

Creating Dependable Web Services Using User-transparent Replica

Markus Hillenbrand, Joachim Götze, Paul Müller
University of Kaiserslautern
Department of Computer Science
67663 Kaiserslautern, Germany
<http://www.icsy.de>
{hillenbr,j_goetze,pmueller}@informatik.uni-kl.de

ABSTRACT: *Dependability is a major concern in software development, deployment, and operation. A commonly accepted solution for providing fault tolerant services on the Internet is to create replica of the services and to deploy them to several hosts. Whenever the service or the underlying node or network fails, another service is ready to take over. In the Venice¹ project, several techniques are combined to create a dependable framework for deploying and managing distributed services using replica on several distinct network nodes.*

Categories and Subject Descriptors

C.2.1 [Network Architecture and Design]; C.2.5 [Local and Wide-Area Networks]; Internet; C.2.6 [Internetworking] B.4.1 [Data Communication Devices] H.3.5 [Online Information Services] : Web-based Services

General Terms

Internet Protocol, Voice over IP (VoIP), Web Services, Internet Technology

Keywords: Software development, Venice framework, XML, Dependable web services, Web service technology

Received 28 Oct. 2005; Revised 15 Dec. 2005; Accepted 16 Jan. 2006

1. Introduction

The Internet has become one of the most important media for communication. Often the Internet is the basis for a convergence of different media. Today it is expected to use the Internet protocol (IP) not only for the classical Internet services, but also for all types of communication from television to radio and telephone. Voice over IP (VoIP), i.e. telephony on the basis of Internet technology, gains more and more importance. Compared with the classical ISDN or PSTN telephony system, VoIP does not only allow cost saving because of a more efficient use of physical lines in the scope of long-distance calls, but also because of a reduced effort in in-house cabling. Additionally VoIP has potential for providing a new manifold of features beyond telephony. Current products and solutions in the area of Internet-based telephony are highly developed and allow a comparable comfort like their predecessors in the classical telephony, but it is possible to identify several important facts that can be the reason why VoIP still has not yet spread as much as desired by the driving forces. One of these reasons is the lack of overall dependability.

VoIP enables users to utilize standard Internet technology for making a phone call to other participants – which themselves either also use VoIP or a standard telephone. VoIP has been developed over the past decade and is currently on its way to replacing the plain old telephony system. But current VoIP solutions lack several features standard telephone systems have offered a long time. These features are called supplementary services and provide some kind of end-user comfort before or during a phone call. Among these supplementary services are: call forwarding, voice mail, call center functionality, roaming, etc. The modern ISDN system has introduced several dozen supplementary services currently available to end-users. But current VoIP systems do not integrate and offer them in a *standardized* way – if they support supplementary services at all.

The Venice project aims at creating an open and distributed framework for VoIP that allows the easy creation, integration, and usage of supplementary services. The means to achieve this are virtualization of the resources used and abstraction from the underlying technology. It uses Web services technology to implement supplementary services and defines XML data types which are used for exchanging messages between the services and the

1 The project is funded by Siemens AG, Munich, Germany

Uncorrected Proof

clients. The usage of Web services assures standardization, flexibility, and interoperability as well as a platform, language and vendor neutral implementations.

A complete overview of the Venice project will be given in section 2. Dependability is a very important aspect for both a VoIP provider and the customers. Only an open and dependable system has the potential to broaden the acceptance of VoIP. Therefore in section 3 the term dependability will be explained thoroughly. Then it will be transferred to the service-oriented VoIP domain with a special focus on availability in section 4. The paper concludes with a short summary in section 5.

2. The Venice Framework

In order to circumstantiate the architecture of the Venice framework, it is necessary to explain some of the major preconditions immanent to it. There are several hypotheses on which the architecture is built upon. These hypotheses directly lead to some technical and non-technical considerations that have to be dealt with when designing a next generation VoIP infrastructure:

