

The SOA Magazine

Feature Article



Smart Enough for SOA: Incorporating Enterprise Decision Management into Service Design

by James Taylor

Published: November 6, 2007 (SOA Magazine Issue XII: November 2007, Copyright © 2007)

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

Abstract: Enterprise decision management is a design approach that advocates the centralization and dynamic application of business rules for the creation of "smart enough systems" comprised of intelligent decisions services. This article includes excerpts from the new book "Smart (Enough) Systems: How to Deliver Competitive Advantage by Automating Hidden Decisions" [REF-1] that introduce the concepts and mechanics behind smart enough systems and further explore how the decision service model can be architecturally positioned to abstract business rules in support of a larger service inventory. Also provided are guidelines for choosing the right type of decision logic to abstract.

Introduction: What is a Smart Enough System?

What kind of system can deliver a vision of operational decisions? It would take one smart enough to abstract and centrally manage the complexities behind those decisions. A smart enough system is not some kind of artificial intelligence device like HAL 9000 (from 2001: A Space Odyssey). Equally, a smart enough system can't be developed the same way you build traditional "dumb" information systems. Building smart enough systems means taking a new approach to bringing automation to operational decisions.

Instead of hard-coding decision rules into systems, it means using separate tools to build, manage, and carry out decisions in concert with other operating processes. It means developing new services that can deliver operational decisions that perform well enough to be used in real-time front-line systems and processes. It also means developing services that are agile enough to keep up with a changing world and, indeed, learn from it. It means services that make customer-centered decisions and services that can support an extended enterprise.

Besides being customer-centered and compliant and supportive of an increasingly extended enterprise, smart enough systems have the following key characteristics:

Operational

The increasingly distributed and always-on nature of organizations puts a premium on high-performance execution, which means operating quickly, flawlessly, legally, and profitably at every level. It's about more than how your employees perform; it's about how your systems perform. For organizations to exhibit high-performance execution in day-to-day operations, their top performers' expert judgment must be made available everywhere. This means making sure the systems everyone uses embody that expertise - not in an expert system, ask-if-you-get-stuck way, but with systems that are embedded in the operational processes necessary to the business.

Capable of Real-Time Performance

Smart enough systems must operate in a no-wait, multi-channel world where customers expect responses, actions, and decisions immediately. Suppliers expect immediate updates on the demand chain, and retailers and distributors expect to know about problems in the supply chain instantly. Real-time connections between organizations are also

essential, because organizations must become more loosely coupled. They must deliver their products and services by coordinating and orchestrating many distinct organizations, both internal and external. In the past it was sufficient to coordinate operations within an enterprise, but today successful organizations must be able to operate with both known and unknown entities without delays. Systems can't wait for someone to wake up before acting, and people want to be told what has been done to make their life easier, not asked for decisions. Smart enough systems must make decisions fast enough to be used in operational, real-time environments.

Agile

An agile organization can effectively change the way it operates when it needs to, but only if it has a good understanding of how it's operating and why it operates that way. Smart enough systems support this agility by making how they operate explicit, easy to understand, and easy to modify. Agility is a measurement of the total time and cost in getting from having the data that means you should change your business to actually making the change.

Capable of Learning

Smart enough systems need to "learn" as new data is collected. Organizations collect an enormous quantity of data, and the volume of data is increasing steadily. Generally, organizations don't have systems that are smart enough to take advantage of this data. You have to do more than collect, organize, and report on your data. You, and your systems, have to learn from it, mine it for insights, and interpret what it means for the future. Doing so is a key to taking advantage of one of your last remaining areas of competitive advantage - knowledge of your customers - and is a way to improve performance by insisting on realism. If your business decisions are based on what your data tells you, you're more likely to get realism than if you rely on hunches and how you have always done something.

EDM and SOA

EDM or Enterprise Decision Management is a systematic approach to automating and improving operational business decisions. It aims to increase the precision, consistency and agility of these decisions and reduce the time to decide and the cost of the decision.

Service-oriented architecture (SOA) is one of those phrases that gets thrown around in everything from technical standards to business books. Thomas Erl makes four key points in his books:

- SOA can establish an abstraction of business logic and technology that allows a looser coupling between an organization's processes and its technology.
- SOA is an evolution of past approaches, preserving successful characteristics of traditional architectures and adding distinct new principles that foster service-orientation.
- SOA is ideally standardized throughout an enterprise, but achieving this requires a planned transition and still-evolving technology.
- SOA is a technology architecture that supports and promotes service-oriented principles throughout an enterprise.

