



SOA Governance: Rules of the Game

To succeed, an enterprise SOA demands an enforceable set of policies for building, deploying, and managing services

SOA (SERVICE-ORIENTED ARCHITECTURE) PROMISES ENTERPRISES ENDLESS advantages: increased code reuse, reduced integration expense, better security, and — the big payoff — greater business agility. Whether you achieve those benefits, however, probably has more to do with your policies and procedures than the quality of your code.

Counterintuitive as it may seem, SOA requires more organizational discipline than previous development models. Your intuition might tell you that flexibility results from fewer rules, not more, but that's not the case.

Think of today's automobile: The reason you don't have to go to the Ford tire store to buy new tires for your Ford car is a direct result of standards, processes, and regulations. These rules and regulations give you the flexibility to buy tires anywhere and know that they will work on your vehicle.

BY PHILLIP J. WINDLEY | ILLUSTRATION BY GENE GREIF

“Build schools, not prisons. The goal is to help people conform to best practices, not police them.”

— Mark Ericson, Mindreef

That sort of standardization provides the underpinnings for SOA across an organization. To prevent IT from being overwhelmed by this new complexity, the industry has created classes of software — registries, repositories, and run-time management systems — that help keep all the rules straight (infoworld.com/3554). But creating an SOA demands more than using SOA-based tools. It requires that IT organizations make serious choices about design, which results in design rules.

Rules to Live By

Design rules specify interfaces, eliminate certain paths from consideration, and delineate boundaries between subsystems. A primary goal of architecture is to increase modularity and create well-defined abstractions based on service APIs. After those choices have been made, they must be recorded, communicated, and enforced. Design rules combined with enforcement are typically called “policies.”

The development and enforcement of SOA policies and procedures goes by the name SOA governance. Governance and architecture go hand in hand. In the same way that building codes, standards, and even inspec-

tions give building architects a context within which to work and ensure that their designs will fit in the community, SOA governance provides context for system architects and designers.

“Without SOA governance, you end up in a Web services version of DLL hell,” says Roman Stanek, chief software architect and founder of SOA registry provider Systinet. “SOA governance gives consistency, predictability, and allows big apps to be built from small pieces.”

If you’ve already got a strong IT governance process in place, it will serve you well as a foundation for SOA governance. If you’ve relied on informal governance in the past, moving to SOA will likely require some changes in how you manage development and operations. “Most organizations will fail miserably if they don’t implement the right form of governance,” says Anne Thomas Manes, vice president and research director at Burton Group. “SOA is about behavior, not something you build or buy. You have to change behavior to make it effective.”

The very mention of policies makes many IT professionals break out in a cold sweat. Developers, particularly innovative ones, worry that policies

and rules will straightjacket them. Worse, they know from experience that many policies are unrealistic, and they fear that governance will lead to bottlenecks and impractical, ivory-tower restrictions. With care, however, it’s possible to create a governance process that promotes service enablement and earns a buy-in from the people who have to live with it.

Creating Palatable Policies

According to Systinet’s Stanek, effective governance hinges on the processes that produce policy decisions — that is, the methods by which you make, communicate, and enforce choices.

Good SOA governance processes resemble town meetings more than they do dictatorships. “The biggest governance mistake organizations make is shortchanging communication and collaboration,” Burton’s Manes says.

The decision-making process can be organized in various ways, but ultimately it’s a social process that has to work in your organizational culture. “The rise of social software systems like LinkedIn has helped me gain an appreciation for SOA governance,” says Miko Matsumura, vice president of marketing at Infravio. “It’s about recognizing that people are socially organized. SOA governance incorporates best practices around organizational dynamics and how human beings behave in organizations.”

An enterprise architect at a major financial services firm describes how a lack of communication can create animosity toward governance: “If you set up a formal review of SOA projects as part of your governance process, it’s unreasonable to expect developers to adhere to policies that they aren’t aware of up front. Things have to be set down on paper. Create awareness,

Best Practices for Good Governance

SOA governance is social in nature, inviting continuous dialog between developers and architects.

Create a board of review. Governance policies should be developed, maintained, and modified by committee rather than decree. People who know what’s happening on the ground must have real input.

Develop an interoperability framework first. Standards are the basis of SOA. Start by building an extensible interoperability framework that details the protocols used by your organization.

