Ensuring sustainability of web services dependent processes

Tomasz Miksa* and Rudolf Mayer

Secure Business Austria,
Favoritenstrasse 16, 1040 Vienna, Austria
Email: tmiksa@sba-research.org
Email: mayer@ifs.tuwien.ac.at
*Corresponding author

Andreas Rauber

Institute for Software Technology and Interactive Systems,
Vienna University of Technology,
Favoritenstrasse 9-11, 1040 Vienna, Austria
Email: rauber@ifs.tuwien.ac.at

Abstract: High dependence on web services and service-oriented architecture affects not only business solutions, but also scientific research. Web services may be delivered by third parties, and thus are candidates for outsourcing. However, they represent a source of risks, which can jeopardise the robustness of processes. Hence, there is a need for actions which can contribute to the mitigation of possible threats to the continuity of processes. In this paper, risk affecting processes are classified, followed by a discussion about particular changes stemming from web services. Three distinct approaches allowing improvements are described: a newly proposed web services monitoring framework supported by a software solution, the concept of resilient web services, which specifies new design requirements for web services, and digital preservation strategies, which apart from long-term benefits can support sustainability of currently running processes.

Keywords: web service; resilient; monitoring; service-oriented architecture; SOA; business continuity management; BCM; digital preservation; escrow; eScience; sustainability; business process.

Reference to this paper should be made as follows: Miksa, T., Mayer, R. and Rauber, A. (2015) 'Ensuring sustainability of web services dependent processes', *Int. J. Computational Science and Engineering*, Vol. 10, Nos. 1/2, pp.70–81.

Biographical notes: Tomasz Miksa has been working as a researcher at SBA since 2012. Currently, he is involved in the preservation of business processes in the EU-funded FP7 project TIMBUS. He is also a student of Vienna PhD School of Informatics, where he is conducting his research on eScience, digital preservation and research infrastructures. He received his MSc Engineering degree in Systems and Computer Networks from Wroclaw University of Technology, Poland.

Rudolf Mayer has been working as a Research Assistant at SBA since 2011. He received his MSc in Business Informatics from Vienna University of Technology in 2004, his MSc in Computer Science in 2012, and is currently working towards his PhD. Previously, he has worked for several years as a researcher at the Vienna University of Technology, where he has been involved in numerous national and international research projects, including DELOS, multimedia understand through semantics, computation and learning (MUSCLE), preservation and long-term access networked services (PLANETS), and digital memory engineering (DME). His current research focus lies on the preservation of business processes in the EU-funded FP7 project TIMBUS. He is an author of numerous papers at refereed international conferences, journal articles and book chapters. He is reviewer and co-reviewer for several international conferences.

Andreas Rauber is an Associate Professor at the Department of Software Technology and Interactive Systems (IFS) at the Vienna University of Technology (TU-Wien). Furthermore, he is President of AARIT, the Austrian Association for Research in IT and an Honorary Research Fellow in the Department of Humanities Advanced Technology and Information Institute (HATII), University of Glasgow. He received his PhD in Computer Science from the Vienna University of Technology in 2000. His research interests cover the broad scope of information spaces, including specifically text and music information retrieval and organisation, information visualisation, as well as data analysis and digital preservation.

1 Introduction

Service-oriented architecture (SOA) is a firm foundation of cloud computing. By modular and elastic approaches, SOA allows to create IT architectures which can be adapted quickly to changes of business requirements (Mulholland et al., 2008). Business processes (BP) in SOA consists of discrete, loosely-coupled services which interoperate using a network and have clearly defined communication interfaces. Web services (WSs) are the most common way to realise SOA services. New business needs can be quickly embodied, as services can be reused and easily plugged into new processes, which decreases the time needed for development (van den Berg et al., 2007). Not only internal resources can be accessed with the use of WSs, but also external resources, which are hosted and maintained by third-party organisations. Nowadays information ranging from credit scores to weather data are provided by a small set of organisations as opposed to the large number of organisations which consume this data.

High dependence on WSs and SOA is not limited to the business domain, but can be encountered also in the domain of scientific research. In particular, the emerging paradigm of science referred to as e-Science (Hey et al., 2009) is highly influenced by this approach. The increasing computational power of computers and throughput of information systems have enabled researchers to make their scientific breakthroughs by processing, linking and exchanging multiple huge datasets (Elmroth et al., 2009). These datasets are very often referred to as 'big data' (Thanos et al., 2012). Results obtained by research conducted in this way are said to be 'born digital' (National Science Foundation, 2009). e-Science cooperation is very often realised within research infrastructures (RI), where different stakeholders cooperate and share information to look for solutions to important problems of society. Problems tackled span across several fields, for example energy, food, transportation, climate, societies, or health (Copenhagen Research Forum, 2012). RI rely on information systems and allow exchange of facilities, resources and services. Their cooperation would be hardly possible if SOA architecture and extensive usage of WSs had not been applied.

While WSs bring a wealth of new possibilities and flexibility to business and scientific processes, they also introduce new risks for the process execution. We can distinguish three basic possible threats. Firstly, the WS hosted by a third party can become unavailable, which can bring the execution of the process to a halt. The reasons for such unavailability can range from temporary technical problems, to a permanent outage due to, e.g., bankruptcy of the service provider. Secondly, the WS can change its communication interface. This might not always jeopardise the full execution of the process, but may cause short downtimes in process execution, until the changes will be adopted into the process. Thirdly, the behaviour of the WS may change, while the interface stays the same. Unlike the first two threats, this threat is extremely hard to detect, as the process may not break, but will still deliver the outputs, which, however, might not be correct, or different from what is expected. This jeopardises correct reasoning when making scientific discoveries, as well as endangers business continuity.

Failure to provide business continuity may often lead to significant financial losses and may result in loss of business. A highly widespread approach used to tackle potential adverse circumstances is business continuity management (BCM). The main objective of BCM is to assess possible risks and establish plans which can minimise the negative impact of hazardous events on managed BP. In case of BP which rely on WSs, the process operator may not be the owner of all dependent WSs. In situations when a BP operator must be able to prove correctness of process execution because of liability cases, service level agreements (SLA) or other legal contracts, it is of particular importance to them to be able to detect any alterations in process execution. Thus BCM plans must take into account the threats posed by SOA architecture. Therefore, tools that monitor correct execution of WSs and new requirements for WS implementation are needed.