1. *All VoIP providers act mainly autonomously.* In the context of a world-wide operable VoIP market, this means that every provider is primarily responsible for his own customers. But there also has to be a secure and standardized passage to other providers' systems. This can also be regarded as a shade of interoperability between all VoIP providers.
2. *A provider wants to re-use existing infrastructure, i.e. hardware and software.* Phone companies have invested a lot into their existing hardware and software. Although currently changing backbone infrastructures towards IP technology, they still rely on legacy systems that will prevail for some time. It is thus significant to re-use these components without having to replace them.
3. *Supplementary services and their interoperability are the assets of a VoIP provider.* Every VoIP provider uses supplementary services to attract customers. This effect has started with the ISDN system and has dramatically increased in mobile communication. Customers are willing to pay for services they can swank with or separate themselves from others. Hence individuality is one key to satisfied customers. Different services at different price levels make sure that every customer can find the combination of services she or he searches for. And the service capabilities offered by one service provider distinguish him from others. Thus, it is a very important issue to be able to integrate and change supplementary services whenever necessary and without great effort. It is imperative that a framework for VoIP is designed in that way.
4. *Customers are no computer specialists and urge for easy-to-install, easy-to-update, and easy-to-use software.* Current VoIP solutions are mostly realized as flexible software products for personal computers. Installation and maintenance is incumbent on the PC's administrator. But making VoIP accepted on a large scale means to diminish or eliminate administration tasks. Using a VoIP client has to become as easy as using a standard telephone. This can be achieved by creating simple (i.e. less complex) client software and sourcing out to the VoIP provider as much code as possible. The knowledge how to maintain and manage the code is inside the VoIP provider's domain, and the code should also be located there whenever possible. The clients can then access the code by calling a standardized interface to its functionality.

There are also some constraints that have been taken into consideration:

1. *The failure of one component must not result in a complete system failure.* Dependability is one key issue of current software systems and as such it is imperative for an open distributed system to be fault-tolerant. As each component runs on its own server, a failure of such a component (or its communication infrastructure) must not affect the system as a whole. In case of a distributed telephone system, this means that a failing supplementary service should be compensated for so that a phone call is still possible. And the basic service responsible for managing a call must always be available. If it fails, a service replica must stand by to take over the old one's duties.
2. *It must be possible to support every established telephony technology without any special treatment by the users.* In the classic telephony system it is possible to make a call to other countries where other telephone standards are in use (e.g. ISDN in Europe and the US). This is transparent to the customer. Telephone providers are responsible for technically accessing other systems. This should also be the case for an Internet based telephony system. Different VoIP providers use different technology (like H.323, SIP, or proprietary products) and should be responsible for translating between these different protocols. Only if that happens transparent to the user, VoIP will prevail.
3. *Using existing infrastructure should be easily possible.* Virtualization is the key to leveraging existing technologies or solutions. Building a new system on top of existing systems is the best way to support both older and newer technology at the same time. Telephone providers therefore should have the possibility to extend their existing hardware and software to support a new VoIP infrastructure.

In order to encompass all these considerations, Web services technology has been chosen to become the realization platform for the project's prototypes. Web services technology can be used to leverage legacy systems and offer their functionality in a service-oriented environment. It has proven to be an acceptable means for making these legacy systems accessible by current (i.e. newer) technology while allowing for innovative solutions by abstracting from an underlying concrete implementation or technology.

The services offered by a VoIP provider can be classified into three distinct categories: *Management Services* are services necessary for the common usage and maintenance of services. They are not bound to VoIP services and can be used in any service-oriented application scenario. The *Basic VoIP Services* contain all elementary functions that are responsible for making a phone call with other participants. The category of *Supplementary Services* finally contains all additional services that complement the Basic VoIP Services and contribute comfort before, during or (shortly) after making a phone call. An overview of this architecture can be seen in figure 1.

The *Management Services* help VoIP providers to actually create and maintain a distributed and open VoIP infrastructure. Currently, the following services can be identified:

- *Role-based Single Sign-on (SSO).* In order to access all services of a VoIP provider, a token-based single sign-on strategy is the most promising approach. A user has to authenticate once and will then receive a token allowing him to proof his identity in any further communication by providing this token. A more detailed description of this token-based single sign-on solution developed within the project can be found in [6, 7].
- *Metering, Accounting, and Billing (MAB).* Because of the service-oriented approach used in this architecture, it is possible to use services from different providers that implement the same service with different characteristics, e.g. quality of service. In order to have a reasonable accounting for services used, the VoIP providers need to have a metering system that keeps track of all requested and utilized services in the VoIP system.
- *Software Deployment (SDS).* In order to provide an easy-to-use VoIP application to the end-user, it is important to

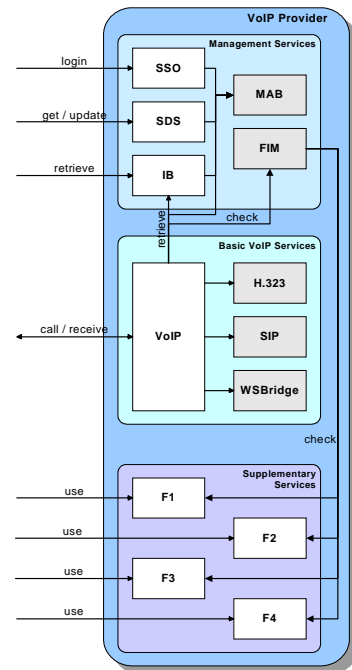


Figure 1. The Venice Framework (Overview)

dispense the user from tasks like installing and/or updating software. Additionally this service enables a VoIP provider to automatically and dynamically replicate certain services in order to compensate high load situations and establish a replica management.