What SOA does, at a fundamental level, is allow the development of individual pieces of business functionality in a way that lets them be combined and modified effectively and without tightly coupling them to each other.

The most effective way to apply EDM to operational decisions is to design what's called decision services - applications in your application portfolio or services in your service inventory that automate and manage highly targeted decisions that are part of your organization's day-to-day operation.

A decision service can be defined as a self-contained, callable component with a view of all conditions and actions that need to be considered to make an operational business decision. More simply, it's a component or service that answers a business question for other services.

Decision services use your data and the insight derived from it for automated decision making. They also isolate the logic behind your operational decisions, separating it from business processes and the mechanical operations of procedural application code. A decision service represents a single point of decision making throughout all your systems and processes, so it allows you to focus resources on improving and even optimizing that decision.

You can reuse decision services in multiple applications in many different operational environments. Decision services can also eliminate the time, cost, and technical risk of trying to reprogram many systems simultaneously to keep up with changing business requirements. For instance, the decision of whether to pay an insurance claim can be removed from the definition of the claims-processing business process. The legislative change cycle (that changes the decision) is different from the business cycle that drives process change, which allows these two update cycles to run separately. The business rules in this decision service could also be reused, such as for helping customers tell whether they have a valid claim before submitting it or for supporting third-party agents.

A decision service can be used to provide a "brain" for your composite applications. Composite applications are an effective way to assemble existing, working functionality to serve a new business purpose. You can put together purchased, built, or legacy components and create new applications more quickly. With decision services plugged into this approach, you can make these composite applications "smarter" and less reliant on people for decision making. Sometimes this approach makes it easier to connect existing services, and sometimes it lets you take more advantage of the services you have.

Case Study: Vehicle Fees and a Vehicle Registration Department in a Large U.S. State

Old Way

With both a centralized batch computer system and online systems at each local office, the complex task of calculating registration fees for a huge population of cars, trucks, and other vehicles was a problem. The systems were 30 years old, hard to maintain, and often out of synch with each other. Changes and updates required two separate development efforts, which made coordinating changes and ensuring consistency between the two systems difficult.

Because of the complexity of programs and duplication of effort, the vehicle registration department was challenged to meet legislatively mandated deadlines for fee changes. Any change to systems required a major IT project and ran the risk that the change would introduce unintended side effects. Risk was unacceptable because the systems handled more than \$4 billion in annual registration fees, revenue the state needed. There were many rules for each kind of vehicle and regular legislative changes to those rules.

EDM Way

A decision service was introduced and now calculates vehicle fees for both systems, batch and online. The service uses more than 2,000 business rules to process several hundred thousand transactions per day. Non-technical analysts who are responsible for overseeing legislative compliance can ensure correct implementation of policy rules throughout the systems without having to become programmers. These analysts develop and test problems with existing and newly mandated rules without involving IT, resulting in rapid turnaround and better accuracy in rule changes. The decision service complements existing systems, and by modernizing legacy systems, it reduces risk and expense.

Benefits

- Consistent rules could be made available rapidly to the new external-facing self-service Web site and telephone response systems.
- The decision service enhanced existing systems without having to replace or rewrite the majority of legacy applications and systems.
- In the first year alone, 13,000 hours of IT work was saved.
- The time to make a change in response to a transaction error was reduced 97 percent - from 8 hours to 15 minutes.

More About the Decision Service Model

A major benefit of adopting an SOA is supposed to be an increase in business agility, mostly because of the reduced time, cost, and difficulty of making a change. The definition of functionality as coherent components or services with

well-defined interfaces helps limit a change's impact to a single service, which makes change easier to control and implement. Well-defined services are loosely coupled - they use service contracts to allow services to interact without having to depend on interaction. These services change independently, and as long as the interface to the service doesn't need to be changed, independent service changes shouldn't affect other services.

SOA contrasts with the typical result of changing monolithic applications - a change is likely to cause a ripple effect throughout the application stack. SOA also supports a more iterative approach to defining services because of this control over the impact of change, which also helps in agility by eliminating the need to define a complete set of requirements upfront. SOA makes more agile development possible.

When you define business services with SOA, you can decouple the business from automation of the business. Business services are independent of a particular process; they perform a business function you can use in many processes. In this way, you can define new composite applications and business processes that use existing business services, which increases reuse as well as agility. Now you can assemble a new process - such as for handling a new channel, for example - mostly by orchestrating existing business services, especially with entity-centered business services in which functionality is associated with a defined entity or set of information, such as customers or accounts.

