zapthink zapnote

Doc. ID: ZTZN-1200
Released: Sept. 15, 2006

ZAPTHINK ZAPNOTE™

AVOIDING ACCIDENTAL SOA
HOW TO BUILD THE RIGHT *SERVICES*

Analysts: Jason Bloomberg and Ron Schmelzer

Abstract

Today, it seems that every IT shop around the world is building Services. Whether those Services are standards-based Web Services or other kinds of software Services, exposing existing functionality and data via Service interfaces has become the leading approach to building distributed systems. As companies develop an increasing array of Services, they soon face the challenge of organizing those Services into an architecture that can respond effectively to ever-changing business requirements. We call such architecture Service-Oriented Architecture (SOA). While a few architecturally savvy companies begin their Service implementation initiatives by planning the Services they will build ahead of time, most companies approach SOA by first building several Services, and tying them together and calling the result an SOA, without an architectural plan as the starting point.

While such an "accidental" SOA is within the realm of possibility, getting SOA right is more challenging than simply developing and collecting a bunch of Services. It's critical to have a high-level plan that can guide the whole SOA initiative, so that your step-by-step efforts don't end up with a collection of Services that don't meet long-term business needs. At the same time, it's essential that you ground the architectural plan in the realities of the existing technology infrastructure. Therefore, it's important to understand how to balance the long-term plan with the tactical steps you should take to make SOA a success.

Balancing Top-Down and Bottom-Up

To create this balance, we're able to flesh out two intertwining approaches to SOA, which we call top-down and bottom-up. In the top-down approach, architects get involved with the SOA project by putting together a long-term architectural design. It's important to have the right level of detail in this plan, since too much detail can slow down the project, and too little can lead to a poorly defined architecture.

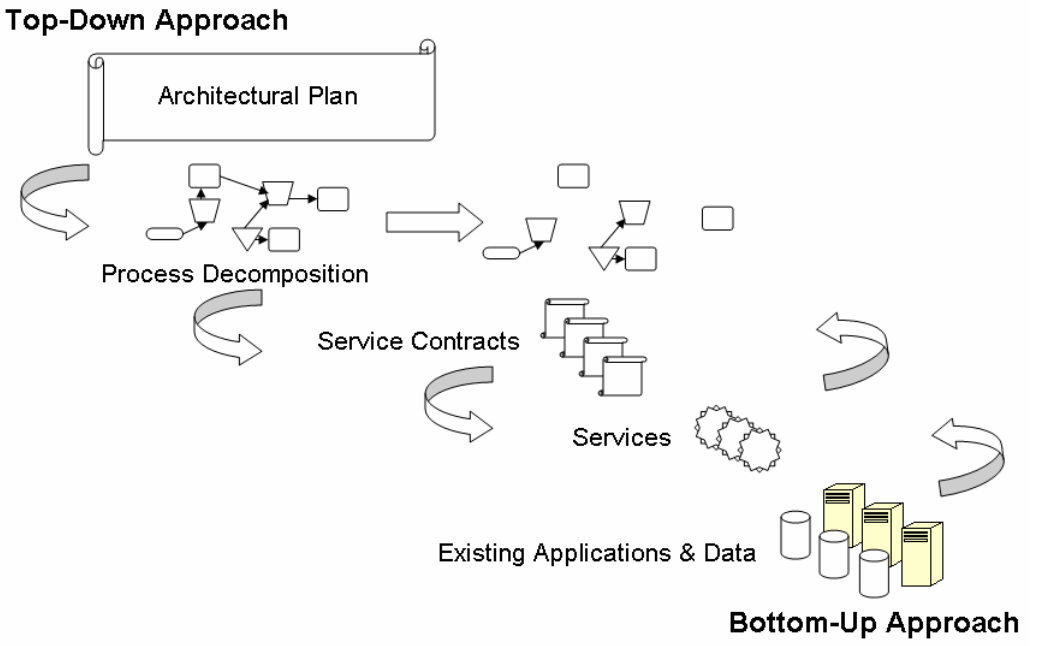
Once the architects finish crafting their first pass at the high-level plan, the next step in the top-down approach is process decomposition, which seeks to take existing business processes and break them up into subprocesses with an eye to identifying the right Services to build. At that point, you can proceed to detail the Service contracts for those Services, and the contracts in turn drive the development of the software that underlies the Services. You should decompose processes iteratively by completing part of the decomposition, defining the Services to some extent, and then repeating the steps to maximize reuse of the component Services across multiple processes.

