Applications of Aspects of BPMS

Bob Urry

Introduction

Processes are complex. Any process can be described simply in a conceptual model. However, as the process is refined and detail added, one complicated thing after another threads itself through the process, like entangling ivy on a tree, hiding what was once an elegant and simple creation. This tangling occurs in programs and applications too, but a mechanism, Aspect Oriented Programming (AOP), has been devised that can manage such complexity. AOP allows for dynamic inclusion of function or different *aspects* to a program. This approach can be applied to processes as described by BPEL to help manage such complexity, freeing designers to concentrate on one *concern* at a time, without interference from other issues.

Take a simple do it yourself job to fix a shelf. So you have the shelf, but you need to get something to fix it to the wall. In the hardware store, there are dozens of different fixing products from nails to screws to special fixings for dry walls, brick walls, stone walls, and wooden walls. Suddenly, the complex world of fixings opens up when the basic problem seemed so simple. There is a need to get special help in such situations where someone with the right knowledge can help. You just want to put up a shelf (your process), not become a carpenter. The store employees understand exactly which products to use for the job. Owners of hardware stores understand this need and ensure that they employ people with the right knowledge.

Aspect Weaving in BPMS

The approach is to introduce Aspect Weaving into Business Process Management Systems (BPMS) to modify dynamically how a process runs. This has been investigated by Corbis and Finkelstein[i,ii] with a view to adding new features in the context of changing requirements. Their papers advocate operating at three levels – within the semantic analyzers, within an adaptable business process engine, and within an extensible business process. There is an overlap of approach between those authors and the approach we propose, although the mechanism is different. This difference is the provision of a single (but complex) variable that can be attached to the process. It is primarily this mechanism that dictates which Aspect should be adopted.

The underlying approach behind Aspect Oriented Programming can be adapted to describe Aspects for processes. The following could be considered a definition of processes and a process system providing Aspect Orientation:

> The *aspect* is the process functionality that can be selected, based on the *advice* identified in the process profile, and held in the system configuration. The Aspect is applied through the *join points* at the point of process invocation. Multiple *join points* or *point cuts* can be achieved by invoking multiple serial or parallel processes where normally one would be expected.

AOP

Aspect Oriented Programming (AOP) is complementary to Object Oriented Programming. It was developed to allow a program to be broken down into distinct parts in which functionality is separated as much as possible, essentially in loose coupling. The separated functions are the different *concerns*, and they interact with the main program by crosscutting the program flow at specific points. The point at which an Aspect cross-cuts a program is known as a *join point*. Multiple *join points* are often termed *point cuts.* By intersecting with the program this way, the original behavior at a join point can be modified or extended. The information that describes the modification is called *advice.* It is a way of untangling ordinarily tangled code (such as logging or exception handling), and allows the addition of new functionality to a completed application without necessarily rebuilding the entire application.

Inheritance

The object and its classification are central to the definition of Object Oriented Programming. Object classification is managed through inheritance. Like a tree that has a root object, objects are classified by child objects taking on the characteristics of their parent object and then adding something of their own. So a parent object that represents all cars could have child objects representing saloons, all-terrain, compact, and so on. Inheritance gives a way to add to objects so something more specialized is created.

Weaving

In this discussion, an important concept of AOP is that of *weaving*. The originators of the AOP concept (Gregor Kiezales and his team at Xerox PARC) listed the following as possibilities for weaving:

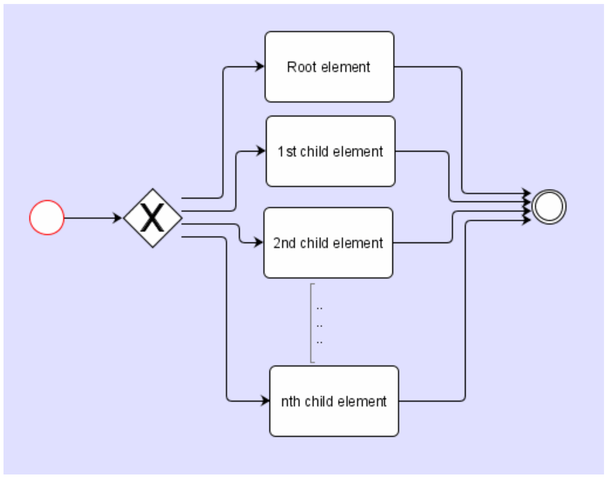
- A source pre-processor
- A post-processor for patching binary files
- An ADP-aware compiler that generates woven binary files
- Load-time weaving
- Runtime weaving (trap each join point at runtime and execute all relevant advice)

