extended error detection and recovery, which is not found in DCOM.

2.2 Industrial IT Tools

At the application level the predominant Industrial IT tool being used is the Plant Explorer (Figure 3). The usage of Plant Explorer resembles the usage of Windows explorer.

Navigation, creation of Aspect Objects/Aspect instances is as simple as the creation of a new Windows Folder making the usability of the tool very easy to new users. Viewing and operating a plant process can be easily done by navigating to Aspect Objects and viewing/operating on data values by simple point and click operations. Plant Explorer can be also configured for and used by a plant's Operators. Operators use Plant Explorer to monitor and perform privileged operations/configurations that cannot be performed by normal users.

At the programming level, one can program directly using C++ along with COM, or using a scripting language (e.g. Visual Basic) along with the Aspect Automation Model. The Aspect Automation Model provides the necessary interfacing between the core platforms services and Visual Basic code. The Aspect Automation Model can also be used to collect metadata about a running Industrial IT system. This facilitates the retrieval of running instances of Aspect Objects, Aspects, Structures, Aspect Categories, and Aspect Types.

3. Structures

Structures inside Industrial IT are a means to represent relationships as well as context specific information about Aspect Objects in a hierarchical manner. For an Aspect Object to exist inside Industrial IT, it has to be at least a member of a structure. Structures have the same feel as tree structures, for example directory structures inside Windows, and can denote a parent-child relationship, or containment relationship between Aspect Objects. Predefined structures can be found inside an Industrial IT installation, however new structures can be easily created.

Organizing Aspect Objects in a structure, the user is free to denote the relationship that he wants to capture. Examples of structures found in Industrial IT are

- The Location Structure that denotes the location relationship between different Aspect Objects. Under Boiler Room you will find a Boiler two pumps and a water tank.
- Batch Structure, which denotes the different steps a batch process, has to go through before completion. Each sub-process is represented as a child inside the structure.
- Maintenance Structure, can denote the physical objects having as their children, possible replacement objects.

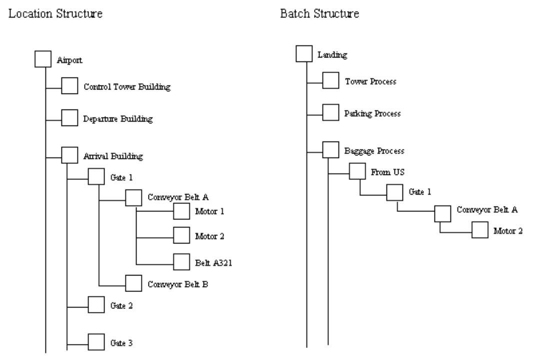

Figure 4 Location and Batch Structures for an Airport

By placing the same instance of an Aspect Object inside different structures allows one to express the different relationships as well as environments within which an Aspect Object participates in.

For example, the motor "Motor 2" can be found inside the location structure but also inside a Batch Structure. Placing the "Motor 2" in the location structure we denote the physical relationship between the motor and conveyor belt A (the motor is part of the rotating conveyor belt). Looking at the Batch Structure, "Motor 2" holds a position that denotes its relationship to the baggage process.

Users can thus concentrate on the structure that best represents the view of the plant on which they are interested in. The maintenance engineer can use the location structure to find out where the replacement motors are located. The process engineer can use the Batch Structure to view and control equipment that is currently involved in a process batch.

At the application level structures can be created so that they reflect a users specific view of the plants operations or structure. The relationships denoted inside the structures are the way by which context is defined for the Aspect Objects that are members of this structure. The important thing to note here is that each Aspect Object has a unique instance and that instance can participate in multiple structures. In this way a single software object can have multiple contexts within which it operates and is being operated on. In this way, context specific information is presented to the systems users.

At the implementation level structures along with the mechanism of Name Paths and the Structure Cursor provide the means to develop programs that manipulate context sensitive information.

