Service-Oriented Architecture in Application and Integration Projects

Lothar Wieske

Enterprise Application Architecture
Zühlke Engineering GmbH
Düsseldorfer Str. 40a
D-65760 Eschborn (Frankfurt)
lwi@zuehlke.com

Abstract: The emergence of service-oriented architecture (SOA) as the foremost platform for contemporary business application and integration solutions offers enterprises new levels of flexibility and agility. In this contribution we compiled major project experiences of early adopters into 13 best practices for SOA evolution at the application and/or integration level.

Introduction

The emergence of service-oriented architecture (SOA) as the foremost platform for contemporary business application and integration solutions offers enterprises new levels of flexibility and agility. Experiences of early adopters in service-oriented application and integration projects may help fast followers in reducing risk, complexity and cost of introducing, implementing, and evolving SOA in their companies. In this contribution we compiled major project experiences into a set of 13 best practices for SOA adoption at the application and/or integration level.

General Best Practices

1. Control your XML and SOA technology adoption. Identify and recognize knowledge gaps and use consulting, training, and mentoring as methods to fill them. Setup enterprise-wide roadmaps for the intra-application and inter-application adoption of current and emerging specifications and technologies.

2. Make XML standardization, data modeling and domain-driven design a synergistic endeavor. Integrate people from several functional departments and over several hierarchical levels into a task force to form a seed crystal for your mission-critical SOA initiative. In this cross-functional and cross-hierarchical team strive for an enterprise-wide service-oriented security model guiding future application development and integration activities.

Intra-application SOA Adoption Best Practices

3. Make interoperability requirements a mandatory part of your requirements engineering process. Consider functional requirements as well as internal and external systemic qualities.

4. Systems management becomes an integral part of your SOA application architecture. Define and prototype it early, incorporate it into your test strategy, and test it thoroughly.

5. Services can be phased in at different complexity levels. Define and keep a concise strategy for the development of basic services and reliable/transacted/secure services.

6. Consider knowledge management and project marketing as a non-technical means to open up your application and make your services used.

Inter-application SOA Adoption Best Practices

7. Distinguish service categories. An initial model might give you data, application, business, and presentation services as functional categories and event, instrumentation, and lifecycle services as systemic categories as candidates. Establish a reference architecture.

8. Distinguish service domains. Packaging and visibility are major topics in every programming language. Your service-oriented security model should lay the foundation for appropriate service domains.

9. Systemic qualities must be understood at the intra-application and at the inter-application level as well. Don't integrate any application without full definition and understanding of its systemic qualities and the applications to be integrated.

10. Train you architects, designers, and developers by studying existing SOA implementations like tomcat/axis or mule.

11. Consider the organizational impact of being service-oriented not only as a technological metaphor but as a chance for a major shift in organization culture, structures and styles.

Portfolio- and Programme-Management Best Practices

12. Start with 3-5 intra-application SOA adoption projects, prepare at least two of them for inter-application SOA adoption by offering external services.

13. Consider viewing services as products, define service life cycles and follow product management best practices for internal service management. (There's a difference between usable, useful and used services.)