This paper discusses the issues of ensuring continuous faithful execution of processes in highly distributed environments which use WSs to perform tasks. It outlines major concerns which jeopardise business continuity and pays particular attention to alterations which may occur at WS level. The motivation for the necessity to automate monitoring of WSs is discussed in detail. We also introduce a solution which enables to intercept messages exchanged between a WS and a client, analyse their content, reason about the nature of the analysed WS, and finally create a mock-up of it. Furthermore, a set of recommendations and guidelines enabling categorisation and improvements in the design of WSs are presented. The concept of resilient WSs is shown as a solution which would reduce threats to business continuity and possibly decrease the need for active monitoring of services. Finally, preservation strategies, which allow to secure continuous execution of WSs in their current form, are presented.

The paper is organised as follows. Section 2 presents threats to business process continuity and gives an outlook on monitoring changes in processes. A framework for monitoring WSs for changes is presented in Section 3, the implementation of which is detailed in Section 4. Section 5 introduces the concept of resilient WSs. In Section 6 preservation strategies that enhance sustainability of currently running processes are shown. Related work is discussed in Section 7, and conclusions and future work are provided in Section 8.

2 Business process continuity threats and monitoring

Nowadays the saying 'Change is the only constant factor' is one of the most common business principles (Frame, 2002). External factors are the main drivers that force organisations to continually adapt. This can be caused by financial crisis, legislation and regulations, compliance criteria, market

opportunities, etc. SOA supports the requirement of being ready for changes (Mulholland et al., 2008). However, we argue that the application of SOA principles in the structuring of processes and systems can itself introduce another set of possible change factors affecting the business directly. These are the changes coming from the information and communication technology (ICT) systems in which the SOA solution is deployed. Threats arising from the usage of SOA and especially from WSs are presented later in this section. Owing to both external and internal influencers BP have to be monitored. This is explained in more detail in subsection 'business process monitoring'. This paper deals with changes coming from ICT. The external business drivers, like those presented at the beginning of this paragraph, are not in scope of considerations.

2.1 Business process monitoring

A need to monitor execution of processes is widely recognised and implemented. It is manifested by business activity monitoring (BAM) (McCoy, 2002). The goal of BAM is to provide real time information about the status and results of various tasks and processes, thus enabling to make better business decisions and quickly address detected problems and opportunities. The characteristic feature of BAM solutions are dashboards containing key performance indicators (KPIs), which orchestrate reasoning over the process state and support business process management (BPM) decisions. Ready to deploy solutions are available at portfolio of key players (IBM, HP, Oracle) in BPM.

However, the focus of BAM tools is on monitoring and analysing the processes in view of maximising revenue and modelling business needs. They are powerful tools for managers, but are not able to detect directly any change of execution stemming from changes in the underlying ICT infrastructure – they were designed to facilitate responses to external drivers for change. These BAM tools are business tier tools, while technical tier tools are scarce. Currently, monitoring for the equivalent execution of a process is not common practice.

However, such a monitoring should be done continuously in order to detect any change in the process execution, before it affects a higher tier such as the business tier. Furthermore, all tasks forming a business process must be monitored separately, so that the variations can be detected at the source. Otherwise, alterations in single tasks may not be spotted when the KPIs or outputs of the whole process are monitored at the macro level, as changes simultaneously occurring at several stages of process execution may result in the same KPI, while in fact they will significantly vary in the way they were executed. Only if proper execution of processes is monitored at technical level, a trustworthy business process execution can be guaranteed.

2.2 Sources of change

The ICT changes can be divided into two categories: internal and external. All modifications done in software or hardware owned by the process owner (PO) are classified as internal. The external changes are all these variations which are beyond control of the PO.

2.2.1 Internal changes

This category of changes includes all cases in which the PO has the possibility to influence the behaviour of the system performing a particular task. For example, changes in software or hardware setup may result in alteration of task execution. The problem of tracing the effects of, for example, security updates applied to an operating system or software components, driver updates, hardware reconfigurations are well known, complex problems. When the PO has scheduled some of these actions, he or she should be aware that the correctness of business task execution has to be verified. In the case of internal changes, the PO can plan for these changes, and can very often predict the possible threat.

2.2.2 External changes

This category of changes includes all cases in which the PO does not have much possibility to influence the behaviour of the system which performs a particular business task. In cases when services are hosted or provided by third parties, e.g., in a solution such as described by Hsieh et al. (2011), the PO cannot suspect any changes unless being explicitly informed in advance. For example, if part of the process takes benefit of cloud computing at infrastructure as a service (IaaS) or platform as a service (PaaS) level, and the administrator of the cloud computing environment reconfigures the hardware, modifies the software by installing updates, or the cloud computing operating system allocates different resources than usually, then the business task can be affected. Changes can range from variations in timing characteristics (e.g., higher delays), to unavailability of the service.

2.2.3 Changes in WSs

WSs can be classified under both of above presented categories, as they can be both run in-house or provided, as in their original meaning, remotely by an external party. They are also particular source of changes coming from ICT affecting business process execution. Owing to their widespread application nowadays, a special subsection of this paper focuses on categorising possible ways of their alteration. Four categories have been identified.

Firstly, the WS can become unavailable. This will likely stop execution of the process (unless alternative paths and exception handling has been implemented for such a case). The reasons for its unavailability can range from temporary technical problems, to bankruptcy of the service provider. Such situations are straight forward to detect, for example by using time-outs which would alert to unavailability of the WS.

Secondly, the WS can change its communication interface, not always jeopardising the full execution of the process. Such situation may also be easily detected. It may require short pauses in the process execution until the changes will be adopted into the process. Of course, in case of significant changes in the communication interface (e.g., switch from REST to WSDL), time needed for reconnecting the WS into the process may require more effort.

Thirdly, the functionality of the WS may change, which denotes that the outputs of the WS change, while the interface stays the same. Unlike the first two threats, this threat is hard to detect, as the process may not break, but instead will be delivering outputs which are not correct or different from expected. Such a situation might be detected only much later and on a different level, e.g., in the earlier mentioned KPIs. Such a situation may occur for several reasons. One of them is changes at the semantic level, e.g., switching the unit of measurement from inches to centimetre owing to a server configuration change. Other possibilities are bug fixes in the underlying algorithm (which may introduce other bugs as well), or intentional changes in the functionality, e.g., faster but less accurate computational algorithms.

