Decision Support for Web Service Adaptation

Apostolos Papageorgiou[a,*], André Miede[b], Stefan Schulte[c], Dieter Schuller[d], Ralf Steinmetz[d]

[a]*NEC Laboratories Europe*
[b]*Hochschule für Technik und Wirtschaft des Saarlandes*
[c]*Technische Universität Wien*
[d]*Technische Universität Darmstadt*

Abstract

With the Internet of Services, Web services from all areas of life and business will be offered to service consumers. Even though Web service technologies make it easy to consume services on arbitrary devices due to their platform-independence, service messaging is heavyweight. This may cause problems if services are invoked using devices with limited resources, e.g., smartphones. To overcome this issue, several adaptation mechanisms to decrease service messaging have been proposed. However, none of these are the best-performing under all possible system contexts.

In this paper, we present a decision support system that aims at helping an operator to apply appropriate adaptation mechanisms based on the system context. We formulate the corresponding decision problem and present two scoring algorithms – one Quality of Service-based and one Quality of Experience-based.

Missing data and, thus, an incomplete system context is a serious challenge for scoring algorithms. Regarding the problem at hand, missing data may lead to errors with respect to the recommended adaptation mechanisms. To address this challenge, we apply the statistical concept of imputation, i.e., substituting missing data. Based on the evaluation of different imputation algorithms used for one of our scoring algorithms, we show which imputation algorithms significantly decrease the error imposed by the missing data and decide whether imputation algorithms tailored to our scenario should be investigated.

Keywords: Web Services, Pervasive Computing, QoS, QoE

1. Introduction

The list of advantages of combining Web service technologies and mobile computing is long and compelling [1]: Most notably, the outsourcing of data- and processing-intensive software tasks from mobile devices to more powerful systems is a major reason for the usage of Web service technologies on mobile devices like smartphones. Furthermore, Web services enable quick mobile application development through the reuse of existing software artifacts.

Web service message formats are characterized by a verbose, self-descriptive nature. On the one hand, this leads to platform-independence and high interoperability. On the other hand, it renders service messaging heavyweight due to the high latency and communication overhead in standard SOAP transport protocols like SOAP-over-HTTP and SOAP-over-TCP [2]. Even though mobile devices have evolved into full-fledged computing devices, there is still a gap between their computational power and the available connection bandwidth. Notably, the data volumes that can be processed by mobile devices and the bandwidth available are growing at the same pace[3]. Furthermore, not all mobile devices that may consume a Web service are necessarily from the latest smartphone generation, i.e., devices with less computational power than the most recently unveiled smartphones may consume Web services. As a result, the mentioned gap will not vanish in the future, which makes it necessary to research technical solutions that are capable to overcome the limitations resulting from it.

The communication overhead introduced by Web service standards like the Web Service Description Language (WSDL) or SOAP, can lead to unacceptable Quality of Service (QoS) and Quality of Experience (QoE). Thus, standard transport protocols are not always a good match for the resource-constrained nature of mobile, wireless devices.

[*]Corresponding author: Apostolos.Papageorgiou@neclab.eu

Since the birth of pervasive computing, the adaptation of the communication in order to enhance the QoS of applications has been one of the biggest concerns in the field [4]. Such adaptations can be performed on different levels, e.g., on the level of the communication channel, such as in the much investigated *Always Best Connected* (ABC) issue, or on a higher level of the OSI model [5], as is done during Web content adaptation. Another possibility appears at the level of software services, where the protocol (or the access method) that is used to communicate with particular services is adjusted to the system context. It is at this level and with such protocol or access method adaptations that the rest of this paper handles the issue of mobile Web service performance.

Not surprisingly, a number of adaptation mechanisms for Web services have appeared. An *adaptation mechanism* means the re-offering of a Web service with a different protocol or access method, e.g., Wireless SOAP, JAVA RMI, or SOAP-over-UDP [6]. As it has been shown in our former work [6], the beneficial effects of existing Web service adaptation mechanisms depend not only on the Web service, but also on the system context in terms of device capabilities or the network connection. Thus, provided that no single adaptation mechanism is the best-performing under all possible system contexts [6], an algorithm for decision support is needed. This algorithm needs to score how well the possible adaptation mechanisms match the particular context. Different decision support algorithms would have a different perception of what a "good match" is.

Within this paper, we present two *scoring algorithms* for decision support, with the first one being based on QoS, while the second one being based on QoE. These scoring algorithms rely on the use of historical data, i.e., the *system context* of former Web service invocations. Quite often, it is difficult to get a complete system context due to transmission errors, reluctance, or inability of the data source to provide the data. Therefore, missing data, e.g., in the form of incomplete data logs, significantly deteriorates the outcome of a scoring algorithm with respect to the scored adaptation mechanisms. In our previous work [7], we have presented first evaluation results of these algorithms and have identified the need for reducing the serious impact of missing data. In order to overcome this issue, we use the statistical concept of *imputation*, i.e., substituting missing data with other values, and evaluate how using different imputation algorithms for one of our scoring algorithms affects the quality of its scoring results. The goals of this evaluation are, first, to identify which state-of-the-art algorithm achieves the best results and, second, to decide whether new imputation algorithms specifically dedicated to our scenario should be investigated or not.

The remainder of the work at hand is organized as follows: First, we present some background information that is necessary for the understanding of our work, namely the *Internet of Services* (IoS) scenario applied in this paper, the *Mobility Mediation Layer*, and some general information about the usage of proxies for the adaptation of Web services. In Section 3, we formulate the decision problem that is at the core of our work. Afterwards, we present the two QoS- and QoE-based scoring algorithms for decision support. In Section 5, we introduce state-of-the-art imputation algorithms that address the challenge of missing data. Afterwards, the imputation algorithms are applied to our QoS-based scoring algorithm and evaluated (Section 6). We discuss related work in Section 7 and, finally, conclude this paper in Section 8.

2. Background

2.1. The Internet of Services Scenario

In short, the Internet of Services (IoS) refers to a globalization of service-oriented solutions, where Web services are offered by different providers through global services marketplaces. The IoS should be understood as a future scenario for service-orientation [8, 9]. The realization of the IoS is supported and accelerated by certain enabling technologies, such as the *Unified Service Description Language* (USDL) [10], which describes the business and operational aspects of a Web service in addition to the technical details. Thus, Web services are turned into perfectly tradable goods.

In the following compilation, we list features of the IoS that create new challenges and opportunities for performing Web service adaptations with the use of a mediating platform or middleware, as envisioned in the work at hand:

- *Many Web services gathered within a marketplace*: If service marketplaces offer homogeneous, easy access to a large number of Web services, particular adaptation mechanisms can be performed at once for many of them.

- *Less predictable Web service usage characteristics*: Traditionally, Web services have been developed for a set of consumers that has been more or less known a priori. In a scenario where Web services are published as tradable goods, it is much more difficult to predict under which system conditions the services will be actually used. For service adaptation mechanisms, this means that they should consider a wide spectrum of different possible systems contexts.

- *Less control or influence over third-party Web services*: Another side-effect of the loose relationship between service providers and consumers at a marketplace is the fact that consumers have little influence on the implementations of third-party services. This means that adaptation mechanisms that need any kind of modifications in the code or the hosting system of the services do not come into question in the IoS (cf. Section 7).

As a result of the last list item, the software that hosts the adaptation mechanisms is expected to lie inside the control sphere of an IoS stakeholder (e.g., a service broker) who differs from the Web service provider. The most obvious solution for adapting Web services without access to the provider system is through the generation of proxies in a mediation layer [1]. This mediation layer retrieves service descriptions from service marketplaces in the IoS, generates proxies based on these descriptions, and offers an enhanced access to the original services. Such a *Mobility Mediation Layer* (MML) is presented in the next subsection.

2.2. *The Mobility Mediation Layer*

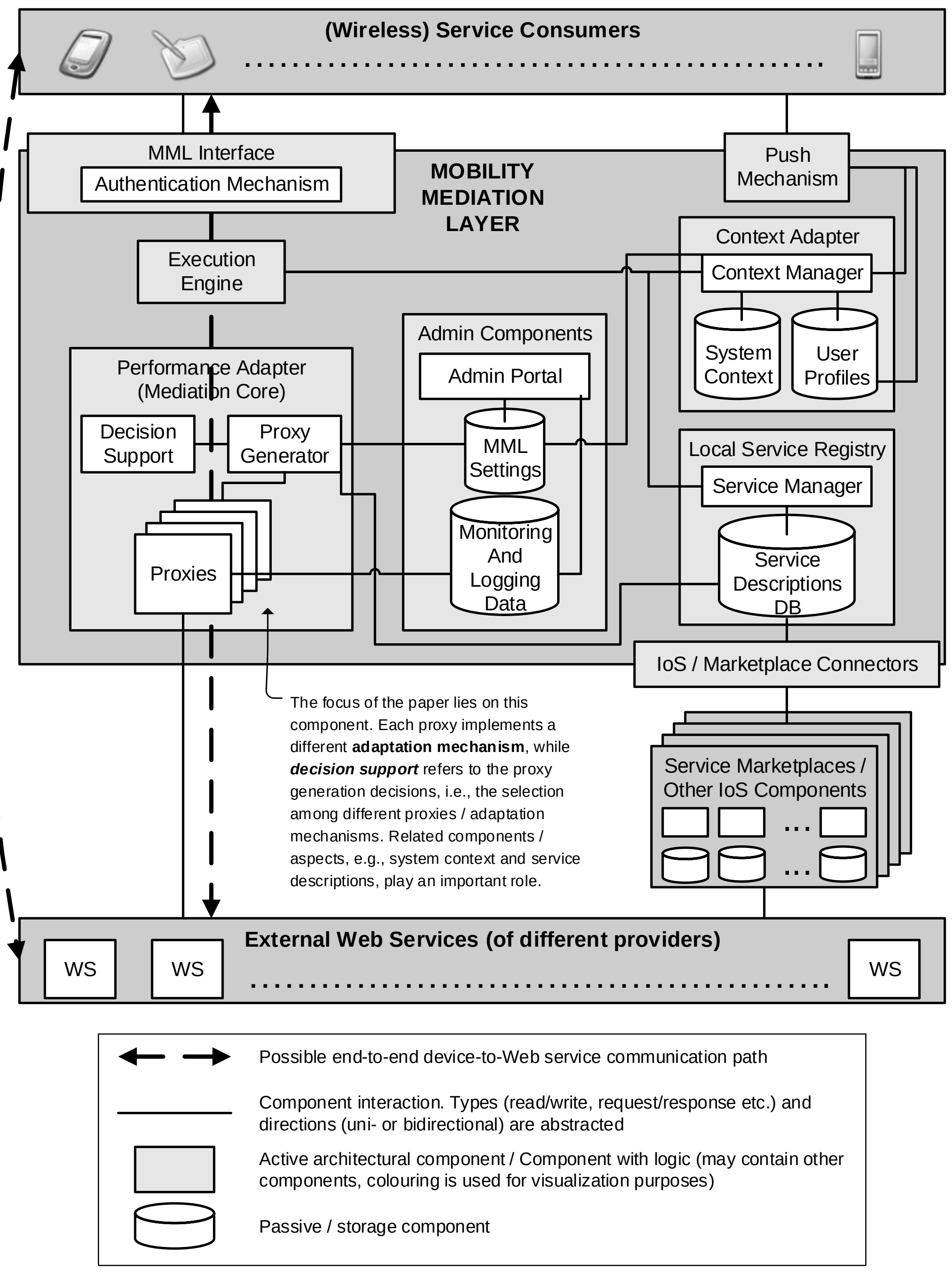
The MML has been conceived as a service adaptation layer that suits the IoS scenario as presented above, i.e., it has been designed for working with service marketplaces that give access to various services, it assumes no access to the implementations or the hosting systems of the services, and it focuses on the types of adaptation that are dictated by the needs of mobile devices. Such a layer could be operated by a service marketplace host that desires to offer services with different access methods or communication protocols, by a mobile application developer that desires to adapt the way services are consumed by the developer's application, or by any stakeholder that has the business model of offering enhanced access to existing third-party services.

