Service-Oriented Architecture (SOA) and Software Product Lines: Pre-Implementation Decisions

Dennis Smith and Grace Lewis
Software Engineering Institute, Carnegie Mellon University
Pittsburgh, PA, USA
{dbs,glewis}@sei.cmu.edu

Abstract

This paper examines the use of Service-Oriented Architecture (SOA) services as core assets in a Software product Line (SPL). After a brief introduction to the main concepts of SOA and SPL, the paper identifies a small set of decisions that are required before implementation of SOA-SPL systems. These decisions have to do with 1) the mapping of SOA concepts to the SPL framework, and 2) an initial set of potential variation mechanisms. The paper also identifies future work to more completely address SOA-SPL implementation planning.

1. Introduction

A Software Product Line (SPL) is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way [1]. Successful products lines have enabled organizations to capitalize on systematic reuse to achieve business goals and desired software benefits such as productivity gains, decreased development costs, improved time to market, higher reliability, and competitive advantage [1, 2].

Service Oriented Architecture (SOA) is a way of designing, developing, deploying and managing systems, in which

- Services provide reusable business functionality.
- Applications or other service consumers are built using functionality from available services.
- Service interface definitions are first-class artifacts.
- An SOA infrastructure enables discovery, composition, and invocation of services.
- Protocols are predominantly, but not exclusively, message-based document exchanges [3].

The business case for both SPL and SOA emphasize efficiencies and cost savings through reuse. SPL focuses on the use of a common, managed set of core assets for rapidly producing multiple products or systems, according to a centrally managed production plan. An SOA implementation exposes standard interfaces to make services available for authorized service consumers to use in a variety of ways. These consumers of services are not necessarily anticipated by service providers, or controlled by a central authority.

A number of authors have suggested a relationship between SPL and SOA [4, 5, 6, 7, 8]. These works focus primarily on the use of services as core assets in SPL in which the services handle variability, and they address the relationship between SPL and SOA from a conceptual level but do not go into details about implementation. The goal of this paper is to identify an initial set of decisions that have to be made before going into implementation. These decisions are based on 1) the mapping of SOA concepts to the SPL framework and 2) an initial set of potential variation mechanisms.

The paper is organized as follows. Section 2 briefly outlines SPL concepts and identifies a set of specific SPL practice areas to be addressed. Section 3 outlines key SOA concepts that are relevant for using services at SPL core assets. Section 4 identifies a set of pre-implementation decisions. Section 4.1 outlines decisions that map to a selected set of SPL practice areas. Because variation points are central to a successful SPL implementation, Section 4.2 identifies an initial set of potential variation mechanisms for services. Finally, Section 5 provides a summary, conclusions and next steps.

2. Software Product Line Concepts

The SPL framework identifies three essential product line activities: core asset development, product

development and management. The framework also defines a set of practice areas that are essential for carrying out these three essential product line activities. A practice area is a body of work or a collection of activities that an organization must master to successfully carry out the essential work of a product line. Practice areas help to make the essential activities more achievable by defining activities that are smaller and more tractable than a broad imperative such as "develop core assets" [2].

Because this paper addresses the use of services as core assets, the SPL activity of most immediate concern is that of core asset development. While a number of SPL practice areas are relevant, we identify a set of pre-implementation decisions that need to be made in the practice areas of:

- Architecture definition
- Using externally available software
- Mining existing assets
- Testing

After these decision points are identified, some potential mechanisms for using services as core assets are outlined.

3. Service-Oriented Architecture Concepts

This section identifies key SOA concepts that are relevant for using services as SPL core assets.

At a high level, as shown in Figure 1, there are three major components of service-oriented systems: services, service consumers and SOA infrastructure.

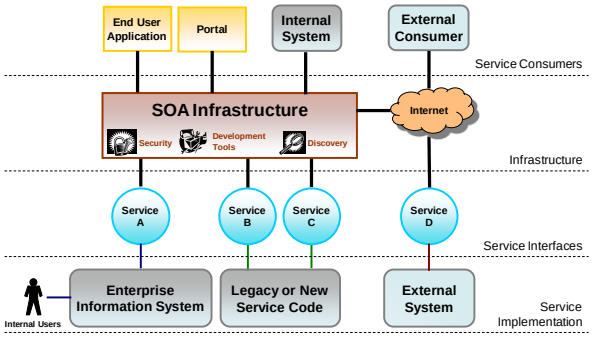


Figure 1. High-Level Representation of a Service-Oriented System

- Services are reusable components that represent business tasks, such as customer lookup, weather, account lookup, or credit card validation and that can be globally distributed across organizations and reconfigured to support new business processes. A distinguishing factor is that service interface definitions are standardized, well-defined, first-class artifacts, available in some form of service registry so that services can be discovered by service consumers.
- Service consumers are clients for the functionality provided by the services. Examples of service consumers are end-user applications, portals, internal and external systems, or other services in the context of composite services. A product in a product line could be a consumer of a service.
- SOA infrastructure connects service consumers to services. It usually implements a loosely coupled, message-based communication model. The infrastructure often contains elements to support service discovery, security, data transformation and other operations. A common SOA infrastructure is an Enterprise Service Bus (ESB) to support Web Service environments [9].

