A Practical Perspective on the Design and Implementation of Service-Oriented Solutions

Alan W. Brown, Marc Delbaere, and Simon K. Johnston

IBM Software Group
awbrown@us.ibm.com,delbaere@be.ibm.com,
skjohn@us.ibm.com

Abstract. Business-driven development is an approach that focuses on automating the path from business understanding to IT solution. IBM's experiences with customers taking a business-driven approach to develop services-oriented solutions are highlighting a number of best practices that are important to share and discuss. This paper focuses on how companies adopting a service-oriented approach are assembling the appropriate environment to be successful. The paper identifies three design techniques for SOA and describes when each of them can be used in practice, depending on the business and IT drivers and the organization's maturity. We then highlight how to use structured enterprise models together with the tools and methods to automate the design of service-oriented solutions. These scenarios and examples are playing an important role in the development of future method content and tooling requirements for IBM Rational tools.

1 Introduction

Intense pressures on developing software-intensive solutions in many business domains are the result of several convergent factors, including [1,2]:

- End users want more timely, synchronized information to be available everywhere, on demand, with no downtime.
- To be competitive, enterprise IT organizations are taking advantage of lower labor rates around the world, integrating components from a variety of suppliers, and reusing solutions across product lines and solution families.
- The fast pace of business change is leading to a greater number of enterprises focused on agile delivery of on time solutions that are "good enough" rather than miss key business deadlines.
- Today's distributed solutions platforms offer a rich collection of distributed infrastructure technologies allowing greater flexibility in how core application capabilities are deployed, optimized, and redeployed as the business needs change.

As a result, existing software design practices are under pressure to support different design styles, architectures, and team structures. The new solutions, often described under the broad umbrella of Service-Oriented Architectures (SOA), exhibit

G. Engels et al. (Eds.): MoDELS 2007, LNCS 4735, pp. 390–404, 2007.

common characteristics: a focus on business/IT alignment, support for assembly of components, and recognition that flexibility in design is paramount [3]. A great deal of attention is being given to technology support for SOA.

However, while many technology aspects of SOA solutions have been discussed in detail, there are remarkably few analyses of the kinds of design techniques organizations are adopting to create an SOA. Indeed, where these topics are addressed, they often devolve into broad generalization of interface-based design approaches (e.g., [4, 5]), or technology-specific tutorials linked to a specific implementation technology (e.g., [6, 7]).

This paper describes real-world design techniques applied by IBM customers when developing business applications in a service-oriented style. Following a discussion on design principles for SOA, this paper describes three categories of design techniques that span the services-oriented solutions we have seen. We then further elaborate services-oriented design approaches by describing a model-driven approach to SOA that uses these techniques based on a substantial set of industry models used in a range of enterprise application in the financial services domain.

2 Design for SOA

The goal of service-oriented solutions is to provide the flexibility to treat elements of business processes and the underlying IT infrastructure as secure, standardized components – typically Web services – that can be reused and combined to address changing business priorities. They enable businesses to address key challenges and pursue significant opportunities quickly and efficiently.

To achieve this goal solutions adopt a specific architectural style [8] that focuses on collaborations among independent providers of capabilities. By considering this style of solution, and its common realization as web services, we recognize a set of characteristics for services that inform us of their nature and applicability. Furthermore, efficient use of services suggests a few high-level best practices, particularly in relation to more established object and component-based design approaches. In particular, services are typically:

- **Coarse-grained** — operations on services are frequently implemented to encompass more functionality and operate on larger data sets, compared with component-interface design.
- **Defined via their interfaces**— services implement separately defined interfaces. The benefit of this is that multiple services can implement a common interface and a service can implement multiple interfaces.
- **Discoverable** — services need to be found at both design time and run time, not only by unique identity but also by interface identity and by service kind.
- **Instantiated as a single instance** — unlike component-based development, which instantiates components as needed, each service is a single, always running instance that a number of clients communicate with.
- **Loosely coupled** — a SOA is loosely coupled architecture because it strictly separates the interface from the implementation. In addition, runtime discovery further reduces the dependency between service consumers and providers and makes SOA even more loosely coupled. Services are connected to other services