Figure 1 shows the MML architecture. As abstractly shown in the figure, the goal of the MML is to provide clients with limited resources or mobile clients with an interface to services which are hosted on various external providers and are made available through global service marketplaces, while mediating the service consumption by performing various adaptation mechanisms. Although the MML has further capabilities (e.g., automatic context enrichment), we focus in the work at hand on the overhead reduction through the generation of proxies. More concretely, the MML includes also modules for user context exploitation, e.g., modules that retrieve context information such as user preferences or personal data (e.g., user address) in order to better serve Web service requests. However, this paper is not concerned with *user context* any further. It is rather concerned only with Web service access adaptation through proxies in order to enhance performance based on *system context*.

The MML can, of course, be accessed by all types of clients. However, its target group are mobile and wireless, often constrained, Web service clients. Thus, these clients can substitute their direct Web service calls (left dashed arrow in Figure 1) with mediated calls through the MML (right dashed arrow in Figure 1). For that purpose, they use the *MML Interface*[1], which also includes an *Authentication Mechanism*.

The calls are then handled by the *Execution Engine*, which involves the *Context Manager* in order to personalize the calls and perform automated context enrichment, and the *Service Manager* in order to find the appropriate services (and their *Proxies*) that will be used. Particular approaches for implementing or enhancing the application-spanning personalization and context-enrichment of the service calls are out of scope of this paper and thus not examined any further. The actual call of the external Web service is then performed by a proxy, which may vary from a simple "dummy" request/response forwarder to the enforcer of any adaptation mechanism. The proxies are generated by the *Web Service Proxy Generator*. Further details about this are provided in the next subsection.

[1]Through the MML Interface, the client implicitly tells the MML which Web service it wants to connect to, though sometimes there are also exchangeable external WS. In our implementation, the MML interface is an homogeneous REST interface of the form `http://ip:port/mml/externalServices/serviceX/parameters`, which redirects to the appropriate proxy, or the MML Interface is by-passed (dummy) and the proxy WS is called directly with SOAP (or RMI etc.). However, this can also be done differently without affecting the remainder of the decision support issues that are examined in this paper. The overhead reduction for the wireless part of the communication is performed in the same way in all cases.



(Wireless) Service Consumers
MML Interface
Authentication Mechanism
MOBILITY MEDIATION LAYER
Push Mechanism
Context Adapter
Context Manager
System Context
User Profiles
Execution Engine
Admin Components
Admin Portal
MML Settings
Monitoring And Logging Data
Performance Adapter (Mediation Core)
Decision Support
Proxy Generator
Proxies
Local Service Registry
Service Manager
Service Descriptions DB
IoS / Marketplace Connectors
The focus of the paper lies on this component. Each proxy implements a different **adaptation mechanism**, while ***decision support*** refers to the proxy generation decisions, i.e., the selection among different proxies / adaptation mechanisms. Related components / aspects, e.g., system context and service descriptions, play an important role.
Service Marketplaces / Other IoS Components
External Web Services (of different providers)
WS
WS
WS
Possible end-to-end device-to-Web service communication path
Component interaction. Types (read/write, request/response etc.) and directions (uni- or bidirectional) are abstracted
Active architectural component / Component with logic (may contain other components, colouring is used for visualization purposes)
Passive / storage component

Figure 1: Architecture of the Mobility Mediation Layer.

In order to know how to best mediate the communication, the MML also includes mechanisms for the monitoring (and logging) of the performed mediated Web service calls. However, it is often difficult to know all the details of a call. For example, information about the calling device must have been reported by itself (which is not always the

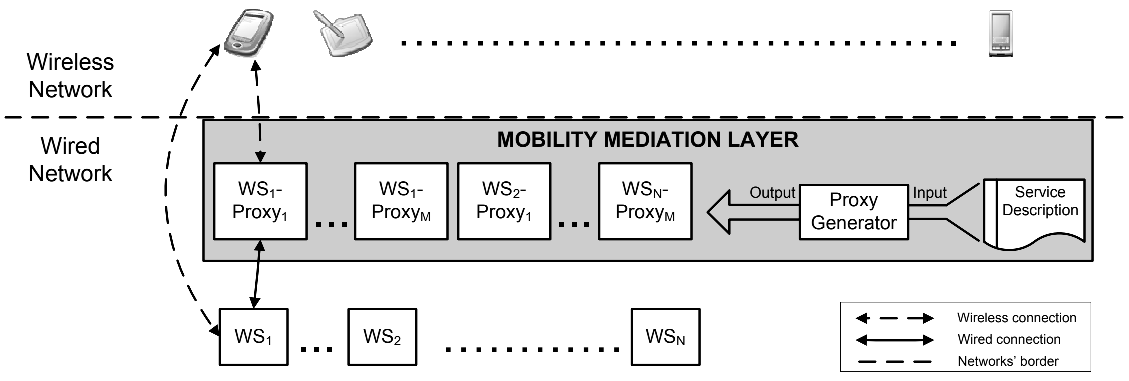


Figure 2: Simplified view of the Mobility Mediation Layer proxying concept.

case), while information about the network, e.g., about packet loss, may not be known for all the used connections. This monitoring difficulty is important for the present work, since the following sections will focus, among other matters, on such missing data.

To be available, proxies need to be generated, as they are not generated on the fly (cf. Section 2.3). The logic for the proxy generation depends on the *MML Settings*, which, in turn, depend on the *System Context*, as depicted in Figure 1. Furthermore, the service descriptions offer valuable input for the proxy generation process.

Further, the MML includes an *Admin Portal*, which can be used to configure/administrate the MML or to manually examine monitoring and logging information stored by the proxies for every Web service call. Finally, the MML accesses service descriptions and other service-related information from the IoS through its *IoS/Marketplace Connectors*, while a *Push Mechanism* is responsible for the reverse communication with the wireless consumers. Some of the MML components can be derived and/or are inspired by abstract components of general-purpose QoS-middleware concepts such as the one presented by Nahrstedt et al. [4].

2.3. Using Proxies for Web Service Adaptation

Figure 2 shows that Web services can be consumed either directly or through different proxies. To implement this, the MML uses a *Web Service Proxy Generator*, a software component which performs automatic code enrichment and deployment actions upon the code that is generated for the target service. The proxy generation does not take place on the fly to serve particular Web service calls, because this would have a negative effect on the performance of a particular call. Even though the proxy generation usually takes only a couple of seconds, this timespan can constitute a large delay for single Web service invocations. Thus, the proxy generation is performed in advance for services whose clients *could* make use of this proxy in the future.

Proxying is an abstract concept that implies the interception of requests and it may be used in many different fields. In our work, a *Web service proxy* is a module of the MML that can intercept the calls to a particular external Web service in order to enforce an independent and well-defined adaptation mechanism. More concretely, as depicted in the simplified view of Figure 2, Web service proxies have the following mission: They try to avoid heavyweight communication taking place over the wireless channel by replacing direct calls (long dashed arrow in Figure 2) of wireless service consumers to Web services with proxied service calls. The latter consist of two parts: A wireless call (short dashed arrow in Figure 2) to the proxy and a wired call (short direct arrow in Figure 2) from the proxy to the Web service. Since a proxy enforces an adaptation mechanism, the wireless call is expected to be more lightweight, i.e., to be completed by exchanging less data, while the wired call normally causes exactly the same data exchange as the original direct call (long dashed arrow in Figure 2). Multiple proxies may exist for the same service and enforce different adaptation mechanisms, i.e., in the context of this work the usage of an access method like SOAP-over-UDP or Wireless SOAP (cf. Section 3). Of course, the adaptation mechanisms presented here are not middleware- or proxy-specific and could lie inside any other similar layer.

It must be noted that the Web service proxies could be implemented manually. However, such a manual implementation requires some development effort for every individual proxy generation. Instead, our work aims at automatic

Table 1: Characteristics of possible adaptation mechanisms. The complete table can be found in [6]. *s* (=small), *m* (=medium), and *h* (=high) are ordered categorical values with $s < m < h$, so that, for example, "$\geq m$" means "*m* or *h*". "–" refers to values that are either unknown or unimportant.

Approach	*Bandwidth*	*Latency*	*Packet loss*	*Stability*	*CPU power*	*Data size/SOAP size*	*SOAP message size*	*Processing time*	*Service call frequency*	*Service call criticality*	*Expected response time Improvement over SOAP/HTTP/TCP*
SOAP-over-UDP [2, 12]	–	≥m	≤s	–	–	–	≤m	≤s	–	≤s	8–10x
Compression [13, 14]	≤s	–	–	–	≥m	–	≥m	–	–	–	1–1.5x
...	...	...	...	...	...	...	...	...	...	...	...

