Contents

A Framework for Service Composition by Recommendations from Service Executions
Chaman Wijesiriwardana, Supervisor: Harald Gall 5

A Template Based Semi-Automatic Web Services Composition Framework Case Study of Smart Home Management
Abdaladhem Albreshene, Supervisor: Jacques Pasquier 11

An Active Negotiation Model for Service Selection in the Cloud
Mahboobeh Moghaddam, Supervisor: Joseph G. Davis 18

User-controlled data sovereignty in the Cloud
Marc Mosch, Supervisor: Alexander Schill 25

A Secure Cloud Gateway based upon XML and Web Services
Sebastian Graf, Supervisor: Marcel Waldvogel 30
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 five contributions. Each submission was reviewed by at least three 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 organizer Cesare Pautasso for his organizatorial support. We thank the international programme committee consisting of

Antonio Brogi (University of Pisa, Italy)
Birgitta König-Ries (University of Jena, Germany)
Thomas Gschwind (IBM Research Labs Zürich, Switzerland)
Kung-Kiu Lau (University of Manchester, U.K.)
Welf Löwe (Linneaus University, Sweden)
George Angelos Papadopoulos (University of Cyprus, Cyprus)

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, August 31, 2011
Abstract—A large number of solutions have been proposed in the literature for web service composition spanning from manual to automated approaches. However, the manual approaches encounter some substantial complexities and are error-prone since they require a deep understanding of the domain as well as composition languages. Automatic composition approaches still lack verification mechanisms to ensure the correctness of the composition for complex user requirements. In this paper, we propose a service composition framework, which recommends users suitable composite services based on previously executed services. These recommendations function in a problem domain and help users in new service scenarios. Hence, it is not necessary to create new composite services from scratch. This research will be carried out by splitting it into three work packages: (1) properties extraction from previously executed services; (2) model to represent the extracted properties; and (3) recommendations and tool support. We will validate our results with a large case study in the domain of software analysis services. We want to demonstrate that the knowledge from former service compositions can be captured and reused for new service compositions.

I. INTRODUCTION AND MOTIVATION

The manual composition of web services is difficult (i.e., all possible control and data flow variations has to be considered), inflexible (i.e., services are tightly connected to the composition) and is error-prone (i.e., it lacks of ways to verify the properties such as reachability and existence of deadlock). Usually it is challenging and requires a domain expert to specify the requirements of a composite service. In contrast, automatic and semi-automatic approaches provide different levels of automation in web services composition. The automated methods such as Golog [1] and rule based planning [2] are appropriate for generating implementation skeletons, which can be refined into a formal specification. A key problem of these approaches is the lack of correctness verification of the composite services. Hence, it leads to provide incorrect or at least unverified results to the user. Methods such as automata, event calculus [3] and petri-nets [4] could be used to verify the composition correctness.

To address the composition of services, we propose a solution that semi-automatically captures the domain knowledge of the experts from the executions (i.e., BPEL code and BPMN models) and reuses it when necessary to recommend composite services rather than building them from scratch. The domain knowledge in previously executed composite services cannot be fully leveraged because such knowledge is not properly captured since there is no adequate knowledge representation to leverage.

Therefore, how to systematically maintain that domain-specific service composition knowledge, and how to support its effective reuse in a new problem scenario is essential to users. So far, these aspects have been widely neglected in the literature.

In our work, we plan to address the following research questions (RQs):

RQ1 Can domain-specific service composition knowledge of previously executed services be effectively captured and reused in a framework for composite service recommendation?

RQ2 Can domain-specific ontologies make the composite services recommendation process more efficient and precise?

RQ3 Which property has the most impact on composite service recommendations among case-specific parameters, control flow, and data flow information?

Next, we present our proposal for service composition. Then we discuss related work in Section III, propose an evaluation scenario with a case study in Section IV, and discuss our conclusions of this research approach in Section V.

II. PROPOSED APPROACH

To address the problem we propose a framework, which allows reusing domain-specific knowledge of previously executed composite services to help service users in decision making by recommending most suitable composite services for a new problem.