To add Aspects to processes, a variable is used to contain a profile. The profile consists of information about the process owner or invoker, the role or roles involved, and any other information that can be used to identify *advice*. There are two ways of using such a profile. The first is to select from a range of partner links or sub processes that represent different Aspects. The *join point* occurs at the invocation point inside the process engine from which the subprocess is called. The second mechanism is to use the profile to introduce inheritance to processes and their activities. When launching the process, the required level of inheritance is applied and the process "constructed," based on the appropriate advice. These mechanisms can be used so that the *selection* works as process specialization, and *inheritance* as process customization. Specialization is like the trend toward organic produce. These special products are laid out in clearly defined areas. Organic foods cover more than one range of produce too. You have several different organic areas, one for each range of produce – organic bread, organic meat, organic vegetables, and so on. Customization is like having special needs: If you have food intolerance to dairy products you look for foods without them. Perhaps you prefer dark chocolate to milk chocolate.

Benefits of using Aspects in BPMS

- Rapid development, agility, and flexibility of process
- Specialization of process into services of different concerns
- Management of the different Aspects of a BPMS (security, users, BAM, etc) by provision of different advice
- Process clarification through separation of concerns (each process is simpler when not complicated by unrelated concerns)
- Loose coupling of different process Aspects
- Addition or change of function to one process with no, or minimal, change to other processes
- Process customization
- The ability to use inheritance in processes by sub-classing processes and activities
- Greater re-use as process patterns will be more readily identified
- Simpler to convert high level process design to implementation (faster design to implementation time, less need for highly skilled practitioners)
- Provide a means to run special processes (simulation, debugging, etc) transparently
- Better control of Business Processes
- Makes it harder to bypass security measures
- Managed data access

These mechanisms can be generally synthesized by the BPMN model below where the choice of mechanism can be *inheritance* or *selection*. However, while this simulation could be implemented directly into a process, it is obviously too complicated and fails to provide one vital benefit of hiding the profile from the process. The ability to hide the profile from the process can only be done in the BPMS, as part of the process engine.



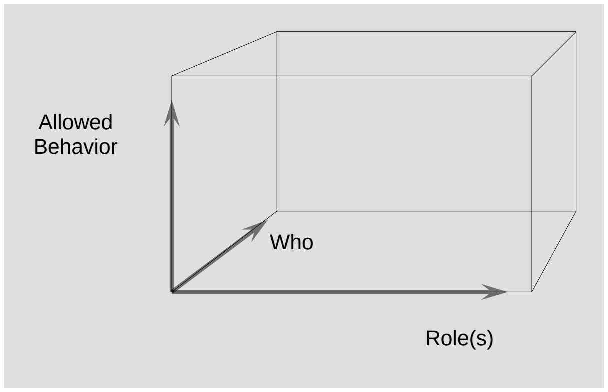
By having the BPMS apply the profile to a process it is possible to apply strict control over which Aspect is selected. It provides a tamper proof mechanism, inaccessible either from within the process or from any other process. In principle, it would even be possible to import a process from a different system. Such a process would have a profile, attached locally in the receiving system. The system identifies it as external and identifies from whom it originated. This would allow process Aspects to be utilized that prevent inappropriate action or access to sensitive or personal information. Where a firewall can be viewed as a single barrier to penetrating a system, the use of Aspects can provide protection at every point in the system, not just at the periphery. This provides a potentially strong security model.

The use of Aspects in the process design cycle need not add a significant overhead. The BPMS selects specialized and appropriately customized processes, determined by a set of selection rules and templates. The design, implementation, and running of processes with Aspects means there is a need to create more processes. However, the outcome of using Aspects is that processes are more likely to be reused and process changes are restricted to fewer processes. The role of the designer is to provide a set of alternative runtime process options from which to select. The system administrator can configure the system by associating each process Aspect to particular roles. The profile need not contain just role information. It can hold other information that advises on the purpose of the process and help choose the appropriate Aspect.

