



# Service-Oriented Architecture and Legacy Systems

Nicolas Serrano, Josune Hernantes, and Gorka Gallardo

Enterprise systems are quickly evolving from monolithic silos to distributed applications with service-oriented flexible usage schemes. To keep up, IT organizations must adapt their legacy systems to meet changing business challenges almost in real time, with no second chances. Service-oriented architectures (SOAs) have evolved to flexibly operate and federate business processes and underlying systems. Authors Nicolas Serrano, Josune Hernantes, and Gorka Gallardo provide an overview of current SOA technologies and how to evolve in legacy environments. I look forward to hearing from both readers and prospective column authors about this column and the technologies you want to know more about. —*Christof Ebert*



**TODAY'S BUSINESSES MUST BE** able to flexibly and quickly adapt to market needs, but even minor changes to processes can involve rework in multiple IT systems that were originally designed as application silos. To stay competitive, maintenance efforts must be reduced, yet IT systems must continue to evolve.

Service-oriented architecture (SOA) enables the transition from a silo-based system to a service-oriented one. It facilitates loose coupling, abstraction of underlying logic, flexibility, reusability, and discoverability.<sup>1,2</sup> The SOA Manifesto outlines additional guiding principles ([www.soa-manifesto.org](http://www.soa-manifesto.org)).

## SOA Primer

SOA's novelty is in how it designs infrastructure architecture based on services

instead of focusing on entire applications. Services are small, discrete units of software that provide a specific functionality and can be reused in multiple applications. SOA applies a loose-coupling design principle, which means that each service is an isolated entity with limited dependencies on other shared resources such as databases, legacy applications, or APIs. It helps provide an abstraction layer between producers and consumers, which promotes flexibility in changing service implementations without impacting consumers.

SOA offers several benefits to businesses, but it isn't the best architectural choice for all cases. On the plus side, it

- provides a natural way to modularize complex systems by integrating

services from different vendors independent of platform and technology;

- promotes loose coupling, helping the interface with legacy systems;

there are other alternatives such as JSON);

- has numerous security vulnerabilities due to process-sharing applications and systems; and
- involves complex transaction

the most common way to implement a SOA—indeed, many systems use Web services without defining them as a SOA.

The main advantage of SOA (and therefore of Web services) is that the same service can be consumed by different clients. The data that was originally designed for a Web application to consume can be used, without any modification, by any other type of client. Examples include desktop applications that get data from a server without providing explicit SQL queries of the database or external systems such as a vending or public client information point that gets its data from a SOA service.

A legacy system can be wrapped as a SOA service and respond directly to the HTTP protocol or work behind a proxy that translates the request to the legacy system's language. Ultimately, the message in HTTP is plaintext, which any system or programming language can produce.

### Technologies

SOA is a good option for building more flexible applications, but choosing the right technology to achieve it will depend on your needs and environment. Let's review the most relevant technology considerations for organizations wishing to adopt SOA in their own business processes.

### SOAP vs. REST

When designing a Web service, we need to define the set of rules we're going to use to exchange information. The main tools for doing this today are SOAP and REST.<sup>3</sup>

SOAP is the older protocol; it was developed as an Internet-capable alternative to established technologies such as CORBA. SOAP can use vari-

For efficiency and flexibility we recommend an incremental transition to SOA in legacy environments.

- increases efficiency by allowing applications to be reused, thus reducing cost and development time;
- improves flexibility and scalability because multiple services can be easily developed from the integration of existing applications;
- allows a reduction of costs associated with maintenance;
- enables standards-based interoperability between systems;
- provides location independence to access data via any channel such as smartphones, tablets, or laptops; and
- allows an incremental approach to be taken, which helps meet customer demands faster by adding new services in response to specific business needs.

However, on the minus side, it

- is difficult to implement asynchronous communication between applications;
- is challenging to implement real-time responses or high data transfers because XML brings robustness, not speed (although

management in interactions between logically separate systems.

