RESOLVING APPLICATION DEVELOPMENT ISSUES USING SOA

Y. KIRAN KUMAR1, G. SUJATHA2, G. JAGADEESH KUMAR3

1Asst Professor, Dept of MCA, SVEC, A. Rangampet.
ykkumar83@gmail.com, sujatha229@gmail.com

ABSTRACT

SOA is an architectural style for building software applications that use services available in a network such as the web. Distributed service-oriented architectures allow system architects to create a distributed environment in which any number of applications, regardless of geographical location, can interoperate seamlessly in a platform and language neutral manner. Service-oriented architecture (SOA) is a set of principles for designing extensible, federated and interoperable services, and it represents a new evolutionary spiral in the program-applications development and in the evolution of the information systems concept. Development, evolution and maintenance of SOA-based systems demands rethinking of the traditional roles for performing these activities. The key objective of this paper is to present preliminary ideas on the roles required for developing, evolving and maintaining SOA-based systems.

Keywords: Service Oriented Architecture, Distributed services, Web Services, Modular Programming, Web Applications, Software as a Service.

1. INTRODUCTION

Distributed Service Oriented architectures allow system architects to create a distributed environment in which any number of applications, regardless of geographical location, can interoperate seamlessly in a platform and language neutral manner. SOA represents a model in which functionality is decomposed into small, distinct units (services), which can be distributed over a network and can be combined together and reused to create business applications. These services communicate with each other by passing data from one service to another, or by coordinating an activity between one or more services. It is often seen as an evolution of distributed computing and modular programming.

SOA is an architectural style for building software applications that use services available in a network such as the web. It promotes loose coupling between software components so that they can be reused. Applications in SOA are built based on services. A service is an implementation of well-defined business functionality, and such services can then be consumed by clients in different applications or business processes. In the traditional application architecture, and even in the more modern world of Web services, your infrastructure resources typically support a specific application or organization within your enterprise. As you plan for capacity and performance, you can test the limits of your system resources and infrastructure with a fairly good understanding of where the demand will come from and how it will be processed. With SOA, this type of planning and testing is more challenging because our infrastructure resources may support a community of users and applications through services spread throughout your enterprise.

2. SERVICE ORIENTED ARCHITECTURE

A Service Oriented Architecture (SOA) is a form of distributed systems architecture that is typically characterized by the following properties:

Logical view: The service is an abstracted, logical view of actual programs, databases, and business

Message orientation: The service is formally defined in terms of the messages exchanged between provider agents and request agents, and not the properties of the agents themselves. The internal structure of an agent, including features such as its implementation language, process structure and even database structure, are deliberately abstracted away in the SOA: using the SOA discipline one does not and should not need to know how an agent implementing a service is constructed. A key benefit of this concerns so-called legacy systems. By avoiding any knowledge of the internal structure of an agent, one can incorporate any software component or application that can be "wrapped" in message handling code that allows it to adhere to the formal service definition.

Description orientation: A service is described by machine-process able meta data. The description supports the public nature of the SOA: only those details that are exposed to the public and important for the use of the service should be included in the description. The semantics of a service should be documented, either directly or indirectly, by its description.

Granularity: Services tend to use a small number of operations with relatively large and complex messages.

Network orientation: Services tend to be oriented toward use over a network, though this is not an absolute requirement.

Platform neutral: Messages are sent in a platform-neutral, standardized format delivered through the
interfaces. XML is the most obvious format that meets this constraint.

A service is an abstract resource that represents a capability of performing tasks that form a coherent functionality from the point of view of provider’s entities and requester’s entities. To be used, a service must be realized by a concrete provider agent. In this paradigm, service providers register their service in a public registry. This registry is used by consumers to find services that match certain criteria. If the registry has such a service, it provides the consumer with a contract and an endpoint address for that service.

![Diagram of SOA's Find-Bind-Execute Paradigm](image)

To successfully build and deploy a distributed service-oriented architecture, there are four primary aspects to be considered:

1. **Service enablement**: Expose each discrete application as a service
2. **Service orchestration**: Configure distributed services and orchestrate the services in a defined distributed process.
3. **Deployment**: Move from the test environment to the production environment, addressing security, reliability, and scalability concerns.
4. **Management**: Audit, maintain and reconfigure the services. Make changes in processes without rewriting the services or underlying application.