Service-oriented systems support three main types of operations: service discovery, service composition, and service invocation.

- Service discovery: Service providers place information about their services in a service registry and service consumers query this registry for services with desired characteristics.
- Service composition: Applications and other service consumers compose functionality provided by services to fulfill their goals. Languages such as the Business Process Execution Language (BPEL) support the orchestration of services in a Web Services environment [10].
- Service invocation: There are two common patterns for service invocation
  - a simple invocation pattern where service consumers directly invoke services over a network, typically via synchronous, direct, request-reply connections
  - a richer invocation pattern where service consumers invoke services via a middleware component that supports SOA environments, such as an Enterprise Service Bus (ESB) [9]

These basic SOA components and activities are addressed in the decision points that follow.

4. Pre-Implementation Decisions for SOA-SPL Systems

As mentioned earlier, this paper focuses on pre-implementation decisions for systems that use as SPL core assets available as services. Subsection 4.1 outlines decisions in four SPL practice areas: Architecture Definition, Using Externally Available Software, Mining Existing Assets, and Testing. Subsection 4.2 identifies decisions to make on potential variation mechanisms.

4.1 SPL Practice Area Decisions

Architecture Definition

The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them. [11]. If core assets are to be made available as services, decisions include:

- Decisions on the specific sets of SOA standards, interfaces and technologies for implementation and how will these will interoperate with the rest of the product line architecture.
    - The first choice concerns the standards to use. The most common SOA standards are web services. Alternative SOA standards may be used, such as REST. Decisions on standards require an analysis of both Quality of Service (QoS) needs and functionality. Web service standards have greater support for QoS needs such as security, availability, and performance. REST is more flexible and easier to compose, but has less support for most of the standard QoS needs. It is more appropriate for read-only functionality, typical of mashups, where there are minimal QoS requirements and concerns.
    - Once standards are chosen, the next decisions focus on specific standards and tools required for specific SPL QoS needs. In the case of web services, a large number of choices are available—approximately 250 WS-* web service standards. Each standard in turn may have a number of different versions and tool support. Many of these decisions will have architectural implications.
- Decisions on how discovery, composition and invocation are to be accomplished. A number of options can be considered, such as building a SOA infrastructure that supports these operations, buying an ESB product or embedding these operations within the context of a broader product line architecture. Each of these basic operations requires a set of decisions.
    - Discovery. In most literature that discusses the relationship between SOA and SPL, there is an expressed need for discovery to take place at runtime. However, the current state of the practice only supports design-time discovery. Decisions need to be made on the specific mechanisms for discovery and for interacting with a service registry. If some type of runtime discovery is required, there may need to be some form of user intervention to choose an appropriate service or the non-trivial implementation of a service broker. Decisions in the former case include how to handle user intervention and in the latter case how the broker is to be implemented and used by SOA-SPL developers.
    - Composition. Policies for composing services and allowed usage of services need to be established. These policies need to be made explicit and enforcement mechanisms need to be established. Decisions include specific mechanisms for performing composition, such as WS-BPEL, and where to implement composition, such as in the infrastructure.
    - Invocation. In most cases the infrastructure will support the invocation of services. In this case decisions need to be made on the type of functionality to be handled by the middleware (e.g., routing, mediation, process orchestration, complex event processing). On the other hand, if services are to be invoked directly by the service consumer, decisions need to be made on how this will be accomplished.
- Services may be accessed via an Intranet or the Internet. Decisions need to be made on the scope of access, firewalls, permissions, and control of services. If the Internet is to be used, performance and availability metrics need to be identified, measured and tracked; sources of bottlenecks need to be identified; and satisfaction of service level agreements (SLAs) needs to be monitored.

Using Externally Available Software

There is a growing market of externally developed services that can be purchased or licensed. In addition, ERP vendors are making significant investments to add service interfaces to their existing ERP solutions to be used by custom applications. Common business services, such as check credit, customer lookup and check inventory are strong candidates for core assets if they are relevant for the product line.