Web service proxy generation that is based on the existing service descriptions. Once implemented, it can be used for as many proxy generations as desired. The automatic proxy generation is based on a novel process that: (a) creates proxy code by parsing the service description, (b) adjusts and enriches the created code according to the desired type of proxy, and (c) transparently deploys the generated proxy, offering it as a new service that runs on the MML but can address the original service. More details about the methods and the algorithms of the proxy generation process have been presented in [11]. For this work, it is sufficient to understand that proxies are accessed by the mobile devices in a very similar way that the original services would be accessed. For example, in most of the cases, the proxy itself is a standard SOAP Web service like the original one, so that the mobile device simply needs to replace a URL in order to access the service.

3. Formulation of the Decision Problem

As mentioned in Section 1, there is no generally applicable best-performing adaptation mechanism that would fit all cases in terms of device capabilities, network connection, or the actual service (system context). Thus, decision support based on a *scoring algorithm* is needed. Such an algorithm determines how adequate an adaptation mechanism (i.e., its corresponding proxy) would be for each known service for a particular system context. Our according research question is *"Which proxy should be generated/activated for each Web service under the current system context?"*.

As a first step towards answering this question, we analyzed possible adaptation mechanisms (access methods) for Web services according to the conditions under which they are expected to achieve their maximum benefit [6]. Table 1 shows an excerpt of the results. For example, as shown at the bottom of the table, a *Compression* proxy is adequate when the bandwidth is *small* (*s*), the CPU power of the device is *medium* (*m*) or *high* (*h*), and the message sizes of the Web service are also *medium* or *high*. The lines of this table already look like rules for the generation of proxies, but they are far from being deterministic for the decision process, since other factors may come into play, such as weighting, different goals or utility functions, user feedback etc.

The decision problem handled in this work is referred to as *Always Best Served* (ABS), in accordance with the well-known and much investigated *Always Best Connected* (ABC) problem [15, 16]. ABS is a problem similar to ABC which appears when moving up in the OSI model [5], from the network layer to the transport and session layers. There, instead of selecting access networks as in the ABC, access methods to Web services have to be selected. In short, while ABC scores available access networks, ABS scores possible proxies [7].

Appearing on a different layer, ABS needs partially different context information [6]. Depending, first, on the granularity with which the context information can be rated and, second, on how deterministically the context pinpoints the appropriate alternatives, certain adaptation mechanisms may be adequate in one particular context, but not in any other. Furthermore, the conditions that make a proxy adequate for use have been researched to a much lesser extent than the conditions that render access networks adequate. Because of the time required for the proxy generation and due to the complex logic that would be otherwise needed on the client-side, the decisions in ABS are normally not taken per device and on-the-fly (or "real-time") for particular tasks, but they rather refer to (and affect) a set of

Table 2: Possible proxies and their characteristics.

Proxy	*Bandwidth* (a_1)	*Latency* (a_2)	*Packet loss* (a_3)	*Stability* (a_4)	*CPU power* (a_5)	*Data size/SOAP size* (a_6)	*SOAP message size* (a_7)	*Processing time* (a_8)	*Service call frequency* (a_9)	*Service call criticality* (a_{10})
p_1	u	≥m	≤s	u	u	u	≤m	≤s	u	≤s
p_2	≤m	u	u	u	u	≤m	≥m	u	u	u
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
p_N	$\in T$	$\in T$	$\in T$	$\in T$	$\in T$	$\in T$	$\in T$	$\in T$	$\in T$	$\in T$

Table 3: Monitored Web service call records and their characteristics.

Record	*Bandwidth* (a_1)	*Latency* (a_2)	*Packet loss* (a_3)	*Stability* (a_4)	*CPU power* (a_5)	*Data size/SOAP size* (a_6)	*SOAP message size* (a_7)	*Processing time* (a_8)	*Service call frequency* (a_9)	*Service call criticality* (a_{10})	*Chosen proxy* (a_{11})
r_1	u	m	s	u	u	u	s	s	s	m	p_4
r_2	u	u	h	u	h	u	m	s	m	h	p_1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
r_K	$\in V$	$\in V$	$\in V$	$\in V$	$\in V$	$\in V$	$\in V$	$\in V$	$\in V$	$\in V$	$\in P$

devices. As a result, it is less meaningful to talk about optimization in the case of ABS. The reason is that in ABC, a device may have all the information that it needs in order to optimize the selection of an access network for a given action. In ABS, a proxy is generated for future usage and for many devices/contexts, the exact characteristics of which are unknown.

The outcome of the ABS problem, i.e., the scoring of the alternative proxies can be used for various purposes. For instance, the scores may be seen as suggestions to system operators or they may be used for the automated triggering of proxy generations. These diverse ways of exploiting the scores are outside of the scope of this work. Instead, the scoring algorithms are assumed to be part of a general-purpose decision support system. The latter is provided with information about the proxy characteristics and about the past Web service usage, and is expected to suggest how suitable each proxy would be for a service.

The problem that is to be formulated and finally solved by a scoring algorithm consists of exact descriptions of the input and the expected output. Related descriptions and definitions are provided in the following.

3.1. Input

With respect to the nature of the data that shall determine the proxy scores, and taking the involved entities and attributes, their possible values, and the missing data issue into account, the following sets and variables are defined:

Table 4: Example output of a scoring algorithm.

	p_1	p_2	$\dots$	p_N
s_1	0.1	0.67	$\dots$	-0.55
s_2	0.33	-0.5	$\dots$	0.8
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
s_L	$\in B$	$\in B$	$\in B$	$\in B$

- P, as the set of possible proxies, with the elements $p_i \in P, i \in [1, N], i \in \mathbb{N}$, where N is the number of possible proxies, each representing a different adaptation mechanism.
- R, as the set of Web service call records (i.e., monitored Web service invocations), with the elements $r_i \in R, i \in [1, K], i \in \mathbb{N}$, where K is the number of records.
- A, as the set of attributes used for characterizing the elements of P and R, with the elements $a_i \in A, i \in [1, 11], i \in \mathbb{N}$. A has 11 elements, corresponding with the context attributes derived from the analysis in [6] (cf. Table 1).
- V, as a set of values v indicating the requirements of a proxy p_i on an attribute a_j in order to be adequate. The values v were chosen with the minimum granularity necessary for the problem regarded within this work [6]. In short, the three discrete categorical values small (s), medium (m), and high (h) have been used for each aspect; u denotes an unknown value. Thus, $v \in V := \{s, m, h, u\}$.
- T, as a set of thresholds, namely $T := \{\leq, \geq, \emptyset\} \times V$, required for denoting thresholds based on these values.
- S, as the set of known external Web services, with the elements $s_i \in S, i \in [1, L], i \in \mathbb{N}$, where L is the number of Web services.

Table 2 and Table 3 visualize the defined sets and variables, providing example instances, as well as indicating the value ranges of the presented attributes. Note that while the attribute values of the elements of R (cf. Table 3) are always elements of V (except for the last column where the chosen proxy is indicated), the values of the corresponding proxy features (cf. Table 2) are expressed with the help of the same values, but they are accompanied by the symbols $\leq$ and $\geq$ (except for the *unknown* value u, which does not feature such a symbol). Attribute a_{11} in Table 3 indicates which proxy has been selected for the recorded service invocation and shall only be relevant when user choices are considered, i.e., when applying a QoE-based scoring. The information contained in these two tables is the input to be given to a scoring algorithm.

3.2. Output

What is needed as output is a set of scores, each score corresponding to a service-proxy pair (s_i, p_j). The range and the meaning of the scores themselves depend on the algorithm that calculates them. Thus, the range of scores is abstractly defined here as $B := [b_{min}, b_{max}]$, with $b_{min}, b_{max} \in \mathbb{R}$. The scores of different algorithms are not directly comparable. However, normally, the higher the score, the more suitable the proxy for the respective service. Table 4 visualizes an example scoring output for $b_{min} = -1$ and $b_{max} = 1$. As already explained, the exact way in which the mediation layer (or its operator) uses these scores is open and outside of the scope of this work.

4. Scoring Algorithms for the ABS

In general, decision problems are defined as problems that can be answered with *yes* or *no*. Decision algorithms are the according algorithms used to solve them. The basic methods and the tools for developing such algorithms can be found in the fields of statistics, machine learning, and operations research [17, 18]. Optimization problems, Bayesian statistics, and decision trees are examples of such basic methods.

We propose two scoring algorithms, namely a QoS- and a QoE-based one. QoS, in the context of our work, refers to a set of technical aspects such as performance, flexibility, scalability, reliability, and more, which can be used for characterizing the overall quality of a system, service, or application. For each of these aspects, case-specific metrics can be defined as it has been done in Section 3.1. In contrast, QoE refers to a set of user-related aspects such as opinion, satisfaction, QoS-perception, and more, which can also be used for characterizing the overall quality of a system, service, or application. Obviously, QoE depends on QoS, but it measures the *effect* that the QoS metrics have on the user. User ratings or user choices as foreseen in Table 3 (column "Chosen proxy (a_{11})") are examples of possible QoE metrics.

In accordance with the definitions of QoS and QoE, two principally different categories of scoring algorithms can be developed for decision support. A QoS-based scoring algorithm would rate each proxy by comparing its characteristics (cf. Table 2) with the monitored service call records (cf. Table 3) in order to see how well the proxy matches the respective invocations. A QoE-based approach would focus on past user decisions, i.e., it would perform the scoring by analyzing the relationships between the values of the parameter a_{11} ("Chosen proxy") and the values of the other attributes.

To make use of QoE metrics, user choices (or user ratings of proxies) need to be recorded. Since we do not focus on the collection of QoE metrics in this paper, we simply assume that the chosen proxies – as depicted in Table 3 – are available and correct. This means that the historical values in the table only comprise these values where a "fitting" proxy has been chosen; wrong choices are omitted from the table. Accordingly, our QoE metric is based on binary values stating either "fitting proxy" or "wrong proxy choice", and only the former ones are used for the decision support presented in Section 4.3.