### 3. CREATION OF A SERVICE ORIENTED ENVIRONMENT

The goal for a SOA is a worldwide mesh of collaborating services, which are published and available for invocation on the Service Bus. Adopting SOA is essential to deliver the business agility and IT flexibility promised by Web Services. These benefits are delivered not by just viewing service architecture from a technology perspective and the adoption of Web Service protocols, but require the creation of a Service Oriented Environment that is based on the following key principals:

- **Service** is the important concept. Web Services are the set of protocols by which Services can be published, discovered and used in a technology neutral, standard form.
- **SOA** is not just architecture of services seen from a technology perspective, but the policies, practices, and frameworks by which we ensure the right services are provided and consumed.
- **With SOA** it is critical to implement processes that ensure that there are at least two different and separate processes—for provider and consumer.
- Rather than leaving developers to discover individual services and put them into context, the Business Service Bus is instead their starting point that guides them to a coherent set that has been assembled for their domain.

![Diagram of Application development through layered approach](image)

### 4. SOA IMPLEMENTATION PROCESS

A SOA implementation is a process in which the organizational and information technical situation will change over time. Different phases in the implementation require different points of attention.

**Planning Phase**: In the planning phase must be determined if the adoption of a SOA may lead to organizational benefits. Therefore business requirements must be translated into web service requirements in terms of features.
While doing this, the business environment and the technology landscape must be taken into account. This phase can also include a financial analysis of the benefits and costs.

![Diagram of Phases of Service Oriented design and development methodology]

**Analysis phase**: The business requirements formulated in the planning phase are further refined and together with an elaboration on how SOA can meet them. This is done by creating an as-is model of the current business processes and current application portfolio to get consensus on the current state of business and its functionality. The as-is model will be improved by changing processes and the application portfolio in order to get a to-be model that will meet business requirements on which cost benefit analysis are applied. Also, the technical platform on which SOA will be based is chosen in this phase.

**Design phase**: In the design phase, well-documented interfaces are created for all major service components. The first task is to define how the services layers are designed so that they will standardize logic representation within the architecture. This is done by selecting basic standards that will support SOA to have characteristics suited for the organization. Then SOA extension specifications are chosen to even more support specific characteristics the SOA should have. Second, the services that will support the process are specified. A service specification contains a structural specification, a behavioral specification, and a policy specification. The first services that are to be specified are the services that represent data entities defined within an organization’s business model. Consequently, the basic services are specified that process demands dictated to them by composed and process services to support the data entities specified earlier. Then all the services are designed that provide support for the execution of the separate tasks within the process. When all the services are specified, the services are composed in process models by defining them in a business process language such as BPEL. Also, the business roles are defined in this step to be used by a business process language.

**Construction phase**: In the construction phase, the services are constructed according to the specifications created in the design phase.

**Testing phase**: Because of the generic nature of services and its likeliness of reuse, services should undergo extensive testing. Testing is characterized by ascertaining that services are according to standards defined during analysis, design, and implementation phase. Services can be tested in 5 ways: dynamic testing, functional testing, performance testing, interface testing, and assembly testing.

**Deployment phase**: In the service phase, the services and its processes are rolled out to all participants, including other enterprises and applications as well. There are three activities that take place in this phase: (1) Publishing the service interface in a service registry (2) Deploying the web services and (3) Integrating them with processes so that they are able to run and publishing service implementation details about where the service can be invoked.

5. SERVICE-ORIENTED DEVELOPMENT

Software vendors have widely adopted the paradigm of service-oriented development. Service-oriented development is complementary to the object-oriented, procedure oriented, message-oriented, and database-oriented development approaches that preceded it. Service-oriented development provides the following benefits:

- **Reuse** — The ability to create services that are reusable in multiple applications.
- **Efficiency** — The ability to quickly and easily create new services and new applications using a combination of new and old services, along with the ability to focus on the data to be shared rather than the implementation underneath.
- **Loose technology coupling**—The ability to model services independently of their execution environment and create messages that can be sent to any service.
- **Division of responsibility**—The ability to more easily allow business people to concentrate on business issues, technical people to concentrate on technology issues, and for both groups to collaborate using the service contract.

Developing a service is different from developing an object because a service is defined by the messages it exchanges with other services, rather than a method signature. A service must be defined at a higher level of abstraction than an object because it’s possible to map a service definition to a procedure-oriented language such as COBOL or PL/I, or to a message queuing system such as JMS or MSMQ, as well as to an object-oriented system such as J2EE or the .NET Framework.

6. **SUPPORTING BUSINESS OBJECTIVES WITH SOA**

SOA is not a new concept but a different approach to designing and building systems that are flexible and adaptable to support a dynamic business environment. An SOA lets you design, build, deploy and integrate services independent of applications and the computing platforms on which they run. These services are then linked together through defined business processes to form composite services, applications and composite applications to perform complete business functions. Some examples of possible services might include: locating billing information for a patient, requesting recent transactions for a financial account, identifying the owner of a registered vehicle, checking warehouse inventory for a particular item, or requesting a list of available flights for a given destination.

