Business Innovation and Service Abstractions

Toby Considine

University of North Carolina and Chair, OASIS oBIX Committee
169 Durham-Eubanks Road, Pittsboro, NC 27312

Toby.Considine@unc.edu

Keywords: Innovation, Service Orientation, Security, Abstraction

Abstract

True Scalability and interoperability require abstraction and security. Most control systems today expose name/value tag pairs as their interface. Interaction with exposed tag pairs requires a deep understanding of the underlying systems. Secure interaction with sets of tag pairs can only practically be exposed as monolithic yes/no decisions for the entire set. Lack of abstractions, in both process and security, are a barrier to new business interactions.

The smart grid will require integration with smart buildings and their associated power capabilities. Abstract models for system interaction will enable large-scale system integrations. Abstract service models will hide underlying system detail while exposing the diverse systems for orchestration.

Security is the application of policy to service. Situation awareness is required of any mature security model. Situation awareness is only useful when applied to abstractions above identity, above process, and above function. Only when these abstractions are defined, can one then define security.

Service abstractions and security abstractions must develop together. Security enables the open provision of business services. Defined services enable the definition of business policies. Service and security together enable open trustworthy interactions with third parties.

1. BACKGROUND

Today's engineered systems are too complex for further integration. Each function is tightly coupled with the next. Integration requires deep domain knowledge of each side of the integration. Multi-domain integrations require deep knowledge of multiple domains. The primary domains for this discussion are Generation systems, Transmission and Distribution SCADA, End Node systems. End nodes include a plethora of systems as even the smallest home may have a half dozen different non-interoperable systems. None of these systems currently communicates using the business semantics and service architectures of the enterprise.

Engineered systems developed in isolation and little overlap with nearby domains or with best practices in enterprise development. Even low level communications share little with nearby domains as a host of non-interoperable low-voltage protocols can be found even within each domain. Since the systems weren't connected, this seemed of little consequence. Now that it has become practical to interconnect both the engineered systems within a facility and systems in multiple sites around the planet, efforts are underway to integrate many systems previously isolated.

Engineered systems have traditionally been integrated at a low or "concrete" level. The geometric increase in complexity that accompanies low-level integration across systems has made such integration increasingly complex. Many current developers and integrators are comfortable with current approaches, which have the advantage of familiarity and result in long backlogs.

Economic forces are driving increasing integration of existing systems, an integration hindered by the growing complexity of integration of these systems. As we build new systems and 'renovate' old ones, there is an opportunity to consider how to link them into a shared infrastructure.

To accelerate these integrations, we must create and leverage a common information architecture. The underlying systems must be properly factored for maximum reusability. Systems will need to accept the output of other systems as input. As systems begin interacting with other systems, we will need a framework of situation awareness, i.e., what system is requesting this service and what is its authority?

These changes will enable the delivery of entire engineered systems as components. Systems engineers will be able to focus on and compete with their core competencies rather than on understanding all the diverse systems on something as large as the North American power grid.

2. LIMITS OF PROCESS-ORIENTED ARCITECTURE

True Scalability and interoperability require abstraction and security. Most control systems today expose name/value tag pairs as their interface. Interaction with exposed tag pairs requires a deep understanding of the underlying systems. Secure interaction with sets of tag pairs can only practically be exposed as monolithic yes/no decisions for the entire set.

2.1. Process Oriented Development

Engineered systems programming is largely procedural. When you receive this signal, energize that relay. Four seconds after this coil reaches temperature, turn on this fan. This style of programming requires access to all the details; hence the name value tag pairs. Integrating two different systems requires a deep understanding of each.

Interoperability of component systems is impossible at this level of integration. Each instance of a control system will have slightly different internal tags. Even two systems with the same part number may have quite different internal components if manufactured a year apart.

This problem is worsened as the number of systems increases. With the domain knowledge required for each new integration, the proportion of systems engineers with enough knowledge of enough domains goes down.