- *Information Brokering (IB).* Due to the potentially huge amount of diverse information, it is important to have a service that is responsible for brokering, searching, and obtaining desired information, while also allowing to scale up to a growing quantity if for example new services are introduced by service providers. Its flexible design allows for brokering services according to pre-defined metadata.
- *Feature Interaction Manager (FIM).* This manager component is commissioned to check any request for a supplementary service of the caller or the callee with respect to their usability and applicability. The main focus is to ensure that there is no undesired side-effect because of the consecutive use of multiple supplementary services.

The actual *VoIP Service* is the core component of the architecture. This component provides the basic functionality for Call Control, i.e. allows a client to initiate, perform and terminate a phone call. This implies that this component is directly connected to the client application and provides abstraction from the underlying telephony technology and the communication with other VoIP providers. As a result it is possible to mediate a communication that has been initiated by a SIP/H.323 client or that is targeted on a SIP/H.323 client without the need of interaction with the user or that this mediation becomes recognizable by the user.

Supplementary services are placed around the core VoIP Service and make it worthwhile for customers to use a specific VoIP provider. They offer functionality that can be combined until a customer's needs are satisfied. Typical supplementary services for private and business customers are Call Waiting, Call Forwarding, Call Hold, Three-Party, etc. And there are extensive supplementary services like a call center that can be found in larger companies. Currently, there are several hundred supplementary services available in ISDN and GSM networks – some more, some less known.

As the integration and usage of those basic and supplementary services has to be dependable, the following two sections will introduce the term dependability itself in more detail and apply it to the VoIP architecture introduced in this section.

3. Dependability

In general, the term *dependability of a system* is understood as the reliable ability of a system to supply a user with the provided services

[2]. Therefore dependability has to be ensured by suitable mechanisms. The user of a system can either be a human user or an attached technical system. Typically today's systems achieve a high dependability by having multiple dedicated resources ready to take over the tasks of systems having a high load. There will soon be a trend towards more flexibility and dynamism regarding the use of free resources [5]. In this context it is necessary to give an accurate definition of dependability. Only with a better understanding of the term and its implications it is possible to extend the dependability of a system using technical and non-technical means.

The working group 10.4 of the International Federation for Information Processing (IFIP, [4]) identified and integrated approaches, methods, and techniques for the specification, design, development, access, validation, and operation of systems that support attributes like *reliability*, *availability*, *safety*, and *security* – and therefore lead to dependable systems. The result of two decades of research and development is an ontology of the concept describing the term dependability from different point of views and making it tangible. According to these results dependability is characterized by three branches. *Threats* endanger a system and can have a serious impact on the correct functioning of a system. Several *attributes* of a system are reflecting different grades of dependability and diverse *means* can be used to attain dependability.

3.1 Threats to Dependability

As long as a service is providing the functionality defined in its specification, the service is working correctly. If *faults* occur during the operation of the service, this can lead to an *error* within several components and finally cause a complete system *failure*. The result is a malfunctioning of the service which can itself be considered as a fault in a larger context (Figure 2). If errors are maskable by several subcomponents, the overall system can nevertheless function properly according its specification [15].

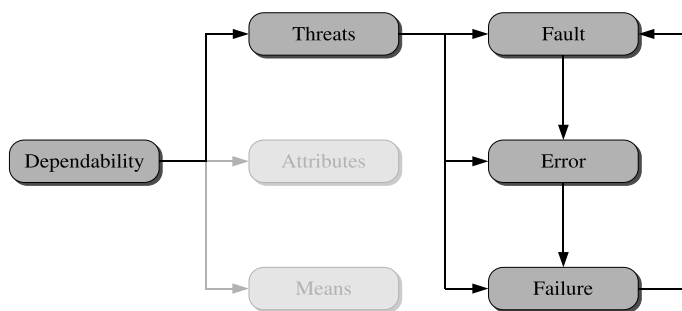


Figure 2. Threats to Dependability, cf [1, 5]

Additionally, the failure of a system and the associated incorrect service provided to the outside can be distinguished in greater detail: The *failure domain* where it appears, the *perception* of external users and the *consequences* are playing an important role in the assessment of the failure. While the first two distinctive features are related to the symptoms of the failure, the third is related to the severity of the faulty result – this severity can range from light to catastrophic. Particularly in distributed systems where dedicated components provide a conjoint – or orchestrated – service, a fault of a single component can result in an error in another component [5]. And this can furthermore result in a failure of the complete system and therefore have severe consequences.