The fourth category of changes is behavioural changes, which may not always refrain the process from correct execution, but can occur temporally and therefore be hard to notice. The examples of such cases could be different timing characteristics or delays, effects of buffering, etc. They also need to be detected, because there may be a threshold from which the WS cannot deliver its functionality properly.

Both functional and behavioural changes are of big importance when dealing with WSs executed by Humans. BPEL4People (Kloppmann et al., 2005) is an extension of Business Process Execution Language (BPEL) for WSs (OASIS, 2007). It allows to specify tasks with a use of WSDL and interweave software and human tasks in a transparent way with a business process which is deployed in SOA system. People are said to be more error prone than machines and therefore the likelihood of a mistake is higher in case of human-based WS. If a task-assigned person sends back incorrect data, then one can observe functional change. When a person is overloaded with requests or is not available, the responses may appear with a delay or get buffered, then one can detect alteration of behaviour.

3 Web services monitoring framework

The risks incurred by changes mentioned earlier call for a regular and automatic monitoring of the WSs utilised in a SOA. Yet, related work (see Section 7) shows that there is a lack of tools available to monitor BP on a technical tier. We therefore present a web services monitoring framework (WSMF), which allows to detect all possible alterations in WS execution and thus enables the sustainable and trustworthy execution of processes. Please note that in some cases, especially when processes span across several organisations or are not well documented, it may be challenging to identify which tasks depend on WSs. This can be achieved by source code or binary code analysis, or by monitoring and analysing network activity caused by a process. This problem is, however, not in the scope of this paper, and therefore the assumption is made that WS monitoring starts with a list of services provided.

Monitoring of WSs is a process which has several similarities to Test Automation from software testing. In automated testing, the idea is that special software controls the execution of tests and compares the actual outcomes of the software under development to a ground truth. Most tools (cf., Section 7) which allow to verify the behaviour of WSs apply this rule. They vary in ways the WS communication is intercepted, levels of abstraction at which the comparison is made, kinds of WSs it can be applied to, and finally the kind of change which is being examined. Details and examples of solutions are discussed in the Related Work section (cf., Section 7).

The common problem of all mentioned solutions is the fact that they demand some specific knowledge: not only the kind and the nature of the WS, but also the kind of change which will be monitored, is required to deploy a proper solution. In typical situations, however, only the URL and interface of the service are known. There might be no information on whether the WS is conversational, stateful, deterministic, etc. This kind of knowledge is, however, required to apply the correct tool. The monitoring framework presented in this paper does not hold these requirements and limitations. It allows to investigate any kind of WS, and facilitates reasoning about the kind and nature of a service. Then, if the WS is deterministic, the monitoring process can be launched and all four types of changes (cf., Section 2.2.3) can be detected. In case of the WS being non-deterministic, the monitoring framework is not able to detect any functional changes, but the other three types of change can still be monitored.

Our proposed WSMF consists of the following steps.

1 Capture

  As a prerequisite for any subsequent monitoring activities, the communication to and from the service has to be intercepted and stored. If traffic is captured at the network layer, it allows to conduct this process transparently without introducing any changes to the examined part of the business process. Solutions which redirect the communication with the WS through a special proxy server, which would intercept the data and pass on the communication to the final destination, are another possibility, but require changes to the process setup and can themselves be a source of errors.

2 Transform

  Once a sufficiently large set of data is collected, it has to be transformed to a form in which requests and corresponding responses are grouped. Additional metadata and metrics also have to be transformed or

calculated at this step. For example, a pair of request and response can be enriched by number of occurrences, timestamps and calculated interval between sending the request and receiving the response. The more information is collected at this step, the more complex reasoning can be conducted subsequently.

3 Reason

At this step the collected data is analysed, and the type of WS is automatically determined from the data. If for the same request different responses exist, then a WS is deemed to be non-deterministic, otherwise it is deterministic, from the point of view of a (ignorant) observer. When it is deterministic, monitoring for changes in functionality is possible, otherwise not. Reasons for perceived non-determinism can be manifold. In many cases, it will be due to the dependence on a specific state, which could, e.g., be the current date and time.

4 Monitor

Requests collected in the first step are used to query the WS. Responses collected at this step are compared to those collected in the *capture* step. This step is replayed according to the planned schedule, for example every day or every week, etc.

The following situations can occur during Monitor step. If no responses are received, then it may mean that the WS is not available: a change in availability occurred, or a change in the interface caused the unavailability. If only some of the messages are missing, then we can assume that the service is available but only a part of the interface has changed. When the service is deterministic and responses do not match the ground truth, then it indicates a change in functionality. If the service is non-deterministic, changes in functionality cannot be detected easily. If timestamps of recorded messages are stored and time intervals between request and response are calculated, then a change in response timing behaviour can be detected regardless of determinism. The time period it takes to detect such changes is mainly driven by the interval of checks defined in the monitoring schedule.

A crucial requirement for using the approach described above is that the WS it is applied to does not cause any changes on the world outside the system observed. In situations where this is not the case, e.g., credit card payment transaction systems, such replaying of messages for monitoring purposes cannot be employed. Thus, while not universally applicable, the approach will be useful for a majority of situations, specifically in e-Science settings, where WSs are deployed primarily for information transformation, collection or computational services.

4 WSMF implementation

In this section, we detail our implementation of the WSMF described above. The solution consists of three tools:

- wsRecorder
- wsTransformer
- wsMonitor.

wsRecorder performs the actions specified in the first step of the framework, the capture step. It collects the data which will be used as a ground truth. wsTransformer realises the second step of the framework. It transforms the data collected by wsRecorder into a database. The third and fourth step of the WSMF is realised by wsMonitor. It enables to query the analysed WS by sending requests stored in the database created in the preceding step, and compare new responses against the ground truth, performing automatic reasoning on the type of WS. The architecture of this implementation is depicted in Figure 1. It depicts a part of a business process in which Tasks X and Task Y communicate over the network. Task Y is the WS being analysed. The other components depicted in the picture are described in more detail below.