Programmatically, one can access Aspect Objects and Aspects by using the provided filtering mechanisms. Filtering allows the programmer to collect, inside *collections*, instances of Aspect Objects or Aspects to which he is interested in. Iterating through the return collection then allows for their manipulation. The filtering mechanism is powerful enough to accept as filtering parameters

- Aspect Types
- Object Types
- Aspect Categories
- Aspects that implement a specific interface

Filtering operations can be combined (denoting a logical 'OR' operation on the filters) so as to create more explicit and complex collections of different Aspect Objects and/or Aspects (e.g. Figure 5.)

```
Dim CategoryAspects As ABBAspects
Set CategoryAspects = System.Aspects
CategoryAspects.OfCategory "Landing Equipment"
Dim TypeAspects as ABBAspects
Set TypeAspects = System.Aspects
TypeAspects.OfType"Guidance Lights"
Dim MyFilteredAspects As ABBAspects
Set MyFilteredAspects = System.Aspects
MyFilteredAspects.IncludeAspects TypeAspects
MyFilteredAspects.IncludeAspects CategoryAspects
```

Figure 5 VB example of filtering Aspects

A very powerful, and useful, feature found inside the Aspect Automation Model is the *Structure Cursor*. One can use a structure cursor to point to specific nodes inside a structure. Amongst all the operations that structure cursor supports are:

- **SwitchStructure(*Structure*)**. The structure cursor is moved to a node, on the same object, found in ***Structure***
- **ParentNode**. Returns the object which is the parent of the currently pointed to object
- **Children**. Returns all objects that are placed as children to the currently pointed to object.
- **Path**. Holds the string that corresponds to the path from the root of the structure to the currently pointed to object

In the code extract in Figure 6 the code first looks at the Batch Structure. Using Name Paths (line 8) we do not have to explicitly hardcode the full path from Landing to Motors of Conveyor Belt A. Instead we can use '*' for the parts of the path that we are not interested in. This action extracts the Motor under Conveyor Belt A (Figure 4). Then we switch to the Location Structure and save the location of this Motor inside the OriginalMotor.

```
1. Dim ChildNumber As Long
2. Dim NumberOfNodes As Long
3. Dim OriginalMotor As String
4. Dim NewMotor As String
5. Dim StrCur1 As ABBStructureCursor

6. ChildNumber = 1
7. Set StrCur1 = System.StructureCursor
8. StrCur1.SetCursor "Landing/*/Conveyor Belt
A/Motor*", "Batch Structure"
9. NumberOfNodes = StrCur1.SwitchStructure("Location
Structure")
10. OriginalMotor = StrCur1.Path
11. NumberOfNodes =
StrCur1.SwitchStructure("Maintenance Structure")
12. StrCur1.CursorDown (ChildNumber)
13. NumberOfNodes =
StrCur1.SwitchStructure("Location Structure")
14. NewMotor = StrCur1.Path
15. MsgBox "Motor at: " & OriginalMotor & " needs to be
replaced with motor found at:" & NewMotor
```

Figure 6 Switching between structures using Structure Cursor

Switching to the Maintenance Structure we then move to the first available replacement motor (line 12). Now we are pointing to a new motor that we have defined as a possible replacement, for the original motor found in the Batch Structure. Switching again to the Location Structure we are able to extract the path to where the new motor can be found. Finally we print out the information in a message box on the user's screen. In this way we are at a position to automatically retrieve and use information in order to automate certain operations within our Industrial IT application.

The fact that the same Aspect Object can be placed in multiple structures is a fundamental concept. Since we are always talking about he same instance of an Aspect Object, switching structures denotes a switch in context. Structures encapsulate relationships between Aspect Objects. These relationships can be user defined, and allow for multiple definitions to coexist within the system. A structure can be also seen as a restricted view of the system from the eyes of the user that defines the structure. Disassociating, in this way, the relationship (structures) as well as the members of the relationship (Aspect Objects), allows for the definition and execution of the same pieces of code(e.g. ParentNode) within different contexts (environments).