Don’t get too granular. Overly detailed policies are difficult to maintain and restrain creativity. For applications low in risk to the enterprise, provide plenty of wiggle room.

Communicate early and often. Policies need to be explained more than once — and managers need to ensure updates are properly distributed, soliciting feedback along the way.

Establish COEs (centers of excellence). In large organizations, COEs have a full-time staff devoted to supporting SOA, including governance. An effective COE provides the guidance and education that holds your governance effort together.

Create policies with teeth. Without enforcement, policies mean little. Build policies into services where possible. Otherwise, make sure developers understand the repercussions of violations.

set expectations, and have things clearly stated. You don't want to create something that looks like the tax code and is similarly overwhelming."

Many organizations create a center of excellence or some other group in the enterprise architecture group to provide resources and guidance, to serve as a repository for best-practice information, and to operate tools that support the SOA governance process.

At Aeroplan, a Canadian consumer-loyalty plan provider, André Hébert has woven SOA governance into the fabric of his operation. "We have this architecture road map that is our conscience, if you like, and any time any project comes about and needs new functionality internally or externally, it is always run through the architecture committee," explains Hébert, Aeroplan's vice president of technology and e-business. "So it is kind of meshed into our day-to-day organization."

Simply put, governance processes should make it easy to do things the right way and hard to do them the wrong way. "Build schools, not prisons," says Mark Ericson, CTO of Mindreef. "The goal is to help people conform to best practices, not police them."

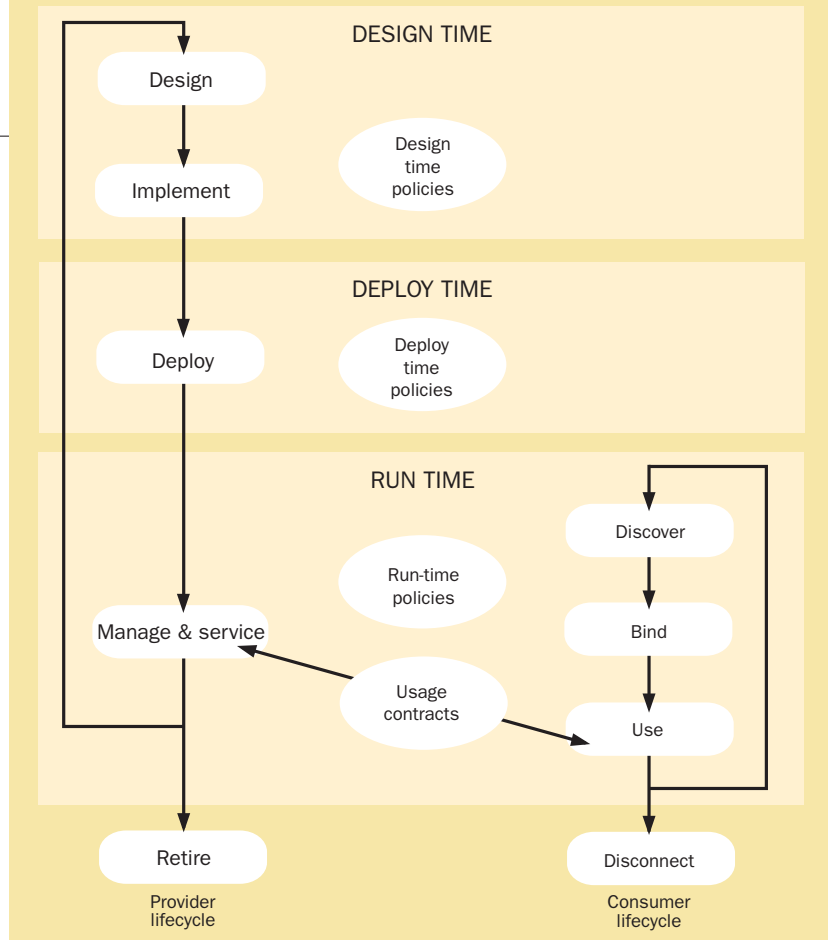
Policy No. 1

Policies can affect every aspect of the service lifecycle, including design, deployment, and operation. For example, a design-time policy might set out a corporate namespace, whereas a deployment policy might require that production-level services meet requirements laid out by the WS-I (Web Services Interoperability) organization. Or an operational policy might require all deployed services to be managed and use the corporate security infrastructure.