The Profile

The profile contains two types of information. The process or activity position within the inheritance hierarchy needs to be provided. So does its relationship with an Aspect definition, and the role and other parameters that identify what Aspect should be applied (depending on the configured rules). This gives two levels of profile; the first relates to the structure based on inheritance. This should be known at design time and so will be part of a process Aspect template. The other part is only known at runtime. This is primarily the identity, which can be assigned through some identity management facility.

Identity is more than just a role. Andrew Baldwin, of CSC's e4 and BPM Center of Excellence, has shown that identity can be considered to operate on three axes:



- **Who** – Identifies a person or an autonomous agent that changes infrequently and could be
    - Name, biometric, token, password, etc.
- **Role(s) – Have** three views:
    - *Available* – all that a person can be
    - *Allowed* – the subset of the available roles that can be performed now
    - *Active* – the subset of what is allowed that is currently being performed
- **Behavior – What** the active role can do at this point in the process fragment
    - E.g. – read/write access when creating an expense claim; read-only once it has been submitted

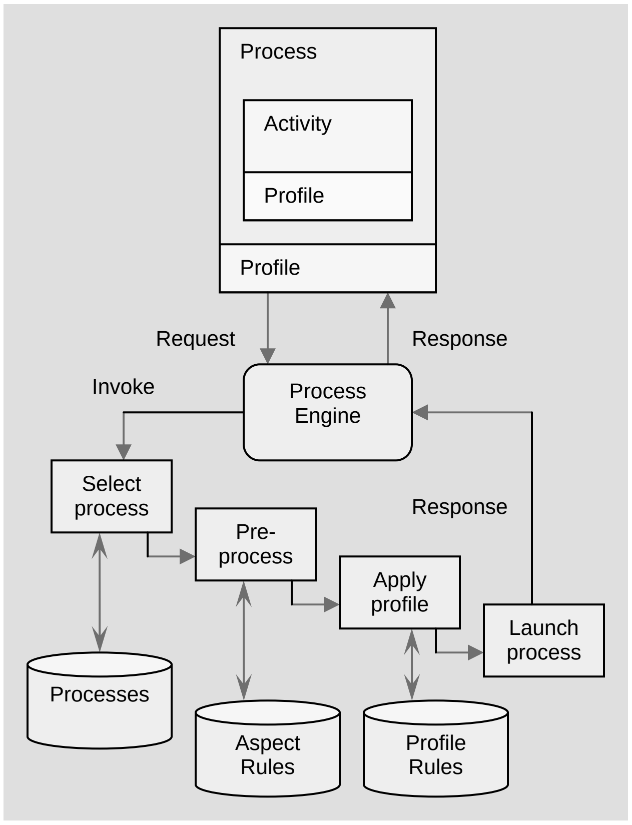
This identity changes over time in response to process context. For example, the initial identity is of an employee submitting an expense claim. The identity changes in the process activity of approval to a designated manager. This is defined by using a separate swimlane in BPMN notation.

The profile would not necessarily hold all this information. A simple identifier is used and a code to indicate the point in the process. The rest of the information is configurable inside the BPMS. It may consist of some or all of the above identity elements. The identity is the primary source of advice when selecting Aspects, but need not be limited to this alone. It could be preconfigured or modified through external events. "You can't authorize a hosepipe ban unless the there is a shortage of water." In this way, Aspect selection can be dynamic. This is part of the process context. The process and any information that it contains can be assessed for validity and accuracy, and is assured by the Aspect components.

With Aspects, the possibility of processes customized for an individual or for groups of individuals becomes possible. In many ways, added functionality, because a

person takes on more responsibilities in a job, can be viewed as a customization of their processes. Processes could also be modified by individuals through the provision of personalized options.