4.1 wsRecorder

This application is responsible for capturing and storing the WS data. It asks for the location of a WSDL file or URL address in case of RESTful services. Via a graphical user interface (see Figure 2) the user can select from a list of available capture devices the one that should monitor the traffic and the location for an output file, which is in the format of a PCAP dump file. PCAP is the *packet capture* API for capturing network traffic. PCAP files can, e.g., also be accessed using WireShark (Combs et al., 2012), a well known packet analysis tool in networking community. wsRecorder is totally transparent to the business process setup and execution, and does thus not require any changes in the ongoing process – no kind of proxy mechanism is needed. wsRecorder collects HTTP packets which have the same destination IP address as defined in the WSDL Port section (SOAP)/URL address (RESTful).

Figure 1 Logical structure of the solution which implements WSMF (see online version for colours)

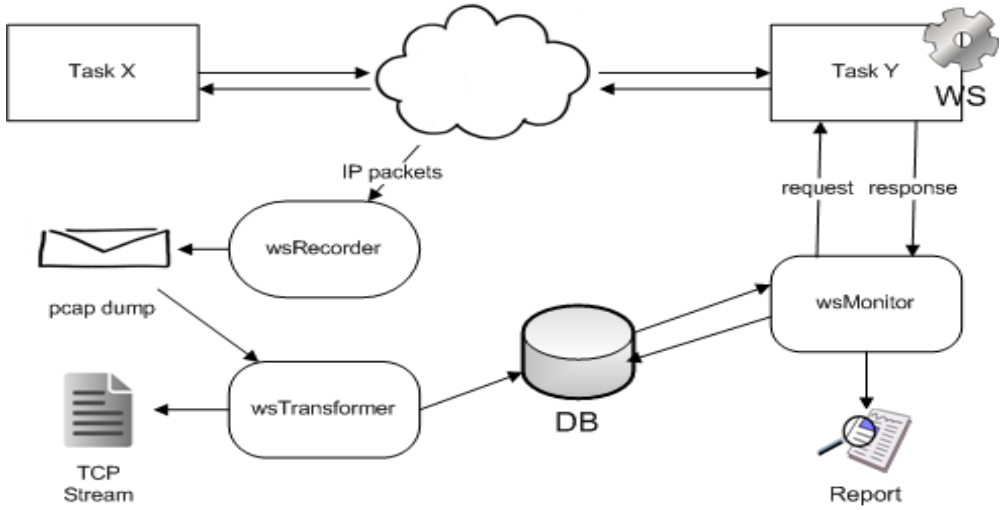


Figure 2 User interface of the wsRecorder module, which allows for capturing of network traffic (see online version for colours)

4.2 wsTransformer

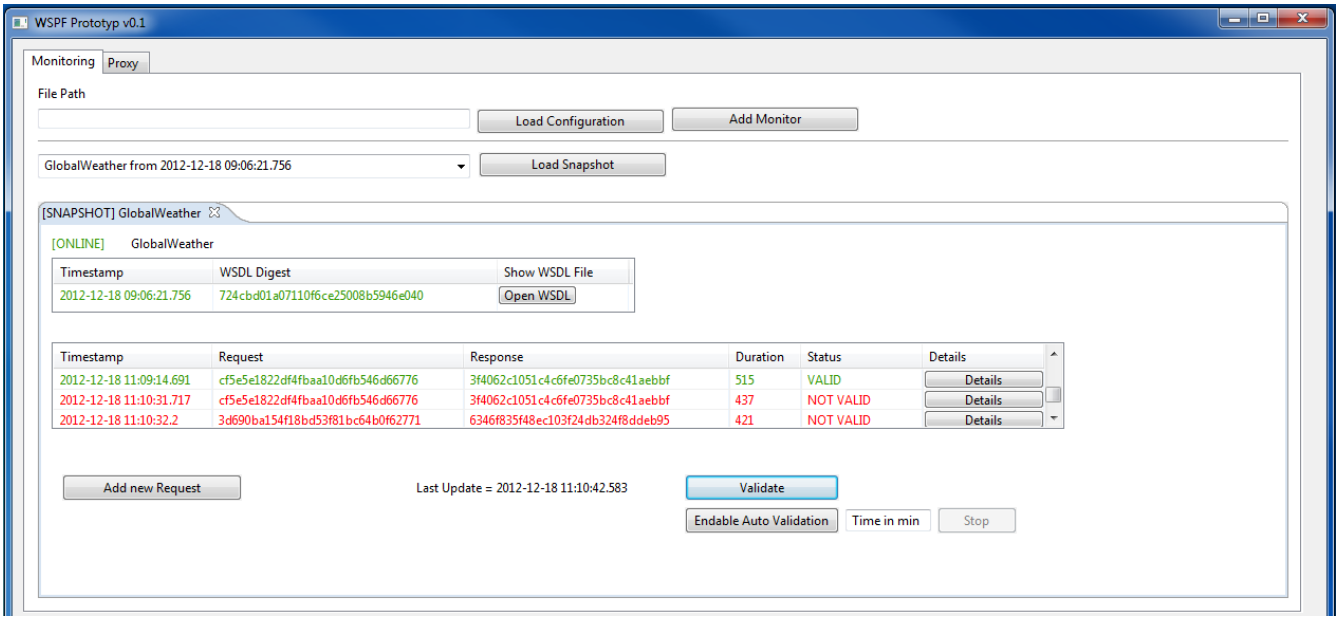
This tool analyses the data obtained by wsRecorder and creates ground truth for wsMonitor. The main task of this program is to transform recorded packets into a set of pairs of requests and responses and stores them in a database. Execution of the program can be divided into two stages. First, a TCP Stream is recreated, taking into account IP fragmenting and TCP segmentation, and stored into a text file. Second, the body of packets is extracted and loaded into the database. Such a design allows to decouple the TCP Stream reconstruction from the database insertion process. The current implementation does not take into the account the HTTP chunking which may occur in some WSs. In such cases, part of the program responsible for TCP Stream creation can be substituted. Currently, there is no Java library that can assemble the TCP Stream correctly with respect to all details. Future implementations of JNetPcap (Bednarczyk, 2012) may provide such functionality. An alternative solution for today is the use of Linux tools like TcpTrace (Ostermann, 2012), but in this case platform independence is lost. The output of the tool are a database and a TCP stream in a text file. The database structures the request and response pairs, together with any metadata captured or computed. This includes timing and duration of the request, and frequency of identical request/response pairs.