3.2 Attributes of Dependability

Dependability has multiple attributes defining the characteristics of the grade of dependability. The readiness of a system to accept and process requests correctly is called the *availability* of a system. The *safety* of a system prevents severe consequences of a failure for the user and the environment in which the service is running in. The attribute *confidentiality* ensures that no data is revealed unauthorized. *Integrity* of a system means that no invalid system states or illegal state transitions can appear. Finally the attribute *maintainability* ensures the possibility to modify and patch a (running) system.

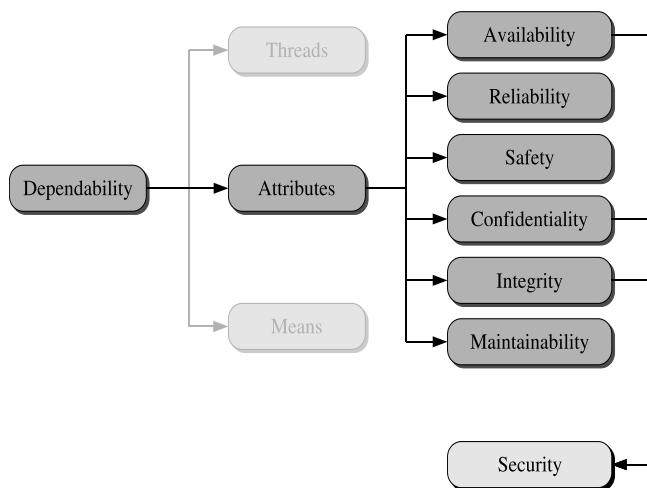


Figure 3. Attributes of Dependability

The term *security* is somewhat isolated in this context and not directly defined. While this property of a system is listed as an independent attribute in [4], it is stated in [2] and [12] as a consequence of availability, confidentiality, and integrity for authorized users. All these aspects of security can be attained on different levels. Infrastructure, transport, and application level security must be regarded to make a system completely secure.

In order to make a distributed system dependable for a corresponding task and environment, it is necessary to define the characteristics of the particular attributes and to assure that these definitions are attained. It is important to note that some of these attributes can mutually influence or stay in conflict with each other. For instance, attaining the absolute safety of a system by separating it from the external world automatically leads to zero availability for external users.

3.3 Means to Attain Dependability

The creation of a dependable system is determined by four significant techniques. These four techniques are illustrated in figure 3 and will be explained in the following. The number of faults occurring during the operation of a system can be reduced by *fault prevention*. In order to achieve fault prevention it is necessary to perform actions for quality assurance during design and development of hardware and software. While operating a system it is possible to increase its dependability by debarring sources of disturbance, e.g. blocking ports using firewalls.

Fault tolerance or *error tolerance* enables a system to provide a service correctly even when faults and errors appear. Fault or error detection techniques can be used to initiate mechanisms that allow a system to return to a valid state, i.e. a failure and error free state according to the service specification (*error recovery*). At first a fault diagnosis has to identify the source of the fault and in a subsequent step this fault has to be isolated logically or physically (*fault isolation*). Afterwards a *system reconfiguration* reconnects all unaffected components and the bad component is replaced by a replica if necessary. A *system re-initialization* finally completes the fault removal and the system can continue its regular operation again. Especially in the area of database management systems, the rollback technique is a typical action and can be used in other software systems, too, if the control flow is designed accordingly. Additional actions to reach a valid state after an error occurred are the return to determined checkpoints or the rollforward technique that transfers the system to a prospective valid state. A special form of fault tolerance is the masking of faults or errors, respectively. The usage of sufficient redundancy allows a switching to an alternative error-free component, while the error itself is not isolated. Furthermore, it must be noticed that also the applied mechanisms and especially the necessary software itself can be fault or error prone.

