University Halle-Wittenberg
Institute of Computer Science

Proceedings of the PhD Symposium at the 8th IEEE European Conference on Web Services
Wolf Zimmermann (Editor)

Contents

A BPEL Implementation of a Security Filter
Michel Embe Jiague, Richard St-Denis, Régine Laleau, Frédéric Gervais 5

Checking of Liveness Properties in Component-based Systems and Service-oriented Architectures
Mandy Weißbach 9

Building Flexible and Collaborative Distributed Applications Based on Java and Web Service Technology
Marvin Ferber, Thomas Rauber 13

Modeling Grid Services with ASM
Luciano Manelli, Sebastiano Pizzutilo, Alessandro Bianchi 17

Self-Adaptive Management of Web Service Discovery
Atousa Pahlevan, Hausi A. Müller 21

Towards realizing WS cross-layer monitoring and adaptation
Chrysostomos Zeginis, Dimitris Plexousakis 25

Information Integration in Service-Oriented Computing
Mohammed AbuJarour, Felix Naumann 29

Towards Realizing Dynamic QoS-aware Web Service Composition
George Baryannis, Dimitris Plexousakis 33
Preface

The goal of the PhD-Symposium is to provide a forum for PhD students to present and to discuss their work with senior scientists and other PhD students working on related topics. As for the main conference, the topics focus on all aspects of Web Services, Service Oriented Architectures, and related fields. In contrast to the main conference, this work is usually unfinished or has just been started in the PhD projects. The programme committee carefully selected eight contributions. Each submission was reviewed by at least two PC-members. In addition to the precise description of the problem to be solved, preliminary results, and first ideas for solving the main problem, the contributions also include a workplan. All these issues are discussed at the symposium with selected senior scientist and the PhD students.

We are grateful to the conference organizers George Angelos Papadopoulos and Christos Mettouri for their organizational support. We thank the international programme committee consisting of

Antonio Brogi (University of Pisa, Italy)
Birgitta König-Ries (University of Jena, Germany)
Kung-Kiu Lau (University of Manchester, U.K.)
Welf Löwe (Linneaus University, Sweden)
Cesare Pautasso (University of Lugano, Switzerland)

for their work. Finally we would like to thank all authors - no matter whether their contribution has been accepted.

Wolf Zimmermann
PhD Symposium Chair

Halle, November 23, 2010
A BPEL Implementation of a Security Filter

Michel Embe Jiague∗†
Supervisors: Richard St-Denis∗, Régine Lalaeau†, Frédéric Gervais†
∗Département d’informatique
Université de Sherbrooke, Sherbrooke, QC, Canada J1K 2R1
†Université Paris-Est, LACL
IUT Sénart-Fontainebleau, Département Informatique,
Route Hurtault, 77300 Fontainebleau, France
{Michel.Embe.Jiague}@USherbrooke.ca

Abstract—Nowadays, organizations from public and private sectors expose their information systems (IS) to the Web. These distributed IS must enforce strict regulations based on laws or corporate policies. A part of EB3SEC and SELKIS projects is to specify these regulations using a formal notation (ASTD) at the most abstract level and automatically derive an enforcing security framework. This paper describes a doctoral project articulated around an implementation of a policy enforcement manager (PEM) similar to the one proposed in the XACML standard in order to impose security policies from formal specifications of security rules.

Keywords—Access control policy; security rule; policy decision point; formal specification; deployment; SOA; ASTD; BPEL.

I. INTRODUCTION

In most B2B and B2C scenarios, enterprise applications are deployed over the Internet. For security reasons, these distributed applications implement encryption and signature protocols as well as mechanisms like authentication. These techniques ensure that only authorized users have access to specific application data and functionality. However, a user once authenticated could still harm the system by misusing data. So from the beginning of IS development methods, access control methods and frameworks have been developed to make sure that data confidentiality and integrity are always preserved. In our doctoral project, problems related to this research theme are examined in a new perspective:

As part of the EB3SEC and SELKIS1 projects, we formulate the thesis that an efficient PEM solution can be derived from a formal security policy and used to enforce access control into a typical WS-based SOA application.

This research project is motivated by the fact that the new Internet applications face new threats. As previously mentioned, access control has been around for a while. A substantial part of the work done in this regard leads to the wide adoption of the role based access control (RBAC) standard [1] upon which are based most of today’s commercial solutions. However, due to long running interactions in new Internet applications, access control must now span three identifiable levels. The lower level targets the database data and CRUD — create, read, update and delete — operations that can be applied on them. The middle level concerns the actions of the IS. User actions are allowed or prohibited based on his predefined privileges. The upper level, which has received a lot of attention from the research community lately [2], [3], [4], [5], [6], is the process level. As examples, a user cannot execute an action if he has previously executed another one, this is known as separation of duty (SoD), or an action must be executed by a user based on an action the user had previously executed, this is an obligation.

Compared to recent work in the access control area, ours distinguishes itself by new aspects. Our approach, which focuses on both the second and third levels, uses the Algebraic State Transition Diagram (ASTD) [7] notation, a formal and graphical notation, to specify security policy rules for IS. Other notations have been proposed to describe security policies peculiar to IS. In the ORKA2 project, policy designers use both UML and OCL to express an RBAC-like diagram of privileges and constraints on these privileges. Our notation goes a step further since powerful constructs for history of executed actions and bindings to Event-B [8] are available. In a similar but however limited attempt, access right constructs have been recently added to the CaSPiS (Calculus of Services with Pipelines and Sessions) notation [9]. CaSPiS is a calculus to specify WS by explicitly defining sessions and properties like graceful termination [10]. All those frameworks and other related research projects [4], [5] have the same drawback: when they do not support at all constraints like SoD, they use different notations or mechanisms to overcome this limitation. In our project, we use ASTD as a unified, intuitive and powerful notation to express permissions as well as various constraints including ordering constraints, obligations and SoD in particular.

There are numerous challenges to tackle in this work. The first is related to the implementation of the XACML abstract model of the PEM. The second concerns the viability of an implementation of the standard in terms of efficiency, scalability, robustness and real-time performance. The third pertains to the dynamic aspect of security policies defined at the third level.

1SELKIS and EB3SEC stand for SEcure heal.th care networkKS Information Systems and EB3Secured, respectively.

2The ORKA Consortium http://www.organisatorische-kontrolle.de/index-en.htm
The rest of this paper is organized as follows. Section II provides an overview of the formal notation used for specifying security rules and describes the example used throughout the sequel. Section III describes potential solutions to a PEM implementation. Section IV describes a prototype implementation of an IS with a PEM to enforce security policies. Section V concludes this paper with ongoing and future aspects of this work.

II. Preliminaries

The ASTD language is a formal notation developed at the GRIL for the purpose of specifying the behavior of IS. The notation does so by defining the valid traces of the system (i.e., the sequences of acceptable events). An ASTD specification is a list of ASTD structures. When using an ASTD interpreter, the software requires an ASTD specification and produces, each time a transition is fired, a new state according to the dynamic described by the structures.

An ASTD is a hierarchical transition system much like statecharts. At the bottom of the hierarchy are the ASTD automata, in which states can be elementary or ASTD structures. The dynamics is based on the transition relation of ASTD automaton structures. These transitions are labeled with events of the form $e(\bar{x})[\phi]$, where $\bar{x}$ is a list of parameters (which can be empty) and $\phi$ is an optional guard that must hold to enable the transition to fire. In the hierarchy, ASTD structures are composed with operators inspired from CSP and LOTOS. Sequence, Kleene closure, choice, parameterized synchronization, quantified choice, quantified synchronization, guard and call are other types of ASTD structures. The complete definition of the ASTD notation and its formal operational semantics are available in [11].

Since an ASTD specification defines valid traces of a system, the ASTD notation can be used to specify security policies. An ASTD security policy restricts the traces of the IS to allow ones that do not violate organizational access control policies. With the usage of parameters and guards on transitions the valid traces can be defined in a fine grain manner. An even better way of doing so is by using patterns as guidelines to express rules of the policy:

- **permission** which authorizes actions to be executed;
- **prohibition** which forbids actions to be executed;
- **obligation** which forces a user to perform an action sometimes in the future after he has performed a specific action, in other words two distinct actions must be performed by the same user;
- **separation of duty** which expresses the fact that a set of tasks cannot be executed by the same users or roles.

To illustrate the specification of a security policy with the ASTD notation, we use a simple banking example. This case study is a simplification of a real system from our industrial partner with respect to EBSEC project. A client brings a cheque to the bank. The cheque is then deposited by a cashier or a head office into the client’s account. According to the bank rules, before the account balance is updated, the deposit must be validated by a head office. Each time an employee uses the IS, he provides his name and role (cashier or head office) along with the action and its parameters. To ensure data consistency as well as avoid inside misuse of the system, a security policy is defined. The rules of this policy are as follows:

Rule 1 (permission): Roles cashier and head office are allowed to make a deposit.

Rule 2 (prohibition): Only head offices can validate or cancel a cheque deposit.

Rule 3 (obligation): For discretion reasons, a client is registered with a cashier and only this cashier or a head office is allowed to perform deposits on the client’s behalf.

Rule 4 (SoD): When entered by a cashier, a deposit is validated or cancelled by a head office. Otherwise — a head office makes the deposit — it must be validated or cancelled by another head office.

The actions of the system are the following. Action deposit(clientId, chequeId, amount) represents the deposit of a cheque with the id chequeId of value amount to the account with id clientId. Action validate(chequeId) validates the deposit made in an account and thus updates the account balance accordingly. Action cancel(chequeId) cancels the deposit of a cheque and action register(clientId) registers a client with id clientId with a preferred cashier for quality of service insurance. To account for the user identity and its role while expressing the policy, these actions are augmented with two parameters: $(u, r)$ where $u$ is the user name and $r$ its role. In the sequel, sets $U$, $R$ and $CHID$ denote user names, roles and cheque ids respectively. Also, due to space limitation only permission and SoD patterns are illustrated. Figure 1a shows the ASTD pattern for permission. It enables the execution of actions $e_1$ to $e_n$ with parameters $\bar{x}_1$ to $\bar{x}_n$ by users $u_1$ to $u_n$ acting with roles $r_1$ to $r_n$ respectively. A Kleene closure ($\ast$) is used, since an action can be repeatedly

\[ e_1((u_1, r_1); x_1) \rightarrow e_2((u_2, r_2); x_2) \rightarrow \cdots \rightarrow e_n((u_n, r_n); x_n) \]

(a) Pattern

(b) Instance

Figure 1: ASTD pattern for permission

\[ \text{deposit}((u, \text{cashier}); x_1) \rightarrow e_1((u, \text{head office}); x_2) \]

(a) Pattern

(b) Instance

Figure 2: ASTD pattern for SoD

---

3 Groupe de recherche en ingénierie du logiciel http://pages.usherbrooke.ca/gril/
executed. Figure 1b illustrates an instance of this pattern. The roles cashier and head office have permission to execute the action deposit (rule 1). The symbol “\( \neq \)“ denotes a don’t care value for a parameter. For example, the values of the user name, client, cheque and amount parameters for the deposit are not constrained in Figure 1b.

The SoD pattern described in Figure 2a allows execution of action \( e' \) only by user \( u' \) different from user \( u \) (predicate \( u \neq u' \) on transition \( e' \)) which has executed action \( e \). Figure 2b illustrates an instance of the pattern. It states that the actions deposit and validate must be executed by two different users. Furthermore, the constraint on the action validate is strengthened by specifying that the only allowed role is head office (rule 4). In this illustration, the separation is about users but the same reasoning may be applied to obtain SoD on roles.

The ASTD security policy is the combination of all ASTDs defined for each rule. Since each rule must be enforced, the ASTD structure parameterized synchronization is used to glue the pieces together.

III. PLAN OF RESEARCH

In a previous attempt to implement a PEM prototype that uses formal security policies, we described in [12] a two-steps automatic translation procedure of ASTD models into executable Business Process Execution Language (BPEL) processes. The transformation can reasonably deal with security rules like permission and prohibition. Regarding SoD and obligation, the form of the rules requires the use of the ASTD parameterized synchronization, which proved latter to be hard to translate into BPEL specially when used in an enclosing ASTD Kleene closure. Indeed, the Kleene closure should be translated into some sort of repetitive construct of BPEL (while or repeatUntil) and the parameterized synchronization should involve BPEL links that might activate or deactivate activities outside the scope where they are defined. This would violate the cross boundary limitation defined by the BPEL standard.

In this paper, another possible implementation of the PEM is explored. A lightweight and BPEL-contained ASTD interpreter is used as the core of the PEM. The proposed solution however requires ease of data computation from the host environment, which BPEL in its simplest form lacks. Nevertheless, the standard has extension points to allow more complex and dedicated behaviors. Our working environment, Glassfish ESB\(^4\), 2.2 version, extends the assign construct with Javascript code that allows more complex data manipulation. The embedded Javascript code snippets may use ECMA Script for XML (E4X)\(^5\), a part of the Javascript language well-suited for XML manipulation.

The third way we envision to follow is to adopt iASTD, an efficient and OCaml-written interpreter for ASTD [13]. iASTD uses a back-end database to store the ASTD’s state.

In this respect the interpreter, wrapped in a WS, will need to be deployed in the target SOA environment.

