

The SOA Magazine

Feature Article



Business Rules in SOA: Decision Services and the Centralization of Rules Management

by James Taylor

Published: October 23, 2006 (SOA Magazine Issue II: November/December 2006, Copyright © 2006)

[Download this article as a PDF document.](#)

Abstract: One of the major benefits of a service-oriented architecture (SOA) is that it can elevate an enterprise to a state of increased agility. That is, an increased ability to respond to change in the business environment or to changing business requirements. Many elements of an SOA can contribute to the potential of this benefit, but it requires close attention to the definition of the technology architecture, as well as its supporting infrastructure, to ensure that the required level of agility is ultimately attained. This article explores aspects of SOA that help deliver agility with an emphasis on business rules centralization.

Introduction (SOA and Agility)

Business agility is typically defined as some variant of the ability to detect a need to change, analyze the possible responses to this change, decide on an appropriate change, communicate this change and then act to realize the change (Figure 1). Service-oriented architecture can support this process by allowing for change to be more easily made, largely thanks to a reduction in the time, cost and difficulty of fulfilling new business requirements through the composition of existing services.

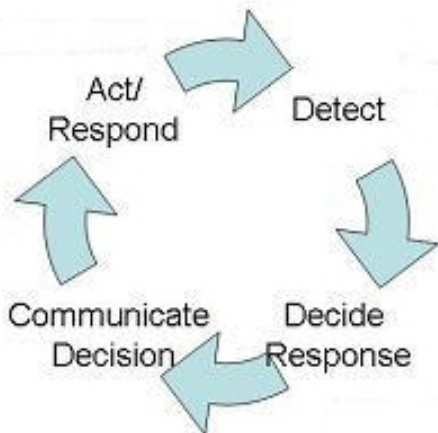


Figure 1: The elements of agility.

The definition of functionality as coherent components or services with well defined interfaces helps limit the impact of a change to a single service, making change easier to control and execute. Well defined services are loosely coupled – they use service contracts to allow services to interact with each other without having to form tight dependencies. Such services can be changed independently and, provided the interface to the service is not affected, this change should not impact other service consumers.

Service-orientation contrasts with the typical experience of changing monolithic applications where a change is likely to cause a ripple effect throughout the application stack. Processes associated with service-orientation support a much more iterative approach to defining these services thanks to the increased level of service independence (which, in

fact, can also lead to a more agile development approach once a sufficiently populated service inventory is established).

Business Services

When *business* services are defined in this way, one can decouple the business from the automation of the business. Business services are defined independent of a particular process usage – they ideally perform a business function that can be used in many processes. This allows new, composite applications and new business processes to reuse these business services. In this type of environment, a new business process can be rapidly automated through the assembly and orchestration of existing business services. This is particularly true of entity-centric business services where the functionality of the service is associated with a defined entity or information set, such as Customer or Account.

“Because the context of the entity-centric business service is agnostic to any one business process, they achieve “process logic independence” and therefore become highly reusable. And, because they represent a well-defined set of logic and data, they establish a level of abstraction and governance over a distinct business domain. This can significantly increase the agility with which business processes that rely on the composition of services can be altered in response to change.” [REF-3]

Additionally, the use of an enterprise service bus (ESB) can further enhance the level of attainable agility by providing an integration layer that allows services on different platforms (and perhaps even with different interface semantics) to be connected and composed. By making it relatively easy to add new services and broker inter-service message exchanges, an ESB can build upon the level of agility established by the service-level application of service-orientation principles.

“If an organization abstracts its IT infrastructure so that it presents its functionality in the form of coarse-grained services that offer clear business value, then consumers of those services ... can access those services independent of the underlying technology issues that support them” [REF-2]

Business Logic and Change

We’ve established that an SOA can deliver agility, especially when a service-oriented approach is taken to the definition of business services. Let’s take a closer look as to the nature of change business services are subjected to in the real world by starting with Dave Linthicum’s simple model for the value of agility, which takes into consideration:

- The degree of change over time.
- The ability to adapt to change.
- The relative value of change.

Clearly, some services will implement a business capability that must change more often and must have a greater ability to adapt to change than others. Some services even provide greater value after the have changed. Either way, there will be services that implement business capabilities that are in constant flux.

