A SERVICE-ORIENTED ARCHITECTURE FOR TELEWORKING APPLICATIONS

Iris Braun, Alexander Schill
Chair for Computer Networks
Department of Computer Science
Dresden University of Technology
D-01062 Dresden (Germany)
Email: {braun,schill}@rn.inf.tu-dresden.de
Tel: +49 351 463 38261
Fax: +49 351 463 38251

ABSTRACT

The working life is changing rapidly since the technical conditions to work anywhere and anytime were developed. New employment models like telework emerged. Telework is the key to a more flexible design of working time and places of work.

But telework is not yet as widely spread as it seems to be. What are the barriers of an all-round implementation? In ongoing research projects we find out, that the different working environments in the office and at home are the main problems in the daily practice of teleworkers. That's why our intention is to develop an easy-to-use environment for teleworkers bundling several applications and services for communication, collaboration and cooperation.

KEY WORDS
Telework, service-oriented architecture, web services, portal, semantics

1. Introduction

In the last years there were many changes in our working life which were driven by the rapid development of electronic networking systems like the Internet and the penetration of web-based applications and cooperative software. This development offers the opportunity for workers to have more choice in terms of where and when they work. [1] As the teleworking trend emerges and evolves, it will ultimately change the current definition of "the workplace". The flexibility of telework can offer wide-ranging, powerful business solutions.

In section 2 of this paper a short overview of the application field telework will be given and usage scenarios for teleworking applications will be presented. Derived from this we have defined requirements for an effective and flexible collaborative environment supporting teleworkers at their daily work.

The main part of this paper in sections 3 and 4 describes the development of a service-oriented architecture for teleworking applications and the technologies to implement such applications. Finally the realization of a workflow for the flexible integration of various services in the teleworking environment will be presented.

2. Application field: Telework

First of all Jack Nilles introduced the concept of telework and initiated the first documented pilot telecommuting project in Southern California in the early 1970s. [2] Since Alvin Toffler popularized his idea 1980 within the book „The Third Wave" [3] all over the world, telework developed rapidly and in various directions.

Telework is an umbrella term for a wide range of alternative office arrangements with variable working time and places of work. The employment models diversify from permanent workplaces at home in the homeworking scenario to all-over mobile workplaces in the mobile teleworking scenario. The common element across all aspects of telework is the use of computers and telecommunications to change the accepted geography of work. [4] Fields of using teleworking solutions can be all businesses using information and communication technologies for performing working tasks such as writing services, data recording, software design and programming, financial transactions, computer aided design etc.

As with most innovative solutions telework represents both - opportunities and risks. A very frequently cited disadvantage of homeworking, as one form of teleworking, is that it is reducing or even eliminating the daily contact of the worker to colleagues and managers [5]. The homeworkers may be barred from internal communication and may find it difficult to reach the manager or chief, when they have problems. That's why an important feature of an environment for teleworkers

should be the support of synchronous collaboration with audio/videoconferencing and data and application sharing between distributed teleworkers.

Teleworking applications are mainly characterized by distribution of their components: one workplace at home or anywhere else and one in the office, connected via telecommunication services.[6] There are many different software solutions on the market to support distributed work using telecommunication and information technologies. But the problem is to bundle all these inhomogeneous applications to one uniform environment. To reach the flexibility for working anywhere and anytime a virtual working environment is needed which can be accessed from any place having a browser and internet access, at home, on the road or in the office. [7] The goal of our activities is to develop a set of technologies fitting in the service-oriented architecture in order to bring teleworking applications to their full potential.