IV. THE SECURE IS

A prototype IS has been implemented as a proof of concept. This implementation relies on Java API for XML-Based Web Services (JAX-WS)\(^6\), the Java framework for WS. The four actions (deposit, validate, cancel, register) of the IS update a database containing client and cheque information. This database also contains user information, although in practice this — security related — information may be located elsewhere. The PEM complies with the one specified in the XACML standard from OASIS [14]. It is based on two main components: the policy enforcement point (PEP) and the policy decision point (PDP). Together, they are responsible for intercepting requests from client applications to services and providing authorization control w.r.t. access decisions for these requests based on security policies (see Figure 3).

A. The PEP

In the security framework, the PEP is responsible of enforcing the security policy at the IS’s entry point. In the concrete SOA architecture, the PEP is implemented by a JAX-WS handler. Its role is to intercept message exchanges between IS services and its clients. Based on the PDP response to the subsequent request for authorization, the PEP forwards the initial request to the targeted IS service which might successfully executes the request or not. In both cases, the PDP is updated with the status of this response (done or not). Finally the response is returned to the client that initiates the original request.

B. The PDP

The PDP component of the PEM security architecture takes authorization decision. As shown in Figure 4, the PDP core centralizes the final decision making as it performs a two-steps validation. The first step is to check that the name and role provided by the final end user are to be found associated in the security database with some kind of a privilege table. The second step is to check authorization to execute the original WS request by sending the request canDo\((E, user, role, v1, \ldots, vn)\), where \(E(v1, \ldots, vn)\) is the end-user original request, to the

\(^4\)http://wiki.open-esb.java.net/Wiki.jsp?page=GlassFishESBDocs
\(^5\)http://www.ecma-international.org/publications/standards/Ecma-357.htm
\(^6\)http://jcp.org/en/jsr/detail?id=224
the BPEL process and is intended to lower the overhead library uses intensively E4X. The library is embedded in used and the functions \( \text{init} \) to implement these operations, the Javascript language is respectively. Since BPEL lacks the expressiveness power and \( \text{init} \) define the semantics of ASTD transition firing. In the computes the next state from the current state. The com-

PDP BPEL process. The latter responds to the request for authorization with a Boolean value: \text{true} indicates that, at the enforcement point, the initial request is authorized to be executed by the IS. At this point, another task of the PDP core takes place. If the authorized request is latter executed or not by the IS, the PDP BPEL process has to be updated accordingly. The core does so by sending the message \( \text{done(isDone)} \) to the BPEL process, where \( \text{isDone} \) indicates if the IS has executed the initial request \( \varepsilon \). The overall process is summarized in Figure 4.

The PDP BPEL process, exposed to the PDP core as a WS, is deployed with the XML-formatted ASTD specification. For each authorization request, the process computes the next state from the current state. The compu-
tation of ASTD states is based on inference rules that define the semantics of ASTD transition firing. In the prototype, this computation also relies on the functions \( \text{init} \) and \( \text{isFinal} \), which compute an ASTD’s initial state and a Boolean value that indicates if a state is final, respectively. Since BPEL lacks the expressiveness power to implement these operations, the Javascript language is used and the functions \( \text{init}, \text{isFinal} \) and \( \text{transition} \) are part of jsASTD.js, a Javascript library for ASTD. This library uses intensively E4X. The library is embedded in the BPEL process and is intended to lower the overhead of using external code and passing parameters.

V. CONCLUSION

Our simplified version of the bank IS along with the PEM is deployed on a Unix-based operating system and is still undergoing functionality and load tests. Table I summarizes the basic LOC metric for the prototype. An interesting and ongoing work is the online update of the policy. Indeed, laws and regulations change very often. It might be interesting to equip the process with an entry point that allows updating the specification while leaving the current state unaffected.

ACKNOWLEDGMENT

This research is funded by ANR (France) as part of the SELKIS project (ANR-08-SEGI-018) and supported by NSERC (Canada).

REFERENCES


[7] M. Frappier, F. Gervais, R. Laleau, B. Fraikin, and R. St-
Denis, “Extending Statecharts with Process Algebra Oper-


[12] M. Embe Jiaque, M. Frappier, F. Gervais, R. Laleau, and R. St-Denis, “From ASTD access control policies to WS-


control-xacml-2.0-core-spec-os.pdf

<table>
<thead>
<tr>
<th>Component</th>
<th>Lines of code (LOC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IS</td>
<td>383</td>
</tr>
<tr>
<td>PEM</td>
<td>130</td>
</tr>
<tr>
<td>PDP</td>
<td></td>
</tr>
<tr>
<td>Core + privilege check</td>
<td>218</td>
</tr>
<tr>
<td>jsASTD</td>
<td></td>
</tr>
<tr>
<td>BPEL</td>
<td>2131</td>
</tr>
</tbody>
</table>

Table I: Metric about the PEM prototype
Checking of Liveness Properties in Component-based Systems and Service-oriented Architectures

Mandy Weißbach
Institute of Computer Science
University of Halle
06120 Halle (Saale), Germany
Email: weissbach@informatik.uni-halle.de
Supervisor: Wolf Zimmermann

Abstract—The popularity of component-based systems and service-oriented architectures in research and industry is shown by the extended use and the high interest of new research results in this area. Service-oriented architectures and component-based systems are used to create applications with a broad application area, e.g., to implement a business process of a credit institution. To ensure the right behavior of service-oriented architectures and component-based systems a wide range of methods are available. The subject of this paper is to prove liveness properties to detect occurring deadlocks and livelocks. Literature research has shown that existing methods are shallow or have strong restrictions. The aim of this project is to find an appropriate approach to verify existing service-oriented architectures and component-based systems which can include concurrency and recursive callbacks, on liveness properties, to ensure the correct functionality of the services or components.

Keywords—component; business process; workflow; verification;

I. INTRODUCTION AND MOTIVATION

A service-oriented architecture or a component-based system is a composition of a set of so-called services or components. Every component or service provides a certain functionality which can be reached through defined interfaces.

Often such systems depict complex business processes of a company. These systems are used and required every day to guarantee company processes. In case of a system failure, time and money gets lost.

This is the reason why verification methods for service-oriented architectures and component-based systems are needed. A failure component or a failure composition of components has to be recognized as early as possible.

If a component systems crashes, the kind of error may be discoverable (although it is different from the symptom). However, some failures are hard to discover. E.g., livelocks may lead to running processes even if clients have the impression that everything has been terminated. Therefore, analysis tools supporting developers in static identification of such kind of errors are an important issue. Many works focus on safety conditions such as e.g., protocol conformance, contract checking etc. Other works focus on proving liveness properties such e.g., analysis for deadlocks. Many of these works abstract source code to Petri-Nets. However, they are based on strong assumption as e.g., the termination of every loop and the exclusion of recursion.

In order to avoid undiscovered running processes and to satisfy the preconditions of the works on Petri-Net based approaches towards analysis of liveness conditions, my project wants to provide an approach of termination analysis of component-based systems and service-oriented architectures. The approach must be compositional, i.e., each component is being analyzed individually, each component publishes information about its termination behavior, each component publishes information that can be used by its clients for analyzing their termination respectively (including recursive callbacks and concurrency) and the above information can be used to check compositions for non-terminating properties.

All these analyses should work fully-automatic and should be implemented in a termination analysis tool.

The upcoming paper shows the main idea in section II, followed by the first results in section III. The related work is discussed in section IV. At the end of this paper I give a short summary and an outlook in section V.

II. THE IDEA

The following section presents the problem, states the goals and the systematic approach to gain the stated goals.

A. Identification of Problems

The state of the art termination analysis of service-oriented architectures or component-based systems is inadequate, c.f., section IV.

While modeling a system, termination is always assumed, e.g., with strong fairness. The number of parallel threads or recursion depth is often assumed to be bounded by a constant.

Fig. 1 gives an example of the problem. It is specified in the Business Process Workflow Language (short: BPEL). BPEL can be used for automatic execution of business processes. In particular, it can be used to integrate external services (which may be implemented as Web Services) into the process and the business process itself can be published as a Web Service. To reduce the variety of possible activities we only include the BPEL activities described in Table I. Fig. 1 shows a system with two components. Component C1 calls components C2 out of a parallel process synchronously. To step further, component
C1 waits for an answer. Component C2 reacts on the given input. The given input leads to a livelock, e.g., an infinite loop occurs. Component C1 is still waiting for a reply of component C2. The system ends up in a deadlock.

**B. Goals**

I want to examine in detail, if all possible paths within a component terminate and if a composition of such components can terminate. Components are allowed to communicate synchronously and asynchronously. Also, recursive callbacks are permitted. In addition, possible problems that could appear as a result of the so-called blackbox behavior of component-based systems and service-oriented architectures need to be considered, c.f., section III.

In summary, the following points need to be considered:
1) the determination of areas for improvement by means of examples,
2) the termination analysis with no asynchronous calls and no recursive callbacks,
3) the termination analysis with asynchronous calls and recursive calls and
4) the implementation of an automatic verification tool.

**III. First Results**

Until now, my approach includes the termination analysis of a component with the help of dominator trees, the termination analysis of every existing loop by using traditional approaches and the termination analysis of invocations that influence loop termination. So I entered stage 2: the termination analysis of components with no asynchronous calls and recursive callbacks.

As I mentioned before, in my first deliberations I focused on the Business Process Execution Language.

In this area for safety and deadlock analysis, Petri-Nets are frequently used. They provide a natural abstraction of workflows since they are able to describe parallel behavior and with a variety of model checking tools, it is possible to verify these workflows. Petri-Nets and their analysis tools usually require a strong fairness assumption, e.g., [1]–[3]. I showed that strong fairness implies real behaviors of business processes that are not represent by Petri-Net abstraction unless loop termination is assumed.

With the help of dominator trees every programming language that is designed according to structured programming principles [4] can be examined regarding termination.

The main approach first identifies loops and their nesting structure. This can be done by textbook approaches, see, e.g., [5], and results in a tree, the so-called dominator tree. The nodes in the dominator tree represent the nesting structure of the workflow, some of them are loops or loop bodies. For the construction of dominator trees invoke is considered as an atomic operation (this is analogous to procedure call statements in the standard construction of dominator trees). Region with loops are identified. The regions are analyzed from the leaves to the root in the dominator tree. If a region is a leaf it corresponds to an atomic action which is assumed to be terminating. If regions corresponding to inner nodes in the dominator tree are being analyzed the children already has been analyzed for termination. If this region is not a loop, it terminates. Otherwise loop termination has to be proven which is shown later. The assumption that each atomic action terminates and that invoke is atomic implies that the operation of the service being called has to be proven as terminating.

For a service-oriented architecture without recursion the calling hierarchy of operations can be represented as a directed acyclic graph, i.e., there are operations that do not call other services. Then there are operations that only call operations that do not call other operations etc. The operations are analyzed in a topological order to the dual of the call graph (i.e., the edges are reversed). Every component is analyzed one after the other.

The termination of every existing loop by using traditional approaches is done next. A termination function and size function is introduced. If its value decreases after the execution of the loop body it can be proven with the Hoare-Triple that this loop terminates. For more detail, I refer to [6], [7].

With a regular termination or a size function the termination of loops including an invocation that influences loop termination cannot be taken into account. So, the next step is to focus on invocations that influence loop termination.

---

Table I  
**BPEL activities considered in this paper.**

<table>
<thead>
<tr>
<th>BPEL activity</th>
<th>short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>invoke</td>
<td>It is possible to call methods in partners using the invoke statement, this remote method call can be performed synchronously or asynchronously.</td>
</tr>
<tr>
<td>reply</td>
<td>A callee returns the focus to the caller.</td>
</tr>
<tr>
<td>receive</td>
<td>If this statement exists at a program point an invoked method call can be received.</td>
</tr>
<tr>
<td>while</td>
<td>To repeat the execution of activities as long as the condition is evaluated to true.</td>
</tr>
<tr>
<td>flow</td>
<td>To process a set of activities in parallel traces one has to use this statement.</td>
</tr>
</tbody>
</table>

---

Figure 1. Example: A business process workflow including loops.
Therefore, the size function as known from literature, has to be extended to represent the behavior of the called function to the size of the given and returned arguments that are purely for the termination of the loop. The effects before the invocation of function \( f \) and the effects after the invocation of \( f \) in the loop body need to be considered. For simplicity, the called function has one argument and returns one argument. To prove termination, 4 new functions are introduced. The function \( sz_{p} \) represents the effect to the size of the argument of \( f \) in the loop before invoke is called. The size function \( sz_{f} \) represents the effect to the size of the returned argument of \( f \), caused by the invocation. \( sz_{o} \) and \( sz_{p} \) represent the effect to the size of the argument and the returned argument of \( f \) after the invocation.

All these function are composed to size change functions. It can be shown if the arguments of these size change functions are represented through non-decreasing function, then the loop terminates. In order to ensure compositional-ity, the size change function of \( f \) should be provided by the interface description of the component offering the operation \( f \). This compositionality demonstrates the challenge: Traditional termination analysis approaches assume the complete availability of the source code. Thus, size change functions can be constructed according to context of the calls of \( f \). Different contexts may lead to different size change functions. In contrast, in compositional termination analysis, size change functions published by component interfaces have to be used for termination analysis, and this might differ from the context-dependent size change function for \( f \) needed by termination analysis. One of the challenges of the project is to bridge this gap.