As an illustrative example, consider a credit department of a bank that needs to compose several component services to approve a loan to a customer. Those services may include document processing service, customer assessing process, loan assessing service, loan approving service and so on. The customer assessing service may consist of sub services such as customer income assessing service and customer credit status assessing service. For the customers who have been maintaining long-time relation with the bank, it may not be necessary to invoke some particular customer assessment services. Therefore, when a customer makes a loan request,
appropriate existing services and data sources need to be properly lined up in a service composition that adequately covers such a complex service. Manual composition of a particular loan approval composite service with respect to the specified goals requires domain experts. It is a time consuming task for non-experts and they might generate wrong composition workflows, which could lead to significant problems (i.e., violations of desired behavior, creation of deadlocks). Therefore, the idea of this paper is to capture the knowledge of such previously executed service compositions that are created by domain experts to approve/reject loan requests. For a new loan request from another user, the system recommends the most suitable composite service to the credit department by reusing the previous loan approve/reject composition knowledge. The knowledge covers composition order, data and control flow, service levels, or synchronization and dependency information. The detailed properties to capture are part of our research project. In this research, we will use software analysis services composition example as a case study.

The research problems that are to be addressed to achieve this goal are classified in to three main work packages WP1, WP2 and WP3. In WP1, we plan to extract the properties from the previously executed composite services. In WP2, we intend to develop a domain-specific ontology and store the properties that are extracted in WP1 as cases in a case library. Finally, the WP3 consists of two main tasks, that are ‘query generation’ and ‘recommendation’. The ‘query generation’ takes care of generating queries from the natural language user requirements and the ‘recommendation’ phase recommends the most suitable composite services. The architecture of the proposed approach is shown in Figure 1.

A. WP1 – Properties extraction

The main objective of this work package is to capture the domain-specific service composition knowledge of previously executed services. Web service compositions can be described using behavioral models and BPEL descriptions. The behavioral models can be represented using a graphical notation such as business process model and notation (BPMN) diagrams or UML activity diagrams. WP1 of the proposed approach involves inspection of the BPMN diagrams and the BPEL codes to extract the properties from the previously executed composite services. The term ‘properties’ refers to the attributes of a considered composite service such as inputs, outputs, pre-conditions, post-conditions, user preferences, control flow information and data flow information. In other words, these ‘properties’ represent the domain-specific knowledge of the considered composite service.

Case-specific parameters - Properties such as inputs, outputs, pre-conditions, post-conditions and user preferences will be extracted from the BPMN diagram and the BPEL description of the composite service. Inputs are the information provided by the user to the system. Outputs represent the information returned by the composite service after the execution. Pre-conditions represent a state of the world that must hold to ensure the successful execution of a certain task. Post-conditions denote a state of the world that must hold after performing a task. User preferences or conditions denote properties of the anticipated result.

Control flow information - Control-flow patterns describe the flow of the control in systems. These patterns have been widely used by practitioners, vendors and academics in the selection, design and development of workflow systems. It was examined how the BPMN Business Process Diagram can be used to represent the workflow patterns [5]. Our intention is to
extract the control flow information from the BPMN diagram of the composite service.

**Data flow information** - Understanding the data flow among the component services is one of the key features in service composition problem. The data flow model of the composite service can be extracted by analyzing the BPEL description of the composite service and the WSDL descriptions of component services [6]. The data flow information provides a different composition view point, focusing on data interaction among several services. However, it is useful to understand the behavior of the main elements of the BPEL in order to extract the data flow information automatically. In general, a BPEL process is made up of activities. These activities can be categorized into two, namely basic activities and structured activities. Basic activities consist of receive, reply, invoke, assign and exit whereas structured activities consist of sequence, if, while, flow and pick.

**B. WP2 – Model representation**

During this stage, the goal is to prepare a representation model of the properties that are extracted in WP1. We plan to define a domain-specific ontology as a shared vocabulary and that can be used in case-based reasoning. This work is divided into two parts.

1) **Preparation of the case base:** Case-based reasoning (CBR), which consists of retrieving, reusing, revising and retaining cases, has been proved effective in retrieving information and knowledge from prior situations and widely researched and applied in a great variety of problem territories [7]. Characteristically a case consists of the problem that describes the state of the world when the case took place and the resulted solution to that problem. We intend to store the properties (case-specific parameters, control flow information and data flow information) that are extracted in the WP1 as cases in a case base. The structure of the case base is a major concern at this step. A CBR system provides both an overall similarity level for all the features and also a similarity level for selected features. Hence, we take advantage of CBR to evaluate RQ2 (i.e., which property has the most impact on composite service recommendation among case-specific parameters, control flow information and data flow information?).

2) **Domain specific ontology:** The objective of this phase is to model a domain-specific ontology as a shared vocabulary. The ontology stores an abstraction of all domain-specific concepts of the properties that are extracted in WP1. In this approach, the composite service recommendation is entirely based on case-specific parameters, control flow information and data flow information. Therefore, the ontology plays a vital role in the retrieval process and also provides a communication base among the extracted properties. The concept of an ontology system is the same as the case structure in a case base. These ontologies will be created with the help of domain experts.