An approach which mirrors our course of action has been presented by Khirman and Henriksen [19] for the task of Web browsing. In their work, the authors measure user dissatisfaction by evaluating user-side request cancellations from the HTTP server protocol logs. There are several more advanced approaches how to obtain QoE metrics that go beyond our rather simple approach: Especially in multimedia research, the objective measurement of QoE has been a major topic in recent years. For example, Wu et al. [20] present an according framework for Distributed Interactive Multimedia Environments, where user ratings are empirically obtained regarding different dimensions using (naturally qualitative) Likert scales. Depending on the observed QoE metrics, they can either be surveyed technically, e.g., by logging usage data, or by asking the user regarding their usage experiences via a questionnaire. In their survey on different techniques to measure QoE [21], Kuipers et al. state that human perception should in general be measured as a Mean Opinion Score (MOS), which is in fact also a Likert scale-based approach. In contrast, Brooks and Hestnes argue that user assessments may easily lead to wrong results [22]. They propose to measure the outcomes of a user's experiences instead of the user experience itself, thus mirroring the aforementioned approach by Khirman and Henriksen. The authors apply quantitative scales for this purpose, since they are expected to provide a more subjective measurement. Applying this approach in our future work would provide us with finer-grained user ratings and could therefore lead to even better decision support.

Our decision algorithms need to solve the problem as formulated in Section 3 and deliver meaningful results. Their logic is intuitively derived from the characteristics of the ABS problem in terms of used context attributes, involved value ranges, etc. The latter are determined by the granularity of the results of our survey on service adaptation mechanisms [6]. The correspondingly developed scoring algorithms are described in the following, after the core concept behind proxy scoring has been explained with an example.

4.1. Core proxy scoring concept

The suitability of a proxy under certain conditions can be judged, as previously explained, based on the results of the survey presented in our former work [6] (cf. also Table 1), when the latter are compared with actually observed values of a monitored system.

Table 5 explains with trivial examples how the elementary matching degrees (of a certain proxy for a given value of a certain attribute) are calculated. For example, the first line shows that if a proxy is suitable for $\geq m$ values of an attribute and the attribute has been observed to have the value h, then the matching degree is +2. The second and the third line provide further examples. This concept is in accordance with the assumptions that had been made during the survey. Such matching degrees can be used in different ways. For example, the QoS algorithm presented in the next subsection calculates, adds, weights, and combines such partial results in a way that leads to meaningful results for the decision support scenario applied in the work at hand.

Table 5: Example trivial calculations of partial matching degrees of a proxy.

Proxy condition (cf. Table 1)	*Observed value (cf. Table 3)*	*Matching degree*
≥m	h	+2
≤s	m	−1
≤m	m	+1
...	...	...

4.2. *Quality of Service-based Scoring Algorithm*

As is usually the case in QoS-related theory, the proposed QoS-based algorithm uses a utility function for the scoring of the different options. This utility function is based on the idea of calculating distances of "ideal" and "actual" conditions, in order to measure "how far" each proxy's *optimal* setting is from the *actual* technical setting. The decision for this approach was driven by the fact that both the service call records and the proxy characteristics are already in a vector-like form with ordered symbols (s, m, h) as values. This makes their distance-based comparison easy and meaningful. Since this is done for each element of R, the results are then aggregated into a total score of each proxy for a given service. Thus, a utility function $n_{p_i}(r_j) \in \mathbb{R}$ must be formulated.

Let $R_{s_k} \subseteq R$ be the set of service call records of service s_k and N_{s_k,p_i} be the set of scores that result from the application of the utility function for p_i upon R_{s_k}. Thus, N_{s_k,p_i} contains the values of $n_{p_i}(r_j)$ for which $r_j \in R_{s_k}$. Let

$$N^+_{s_k,p_i} := \left\{x | x \in N_{s_k,p_i} \wedge x > 0\right\} \quad \text{and} \quad N^-_{s_k,p_i} := \left\{x | x \in N_{s_k,p_i} \wedge x \leq 0\right\} \tag{1}$$

then $\rho^+ := \frac{|N^+_{s_k,p_i}|}{|N_{s_k,p_i}|} = \frac{|N^+_{s_k,p_i}|}{|R_{s_k}|}$ is the quota of calls that would result in a positive score for this proxy.

Let $\gamma \in [0,1]$, $\gamma \in \mathbb{R}$, be the minimum acceptable threshold for ρ^+, $\beta := max\{0, \rho^+ - \gamma\}$ be the maximum of either the positive distance of ρ^+ to the threshold γ or 0, and $\eta := \frac{\beta}{1-\gamma}$ be the ratio of this distance to the threshold of acceptable values for ρ^+. Then, the scoring function is

$$f(R,S,P,s_k,p_i) := \delta\left(\eta, |N^+_{s_k,p_i}|\right) \times \sum N^+_{s_k,p_i} + \delta\left(1-\eta, |N^-_{s_k,p_i}|\right) \times \sum N^-_{s_k,p_i} \quad \text{with} \quad \delta(a,b) = \begin{cases} 0, & \text{if } b = 0 \\ \frac{a}{b}, & \text{otherwise} \end{cases} \tag{2}$$

Explanation: The above description explains how results of single records are aggregated; γ is only used for customization purposes. When the ratio of positive results is lower than γ, the positive results are ignored and only the negative ones are accumulated. If the ratio of positive results is between γ and 100%, the values of positive and negative results are weighted and summed up as the result. For example, γ could be 0.6 for a minimum of 60% positive results, while $\gamma = 0$ means that no lower limit for the positive results is set.

Next, it must be defined how the utility function $n_{p_i}(r_j) \in \mathbb{R}$ works for single records. For each proxy, there is a description that consists of a set of conditions, i.e., thresholds (cf. Table 2). Let Ψ be the set of attributes $\{a_1, \ldots, a_{10}\}$ and $t_{p_i}(x) \in T$ be the threshold of p_i for the attribute $x \in \Psi$. The set of attributes is not denoted with A here, because a_{11} is missing, so that $\Psi \neq A$. Further, we specify t_1 to be the first element of (the tuple) $t_{p_i}(x)$, i.e., the operator $\leq$, $\geq$, or $\emptyset$. Analogously, t_2 indicates the second element of $t_{p_i}(x)$, i.e., the value s, m, h, or u. For instance, regarding proxy p_1 and attribute a_2 in Table 2, $t_{p_1}(a_2) = (\geq, m)$, $t_1 = \geq$, $t_2 = m$. For ease of use in the function to be defined, the symbols are mapped to integers using the function $z(v)$, which takes values $v \in V \setminus \{u\}$ as input. For the different values $v \in V \setminus \{u\}$, the mapping is specified as follows: $z(s) = 1$, $z(m) = 2$, $z(h) = 3$. Using this mapping makes sure that the defined order (small, medium, high) is maintained. As previously stated, the symbol u is a special case, describing the fact that the attribute value is unknown; hence, a mapping to an integer is not necessary.

For $t_{p_i}(x) \in T$ and $v \in V$, let

$$\varphi(v, t_{p_i}) := \begin{cases} 0, & \text{if } t_2 = u \\ min\left\{\varphi(s, t_{p_i}), \varphi(h, t_{p_i})\right\}, & \text{if } v = u \\ z(v) - z(t_2), & \text{if } t_1 = \geq \\ z(t_2) - z(v), & \text{if } t_1 = \leq \end{cases} \tag{3}$$

Explanation: The function φ takes a tuple of values, i.e., a value $v \in V$ and a threshold t_{p_i}, for a proxy p_i and calculates the distance between the two. This is done by mapping the values to numbers, and then calculating the difference. The difference must be positive if the value matches the threshold, negative otherwise. Positive differences indicate that the condition is satisfied. For example, an h value of an attribute where the condition is $\geq m$ gives a positive difference of $+1$. Although, for instance, a value m for a threshold $\geq m$ is also a *match*, the function φ would give "zero" as a result. For this reason, all individual results of the function φ that have constituted a *match* will be later augmented by 1 by the utility function n_{p_i} (cf. Equation 4 and Table 5). These are all the positive results, but also all the "zero" results that have not been caused by an unknown threshold. If the value v is u, the function φ assumes that u could be any value of $\{s, m, h\}$ and thus calculates the minimum difference, so that no positive proxy suggestions are made "by accident". In case the threshold is u, the difference is zero.

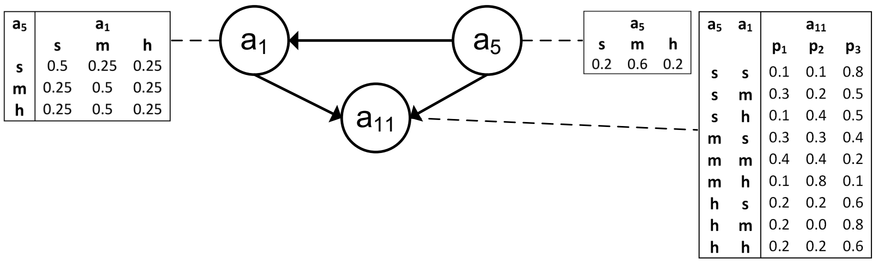
If there is a match for a record r_j for all attributes, then its single score is the sum of the distances of its attribute values to the threshold values. Otherwise, the negative score is calculated as the sum of the value to threshold distances of parameter values not meeting the threshold. Thus, the utility function is defined as

$$n_{p_i}(r_j) := \begin{cases} \frac{1}{\mu(p_i)} \times \sum_{x\in\Psi} 1_{\varphi(r_j(x),t_{p_i}(x))<0} \cdot \varphi(r_j(x), t_{p_i}(x)), & \text{if } \exists_{x\in\Psi}\varphi(r_j(x), t_{p_i}(x)) < 0 \\ 1 + \frac{1}{\mu(p_i)} \times \sum_{x\in\Psi} 1_{\varphi(r_j(x),t_{p_i}(x))>0} \cdot \varphi(r_j(x), t_{p_i}(x)), & \text{otherwise} \end{cases} \tag{4}$$

$$\text{with} \quad 1_f := \begin{cases} 1, & \text{if } f = \mathit{true} \\ 0, & \text{otherwise} \end{cases} \quad \text{and} \quad \mu(p_i) := \sum_{x\in\Psi} 1_{t_{p_i}(x)\neq u}$$

Explanation: If all thresholds are met, the proxy can certainly achieve benefits. This is the only case where the utility function assigns a positive value. Otherwise, it assigns a negative value. Obviously, proxies could offer benefits even when their score is zero or negative. Assigning positive values (as a result of the utility function $n_{p_i}(r_j)$) only to perfect matches is just one feature of the algorithm, which aims at giving positive values only when the benefit is a certainty. Further, the resulting scores of the different possible proxies are probably going to be compared relatively. In all cases, the result is normalized through division by the number of values that are not u.