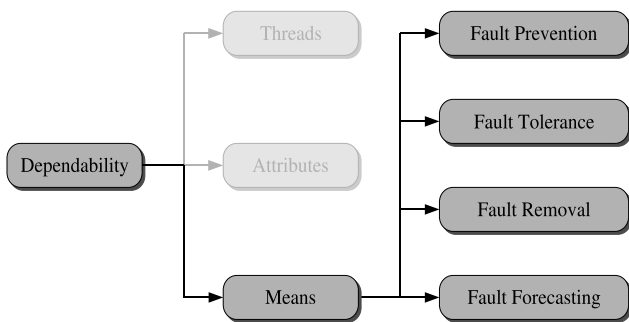


Figure 4. Means to Attain Dependability

Fault removal is used to eliminate faults during development and during ongoing operation and to ensure a correct service delivery. Actions for performing fault removal during the development of a system are the cyclic organized steps *verification*, *diagnosis*, and *correction*. During the verification phase a system is analyzed whether it is responding correctly to external and internal stimuli. In this context the meaning of correct is that the system is working within the guidelines of the system specification and therefore is valid. Fault injection can be used to insert false inputs into the system to test it. The results of verification have to be analyzed afterwards in the diagnosis phase in order to find sources of errors in the system that have to be eliminated. Fault removal during runtime has to identify the fault and its source in order to remedy the deficiencies. This can be done either during runtime or by stopping the operation of the system shortly for the fault removal. The process of fault removal during runtime is strongly connected with the maintainability the system provides.

Fault forecasting is used to predict potential faults and their consequences for a specific system. These predictions can be of qualitative or quantitative nature. A qualitative evaluation of a system is aiming at statements about failures and combination of failures and their consequences on the correct operation of the system as a matter of fact. A quantitative evaluation generally applies mathematical concepts of probability theory to potential fault occurrences and the attributes of dependability of the system in order to get a precise measure for dependability under certain circumstances.

4. Dependability in VoIP Systems based on Web Services

Dealing with dependability in a service-oriented VoIP system basically encompasses dependability aspects of distributed systems in general and long-term application availability in particular. In the following sections these aspects will be investigated in detail and transferred to the VoIP technology where necessary. The main focus here lies in service availability as this attribute of dependability is recognizable by a VoIP user.

In the area of classical telephony a major goal of the design for ESS 1 (ESS = Electronic Switching System) during the early 1960's was to achieve an accumulated failure time of less than two hours in 40 years of operation [3]. During the mid 1980's this aim had been reached. Since that time users are suffering very rarely from failures of today's telephone network and expect such high availability from VoIP, too. Among the attributes of dependability availability and reliability have as a result a high importance. The attributes of dependability introduced in chapter 3.2 will be analyzed in the following subsections with respect to the characteristics they have in a service-based VoIP system and which actions and means are needed to attain these characteristics. The fact that multiple VoIP providers are participating in the delivery of VoIP services is playing an important role in this analysis.

4.1 Availability

Availability is of great importance in distributed systems, because a non-available component may lead to huge restrictions or to the breakdown of applications depending on it. As a result the provisioning of replica of a component or resource is an essential

contribution to ensure their availability [15, 17, 11]. It is important to be attentive to the consistency of the data if state transitions within a component are possible during the utilization of the component.

The Web service technology provides helpful possibilities for the provisioning of service replica. At first it is possible to disassemble a service description on the basis of WSDL into fragments to separate data types from abstract and concrete service descriptions. The flexibility gained can be used for example to perform the service brokering on the level of abstract descriptions and move to the concrete level when actually using the service. Figure 5 illustrates the fragmentation of a service description in WSDL. The data types used can be defined as an XML schema in a separate file and therefore be used in separate service descriptions. On top of this the messages a service is going to exchange can be defined. The simultaneous definition of port types is building the abstract interface description of a service. Furthermore, if the binding of a service to the concrete transport protocols is done separately, this can be regarded as the concrete description of a service interface. Only the following definition of the service implementation is binding a service to its concrete endpoint. Therefore the predefinition of a WSDL service description can be done in four layers building one on top of the other. (It is very common to make this fragmentation with only three layers by including the binding into the lowest layer. However there are only a few bindings in use - mostly SOAP over HTTP - so that it proffers to regard this separately for reuse.) Only on the final layer the connection with the actual service provider is made - by specifying a concrete service endpoint.