But in most organizations, it makes

From Design Time to Run Time

Governance policies are design rules plus enforcement. Policies are enforced from design time to run time, the latter extending to services consumers as well as providers.



sense to begin policy-making efforts with standards. After all, standards make SOA possible. Each enterprise must determine which standards are used where and when. For example, will WS-Security and WS-Policy be used? Under what circumstances will they be required?

You could call out specific standards in individual policies, but a better strategy is to create an IF (interoperability framework). An IF is a special policy that lists the standards that your organization will use, points to reference information, and indicates the status of the choice: approved, de facto, emerging, sustained, sunset, or in process.

These indicators are largely self-explanatory, but two deserve special

mention. "Sustained" indicates that even though the organization has decided to support another standard in this area, use of the older standard is supported. "Sunset" indicates that developers should migrate from the standard as soon as practical.

An IF separates references to quickly changing standards from individual policies, making them easier to manage. Overall, an IF is a great place to begin, if only because agreeing on standards will probably be the easiest policy task you'll face.

Wrangling Governance Assets

The governance process creates policies, XML Schemas, WSDL documents, documentation, and other artifacts that must be distributed to all inter-

“Doing lots of little Web services projects all over the place with no governance isn’t SOA, it’s just playing.”

— Anne Thomas Manes, Burton Group

ested parties. Effective communication means much more than e-mailing these assets or posting them to a Web site. To be useful, governance artifacts need to be searchable, versioned, and easily — and precisely — referenced. Moreover, many of these artifacts should be in a machine-usable format for dynamic discovery and binding.

Registries are the primary tool organizations use for managing and communicating governance artifacts, as well as automating key governance activities. A registry provides a central reference or “system of record” for services. Think of it as a place where services can be advertised by providers and discovered by consumers inside an organization — a control point for governing service availability, versioning, and compliance with internal and external requirements.

Some vendors offer what they call “repositories” — registries that also serve as places to store metadata for services that go beyond WSDL documents. Because registries and repositories are so important to the development process, look for software that deeply integrates with your development environment. For example, if you’re an Eclipse shop, be sure that the registry you choose has a plug-in for Eclipse.

Registries also have APIs that are used by applications at run time to find services and associated policies dynamically. Additional metadata might, for example, tell the service consumer what security policy the service requires so that the client can dynamically adjust to changing service offerings.

Run-time policies are most effective when they can be expressed in a WSM (Web services management) system, such as those provided by Actional, AmberPoint, Blue Titan, or SOA Soft-

ware. Most WSM systems provide a means of enforcing conditions on the SOAP envelope at run time. For example, a WSM can ensure that services use a particular security protocol or that enclosed XML documents conform to a particular schema.

Look for systems that not only allow you to manage, version, and discover policy at design and run time but also provide for policy reuse. Being able to create and manage policies independent of a specific service allows you to fully leverage policy assets.

In general, enterprises should automate as much of the governance process as possible. That requires a centralized investment in people, organizations, and tools to establish the appropriate context for SOA. Properly done, your governance process and its associated infrastructure may be the only centralized elements in your entire SOA deployment.

Laying Down the Law

Building codes would not be very effective at creating safe, pleasant cities if there were no building inspectors. Similarly, SOA policies aren’t worth anything unless they’re enforced.

Some policies will be codified in WSM or development systems and can be enforced automatically. Others will be aimed at changing or regulating the behavior of people — such as an edict that all services used in production-quality applications must be listed in a production registry — and are less easily codified, let alone enforced automatically.

An enterprise’s project management office is often a one-stop shop for policy enforcement. Project managers can guide projects toward compliance without being heavy-handed. Other tactics include project reviews by an

SOA governance body before funding is released.

Most important, recognize that SOA policy goals need to be aligned with financial incentives. Otherwise, service enablement will go on the back burner. “In real life, development organizations run against budgets and due dates and abandon service-enabling. Without a strong governance process, SOA will deploy unevenly across the organization,” says Bob Laird, MCI’s chief architect. “If some parts of the organization aren’t SOA-enabled, you’re at a disadvantage.”

Enforcement requires closing the loop. When a service or process is found to be noncompliant, you need to take corrective action. On the other hand, noncompliance may also indicate that a policy is overly restrictive or poorly written. Ensure that your feedback process has pathways that lead to policy improvement.