4.3 wsMonitor

The wsMonitor (see Figure 3) is capable of sending requests fetched from the database and comparing responses received to the ones stored in the database. The required input is the location of the database and, again, the location of the WS. The output of the tool is a report displaying detected changes or confirmation that no changes occurred. The monitor can detect changes in availability, timing behaviour, in the interface definition (WSDL), and in the functionality. The monitoring component has been tested on a couple of WSs that have been developed in our group, where the following conditions have been simulated:

- unavailability of the service, by stopping the service or moving it to a different location
- change in response behaviour, by adding random delays before processing the requests
- changes in the interface of other services specified in the same WSDL definition, without affecting the service currently tested
- changes in the interface of the service currently tested, by adding a number of bogus parameters
- changes in the functionality of the service, by adding a random component to the computation.

Figure 3 User interface of the wsMonitor module, which allows to monitor for changes in a WS (see online version for colours)



The monitoring framework was able to correctly detect and classify these modifications.

4.4 Comments and future work

Two aspects of the suggested approach have to be emphasised. First, by capturing the network traffic no changes to currently working system have to be done, contrary to solution using HTTP proxies. Future work, however, will focus on implementing also such a proxy solution, to have an alternative way of capturing the communication with the service. Second, the system can support capture of not only WSDL-based WSs but also RESTful services. Moreover, the modular design of the solution allows to substitute or upgrade components without influence on correctness of the execution of the whole process. Finally, the solution allows basic reasoning about frequency of requests and responses, because the database stores the information on how many times the given pair of request and response messages have been detected.

5 Resilient WSs

The WSMF presented in Section 3 is a solution which can be applied to currently deployed services. In fact, the framework is an attempt to minimise negative impacts of insufficient WS specification standards. Analogously to prevention being better than cure, we postulate that a good design is better than ad-hoc actions. Therefore, the aim of this section is to present a set of improvements in the specification of WSs, which should lead to higher sustainability of processes, as well as reducing the need for continuous monitoring. We will call such specified services 'resilient web services'.

We start with a brief overview on the state of the art. The universal description discovery and integration (UDDI) is a registry which holds information on registered WSs. However, the registration of the service is not necessary, nor does the registry contain sufficient additional information on the service, which would allow the user to obtain information on the kind, nature, behaviour, quality, etc. There are some attempts to enrich the purely functional description of WSs (bindings, ports, etc.) with performance aspects, namely quality of service (QoS). These focus mainly on timing aspects, availability, reputation (Comuzzi and Pernici, 2009) and pricing (Liu et al., 2004). Most work is dedicated to the creation of frameworks which enable detection of WSs with different QoS (Tian et al., 2004), rather than solutions which allow to specify explicitly the common qualities for every service. W3C Working Group (2003) specifies requirements for QoS for WSs. It lists 13 points which should be fulfilled, but none of them concerns guaranteeing continuity or non-modifiability. Another approach is represented by works dealing with versioning of WSs. Yet in this case, approaches do not aim at specifying a way to interweave versioning into WS specification, but present workarounds to deal with the currently underspecified WS standards (Kaminski et al., 2006). One of the exceptions to this rule is Kalali et al. (2003), which provides functional requirements for a registry which notifies clients when a version of an interface changes. Kaminski et al. (2006) is a good example of the current common view on versioning. Versioning is understood as a change of interface. Yet, changes in functionality while the interface stays the same are not considered. This is an obvious deficiency. Some best practices are described in Brown and Ellis (2004), but they are predominantly a kind of workaround for only one of the problems, rather than a holistic solution. Summing up, there is a wide range of approaches that deal with assessment of non-functional aspects of WSs. In most cases, the need for these solutions

arises from the fact that current specification of WSs is not sufficient.

Resilient WSs should extend the current specification with information that would ease their long-term sustainability and usage. Thus, resilient WS definitions should also provide information on the quality of the services offered. Here, QoS should also comprise aspects of continuity and non-modifiability. For example, one of the additional qualities could be the 'minimal availability date', which would guarantee minimal deadline for execution of the service in unchanged manner.

The concept of versioning should be revised in order to cover also the changes coming from changes in functionality (i.e., the same interface, but different results) or behaviour (e.g., the same interface and functionality, but faster execution). Resilient WSs should also provide a mechanism to deliver notifications about these changes. Such a mechanism could build on top of HTTP status codes, which would be a solution for both SOAP and REST-based services. For SOAP-based messages, additional information on the changes could be transmitted in dedicated and reserved section of the SOAP envelope. On the consumer side, such notifications could be intercepted and handled accordingly in the program logic. In addition for transmitting such change notifications on request responses, default interfaces could be designed to allow an active querying of WSs, e.g., for the exact version number, or whether changes have occurred since a specified last time of access.

On top of changes to the WS itself, the users of resilient WSs should be provided with notifications about potential functional changes resulting from any modification in the underlying system providing the WS, such as changes to the OS or system. This would allow them to prepare action scenarios in advance which will be deployed when notification about changes arrives.

Another feature useful for the monitoring of WSs that affect the world outside the observed system, such as the previously mentioned payment transactions, would be a testing mode, as it is frequently available for system testing in similar conventional software modules. In such a mode, a computational activity is performed, but not made persistent. This would allow monitoring and testing of a WS by co-activating this feature without actually performing the actual action such as a booking, payment transaction or others.

Among the core methods recommend for resilient WSs, we could consider:

- identifyYourself(), returning the current version number and auxiliary information such as last change date, determinism/statefulness
- serviceChangesSince(Date), returning a log of changes to the WS since the date provided as an argument
- identifySWEnvironment(), returning the (essential) components of the software
- environment, such as the operating system (and version), libraries
- identifyHWEnvironment(), returning the (essential) components of the hardware
- environment, such as sensor specifications
- swEnvironmentChangesSince(Date) and hwEnvironmentChangesSince(Date), returning a log of changes to the environments since the date provided.

Obviously, part of this information might incur vulnerabilities on a security level, the level of detail provided should thus be parametrised on different levels, depending on the trustworthiness of the environment or authentication levels of the client services.

6 Preservation strategies

In this section, we discuss possible *digital preservation* (DP) concepts and actions that can be employed to enhance the long-term sustainability and ensure continuity of both business and scientific processes.