Moving to SOA isn't easy, and enterprises wishing to do so must be aware of the difficulties and inherent issues. Needless to say, every IT organization will experience multiple tradeoffs with SOA implementations; your mileage may vary. For efficiency and flexibility we recommend an incremental transition to SOA in legacy environments.

### Web Services

Web services are, for most organizations, the simplest approach for implementing a loosely coupled architecture. This interoperability is gained through a set of XML-based open standards such as WSDL, SOAP, and UDDI to provide a common approach for defining, publishing, and using Web services.

Web services evolved from Web applications. In fact, they're a simplification of Web applications: instead of serving the user interface along with the data, they only serve the data; the client application is in charge of presenting the information. Consequently, Web services are

ous transports (HTTP, SMTP, and so on), which gives it more flexibility. Data is exchanged in XML, so some performance issues could arise if the amount of information or messages transferred is high. SOAP can be used with Web Services Security, a standard for signing and encrypting messages that offers more secure information exchange.<sup>4</sup>

REST is a newer protocol that uses HTTP transport, but it can handle several data formats (XML, JSON, and so on); it relies on a specific URL instead of XML. REST is a lightweight alternative to SOAP. Because it doesn't impose a rigid implementation, it has a higher degree of flexibility, is lighter, and is less dependent on documentation. As REST can use GET methods, as opposed to the POST-only SOAP implementation, caching can happen not only in the service's design but also on the infrastructure side.<sup>5</sup>

Choosing REST or SOAP will depend on an organization's needs and constraints. Sometimes, you're better off with the enterprise capabilities that SOAP offers, or maybe you would benefit from a more performant and lightweight alternative such as REST. Because SOAP has better security and error-handling capabilities, most enterprise IT shops regard it as their preferred Web service implementation. Simplicity, performance, and a less rigid implementation all make REST the preferred implementation for those working on the collaboration APIs found in Internet services (see the sidebar for an example).

### Legacy Modernization

Although SOA architecture can be the best alternative for seamlessly integrating enterprise systems and avoiding the headaches of protocols



### EXAMPLE: GOOGLE APIS

Google's main business is search. We can use it directly through its homepage, but Google also pushed this function as a service through the Google Web Search API. Originally, this was done with SOAP, but the API was deprecated in December 2006 (although it worked until September 2009).

The company then announced a new generation that used JavaScript as the default environment to access the API, but Google exposed a RESTful interface that used the GET method, and the response was in JSON format. The new generation of Google Web Search API was deprecated in November 2010 although it's still available.

To replace this deprecated product, Google pushed a new product: Custom Search. This tool allows developers to create search engines that can be accessed with three different APIs: JavaScript, RESTful requests with JSON or Atom format, and an XML API.

Through this example, we can see the evolution from SOAP to REST APIs. In fact, this movement can be seen in most companies that provide public APIs.

and platforms, most people need to deal with an existing infrastructure. There's no perfect solution to the problem of modernizing legacy systems as you try to adopt a SOA architecture because a variety of aspects come into play. You need to take into consideration the current technological stack to make the optimal transition based on overall cost and risk.