Decisions required for the use of externally available services include:

- How are the services to be accessed, what standards do they support, what outputs do they return, and in what form?
- Do the external services meet the functionality and quality of service requirements of the SPL?
- What type of testing has been performed on the services, at what level, and what are the results?

- How appropriate and effective is the SLA attached to the service?
- What types of mechanisms are built into the services to handle variations?
- Do the external services have an option to establish variability points?
- Can variability be handled by the infrastructure or by consumers of services?

Mining Existing Assets

If services are to be mined from existing assets, an important pre-implementation decision concerns the viability of exposing services from existing assets. This decision requires per-system answers to a set of questions, including:

- Does it make sense to migrate the legacy system to an SOA environment?
- What services make sense to develop?
- What legacy system components can be used to implement these services?
- What changes to components are needed to accomplish the migration?
- What migration strategies are most appropriate?
- What are the preliminary estimates of cost and risk?
- What is an ideal pilot project that can help address some of these risks?

The Service Reuse and Migration Technique (SMART) [12] provides one systematic method for answering these questions and making decisions. SMART addresses both general migration issues as well as those that are relevant to a specific situation. In the case of using services as part of a product line, specific points to be addressed include variation points, relationship to other core assets, composition strategies and SLAs.

Testing

Issues with testing SPL core assets implemented as services need to be addressed from both the service provider and service consumer perspectives.

From a service provider perspective, systems that expose functionality as services usually have "day jobs". This means that the system operates in a "business as usual" manner and also provides service interfaces so that other systems (internal and/or external to the organization) have access to a subset of functionality that exists in the system. This requires decisions for how to address testing challenges.

- Regression tests cover both conventional interfaces as well as service interfaces to make sure that changes made for one set of users do not affect the other set of users
- Functional tests consider potentially unknown users and uses of the functionality provided by the service. Functional testing needs to cover both current as well as potential usage scenarios.
- Exposing system functionality as services creates the potential for having a greater number of consumers of system functionality. This requires additional security testing, stress testing and load testing.
- Regression testing needs to verify whether existing service-level agreements (SLAs) are affected.
- Some service providers have test instances of their services to allow service consumers to perform end-to-end testing. As a result service providers have to maintain separate instances of their service interfaces as well as their service implementation so that test data does not affect production data. In addition they require extensive logging to use failure data for internal testing and improvement.

Service consumers need to make a greater set of decisions if the core assets use externally available software. For externally developed software, an SLA protects both the service consumer and provider in case of failure, but it does not prevent or eliminate failure. As a result service consumers need to develop and test their systems to consider the case when services are completely unavailable. External services also mean that there is no control over changes made to the service, release cycles, or even shutdown. Service consumers will have to be tested every time there is a new service release.

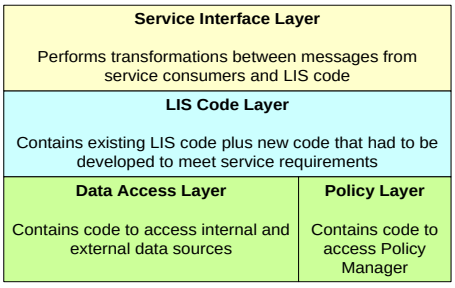
4.2 Variation Mechanism Decisions

Cohen and Tarr both present simple examples of models for product lines that are composed of services for medical records and insurance claims respectively [4, 7]. In the medical example, variations can occur depending on such factors as the medical actor who is involved (physician, nurse, technician), the type of medical practice (cardiology, radiology, endocrinology) and the type of health care organization (hospital, insurance company, physicians office). Core assets may include services for medical treatment, billing, and patient information. In the insurance claims example, variations can occur on such items as type of policy (life, home, auto), and state-specific policies.

Management of variability points is a key to product line success. However, variation mechanisms for SPL

core assets implemented as SOA services have not been systematically addressed. Because decisions on specific variability mechanisms are important, we identify an initial set of decisions:

1. Parameters for invoking services. This is a simple variation mechanism in which parameters are used to invoke different variations of a service, such as different treatment responsibilities that may depend on role (physician, nurse, technician). This has been identified by in the literature as a primary variability mechanism [4,7].
2. Using infrastructure services to hide variability. A number of services can be common tasks that are delegated to the infrastructure. Examples include role-based identity management and data formatting. In the health domain, different sets of actors will require very different access and authorization privileges. Health care insurers will have the right to see financial information, physicians can see detailed treatment information, and research organizations will only be able to see information that is completely anonymous. These types of variations can be effectively handled as infrastructure services that are invoked when needed.
3. Encapsulating variability within a service. This approach isolates core service functionality from aspects that are either highly changeable, or in the case of an SPL, potential variation points. Figure 2 shows an example in which separate service layers are created for the interface, core service code, data access and in this case, access to a policy manager infrastructure service.