DP can be defined as a set of actions and efforts whose goal is to maintain digital objects accessible in an authentic manner for a long term into the future. Its focus is not only on ensuring physical preservation of content, but also on ensuring logical and semantic preservation. A wide range of strategies is possible and there is no optimal solution for every case. DP has emerged mainly from memory institutions and the cultural heritage sector (National Science Foundation, 2002), where static objects (e.g., images, text documents) are the focus of interest. Nowadays, one can observe a broadening of scope towards the business sector. In these areas, DP increasingly has to deal with preserving interactive software, computing systems or even complete BP or even the whole systems. DP is considered as one of the value adding capabilities, which can be utilised by business institutions, even though their main focus is not DP itself.

Preservation of processes is a multilevel, complex task. Preserving a description of a process is not sufficient itself, especially when the ability to re-enact it in future has to be guaranteed. Several dependencies and various metadata have to be collected. Furthermore, validation mechanisms which can verify if the preserved copy of the process is the faithful copy of the original system need to be deployed (Guttenbrunner and Rauber, 2012). When a process is deployed in SOA system and employs WSs provided by third-parties for its execution, then the problem becomes even more complex. Limited access to services needs special solutions. Some of them are discussed in the following subsections. We would like to emphasise that these solutions are also useful for improving continuity of currently running processes. They bring double benefits to the organisation owning the process. Current threats jeopardising the process are decreased while the long-term availability is improved.

6.1 WS mock-up

When the WS is provided by a third party, and no access to the system hosting the service is provided, a suitable solution in view of DP is to create a mock-up. A mock-up is a simulation of the original WS, in its most basic form built around a kind of lookup table. It is able to send back responses to requests which have previously been recorded from the original system. This approach is thus limited to deterministic services, which always provide the same response for the same request. Furthermore, only messages which have been previously intercepted can be replayed. The solution does not have any computing capabilities as the original system had. However, in many cases the information collected may be sufficient to meet legal requirements or compliance regulations for documenting and proofing a business or scientific process at a client.

We have successfully implemented the described approach. For this purpose, the first three steps of WSMF (cf., Section 3) have been used. wsRecorder (see Section 4.1) and wsTransformer (see Section 4.2) were used to record the real network traffic and transform it into a database. Then a tool called wsPlayer, which is a simple HTTP Server, was created. When a request is received by wsPlayer, it extracts the body of HTTP packet, queries the database created by wsTransformer and sends back the response. If the given request existed in the database, a packet with body obtained from database is sent. In other cases, an HTTP error code is sent.

WS mock-up is useful for currently active processes. If the original service is being monitored and changes to its execution have been detected (e.g., unavailability), then the process flow can be switched temporarily to the mock-up until the source of the problem is identified and resolved. When a WS is permanently unavailable, a substitute has to be found. The mock-up can be used as a role model to validate the identity of a proposed substitute WS. The requests stored by the mock-up can be leveraged to query the substitute, and responses received can be compared against responses of the mock-up. This approach increases the likelihood of matching a correct substitute in comparison to relying on WS description only. Similar concept which uses provenance data to find substitutes for missing tasks in scientific workflows is described by Belhajjame et al. (2011).

6.2 Escrow services

“Escrow is an ancient legal term referring to a deed which only becomes effective upon the occurrence of a future event” (CEN Workshop Agreement CWA 13620-1, 1999). Nowadays this term is used when some (non)material objects are deposited by the owner with an independent third party known as the ‘escrow agent’. The agent holds the objects according to a tripartite agreement made between the agent, owner and the software user. In case of computer science and software engineering, one can distinguish between Source Code Escrow, where only the code of the application is deposited, and Software Escrow, which is more holistic approach (Draws et al., 2011). Software Escrow encompasses the source code, but also libraries, test cases, virtual machines, integrated development environments, etc. There are several reasons which can trigger execution of an Escrow agreement, for example when the organisation owning the software goes bankrupt. Then the deposited objects are handed to the customer who can continue to use and maintain the product on its own.

Similar kinds of agreements could be applied to WSs. When a PO decides to include external third party service in their process, they can sign an agreement with the service owner, which will state that when the service becomes unavailable, the PO gets access to the underlying software of the WS. This could enable the PO to redeploy the service at his site and thus provide sustainability of business process. This, of course, has to be a transitive solution, requesting the WS provide to ensure identical Escrow agreements with any third-party WS used by its own service. Which artefacts should be deposited by the Escrow agent, and the conditions that trigger execution of the agreement are a broad topic out of scope of this paper.

7 Related work

There are several approaches that tackle the problem of WS monitoring, testing or validating. The first group of papers have the WS-BPEL language as a common denominator. The framework presented in Cao et al. (2010) generates and executes automatically ‘online’ tests for conformance testing of a composite of WSs described in BPEL. This ‘online’ approach has been combined with passive testing, that verifies time traces with respect to a set of constraints in Cao et al. (2011). Both papers are limited to WSs which are implemented according to BPEL specification. When the specification is unavailable, the methods cannot be applied. Our framework does not have these limitations. It does not require any *a priori* knowledge about WS internal details. Furthermore, our implementation uses intercepted messages from a real system, while authors of above papers generate them. van der Aalst et al. (2008) address the problem of checking how the actual behaviour of a service conforms to the expected behaviour. A process model is defined in BPEL and then transformed into Petri-nets. Conformance checking is performed by comparing Petri-net against recorded messages. This work focuses on a question if “the service behaviour match the service specification”, while in the WSMF we focus on detecting changes in the execution of WSs which may affect process continuity. We use intercepted messages from a live system as a ground truth. In contrary to van der Aalst et al. (2008) we assume that no other specification than WSDL or REST is given.

Verification of behavioural conformance of services during run-time is presented in Dranidis et al. (2009). An idea to apply Stream X-machines in order to check the control flow of a WS and the values of the data in the generated responses is shown. A classification of WSs is also carried out in this paper. They distinguish three major

criteria: conversational/non-conversational, private-state/ shared-state, transient-state/persistent-state. Similar to our work, Dranidis et al. (2009) intercepts traffic from a live system and provides continues monitoring for changes. However, the capturing is done with a use of proxy component and manual development of a Stream X-machine is required. The other implies access to the WS specification which limits the application of this method in comparison to the WSMF.

The WS-TAXI framework, which seems to have higher applicability during WS development and testing rather than for monitoring of already deployed SOA solutions, is presented in Bartolini et al. (2009). It combines the coverage of WS operations with data-driven test generation. It is able to deliver a complete suite of test messages ready for execution, which were generated using a WSDL file. Assumption that WSDL could be the only available specification of a WS is in common with our approach. Yet, the WS-TAXI generates and uses purely synthetic data which may be quite different from the data exchanged in a process. This is a drawback which does not exist in the WSMF proposed by us.