3. Service-oriented architecture for teleworking applications

3.1 What's a SOA?

„SOA is an architectural style whose goal is to achieve loose coupling among interacting software agents. A service is a unit of work done by a service provider to achieve desired end results for a service consumer." [8] Service-oriented architectures represent a framework in which self-contained, modular applications can be described, published, located and dynamically invoked in a programming language and system independent way. Service-oriented architectures (SOA) enable dynamic, flexible applications, which always change rapidly. A SOA provides the ability to more easily integrate new services and old legacy systems and to automate business processes. [9]

In contrast, earlier approaches to building IT systems tended to directly use specific implementation environments such as object orientation, procedure orientation and message orientation. So applications were created which were often tied to the features and functions of a particular execution environment such as CORBA, J2EE and DCOM. The traditional implementations might have better performance, but in many cases the performance issue is less important than the ability to more easily achieve interoperability and flexibility.

3.2 Realizing a SOA with web services

The web service platform contains the basic and extended features necessary to support a SOA such as the described one. Web service technology makes services available independent of proper implementation of the particular application behind. The programming language and operating system used to implement the application behind the web service is completely irrelevant for the service requestor. A service consumer has no need to know if an object oriented or procedural programming model is chosen to realize the web service. It doesn't matter whether it is based on a monolithic application or whether it is realized as a distributed application based on many different components.

The basic web services architecture consists of 3 mechanisms that support the interaction between a service requester and a service provider by sending SOAP messages, the description of the interfaces of the services with WSDL and the potential discovery of the web services with UDDI. [10]

SOAP provides a standardized way to encode different protocols and interaction mechanisms into XML documents that can be easily exchanged across the internet. A big advantage of web services is that SOAP can use the HTTP protocol to pass the mostly used firewalls. WSDL acts as an advanced form of IDL for describing the interfaces of a web service. The UDDI registry is used by providers to publish and advertise available services and by the clients to query and search for concrete services. [11] Because of using standardized XML based protocols, web services can be used in an interoperable way in heterogeneous environments.

3.3 Creating a SOA for teleworking applications

For realizing flexible, interoperable teleworking applications we choose a four-tier architecture, shown in fig. 1.

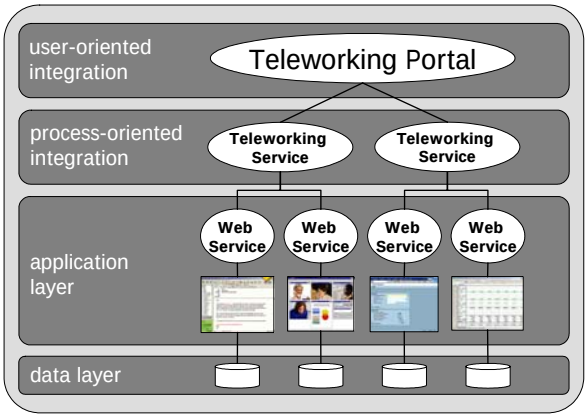

Figure 1: Four-tier architecture for teleworking applications

Unlike common three-tier-architectures of web applications we add a fourth layer for the process-oriented integration between the application layer and the presentation layer. So it is possible to realize complex business processes by combining several web services from different service providers. The main components of our solution are the data and application layer at the

service provider's site, a central teleworking service integrator for the process-oriented integration and a teleworking portal for the user-oriented integration. The specific tasks and requirements of the different layers are described below:

Data and application layer - Service provider:

- Map real processes and workflows from daily office work to generic services realized by a service provider;
- Dissipate complex processes into small basic services and implement as web services;
- Create web services as interfaces to old legacy applications;
- Sample services: calendar service, email service, crypto service.

Process-oriented integration - Teleworking service integrator:

- Make the teleworking services available for access through the portal;
- Orchestrate basic web services to complex teleworking services and business processes;
- Search for suitable basic services over the Intranet or Internet;
- Provide the teleworking services as building blocks of a universal working environment.

User-oriented integration - Teleworking portal:

- Provide a user friendly interface for all working tasks;
- Allow access to all needed applications and data – realize a single-sign-on for all services;
- Provide a framework for describing demanded services and for flexible integration into the portal;
- Personalization according to the individual teleworkers:
  - Adapt the interface to the special requirements of each teleworkers,
  - Individual choice of integrated teleworking services.