The selection mechanism in the process engine is structured as follows:

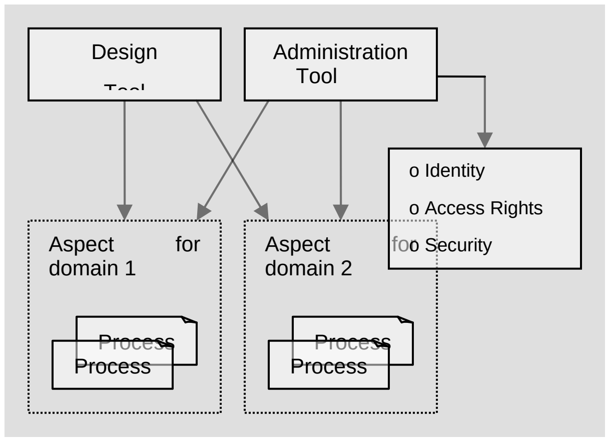


Process Patterns

Now that processes can have their Aspects separated, there is a need to capitalize on this mechanism. Patterns are a good way to manage specific functionality and one that is especially useful is the Model-View-Control (MVC) model. This pattern is used extensively in programming to provide a means of separating the user interface, the data, and the controlling methods. This is one pattern that can be applied to processes. By addition of Aspects, certain processes can represent or relate to the *Model*, others to the *View*, and, finally, the rest as the *Control*. The *Control* processes should become a purer representation of the Business Process. This frees them from other implementation related *concerns*.

Taking this further, Aspects can be used to partition functions so that domain experts can take responsibility for processes that are in their domain of specialty. The design and administration tools should be adapted to cater for domain experts.

As these domain experts are likely to be the Business Process owners, they will be able to provide more direct and complete input to the design of the processes. They gain the ability to deploy their processes without needing special knowledge. Such information could be about the user interface or the processes in the service layer. This shortens development time scales and risk. *Model* processes integrate well with a Service Oriented Architecture (SOA) layer.



Aspect Types

There are many potential Aspects; some BPM vendors provide specialized BPM applications for specific Aspects that BPM addresses. These different applications are good at the things that they specialize in. They have developed collaboration, Business Process Monitoring, and Workflow products, for instance. Aspects can take a specialty and apply its features as an Aspect. Consider:

- Collaboration
- Security
- Logging
- Legislation and compliance
- Data access
- Business process and activity monitoring
- Workflow
- Risk management
- Strategy
- Choreography and orchestration
- IT Services

A selection of such Aspects can be added as required at *point cuts* retrospectively, temporarily or dynamically, as debug and simulation Aspects can also be added. There is no restriction on how many Aspects can be applied at one time. Going back to the shopping analogy, while you're in the shop looking for the fixings for a shelf, the shop assistant can check that you have the right tools for the job with advice on how to go about doing it too. Perhaps a market could develop where companies sell different aspects. Perhaps they could be provided as managed services.

The beauty of debugging or simulation using Aspects is that it can be done safely in a live system. An Aspect process in the model layer can have an Aspect that simulates the original Aspect process. By indicating in the profile that a process is to be simulated, simulation Aspect processes will be invoked. This means test or demonstration information can be used, not live data, thus providing greater confidence in the process under test so that it can be run safely in a live environment.

Aspect Notation

The requirements of a notation for Aspects should not be onerous as a large part of Aspects is in the management of its configuration. Each Aspect would be designed as a simple process. Inheritance would be added using a property of the process or activity and altered or viewed in the property editor. A notation to graphically represent inheritance could be useful. Access to process libraries and process components would also need to be provided. These would link straight into the design and configuration tools. The notation should differ as little as possible from that currently available in BPMN.

Conclusion

The application of Aspects to a Business Process Management System architecture assists in several areas of the development and management of a BPMS. The loose coupling of processes gives the advantage of minimizing pervasive changes to a BPMS because the functionality from different concerns is kept separate. Also, loose coupling is the key to process reuse. The ownership and design of processes can be given to domain experts. They will be able to create and deploy their processes without the need of specialist knowledge of the other domains. Functionality can be dynamically added for various purposes. Things like simulation, testing, logging, auditing, and so on could be typical candidates. Few changes to BPEL are required to support Aspects. By carefully constructing the Aspect mechanisms, a strong security model can be applied. This model can be distributed across the whole BPMS. The strength of the security model is such that it is conceivable that processes from remote systems could be introduced to the BPMS. Aspects would ensure that they can be run in the safe knowledge that no harm will be done. With Aspects, it is easy to ensure that it will not extract information to which it is not entitled.

Bob Urry can be reached at burry2@csc.com

[i] Web Service Orchestration + Aspects = Adaptive Business Process; Carine Courbis & Antony Finkelstein
[ii] Towards Aspect Weaving Applications; Carine Courbis & Antony Finkelstein