Monitoring if SLA conditions are fulfilled by WSs is a problem related to monitoring WSs for changes. In Goel and Shyamasundar (2010) a run-time monitoring framework which allows to concurrently access exchanged messages and compare them against designed scenarios was designed and developed. The work focuses on QoS aspects, an example of time-out mechanism detecting unavailability of the service is given. The same result can be achieved with a use of our framework. Moreover, Goel and Shyamasundar (2010) requires to model the monitored WS in Orc language, whereas our solution can be immediately deployed. In case of Goel et al. (2011) emphasise is put on detection of violations at functional level. SLAs are described formally using temporal logic and are used in SLA monitor to verify the behaviour of WSs at runtime. The authors demonstrate the capabilities of their solution on an example of a detection of violation of maximum response time. Our framework is also capable of doing it without a necessity of being an expert in temporal logics.

Versioning of WSs is discussed by Kaminski et al. (2006) and Kalali et al. (2003). The former introduces a technique called Chain of Adapters which enables to deploy multiple versions of a WS simultaneously. Versioning is understood as being compatible with several interfaces which is not sufficient according to the concept of resilient WSs. The second paper proposes creation of a service-oriented monitoring registry (SOMR) which notifies service requesters when a version of an interface changes or a service becomes disabled. The authors of SOMR express the demand for notification of changes as we do, but in their opinion a centralised registry is needed, while we believe that inbuilt mechanisms of resilient WSs are a better solution. Tian et al. (2004) suggests to use an ontology to enable QoS-aware selection and monitoring of WS. They try to "close the gap between the WSs layer and the underlying QoS-aware transport technologies". Liu et al. (2004) presents a QoS computation model for WS selection. Experimentation with a QoS registry in a hypothetical phone service provisioning is presented. The authors use the following qualities to define QoS: execution duration, reputation and execution price. Both of the above papers emphasise that description of WSs should incorporate information on quality metrics. This opinion is also reflected in the concept of resilient WSs.

Comuzzi and Pernici (2009) presents a framework for QoS-based WS contracting. It focuses on automation of the WS contract specification and establishment. QoS dimensions are also specified in this work and concern mainly timing, reputation and routing. It seems that the matchmaking mechanism described in Comuzzi and Pernici (2009) could be adjusted to take advantage of resilient WSs methods. For example standardised methods to obtain information about resilient WS could speed up the process of finding a substitute for a faulted service.

8 Conclusions

This paper discussed the problem of providing business continuity to processes deployed in highly distributed systems that leverage SOA and use WSs for their implementation. We outlined the current status of processes monitoring and motivations for improvements. A classification of possible change sources affecting processes was conducted and special attention was drawn to changes stemming from WSs. Furthermore, three approaches allowing improvements have been discussed. First, we discussed a WSMF that provides active monitoring of the WSs employed in a business process, and thus allows to detect disruptions at an early stage. The approach represented by our framework provides active monitoring of WSs for changes. It creates a system capable of detecting all kinds of change in WSs identified in this paper. In addition to this, an implementation of the framework was presented, in order to prove the feasibility of the framework. Secondly, this paper introduced the concept of resilient WSs as another way of ensuring sustainability of processes. We provide a set of recommendations and guidelines that would allow notification on changes in WSs without the need of active monitoring. Finally, the applicability of DP strategies is shown, namely a WS mock-up and software escrows as a contractual alternative.

Future work will focus on guidelines for improvements in WSs design. An attempt to embody the concept of resilient WSs has to be made. These improvements should lead towards use of WSs in the same way as with commercial off-the-shelf (COTS) software or hardware products. They are designed to be integrated easily with existing systems without the need for customisation. COTS modules are easier to trace for changes, but still cause difficulties in complex IT landscapes as experience shows. Unless this vision can be achieved, development in active

monitoring of processes not only at the business but also at the technical tier is required.

Acknowledgements

This research was co-funded by COMET K1, FFG – Austrian Research Promotion Agency and by the European Commission under the IST Programme of the 7th FP for RTD – Project ICT 269940/TIMBUS.

References

Bartolini, C., Bertolino, A., Marchetti, E. and Polini, A. (2009) 'WS-TAXI: a WSDL-based testing tool for web services', in *ICST '09, International Conference on Software Testing Verification and Validation*, pp.326–335, April, doi: 10.1109/ICST.2009.28.

Bednarczyk, M. (2012) 'JNetPcap' [online] http://jnetpcap.com/ (accessed 30 August 2012).

Belhajjame, K., Goble, C., Soiland-Reyes, S. and De Roure, D. (2011) 'Fostering scientific workflow preservation through discovery of substitute services', in *2011 IEEE 7th International Conference on E-Science*, pp.97–104, December, doi: 10.1109/eScience.2011.22.

Brown, K. and Ellis, M. (2004) 'Best practices for web services versioning' [online] http://www.ibm.com/developerworks/webservices/library/ws version/ (accessed 30 August 2012).

Cao, T-D., Castanet, R., Felix, P. and Morales, G. (2011) 'Testing of web services: tools and experiments', in *Services Computing Conference (APSCC)*, 2011 IEEE Asia-Pacific, pp.78–85, December, doi: 10.1109/APSCC.2011.23.

Cao, T-D., Felix, P., Castanet, R. and Berrada, I. (2010) 'Online testing framework for web services', in *Third International Conference on Software Testing, Verification and Validation (ICST)*, pp.363 –372, April, doi: 10.1109/ICST.2010.11.

CEN Workshop Agreement CWA 13620-1 (1999) *ESCROWGUIDE – Source Code Escrow – Guidelines for Acquirers, Developers, Escrow Agents and Quality Assessors – Part 1: An Introduction to Source Code Escrow*, June.

Combs, G. et al. (2012) 'WireShark' [online] http://www.wireshark.org/ (accessed 30 August 2012).

Comuzzi, M. and Pernici, B. (2009) 'A framework for QoS-based web service contracting', *ACM Trans. Web*, July, Vol. 3, No. 3, pp.10:1–10:52, ISSN 1559-1131. doi: 10.1145/1541822.1541825 [online] http://doi.acm.org/10.1145/1541822.1541825.