In addition, using an ESB to implement an SOA can increase agility by providing an integration layer and enabling you to assemble services on different platforms and perhaps with different interface semantics. By making it easy to add new services, transform messages to allow services to interact, and so on, an ESB can increase the level of agility beyond what service orientation alone can offer. Figure 1 shows how one organization used publish/subscribe interfaces with a decision service to ensure that specific process steps and messages on the ESB could trigger the same decision. This infrastructure also enabled the decision service to publish additional messages onto the ESB, which allowed for easy integration with other services connected to the ESB.

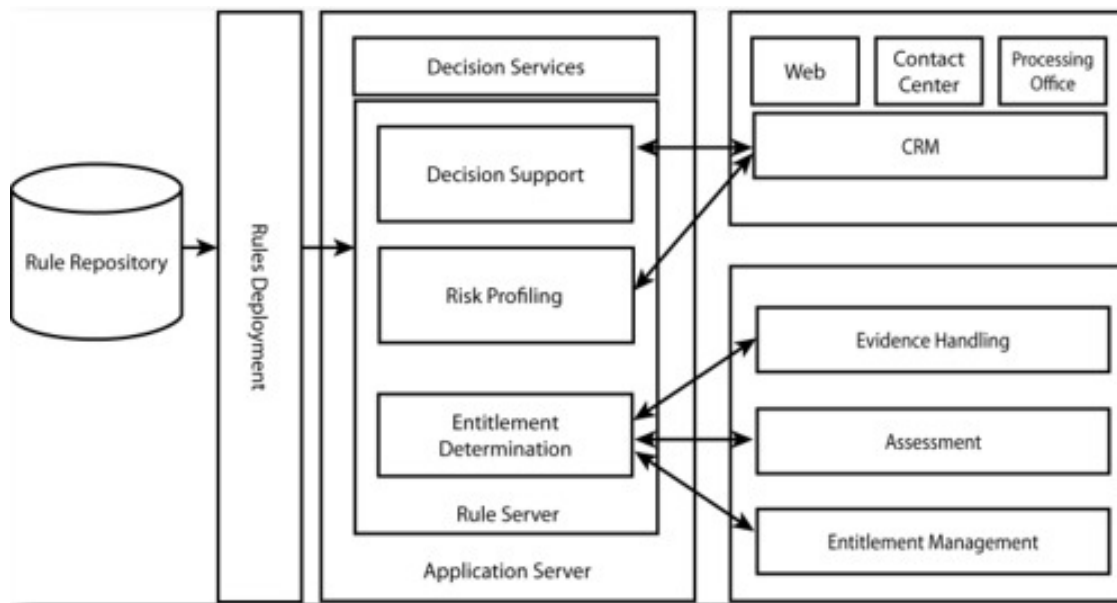


Figure 1: How a decision service can communicate decisions with BPM or workflow software and a message bus by using a publish/subscribe approach.

Clearly, some services implement a business function that must change more often and be more capable of adapting to change than others. Some services increase value when changed. The costs of failing to change some or doing so in a way that can't be audited for compliance might also vary in services. These services are defined as decision services and usually implement business functions that are in constant flux, complex or voluminous business logic, or business functions that aren't easy for programmers to understand or for which business user control is critical. With decision services, business logic can be changed and shared more easily between services in the SOA and non-service-enabled applications in the portfolio.

As shown in Figure 2, decision services are a subset of all possible business services as well as legacy services available for reuse.