The teleworking portal provides an integrated access to all applications and data the teleworker needs for his work. Portals are frameworks that render and aggregate information from different sources and provide it in a compact and easily consumable form to an end-user. In addition to pure information, many portals also include applications like email, calendar, organizers etc. [12]

With the personalization functions of the portal the teleworker can build his own working environment dependent on his preferences and capabilities. Especially for people with handicaps special services for speech input and output can be integrated. So telework could be a new chance for them to take part in the common working life.

Another scenario for useful application of our solution is mobile telework, where the worker changes his working place dynamically and uses different devices. That's why an additional scope of our work is the question how can the portal be adapted dynamically to different devices and how the context and the profile of the worker can be distributed among the different working places. We want to point out the special requirements to the portal solution to support this dynamic switch of working places and devices.

We can distinguish 3 different roles in our scenario:

- The *service provider* makes a web service available by publishing its WSDL to a UDDI-Registry.
- The *portal admin* arranges a basic set of needed services in the portal. Therefore he creates UDDI queries and uses the portal engine to integrate the service invocation. He is familiar with WSDL, UDDI and the portal engine.
- The *teleworkers* mostly don't know enough about web services integration and portal implementation. But they want to integrate new services in their individual working environment. That's why a framework for describing search criteria for new services is needed.

4. Flexible service integration

4.1 Automated discovery of services

On-site or mobile teleworkers need more than a static set of provided services. They need to adapt their working environment to the current working situation and partners. For instance they want to use a service of the customer they are currently working for. To reach the required flexibility a central teleworking service integrator searches and combines the different web services dynamically.

Thereby it should integrate not only well-known web services from the intranet of the enterprise but also services from external providers. That's why problems like QoS and security of web services as well as automatic search and composition of web services are addressed in our current work.

In the portal we want to provide a framework for an easy-to-use description of web services the teleworker wants to integrate in his working environment. The goal of this framework is to enable a teleworker to pick from a rich choice of compliant remote services, and integrate them with just a few mouse clicks and no programming effort.

For the flexible service integration we developed a workflow shown in figure 2. First the teleworker describes the search criteria for a needed service. The teleworking service integrator receives the request and peals off the declarative description of the service required and sends a search query to a UDDI registry.

After that a list of available services is shown to the user. The teleworker can select a service from the list. If he didn't find one that matches all his requirements he can change the search criteria. Finally the teleworking service integrator will bind the service selected to integrate it into the portal.

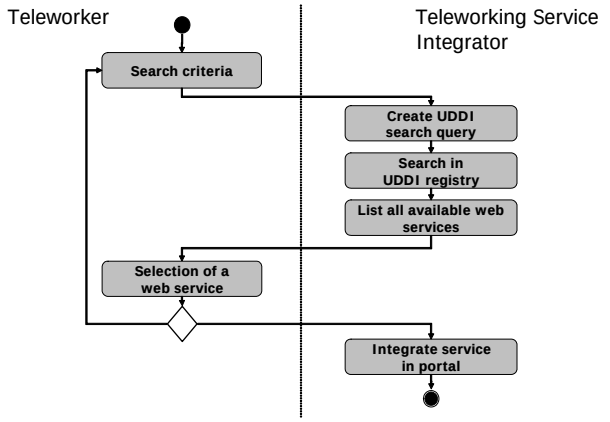

Figure 2: workflow of service integration

If all qualified services are equivalent in their functionality they offer the selection can be automated. The workflow can be changed in this manner that the teleworking service integrator decides itself which web service will be chosen. This decision can be based on overall environmental properties like actual workload at the service provider side, average response time or other Quality of Service parameters. In order to get a complete automation of such interactions, standardized user interfaces of web services are necessary.

4.2 Semantic description of search criteria

To enable the dynamic discovery of services a mechanism is required to describe behavioural aspects of the searched services, e.g. the provided functionalities of a service. A semantic description of a service should include the capabilities it can provide, under what circumstances these capabilities can be provided, what input the service requires to work and what results can be delivered.