4.3. *Quality of Experience-based Scoring Algorithm*



a_5	a_1: s	a_1: m	a_1: h
s	0.5	0.25	0.25
m	0.25	0.5	0.25
h	0.25	0.5	0.25

a_5: s	a_5: m	a_5: h
0.2	0.6	0.2

a_5	a_1	a_{11}: p_1	a_{11}: p_2	a_{11}: p_3
s	s	0.1	0.1	0.8
s	m	0.3	0.2	0.5
s	h	0.1	0.4	0.5
m	s	0.3	0.3	0.4
m	m	0.4	0.4	0.2
m	h	0.1	0.8	0.1
h	s	0.2	0.2	0.6
h	m	0.2	0.0	0.8
h	h	0.2	0.2	0.6

(a) Example Conditional Probability Tables of the variables of a (simple) Bayesian Network to be used in *Step 2*.

	a_1	a_5	a_{11}
r_1	s	m	-
r_2	h	h	-
r_3	s	h	-
r_4	s	s	-

(b) Four example records for the examined service to be used in *Step 3*.

Figure 3: Examples for the illustration of the QoE-based scoring algorithm.

Because users have their own subjective selection criteria, a different indicator of the suitability of the proxies is possible. In QoE-based approaches, this indicator is based on user feedback, which can be explicit or implicit [21]. The algorithm presented here uses past user decisions (i.e., user choices of proxies as foreseen in Table 3) as implicit feedback and calculates a "proxy suitability indicator" as its probability to be selected by the users in the future, if all proxies for this service exist. Since the algorithm that calculates this probability should use part of the data in order to "learn" past user behavior and part of the data in order to set evidence about the service that is examined each time, machine learning techniques are an obvious choice. In particular, an algorithm based on a Bayesian Network (BN) has been developed, because BNs match the problem for two main reasons: First, they do not only classify cases (as, e.g., simple decision trees do), but they compute probabilities, as needed for a detailed proxy scoring. Second, they are an appropriate approach for setting evidences about future attribute values, which has to be done for each examined service [17].

The idea is to let the algorithm learn about a given service by examining the past user behavior on "similar" services that had been offered with all proxies. Two services s_1 and s_2 are similar if $a_i(s_1) = a_i(s_2), \forall i \in \{6, 7, 8\}$ (cf. Table 3), because a_6, a_7, and a_8 are the service-related attributes in our work (cf. [6]). The variables that are likely to determine the user selection are included in the BN, together with the variable about the user selection itself (a_{11}). These are the variables that are most probably known to the user, e.g., $\{a_1, a_5, a_9, a_{10}\}$. Summarizing, the following is done in order to assign each proxy a score for a given service:

- *Step 1*: A logical BN structure is manually built, showing which attributes affect the user decision. Manual construction is preferred instead of learning the structure from test data, because the causal relationships between the attributes are more or less straightforward. As Pearl [23] explains, in this case, the BN structure should be created manually by placing the causes before the effects.

  An example for a simple BN structure would be the following (used as the basis for Figure 3): a_{11} is influenced only by a_1 and a_5, while a_1 and a_5 are, again, related.

- *Step 2*: The records of similar services are analyzed in order to find out how users decided before (generation of the *Conditional Probability Tables* of the BN). Figure 3a shows example probability values for the variables used in a simplified version of the suggested BN. Here, for example, $P(a_{11} = p_3 | a_5 = h, a_1 = m) = 0.8$. These tables are learned by analyzing the history of the services that are similar to the examined service.

- *Step 3*: The records of the examined service are analyzed in order to find out how the service is likely to be used in the future (*Evidence* in the BN). The Conditional Probability Tables generated in *Step 2* are used together with the Evidence in order to answer questions of the kind "what is the probability that a user selects the proxy p_x to invoke the service s_k?". With respect to our example, a_{11} is the query variable, while a_1 and a_5 are the Evidence variables. At this step, Evidence (E) is gathered for the Evidence variables by analyzing the monitored service call records of the examined service as shown in Figure 3b (of course, Evidence has to be gathered from a much higher number of records). Then, the evidence state would be: $e = (P(a_1 = s) = 0.75, P(a_1 = m) = 0, P(a_1 = h) = 0.25, P(a_5 = s) = 0.25, P(a_5 = m) = 0.25, P(a_5 = h) = 0.5)$.

- *Step 4*: Once the Conditional Probability Tables and the Evidence have been calculated, the BN is used in order to infer the probability of each proxy to be used for the examined service during the next calls. This probability is the final score of the proxy.

Given the Conditional Probability Tables and the Evidence state in our example as described above, beliefs for the probabilities of the query variable can be inferred based on basic probability theory rules. For example, $P(a_{11} = p_1 | E = e) = P(a_{11} = p_1, E = e)/P(E = e)$. This would be then also the final score for a_{11}.

5. Imputation Algorithms

The monitoring data necessary to apply QoS- and QoE-based decision support is not necessarily always complete; in fact, it is expected to be incomplete in our envisioned scenario (as discussed before). In our previous work [7], we have shown the impact of missing data with respect to:

- *Network-incompleteness*, where knowledge about the network, i.e., bandwidth, latency, packet loss, and stability (a_1-a_4 as defined in Section 3) is missing,

- *Client-incompleteness*, where knowledge about the client devices and applications, i.e., CPU power, Data-size/SOAP size, and SOAP message size (a_5, a_9, a_{10} as defined in Section 3) is missing,

- *Feedback-incompleteness*, where knowledge about user decisions, i.e., the chosen proxy (a_{11} as defined in Section 3) is missing.

In the mentioned paper, we have shown that an absence of 25% of data items can lead to a deviation of the scoring outputs in comparison to the baseline (i.e., decision based on complete information) of about one third (calculated using the Euclidean distance between decision support vectors). This situation can, of course, alter the decision support system's outcome to a critical extent and is therefore not acceptable.

Dealing with incomplete information and uncertainty is mentioned as an important research challenge of self-adaptive systems in both [24] and [25], which are the most recent research roadmaps for self-adaptive systems. Notably, the *data missingness* problem is well-known in other research areas, especially in the data analysis community, which has introduced a number of *imputation algorithms* in order to decrease the negative influence of data missingness. In the following, we will briefly introduce the theoretical foundation for the inclusion of data imputation algorithms in our work based on [26, 27, 28, 29].

Assume a dataset U of data units $U(n)$ and size N ($n \in [1, N], n \in \mathbb{N}$). Each data unit consists of J attributes j, $j \in \mathbb{N}$. Thus, each tuple (n,j) describes one variable, with the value $U(n, j) = w$, $n \in [1, N]$, $j \in [1, J]$, from a predefined set of values $W(j)$ for each attribute. For modeling missing data inside such a dataset, the fact that a value could not be obtained needs to be stored. Hence, for each variable $U(n, j)$, an indicator variable $r(n, j)$ can be added, which is set to 1 if the value is present, 0 otherwise. $r(j)$ denotes the set of $r(n, j)$ for all $n \in [1, N]$ and r denotes the set of $r(j)$ for all $j \in [1, J]$. The missing values of variables of a dataset are also referred to as *missingness*.

The properties of missingness can be specified further by looking at what is called the distribution of missingness, that is, the distribution of the $r(j)$. Let:

- $U_{obs}(j)$ be the variables of attribute j where the values could be obtained (or observed), that is, $r(n, j) = 1$ for all elements of U_{obs}.
- $U_{mis}(j)$ be the missing values, i.e., $r(n, j) = 0$ for all elements of U_{mis}. Hence $U = U_{obs} \cup U_{mis}$, $U_{obs} \cap U_{mis} = \emptyset$.

Different types of data missingness can be differentiated. For this, the relation of U_{obs} and U_{mis} to the distribution of missingness is taken into account. In the following, we will give a brief overview of different classes – for a more detailed description, especially with regard to probability theory, we refer the interested reader to [28]:

- The class *Missing Completely at Random* (MCAR) includes the cases where the distribution of missingness is independent of U_{obs} and U_{mis}. That is, the distribution of the observed and the missing values have no influence on the missingness.
- *Missing at Random* (MAR) is a class of cases where the distribution of missingness depends on observed values, but not on missing values. For a single random variable, that means that its missingness distribution may depend on values of some other random variable, but not on its own.
- *Missing Not at Random* (MNAR) denotes distributions of missingness that are neither MAR nor MCAR. That means that the missingness *might* be associated to any other values, missing or observed.

The ABS problem involves the use of discrete categorical data with a very restricted value range. This fact may affect the importance of particular statistical properties (e.g., mean values, standard deviation) used by imputation algorithms. Furthermore, ABS presents its own types of missingness, which is not expected to be MCAR. In fact, monitored records are rather expected to have dependencies which may be explicitly or implicitly caused by scenario-specific properties such as: "More capable devices are expected to use better connections", "Records from better connections are expected to include less network-related missing data", etc. The possibilities that the monitored data have such characteristics will be analyzed in more detail and reflected in the evaluation test cases.

Different approaches have appeared for handling missing data. Most of them fall under the category of "imputation algorithms", because they substitute missing values with other values, which are expected to be close to the original ones or to affect the further data processing as little as possible. Some of the most commonly used approaches are summarized here; one of the most complete lists of such approaches can be found in [30], along with more detailed descriptions of the underlying mathematical models.

1. *Case Deletion* is an approach according to which data units that miss values are removed from the data set, i.e., for all $n \in [1, N]$: if $\sum_{j \in J} r(n, j) < J$, then $U(n)$ is removed from the data set.

2. *Modus Imputation* is an approach according to which the value that appears most frequently among the observed values of an attribute (*modus*) is used in order to replace all the missing values of that attribute. *Mean Imputation* is a similar approach, which uses the arithmetic mean instead of the modus.
3. The *Random Hot Deck* approach replaces missing values with an observed value that is randomly chosen from the current dataset.
4. *Distance Function Matching* is an approach based on the calculation of distances between data units. The logic is similar to that of Random Hot Deck, but the value is chosen with higher probability from data units that lie spatially or temporally close to the data unit of the missing value. In the deterministic variation of the approach, the value "closest" to the missing one is always used for the replacement.
5. *Multiple Imputation* is considered to be state-of-the-art among the approaches that are based on the *maximum likelihood* concept. With Multiple Imputation, standard statistical methods are applied repeatedly in order to calculate estimates and confidence intervals of the examined variables, so that a certain degree of uncertainty exists during the replacement of missing values and certain statistical characteristics, such deviation, are not lost from the final dataset.

