Service-Oriented Architectures: Potential and Challenges

Prof. Dr. Alexander Schill
TU Dresden, Computer Science
01062 Dresden, Germany
www.rn.inf.tu-dresden.de

Abstract

The paper gives an overview of the service-oriented approach, focusing on its general strategic issues as well as on its technical foundations using WebServices. Based on example scenarios, the overall potential of this emerging technology is outlined with respect to open systems integration, dynamic binding, and flexible management of distributed systems.

Simultaneously, various research and development challenges are elaborated, ranging from heterogeneous data integration via security issues towards future teleservices. In particular, quality of service enhancements and mobile systems and network support are to be addressed. Moreover, the relationship of service-oriented architectures and grid computing, and current efforts towards their synthesis are discussed in order to outline further research areas.

Introduction

Distributed systems and their applications become more and more important for industry. Their traditional advantages include flexible, decentralized administration, scalability, replication and fault tolerance of critical system components, and speedup by parallel execution. However, traditional architectures of distributed systems impose limitations concerning their openness, their possibilities to integrate heterogeneous systems, and their flexibility in terms of reconfiguration and dynamic restructuring. Therefore, service-oriented architectures have gathered significant attention recently. They offer a distributed, discovery-based model to expose and manage a collection of software assets as services in a highly dynamic way /1/.

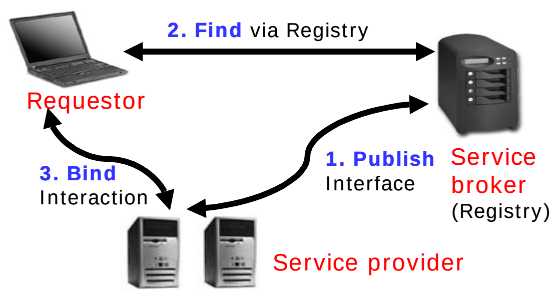

Figure 1: Basic structure of a service-oriented architecture

As shown in figure 1, a service provider offers various services at the application level via an intranet or also via the internet. These services are being registered with service brokers in distributed registries, and can be searched for by requesting clients. When service interfaces and client demands match, a dynamic binding is performed, and the client can interact remotely with the service. This model is rather flexible based on dynamic service selection, variable service orchestration, and possibilities for simple reconfiguration. It also offers new options for integrating heterogeneous services based on universal XML interface descriptions, simple XML-based data integration and mapping, and architectural integration of backend legacy services via middleware and application servers.