With semantic markup languages, the information necessary for web service discovery could be specified as computer-interpretable semantic markup at the service web sites, and a service registry or ontology-enhanced search engine could be used to locate the services automatically. There exist many description languages and ontology methods to add semantic information to web services, e.g.

- SWSL (Semantic Web Service Language)
- WSMF (Web Service Modeling Framework)
- RDF-S (Resource Description Framework - Schema)
- OWL (Web Ontology Language)
- Topic Maps

We developed a domain specific ontology based on RDF to describe specific teleworking service characteristics to improve the search process. Part of this ontology is a teleworking service taxonomy based on a detailed analysis of suitable Web Services in the teleworking field. The teleworking service integrator matches the domain ontology to OWL-S or a UDDI search query.

4.3 Integration of the teleworking services in the portal

Traditional data-oriented or RPC-oriented web services require aggregating applications to provide specific presentation logic for each web service. Furthermore, each aggregating application communicates with each web service via its unique interface. Common portals require different rendering and selection mechanisms for different kinds of applications, but all of them rely on the portal's infrastructure and operate data or resources owned by the portal, like user profile information, persistent storage or access to content. Consequently, most of today's portal implementations provide a component model that allows plugging components referred to as portlets into the portal infrastructure. Portlets are user-facing, interactive web application components rendering markup fragments to be aggregated and displayed by the portal. [13]

Typically, portal users have to write special adapters to enable communication with applications and content providers using a variety of different interfaces and protocols. The process of making new portlets available is tedious and expensive. This approach is not well suited to dynamic integration of business applications and services as a plug-and-play solution.

That's why we developed a portlet container for integrating various web services in the portal. For describing a user interface for the web services we use the WSGUI-Standard which was developed as part of the FX-Agents research project at Stanford University [14]. The WSGUI standard allows a GUI Deployment Descriptor (GUIDD) to be placed side by side with a WSDL file to make the internal operations visually accessible to the user, independent of the environment like desktop toolkit or web browser. We improved this WSGUI protocol within our project work. For testing and evaluating the presented architecture a prototype using the Apache Jetspeed portal engine is implemented.

WSGUI provides additional information for building up the input and result output dialogs. Both web forms and rich-client GUIs allow for many standard and custom widgets and form elements to be used in dialogs. For WSGUI however, it suffices to concentrate on a selection of them, as long as they support all required properties. These elements are borrowed from the XForms standard to allow reusable GUI definitions.

In all the GUI elements which present particular strings to the user (as do *label* and *help*), multiple tags can occur as long as they are distinguishable by means of their *lang* attribute, which specifies texts for different languages. To provide a fallback solution for missing translation, a variant without any *lang* tag should always be present in such a case. Each GUI element is contained in a *formComponent* tag, which provides an XPath locator (xpath) to find out which elements to apply to. In figure 3 a sample GUIDD for an email web service is given.

```
<?xml version="1.0" encoding="UTF-8" ?>
<wsgui:deployment
 xmlns:wsgui="http://fxagents.stanford.edu/
 2002/10/wsgui"
 xmlns:targetns="urn:email"
 xmlns:xsl="http://www.w3.org/1999/XSL/
 Transform">
  <wsgui:wsdl href="email.wsdl" />
  <wsgui:formComponents>
    <wsgui:formComponent
     xpath="//xsd:element[@name='from']">
      <wsgui:input>
        <wsgui:label>Von</wsgui:label>
        <wsgui:help>Ihre E-Mail Adresse.
        </wsgui:help>
      </wsgui:input>
    </wsgui:formComponent>
    <wsgui:formComponent
     xpath="//xsd:element[@name='to']">
      <wsgui:input>
        <wsgui:label>An</wsgui:label>
        <wsgui:help>Die E-Mail-Adresse, an die
         gesendet werden soll.</wsgui:help>
      </wsgui:input>
    </wsgui:formComponent>
    <wsgui:formComponent
     xpath="//xsd:element[@name='subject']">
      <wsgui:input>
        <wsgui:label>Betreff</wsgui:label>
        <wsgui:help>Betreffzeile der E-Mail.
        </wsgui:help>
        </wsgui:input>
    </wsgui:formComponent>
    <!--  basetype: GUIDD extension -->
    <wsgui:formComponent
     xpath="//xsd:element[@name='message']">
      <wsgui:input basetype="multiline">
        <wsgui:label>Nachricht</wsgui:label>
        <wsgui:help>Der Text der Nachricht.
        </wsgui:help>
      </wsgui:input>
    </wsgui:formComponent>
  </wsgui:formComponents>
</wsgui:deployment>
```