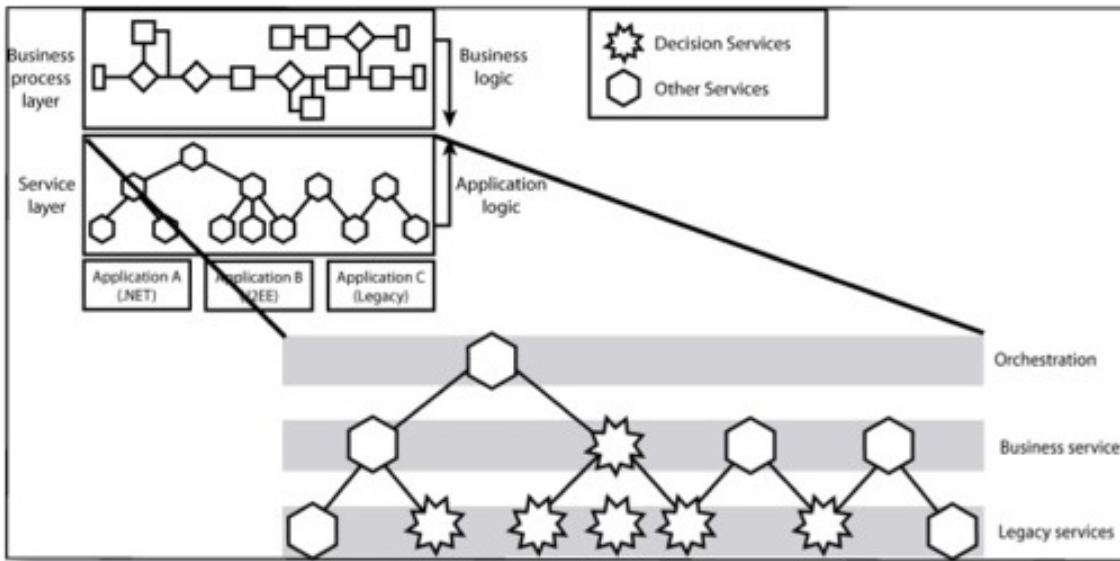


Figure 2: Decision services are centrally positioned to support other business services.

Figure 3 shows an SOA implementation of decision services for a mortgage lender in the United States. Various services provide credit retrieval and summarization, secondary market analysis, customer scores (based on risk models), and product and pricing information. The core decision service then handles the mortgage origination decision.

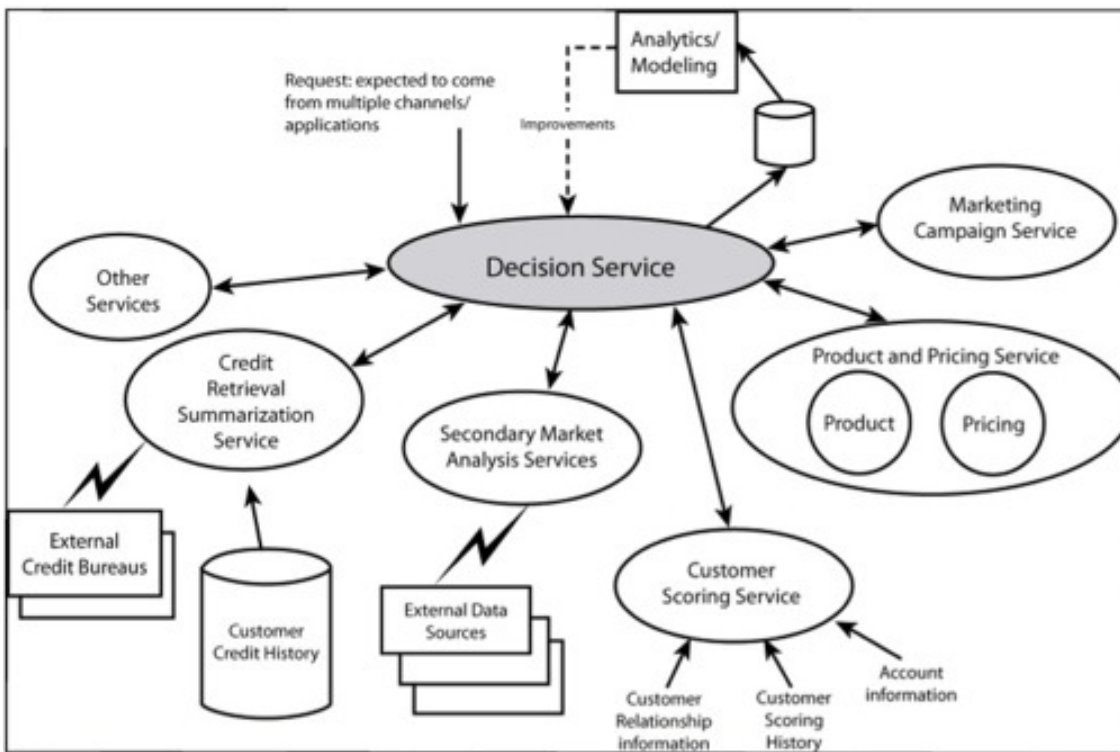


Figure 3: A service-oriented mortgage lending solution with decision services.