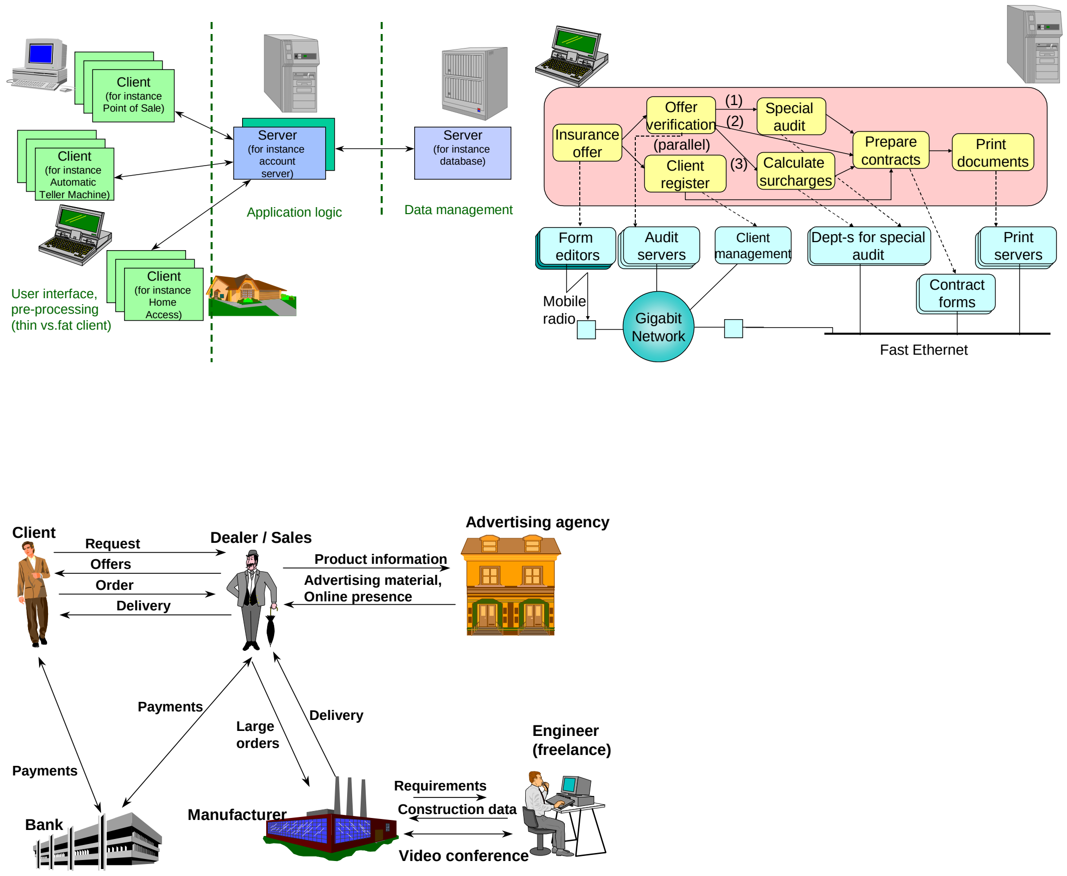

Figure 2: Scenarios of applying service-oriented architectures

The service-oriented model can be applied in various ways, ranging from closely coupled services via flexible compositions of complex business services to a highly dynamic business web structure.
This is illustrated in figure 2: Scenario (1) represents a 3-tier architecture of a banking application with closely coupled services. A client / user interface layer accesses application services of a middle tier for account management, and the actual account data are processed within databases of the data management tier. The whole application runs within an intranet with tightly coupled services, offering selective access via internet access portals.
In scenario (2), a complex business service as part of an insurance application is shown. It represents a complex workflow with a number of sequential and parallel tasks for processing an insurance application. Each task is being mapped onto a basic service of an underlying network and server infrastructure. Service coupling is more flexible and dynamic than in the first scenario, but still the whole application is executed within a single organizational domain.
In scenario (3), however, a completely dynamic business web is set up. Clients, sales agencies, manufacturers, external freelancers (engineers, advertising agencies etc.), and banks interact in a highly flexible way in order to perform business transactions. All participants can be selected dynamically based on service offers and requests via service brokers, participants can be replaced by others, interacting services and their interfaces are largely heterogeneous, and even advanced interaction techniques such as video and audio are being employed. A further step might be the so-called “agile company” offering an even more dynamic and comprehensive service integration, also with customers and partner enterprises etc. In

summary, the service-oriented model and architecture offers a rather new dimension of structuring and configuring distributed systems and applications in an advanced way.

Basic Technology

Service-oriented architectures require various basic implementation technologies. Their major foundation is given by web services as standardized by W3C /2,3/. This approach (see figure 3) offers a remote invocation mechanism between objects, the simple object access protocol (SOAP). It represents a remote procedure call based on internet protocols (http) and based on a universal data encoding mechanism using XML. The interfaces of participating servers are specified in the XML-based web services definition language (WSDL); such interfaces can easily and automatically be mapped onto all kinds of major current and legacy programming languages such as Java, C++, C, Fortran, Cobol, and many others. The service brokerage is implemented by a universal description, discovery and integration service (UDDI); it represents a global, network-wide directory infrastructure for registering and locating application services.
After registering, searching and locating a remote service, it is invoked via SOAP, the invocation is passed to a web services runtime infrastructure, possibly security-checked via an application-level firewall, and then passed to the actual backend business application. This is typically implemented by an application server using Java component technology or related approaches, and can be further distributed onto various machines internally for load balancing etc.