The provided first results towards termination analysis of loops in business processes thereby ensure the preconditions required by many Petri-Net verification tools. As a result, a Petri-Net as shown in Fig. 3 can be given (using compiler methods) to allow the verification of further properties.

**IV. RELATED WORK**

This project has the termination analysis in focus. Here, also a wide variety of existing knowledge is available. Program verification and termination analysis has its origins way back in 1967. Floyd published his work [8] concerning partial and total correctness. Hoare introduced in 1969 his rules, by means of which the partial correctness in terms of loop invariant can be described [7]. In 1976, Dijkstra published [9]. His work is generally considered as the fundamental work for developing proofs and programs. He is also known as the one who introduces the so called loop variant. Gries introduced the bound function [10]. [11] introduced the size change principle. In this area, related work goes back to [12]. It is an extension of Floyd’s proposed termination observation based on well-ordered sets. The size change principle where a size change graph and a size change tree is needed to compute the information for termination analysis, allows the consideration of recursive and mutual functions. Based on this work methods of refining exist ([13]).

Many research work on representing business process workflows as a Petri-Net for further verification exist. Early work, showing the termination of a business process workflow goes back to [14]. Other research work concerning termination of business process workflows using the Petri-Net representation is done by van der Aalst et al. In [1] their approach uses Petri-Net properties to show soundness. With the assumption of strong fairness they state termination with the remark one should not introduce infinite loops. In [15] special stop places are introduced to have a single process always terminate. The stop places force the termination. On [16] available verification tools
for Petri-Nets are listed.

With these approaches deadlocks in context with lacking resources can be found. Deadlock resulting from infinite loops were not taken into account.

[17] presents a top-down termination approach. A software framework, called BIP, provides support for developing deadlock-free software for embedded systems. They state deadlock freedom if every component of the software is deadlock-free.

To our knowledge, the proof of termination in combination with the occurrence of infinite loops, concurrency and recursive callbacks is not illuminated by other researchers.

V. SUMMARY AND OUTLOOK

In this work I proposed a new approach for termination analysis of component-based systems and service-oriented architectures.

The approach is based on existing methods, languages and notations to allow the use of other existing verification tools.

First results are available on the termination behavior of defined business process workflows. Possible deadlock or livelock occurrences can be predicted.

Currently I am implementing a completely automatic verification tool (e.g., for business process workflows and among others) by applying tools implemented for traditional programming languages [18]–[20].

Thereby, I will try to integrate synchronous and recursive service calls without any restrictions. As mentioned, by using compiler methods I try to combine termination analysis with transformation of the existing business process workflow into a Petri-Net representation to allow further verification analysis. On top of that a transformation to a Process Rewrite System will be applied to verify protocol conformance checking [21].

I aim to test the new automatic verification tool with our cooperation partner to test the practical suitability.

REFERENCES


Abstract—Modern applications need to scale on parallel computing resources, since single processor speed will not grow as fast as in the past. Additionally, the collaboration between different applications or application modules over networks like the internet is desirable in order to generate more complex applications, in scientific as well as in the business domain. This dissertation will investigate how SOAP Web service technology can help to address both issues, decomposition and collaboration of applications. Therefore, the Web service technology will be enriched with suitable extensions in order to facilitate the transformation of an application into a distributed application that uses SOAP Web services as communication middleware. Furthermore, mechanisms will be developed to combine distributed object processing and service orchestrations, e.g., in BPEL, in one application. As ordinary Web service development normally does not address the issue how an existing application can be transformed into a distributed Web service application, this dissertation will also deal with the question how a debug mechanism can be applied to SOAP Web services and how such a debug mechanism can help to transform an existing single machine Java application into a distributed application that uses SOAP Web services as communication middleware.

I. BACKGROUND

The ability of programs to be executed on distributed machines can be useful in a variety of cases, including high performance as well as ubiquitous computing. However, it is normally hard to transform an existing software into a distributed version. Although, a lot of use cases exist in this context.

The extraction of compute intensive parts can be useful to enable a software to be delivered as a smartphone application while the compute intensive parts are moved to a powerful server. Another use case for the externalization of a software component is the introduction of licensing. Innovative program modules may be offered as a remote software service and sold over the internet in order to use the functionality in different client programs. Nonetheless, the reuse of externalized software modules can also be useful in business processes, e.g., Business Process Execution Language (BPEL), to avoid the reimplementation and to facilitate the reuse of business logic that is already available.

In order to build components that can be exposed as a Web service, a correctness preserving source code transformation is desirable. Tool support has been developed in form of Integrated Development Environments (IDE) like Netbeans or Eclipse. Source code transformations include class extraction or to move methods between classes and many more. Therefore, the software under investigation is normally mapped into an abstract software representation. Special tools like the TRANSFORM toolset can apply more complex transformations, e.g., the externalization of a whole class from an existing Java software design using RMI [1].

When using interpreted code like the Java Bytecode, an extracted software module can be executed remotely on a variety of machines, because the Java Virtual Machine (JVM) provides a homogeneous execution environment that is independent from the underlying hardware and operating system. The JVM also facilitates the sharing of data structures between different applications, because even complex structures can be transferred easily through serialization.

SOAP Web services are very suitable to serve as basis for collaborative distributed applications. The XML format is independent from programming languages and can therefore be used to glue together different technologies, e.g., Java components and BPEL processes. In contrast to other distributed object technologies like RMI or CORBA, which transfers binary data, XML data can be transferred directly in SOAP messages and does not have to be converted. XML data is used in a variety of applications from the business domain (e.g., office documents like docx) as well from the scientific domain (e.g., bioinformatics data like RNAML), which shows the relevance of XML in real world applications. The Web service technology itself provides a lot of extensions concerning a variety of application specific issues that can already be used and do not have to be implemented separately. Such extensions address resource discovery, security, quality of service and many more.
II. THE PROBLEM

Although, a lot of research has already been done in the field of distributed object processing and Web service architecture, there are still some problems when a Web service application should reuse existing software modules or when an existing software should be enabled to benefit form Web service technology.

When a component of an application is extracted and reconnected via Web service technology, the whole application is transformed into a distributed application. This makes it necessary to inspect data coherency and the temporal alignment of activities in the transformed application. A static source code analysis might only find class or call dependencies in an object-oriented software design. Normally, this is not enough to provide a decision if specific program components can be externalized without affecting the behavior of the program, especially with complex software designs. Besides a statical source code analysis, a runtime analysis can be used to provide appropriate information about a software’s behavior before and after the transformation. Unfortunately, a runtime analysis is mostly not possible for a distributed program, especially in a Web service environment.

In contrast to new distributed Web service applications that are mostly derived from a checkable high level software model, the correct behavior of a software that has been transformed into a distributed Web service application by object extraction can hardly be verified. This is due to the fact that the protocol of the extracted Web service component may not be fully specified according to the complexity of the component. Furthermore, the absence of sophisticated analysis and debugging mechanisms for distributed Web service applications make it difficult to perform a correctness preserving component extraction.

The Web service middleware standard does not address the issue how a class interface can be exposed as it can be done in CORBA or RMI. An object-oriented WSDL interface would be the basis of a collaboration between a service orchestration and a distributed object design under the roof of Web service technology. However, such a collaboration is hard to achieve due to the different concepts of the technologies. In an object-oriented software design, objects contain an internal state, whereas services orchestrations are normally composed of stateless Web services. The Web service Resource Framework (WSRF) or BPEL correlation sets do not provide enough functionality to expose a class interface as Web service, because standardized mechanisms for object creation or for complex method parameters (class parameters) are missing. The insufficient support for the extraction and provision of existing software components as Web services makes it hard to develop complex applications based on Web services.

III. RESEARCH CHALLENGES

Despite the problems described above that can appear when an application component should be exposed as a Web service, it can be observed that existing complex programs can be transformed into a distributed version manually without too much effort.

The goal of this dissertation is the development of methods and tools to enable the transformation of existing Java programs into distributed programs by externalizing classes or groups of classes as components to execute them remotely. One aspect of the externalization is to achieve a universal modeling in order to use the externalized software components in a distributed object application as well as in a BPEL service orchestration. Therefore, the definition of an object-oriented Web service interface will be a challenging task.

Another objective is to develop an execution environment based on Web service technology to execute the transformed application. To facilitate the transformation, the environment will be enriched with a remote debugging and monitoring mechanism to detect undesired behavior of the transformed application. One research challenge will be to define and implement a suitable debug mechanism for Web services as well as a frontend for monitoring such debuggable Web service-based applications.

A final goal is the provision of methods and tools that support a software developer to transform a Java software into a distributed version that can be executed on the mentioned execution environment. In this context, the research challenge includes the provision of a suitable externalization strategy along with a collection of source code patterns that can be externalized.

IV. METHODOLOGY

The different research challenges will be analyzed, solutions will be developed and prototype implementations will be created. The methods and tools developed will be applied to suitable real-world applications in order to ensure the practical applicability of the approach.

A. An object-oriented Web service interface

To define an object-oriented Web service interface, general object attributes and the behavior of distributed objects must be summarized. Afterwards, existing Web service standards and implementations are investigated regarding their ability to handle an object-oriented WSDL interface. To achieve compatibility with existing technologies, different BPEL engines have to be evaluated regarding their ability to handle the special object-oriented WSDL modeling approach.

Additionally, method parameters must be taken into account to find a suitable mapping between XML types
and Java types. In this context, it must be distinguished between simple types (call by value) and complex types like classes (call by reference).

To integrate Web services and BPEL Web service orchestrations into distributed object applications, suitable wrapper classes that present a normal object interface to an object-oriented Java application must be defined. Such wrapper classes also need to handle possible asynchronous messaging.

B. Externalization

The externalization strategy should support the software developer in externalizing components correctly. Following recent proposals [1] [2], it will be investigated how a pattern search in an abstract source code representation can help to find attributes of the software design that inhibit the externalization of a special component. As a basis for the software transformation and analysis tools, the abstract software representation of the Eclipse Java Development Tools (JDT) and the Flexible Software Representation (FSR) of TRANSFORMR are used.

Because behavioral obstructions are hard to determine by a static source code analysis, an additional runtime analysis will be developed that is based on the debug information provided by the Web service environment that will be developed.

It is planned to compare the runtime behavior of the original application to the behavior of the transformed application. Therefore, it will be investigated how Linear Temporal Logic (LTL) specifications like in the Finite LTL runtime Monitor (FiLM) [3] can be utilized to define and check object access patterns. Protocol conformance checking might also be useful to verify a correct extraction of a component as Web service [4].

Starting from a simple example (only primitive method parameters and no call dependencies from the externalized class to other classes) the externalization strategy will be extended gradually in order to support real applications at the end.

The scope of the externalization strategy covers only the detection of obstructions in a statical source code analysis and the provision of information regarding runtime critical source code parts. The software developer can use the provided information to apply suitable refactorings to the transformed application in order to patch the software. Furthermore, the provided information can be used to request specific debug information from the execution environment in order to detect runtime errors.

C. Debuggable Web services

The debugability will be strongly focused on the underlying Java Virtual Machine. As debugging of distributed application is not an easy task, runtime violation checking as in [5] and runtime monitoring will also be taken into account. Different execution environments will be investigated regarding their capability to serve as basis for the implementation of such a monitoring mechanism. Nonetheless, it should also be possible to apply the debug functionality to BPEL processes. Therefore, a suitable debug protocol and a mechanism for providing a debuggable version of a Web service must be developed. Finally, a prototypical implementation of the execution environment will be made.

Based on the developed externalization strategy and the preferred Web service middleware implementation, tools for externalizing a component from an existing Java application onto the execution environment will be developed and implemented.

V. Discussion

At first sight, communication middlewares like CORBA or RMI seem more dedicated to the field of distributed object processing than SOAP Web services. However, all those technologies are comparable regarding their functionality. The most significant point of the decision for Web service technology is that collaboration between applications is best supported with Web service technology. Web services use XML messages that can be processed in nearly all programming languages using similar APIs (SAX and DOM). The drawback of the Web service technology is mostly the poor performance regarding bandwidth and latency, due to the SOAP stack. Because the focus of this work is on flexibility and collaboration, this is not a significant limitation.

As this work aims at the development of a general approach for providing objects via Web service technology, the restriction to Java applications seem to be unnecessary. The restriction to Java is made to cut the scope of this work. However, many technologies in Web service contexts are built on top of the Java Virtual Machine, so the relevance of the approach will not be limited significantly. On the contrary, many non-Java related Web service technologies are not open source and this work is based on open source technologies in order to make the approach usable to the community.

Goals of the proposed work are, among others, the provision of a universal approach for Web service debugging and the definition of an object-oriented Web service interface. On the other hand side, most of the work will be limited to Java and BPEL as example technologies. However, the modeling of new Web service extensions and definitions is strongly focused on the usage of existing Web service standards. The BPEL language is chosen as an example to investigate the applicability of the object-oriented WSDL interface, because BPEL is widely accepted as language for Web service orchestrations and many open source implementations exist. Incompatibilities to other Web service related technologies
Investigation of Web Service standards and implementations in order to define an object interface as WSDL document.

Evaluation of BPEL engines regarding the capability to handle the object-oriented WSDL interface.