Also, no matter how carefully crafted the policy, developers are sure to run into situations where an exception to the policy would be in the organization’s best interest. Make sure every policy has a route for raising and approving exceptions. Policies often name, by role, a policy owner who has spot authority to grant exceptions.

Crafting Contracts

Policies must extend beyond broad-based governance to service-specific rules — wrapped into a contract of some kind. It may seem odd to think of one part of your organization creating a contract with another part, but that’s part of what makes services different from just delivering code.

“Contracts tie together the consumer and producer lifecycles,” Systinet’s Stanek explains. “Contracts are a reference point that delineates the needs

of both parties.” Contracts are not a one-way service-level agreement; they may also define acceptable consumer behavior so that service operators can plan consumption. The idea is to let the producer monitor and anticipate consumer demand.

Contracts specify the set of policies by which both sides agree to abide. There are no standards for contract representation yet, but much can be accomplished with predefined service levels — think “bronze,” “silver,” and “gold” — and templates that map concepts to established policies. Contracts in this context are much less legal documents than they are agreements that support the producer-consumer interaction and govern the transactions that occur in that context.

Getting Off on the Right Foot

Governance is easy to ignore early on. That’s OK during the experimental stages of SOA but quickly changes as services approach the production phase. As Burton’s Manes points out, “Many organizations don’t start to think about governance until things are completely out of control.”

SOA governance starts with a vision of what the governance process will accomplish. That vision is a collective effort of the people who will design, build, and use services — not only developers but IT managers and business unit leaders as well. A robust vision based on consensus derives from broad participation.

Ed Horst, vice president of product strategy at AmberPoint, recommends

building your governance infrastructure early. “Try to get a little bit of management, registry, and security in use early on. Bad habits are established in each of these areas that will complicate attempts to govern as the SOA effort scales. If there’s no management system in place, developers will hardcode management logic into the service and waste time writing it. What’s more, they’ll have no idea how to get management information when they actually deploy with a real management system,” he says.

Because SOA is inherently decentralized, governance is the difference between success and chaos. As Manes says, “Doing lots of little Web services projects all over the place with no governance isn’t SOA, it’s just playing.” ☞

A Degree of Tolerance

GOVERNANCE, LIKE SOA ITSELF, COMES WITH ITS OWN CATCH-22: You want to create a broad-based, modularized IT environment that supports unprecedented agility, but you can’t possibly anticipate every variation — nor can you create policies that take into account every feasible future scenario.

That’s where the idea of “tolerance” comes in. For SOA applications that involve actual business transactions, you need careful, detailed rules and formal change management. But in other instances, exemplified by a number of Web 2.0-style apps, developers should be able to get creative without being smacked down by some architectural committee.

Take CheapGas (infoworld.com/3790), a great site for finding the gas stations in your neighborhood with the cheapest gas. CheapGas is a “mash-up” of Google Maps and data from another site, GasBuddy.com. Unlike most enterprise-level Web services, CheapGas exists without much governance at all. There are some necessary standards, such as HTTP and Google Maps API, but nothing resembling the stack of WS-* protocols that could accompany some mission-critical enterprise app.

Dion Hitchcliff, CTO of Sphere of Influence, remarks that “90 percent of the time if you’re presented with a service with data in it, it’s going to be RSS, not SOAP.” That’s no accident: RSS delivered over HTTP is a remarkably tolerant format. SOAP has its

place, but often a simpler, more tolerant approach will deliver results sooner and with more opportunities for reuse. Where you use one or the other depends on the application and the level of risk you can tolerate.

In *Design Rules*, their groundbreaking book on modularity, Carliss Baldwin and Kim Clark describe how architecture “free[s] designers to experiment with different approaches” within the design rules of the architecture. They argue that this experimentation yields tremendous value. Conversely, governance that is overly strict can tie the hands of developers and drive value out of your SOA effort.

One issue requiring governance might involve choosing a standard for security tokens. Should you support SAML, Kerberos, or something else? An alternate approach could be tolerance of different security token formats, which would require installing or licensing a token exchange service.

Tolerant protocols and processes aren’t at odds with governance; they’re complementary, but the more tolerant your service, the less governance you’ll need to create robust systems. The ultimate reward of well-designed services is that they get used in unintended and serendipitous ways, making them more useful than their designers could have imagined. This kind of reuse is the real promise of SOA. — P.J.W.