3.1 Name Paths

Name Paths provide a flavor of structure shy programming with their ability to define wild cards and operations on Aspect Categories, Aspect Types and Object Types. One can view Industrial IT Name Paths to correspond to DJ Strategies[1]. The mechanisms are the same, although certain facilities may differ. In Industrial IT, Name Paths are used to obtain the targeted objects with no facilities to operate on the objects that are found on their path(s). In DJ however with the usage of a Visitor behavior can be added that attaches to the path between source and targeted objects. Also the mechanism itself differs since in Industrial IT a Name Service provides all information needed to resolve wild cards found in Name Paths. In DJ however, information is extracted both from the Class hierarchy of a program as well as the instances found in the running application.

On the other hand DJ applications work with one global structure upon which they operate; the Class Hierarchy. Industrial IT allows for multiple such structures but also the ability to hop from one structure to another structure and still pointing on the same software object.

Still the 2 approaches allow for a certain level of obliviousness and quantification as defined in [6]. Operations can thus be defined without hardwiring path information on structures. This makes the code more adaptable and easily reusable both within the same application but also between different applications.

In this way you are able to obtain context sensitive information (get parents of all pumps in Process Control) using Adaptive Programming. As a result, you are able to create structure shy programs performing operations on context sensitive information that are found inside Industrial IT structures. The combination of these two methods provide you with a higher level of code reuse over multiple structures, where structures hold context specific information about the same Aspect Objects in an Industrial IT application.

4. Conclusion

Representing a large industrial system's day-to-day operations by integrating multiple applications pauses an issue on how to still support the different views of data and operations that all these different applications provide to their respective users. Industrial IT allows for the definitions of different views by the usage of Structures. Structures allow an encapsulation of the relationships between modeled domain objects. Operations on these structures are allowed through specialized tools to expose context sensitive information. As a result, domain objects can exist in multiple different structures representing relations and information according to a users view of the system.

Through specialized tools provided by Industrial IT, context sensitive information can be obtained and operated on. Programmatically, through the usage of Name Paths and Structures a form of Adaptive Programming allows for reusable and more adaptive programming inside a dynamically changing system.

5. References

[1] Doug Orleans, Karl Lieberherr. "DJ: Dynamic Adaptive Programming in Java". Reflection 2001: Meta-level Architectures and Separation of Crosscutting Concerns. Kyoto, Japan. Springer Verrlag, Sept. 2001

[2] Karl Lieberherr, Doug Orleans, Johan Ovlinger, "Aspect-Oriented Programming with Adaptive Methods", Technical Report NU-CCS-2001-01, College of Computer Science, Northeastern University, Boston, MA, February 2001

[3] Tzilla Elrad, Mehmet Aksit, Gregor Kiczales, Karl Lieberherr. "Discussing Aspects of AOP". Communications of the ACM.Vol.44, Issue 10, pp33-38, 2001.

[4] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Videira Lopes, Jean-Marc Loingtier, John Irwin. "Aspect Oriented Programming". European Conference on Object Oriented Programming. Finland, Springer Verlag LNCS 1241, June 1997

[5] L. G. Bratthall, R. Geest, H. Hofmann, E. Jellum, Z. Korendo, R. Martinez, M. Orkisz, C. Zeidler, J. S. Andersson. "Integrating Hundred's of Products through One Architecture – The Industrial IT Architecture". International Conference on Software Engineering. Orlando, Florida, May 19-25,2002.

[6] R. Filman, D. Friedman. "Aspect-Oriented Programming is Quantification and Obliviousness". Workshop on Advanced Separation of Concerns, OOPSLA 2000, Minneapolis, October 2000.

[7] T.Pauly. "Architecture Description: Aspect Object Architecture Overview", ABB. URL:http://www.ccs.neu.edu/skotthe/com/papers.html.

[8] ABB Corporate Research. “Aspect Integrator Platform: Integration Guideline”, ABB. URL:http://www.ccs.neu.edu/skotthe/com/papers.html

[9] Otto Preiss, Martin Naedle. “Architecture Support for Reuse: A Case Study in Industrial Automation”, Chapter 17 in I. Crnkovic, M. Larsson [Editors]. “Building Reliable Component Based Systems, Artech House. 2002.