Evaluation of tools for source code and runtime analysis of Java programs. Definition of necessary debug information for externalization. Implementation for source code analysis tools as preparation for the transformation.


Implementation of transformation tools for externalizing a Java component to a remote object provider. (debuggable Web Service) Applying the tools to some test applications to ensure the applicability. Measure software and performance metrics.

Writing the dissertation document.

might occur, but it is beyond the scope of this work to cover all Web service standards or technologies.

VI. Preliminary Results

As a first step it has already been shown that SOAP Web services can be used to express not only stateless service behavior but also stateful (remote) object behavior within the WSDL specification. Therefore, a special dialect of the WSDL has been introduced as Remote Object over Web service (ROWS) [6].

As an example, the open source raytracing software Sunflow has been transformed successfully into a distributed version that uses Web service technology (ROWS) as communication middleware. The transformation was performed by using available transformation tools and techniques. The performance improvements that were achieved with the transformed application are promising.

Another promising preliminary result is the successful provision of a BPEL process as program object similar to an ordinary class through a special modeling of the BPEL process (BPELRO) and wrapper classes [7]. Hence, it has been achieved to use BPEL orchestrations in object-oriented applications. BPELRO compatible Web service orchestrations and ROWS compatible Web services can be substituted with each other, because they share the same interface definition. This facilitates the utilization of distributed objects and service-oriented concepts within one application.

VII. Time Schedule

Generally, the time schedule is divided into five phase that are illustrated in Figure 1. It can also be seen that the first major phase has already been completed.

References


Abstract—Web-Service technologies represent a standard for electronic communication in information technology systems. The evolution of Grid computing, based on specification of OGSA (Open Grid Services Architecture) service oriented environment standard carries on the convergence with Web services. Under this view, it is an important challenge to study Grids as a set of services in an heterogeneous and distributed system. This paper introduces an abstract model for defining the job management and execution services of a Grid architecture. This type of Grid service is needed by most Grid applications: the goal is to improve control flow descriptions and distributed execution of applications and to provide abstractions useful for all application classes. We propose as formal model an Abstract State Machine (ASM) to analyse and design of OGSA capability. Starting from a ground model of job management and execution service we have built an ASM model for the Grid service. In this work the CoreASM has been used, it is a graphical editor for design and validation of ASM.

Index Terms—Grid programming models; OGSA; Abstract State Machine; Web Services; Distributed systems; CoreASM.

I. INTRODUCTION

The Grid system architecture is defined by I.Foster and C.Kesselman [1] as a wide-scale distributed computing infrastructure to support large computing resources sharing and strict cooperation to solve problems in a dynamic multi-institutional Virtual Organizations. There are three main properties of Grid [2], that characterize the goal of the Grid: 1) a large-scale coordinated management of resources belonging to different administrative domains, 2) standard, open, multi-purpose protocols and 3) good performance parameters. Because of the wide range of heterogeneous resources and their uses, Grid systems are gaining significance as alternative to the traditional supercomputing environment. Supported by the emerging of Web Services, The Globus Project (now Globus Alliance[6]) proposed in 2002 the Open Grid Service Architecture (OGSA) [3], that extends and complements the first definition of Grid by defining the architecture in terms of Grid services. Due to later refinements in 2006 [4], OGSA rapidly became the de facto standard for Grid based systems. OGSA presents a Grid as a Service Oriented Architecture, representing in this way a successful integration of Grid technologies with Web Service Architectures. In fact, Grid uses standard service interfaces (encapsulating heterogeneous distributed software and hardware resources), such as Web services or Web Services Resource Framework services[19]. Many entities, such as the OGF[5], the Globus Alliance[6], the OASIS[7], the W3C[8] and the IETF[9], are involved in the process of standardization of a Grid system.

The OGSA standard describes requirements (such as interoperability and resource sharing, optimisation, quality of service, job execution, data services, security, scalability and extensibility [4]), and considers six important independent capabilities needed to support Grid systems and applications[4]: Execution Management Services; Data Services; Resource Management Services; Security Services; Self-Management Services; and Information Services. In particular the Execution Management Services (EMS) address the job management and execution capability of a Grid system and it is concerned with the research of candidate locations for execution, the preparation for execution, the initiating and managing the execution of jobs until the end. These requirements are also partially fulfilled by OGSA specifications in the Basic Execution Service [21]. In particular, every Grid middleware offers a job management and execution capability, as it enables users to use distributed resources for computationally intensive applications. In fact, EMS are also implemented in the Globus Toolkit [6] and in gLite [24], both used in several Grid deployments. Unfortunately, a uniform access to resources is not available across these two different middlewares. That means, for example, that jobs originated on Globus Toolkit cannot be forwarded to gLite, even if they have access authorizations to resources. So, the use of Grid formal models can help the high-level middleware design with the reduction of the risk that a change in the dynamics specifications could have a large impact on the specification of other aspects. Furthermore, the Grid model-based approaches allow the specification of dynamic aspects in a more intuitive way, without the necessity of having programming skills.

In this work we have proposed an approach to model with an Abstract State Machines (ASM) [10] some aspects of Execution Management Services (EMS). In general the programming models influence the entire software life cycle from design and implementation to debugging and maintenance, and they have the ambition to improve the use of development tools, the composition of services, the management of resources, the reuse for different distributed architectures and finally the standardization of interfaces.

Currently, graphical modeling tools are used to support the
OGSA-EMS description is given in order to analyse the simulation, validation and verification of systems. Eclipse, that performs graphical model for the design, analysis, that fully satisfy the users purposes. In this work CoreASM allow developing algorithms, for the execution management a service model, it is important to find automatic systems that supports and implements OGSA EMS. Considering that gives an abstract representation of the implementation tool job management and execution capability. This model also in particular the standard mechanisms defined in OGSA for single capabilities independently.

In the recent years many efforts have focused on the analysis of Grid models to develop tools used to implement distributed applications. Researchers have also focused their work on Grid concepts to develop also visual tools supporting Grid modeling. There are some related works that use ASMs to describe and to model Grid and distributed systems under different points of view.

In [15] the authors have studied and modelled the abstract communication in distributed systems based on message exchanges. They have used asmL tool for simulation. In [16] the authors have proposed an hybrid approach to the specification, analysis and testing of Grid middleware. They have modelled the Grid system with ASM and have proposed an environment based on asmL tool. In [13,14], the authors have formalized and have described differences between traditional distributed systems and Grid systems. An abstract model of Grids is developed and compared with a traditional distributed systems one. This model analyses every aspect of a Grid from the resource selection to the access control mechanism, and develops a set of abstract rules. Its focus is on the abstraction of resources and users belong to a VO and on their mapping. In [12], the authors have studied Grids, starting, differently from the previous work, from similarities between distributed systems and Grids. They have developed the model according to faults and unreliability of resources. They have defined the Grid entities and the behaviour of the system, and have presented two programming systems using the proposed structure. In [18] the author have focused their attention on a virtual organisation model, in which they have considered performance parameters for the user request. They have defined a pool of resources and VO services that are selected according to the user quality constraints. We have proposed a model that has a different point of view. We have studied and have modelled Grids addressing their OGSA-service structure. It is not necessary to create a global framework, but it is possible to analyse the single capabilities independently.

The aim of this research is to build an ASM to model in particular the standard mechanisms defined in OGSA for job management and execution capability. This model also gives an abstract representation of the implementation tool that supports and implements OGSA EMS. Considering that the evolution of the Grid models has brought to the full use of a service model, it is important to find automatic systems that allow developing algorithms, for the execution management that fully satisfy the users purposes. In this work CoreASM [20] has been used, a free tool environment developed in Eclipse, that performs graphical model for the design, analysis, simulation, validation and verification of systems.

The proposed research is organized as follows: in the next section OGSA-EMS description is given in order to analyse the model and the ASM-based method is described. The definition of Asynchronous Multi-Agent ASM is discussed in order to better justify its suitability in describing the EMS-OGSA. In section three an ASM-based model is carried out for the EMS-OGSA. Conclusions and expected results are reported in the last section.

II. BACKGROUND

A. Grid job management and execution service

OGSA standard has proposed a guideline composed by six services for addressing the main capabilities of a Grid system. Every service has a precise scope, and all together (or some of them) interact to organize a Grid system. This approach allows a simplification of the architecture, in fact every single piece of the complex system can be analysed independently. These services are: Data Service (for common access facilities, efficient transport, and replication services); Resource Management Service (for discovery, monitoring, and control resources); Information Service (for registry, notification, and logging); Security Service (for cross-organizational users, trust, and authorized access); Self-Management Service (for self-configuration, and self-optimisation). For job management and execution service, the currently standard is based on the Execution Management Service (OGSAEMS)[4]. This is the service dealing with the problems of job submission and management over a great number of heterogeneous resources (not just compute resources, but also network or data resources). Its functionalities can be summarized in the support for various job types; in the ability to manage jobs during their lifetimes and to schedule and to execute jobs based on performance information based on allocation of resources; in the ability to deploy and to configure applications and data to Grid resources. These requirements are partially addressed by Basic Execution Service (BES) [21] and Job Submission Description Language (JSDL) [22] specifications. These standards are differently implemented in the main Grid existing-platforms (eg. Globus or gLite) by independent middlewares. Anyway, most Grid systems have similar features, in fact, based on standard specifications, they define Web Services interfaces for creating, monitoring, and controlling computational entities such as Web Services or parallel programs [21], and clients define jobs using the a XML Schema language for describing jobs [22].

B. ASM-based Method

In this work the Grid job management and execution service is discussed with respect to its abstract model. We use an Abstract State Machine (ASM) to model the Grid infrastructures to provide a higher-level abstraction. There are experimental and theoretical confirmation [10,11] that ASM method simulates, in a step-for-step manner, every algorithms behaviour. In an ASM each step computes a set of updates with given transaction rules: on completion of a step, all updates are committed simultaneously. Furthermore, the abstraction of the model may lead to later refinements as exhaustive as possible, if required. Finally, the language (CoreASM) can
also simulate the system behaviour by specifying the pre and post conditions.

Due to the concurrent and reactive nature of its components (such as web services and computing element) through communication networks, Grid system can be considered as an asynchronous distributed system [10,17,23]. It is distributed because contains different types of entities acting and interacting with their neighbourhood. The global behaviour of the Grid system is the sum of the local activities carried out by the single entities. It is asynchronous because all the entities do not share a unique clock coordinating their activities, but each of them follows behaviours governed by its own internal logic. Under this assumption a given Grid system can be modelled with a set of Asynchronous Multi-Agent ASM, also called Distributed ASM (or DASM). DASM consists of a finite collection of autonomously operating agents that perform their computation steps concurrently.

III. GRID JOB MANAGEMENT AND EXECUTION MODEL

A Grid system offers the capability of managing and executing jobs on distributed computational resources. This research is focused on the construction of a model that explains the management and execution functionalities at a desiderate level of detail, while the aspects concerning the language for describing job are outside the scope of the proposed work.

A. Informal description and requirements

The first step is to define Ground Model with the formalization of the requirements for Grid and Execution Management Services[4,6]. Ground Model represents accurate description of the system one wants implement, so it needs to be an enumeration of accurate definitions and requirements that can relate application domain terms and system behaviour. It has to result not ambiguous and abstract.

The system is always initially in a state of inactivity, i.e. waiting for a job is submitted. When a client application uses the Grid system, a job is submitted. A user can also cancel the submitted job. A job is the smallest unit that the Grid system manages. The system controls the availability of needed resources. Available resources are those resources that meet the system necessities (the matchmaking). Every different Grid middleware controls the resource discovery, allocation and reallocation in different way, that addresses questions of efficiency, stability and scalability and each resource is controlled by its owner host. If the necessary resources are available, these are allocated and ready to begin the computation. The allocation consists of assigning and queuing the job, eventually scheduled, to their local manager, otherwise, the system returns inactive aborting the execution of the submitted job and returning ready for a new job, or for a different use by the host. If all the sub-processes are correctly performed, the execution of the submitted job can be considered completed without problems. On the other case, the system leaves all the resources and returns inactive. A resource could fail the execution of its job because of any problems. If a resource fails the job, this returns inactive.

B. Job State Model

It is evident, from analyses requirements, that the system, during execution of a job, passes through various states. Considering the system states emerged in the process and requirements explanation, jobs traverse the following set of states: IDLE. The system starts in state of inactivity. READY. After the control of the availability of every resource, the system is enabled to start execution of a job on such a resource, which matches job requirements. RUNNING. The job is executing on computational resources. FAILED. The computation can fail due to some error or failure event (such as failure of a computational resource). DONE OK. The job is terminated successfully. CANCELLED, REMOVED. The job has been successfully cancelled on user request. ABORTED. The job processing is aborted by Grid middleware due to some errors or failure events.

C. Transitions

The next step is the definition of the operations that characterize ASM, by appropriate ASM locations, conditions and rules. 1.Once defined a job to submit to the Grid system, if the system is ready, the job is really submitted to the system (SUBMIT). 2.After the job has been submitted, the system needs to find available resources necessary for the computation; if control is positive, the resources are allocated and therefore the job is accepted (ACCEPT); if control is negative, the job is rejected (REJECT), because of the fact that it can not been executed. 3.Once accepted the job, the resources are allocate and the computation starts: the job and the relative sub-jobs will complete the computation with success and the system traces the result of the computation (COMPLETE); otherwise, if problems arise, the system fails and returns to the start point, ready for a new job (FAILURE): in this case a report of the individuated problems can be traced. 4.If the job ends, the system resets its situation and returns in a state of inactivity (FINISH). 5. The user can cancel job before running it (CANCEL) or remove it before it ends (REMOVE).