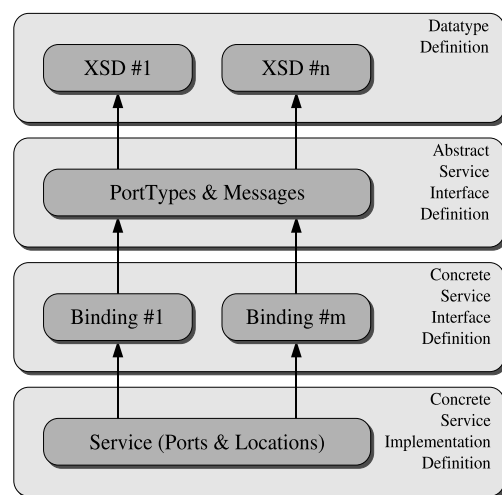


Figure 5. WSDL Fragmentation

The second important property of the Web service technology is the possibility to bind a service dynamically (dynamic invocation). Therefore a service requestor is not bound to a certain service, but the service actually used is specified and connected to at runtime. One way to implement dynamic invocation is to search at runtime for suitable service instances, e.g. via UDDI or another directory service. A key to search for can be the abstract service description (or a concrete binding with a four-layered fragmentation). All services found implement the interface and provide an endpoint for it. The application extracts the information about the concrete endpoint from one of the WSDL files found and uses it afterwards.

The dynamic binding of Web services identifies the endpoint to use from a WSDL file not during compile time of the client application but during runtime. This allows a service provider to provision multiple endpoints of a service by a single WSDL file for the service requestor. Figure 6 illustrates this procedure. In order to ensure the availability or to compensate load, a new endpoint can be inscribed in the WSDL file referring to an alternative implementation of the service. The huge advantage is the complete transparency for the service requestor. If the Web services are stateful, appropriate means for exchanging state information have to be applied.

The service provider is responsible for the modification of the WSDL file and therefore has to select a suitable mechanism implementing the strategy. The WSDL file can for example be generated dynamically or be replaced completely. For the selection of the used endpoint there has to be a suitable strategy. Defining a metric, e.g.

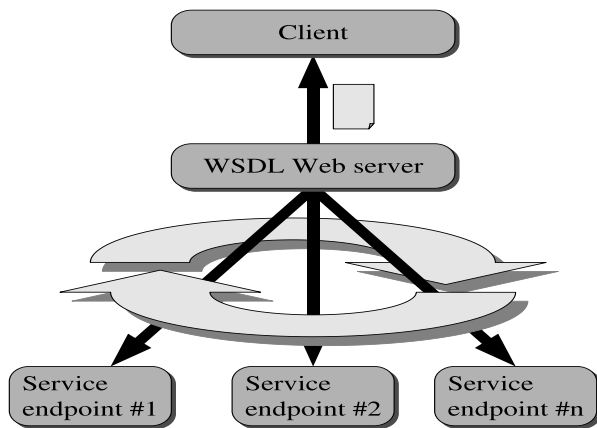


Figure 6. Round-robin created WSDL Files

comprising server load, network load, and the number of users of a service can help to select a suitable next endpoint. A much simpler strategy would be round-robin using all available replica one after another. Nevertheless, the whole approach is very flexible and can be easily adapted to the needs and possibilities of the service provider.

4.2 Reliability

Apart from the requirements of a VoIP solution concerning the underlying network infrastructure (with regard to the compliance with certain quality attributes like e.g. delay or failsafe performance through redundant connections), there are two metrics for reliability defined in [10] from the point of view of a potential VoIP provider. The *End-to-End VoIP Service Downtime* measures the time where two parties are not able to initiate, maintain or end a call regularly. This metric has to be kept small from the point of view of a service provider, because there is a direct impact on the customer. The second metric covers the whole system and is not focused on the customer only: *Defects per Million* numerically measures all service requests of the customer that have not been satisfied. A service request in this context is understood more precisely as the availability of a service, the continuous usability (especially important for long lasting telephone calls) and the fulfilling of all quality requirements of the customer. In addition to error tolerance of hardware and software (refer to chapter 3.3), redundancy of all VoIP sub-systems is favored to attain availability and reliability – the high availability of single components leads to high availability of the whole system by a redundant dimensioning of components.

As a consequence, this results in the strong connection between the attributes availability and reliability in the VoIP scenario, because performing a call starts with the availability of a service, but the reliability of a service has to be guaranteed until the call is completed. Here, a timespan of several seconds up to several hours has to be considered. A high reliability means that a telephone call can always be initiated, performed with all supplementary services and quality features and finally terminated regularly.

4.3 Safety

The safety of a system prevents fatal consequences of a failure for users and the environment the service is provided in. In the context of Web services, this implies that the failure or breakdown of a service does not influence the execution of other services essentially. Because of the typically loose coupling of Web services the breakdown of a service can be prevented by using replica (refer to chapter 4.1). The failure of a service can be compensated with replica, too. Prerequisite is the detection and propagation of such a failure in the system (refer to chapter 4.7).