Concurrent with the top-down approach is the bottom-up approach to SOA, which exposes existing IT functionality as Services in order to compose them into business processes. The bottom up approach by itself is a good start, but isn't SOA, because it doesn't include an architectural plan. The problem is that neither of these approaches is sufficient on their own. Taking the top-down approach alone may result in specifying a collection of Services that are beyond the capabilities of the existing infrastructure, while taking only the bottom-up approach typically results in a collection of redundant Services that do not provide the benefit of reuse.

At the crux of the problem is the issue of Service granularity. On the one hand, the top-down approach leads to the definition of business-oriented Services that are coarse grained, in that they have the flexibility and breadth to provide value within a business context. The bottom-up approach, in contrast, more likely leads to fine grained, basic Services that expose some specific IT capability. As Joe Gentry, VP of Enterprise Transaction Systems at Software AG says, "Generally the first step customers take is to create basic Services—with a focus on the service methods and granularity. Once success and ease-of-use of the services has been proven, the number of Services soon grows substantially. At that point we talk to architects more than developers."

The obvious answer to the challenge of Service granularity is that your approach to SOA should be both top-down (through process decomposition) as well as bottom-up (exposing existing functionality as Services), as shown in the figure below. As you begin your SOA rollout, it's important to identify those IT resources you wish to include in your Service-building activities. However, don't simply build Services based on the capabilities you already have—think about the Services you need in order to make your pilot a success.

The Top-Down and Bottom-Up Approaches to SOA



Source: ZapThink

The Service Contract

As the figure above shows, Service contracts are at the crux of the two SOA approaches. A Service contract specifies the requirements of Service consumers and providers, abstracting the implementation while clearly defining the requirements for the Service. Because Service contracts should drive the development of the software that underlies the Services, it's important to have a clear understanding of what contracts are and how they work. Gentry points out, "Contracts are crucial. It's important to follow a proper services design cycle up front. The failure to follow good services design principles tends to result in service interoperability issues and multiple changes that impact the stability and reusability of services. Start by establishing fundamental guidelines, such as specifying metadata to capture, defining service interfaces, and modeling the service use case and role." Just as legal contracts make for solid business relationships, Service contracts make for solid, loosely coupled architectures. Here's what goes into a Service contract:

- Contracts should describe what a Service provider will give to any Service consumer that chooses to abide by the terms of the contract. The contract should define what functionality the provider offers, what data it will return, or typically some combination of the two.
- Contracts must delineate the responsibility of the Service providers for providing their functionality and/or data, as well as the expected responsibilities of the Service consumers and what they will need to provide in return. In other words, they specify a Service level as part of a Service Level Agreement (SLA) that the Service providers must adhere to.
- Contracts also specify the rules of engagement between consumers and providers (known as policies) that govern who can access a provider, what security procedures the participants must follow, and any other rules that apply to the exchange.

In short, contracts specify policies, processes, procedures, and governance as they apply to the Services and their consumers. What contracts never include are the data that providers

and consumers actually exchange, or any specifics about how a provider or a consumer will go about meeting the requirements of the contract. Instead, contracts consist solely of information about those specifics, in other words, metadata. Contracts, therefore, consist of metadata, because all the information in the contract refers to how the provider and consumer are to deal with the data they wish to exchange.

In SOA, contracts enable us to move away from the typical way of making software work by coding its functionality into computer programs, instead moving toward describing how software should work with metadata. Essentially, if all we have to do to change how a complex, distributed application behaves is to change some of the metadata, then we no longer have to bring in the programmers and make expensive, risky changes to the programming. Repositories provide a central store for such metadata, and switching around the metadata takes far less time and effort than reprogramming, and is also far less dangerous—you're much less likely to break the application by changing metadata. You can think of metadata as a kind of lubricant that makes the various pieces of a complex IT situation move against each other more smoothly.

Understanding Top-Down Service Development

Service contracts are critical for enabling one of the major tenets of SOA: the ability to reuse Services as broadly as possible to support changing business requirements. As companies adopt SOA, they will gradually shift over time from focusing on creating Services to thinking about reusing Services by composing them in different ways to meet different business needs. Rather than spending time and money trying to figure out which new Services to build, companies will focus their efforts and resources identifying existing Services in the enterprise, and composing them into new processes or composite applications.