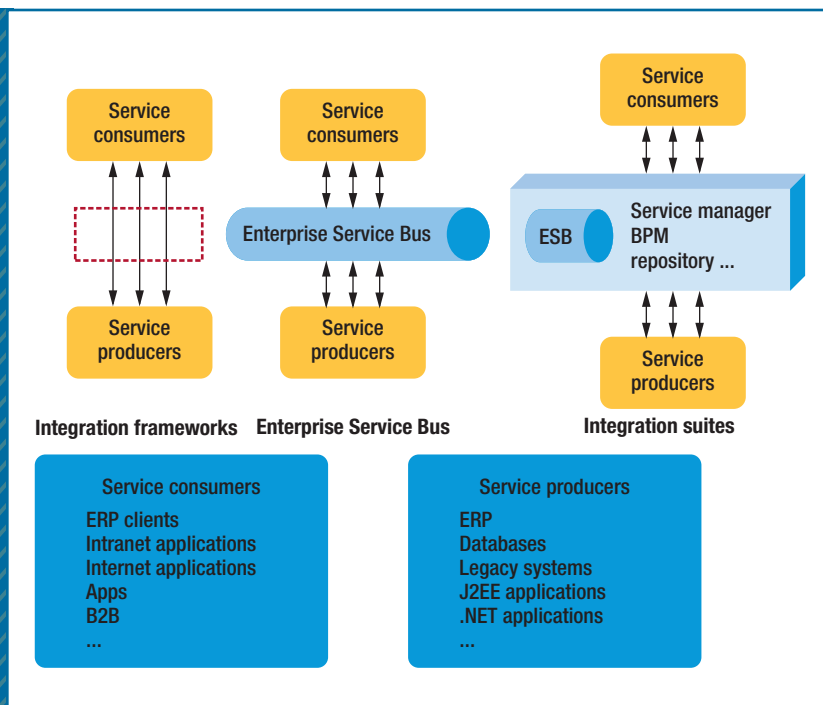
Because legacy systems usually support key business processes, a step-by-step change plan should be developed and a feasible evolution of the current systems using a hybrid approach should be designed to achieve a pure SOA architecture. There are several strategies for converting legacy systems to SOA.

The first approach is to replace the current legacy system with another system or set of systems. Normally, this is a good way to go if a current commercial off-the-shelf system (COTS) option matches the legacy system's requirements and func-

tionalities. This solution reduces maintenance but increases costs in future modifications.

A second alternative is to wrap the current legacy system with a middleware that can offer the legacy system interface through a Web service. In this way, legacy functionality is "wrapped" with a service layer and "plugged" into a SOA environment. This might not solve some of your problems: the legacy application can integrate several possible services, and you won't get the expected decoupling. However, it's a good option when the legacy system is expensive to rewrite, can be reused, and requires a cost-effective solution.

Finally, the third alternative is to redevelop or recode the current legacy system. This can be a very good approach, as you can act on the application architecture and achieve the optimal levels of decoupling. However, legacy applications are normally critical, and they're sometimes difficult or expensive to



**FIGURE 1.** Three alternatives to enterprise application integration: integration frameworks, Enterprise Service Bus (ESB), and integration suites.

change due to the old technologies involved and the lack of documentation, which could lead to some problems and increase the project risk. In this case, correctly assessing all the involved risks is a must.

### Enterprise Application Integration

When planning for application integration in any SOA initiative, many vendor products can help you ease into the transition. However, the different offerings range in capability and complexity, so selecting the right one is vital for success. You can divide the alternatives into three different groups based on the level of integration complexity (See Figure 1):

- Integration frameworks are the lightest of the alternatives and are basically formed by the libraries that implement APIs for
- Enterprise Service Bus (ESB) products offer integration framework capabilities plus tools for deployment, administration, and monitoring at runtime. ESB supports the connections between service producers and consumers, thereby offering the advantage of better tooling, which reduces cost and complexity significantly and solves integration problems at a higher abstraction level. Examples of ESB products are Oracle Service Bus and Mule ESB.
- Integration suites offer a complete stack that not only gives ESB capabilities but also more

different development environments. Examples of these integration frameworks are Apache Camel and Spring Integration in the Java environment and NServiceBus for .NET.

business-specific tools such as BPM (business process management), business activity monitoring, master data management, and a repository. All of these features help you respond to changes more quickly.

It's difficult to understand how competing solutions stack up against one another, so Table 1 compares these three integration solutions.

### Making Choices

Obviously, the best alternative depends on specific needs and complexity. First, you must decide whether a framework is sufficient. For instance, if you have only two applications to connect or you can work with a single technology (such as REST), you might opt for the simplest approach (integration framework) in spite of the scarcity of tools and support; otherwise, an ESB is the better choice. However, if additional features are required, you might be better off with a more capable and complex stack such as an integration suite.