4.4 Confidentiality

Confidentiality of data is playing an important role for communication via insecure ways of transportation. If Web services are addressed through the Internet, the use of mechanisms to exchange encrypted messages is essential to ensure confidentiality. In order to exchange messages in the XML format, e.g. via SOAP, it is possible

at this point to revert to XML-Encryption, a standard to encrypt selected branches of an XML document. Confidentiality of data on the infrastructure level protects data against unauthorized access. In order to achieve this, well-known mechanisms like firewalls and additional SOAP firewalls can be used. The first type protects the affected systems against unauthorized access, while the second type is specialized on the processing of SOAP messages and therefore protects the access to certain resources on the protocol level.

4.5 Integrity

Data integrity has to be assured when transporting data over an insecure network. For XML documents it is possible to apply XML-Signature to ensure data integrity. This standard allows signing parts of an XML file and therefore ensuring its integrity. In combination with XML-Encryption there is a technique at disposal to encrypt and sign parts of XML encoded messages.

4.6 Maintainability

If Web services are used across multiple provider domains, the provider of a service has to take into account that errors and failures influence remote service users and other providers. In addition to ensuring plain availability it is important that maintenance work at a service does not lead to discontinuity of the service provisioning for third-parties at all (depending on the requirements a small discontinuity may be acceptable). A part of maintenance work is the provisioning of new or updated software versions of a service. Especially in this occasion, it is important to ensure that the new service at least provides the same functionality as the old one and that this functionality provides correct results. Maintenance and updates have to be done as smoothly as possible during operation. There is no standardized procedure to signal a service user that a new version of a particular Web service is available – neither UDDI nor WSDL do provide this in their current specifications. In [11] a notification mechanism especially for compound Web services is proposed, notifying the service users about the existence of newer or alternative versions, while old versions of the service still continue their work. Further such a mechanism can use a special metric to measure the confidence in Web service correctness and with this facilitate heuristics for the service users to select and integrate a suitable service.

4.7 Means to Attain Dependability in Service-oriented VoIP-Systems

Within the scope of software engineering several approaches for attaining fault and error tolerance, fault prevention and prediction have been elaborated and can be used to build software components. Some of the more important strategies in the context of hardware and software are outlined in [5]. Especially in the context of Web services additional methods are subject of further research. For example in [16, 17] the concept of Web Services Composition Action (WSCA) is introduced, subsuming that transactions across compound services of different domains are not applicable and the locking of resources is not feasible. The guiding principle of this approach is the forward propagation of faults and errors in the processing chain to allow for error handling at the relevant places. Within the application there is more knowledge available to handle faults and errors in its current context. This secures the self-sufficiency of the services used; services do not have to react to external matters. However on the application level it is needed to ensure that state transitions already made can be compensated and revoked by special functions of the services involved. The waiver of a model of transaction is gained by more functionality on the application layer.

Also the topic of fault removal for applications based on Web services is taken into account by current research projects with techniques for fault injection. An interesting approach to that is WS-FIT (Web Services Fault Injection Technology) [14, 13]. It can be used to inject errors into SOAP messages in order to validate the services offered for the reaction upon incoming defective messages. This allows mainly the usage and testing of processing steps rarely used within an application.

5. Summary

As dependability is of major concern it is necessary to deal with all its aspects during specification, design, implementation, and operation of software components. In the context of a service-oriented VoIP framework and its architectural design a lack of dependability is directly recognizable by end-users. Therefore the threats to dependability and the (measurable) attributes of dependability itself have been introduced. Afterwards some means to attain dependability have been investigated. With regard to Web services an approach for gaining availability has then been presented. This approach basically comprises WSDL fragmentation as well as service replica management.