In ABS, scenario-related scoring algorithms should run upon the imputed data. The nature of the algorithms that run upon the handled data is also critical for the efficiency of an imputation algorithm. It must be noted that imputation algorithms may be evaluated either by measuring how correctly they recover missing values or by measuring how well they eliminate the effect of missing data on an algorithm that uses the data after the imputation. It is *not* necessarily true that imputation algorithms which perform better with respect to the first criterion (i.e., perform a more exact guessing), also minimize the error of the *final* result (i.e., after some algorithm runs upon the imputed data). The authors in [30] state accordingly that "it is probably a popular misunderstanding that the goal of imputation is to predict individual missing values".

It can rarely be judged without further experiments which approach best suits a given problem, because the success of imputation depends on the characteristics of the missingness, on the values of the dataset, but also on the nature of the problem and on the algorithms that will run upon the dataset after the imputation, i.e., our scoring algorithms. Hence, the goal of our evaluation (cf. Section 6) is the application of the five presented imputation algorithms in ABS, in order to examine, first, which state-of-the-art solutions should be preferred and, second, if the investigation of new imputation techniques tailored to the ABS scenario is worthwhile.

6. Evaluation

In our previous work [7], first evaluation results for both the QoS- and the QoE-based scoring algorithm have been presented, showing that missing data have a severe impact on them, and thus must be addressed. Therefore, we focus here on the comparison of imputation algorithms and apply these to the QoS-based scoring algorithm. First of all, this is due to the space constraints we have to meet while presenting a thorough and complete evaluation. Furthermore, we consider the QoS-based scoring algorithm to be more representative for handling the ABS problem, because it takes into account all context attributes (and not only those that may affect the user's choice), it is more directly based on the survey results, it is more intuitively derived from them, and because QoS-approaches are in general less subjective and probably more widely used in the domain of networking. The QoS-based scoring algorithm also uses a straightforward logic and only a few steps for the calculation of the scores. Thus, it seems to introduce less "noise" than the QoE-based algorithm (or similar algorithms) with regard to the effect of missing data. For example, as shown in the experimental results of [7], the effect of missing data in the case of the QoE-based algorithm may depend heavily on the number of the available Web service call records, since a minimum amount of records is necessary in order to "learn" the user behavior efficiently.

For the case of handling the ABS problem with the presented QoS-based scoring algorithm, five imputation algorithms are compared in six different test scenarios, in order to investigate which one minimizes the error caused by missing data. We make use of "No Imputation" as a baseline and compare the five imputation algorithms against it. The algorithms that have been examined are those that have been listed as state-of-the-art imputation algorithms in Section 5. These algorithms cover different classes of algorithms and are often used in similar work [30].

6.1. Test Scenarios

The evaluation considers six scenarios for building the test datasets. More concretely, two ways for generating complete test data (without unknown values) and three ways for "inserting" unknown values into them have been used, resulting in six different combinations concerning the way the final datasets are generated. These combinations are called *test scenarios*. The *data* themselves, before the artificial generation of data missingness, have been generated in the following two ways: Firstly, *random*, where all values of the attributes of the Web service call records are generated randomly and secondly, *scenario-based*. There, scenario-related assumptions are used for data generation, such that randomly generated attribute values have an effect on the probabilities of particular values for other attributes.

After that, *missingness* is inserted into the generated complete data sets with one of the three following methods:

- *Random*: Randomly selected values of the complete data set are marked as unknown (u).
- *Unreliable data sources*: The insertion of unknown values is based on an analysis of the sources that commonly monitor or collect the respective data. The attributes are divided into those that can be measured by the mediation layer and those that are potentially only known by the service consumer itself.
- *Unreliable data collection*: The previously described way of inserting missingness is extended here by taking into account not only the sources of the data, but also their collection and transmission.

The exact probabilities and correlations have been chosen in a way that all cases end up having a missingness of ca. 25% in order to obtain comparable results. The exact values are not critical, since it should first be determined whether the different scenarios affect the efficiency of the algorithms at all.

6.2. Metric

The metric used for the comparison of the examined imputation algorithms is the error imposed by the missing values on the result of the scoring algorithm. The way the application of the imputation algorithms reduces this error will be examined. Since there are different ways for defining error metrics, a series of formal descriptions is provided; not only in order to mathematically define the used error metric, but also in order to provide a better understanding of the steps of the conducted measurements. References are also made to the symbols and definitions of Section 3, including the sizes of the sets defined therein.

A dataset $D \in V^{|R| \times |A|}$ is a matrix that contains the values of the attributes of the monitored Web service call records (cf. Table 3), G is the set of the (two) data generation scenarios, M is the set of the (three) missingness scenarios, H is the set of the examined imputation algorithms (as well as the "No Imputation" approach), and I is the set of the performed repetitions of the experiment. Then:

- The set of the *test cases*, each of which is going to be repeated $|I|$ times, is $TC = G \times M \times H$.
- For $i \in I$ and $tc \in TC$, $D_{i,tc} : I \times TC \longmapsto V^{|R| \times |A|}$ is the dataset resulting from the i-th iteration of the test case tc. *For each* such dataset, a reference dataset exists, which is used for the calculation of the error. This reference dataset refers to the respective case without missingness and without imputation, such that it is denoted as $D_{i,b_{tc}} : I \times G \longmapsto V^{|R| \times |A|}$, where b_{tc} simply denotes the reference test case of the test case tc.
- Thus, safely abstracting from any other information used by the scoring algorithm, the scoring function (f) can be defined as a function that maps a dataset to a matrix of scores for the service-proxy pairs (remember that B is the range of scores that can be assigned): $f(D) : V^{|R| \times |A|} \longmapsto (B \subset \mathbb{R})^{|S| \times |P|}$.
- The error (e) of a scoring output is calculated as the difference between this scoring output and the scoring output for the respective reference test case. This error is calculated for all test cases. Thus:

$$e(D_{i,tc}) = |f(D_{i,b_{tc}}) - f(D_{i,tc})|, \quad e(D_{i,tc}) : V^{|R| \times |A|} \times V^{|R| \times |A|} \longmapsto [0, b_{max} - b_{min}]^{|S| \times |P|} \tag{5}$$

- The final metric is the error e, normalized by the observed range of its possible values and calculated as percentage of this range. This metric represents the extent (in percent) to which the missing data affects the scoring output and is the variable that will be plotted in the results of Section 6.3. Thus, the final metric is:

$$e_{norm}(D_{i,tc}) := \frac{e(D_{i,tc})}{b'_{max} - b'_{min}} \times 100\% \tag{6}$$

The average values and the standard deviation of e_{norm} can be calculated and used for the comparison of the imputation algorithms. Different error metrics could have been used, e.g., comparing the changes in the ranking of proxies caused by missing data. However, the scores are supposed to be suggestions. Therefore, the errors of each individual value are equally valued, which leads intuitively to the use of the calculation of differences.

6.3. Results

The main software program for the evaluation has been implemented with the Java programming language, while for the tests of the Multiple Imputation algorithm, the external statistics program R[2] has been integrated. The Multiple Imputation implementation that is used in R is described in [31].

The experiments have been performed with $|R| = 10000$, since this size is big enough for eliminating random errors and can still be processed in reasonable time. The sufficiency of this number can be understood better by examining the results of previous experiments that we have presented in [7]. There, it has been shown that the results are already stable with 5000 entries, and that increasing the number of entries to 10000 did not have an influence on the impact of missing data on the scoring results. The same is true for the number of services that appear in the records (six have been used). Again, in [7], it has been shown that the errors caused because of missing data were similar for all services, i.e., that it does not play a notable role to which service the calls were performed. Further, as explained in detail in [6], a service is characterized by only three of the attributes used by the algorithms, namely the attributes a_6, a_7, and a_8 of Table 2. Thus, there are not so many combinations of "different" services, while the six used services are different to each other in terms of these attributes. The rest of the information needed by the QoS-based scoring algorithm (proxy characteristics) is taken directly from the survey results of [6]. Each test case has been repeated ten times (i.e., $|I| = 10$). This number of repetitions is large enough to achieve sufficiently small confidence intervals for the average errors, as will be seen in the results.

Concerning general system features – but also specific details of the proxy-based adaptation scenario –, it is important to understand the underlying assumptions. A basic assumption is that the mediation layer has limited capacity and/or is concerned with security. Otherwise, it could generate all possible proxies. Each new proxy needs some resources and means the opening of new interfaces/ports on the mediation layer. So, it is obvious that not every possible proxy can be generated for the arbitrary large number of services in the IoS. Furthermore, the fact that the proposed algorithms are tailored to the characteristics of the results of the related survey (i.e., the survey presented in [6]) supports the argument that they provide an educated scoring. This is also one of the reasons why they are not directly compared to alternative scoring algorithms. Therefore, the correctness of the methodology used throughout the survey, but also of the results of the survey, is a requirement for the scoring algorithms to be considered useful. With respect to this survey, it has also been avoided to include device details, which may change from year to year or from a release of a mobile operating system to the next one. Therefore, we focused on always-important, rather static system contexts. However, it may occur that particular clients are incompatible with certain proxies.

The results, i.e., the average values and the corresponding (99%) confidence intervals of e_{norm} for the examined imputation algorithms in the six different test scenarios are presented in Figure 4. The observations that can be made based on these results do not only relate to the efficiency of the examined imputation algorithms, but also to the meaning of the standard deviations and the confidence intervals that appeared in the results, as well as to the meaning of the differences (or similarities) between the results of the different test scenarios.

Three of the imputation algorithms, namely *Case Deletion, Random Hot Deck,* and *Multiple Imputation*, deliver the best overall results, minimizing the error caused by missing data in the scoring output. This is true for all the results and the statement cannot be affected by the deviations that appear, because the named algorithms provide, in *all* cases, an error of under 2.5%, which is comparable only with best-case errors (outliers) of the other approaches, while the average errors of the latter are usually many times higher. Among the best approaches, Multiple Imputation seems to have the smallest average error, but without statistically significant differences (i.e., not always and with overlapping confidence intervals). Keeping in mind that the results are used for decision support, the errors of the three best algorithms are in any case so small that their application should be sufficient for the majority of scenarios that can possibly appear. Since the error cannot get much smaller, it makes little sense to search for new, problem-specific imputation algorithms unless a particular scenario with extreme requirements about the examined error appears. The

[2]`http://www.r-project.org` (Last accessed in July 2012).