Copenhagen Research Forum (2012) *Visions for Horizon 2020*, Copenhagen.

Dranidis, D., Ramollari, E. and Kourtesis, D. (2009) 'Run-time verification of behavioural conformance for conversational web services', in *ECOWS '09. Seventh IEEE European Conference on Web Services*, pp.139–147, November, doi: 10.1109/ECOWS.2009.19.

Draws, D., Euteneuer, S., Simon, D. and Simon, F. (2011) 'Short term preservation for software industry', in *Proceedings of the 8th International Conference on Preservation of Digital Objects (iPres 2011)*, pp.130–139.

Elmroth, E., Nylen, M. and Oscarsson, R. (2009) 'A user-centric cluster and grid computing portal', *International Journal of Computational Science and Engineering (IJCSE)*, July, Vol. 4, No. 2, pp.127–134.

Frame, J.D. (2002) *The New Project Management: Tools for an Age of Rapid Change, Complexity, and Other Business Realities*, John Wiley & Sons, San Francisco.

Goel, N. and Shyamasundar, R.K. (2010) 'Automatic monitoring of SLAs of web services', in *Services Computing Conference (APSCC)*, 2010 IEEE Asia-Pacific, pp.99–106, December, doi: 10.1109/APSCC.2010.58.

Goel, N., Kumar, N.V.N. and Shyamasundar, R.K. (2011) 'SLA monitor: a system for dynamic monitoring of adaptive web services', in *Web Services (ECOWS)*, *Ninth IEEE European Conference on*, pp.109–116, September, doi: 10.1109/ECOWS.2011.22.

Guttenbrunner, M. and Rauber, A. (2012) 'A measurement framework for evaluating emulators for digital preservation', *ACM Trans. Inf. Syst.*, May, Vol. 30, No. 2, pp.14:1–14:28, ISSN 1046-8188, doi: 10.1145/2180868.2180876 [online] http://doi.acm.org/10.1145/2180868.2180876.

Hey, T., Tansley, S. and Tolle, K. (Eds.) (2009) *The Fourth Paradigm: Data-Intensive Scientific Discovery*, Microsoft Research, Redmond, Washington.

Hsieh, M-Y., Lin, H-Y. and Li, K-C. (2011) 'A web-based travel system using mashup in the RESTful design', *International Journal of Computational Science and Engineering (IJCSE)*, Vol. 6, No. 3, pp.185–191.

Kalali, B., Alencar, P. and Cowan, D. (2003) 'A service-oriented monitoring registry', in *Proceedings of the 2003 Conference of the Centre for Advanced Studies on Collaborative Research, CASCON '03*, pp.107–121, IBM Press [online] http://dl.acm.org/citation.cfm?id=961322.961340.

Kaminski, P., Muller, H. and Litoiu, M. (2006) 'A design for adaptive web service evolution', in *Proceedings of the 2006 International Workshop on Self-adaptation and Self-managing Systems, SEAMS '06*, pp.86–92, ACM, New York, NY, USA, ISBN 1-59593-403-0, doi: 10.1145/1137677.1137694 [online] http://doi.acm.org/10.1145/1137677.1137694.

Kloppmann, M., Koenig, D., Leymann, F., Pfau, G., Rickayzen, A., von Riegen, C., Schmidt, P. and Trickovic, I. (2005) 'WS-BPEL extension for people – BPEL4People', White paper, IBM/SAP, July.

Liu, Y., Ngu, A.H. and Zeng, L.Z. (2004) 'QoS computation and policing in dynamic web service selection', in *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters*, *WWW Alt. '04*, pp.66–73, ACM, New York, NY, USA, ISBN 1-58113-912-8, doi: 10.1145/1013367.1013379 [online] http://doi.acm.org/10.1145/1013367.1013379.

McCoy, D.W. (2002) *Business Activity Monitoring: Calm Before the Storm* [online] http://www.gartner.com/resources/105500/105562/105562.pdf (accessed 30 August 2012).

Mulholland, A., Daniels, R. and Hall, T. (2008) *The Cloud and SOA – Creating an Architecture for Today and for the Future*, Capgemini and HP [online] http://www.hp.com/hpinfo/analystrelations/wp_cloudcomputing_soa_capgemini_hp.pdf (accessed 30 August 2012).

National Science Foundation (2002) 'It's about time: research challenges in digital archiving and long-term preservation', Final Report of the *Workshop on Research Challenges in Digital Archiving and Long-Term Preservation*.

National Science Foundation (2009) 'Harnessing the power of digital data for science and society', Report of the Interagency Working Group on Digital Data to the Committee on Science of the National Science and Technology Council.

OASIS (2007) *Web Services Business Process Execution Language (WS-BPEL) Version 2.0*, Organization for the Advancement of Structured Information Standards, April.

Ostermann, S. (2012) 'TcpTrace' [online] http://www.tcptrace.org/ (accessed 30 August 2012).

Thanos, C., Manegold, S. and Kersten, M.L. (2012) 'Big data – introduction to the special theme', *ERCIM News*, Vol. 89.

Tian, M., Gramm, A., Ritter, H. and Schiller, J. (2004) 'Efficient selection and monitoring of QoS-aware web services with the WS-QoS framework', in *Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence, WI '04*, pp.152–158, IEEE Computer Society, Washington, DC, USA, ISBN 0-7695-2100-2, doi: 10.1109/WI.2004.60 [online] http://dx.doi.org/10.1109/WI.2004.60.

van den Berg, M., Bieberstein, N. and van Ommeren, E. (2007) *SOA for Profit, A Manager's Guide to Success with Service Oriented Architecture*, IBM Press, ISBN 9075414145, 9789075414141 [online] www.us.sogeti.com/it-research-insight/ebook/SOA_for_Profit_Eng.pdf (accessed 30 August 2012).

van der Aalst, W.M.P., Dumas, M., Ouyang, C., Rozinat, A. and Verbeek, E. (2008) 'Conformance checking of service behavior', *ACM Trans. Internet Technol.*, May, Vol. 8, No. 3, pp.13:1–13:30, ISSN 1533-5399, doi: 10.1145/1361186.1361189 [online] http://doi.acm.org/10.1145/1361186.1361189.

W3C Working Group (2003) 'QoS for web services: requirements and possible approaches' [online] http://www.w3c.or.kr/kr-offce/TR/2003/ws-qos/ (accessed 30 August 2012).