For example:

- A service that generates cross-sell offers, for instance, might change daily or even hourly.
- A service might need to allow each store manager to set different pricing rules based on their own observation of customers.
- A service for assessing the current status of a client might have to be changed after weekly customer service team meetings.

Then there are the services that need to implement potentially complex business logic. While this logic might be fairly stable, change might still be required periodically. Given the nature of complex rule logic, the ability of the service to be adaptable – to be changed readily – will be crucial to the agility of the business as a whole.

The change to which the business must respond might be something as fundamental as a pricing algorithm for a product. Ensuring that the impacted service can be appropriately modified, tested and updated without a lengthy

development cycle will ensure an acceptable level of agility.

Similarly, some services may implement simple but voluminous logic – the rules for managing Medicare payments, for example. Here the sheer scale of the problem can hinder agility as finding the right part of a service to change can be difficult.

Furthermore, there may be services that will need to implement business capabilities that are not easy for programmers to understand. A service that, for instance, determines if a proposed drug reacts with medication already being taken by a patient could be coded by someone who has no medical qualifications. This business logic would be far better (and far safer) if someone who actually understood the drugs involved could participate in its definition in a hands-on manner.

Finally, there is the issue of compliance. How does a service remain agile while also having to comply with regulations or policies that require it to communicate how it did something, and why? Ensuring that the way a service operates can be demonstrated to be compliant can require developing it in ways that make it harder to change. If I have a service that must be compliant and yet also changes, then I could face some significant challenges.

In all these cases we have the need for services to accommodate change more effectively than programs written and compiled using traditional development languages. No matter how talented a development team is and no matter how agile a development approach may be, services that implement lots of rules (especially complex rules) require business expertise during their initial design and subsequent maintenance. Otherwise, they will be wrongly evolved, compromising their ability to support to the overall business agility requirements. How then can this be addressed?

Business Rules Management 101

Let's start with some definitions . Business rules are abstractions of the policies and practices of a business organization. The *business rules approach* is a development methodology where rules are in a form that is used by - but not embedded in - services. These business rules are in a language that business and IT can both understand.

Business rules or *business rulesets* describe the operations, definitions and constraints that apply to an organization in achieving its goals. For example, a business rule might state that “no credit check is to be performed on return customers.” Another rule might define a tenant in terms of solvency or listing preferred suppliers and supply schedules.

Rules are essentially used by the organization to conduct its business. They can be abstracted into a distinct architectural layer that can be managed by a *business rule management system*. These types of platforms often position a special *decision service* (or a *decision-centric business service*) that is responsible for carrying a callable “decisioning” process as a service.

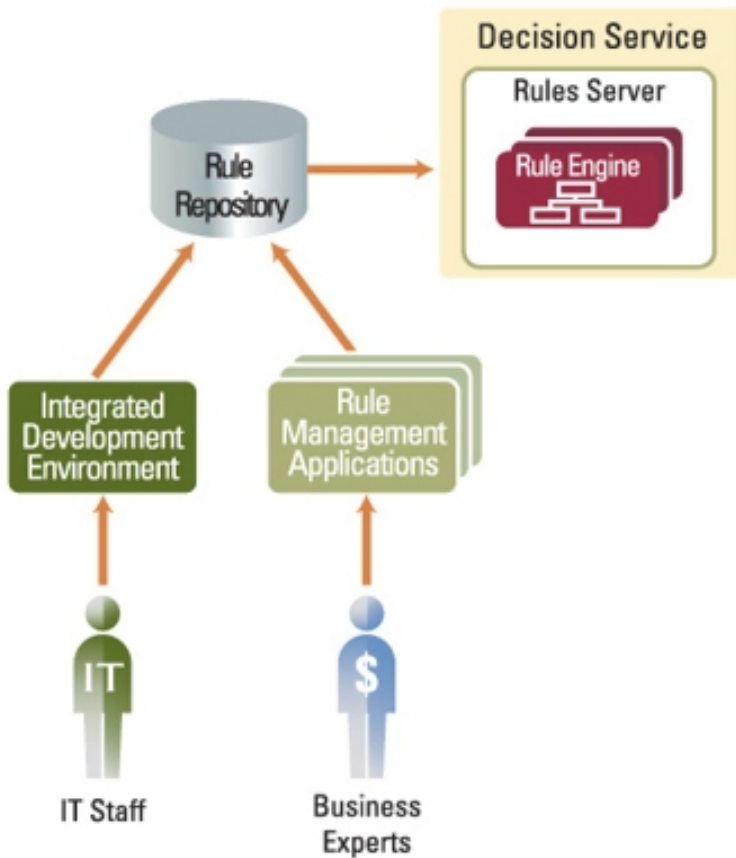


Figure 2: Common architectural components required to manage business rules.