The research focuses on the abstraction used to formally model the behaviour of Grid system and job management and execution capability. We have captured capability properties and have generated ASM ground model, formalizing it in the ASM rules of the activities, and analysing research problems on appropriate abstraction level. The Grid system is supposed to be multithread: it can be submitted many jobs. Once submitted a job, the ASM results active. If we consider the DASM perspective, the single agent is not available for other job, and other agents of the same type are ready to accept new job; while different agents and observers belong to environment can read locations for analyse the job situation.

IV. CONCLUSION

In this research a new and general abstract model of the job management and execution capability of OGSA has been proposed, where OGSA defines a Grid architecture in term of independent and open standard services. The aim is to formulate an abstract and general schema from the submission to the
completion of a job. The approach has allowed a substantial simplification and abstraction of the architecture. We have used the ASM method, that constitutes an abstract interpreter for the Grid standard system: it results as mathematically accurate, as abstract and intuitive. ASM model provides a powerful software abstraction for Grid application development and deployment. This approach is able to describe components of a Grid system, dynamical properties and relations between them in a service-oriented view, that can be abstracted or refined as the designer needs, based on environment specifications. A ground model to fully capture the informal requirements in a mathematically verifiable code has been constructed. It is a challenge for research to investigate how the ASM models can help the process of interoperability and standardization of Grid computation, interface and collaboration, reducing development costs. The resulting ASM models can be also used as basis for communication between different domain experts and IT technologists.

REFERENCES

[8] W3C - www.w3.org
Self-Adaptive Management of Web Service Discovery

Atousa Pahlevan and Hausi A. Müller
Department of Computer Science
University of Victoria
Victoria, BC Canada
Pahlevan@cs.uvic.ca Hausi@cs.uvic.ca

Abstract—Current web service discovery mechanisms return many irrelevant results because they do not account for attributes that change over time, such as availability. Our work aims to overcome these limitations by tracking and considering dynamic attributes. We show that these attributes play a crucial role in finding relevant services by allowing the utilization of context information to adapt service discovery to dynamic changes. Our algorithm is based on mechanisms similar to those that benefit autonomic computing. The aim of our approach is to increase the quality and ultimately consumer’s satisfaction with the returned results, since many fewer extraneous results are returned. We present experiments to evaluate our approach using a prototype implementation of our model.

Keywords—Web service discovery, Dynamic attributes, Autonomic computing.

I. INTRODUCTION

Web services are characterized both by the functionality they offer and by the quality of the service they provide (e.g., availability or response time). While functionality is typically fixed, quality of service (QoS) changes over time. Existing web service discovery methods assume that characterization of web service is static and provides no support for dynamic attributes [16]. We define the set of dynamic attributes that go along with the discovery request as the context of the request.

While consumers look for services that meet their requirements, they also care about the quality of service they will receive. Without assurance of expected quality, the plethora of results returned from a UDDI registry is not particularly useful [2][3]. We therefore contend that quality of service is part of the context of the request; its expected value determines the kind of results that should be returned [8][9]. Further, consumers may have additional requirements in terms of the data received; such as cost, resources consumed, or geographical location. These should also be considered as part of the context of the request.

The combination of the consumer’s network connectivity, requirements, and quality of service form the context in which the request is formulated. This dynamic context needs to be evaluated in a timely fashion and be taken into consideration when servicing the discovery request.

We modify the service discovery protocol to communicate context along with the request, in order to guide the search engine towards relevant results.

Several context-aware techniques have been developed [7], but none of them apply directly to web service discovery. These techniques are designed for tightly coupled systems, but are not suitable for pervasive environments. Our proposed approach to high-quality service discovery aims to overcome these limitations by considering the situation in which the request was issued [5][6].

We employ autonomic computing technologies to manage dynamic context information in the service discovery process. The discovery process adapts to changing conditions automatically by updating the information in the repository. In other words, a control loop monitors some resources (i.e., dynamic attributes) and autonomously keeps the repository updated. Besides making our system self-adaptive, autonomic computing aids in identifying the quality selection criteria based on both service and user contexts [24]. Finally, the chosen criteria can be used to filter the relevant services.

Our short term objectives include the following methods and algorithms:

- An advanced model to register, collect, measure, and aggregate dynamic attributes, storing them, as well as continuously detecting change and identifying anomalies.
- A means for consumers of a web service to add constraints to the search operation (e.g., preferences, feedback) that reflect the context of the request.
- An enriched method to automatically evaluate the service and the consumer’s context in order to identify the selection criteria automatically, such as type of the service.
- A filtering mechanism for using dynamic attributes as a secondary criterion for service selection.
- A self-adaptive service discovery process that considers situational changes.

II. RELATED WORK

There are several assumptions about web services: one is that public web services are indexed based on syntactic, text-based methods. Intra-corporate services are published and discovered through proper business models that can control the process [4]. To discover public services, three main approaches have been introduced: keyword-based, table-based, and concept-based [12]. The keyword-based approach is simple, yet ineffective, due to the lack of semantics during the discovery process. In the table-based approach, a table of both services and requests is maintained; this service discovery process produces higher precision than the keyword-based approach because
each service’s property values are compared against the query values. Concept-based discovery uses the semantic web, thereby enabling consumers to search by type rather than keyword, and thus producing the higher recall and precision [12]. Our work is a hybrid approach, building on both concept- and table-based discovery methods.

Our proposed approach is based on service-oriented architecture—that is, a scalable and robust framework to manipulate public service discovery [7]. The CB-Sec framework [21] is an approach that combines service-oriented and context-aware computing in order to provide consumers with better services. This framework is designed as middleware to capture the consumer’s context information; however it provides no formal definition of context for services.

Another service oriented context-aware architecture is SOCAM [23], a distributed middleware designed to support the acquisition, discovery, and interpretation of various contexts. It is similar to the context gathering mechanism in our approach: context information can be sensed through physical sensors, defined by users, or interpreted through context reasoning. However, the system is not targeted at web service discovery.

Another approach for context-aware service oriented architecture is CA-SOA [22]. This model is aimed towards ubiquitous service discovery that can support contexts via different components: service agent, user agent, and broker agent. This approach stands out because it is equipped with a real-time context acquisition method and comes with a rule-based algorithm for context matching. The matching algorithm is predefined and cannot change according to context. We attempt to reduce the rigidity of service discovery and increase adaptability, encouraging the use of autonomic computing (AC) in the context-aware service discovery [10]. Thus, our model can be characterized as an adaptive, context-aware, and service-oriented architecture that uses autonomic control loops to provide dynamic web service discovery.

### III. Quality Web Service Discovery

Our static-discovery dynamic-selection (SDDS) architecture for enhanced service discovery is based on evaluating static and dynamic attributes. SDDS is a generic model that can accommodate a variety of service registries and evaluate dynamic attributes at the time of discovery [19][20]. The SDDS architecture supports the provision of dynamic information from both consumers and providers as well as from the UDDI. Two kinds of dynamic attributes can be specified in SDDS: service and user. Our architecture for high-quality web service discovery comprises two phases: Static Discovery (SD) and Dynamic Selection (DS). While the SD process resembles an index method to enable service discovery within a collection of services classified by static attributes, the DS process is initiated for service retrieval based on the dynamic attributes. Figure 1 depicts the overall system architecture for our SDDS service discovery comprising the following components:

- **Service Provider:** Provides services to others; it may also provide facilities for deriving dynamic attributes of services at the time of service discovery.
- **Standard UDDI Database:** Stores the static service descriptions.
- **Dynamic Repository:** Stores the dynamic attributes of monitored services. The dynamic repository is self-managed, self-adaptive, and used for storing recent values of dynamic attributes.
- **User Dynamic Manager:** Deals with dynamic values on the user side. The goal is to select relevant services based on the consumer’s context (preferences, current situation, history). We perform this in three phases: query
be broken down into the following components:

**Autonomic Quality Selection (AQS):** Manages the dynamic repository through the monitor-analyze-plan-execute (MAPE) autonomic loop (i.e., IBM’s autonomic manager) [11]. This enables automated quality checking through an intelligent attribute recognition, measurement, and evaluation process.

AQS continuously monitors the UDDI for changes through sensors looking at the current state of the static repository. When new services are published in the UDDI, the entries are analyzed—a process known as attribute recognition—using information such as user context, service context, domain knowledge, and user feedback. The planning phase, which is policy-driven, identifies a strategy to measure the relevant attributes in order to evaluate the quality of the service [16] [17] [18]. Finally, executors evaluate the service according to the selected strategy by running through the plan just created. In the end, effectors update the dynamic repository with the results of the evaluation. Furthermore the dynamic repository is updated periodically to ensure that the currently tracked attributes fulfill the requirements of the current situation.

The knowledge source that drives this entire loop can be broken down into the following components:

- **Domain Information Base (DIB):** Contains measurable dynamic information related to the semantic domain. It maintains a default weight vector, defined by domain experts, for each service type.
- **Ontology Information Base (OIB):** Stores the database of semantic knowledge in the ontologies of the external domain.
- **Policy Information Base (PIB):** Maintains guaranteed quality requirements in a machine-readable format referred to as service-level agreement (SLA) [16] [17]. This information is considered by the policy as an additional attribute to be used in planning the evaluation strategy for a given service.

Some of this information is not available in advance and should be captured, decided, and operated on at the time of service discovery. In contrast, most existing service selection mechanisms are based on predefined information and are hard-coded.

IV. **EXPERIMENTS**

We implemented the first research prototype of our adaptive, context-sensitive service discovery system as a front-end to a collection of web services accessible through jUDDI [1]. We intend to gather and publish services from sources like geonames.org or seekda.com in a UDDI repository. The services will be chosen from at least three different domains and be either WSDL-based or REST-based. For experimental purposes, we have chosen 250 sample services, a set of service requests, and relevance judgments. To make the retrieval challenging, we look for related services or simply services with similar functionality. The relevance judgments are provided by human experts.

We plan to evaluate the following scenarios:

1) To analyze the impact of dynamic attributes on the quality of web service discovery, we will compare the number of services returned for several queries between the jUDDI and SDDS systems. Preliminary experiments show a significant reduction in the number of returned results against the static service repository, while ensuring that all the returned data from the context-aware search are relevant.

Query 1: Consumer wants a set of services among many functionally identical ones. SDDS achieves better results by returning a smaller set of web services selected through a set of quality attributes. We plan to ask a set of users to label the 300 sample services in order for us to compare the static and dynamic repository. The evaluation is primarily based on a multi-dimensional graded relevance scale [13][14][15] instead of IR retrieval effectiveness measures that are based on recall and precision.

Query 2: Consumer’s preferences are either specified explicitly or captured from their context history. The context is classified as internal (logical) [7]. SDDS achieves a higher degree of correspondence when compared to keyword search mechanisms due to its consideration of preferences in service discovery.

2) We are aware that services from different domains may have different critical properties that might change under different situations. The system’s recognition of such properties through analyzing domain and both user and service context is referred to as adaptability in our system. To evaluate the effectiveness of our adaptive approach, we conduct experiments with and without the AQS component.

Query 3: The critical quality attributes for a given query are automatically chosen by SDDS from the domain information repository. This information is basically static, specified in advance, and thus is neither flexible nor scalable. In fact, these critical attributes are domain specific and cannot be affected by external context information. Examples of external contexts are: light, location, movement, or user emotion. For a given query from different consumers, the results from SDDS always satisfy availability and response time requirements. However, for the same query with AQS, the results are different for each consumer and are based on availability, response time and throughput as seen from the point of view of the consumer.

V. **CONCLUSIONS**

This paper discussed our approach to returning high-quality results in web service discovery and proposed
dynamic attributes as the factor that most significantly contributes to reducing the set of results returned while maintaining quality. We introduced dynamic attributes as part of the context information that affects the web service discovery quality to adapt to the current conditions of both services and web service consumers. We proposed a new model in which functional and context information is to be taken into account for finding high-quality relevant web services.

We employ autonomic managers to monitor dynamic attributes and user context to obtain quality selection criteria. Finally, the chosen criteria are used to constrain the discovery of relevant services. The feasibility of our method is demonstrated with a prototype implementation of our model.

We show that increasing the adaptability of the web service discovery by including context information gives us a significant reduction in results returned compared to static web service discovery methods.

Further research is needed to establish the matching algorithms that can handle multiple attributes and work under different matching criteria, as well as to find a good ranking mechanism for the discovered web services. Moreover, we want to utilize a policy-based mechanism in order to select the quality checking actions that need to be applied to the system. The policy would describe the adoption plans for the service discovery based on the changing attributes of services and consumers.