In this open framework, services can be shared and reused across several business processes. The result is a highly adaptive environment, with lower costs for application development, improved integration and quicker deployments. Business rules, which can be changed dynamically, now define the application layer of business functions. A single SOA-based service can, in fact, be widely reused throughout your enterprise by many business processes. And these business processes can be changed at any time to request other new and different services. Once you deploy SOA for your core business functions, the ability to dynamically add new capabilities through services can help reduce your development costs and almost eliminate traditional development cycles to more quickly deliver new customer services and open new market channels.

7. **IMPACTS OF SOA ON TODAY’S IT INFRASTRUCTURE**

SOA is usually realized through web services. Web services specifications may add to the confusion of how to best utilize SOA to solve business problems. In order for a smooth transition to SOA, managers and developers in organizations should know that:

- SOA is an architectural style that has been around for years. Web services are the preferred way to realize SOA.
- SOA is more than just deploying software. Organizations need to analyze their design techniques and development methodology and partner/customer/supplier relationship.
- Moving to SOA should be done incrementally and this requires a shift in how we compose service-based applications while maximizing existing IT investments.

A service may be widely used throughout the enterprise by many applications. SOA places additional performance and scalability requirements on the middleware infrastructure. While a primary benefit of SOA is that applications can be built by combining loosely coupled and interoperable services, the fact that a service may be widely used throughout the enterprise by many applications can lead to the following exposures for your IT infrastructure:

- Poor response times for users and business processes
- Missed service levels for critical business functions
- Noncompliance with industry and government regulations
- Security breaches
- Inadequate service management
- Governance gaps and limitations
- Testing challenges

Until now, your IT organization has most likely been focused on managing the infrastructure as an asset to support applications and business units. With SOA, our focus needs to shift toward managing the services which support business processes and, therefore, business results. SOA changes the role of the IT infrastructure. We will now need to manage your IT resources so that the services that support the business can leverage that infrastructure.

8. **ADVANTAGES OF SOA INTEGRATION**

When using SOA, organization gets a wide area of opportunities. These opportunities are defined by the
characteristics of Service Oriented Architecture. The main opportunity is the business respond speed on the changes of the environment and the evolution of business processes with minimal expenses. The main characteristics of Service Oriented Architecture are:

• **Week-binding.** Week-binding provides simple and quick adaptation of system to the changes in the structure and realization of the services. It gives opportunity on-the-fly reorganize business processes accordingly to the changes of the market (due to the week bindings between the services). Week-binding increases competitiveness because of the full confrontation of service and business process. It decreases price of implementation due to the high level of reusing of the services.

• **Modularity.** Module principle of building gives opportunity to organize graft on development, implementation and maintenance. This principle also gives opportunity for the gradual replacement of applications and hardware.

• “**SOA is not aggressive**”. SOA gives opportunity to use all investments, which were earlier invested in the information technologies, allows avoiding the reorganizing and retesting of existing systems when including them in SOA.

• **Standardized.** Platform-independence of SOA gives opportunity to choose any software and hardware. Standardization allows avoiding software lock-in, decreases complicity and fragmentation of the result from the using of closed products.

• **Technological.** One and the same technology can be applied for the wide spectrum of business problems. Oriented Architecture – architecture of information systems which is developed and based on week-binding of system parts principle. The smallest parts of such kind of systems are services. SOA is fully independent from programming languages, protocol specifications and platforms. The main reason of SOA appearance on the market is the replacement of coding of programs to assembly of applications from the standard parts.

**CONCLUSION**

An important aspect of SOA is the separation of the service interface (the what) from its implementation (the how). Such services are consumed by clients that are not concerned with how these services will execute their requests.

The positive result of SOA implementation can give the following advantages:

• Quicker and more flexible changes of business processes.
• Decrease of expenses for the IT operations.
• Quick implementation of updates and additional opportunities of software products.

The integration is a successful relation between data, applications, processes, people and organizations. From this point of view integration is a process, not technology. That is why while implementing SOA, it is important to think about the investment which the architecture will invest in improvement of business process from which depend existence and evolution of the organization. From the methodological point of view SOA needs an iterative approach. This is because the requirements are constantly changing. At this moment technological aspects are worked over much better than methodological questions of project realization with integration of SOA.

**REFERENCES**

5. Gold, N., Mohan, A., Knight, C.,