**T**he next evolutionary step forward will be how to make the convergence of SOA and cloud computing easier. The rise of the cloud has brought advantages to enterprise: cloud computing provides resources you can leverage on demand, including those that host data, services, and processes. You can extend your SOA outside of the enterprise firewall to cloud computing providers.

Thus, cloud integration is one of the main challenges enterprises are dealing with today. In this context, iPaaS (integration platform as a service) is emerging as a suitable option


TABLE 1

Comparison of solutions for enterprise application integration.

|  | Integration framework | ESB    | Integration suite |
|--|-----------------------|--------|-------------------|
| Integration complexity   | Low                   | Medium | High              |
| SOA-enabled  | Yes                   | Yes    | Yes               |
| Application server   | No                    | Yes    | Yes               |
| ESB (Web services, message transformation, protocol mediation) | No                    | Yes    | Yes               |
| Business process management                                    | No                    | No     | Yes               |
| Business activity monitoring                                   | No                    | No     | Yes               |
| SOA security   | No                    | No     | Yes               |
| SOA management   | No                    | No     | Yes               |

for a wide range of integration needs. iPaaS is a suite of cloud services that enable users to create, manage, and govern integration flows connecting a wide range of applications or data sources without installing or managing any hardware or middleware.

Looking to the future, research firm Gartner predicts that by 2016, at least 35 percent of all large and mid-size organizations worldwide will be using one or more iPaaS offerings in some form. However, experts opine that iPaaS won't replace SOA: tradi-

tional SOA will still be required for complex integration scenarios such as low-latency messaging and data-intensive transactions within and between enterprises. 

## References

1. N. Gold et al., "Understanding Service-Oriented Software," *IEEE Software*, vol. 21, no. 2, 2004, pp. 71–77.
2. S. Jones, "Toward an Acceptable Definition of Service," *IEEE Software*, vol. 22, no. 3, 2005, pp. 87–93.
3. S. Mumbaikar and P. Padiya, "Web Services Based on SOAP and REST Principles,"

*Int'l J. Scientific and Research Publications*, vol. 3, no. 5, 2013, pp. 1–4.

4. P. Louridas, "SOAP and Web Services," *IEEE Software*, vol. 23, no. 6, 2006, pp. 62–67.
5. S. Vinoski, "REST Eye for SOA Guy," *IEEE Internet Computing*, vol. 11, no. 1, 2007, pp. 82–84.

**NICOLAS SERRANO** is a professor of computer science and software engineering at the University of Navarra's School of Engineering. His research interests include information technology and its application to personal and professional development. Contact him at [nserrano@tecnun.es](mailto:nserrano@tecnun.es).

**JOSUNE HERNANTES** is a professor of computer science and software engineering at the University of Navarra's School of Engineering. Her research interests include software engineering and information systems. Hernantes received a PhD in computer science from the University of Navarra's School of Engineering. Contact her at [jhernantes@tecnun.es](mailto:jhernantes@tecnun.es).

**GORKA GALLARDO** is a professor of information systems at the University of Navarra's School of Engineering. His research interests include information technology. Contact him at [ggallardo@tecnun.es](mailto:ggallardo@tecnun.es).

# IEEE Intelligent Systems

THE #1 ARTIFICIAL INTELLIGENCE MAGAZINE!

*IEEE Intelligent Systems* delivers the latest peer-reviewed research on all aspects of artificial intelligence, focusing on practical, fielded applications. Contributors include leading experts in

- Intelligent Agents • The Semantic Web
- Natural Language Processing
- Robotics • Machine Learning

Visit us on the Web at  
[www.computer.org/Intelligent](http://www.computer.org/Intelligent)



Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.

Copyright of IEEE Software is the property of IEEE Computer Society and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.