REFERENCES


Towards realizing WS cross-layer monitoring and adaptation

Chrysostomos Zeginis,1,2
1Department of Computer Science
University of Crete
GR 71409 Heraklion, Greece
{zegchris,dp}@csd.uoc.gr

Dimitris Plexousakis,1,2
2Institute of Computer Science
Foundation for Research & Technology - Hellas
GR 71110 Heraklion, Greece
{zegchris,dp}@ics.forth.gr

Abstract—In this paper we pinpoint the problems stemming from the lack of cross-layer monitoring and adaptation techniques across Service-Based Applications. We make a short review of the corresponding fragmented state-of-the-art approaches and identify the basic research challenges for each of these issues. Finally, we propose a tentative research plan for our future work.

Keywords—monitoring; adaptation; cross-layer;

I. INTRODUCTION

Web services are an emerging technology which has attracted a lot of attention from both academic and industry areas in recent years. Many European projects have funded research towards Service-Based Applications and more and more businesses adopt them in order to facilitate and automate their business processes. There are many issues involved throughout the life-cycle of a Service-Based Application. In this paper we concentrate on two of them, service monitoring and service adaptation.

Once services and business processes become operational, their progress needs to be managed and monitored to gain a clear view of how services perform within their operational environment, take management decisions, and perform control actions to modify and adjust the behaviour of Web-services-enabled applications. In [1] Web service monitoring is defined as a process of collecting and reporting relevant information about the execution and evolution of Web services. Many monitoring approaches have been proposed so far, most of them focusing on measuring QoS metrics, gathered on the basis of SLAs, given that an SLA dictates what are the service obligations in terms of performance. The measured metrics are then compared with the corresponding SLA bounds in order to detect possible violations.

The dynamic and ever-changing nature of the business and physical environment requires Web services to be highly reactive and adaptive to the changes and variations they are subjected to. They should be equipped with mechanisms to ensure that they are able to adapt to meet changing requirements. Such changes can be identified, detected, and foreseen in the running Web service during the, previously mentioned, monitoring process. In [2] the authors define Web service adaptation as a process of modifying Service-based Applications (SBAs) so as to satisfy new requirements dictated by changes of the environment on the basis of Adaptation Strategies designed by the system integrator. An adaptable SBA is an application that continuously monitors and modifies itself on the basis of these strategies.

It is obvious that Web service monitoring and adaptation are two well-connected processes that result in the correction and customization of faulty services, so as to coincide with the new requirements. But, where should these processes take place? An application based on Web services can be presented by three functional layers, namely the Business Process Management layer, the Service Composition and Coordination layer and the Service Infrastructure layer. These layers are closely related and are all affected during the execution of Web services. A thorough review of current monitoring and adaptation techniques reveals that these techniques are very fragmented and are not applicable on more than one layer.

Although current approaches cover a wide spectrum of monitoring and adaptation, none of them can efficiently cover all layers. They are usually considered in isolation from each other and focus on a local solution for a specific monitoring or adaptation requirement without taking into account the effects to the other SBA layers. While they seem to be quite effective on a specific layer when considered in isolation, they can cause problems and incompatibilities to the other layers when adaptation actions take place. Different artifacts at one layer may refer to the same artifacts of another layer, while such relations are ignored by the isolated monitoring and adaptation solutions. As a consequence, wrong problems are detected, incorrect decisions are made, and the modifications at one level may damage the functionality of another layer.

The rest of this document is organized as follows. Section II explores the research challenges that are associated with cross-layer monitoring and adaptation, section III offers a preliminary plan for addressing the identified challenges and section IV concludes.

II. RESEARCH CHALLENGES IN CROSS-LAYER MONITORING AND ADAPTATION

While many organizations are adopting service-orientation due to the advantages that it promises, the technology that realizes it is still immature and various research issues have to be solved before service-orientation is able to meet all of its promises. In particular, many of...
the service life-cycle activities are not so well supported. Among them, web service monitoring and adaptation offer many perspectives for the future and researchers will have to focus on offering more advanced approaches and solutions. More precisely, novel research must be conducted on cross-layer issues, such as the cross-layer monitoring and adaptation of business services.

In this section, we provide the details of the aforementioned literature review on web service adaptation, revealing some key research challenges in adaptation and monitoring of SBAs. Specifically, we pinpoint the need for developing new, more advanced approaches towards cross-layer monitoring and adaptation of web services. This review [3] is based on a representative sample of the current adaptation approaches, which were compared according to a set of criteria including adaptation goal, the applied functional layer and adaptation trigger.

Taking into consideration the advantages of all these approaches we can conclude that adaptation has gained a significant attention for service integration, repairing the failures and reconfiguring the systems. Furthermore, there are still no approaches addressing the adaptation across all the functional layers of the service based systems efficiently. The proposed approaches address either a particular functional layer, e.g. [4], or a particular problem, e.g. [5] addresses only the problem of identifying mismatches. Moreover, some other important aspects such as contextual information are worth investigating. SBAs are commonly designed to be offered to various types of users, and therefore they should be able to be customized and personalized according to information collected by them. However, little investigation has been conducted and most current monitoring and adaptation approaches often omit the role of application context.

As mentioned before, there are many works focusing on a single functional layer of SBAs. [6], for example, addresses the problem of monitoring in the Service Infrastructure layer, proposing a Grid Monitoring Architecture. In the same layer, Rashid et al. [7] relate QoS to Grid Services in order to optimize resource utilization. In the Service Composition and Coordination layer, Baresi et al. [8] monitor dynamic service compositions with respect to contracts expressed via assertions on services, while in [9] the authors propose an approach towards dynamic service composition based on multi-dimensional optimization of quality of service metrics. At the top level of Business Process Management, Castellanos et al. [10] deal with the problem of how to create a monitoring solution, which not only enables the KPI measurement, but also the identification of the causes of undesired KPI values, and prediction of future values, while Ardagna et al. [4] introduce PAWS, a framework for flexible and adaptive execution of managed service-based processes. Figure 1 depicts some of the current fragmented monitoring and adaptation techniques adopted in each of the three functional layers of an SBA.

A key challenge stemming from the state of the art approaches is to provide a comprehensive and complete framework integrating them in an aligned manner. Gjørvøen et al. [11] make a first attempt towards cross-layer Self-adaptation of SOAs, but this approach is coarse-grained, as they adopt the SBA layer discrimination proposed by Erl [12], namely the Service Interface layer and the Application layer. The resulting system is able to perform cross-layer adaptation exploiting mechanisms in both layers in a coordinated fashion. In addition, Kazhamiakin et al. [13] define the mechanisms and techniques that are necessary for addressing the requirements and for constituting an integrated cross-layer framework. These two approaches tackle the cross-layer adaptation problem in an inflexible manner, as the adaptation logic is predefined and static. Finally, Popescu et al. [14] provide support for cross-layer adaptation using adaptation templates that are composed either directly, through invocations of WSDL operations or indirectly, through events.

A major problem of fragmented adaptation techniques is that adaptation actions at one layer may lead to unpredictable malfunctions to another layer. This lack of effectiveness is caused because adaptation strategies don’t take into consideration the functional properties of the other layers. For example, at the Service Composition and Coordination layer in order to reduce execution the adaptation aims to execute independent tasks in parallel by delegating them to different services. In order to achieve this effect it is necessary that those services are independent also at the Service Infrastructure layer.

Another problem caused if monitoring mechanisms are performed in isolation from each other is that the corresponding monitoring events are not aligned and the information needed for possible adaptation actions is not propagated adequately across the layers. This may lead to the situation where the source of the problem is not defined correctly and this may trigger contradictory adaptation actions at different layers. Consider an example, where due to a bottleneck in the service execution environment the performance of the involved services goes down. This problem may be detected independently from the Service Infrastructure and the Service Composition Layer, which then trigger different adaptation actions.

Similar to the previous problem is the incompatibility between the adaptation actions of one layer and the requirements and constraints posed by other layers. For example an adaptation action that performs resource allocation in the Service Infrastructure layer can infringe some constraints or requirements in the Service Composition layer.

Another limitation of the proposed approaches is that they are not flexible enough to accommodate a wide range of scenarios or use cases. Most of the approaches deal with QoS changes and the related adaptation strategies, such as service replaceability, or with a particular set of application faults and the recovery actions. Accordingly, the kinds of goals, as well as the adaptation actions are rather simple. While on the one hand this allows for simple and efficient adaptation implementations, on the other hand this prevents creation of comprehensive and multi-
dimensional adaptation methodologies. This requires new languages and techniques that would enable more complex and expressive adaptation capabilities, with respect to those currently presented in the literature. In particular, it is necessary to provide notations that interleave the goal-oriented and action-oriented specification that would give more freedom to the platform in identification and definition of the necessary adaptation strategies in the application production mode.

To this end, an important problem is how to integrate various constraints on different aspects of the application functionality with the adaptation technique. On the one hand, there is a need to express real-world domain knowledge and properties, while on the other hand this knowledge still has to be manageable by the underlying adaptation realization. An important issue is also a possibility to incorporate different types of constraints (e.g., behavioural, QoS), and hence to come up with hybrid analyzers and solvers for those constraints within the adaptation framework.

Furthermore, concerning the healing of faulty services, it is very important to take into consideration the QoS description of the services. Semantic QoS descriptions, in particular, allow us to find metrics that are similar or refer to the same service property. In addition, such descriptions facilitate the choice between candidates with similar functionality, when replacement is selected as the preferred adaptation strategy. An approach towards this direction is proposed by Kritikos et al. [15], who present OWL-Q, a novel, rich and extensible ontology-based model for describing the QoS of Web services, that complements OWL-S.

Finally, there is a need to target not only the adaptation types, where the goal is to modify the system in reaction to the changes that already happened, but to adapt the system before these changes take place, i.e., preventive or proactive adaptation. This, however, requires novel techniques and methods for the application diagnosis in order to foresee the possible changes and the potential consequences of them.

III. RESEARCH PLAN

For our research plan we intend to explore possible cross-layer monitoring and adaptation mechanisms that can express monitored events and adaptation strategies in a holistic and aligned manner. These mechanisms must identify exactly which is the source of the problem and then trigger apt adaptation strategies according to the adaptation needs of each layer.

Towards this direction we will try to identify and model the cross-layer dependencies. Policy refinement solutions proposed in [16] can be very helpful. As the complexity of web services continues to increase the complexity of their management systems, including those for QoS management, also increase. Within Policy Based Management systems (PBMS) the desired behaviour of the managed system is specified in a policy. Policies can be specified at many different levels of abstraction from high-level business goals to policies for individual resources. It would be ideal for an organization to derive their low level policies from their overall business goals (high level policies). In the context of SBAs, policy refinement is capable of automatically mapping high level goals of a composite service down to low level policies.

Furthermore, in this research framework we intend to explore the work of Hielscher et al. [2] as a possible proactive adaptation solution in this overall cross-layer monitoring and adaptation framework. Response time is very important in novel SBAs. So, proactive adaptation mechanisms that would anticipate the possible changes and the potential consequences of them would be ideal in a business environment.

Finally, foreseeing the perspectives offered by Aspect Oriented Programming (AOP), we are planning to explore in more depth the work of Karastoyanova and Laymann [17] in combination with the approach of Popescu et al. [14], to investigate a possible way of weaving aspects (i.e. WS operations) in BPEL, where joinpoints are
identified by taxonomies of adaptation mismatches. Taxonomies can be used to semi-automate the discovery and selection of aspects, needed to fulfil cross-layer adaptation requirements.

In a nutshell, the expected results of our research are summarized in a software service management framework able to support both proactive and reactive cross-layer service monitoring and adaptation. We expect our work to be a part of a complete framework that will be enriched with the work of Kritikos and Plexousakis [18] and the work of Baryannis and Plexousakis [19]. This framework would address many issues appeared in all the service life-cycle activities.

IV. CONCLUSION

In this paper, we identified the major problems stemming from the lack of cross-layer monitoring and adaptation techniques. Current approaches are very fragmented and there is no complete cross-cutting technique that would take into consideration the dependencies between the functional layers of SBAs. Thus, it is crucial to explicitly relate different conceptual elements to each other across different layers. To achieve this, there is need to develop high-level models, which relate specific elements of the SBA, specific associated adaptation strategies and mechanism, and specific monitoring events and mechanisms available at different layers and peculiar to different approaches. Based on these models, novel cross-cutting approaches can propagate aligned monitored events and reflect the relations and the impact of the adaptation activities on the different layers. Finally, the underlying research challenges were also analyzed and a tentative research plan to address them was outlined.

ACKNOWLEDGEMENT

The research leading to these results has received funding from the European Community’s Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

REFERENCES


Information Integration in Service-Oriented Computing

Mohammed AbuJarour, Felix Naumann
Hasso-Plattner-Institut
University of Potsdam, Germany
{firstname.lastname}@hpi.uni-potsdam.de

Abstract—Information Integration has been the typical approach to data-driven applications in several domains, such as, enterprise applications. However, applying information integration techniques in Service-oriented Computing (SOC) is not straightforward, because of the lack of adequate information resources, such as rich service descriptions. This lack of rich service descriptions reduces the (re)usability of web services. In this work, we propose a novel approach and platform to alleviate this problem and investigate the benefits of information integration in SOC, where information about web services is gathered from multiple sources, e.g., service providers, consumers, invocations, etc., and integrated in rich universal service descriptions.