Figure 3: GUIDD-description of an Email Web Service

Another specification which can solve the problem of the integration of external services in a portal is the WSRP (*Web Services for Remote Portlets*) specification [15]. It solves this problem by introducing a presentation-oriented web service interface that allows the inclusion of and interaction with content from a web service. Such a presentation-oriented web service provides both application logic and presentation logic. WSRP defines how to plug remote web services into the pages of online portals and other user-facing applications. This allows portal or application owners to easily embed a web service from a third party into a section of a portal page.

How can a reasonable application scenario of the flexible service integration look like? An on-site teleworker works for a software project in the office of the customer enterprise for 6 month. For a better coordination with the other employees involved in the project he wants to use the group calendar of the project team. Figure 4 illustrates the integration of such a calendar service as WSRP-Service in the teleworking portal.

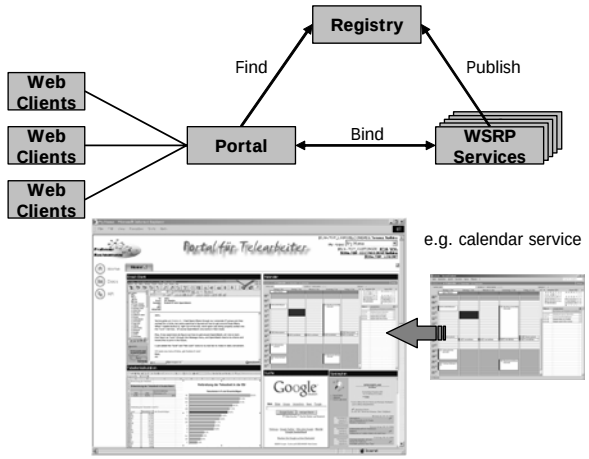

Figure 4: Integration of a WSRP-Service in a portal

The interfaces of a WSRP Service are described in WSDL. Such a WSDL document includes common technical information about the bindings between the service requestor and the service provider and the used infrastructure. The WSRP services are integrated as remote portlets which work like local JSR-168 portlets independently of the implementation at the provider's side. The following XML document (Figure 5) is used for providing a JSR-168 standardized portlet as WSRP service.

```
<?xml version="1.0" encoding="UTF-8" ?>
<request type="update"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-
 instance"
 xsi:noNamespaceSchemaLocation="PortalConfig
 _1.2.1.xsd">
 <portal action="locate">
   <web-app action="locate" active="true"
    uid="stdTestsuite.war.webmod">
     <portlet-app action="update"
      uid="stdTestsuite.war">
       <portlet action="update"
        name="TestPortlet1" provided="true"/>
     </portlet-app>
   </web-app>
 </portal>
</request>
```

Figure 5: providing a JSR 168 portlet as WSRP service

In order to get the WSRP service description, the requestor must first send a *getServiceDescription* request

to the provider. To this request, the provider responds with a *getServiceDescriptionResponse* document. The requestor can now use this response to aggregate this portlet, and offer that service to its users.

4. Conclusion

The presented teleworking environment offers the teleworkers an integrated individual access to all needed applications. The teleworking portal can be accessed from clients only having a web browser installed. This principle of thin clients minimizes the effort for maintenance and support. Using the same environment at the office and at home for accessing various applications also decreases the need for training the employees.

