What is SOA? What is Different This Time?

by Paul S. Clarke

http://www.paulsclarke.com

There has been much buzz over the term service-oriented architecture (SOA). What is SOA? Is it just hype or is something really different this time?

A *service-oriented architecture* (SOA) is an architectural style based on a model of independent, distributed software-based services communicating with each other widely across the network. In the context of an SOA, a *service* is a network-accessible functionality that can be found, bound to, and used on demand. A service is defined in terms of an interface contract that is independent of its implementation. [McGovern et al 2003]

Based on this generic definition, the following features of an SOA can be deduced:[1]

- An SOA involves three main types of entities. A *service provider* registers its service with one or more *service registries*. A *service consumer* looks up a desired service in a service registry; upon finding and choosing a service, binds to it and invokes the service.
- Since service access is based on a contract-defined interface, services are modular, encapsulating, implementation-independent, and composable.
- Since services are in a networked registry, they are network addressable and composable *across the network*. Since services are network addressable, they could be on heterogeneous hardware or software platforms.
- Since services are dynamically found, bound, and used, they are dynamically discoverable and location transparent. Services need not be pre-wired together, but can be interconnected on demand or used on an ad-hoc basis if necessary.

Service-Oriented Architecture: What is Different This Time?

Of course, these sorts of systems have existed for some time; EJB, DCOM, and CORBA technologies facilitate this sort of distributed architecture. These technologies are based on the distributed object (or component) model; simply replace *service* with *object* and the above description could almost be talking about a distributed object architecture.

Advantages of SOA

Yet, there are real differences between service-oriented architecture and distributed-object architecture. First, SOA is less constrained: it does not require the notion of objects at all and allows us to choose our underlying implementation. Consequently, SOA represents a useful shift in viewpoint for the software industry: it shifts emphasis away from the technology of implementation (e.g. objects) to the conceptual level of the architecture (service-oriented). This focus on architecture is beneficial because architecture is finally receiving the emphasis it has always deserved.

Second, SOA facilitates an emphasis on business-oriented services that map directly to the business itself. This business-oriented frame of reference is a very valuable contribution of the SOA paradigm. Software is no longer defined only by the software boundaries of applications or components, but can be defined by business concepts aligned to focused business services, with each service having a specific business meaning.

Finally, the common terminology of *services* is closer than prior approaches to having the same meaning in the business and technical worlds; this is a big benefit if the business and technical staff can finally use a similar language. Although better communication between business and technical staff was one of the expected benefits of object-oriented software, it was not fully realized since an *everyday object* is actually quite different from a *software object*.[2]

Emergence of Web-Services-Oriented Architecture (WSOA)

Although a focus on services produces a conceptual change in frame of reference, it does not necessarily produce a substantive change to the technology of software implementation. Yet, the mechanism used for service implementation in SOA does matter, as a new form of SOA has now become viable: an SOA that is simpler to integrate and is more interoperable than prior versions. The key additional feature of this new SOA is:

- Services can be accessed and used through *universally accepted standards-based*, *platform-independent*, *programming-language-less*[3], *data-oriented* interfaces.

Data-oriented messaging technologies using a universally accepted text-based format (XML and related standards like SOAP, WSDL, XSLT, etc.) running over universally accepted communication protocols (HTTP, TCP/IP) make this possible.[4] This collection of technologies goes by the name of Web Services.[5] In this article, an SOA based on Web Services is given the name Web-Services-Oriented Architecture (WSOA).

Only recently have several key factors come together to make WSOA achievable. First, universal software industry acceptance of these standards have allowed them to permeate the software vendor community. Second, relentless progress in computing hardware performance compensates, in many cases, for the inefficiencies in using these technologies. Third, vast increases in network bandwidth capacity can now support the message sizes and quantities necessary for this architecture to succeed on a large scale.

Drivers of WSOA: Greater Business Integration and Flexible Business Evolution

WSOA addresses two critical needs of enterprises today: for far greater integration across the entire business[6] and for greater flexibility in modifying business operations. Greater integration and flexibility, in turn, must enable faster responsiveness to customers and to current conditions.

WSOA eases integration through a huge improvement in interoperability across the enterprise. Many enterprise architectures have evolved over the years, with a disparate set of silo-like applications on heterogeneous platforms throughout the enterprise. [7] WSOA alleviates software integration of these diverse applications, regardless of platform or programming language—heterogeneous applications can more easily exchange critical business information. Furthermore, XML-based data can readily enter traditional IT systems, since XML is a structured data type that can be mapped to documents and

database records. WSOA enhances flexibility by simplifying changes to service interfaces since compile-time dependencies are only on the data itself, and not on software code.

Conclusion

The benefits of WSOA are very tangible. This explains the explosion in interest in these technologies. This time the benefits are real. WSOA is here to stay.

References

Kreger, Heather, *Web Services Conceptual Architecture* (WSCA 1.0), IBM Software Group, May 2001, http://www.ibm.com.

McGovern, James, Sameer Tyagi, Michael Stevens, and Sunil Mathew. *Java Web Services Architecture*, Morgan Kaufmann Publishers, 2003.

---

[1] See [McGovern et al 2003].
[2] Software objects are loaded down with the mechanics of software, e.g. programming-language syntax and conventions, object lifetime semantics, etc.
[3] By *programming-language-less* is meant there are no programming language-specific bindings.
[4] Programming-language-less data-oriented messaging has been around for a long time; however, XML is the first universally accepted, standard, structured format for such messaging.
[5] [Kreger 2001] describes the Web Services software stack.
[6] Poor business integration was accentuated by the Internet; it was often obvious to customers when a Web site was poorly integrated with other business operations.
[7] There are good reasons for this. Independent silos can occur through mergers/acquisitions or the need for decentralized control and use of business data and logic; they are also easier to build and change.

