FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

# Resilient Web Services for Preservation of Business Processes

FRAMEWORK CONCEPT

**Johannes Kurz**
0727957
j.kurz@gmx.at

**Stefan Weghofer**
0825465
e0825465@student.tuwien.ac.at

# 1 Motivation

Service-oriented Architectures (SOAs) are widely used when dealing with business processes. A convenient way to implement SOAs is using web services. Functional properties of web services can be described in a standardized way (WSDL), as can non-functional properties be (WS-Policy). However, the supported non-functional properties are limited and not ideal for use in Digital Preservation scenarios. To enable easier preservation of web services and the whole business process they represent, the concept of **resilient web services** was introduced in (Miksa, Mayer & Rauber 2013). This concept consists of several approaches to make web services better preservable:

- Enhancing the non-functional properties of the WS specification for fields like continuity, minimal available date, etc.
- Users of resilient web services should be able to detect functional changes to the service. This can either happen through notifications by the service or by providing means for querying the service version and last service changes
- Service behavior or functionality might change due to changes to the underlying system (either hardware or software). Resilient WSs should notify users when changes occur or provide means to let users query for such changes

To summarize this, the specification of resilient web services as well as the functionality offered by them should be enhanced. Our task is to provide a framework which allows the transformation of a regular web service to a resilient one in a simple fashion. Especially the following methods should be provided by the resilient web services:

- identifyYourself()
- identifySWEnvironment()
- identifyHWEnvironment()
- serviceChangesSince(Date)
- swEnvironmentChangesSince(Date)
- hwEnvironmentChangesSince(Date)

Although some of the information returned by these methods is security relevant, we do not focus on security aspects such as authentication. Our focus rather lies on gathering information and providing an easy was to enrich an existing web service with this information.

# 2 Architectural Approach

We will split our framework into two separate parts: information gathering and service enrichment. Figure 1 provides an overview of the components involved in our solution. We will describe each component in more detail in the next sections.

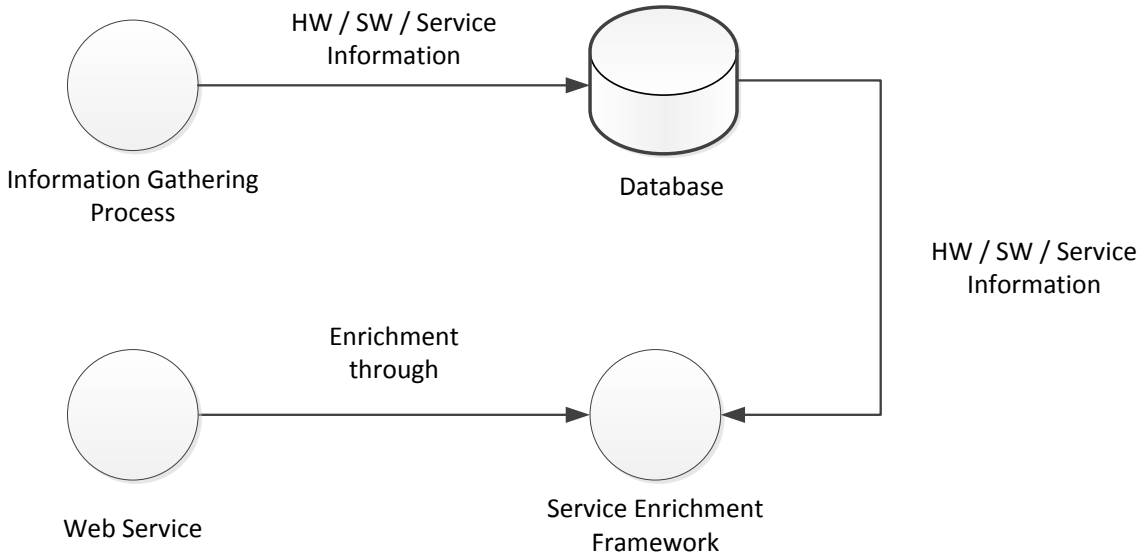


*Figure 1: System Overview*

## 2.1 Information Gathering

We will create a process to gather all relevant information about the hardware and software environment of the system the web services are hosted on. This process will gather a snapshot of the systems hardware and software in a periodic fashion (e.g. every day at 3am). Furthermore, it will gather information about changes to the web service directory (e.g. file updates). The information will be stored in a database and can be accessed by the Service Enrichment Framework. The information gathering process is the only OS-dependent part of our implementation. Although we will provide interfaces to the database that will allow OS-independent implementations, our example implementation will only cover Microsoft Windows 7.

## 2.2 Web Service Enrichment Framework

The second part of our work deals with the enrichment of existing web services. As described in Chapter 1, we want the resilient web services to provide several methods additionally to their business logic. To allow an easy adoption of existing web services, we will provide a base class for a programming language of our choice (Java). This base class will implement the suggested methods for resilient web services. Therefore, any sub class inherits the resilient web service functionality.

## 2.3 Database

We will use MongoDB as database for our framework. The database has to be accessed from two different components: the data is inserted from the Information Gathering Process and read by the Service Enrichment Framework. We will create two separate interfaces for each of the access situations.

# 3 Bibliography

Miksa, T, Mayer, R & Rauber, A 2013, 'Ensuring sustainability of web services dependent', *International Journal of Computational Science and Engineering (IJCSE)*.