As illustrated in Figure 2, IT staff and business subject matter experts collaborate in the maintenance of the rules that are kept in a central repository. How the rules are utilized in support of other services is determined by the decision service which may encapsulate the processing capabilities of the underlying rules engine.

The Decision Service Model

There are generally two ways to position business rules management within an SOA. These approaches can be carried out independently or combined.

The first is to develop specific decision services within the business or utility services layer. These are services that can be positioned as generic or business-specific resources that deliver an answer to a specific question, but generally do not update information or make any permanent change to the organization's state.

The second approach is to define standard entity and task services within the business layer, each representing its own business-centric behavior. Services can independently interact with the business rules management platform as needed.

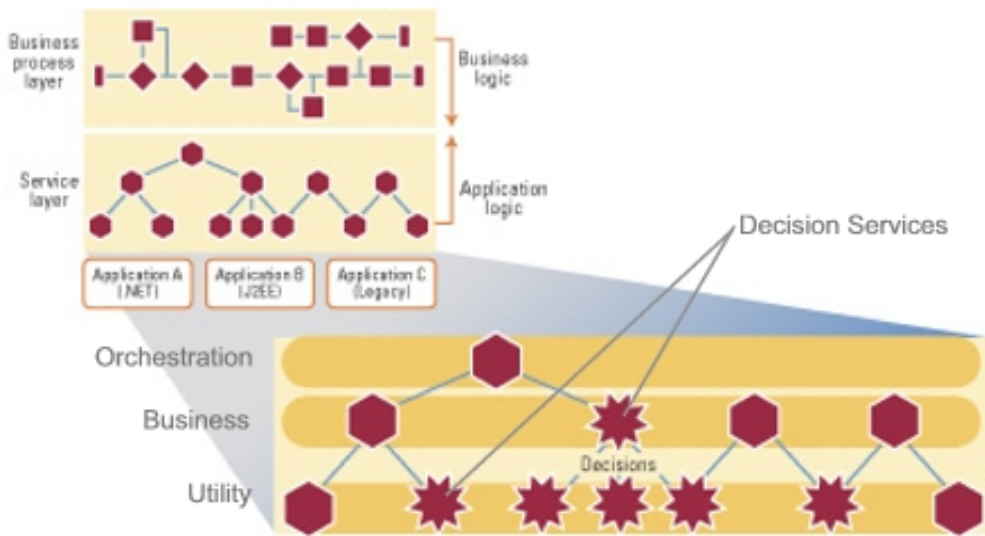


Figure 3: Decision services can be positioned to establish a subset of business services or as standard cross-cutting utility services.

The key benefit of these two approaches is that services, like the composite applications and processes of which they are part, must be “built to change.” In other words, services have to operate in the real world, where nothing can be taken for granted and nothing is set in stone. By centralizing and abstracting business rules through a central decision service other services are capable of sustaining frequent change while remaining policy-compliant and still easily manageable.

The decision service can access enterprise resources and other services (using the same SOA infrastructure) and can also expose various business-level interfaces for other services to use. With a supporting business rules platform, business logic that is more susceptible to change can be cleanly abstracted, allowing this logic to be more easily modified and shared across both services and non-service enabled applications elsewhere in the enterprise.

When this approach is followed, it becomes evident that decision services end up encapsulating the type of logic that truly differentiates an organization and how it carries out its business. This is because business rules define how an organization decides to act and respond within standard business process frameworks. The fact that other services are alleviated from having to encapsulate business rules allows for the creating of more generic and agnostic services with high reuse potential. These agnostic services can then be composed together with decision services to still fulfill unique, competitive, and differentiated business automation requirements.