Companies that have properly defined their Services in a top-down fashion by starting with an overall architectural plan, rather than building their SOA from the bottom up by building Services as legacy wrappers, will be better able to repurpose their Services. Therefore, companies that develop composite Services will be able to extract continuing value from their Service investments. On the flip side, companies that only focus on Service development, rather than composition and reuse, will hardly get the chance to benefit from the agility that SOA promises.

Some companies are already enabling Service-oriented processes by creating composite applications of Services, while others are making the transition from bottom-up Service development to the architectural design that's the essential precursor of such applications. As Gentry points out, "SOA is about business agility, and the second goal from the IT perspective is reusability. Reuse means more money for less maintenance."

The most significant question that architects must grapple with is which Services to build from scratch and which to repurpose to meet changing needs. The secret to building the right Services, therefore, centers on creating reusable, coarse-grained business Services that the organization can reuse by composing them into different processes. The key, in turn, to creating such Services lies in the Service contracts—metadata that describe how Service providers and consumers should interact. In essence, SOA moves the hard work of building distributed applications from the programming code to the metadata, providing both greater flexibility as well as greater control over applications for the business.

Profile: Software AG	September 2006

Funding:
Trades on Frankfurt Stock Exchange
CEO:
Karl-Heinz Streibich
Employees:
2700
Address:
US Headquarters:
11700 Plaza America Drive
Suite 700
Reston, Virginia 20190

Worldwide headquarters:
Uhlandstrasse 12
64297 Darmstadt
Germany
URL:
http://www.softwareag.com
Phone:
877-724-4965
+49-6151-92-0
Contact:
info@softwareag.com

Related Research

- *Process-Driven SOA: Leveraging Service-Oriented Architecture for Business Process Innovation* White Paper (WP-0143)
- *ZapForum Podcast: Practical Advice for the SOA Registry/Repository Buyer* ZapForum Podcast (ZTP-0228)
- *NetManage* ZapNote (ZTZN-1174)
- *Seagull Software* ZapNote (ZTZN-0160)
- *Software AG* ZapNote (ZTZN-1156)

About ZapThink, LLC

ZapThink is an IT advisory and analysis firm that provides trusted advice and critical insight into the architectural and organizational changes brought about by the movement to XML, Web Services, and Service Orientation. We provide our three target audiences of IT vendors, service providers and end-users a clear roadmap for standards-based, loosely coupled distributed computing – a vision of IT meeting the needs of the agile business.

ZapThink helps its customers in three ways: by helping companies understand IT products and services in the context of Service-Oriented Architecture (SOA) and the vision of Service Orientation, by providing guidance into emerging best practices for Web Services and SOA adoption, and by bringing together all our audiences into a network that provides business value and expertise to each member of the network.

ZapThink provides market intelligence to IT vendors and professional services firms that offer XML and Web Services-based products and services in order to help them understand their competitive landscape, plan their product roadmaps, and communicate their value proposition to their customers within the context of Service Orientation.

ZapThink provides guidance and expertise to professional services firms to help them grow and innovate their services as well as promote their capabilities to end-users and vendors looking to grow their businesses.

ZapThink also provides implementation intelligence to IT users who are seeking guidance and clarity into the best practices for planning and implementing SOA, including how to assemble the available products and services into a coherent plan.

ZapThink's senior analysts are widely regarded as the "go to analysts" for XML, Web Services, and SOA by vendors, end-users, and the press. Respected for their candid, insightful opinions, they are in great demand as speakers, and have presented at conferences and industry events around the world. They are among the most quoted industry analysts in the IT industry, and their recent book, *Service Orient or Be Doomed!,* is the leading book on the business concept of Service Orientation.

ZapThink was founded in October 2000 and is headquartered in Baltimore, Maryland. Its customers include Global 1000 firms and government organizations, as well as many emerging businesses. Its analysts have worked at such firms as IDC, marchFIRST, and ChannelWave, and have sat on the working group committees for standards bodies such as RosettaNet, UDDI, and ebXML.

Call, email, or visit the ZapThink Web site to learn more about how ZapThink can help you to better understand how SOA will impact your business or organization.

ZAPTHINK CONTACT:
ZapThink, LLC
108 Woodlawn Road
Baltimore, MD 21210
Phone: +1 (781) 207 0203
Fax: +1 (786) 524 3186
info@zapthink.com
www.zapthink.com