Decision services also allow a more effective "build or buy" decision. Organizations can buy services based on best practices and standards where the functionality of those services is not critical to the organizations' competitive differentiation. They can then build services with competitive potential and use an SOA infrastructure to compose them into effective applications and processes. Many services that differentiate an organization - that is, that define how it acts differently within a standard process framework - are decision services. Focusing on decision services can, therefore, make it possible to construct composite applications mostly from standard services that still deliver a unique and competitive customer experience. In addition, integrating analytics in decision services is a more effective way to apply data to improving processes than trying to "service-enable" traditional business intelligence (BI) tools.

Completing Application Decomposition

If you embed decisions in your applications, you hide these decisions from view and delegate details to the wrong people - systems developers, not business users. Traditional application development techniques hide decision logic deep inside software, making development time-consuming and costly. Developers have to translate business requirements ("If this condition is encountered, respond in this manner") into abstract representations in programming languages - a laborious process full of possibilities for error.

By embedding decisions in applications, your decisions become a liability. By managing decisions, however, you can enable business users to make their own changes, which reduces the time to make changes and reduces maintenance expenses. Focusing on decisions as a separate component is, in many ways, the last step in the decomposition of traditional applications.

Not so long ago, applications were monolithic, containing data, user interfaces, business logic, and process flow in one block of code. Then the process of decomposition shown in Figure 4 began. With the advent of databases, managing and reusing data became easier if it was removed from applications. For the first time, data was defined so that people knew what it represented, which allowed business users to access data for themselves. Next, client/server, thin clients, portals, and rich interfaces improved and separated the interface from the application. The same interface could access multiple applications in more sophisticated ways. Most recently, BPMSs have been adopted, which makes it possible to externalize work flow and build cross-application flows effectively. All that's left in applications is technical code and business logic. Decomposing applications one more step by separating business logic into its own managed environment makes more sense now and could be the most important advance to date.

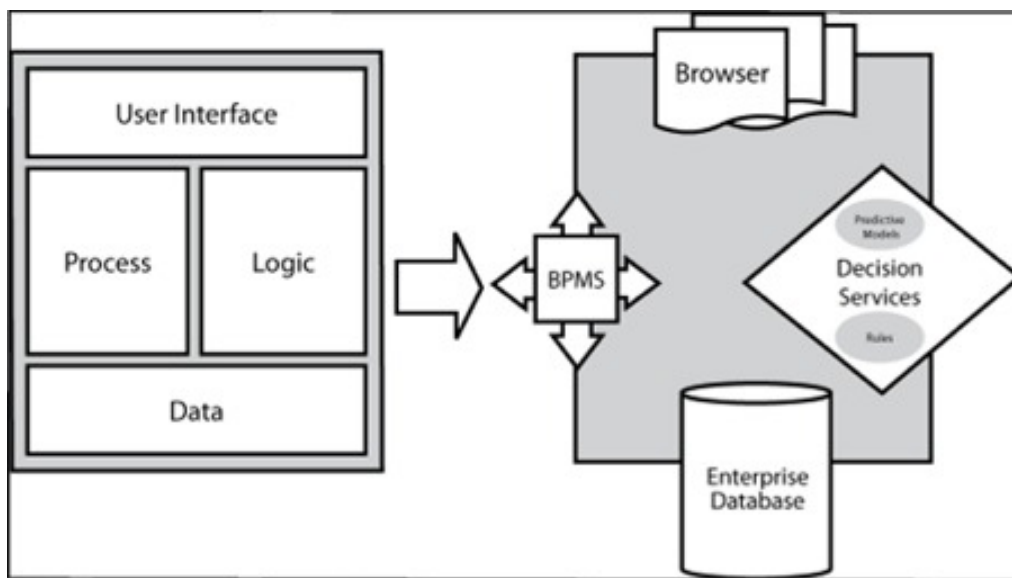


Figure 4: The evolution of applications from monolithic to completely decomposed.

Conclusion

Understanding the concepts behind smart enough systems and enterprise decision management provides you with an opportunity to structure your inventory of services around a new service model dedicated to business rules abstraction. Decision services can be strategically positioned within service-oriented architectures to enable central management of business rules and to provide the constant ability for the dynamic application of these rules.

References

[REF-1] ["Smart \(Enough\) Systems: How to Deliver Competitive Advantage by Automating Hidden Decisions"](#), James Taylor and Neil Raden, Prentice Hall (ISBN: 0132347962)