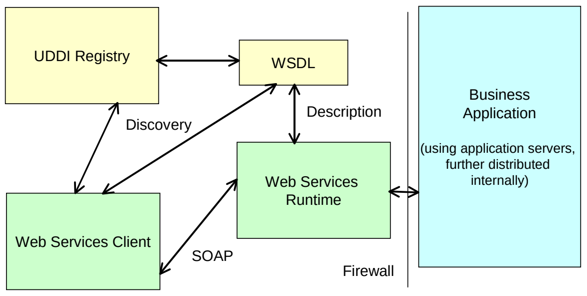

Figure 3: Basic technology for service-oriented architectures

```
<binding name=“BankServer“ type=“tns:BankServerPortType“>
  <soap:binding style = “rpc“ transport = “http://... “/>
  <operation name = “getBalance“>
    <soap:operation soapAction = “urn:xmethodsBankServer“/>
      <input>
        <AccountIdentification>
                <AccountNumber>3044005</AccountNumber>
                <pin>****</pin>
                <name>John Smith</name>
        </AccountIdentification>
      </input>
      <output> ... </output>
  </operation>
</binding>
```

Figure 4: WSDL example

Figure 4 shows an example of a WSDL interface specification in XML in order to illustrate the linguistic features of this language. A remote procedure call (RPC) invocation mechanism

is specified, the remote operation is given, and its signature with a detailed parameter structure and type specification is detailed. As an alternative to a request-response-based remote procedure call, asynchronous oneway invocations or server-to-client notifications are also possible.
Recently, several enhancements have been standardised or have been proposed as draft standards; examples are integrated security techniques (web services security), reliable messaging, asynchronous notifications, or high-level coordination protocols. Moreover, a higher-level business process execution language for web services (BPEL4WS) was also specified; it enables the integration of complex services consisting of underlying tasks as described in the scenarios (2) and (3) of figure 2 above.

Current Trends and Research Challenges

While service-oriented architectures present a new and flexible model for distributed processing, they also impose a number of additional research issues and of options for further development and evolution.

Grid computing

First of all, a related major trend is grid computing /4/; associated distributed processing models enable the network-wide usage of parallel processing resources in order to speed up processing of complex computational tasks in scientific applications or in business processing. The emphasis of grid computing – as opposed to service-oriented architectures – is more on parallelisation of applications, and on systematic use of idle resources in an overall network environment, or even within the global internet. However, service-oriented technology can be integrated with grid computing: A grid resource broker, potentially implemented by UDDI, is used for registering grid resources, and client applications can dynamically bind to such remote resources this way. Interactions can be performed via SOAP or related protocols, and XML-based interface notations such as WSDL can also be applied to grid computing. This is sketched in figure 5 based on the notion of a world-wide grid scenario with a number of computing resources Ri.

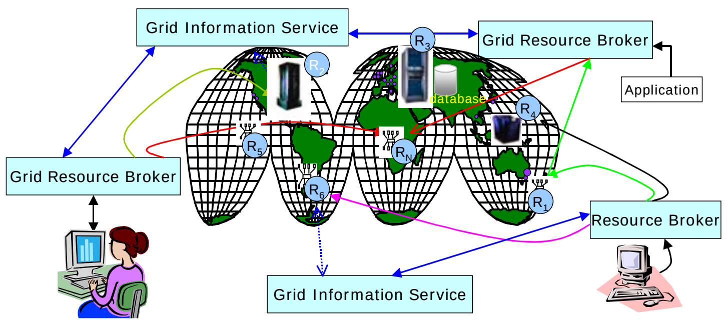

Figure 5: World-wide grid environment enabled by service-oriented architectures

Major open research issues in such a scenario are how to guide service selection not only by syntactic interface matching but rather by semantic service description and semantic matching of appropriate services. This requires advanced semantic specification techniques, and possibly the notion of ontologies known from software reuse can be applied towards such a flexible matching mechanism. Another open issue includes the problem of adapting and integrating global security and privacy policies: In some cases, it is rather desirable to access powerful backend resources for high-performance computing, but necessary security restrictions might prevent client processes from doing so, even with correct authentiation. Therefore, flexible security mechanisms with continously tight access control, but with the option of controlled limited access to grid resources are desirable and are to be further investigated.

Distributed multimedia
Another major trend is the integration of synchronous multimedia interactions using video and audio with more traditional distributed processing, now also based on service-oriented architectures. Figure 6 shows a multipoint videoconferencing system and its related internal architecture, the VidConference system /5/. It enables multi-party audio and video conferencing with adaptive video quality, dynamic sharing of applications, and dynamic integration of conference partners. The video and audio encoding is done completely in software, facilitating upgrades and dynamic installation. A conference server is used for registering users, for access control, and also for scaling and distributing video and audio streams. All interactions are controlled by a dynamic session management. Specific multimedia components include codecs, network access modules, and advanced media processing. On top of conventional transport protocols, advanced quality of service provisioning and realtime scheduling techniques are added; they guarantee a relatively constant quality of video and audio interactions even in the case of limited varyations of network-level quality. Additional components are file and compression servers, gateways and other administrative services accessed by conferencing clients.