References

- [1] Avižienis, Laprie, and Randell (2000). Fundamental Concepts of Dependability. Technical report, UCLA, LAAS, Newcastle University.
- [2] Avižienis, Laprie, and Randell (2002). Fundamental Concepts of Dependability. Proceedings of 3rd Information Survivability Workshop ISW 2000 (Boston, USA), pages 7–12.
- [3] Downing, Novak, and Tuomenoksa (1964). No. 1 ESS maintenance plan. The Bell System Technical Journal, 5:1961–2019.
- [4] International Federation for Information Processing. IFIP WG10.4 on Dependable Computing and Fault Tolerance. <http://www.dependability.org/wg10.4/>.
- [5] Helvik (2004). Perspectives on the dependability of networks and services. *Teletronikk*, pages 27–44.
- [6] Hillenbrand, Götze, Müller, and Müller (2005). Role-based AAA for Service Utilization in Federated Domains. Proceedings of 19th DFN Arbeitstagung (Düsseldorf, Germany).
- [7] Hillenbrand, Götze, Müller, and Müller (2005). A Single Sign-On Framework for Web-Services-based Distributed Applications. Proceedings of 8th International Conference on Telecommunications ConTEL (Zagreb, Croatia).
- [8] Hillenbrand, Götze, and Müller (2005). Voice over IP – Considerations for a Next Generation Architecture. Proceedings of 31st EUROMICRO Conference (Porto, Portugal).
- [9] Hillenbrand and Zhang (2004). A Web Services Based Framework for Voice over IP. Proceedings of the 30th Euromico Conference 2004 (Rennes, France).
- [10] Johnson, Kogan, Levy, Saheban, and Tarapore (2004). VoIP Reliability: A Service Provider's Perspective. *IEEE Communication Magazine*, pages 48–54.
- [11] Kharchenko, Popov, and Romanovsky (2004). On Dependability of Composite Web Services with Components Upgraded Online. Proceedings of the International Conference on Dependable Systems and Networks, DSN 2004 (Florence, Italy), pages 287–291.
- [12] European Commission (1991). Information Technology Security Evaluation Criteria – ITSEC. Technical report.
- [13] Looker, Munro, and Xu (2004). Practical Dependability Analysis of SOAP Based Systems. Proceedings of the UK e-Science All Hands Meeting (Nottigham, UK), pages 1126–1129.
- [14] Looker, Munro, and Xu (2004). WS-FIT: A Tool for Dependability Analysis of Web Services. Proceedings of the 28th Annual International Computer Software and Applications Conference – Workshops and Fast Abstracts – COMPSAC (Hong Kong), pages 120–123.
- [15] Tanenbaum and van Steen (2002). *Distributed Systems: Principles and Paradigms*. Prentice Hall.
- [16] Tartanoglu, Issarny, Romanovsky, and Levy (2002). Dependability in the Web Service Architecture. Proceedings of the ICSE 2002 Workshop on Architecting Dependable Systems (Orlando, Florida, USA).
- [17] Tartanoglu, Issarny, Romanovsky, and Levy (2003). Coordinated Forward Error Recovery for Composite Web Services. Proceedings of the 22nd International Symposium on Reliable Distributed Systems, SRDS (Florence, Italy).
- [18] Zhang and Hillenbrand (2004). Implementing SIP and H.323 Signaling as Web Services. Proceedings of the 30th Euromico Conference 2004 (Rennes, France).



Markus Hillenbrand

Markus Hillenbrand has studied computer science at the University of Kaiserslautern, Germany. As a research staff member of the Integrated Communications Systems Lab at the department of computer science at the same University he has worked in several projects ranging from Web portal technology to Voice over IP. His current research interests are distributed systems (Web Services, Grid and P2P) with a focus on dependability, service description, service discovery and service brokering.



Joachim Götze

Joachim Götze is a research-assistant at the department of computer science at the University of Kaiserslautern, Germany. His research focus is on distributed systems, especially Grid and P2P technologies. He received his Dipl.-Inform. in 2004 from the same University.

Prof. Dr. Paul Müller

Before Paul Müller studied mathematics, information-technology and economics at the University of Bochum (Germany), he worked as an engineer with SEL (Alcatel) with a focus on telecommunication.

He started his scientific career as a researcher at the University of Tübingen (Germany) where he developed a large computer-based statistical information system in 1975. In 1981 he changed to the Federal Statistical Office of Germany in Wiesbaden, where he was responsible for the further development and implementation of this statistical system. Furthermore, he designed and implemented several statistical software packages during his time there. In 1982 he started working with the University of Ulm (Germany) where he earned his doctoral degree in mathematics in 1983. Thereafter, he was responsible for various research projects and the development of a state-wide computer network.

In 1995 he accepted an offer from the University of Kaiserslautern (Germany) on a full professor position in the department of computer science in conjunction with heading the university's central computing department. His research group on Integrated Communication Systems (ICSY) is broadly interested in multimedia technology and its requirements with regard to communication systems. Currently, service-oriented architectures (SOA) become an important research topic in the ICSY lab, with the intent to ensure dependability and adaptability. In particular, Web and Grid services providing a flexible and secure foundation for the implementation of open service-oriented systems are in the focus of ICSY.