Business Rules and Orchestration

Besides developing decision services with business rules, there are some other potential uses for business rules within SOA. When we take a closer look at an orchestration that stitches together various services in order to automate a larger business process, we can see that it is the business rules that dictate the logic and flow behind the workflow and routing.

This is an important observation when designing an architecture in which both business rule and orchestration logic can be separately abstracted and centrally positioned. Some business rules (such those related to security policies or rules associated with the execution of business exception routines) may be more effectively implemented by embedding a business rules engine within the SOA fabric rather than by externalizing business rules into a separate rules management product.

Lastly, orchestrated business processes may introduce rules-related transformation logic. The responsibility of carrying out this form of processing can also be deferred to a rules management system. However, in general, these transformations should probably be encapsulated by (or even defined as separate) business services.

Recap

Figure 4 revisits the elements of agility we established earlier but further highlights how the management of business rules can aid in all the areas of agility we identified. This makes it easier to detect the need for change by helping the business understand what happened. It further helps IT and business communities collaborate on the maintenance and governance of organizational business intelligence.

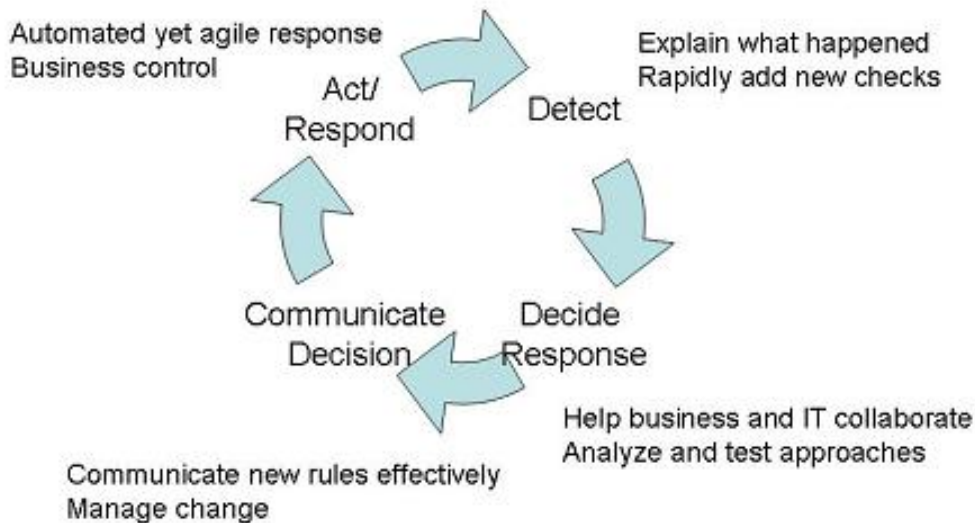


Figure 4: How business rules aid in the identified elements of agility.

By centrally managing business rules we are able to:

- respond to change more efficiently because all business rules are located in one repository
- better control and respond to policy decisions that are prone to change by keeping them separate from processes and focusing on their management individually
- automate complex decisions using technology designed for that purpose, reducing the number of manual steps in a process and increasing the percentage of transactions that can go “straight through” the process
- give business experts control over decision and process behavior by separating out the process changes and decision changes, making it easier to identify the right group of users to participate in each kind of change
- establish and reuse decision services across different applications and service compositions

Conclusion

By applying the fundamental service-orientation principle of abstraction to the domain of business rules we are able to establish a distinct service layer represented by the decision service model. This presents us with a unique opportunity to further increase the level of attainable business agility as part of an SOA implementation.

References

- [REF-1] "Business analysis and SOA Part 1 of 6: The Benefits of Business Services", Thomas Erl; SearchWebServices.com; February 2, 2006 available at http://searchwebservices.techtarget.com/tip/1,289483,sid26_gci1168438,00.html
- [REF-2] "Principles of SOA", Jason Bloomberg; Application Development Magazine; February 28, 2003 available at <http://www.adtmag.com/article.aspx?id=7380&page=>
- [REF-3] "Business analysis and SOA Part 2 of 6: Business Service Models and the Entity-Centric Business Service", Thomas Erl; SearchWebServices.com; March 22, 2006 available at http://searchwebservices.techtarget.com/tip/1,289483,sid26_gci1174702,00.html
- [REF-4] <http://www.serviceorientation.org>