4. Differential composition of atomic services. Services are often developed as atomic services that perform specific tasks. In situations where different configurations are required (such as SPL), or where external policies in the business environment require frequent unplanned changes (such as health care), building in a capability for composing services from a number of atomic services enables variability. The composition of services can be delegated to the infrastructure through a standard such as WS-BPEL (Web Services Business process Execution Language) or through proprietary or custom developed Business Process Management (BPM) functionality. This enables applications or products in a product line to be developed through the integration of functionality from existing services.
5. Using different protocols for interface implementations. In service-oriented applications, there may be a need for different interface implementations where the same business functionality is available through different interfaces. For example internal consumers may be able to use an internal EJB interface, and external consumers will use a web service interface to the same functionality.

5. Conclusions and Next Steps

The implementation of an SPL using core assets implemented as services has significant potential. However, to gain the full potential requires making a set of pre-implementation decision points and engineering tradeoffs. The Framework for Product Line Practice offers a good starting point. It identifies 29 key product line practice areas. We have focused on four practice areas that have strong relevance for SOA services and have identified an initial set of decisions in these areas. We have also identified a set of potential variability mechanisms that have relevance for SPL core assets implemented as services.

In addition, proof-of-concept analyses of the relevance of specific technologies, tools and methods to the context for which they were developed can also be instrumental in building up a body of knowledge in the area [13]. For example, Sidharth Surana from the Carnegie Mellon University Master of Software Engineering program is currently conducting a proof-of-concept analysis of the relevance of different variation mechanisms for a simple product line example.

Future directions will require a validation and updating of the initial mappings, more complete mapping of services to SPL product line practices, and empirical research on actual SOA-based SPL implementations. This can ultimately lead to a codification of best practice for the use of SOA in the context of SPL.

6. References

[1] Clements, P. & Northrop, L. M. *Software Product Lines: Practices and Patterns*. Addison-Wesley, 2001.

[2] Northrop, L. & Clements, P. A Framework for Software Product Line Practice, Version 5.0 http://www.sei.cmu.edu/productlines/framework.html (2009).

[3] Lewis, Grace. *Service-Oriented Architecture (SOA).* SEI Webinar Series www.sei.cmu.edu/collaborating/spins/081408webinars.html

[4] Cohen, Sholom & Krut, Robert. Proceedings of the First Workshop on Service-Oriented Architectures and Product Lines (CMU/SEI-2008-SR-006). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2008.

[5] Cohen, Sholom & Krut, Robert. Managing Variation in Services from a Software Product Line Context. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2009 - forthcoming.

[6] Ralph Mietzner, Andreas Metzger, Frank Leymann and Klaus Pohl. Variability Modeling to Support Customization and Deployment of Multi-Tenant-Aware Software as a Service Applications. In Proceedings of PESOS Workshop, ICSE, Vancouver, Canada, May 18-19, 2009.

[7] Tarr. P. *Technologies for Software Product Line Development*. https://www-950.ibm.com/events/wwe/grp/grp004.nsf/vLookupPDFs/tarr-product-lines-033009-slides/$file/tarr-product-lines-033009-slides.pdf

[8] S. G¨unther and T. Berger, "Service-oriented product lines: Towards a development process and feature management model for web services," in *SPLC '08: 12th International Software Product Line Conference*, pp. 131–136, 2008.

[9] D. Chappell, *Enterprise Service Bus*, O'Reilly, June 2004.

[10] Organization for the Advancement of Structured formation Standards, "Web Services Business Process Execution Language Version 2.0", 2007, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html/

[11] Bass, Len; Clements, Paul; & Kazman, Rick. *Software Architecture in Practice*, 2nd ed. Boston, MA: Addison-Wesley, 2003.

[12] Lewis , Grace A.; Morris, Edwin J.; Smith, Dennis B.; Simanta, Soumya. SMART: Analyzing the Reuse Potential of Legacy Components in a Service-Oriented Architecture Environment (CMU/SEI-2008-TN-008). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2008.

[13] F. Hueppi, L.Wrage, and G. Lewis, "T-Check in Technologies for Interoperability: Business Process Management in a Web Services Context", CMU/SEI-2008-TN-005, *Software Engineering Institute, Carnegie Mellon University,* Pittsburgh, PA, June 2008.