While the engineer looks to maximum efficiency or process, the efficiency of integration decreases.

2.2. Process Oriented Security

Process interactions are targeted only at interactions with other processes that are known a priori. All name/value tags are like all others; no categorical distinctions can be made between them. No metadata is known about the underlying business function.

Without metadata, there is no way to secure one of these systems. Security requires situation awareness. Security is the art of offering the right person in the right situation unimpeded access to functionality. Security requires each systems recognize its relationship to other agents, whether human or automata.

Process oriented security is inherently at the lowest level. Without clear definitions at the level of the system of the business function provided, there can be no recognition of appropriate interactions with external agents. Without recognition of appropriate interactions, there can be no nuances of security, and no distinguishing between external agents. The process is left with only two security modes: full and unrestricted access, or complete restriction of any access.

This limited security applies whether the security is enforced by access lists, by network addresses, or by list based, or by encryption.

3. SEMANTICS AND SERVICE ORIENTATION

In systems, the term service refers to a discretely defined set of contiguous and autonomous business or technical functionality. OASIS defines service as "a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description."

In economics, service is the non-material equivalent of a good. Service provision has been defined as an economic activity that does not result in ownership. A service is the result of a process that creates benefits by facilitating either a change in customers, a change in their physical possessions, or a change in their intangible assets

In engineered systems, the service is not the underlying process, but the reason why that process was procured. The service provided by a Heating, Ventilation, and Cooling (HVAC) system is not the blowing of fans and compression of coolant. The service provided by an HVAC system is the economic provision of healthful and comfortable air. In another situation, the service provided by an HVAC system is the preservation of an economic asset by providing an optimum physiochemical environment. The service is what the owner actually wishes to buy.

3.1. Semantics of Service

As we discover the core services provided by the underlying process, we need to categorize each service. We do this by defining standards names for each function exposed as a service. We refer to these names as the service semantics.

Semantics lets us group similar functions. By properly factoring functions that share the same semantics, we can discover the operational inputs that these functions require. Semantics and the factored operational inputs define the surface of a system.

When different systems share a common surface, we have interoperability. Interoperability does more than let us swap out one system for another. Interoperability lets us interact with different systems over space, at many locations, and over time, as technology changes.

3.2. Security and Situation Awareness

System semantics give us the means to define more nuanced security. Whereas under process, we merely had points, with semantic services, we can see business situations that we can permit or obstruct agents from interacting with.

Let’s examine, as an example, the HVAC system whose service is the economic provision of healthful and

comfortable air. We may determine that certain classes of business users may adjust the comfort portion of each defined space. Each system may offer up a different set of points as the comfort related settings. Process overlaid with semantics defines situations.

3.3. Security enables Services

Security creates an awareness of who is asking for a service. System semantics names what services are available. Both requester and service are required for situational awareness.

Without nuanced security, systems are unable to expose surfaces. As we define security, the range of services that a system can offer expands. Security is the great enabler of business services.

Abstract surfaces occult the inner working process of each system exposing only the abstract operations as services. The service defines the purpose of each component system within the larger integration and within its local ecosystem. I like to call this purpose the system's mission.

Each building system's first job is to defend its mission. Defending this mission may include preventing all but those with the highest authority (relative to the system) from reconfiguring the system. Integrators get to perform loop tuning; tenants get to modify comfort settings.

3.4. Semantics enable Discoverability

Discoverability is an important feature for systems that can be modified without central engineering control.

Consider networked printers on a modern network. They can be discovered by asking one's system to search for all nearby printers. If you wish, you can print immediately, or you can discover the heterogeneity of the interface. This one has two bins. That one offers color.

When properly implemented, services and their semantic tagging can create the "Plug and Play" self configuring system.

4. EFFECTS ON ENGINEERED SYSTEMS OF SERVICES INTERFACES

Systems that are quite different in complexity and technology can provide the same service. Owners and integrators will be able to compare different systems as to how safe, effective, and economic their operation is without changing the higher level integration.