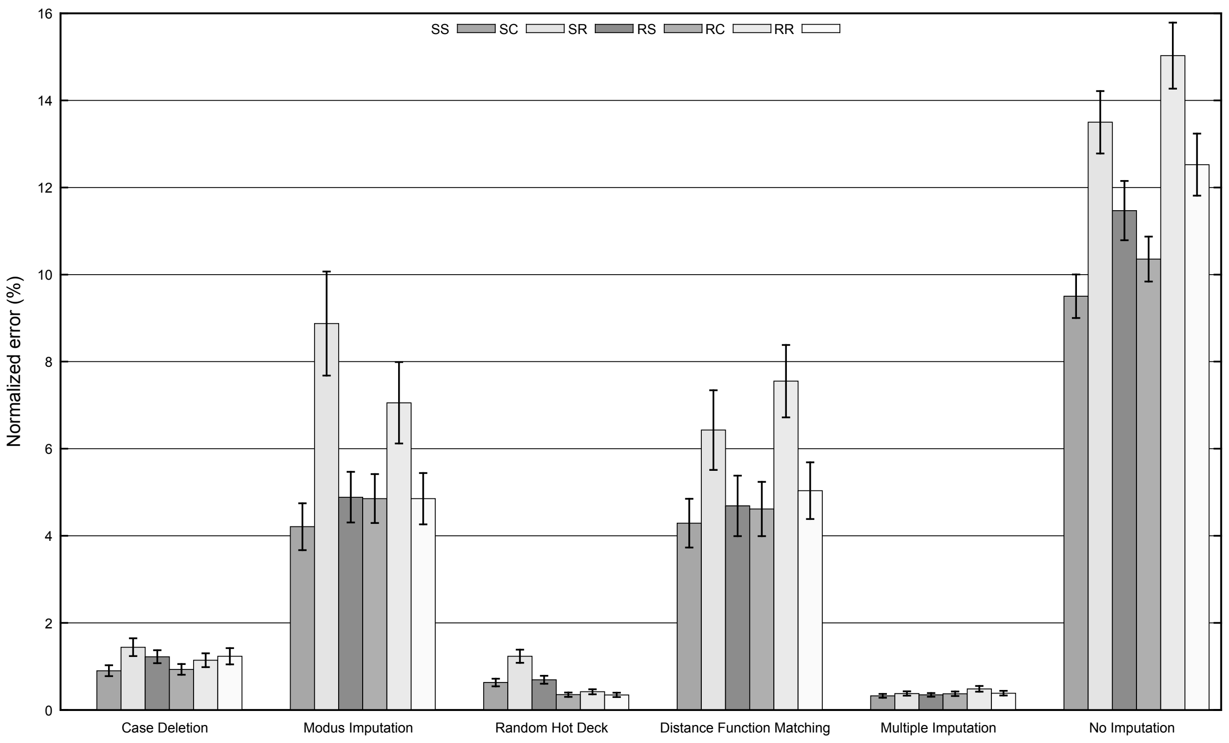


Figure 4: Average normalized errors of the imputation algorithms for each test scenario. SS = scenario-based data, missingness from unreliable data sources; SC = scenario-based data, missingness from unreliable data collection; SR = scenario-based data, random missingness; RS = random data, missingness from unreliable data sources; RC = random data, missingness from unreliable data collection; RR = random data, random missingness.

authors of this paper are not able to come up with such a scenario. Therefore, the choice should normally be between Case Deletion and Random Hot Deck, in order to avoid the (implementation- and time-) complexity of Multiple Imputation. Finally, the complete absence of imputation leads to large errors in the scoring output, usually higher than 10%.

An interesting observation regarding the results per test scenario is that *no significant differences in the efficiency of the imputation algorithms can be observed for the different test scenarios.* Only in the two scenarios with data collection-related missingness (RC, SC) the difference between the well-performing and the bad-performing algorithms seems to become larger. This can probably be explained by the MAR missingness and the fact that the missing data of these test scenarios cause a higher "initial" error (cf. the results for "No Imputation"). However, the well-performing algorithms achieve very similar errors for these test scenarios as for the other test scenarios. This similarity between the results leads to a very important conclusion: The differences in the efficiency of the imputation algorithms lie rather in the nature of the problem (use of discrete categorical values, use of survey results for the proxy characteristics, scoring algorithm logic) than in the distribution of the values and the types of missingness.

As expected, because of the different complexities of the algorithms, *some of the imputation algorithms have been much more time-consuming than the others.* Although many statistical software packages (such as R, which has been used in the experiments) provide practical and relatively efficient implementations of complex imputation algorithms, some of them are very computationally intensive and time-consuming [32]. Indeed, while most of the algorithms completed their work in seconds, Multiple Imputation and Distance Function Matching often needed many minutes for a single iteration. Although time complexity may not be critical, given the very similar results of Multiple Imputation with, e.g., Case Deletion, it is very probable that the first one would be avoided, because such long execution times are often undesirable or disturbing even if the process is performed offline.

7. Related Work

In the following, we will give a brief overview of the related work in the field, namely the MML, approaches from the *ABC* field, scoring algorithms for QoS and QoE optimizations for mobile devices, and handling of missing data.

Middlewares for Web service consumption have been a vivid field of research, especially in the field of QoS support and Web service compositions, e.g., [33, 34]. However, there are only few examples of middlewares that explicitly support the consumption of mobile services. While not being a real middleware but rather a proxy generator, *CRISP* allows caching for Web services consumed by mobile devices, but does not generate the needed proxies automatically [35]. Instead, manual configuration for each individual service is required. *MundoCore* allows a lightweight encoding for remote calls that can be used as an alternative to SOAP [36]. In subsequent work, the same authors present *MundoPPP*, which is a middleware capable of further functionality for mobile service consumption, i.e., data prefetching and caching [37]. To the best of our knowledge, there is no middleware for mobile service consumption that is capable to (automatically) choose from a number of different adaptation mechanisms, as we have presented in this paper.

ABC is a well-known and heavily investigated issue, concerned with letting wireless devices switch among different access networks that they can possibly use (e.g., WLAN, UMTS, GPRS, or Bluetooth) [15, 16]. The goal is, of course, to each time choose the access network which is most appropriate in the current context. The corresponding selection problem is often modeled and handled as a knapsack problem (NP-hard) [38], while QoE-based approaches have also appeared [39]. However, an issue similar to ABC appears if we move up in the OSI model [5], from the network to the transport and session layers. There, adapted Web service access methods have to be examined and selected, as described in the introduction. In accordance with ABC, we have introduced the term "Always Best Served" (ABS). As discussed in Section 3, despite similarities to ABC, ABS appears on a different level (needs partly different context), is less deterministic (conditions that match each of the alternatives have not been researched in such detail), and the technologies that make the issue arise have been immature until quite recently.

ABC is, of course, not the only domain related to networks or computing in which similar decision support or scoring algorithms have been developed. For example, [40] provides a detailed analysis of QoS- and QoE-Management for UMTS cellular networks, where decision algorithms play an important role. [40] is not concerned though with ABS-specific attributes, such as the characteristics of Web services, and it rather presents solutions for problems such as routing or mobile network configuration. Concerning, for example, the structure of the used context and the types of knowledge incompleteness that are likely to appear, ABS obviously bears many differences compared to the problems discussed in [40]. Further interesting scoring algorithms can be found in the domain of event-detection. For example, [41] scores the relevance of events detected by sensors in order to decide if they should be propagated to decision makers or not. However, the scoring of [41] is based on structured score sheets and decision-maker weightings. Not only would such score sheets be impractical in the ABS scenario, but the goals (used bandwidth, user-perceived latency, energy consumption) are also so close to each other that putting weights on them would not change the results dramatically. In our approach, the description of each proxy already indicates which attributes are important, so attribute-weighting is implicitly present. Goal-weighting, as it appears in related work, i.e., weighting of the kind "focus on energy consumption rather than on performance" is irrelevant to us, because our approach is designed specifically for performance, knowing though that performance enhancement often enhances the rest as well.

All in all, decision support or scoring algorithms may be used in any domain and may be based on any mathematical foundation. Since the focus of the work at hand is *not* on providing optimal decisions of any kind, but rather on designing algorithms that match the qualitative characteristics of the problem and on examining and enhancing their behavior against missing data, the examination and comparison of any further scoring approaches is out of scope of this work.

Last but not least, we want to discuss research related to the handling of missing data (cf. Section 5). A common approach is to use scenario-specific correlations of the missing data in order to repair the sources of the errors. For example, [42] presents such an approach for erroneous sensor data sources. Such solutions are, however, not considered here, because it is assumed that data missingness cannot be avoided and that the control over the error sources (user devices, mobile networks) is low or completely absent. Thus, the focus is laid on imputation algorithms in this work. The state-of-the-art imputation algorithms are general-purpose and their usefulness in certain domains depends on the peculiarities of the domain. Hence, various researchers have revisited imputation for particular cases, e.g., for databases [43], sensor data [44, 45], or audio data [46].

8. Summary and Outlook

Researchers have presented a large number of possible Web service adaptation mechanisms in recent years. A number of these approaches aim at the reduction of data communication in order to meet the demands of limited connectivity and mobile devices like smartphones. However, none of these approaches are generally the best-performing one in all possible system contexts. Instead, the usefulness of a Web service adaptation mechanism depends on aspects such as the device capabilities, network connection, or the actual service to be invoked. In order to choose the best-performing adaptation mechanism, decision support is needed. To the best of our knowledge, there is no equivalent decision support system so far. Hence, we introduced this new "Always Best Served" (ABS) problem.

In this paper, we presented the underlying Internet of Services scenario, showed that adaptation mechanisms can be wrapped by proxies and that the according decision support is realized by scoring the usefulness of the different proxies/adaptation mechanisms in a certain system context. Based on this, we formulated the decision problem mathematically and presented two according scoring algorithms, namely a QoS- and a QoE-based one. Since these algorithms are based on historical data, we focused on the challenge of missing data, which arises if it is not possible to monitor the complete system context, which is quite often the case in the Internet of Services.

In order to overcome this problem of data missingness, we proposed and evaluated the usage of five data imputation algorithms to make decision finding resistant to missing context data, i.e., improving the result quality of our proposed scoring algorithms. In this context, we have demonstrated that it is possible to significantly reduce the error rate of the QoS-based decision support algorithm using data imputation.

Case Deletion, Random Hot Deck, and Multiple Imputation have been shown to be the most promising imputation algorithms for maintaining the quality of the proxy scoring results in a scenario with missing context data. Further, the evaluation has shown that striving towards the development of a new, scenario-tailored imputation algorithm is not a worthwhile goal, unless extreme requirements for the accuracy of the scoring output are set. This is because the results of Case Deletion, Random Hot Deck, and Multiple Imputation cannot be enhanced much.