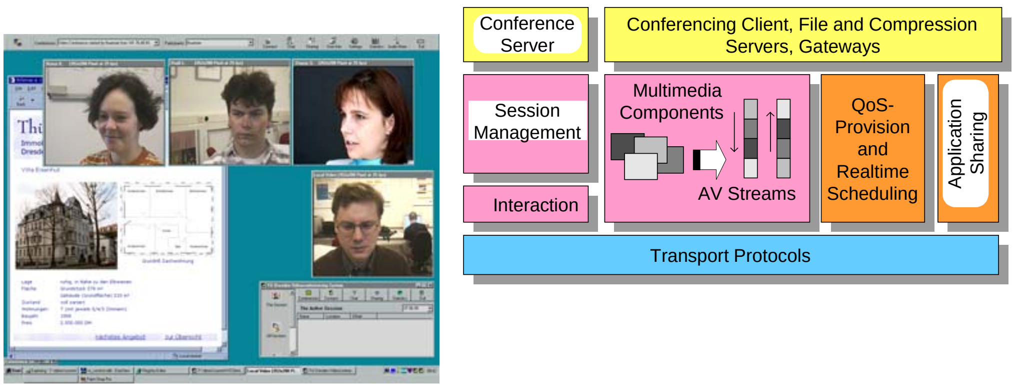

Figure 6: Distributed multimedia scenario

This scenario can also benefit from service-oriented architectures by mapping the different services onto web services technologies. However, major open research issues include the identification and analysis of quality of service behaviour of complex, composed multimedia services, the integration of quality of service requirements into the overall software lifecycle, also at the early specification and design level, and the seamless integration of synchronous media interactions with conventional asynchronous communication.

Mobile computing
Finally, a major trend in distributed systems is of course mobile computing, i.e. access to distributed services from mobile devices via various media, especially via wireless communication mechanisms such as GSM, UMTS, and wireless LANs. A related mobile application scenario employing service-oriented mechanisms is shown in figure 7: A service technician caches important product data within the central office of his company, the accesses upgrades and additional customer data dynamically on the road via mobile

communication, and finally attaches dynamically to a customer network where some repair job has to be done to customer machinery.

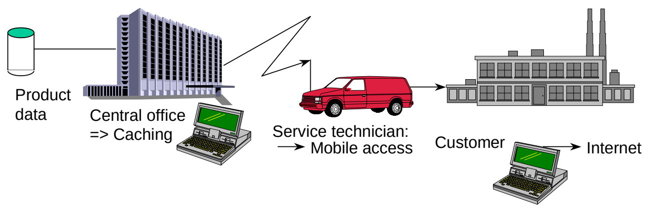

Figure 7: Mobile computing scenario

Within this scenario, existing service-oriented technology can be applied. However, additional open issues include the problem of dynamic coupling and decoupling of devices to/from the network. This requires efficient and intelligent prefetching and caching mechanisms to prepare a future disconnection phase, as well as consistent reintegration techniques to synchronize data modifications on the mobile device with backend data modifications by other client. In addition, an adaptation to various network quality levels of mobile communication carriers such as GSM or UMTS might be required; for example, more efficient compression of media data might be recommended. Moreover, adaptation of user interfaces to device characteristics of different mobile devices such as PDAs, mobile phones or notebooks are also important. Finally, context management of user and environment context, personalization of applications towards user requirements, and multimodal application support with speech integration are additional interesting research issues.

Conclusions

The paper has illustrated the potential and some typical usage scenarios of service-oriented architectures. Based on the discussion of basic technologies, it has been outlined that this approach is very flexible, is able to cope with dynamic infrastructures and environments, and is also able to coexist with existing backend services and applications. Various research issues and trends have also been outlined, such as the integration with grid computing, dedicated support for multimedia, and system support for mobile computing. In the future, it will be challenging to pursue these directions in a technical yet application-driven way and to come up with even more advanced notions of service-orientation.

References

/1/ www.service-architecture.com
/2/ SOAP specification; www.w3.org
/3/ WSDL specification; www.w3.org
/4/ Global Grid forum; www.gridforum.org
/5/ VidConference architecture; www.vidsoft.de