This reduces the friction on decisions to switch from one service component to another. There will no longer be a large cost of integration associated with each system purchase or upgrade. Competition between system alternatives will be increasingly based on price and performance, and less on compatibility with installed base. This will reduce sales cycles and increase the incentives for innovation.

5. IMPLICATIONS FOR THE POWER GRID

Systems that expose their services using standard semantics become discoverable. These services can be listed in a registry and each registry will include the standard name for the service provided. If there is no registry, but the systems are discovered by some other process, each will still be able to name itself when contacted.

Discoverability is essential for a system as large as the power grid. Discoverability enables grid models to understand building systems as they are installed and changed by building owners and tenants.

Alternately, discoverability of standard services opens up a market for standards based agents, interacting with business and home activities within the building, and with the buildings embedded systems.

An important effect of this model is that the power grid itself must manifest itself to the in-building agent as a service. The service should provide features to analyze the effectiveness of building operations (instantaneous electricity usage) as well as their cost (instantaneous pricing). Power from the grid, with its price, and power from an on-site generator, with its price, and even power from an on-site renewable source are all merely instances of the same service to the on-site agent.

More advanced systems will want to receive metrics of service and reliability from the power source services. Committee members in The Green Grid, a data center operations standards group have already asked for information on immediate projections of reliability from the building transformer and from local distribution. Data center operations want this projected reliability information to "reflect deep domain understanding to produce engineered information that does not require operators to acquire their own domain expertise."

5.1. System Security on the Grid

Significant segments of people and businesses will not give up autonomy over their private resources to any third party. Power Grid assets must provide secure access to their information while not sharing information gleaned from inside the buildings.

In-building agents may be controlled by building owners and tenants or by third parties deputized to make decisions in their behalf. The grid, the services, and other agents must be able to understand the chains of authority that accompany each transaction.

6. IMPLICATIONS FOR THE POWER GRID

A service can abstract the internal operations of each system. This service defines the mission of the internal operations each system. Each building system should defend its mission. Systems that are quite different in complexity and technology can provide the same service. Owners and integrators will be able to compare different systems as to how safe, effective, and economic their operation is without changing the higher level integration.

Services enable security, and security enables allowing the tenant or owner to interact with building systems. Agents can be restricted to which services they interact with, and what performance they request using understandable business rules. This level of abstraction will support internal tenants or third party service managers to safely and effectively interact with the building systems.

Service oriented architectures and integrations make possible large scale interactions. Service discovery enables ad hoc interactions. Services hide implementation details. Service oriented architecture will enable orchestration of building systems including site-oriented energy generation and storage. New business models will take advantage of these new interactions to drive energy use reduction through innovation.

References

Pat Helland, "Metropolis", Microsoft Architect Journal, April 2004
http://msdn2.microsoft.com/en-us/library/aa480026

OASIS SOA Reference Model Technical Committee, "OASIS Reference Model for Service Oriented Architecture 1.0" http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm

Biography

Toby Considine has been working with enterprise applications and integration of embedded control systems for 25 years. As a Systems Specialist at the University of North Carolina, Mr. Considine has struggled with the network demands, poor data sharing, and non-scalable security issues posed by last-generation embedded building systems. This work led him to speaking and writing on open discoverable data standards for control systems. For the last five years, Mr. Considine has chaired the OASIS oBIX (open Building Information eXchange) standards committee.

In earlier work, Mr. Considine was for six years on the faculty of the Institute for Facilities Management, running its IT for Facilities Professionals track. Before coming to the University, Mr. Considine worked to integrate other silo processes into the enterprise for companies including The Architect's Collaborative, Reebok, Digital Equipment Corporation, and Southco Distributing.

He also worked on one of the largest of the early public access computer systems, CityNet, and so can claim to be one of the few participants in a DOTCOM meltdown in the '80s. He acquired his respect for the power and limitations process oriented programming writing device drivers for some of the first microcomputers sold.