I. SERVICE-ORIENTED ARCHITECTURE

The basic principles of Service-oriented Architecture (SOA) are captured by the triangular SOA operational model with the three roles of service-provider, -consumer, and -registry (cf. Fig. 1). The goal of this operational model is to achieve the main features of SOA, e.g., high interoperability, flexibility, scalability, fault tolerance, etc. However, to achieve such goals, service consumers should have sufficient knowledge about the considered services. Much of this knowledge comes from service providers themselves in the form of service descriptions, e.g., WSDL.

The popularity of SOA has been increasing in enterprise and web applications, especially due to the emerging trends of Software-as-a-Service and Cloud Computing. This increasing popularity has been reflected in an increasing number of public web services. However, this increasing number of web services has resulted in limiting their (re)usability. One of the aspects of this limited usability is the removal of service registries from the traditional SOA operational model [9]. One of the main reasons behind the limited usability of web services is the lack of adequate information about them. This information is typically provided by service providers in the form of service descriptions. The lack of rich service descriptions has been a vital research topic in Service-oriented Computing (SOC) [8].

To increase the usability of the offered web services, several approaches have been proposed to enrich service descriptions. Typically, a single source of information is used to enrich service descriptions, e.g., service providers, service consumers, or domain experts. In our approach, we propose an information integration approach, where information about web services is gathered from several sources and integrated into rich universal service descriptions.

The remainder of this paper is organized as follows. Sec. II introduces the research context and gives further details about the research problem and challenges. We explain our proposed approach in Sec. III. Then, we highlight the significant research papers in this field in Sec. IV. Finally, we conclude and summarize our next steps in Sec. V.

II. RESEARCH CONTEXT

The available tools that enable service providers deploy their systems as web services and the popular Software-as-a-Service and Cloud Computing trends have helped increasing the number of public web services. However, this easiness in service creation has complicated the problem of service usability because such services usually lack rich description artifacts that are vital in service discovery.

Service providers tend to focus on the implementation aspects of their services rather than giving rich service descriptions. Usually, service descriptions appear in the form of comments and notes by service developers [10]. Such descriptions are technical-oriented and not suitable for non-IT people. In the Internet of Services (IoS) initiative, this fact has been reflected in the new concept of business services that abstract the technical web services to fit the background of their intended users, namely, business people [4].

In the traditional SOA operational model (cf. Fig. 1), the service registry acts passively. It reacts to register-requests from service providers by adding the intended service to its service collection. It reacts to discovery-requests from service consumers by finding the best matching services among its service collection. It reacts to discovery-requests from service consumers by finding the best matching services among its service collection, with respect to the submitted query. Although its role is vital, we believe that service registries can do more, especially in enterprise applications.

In our research, we aim at enriching service descriptions, maximizing the benefits of existing descriptions, and inte-
The expected contributions of this work are: benefits of enriching service descriptions in SOC? The key challenges that drive our research include:

- Dynamic SOA and business environments: Business needs and conditions are always changing by nature. To respond to these changes, services and service providers have to adapt by modifying their services or providing new services in a time-effective manner.
- Increasing complexity: This complexity of services is due to the increasing number of tools, frameworks, domains, and business needs that support or require web services. In most cases, this is reflected in the lack of adequate service description artifacts.
- Inadequate criteria for service discovery and selection: Full-text search is usually used by service registries as a means of service lookup, but the quality of the result list depends on the information used to lookup a service. Furthermore, non-functional requirements of the services should be considered in the lookup process to make service discovery more comprehensive.

The expected contributions of this work are:

- A set of techniques to enrich service descriptions.
- A novel approach to enhance service selection and discovery.
- Assessment methods of service discovery by service consumers.
- An integration environment for SOA applications.
- A proactive service registry and repository, coined “Depot”.

III. DEPOT: A PROACTIVE WEB SERVICE REGISTRY

Our SOC information integration environment is coined “Depot”. The architecture of Depot is shown in Fig. 2. Depot uses a focused crawler (component 1) to collect public web services from the websites of their providers. The collected web services are validated, parsed, and annotated with information, which is gathered from the websites of their providers using the WSDL and information parser (component 2). This gathered information is stored in a service database. Service consumers can discover web services through the web service explorer (component 3) that provides a personalized result list of web services based on the profiles of service consumers through the personalization filter (component 4). Moreover, service consumers can invoke the selected web services using the web service executor (component 5), where HTML forms are provided to collect service inputs from service consumers. This feature enables Depot to gather further information about the invoked web services and the invoking service consumers. This information is gathered using the invocation analyzer (component 6) and stored as service metadata. Further service metadata can be provided explicitly by service consumers in the form of service annotations through

The community annotation handler (component 7). Depot can use the collected information about web services, service providers, and consumers to notify consumers about new web services that might be relevant to their business through the web service recommender (component 8).

In our initial prototype of Depot [1], we introduced (among others) a focused crawler and a web service executor. In [2], we have extended this prototype by introducing the WSDL and information parser (component 2) and the web service explorer (component 3). Further components will be added incrementally.

As an information integration environment, Depot uses three different sources of information about its managed web services, namely, service providers, service consumers, and invocation analysis (cf. Sec. III-A). Internally, this information is divided into service data – e.g., the location of its WSDL – and service metadata – e.g., number of invocations of a web service – (cf. Sec. III-B). Depot performs several tasks proactively. It employs web crawling techniques to collect public web services automatically. Personalized lists of web services are returned to service consumers that reflects their requirements and behavior. New web services are recommended to their potentially interested consumers based on their profiles proactively (cf. Sec. III-C).

A. Sources of Information in Depot

To handle the challenge of inadequate criteria for service discovery and selection, Depot uses several sources of information to enrich service descriptions. These sources are service providers, consumers and invocation analysis.

**Service providers**: service providers release information about their offered web services in the form of service de-
scriptions. Typically, in XML-based formats, e.g., WSDL, WADL, etc. However, it has been observed that service providers tend to focus on the implementation aspects of their web services rather than providing rich service descriptions. To handle the problem of poor service descriptions released by service providers, Depot follows a proactive approach, where it collects public web services and additional information about the collected web service from the websites of their providers through web crawling techniques.

**Service consumers:** Feedback and ratings are the most common types of information that service consumers provide about the web services they have used. The goal of such information is to assess the quality of the used web service(s), e.g., response time, cost, performance, security, etc. This information is also used to evaluate the reputation of service providers. We extend this approach in Depot by allowing service consumers to annotate the web services they used to enrich their descriptions. Users annotations can be simple tags or detailed descriptions on how a web service behaved or functioned when they invoked it.

**Invocation analysis:** Depot provides interfaces where consumers can use web services directly. Web forms are used to collect inputs from users [1]. This feature gives Depot the opportunity to analyze such service invocations and learn more about the considered web services. Several things can be extracted from such analysis: caching, classifying service consumers, tagging web services, resolving ambiguity in terms used to describe services, the context where a web service is invoked.

**B. Service Data and Metadata**

The information about web services, providers, and consumers that Depot manages is split in two classes: service data and service metadata. Service data includes information that identifies each web service, such as its provider, the URL of its WSDL, its name, category, etc. This class is typically provided by service providers during service registration or by the focused crawler in Depot. Our approach to collect public web services and their information is introduced in [2]. Service metadata includes other information that Depot stores about each web service, such as invocation frequency, statistics, invoking service consumers, etc.

**Service metadata:** In Sec. III-A, we described the sources of information that Depot uses to gather information about web services. Part of this information is taken from service consumers in the form of service annotations. Additional information is extracted from service invocation analysis. Furthermore, a user’s profile tracks service usage history where information about previous services’ invocations by each user is tracked and gathered. Depot can make use of this history of invocations to rank the list of retrieved services according to user’s preferences (tastes) that are derived from its previous service usage.

**Web service explorer:** Depot uses the information about web services, service providers, and service consumers to provide an enhanced service exploration and discovery for service consumers. Four types of service exploration are provided by Depot:

- **Browse by provider:** This type of service exploration enables service consumers to find relevant web services from specific service providers. For instance, service consumers prefer to use web services from service providers with high reputation or well-known providers.
- **Full-text search:** This type requires basic knowledge in the application domain to choose “good” keywords, e.g., address normalization, credit card validation, etc.
- **Browse by category:** The increasing complexity of web services and their driving business needs makes finding “good” keywords for full-text search a difficult task. For such cases, Depot provides web service browsing based on categories. Collected web services are automatically classified in several application domains, e.g., education, finance, entertainment, etc.
- **Browse by tag cloud:** For a quick way of exploring common web services, regardless of their providers or categories, Depot provides a tag cloud that enables service consumers to browse through common tags attached to web services. Part of these tags are automatically generated from websites of service providers during service crawling through WSDL and information parser (component 2). Additional tags are provided by service consumers in the form of community annotations.

**C. Proactive Operations of Depot**

Depot performs three operations proactively, namely, web service crawling (component 1), personalization of web service discovery (component 4), and web service recommendation (component 8).

**Crawling of web services:** With the increasing number of public web services and the increasing complexity of web services and their functionalities (cf. Sec. II), it is no longer a feasible task for service providers to manage all their web services and keep their related information up-to-date in all service registries they use. It has been shown that 48% of a production UDDI registry has links that are unusable (tModels tested only); these links point to contain missing, broken or inaccurate information [6]. Therefore, Depot uses web crawling techniques to avoid such situations. Depot collects public web services from the websites of their providers, gathers additional information about these web services from their provider’s website, and registers them. Though, iterative and incremental crawling of the considered websites is required to keep all information about web services up-to-date.

**Personalization of web service discovery:** Part of the metadata in Depot is consumers profiles that track service invocations by each service consumer. This information is used to provide a personalized ranked result list of web services during service discovery to each service consumer. This information is used by the personalization filter (component 4) to filter and rearrange web services in the result list to fit the requirements of each service consumer. This proactive service
discovery meets the challenge of increasing complexity of web services.

**Recommendation of web services:** In today’s marketplaces, change has been the rule rather than the exception. IT systems should adapt to changes quickly. On the other hand, organizations would like to utilize each available IT technique to perform their business efficiently. This situation requires a proactive behavior to find new web services (crawling) and announce them to potential interested service consumers (recommendation). This recommendation is based on the metadata that Depot maintains about web services and consumers. The profile of each service consumer helps Depot identify such potential consumers whenever new web services are found.

**IV. RELATED WORK**

Most existing service registries and repositories are based on UDDI, ebXML, or a mix of both: Centrasite is a UDDI service registry that is limited to the web services inside a single organization [5]. Sun’s service registry is based on ebXML 3.0 with added support for UDDI 3.0 [11]. IBM’s WebSphere Registry and repository mainly manages services’ metadata that is gathered from all available resources, such as UDDI registries [7]. Most of these registries use service providers as a single source of information.

The limitations in the traditional SOA operational model have been highlighted by several researchers. In [9], the authors show that the triangular model is not used widely in practice because of the limited role of service registries. Removing the role of the service registry violates the basic principles of SOA, namely loose coupling and dynamic binding. Another approach to achieve active web service registries is introduced in [12]. The authors use RSS feeds to announce changes in the registered services to interested service consumers. The information provided by such feeds is generated by service providers, who tend to focus on the technical parts of their services. Moreover, such a service registry cannot force service providers to notify them about any updates so that they can add RSS feeds to announce the corresponding changes.

The main reason behind the highlighted limitations of the triangular model in the aforementioned work is the lack of rich service descriptions [8]. Therefore, researchers have proposed several approaches to gather information about services to handle the problem of poor service descriptions.

In [3], the authors use a specialized crawler to collect web services from multiple UDDI registries. Although the idea of using crawlers to collect web services is innovative, restricting it to UDDI registries does not give the maximum benefit of web crawling, as such an approach is still limited to what service providers announce during service registration.

Another web service crawler has been introduced by the EU project Seekda (http://www.seekda.eu). In this recent project, the authors use a specialized crawler to collect public web services over the web, and present them in a web 2.0 environment, which allows users to annotate, tag, and use them. We extend this approach by annotating the collected web services with automatically extracted descriptions and generated tags.

Additionally, we enrich service descriptions with metadata extracted from service invocation analysis.

**V. SUMMARY AND ROADMAP**

In this paper, we introduced an approach to increase the (re)usability of public web services by enriching their poor descriptions through a proactive service registry, called “Depot”. It uses three sources of information about web services: service providers, service consumers, and invocation analysis. Depot achieves three features proactively, namely, crawling of web services, personalization of web service discovery, and recommendation of web services.

We have already implemented the components of focused crawler, WSDL and information parser, web service explorer, and web service executor. Our next steps include an implementation of the invocation analyzer and other components.

**REFERENCES**


32
Towards Realizing Dynamic QoS-aware Web Service Composition

George Baryannis,†,‡  
1Department of Computer Science  
University of Crete  
GR 71409 Heraklion, Greece  
{gmparg,dp}@csd.uoc.gr

Dimitris Plexousakis,†,‡  
2Institute of Computer Science  
Foundation for Research & Technology - Hellas  
GR 71110 Heraklion, Greece  
{gmparg,dp}@ics.forth.gr

Abstract—In this paper, we identify two major issues related to Web service composition: the lack of formal specifications for services and service compositions and the inability of current service composition approaches to support dynamicity and QoS-awareness in an effective and scalable way. We analyze the underlying research challenges for each of these issues and propose a tentative research plan to address them.