Acknowledgments

This work was funded by means of the German Federal Ministry of Economy and Technology under promotional reference "01MQ09016" ("Green Mobility" project), by the European Union within the SIMPLI-CITY FP7-ICT project (Grant agreement no. 318201), and by the E-Finance Lab e. V., Frankfurt a. M., Germany (`www.efinancelab.de`). The authors take the responsibility for the contents. Many thanks to Liang Han and Jeremias Blendin.

References

[1] M. Adacal, A. Bener, Mobile Web Services: A New Agent-based Framework, IEEE Internet Computing 10 (3) (2006) 58–65.

[2] K. Y. Lai, T. K. A. Phan, Z. Tari, Efficient SOAP Binding for Mobile Web Services, in: IEEE Conference on Local Computer Networks (LCN '05), IEEE, 2005, pp. 218–225.

[3] C. Canali, M. Colajanni, R. Lancellotti, Performance Evolution of Mobile Web-based Services, IEEE Internet Computing 13 (2) (2009) 60–68.

[4] K. Nahrstedt, D. Xu, D. Wichadakul, B. Li, QoS-aware Middleware for Ubiquitous and Heterogeneous Environments, IEEE Communications Magazine 39 (11) (2001) 140–148.

[5] H. Zimmermann, OSI Reference Model – The ISO Model of Architecture for Open Systems Interconnection, IEEE Transactions on Communications 28 (4) (1980) 425–432.

[6] A. Papageorgiou, J. Blendin, A. Miede, J. Eckert, R. Steinmetz, Study and Comparison of Adaptation Mechanisms for Performance Enhancements of Mobile Web Service Consumption, in: IEEE World Congress on Services (SERVICES '10), IEEE, 2010, pp. 667–670.

[7] A. Papageorgiou, A. Miede, D. Schuller, S. Schulte, R. Steinmetz, Always Best Served: On the Behaviour of QoS- and QoE-based Algorithms for Web Service Adaptation, in: Workshop Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PERCOM Workshops 2011), IEEE, 2011, pp. 71–76.

[8] J. Cardoso, K. Voigt, M. Winkler, Service Engineering for the Internet of Services, in: Enterprise Information Systems, Vol. 19, Springer, 2008, pp. 15–27.

[9] D. Oberle, N. Bhatti, S. Brockmans, M. Niemann, C. Janiesch, Countering Service Information Challenges in the Internet of Services, Business and Information Systems Engineering 1 (5) (2009) 370–390.

[10] J. Cardoso, A. Barros, N. May, U. Kylau, Towards a Unified Service Description Language for the Internet of Services: Requirements and First Developments, in: IEEE International Conference on Services Computing (SCC '10), IEEE, 2010, pp. 602–609.

[11] A. Papageorgiou, M. Schatke, S. Schulte, R. Steinmetz, Lightweight Wireless Web Service Communication Through Enhanced Caching Mechanisms, International Journal of Web Services Research 9 (2) (2012) 42–68.

[12] K. A. Phan, Z. Tari, P. Bertok, A Benchmark on SOAP's Transport Protocols Performance for Mobile Applications, in: ACM Symposium on Applied Computing (SAC '06), ACM, 2006, pp. 1139–1144.

[13] M. Tian, T. Voigt, T. Naumowicz, H. Ritter, J. Schiller, Performance Considerations for Mobile Web Services, Computer Communications 27 (2004) 1097–1105.

[14] J. Kangasharju, S. Tarkoma, K. Raatikainen, Comparing SOAP Performance for Various Encodings, Protocols, and Connections, in: Personal Wireless Communications, Vol. 2775 of Lecture Notes in Computer Science, Springer, 2003, pp. 397–406.

[15] E. Gustafsson, A. Jonnson, Always Best Connected, IEEE Wireless Communications 10 (1) (2003) 49–55.

[16] C. Yiping, Y. Yuhang, A New 4G Architecture Providing Multimode Terminals Always Best Connected Services, IEEE Wireless Communications 14 (2) (2007) 36–41.

[17] I. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann Publishers, 2005.

[18] F. Hillier, G. Lieberman, MP: Introduction to Operations Research, McGraw-Hill Companies, 2004.

[19] S. Khirman, P. Henriksen, Relationship between Quality-of-Service and Quality-of-Experience for Public Internet Service, in: 3rd Workshop on Passive and Active Measurement, 2002.

[20] W. Wu, A. Arefin, R. Rivas, K. Nahrstedt, R. M. Sheppard, Z. Yang, Quality of Experience in Distributed Interactive Multimedia Environments: Toward a Theoretical Framework, in: 17th ACM International Conference on Multimedia (MM'09), ACM, 2009, pp. 481–490.

[21] F. Kuipers, R. Kooij, D. D. Vleeschauwer, K. Brunnström, Techniques for Measuring Quality of Experience, in: 8th International Conference on Wired/Wireless Internet Communications (WWIC 2010), Vol. 6074 of Lecture Notes in Computer Science, Springer, 2010, pp. 216–227.

[22] P. Brooks, B. Hestnes, User Measures of Quality of Experience: Why Being Objective and Quantitative Is Important, IEEE Network 24 (2) (2010) 8–13.

[23] J. Pearl, Causality: Models, Reasoning, and Inference, Cambridge University Press, 2000.

[24] M. Salehie, L. Tahvildari, Self-Adaptive Software: Landscape and Research Challenges, ACM Transactions on Autonomous and Adaptive Systems 4 (2009) 14:1–14:42.

[25] B. Cheng, R. de Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, G. D. M. Serugendo, S. Dustdar, A. Finkelstein, C. Gacek, K. Geihs, V. Grassi, G. Karsai, H. Kienle, J. Kramer, M. Litoiu, S. Malek, R. Mirandola, H. Müller, S. Park, M. Shaw, M. Tichy, M. Tivoli, D. Weyns, J. Whittle, Software Engineering for Self-Adaptive Systems: A Research Roadmap, in: Software Engineering for Self-Adaptive Systems, Vol. 5525 of Lecture Notes in Computer Science, Springer, 2009, pp. 1–26.

[26] D. Rubin, Inference and Missing Data, Biometrika 63 (3) (1976) 581–592.

[27] P. Allison, Missing Data, Sage Publications, 2001.

[28] J. Schafer, J. Graham, Missing data: Our view of the State of the Art, Psychological Methods 7 (2) (2002) 147–177.

[29] I. Myrtveit, E. Stensrud, U. Olsson, Analyzing Data Sets with Missing Data: An Empirical Evaluation of Imputation Methods and Likelihood-Based Methods, IEEE Transactions on Software Engineering 27 (11) (2001) 999–1013.

[30] M.-X. Hu, S. Salvucci, A Study of Imputation Algorithms, working Paper 200117, National Center for Education Statistics, U.S. Department of Education (September 2001).

[31] S. Van Buuren, K. Groothuis-Oudshoorn, MICE: Multivariate Imputation by Chained Equations in R, Journal of Statistical Software 45 (3) (2011) 1–67.

[32] H. Y. Chen, H. Xie, Y. Qian, Multiple Imputation for Missing Values through Conditional Semiparametric Odds Ratio Models, Biometrics 67 (3) (2011) 799–809.

[33] R. Berbner, M. Spahn, N. Repp, R. Steinmetz, Heuristics for QoS-aware Web Service Composition, in: IEEE International Conference on Web Services (ICWS '06), IEEE, 2006, pp. 72–82.

[34] A. Michlmayr, F. Rosenberg, P. Leitner, S. Dustdar, End-to-End Support for QoS-Aware Service Selection, Binding, and Mediation in VRESCo, IEEE Transactions on Services Computing 3 (3) (2010) 193–205.

[35] K. Elbashir, R. Deters, Transparent Caching for Nomadic WS Clients, in: IEEE International Conference on Web Services (ICWS '05), IEEE, 2005, pp. 177–184.

[36] E. Aitenbichler, J. Kangasharju, M. Mühlhäuser, MundoCore: A Light-weight Infrastructure for Pervasive Computing, Pervasive and Mobile Computing 3 (2007) 332–361.

[37] D. Schreiber, E. Aitenbichler, A. Göb, M. Mühlhäuser, Reducing User Perceived Latency in Mobile Processes, in: IEEE International Conference on Web Services (ICWS '10), IEEE, 2010, pp. 235–242.

[38] V. Gazis, N. Alonistioti, L. Merakos, Toward a Generic Always best Connected Capability in Integrated WLAN/UMTS Cellular Mobile Networks, IEEE Wireless Communications 12 (3) (2005) 20–29.

[39] K. Demestichas, A. Koutsorodi, E. Adamopoulou, M. Theologou, Modelling User preferences and Configuring Services in B3G Devices, Wireless Networks 14 (5) (2008) 699–713.

[40] D. Soldani, M. Li, R. Cuny, QoS and QoE Management in UMTS Cellular Systems, John Wiley & Sons, UK, 2006.

[41] S. Zöller, A. Reinhardt, S. Schulte, R. Steinmetz, Scoresheet-based Event Relevance Determination for Energy Efficiency in Wireless Sensor Networks, in: IEEE Conference on Local Computer Networks (LCN 2011), IEEE, 2011, pp. 207–210.

[42] A. Wombacher, A-posteriori Detection of Sensor Infrastructure Errors in Correlated Sensor Data and Business Workflows, in: 9th International Conference on Business Process Management (BPM '11), Vol. 6896 of Lecture Notes in Computer Science, Springer, 2011, pp. 329–344.

[43] J. Wu, Q. Song, J. Shen, Missing Nominal Data Imputation Using Association Rule Based on Weighted Voting Method, in: IEEE International Joint Conference on Neural Networks (IJCNN '08), IEEE, 2008, pp. 1157–1162.

[44] Y. Li, L. Parker, A Spatial-Temporal Imputation Technique for Classification with Missing Data in a Wireless Sensor Network, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2008), IEEE, 2008, pp. 3272–3279.

[45] N. Jiang, A Data Imputation Model in Sensor Databases, in: International Conference on High Performance Computing and Communications (HPCC '07), Springer, 2007, pp. 86–96.

[46] X. Rui, Missing Data Imputation Based on Compressive Sensing for Robust Speaker Identification, in: International Conference on Wireless Communications and Signal Processing (WCSP '10), IEEE, 2010, pp. 1–4.