The developed solution facilitates a flexible integration of different services of various service providers. Two different approaches for the dynamic integration of web services into portals were described. Thereby the users will be supported by an easy-to-use framework for searching and combining several web services. The search will base on semantic descriptions using a domain specific ontology for teleworking services we developed. The task of service composers and portal administrators will be much easier because services can be added dynamically to the environment, and users benefit by having more services made available to them in a timely manner.

The working environment can also be used within ordinary employment models. The implementation of the presented solution in the whole enterprise gives the chance to reorganize the workflows and structures fundamentally. The developed service-oriented architecture can also be implemented in B2B and B2C scenarios.

Using a SOA increases reuse, lowers overall costs, and improves the ability to rapidly change and evolve distributed applications, whether old or new. A SOA realized with web services is the ideal combination of architecture and technology for consistently delivering robust, reusable services that support today's business needs und that can be easily adapted to changing user requirements.

Web services combine the ease of use of a document markup language with distributed computing concepts and apply the result to solve IT integration problems easily and cheaply. The major advantages of implementing the presented architecture with web services are that they are pervasive, simple and platform-neutral. The broad adoption of web services standards makes it easy to imagine that all applications will have service interfaces in the future and that they can be combined with each other to individual adapted working environments.

References:

[1] K. Gareis, W.B. Korte, Telework in Europe: Status Quo and Potential, Good Practices and Bad Practices. In: BAuA (Eds.): *Telearbeit: Arbeits- und Gesundheitsschutz aus internationaler Sicht*, Dortmund/ Berlin 2002, 43-74.

[2] J. M. Nilles, *The Telecommunications-transportation tradeoff: options for tomorrow* (New York, Wiley, 1976).

[3] A. Toffler, *The third wave* (New York, Morrow, 1980).

[4] European Telework Online: *Telework and Telecommuting: Common Terms and Definitions.* http://www.eto.org.uk/faq/faq02.htm, 2004.

[5] I. Braun, U. Zschuckelt, Designing a collaboration environment for teleworkers; *Proc. of the WebNet2001*, Orlando, 2001.

[6] I. Braun, K. Borcea, A. Schill, Working and learning at home - Designing a virtual working and learning environment for teleworkers ; *Proc. of the 16th IFIP WCC2000*, Beijing, China, 2000.

[7] I. Braun, A. Schill, Building a universal Teleworking Environment using Web Services; *Proc. of the IASTED IKS 2002*, St. Thomas / USA, 2002.

[8] E. Cerami, *Web services* (Tokyo: O'Reilly, 2002).

[9] E. Newcomer, G. Lomow, *Understanding SOA with Web Services* (Boston [u.a.]: Addison-Wesley, 2002).

[10] W3C Technical report: *Web Services Architecture Requirements - Working Draft 11 October 2002*, http://www.w3.org/TR/2002/WD-wsa-reqs-20021011 (W3C, 2002).

[11] G. Alonso, *Web Services: Concepts, Architectures and Applications* (Berlin: Springer, 2004).

[12] C. Eberhardt, T. Gurzki, and H. Hinderer, *Marktübersicht Portal-Software für Business-, Enterprise-Portale und E-Collaboration* (Stuttgart: Fraunhofer-IRB-Verlag, 2002).

[13] T. Schaeck , R. Thompson, *Web Services for Remote Portlets (WSRP).* Whitepaper, OASIS WSRP TC, http://www.oasis-open.org/ (OASIS, 2003).

[14] M. Kassoff, D. Kato, W. Mohsin, Creating GUIs for Web Services. *Proc. of IEEE Internet Computing 2003*, Vol. 7, No. 4, 66-73.

[15] OASIS Standard: *Web Services for Remote Portlets*, Specification Version 1.0, http://www.oasis-open.org/ committees/wsrp (OASIS, 2003).