Keywords—formal specification; service composition; QoS;

I. INTRODUCTION

The last decade has seen an exceptional wealth of research on topics related to Web services, especially by European researchers. The European Union 7th Framework Programme (FP7) has funded several service-related projects such as the Service Centric System Engineering Integrated Project (SeCSE), the Software Services and Systems Network of Excellence (S-Cube) and other projects collaborating under the auspices of Networked European Software and Services Initiative (NESSI).

Research on Web services covers a multitude of issues that are involved throughout the life-cycle of a Web service or a Service-Based Application (SBA). Among them, service description and service composition have attracted a great deal of attention from researchers. Automated Web service composition, in particular, has been a "silver bullet" in Web service research and many approaches that involve automation in the creation of the composition schema as well as during its execution have been proposed, with varying degrees of success. By conducting an extensive literature review, two major interrelated problems were identified: the lack of formal specifications for service compositions (or even for atomic Web services) and the inability of automated Web service composition approaches (especially the ones that employ AI planning techniques) to simultaneously satisfy requirements such as QoS-awareness, dynamicity and scalability in an effective way.

The existence of complete, formal, rich, semantic-aware specifications for Web services is of crucial importance for both service providers and service consumers. Service providers will be able to provide complete specifications of what they are offering, which would enable them to more effectively advertise their service products to potential clients. Service consumers, on the other hand, will be able to be informed of the exact way in which a service is expected to perform and the produced results, which will allow them to make informed choices and select the service that is the most suitable match for their requirements. By automatically processing specifications, service matchmaking can be achieved in a faster and less costly manner. Of course one needs to find the right balance between formality and tractability in processing specifications. Thus, formal specifications are one step towards the embracing of Service-Oriented Architectures by the industry.

While conducting the aforementioned literature review in automated Web service composition, we pinpointed a series of requirements that need to be met for an approach to be successful. These include dynamicity, semantic-awareness, QoS-awareness, non-determinism, partial observability, scalability, correctness, domain independence and adaptivity. Table I contains the results of a comparative analysis based on these requirements. The complete analysis is provided in [1]. As one can observe in the table, some of the requirements were satisfied by the majority of approaches, while others were rarely explored by the authors. To the best of our knowledge, no approach succeeds in satisfying the complete spectrum of the requirements we identified. Approaches that employ AI planning techniques, especially planning as model checking [2] seem to be more successful to that effect. At the same time, even this family of automated composition approaches hasn’t succeeded in providing a means for dynamic QoS-aware service composition that is effective and scalable.

The rest of this document is organized as follows. Section II explores the research challenges associated with formal service specifications while Section III deals with those related to automated Web service composition. Section IV offers a preliminary plan for addressing the challenges that were identified and Section V concludes.

II. FORMAL SPECIFICATION OF WEB SERVICES AND SERVICE COMPOSITIONS

In this section, an analysis of the open issues concerning the description and formal specification of Web services and service compositions is carried out. These issues concern the ramification problem, the qualification problem and the problem of automatically deriving specifications for Web service compositions.
A. Addressing the ramification problem

In previous work [3], we identified the existence of the frame problem in Web service specifications that use the precondition-postcondition notation and proposed a solution inspired by work on procedure specifications. This solution was applied to the most prominent Semantic Web service description frameworks, such as OWL-S and WSMO. The frame problem is closely related to two other problems in logic-based knowledge representation, the ramification and qualification problems. In [4], the author defines the ramification problem as the problem of adequately representing and inferring information about the knock-on and indirect effects (also known as ramifications) that might accompany the direct effects of an action or event. The frame and ramification problems are somewhat contradicting: if one solves the frame problem by disallowing any effects except the ones explicitly stated, any solution to the ramification problem is precluded.

The ramification problem has not yet been examined in correlation with Web services and the effects it may have on their description and formal specification. In that context, one may deal with effects that are caused by effects of a service execution, i.e. executing a service affects the world in a certain way, which then leads to secondary effects that are derived from the primary ones. Finding a way to include such effects in a formal Web service specification while at the same time avoiding any conflicts with the aforementioned solution to the frame problem poses an interesting challenge.

Ramifications must be included in a service specification since the service consumer must be aware of all (direct and indirect) effects of the service execution. This is particularly important in service compositions, as the lack of knowledge of an indirect effect may lead to the assumption that a composition is valid and correct while that particular effect may contradict a precondition of another participating service, leading to an inconsistent composite service.

B. Addressing the qualification problem

While the ramification problem deals with the effects of an action, the qualification problem deals with the circumstances and conditions that must be met prior to the execution of an action. From its definition in [4] one can derive many correlations to the case of Web services. First of all, updating qualifications when new statements become part of our knowledge is something very common in the ever-changing world of Web services. This facet of the qualification problem is closely related to the adaptation qualities of a Web service framework. Hence, a solution to the qualification problem for Web service specifications would involve adapting such specifications whenever new knowledge is acquired, maybe due to new services becoming part of the repository at hand or due to the service provider modifying a specification.

Moreover, the qualification problem becomes very interesting in the case of composite service specifications. It is interesting to explore how the introduction of a new service in a composition schema (either as a replacement to an existing one that failed or by modifying the schema itself) affects the qualifications of services already participating in the composition. It is even more challenging to address such questions not at design-time, but at runtime, when the composite service is being executed and monitored.

C. Automatic derivation of composite specifications

The final challenge of this section serves as a bridge between service description and service composition. While service description frameworks attempt to describe service compositions using a variety of composition models ranging from orchestrations to choreographies to Finite State Machines, no framework (to the best of our knowledge) attempts to handle the problem of automatically producing specifications for a composite service, based on the specifications of the participating services. Likewise, no automated Web service composition approach attempts to derive a complete specification of the inputs, outputs, preconditions and effects (IOPEs) that should be provided to a service consumer.

Providing such complete specifications can be proven very useful since they can be used as complete formal descriptions of what the composite services are offering to consumers in terms of functionality and under what circumstances this functionality can be achieved. This will promote and facilitate the reusability of composite services. Such specifications would also be of great assistance when one attempts to deduce whether a set

---

1 A ✓ denotes that the particular requirement has been addressed by multiple approaches in the category. A ~ denotes that the particular requirement has been addressed by some approaches in the category.
of services can actually be composed in a meaningful way. If inconsistencies are detected during the process of automatically creating the composite specifications, then the set of services is not composable and a replacement should be found for the service or services that cause the inconsistencies.

Automatically deriving the IOPEs of a composite service is directly linked to the way services are orchestrated. A trivial solution would be to expose the full set of preconditions and postconditions of all participating services. This, however, wouldn’t scale well to large compositions consisting of many complex services. Thus, a middle-ground solution should be explored: composite specifications have to reveal some part of the specifications of some of the internal services. The challenge is to decide what should be included and what should be left out.

III. AUTOMATED WEB SERVICE COMPOSITION

In this section, we will briefly summarize the research challenges that were derived from the state of the art survey and comparison we conducted, the results of which are shown in Table I. Notice that we exclude any discussion about the requirement of adaptivity. We consider service adaptation to be a major research area with many interesting challenges, hence it needs to be dealt with separately from our research agenda, which concerns service composition.

A. QoS-Awareness in AI Planning techniques

When examining Table I with regard to QoS-Awareness, one can deduce that while it has been explored (adequately or not) in several categories of approaches, it has been largely ignored in the case of AI planning. The only AI planning effort in our state of the art survey that deals with QoS is the work of Zeng et al. [5]. This work, however, fails to meet very important requirements such as support for non-determinism and partial observability as well as scalability and domain independence.

Introducing QoS-awareness in AI planning techniques involves making at least three fundamental decisions. The first decision is to choose a planning technique which will then be adapted to be QoS-aware. The comparison table can again be useful to making that decision as we can choose a technique that has shown promising results in meeting the rest of the requirements that we have set for automated Web service composition. Planning as model checking is one such technique, as it is the only one that addresses the issues of nondeterminism and partial observability.

A second decision involves the QoS model that will be employed. Rather than creating one from scratch, a preexisting one could be used, as long as it satisfies some important requirements such as being expressive, formal and fine-grained, as well as being based on Semantic Web concepts. Another important aspect is to decide on which phases of the composition process the QoS characteristics will be applied and the effects of that application. For example, introducing non-functional characteristics in AI planning techniques may assist in speeding up the plan generation by limiting the search space when excluding services that don’t meet preset QoS thresholds. However, it may also do the exact opposite if we include QoS criteria in the plan goal, since it will make the goal more complex and harder to satisfy. It is apparent that introducing QoS-awareness is a rather challenging task.

B. Dynamicity in AI Planning techniques

As is the case with QoS-awareness, dynamicity is another requirement that hasn’t been adequately explored in AI planning techniques. The vast majority of them produce a static composition schema (the generated plan) without exploring the case of run-time composition where composition schemas are abstract and are only made concrete at run-time. The most prominent exception is the work of Klusch et al. [6] which achieve some level of dynamicity through the use of replanning. Peer uses it to recursively generate plans that are incrementally closer to the required goal. This introduces dynamicity to the composition process but only at design time.

On the other hand, Klusch et al. use replanning whenever an agent detects that a service’s preconditions have been violated during the execution of a plan. The replanning module is informed of the position of the error at the plan and tries to fix the problem by searching for an alternative path in the connectivity graph from that position onwards. To reduce the search space, unnecessary services can be blocked to avoid a complete preprocessing phase.

Klusch et al.’s effort is a step in the right direction, however, as is usually the case, it lacks many other important features, such as support for nondeterminism and partial observability, QoS-awareness, scalability and any proof of correctness. It should be challenging to explore how dynamicity at run-time via replanning can be applied to planning techniques that support most of these features, such as planning as model checking. Thus, the advantages of both approaches could be combined to result in a composition framework that achieves most of the composition requirements we have set.

C. Scalability

Another requirement that needs to be satisfied by an effective automated composition framework is that of scalability. Some of the approaches that we examined explicitly mention scalability and provide details of the authors’ efforts to address it, supported by experimental evaluation, but the majority does not provide such comprehensive examination of the problem while others simply ignore the issue.

While examining how a composition approach scales, one should explore the main causes that have a negative effect on the efficiency of the approach and try to remove them. However, this should be done only after evaluating the significance of limiting the approach in that way. Achieving scalability but at the same time crippling the overall applicability of the approach is obviously unreasonable since the advantage offered by the former is
canceled by the effects of the latter. In any case, a complete composition approach should include a detailed account of the way efficiency is affected when the complexity of the composition problem is increased.

IV. RESEARCH PLAN

Based on the research challenges defined in the previous section, we have identified two major research goals that make up our research plan. The first goal is to define and formalize a novel specification language for Web services and service compositions. This language will take into account the frame, ramification and qualification problems and offer robust solutions based on its foundations, similarly to the way Kakas et al. [7] provided an action language that solves the aforementioned problems by the way it is defined. Moreover, this language should provide support for the derivation of composite specifications based on a composition schema.

The initial milestone we have set regarding this first goal is to explore the challenge of automatically deriving composite specifications, given the composition schema and the specifications of the services taking part in the composition. We are in the process of identifying the possible values for preconditions and postconditions in sequential and parallel compositions and calculating the weakest possible sets of them by drawing inspiration from the field of programming language specifications and the Craig-Lyndon interpolation theorem [8]. A second milestone involves identifying the cases in which the frame, ramification and qualification problems appear, to examine the effects it may have and to assess the severity of them. Then, using the insight gained by these previous steps, we will be able to define the aforementioned service specification language.

After the first goal is achieved we can move forward to the second research goal, which is the design and implementation of a dynamic, QoS-aware automated service composition framework. This framework will rely on the results of the first goal in the sense that it will expect services to be specified using the aforementioned specification language and will use the same language to describe the composite services that will be produced. A tentative road map for this goal is to begin with the introduction of QoS characteristics in the specification language by exploring the work of Kritikos and Plexousakis [9], which extends OWL for QoS-based Web Service Description and Discovery. A further milestone will be to select a successful planning technique such as planning as model checking and attempt to introduce dynamicity by applying run-time replanning methods.

V. CONCLUSIONS

In this paper, we identified two major problems in the field of service composition: the inability of current service description and composition frameworks to provide complete formal service specifications and the lack of an effective Web service composition approach that combines most desired requirements, such as QoS-awareness, dynamicity and scalability. The underlying research challenges were also analyzed and a tentative research plan to address them was outlined.

The importance of formal specifications for Web services and service compositions is crucial for the realization of the goals of Service-Oriented Architecture and should prove beneficial to both service providers, who will be able to better describe what they are offering and to service consumers, who will have a complete knowledge of what a service does under certain circumstances. On the other hand, realizing scalability, dynamicity and QoS-awareness in automated Web service composition may prove to be the necessary incentive for the industry to consider adopting such techniques for the composition of Web services. Of course, the eventual adoption of such an approach by the industry also depends on the availability of semantically-enabled Web services and the existence of effective discovery techniques for such services.

ACKNOWLEDGEMENT

The research leading to these results has received funding from the European Community’s Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

REFERENCES

Information about the Author(s)

Wolf Zimmermann (Editor)
Institut für Informatik
Universität Halle
06099 Halle, Germany
E-Mail: wolf.zimmermann@informatik.uni-halle.de
WWW: http://swt.informatik.uni-halle.de/mitarbeiter/zimmermann/