**C. WP3 – Recommendation and tool support**

During this stage, the users are allowed to specify the requirements in natural language and the system recommends most suitable composite services. The work will be organized into two parts: query generation and recommendation.

1) **Query generation:** The goal of this step is to generate queries from the requirements specification in order to establish the interaction with the CBR system. We expect to introduce a natural language interface, which permits users to denote requirements. An ontology will also be introduced as the query vocabulary, which permits users to better express the requirements, hence resulting richer queries. The challenging part is to parse the textual queries and produce the structured queries to match with the case base and the domain ontology that we described in WP2. Initially, we will use an existing query generator such as jCOLIBRI natural language interface to generate the queries [8]. jCOLIBRI is an object-oriented framework written in Java for developing CBR systems. It allows one to deal with several case representation formalisms and incorporates Description Logics reasoning capabilities.

2) **Recommendation:** The objective of this step is to recommend the most suitable composite service (or services) to the user based on her requirements. The most suitable services are selected from the previously successfully executed services. The similarity between the new problem scenario and already existing composite services are calculated based on the new user requirements and the properties that are extracted in WP1. These requirements/properties can mainly be categorized as quantitative and qualitative requirements/properties. Therefore, it is essential to perform direct indexing on the qualitative requirements/properties and fuzzy indexing on quantitative requirements/properties.

The service recommendation process is described in Figure 2. The process starts with the query which is the result of the query generation as described previously. Then the system

![Fig. 2. Service Recommendation Process](image-url)
computes the most similar composite service (or services) from the case library and displays it to the user together with the expected output. If the system is unable to recommend suitable previously executed composite services, it allows the user to change the query. In case the user is not fully satisfied with the recommendations, she can improve the query to obtain more satisfactory results.

As the final step, we intend to develop a user-friendly tool which allows one to formulate the requirements and validate the queries. In addition, this tool should also support expert users by enabling them to create workflows through dragging and dropping of service components.

III. RELATED WORK

At present, there are some manual and automatic approaches for web services composition. Manual approaches refer to using several languages and frameworks such as Business Process Execution Language for Web Services (BPEL), Web Ontology Language for Web Services (OWL-S) and Web Service Modeling Ontology (WSMO) to predefine a composite process. Automatic and semi-automatic approaches include AI planning methods and graph-based methods [9]. However, both these approaches face some substantial composition difficulties [10], [9] and lacks of ways to verify the composition correctness. Hence, several verification approaches could be found in the literature [11], [4], [12]. In contrast, RESTful services composition is less investigated compared to the above mentioned SOAP/WSDL based composition approaches. A notable approach for RESTful services composition was presented by Autasso by introducing BPEL extension for REST [13]. That paper aims to enable the composition of both RESTful and SOAP/WSDL based web services and also describes how to publish a BPEL process as a RESTful web service.

To execute a BPMN model in a BPEL engine requires transformation of BPMN models to BPEL. These transformations are not straightforward and requires complex mappings [14]. However, it is out of the scope of this paper and we focus on extracting the properties from both BPEL codes and BPMN models. The authors of [15], present a mechanism to measure the similarity levels of available BPMN models for a given query model and to rank them accordingly. It needs the query to be presented as a model. In our approach, we plan to generate the queries from natural language requirements. Further, natural language processing for properties extraction from BPEL source code has not been much addressed in literature so far. However, the authors of [16], applied a natural language parsing techniques to information extraction from formally structured information sources, such as programs. They have applied it to Java, C/C++, and Python programs.

Several CBR approaches could be observed in literature for web service composition. Limthanmaphon and Zhang [17] suggested a method where the similarity assessments are based on keyword matching. Hence, the accuracy is low. However, most of these approaches are used in the discovery of component web services in composition. Semantic-based approaches [18], [19] are also widely used in web service discovery. In [20], authors presented a novel mechanism for selecting web services based on past user experiences. In contrast, our research is focused on selecting the composite services from a case library rather than discovering component services. The closest to our approach was presented by Chowdhury et al. [21]. Their proposal is to learn from existing compositions and to advice the developers by providing the learned knowledge. Further, they aim to benefit from reuse by simplifying the overall development process by providing an interactive advice mechanism to the developers.

IV. EVALUATION

The aim of the evaluation is to achieve deeper insights into the solution with a particular focus on concrete estimation of what can be expected from the proposed solution. Further, our evaluation is extended to find a practical combination of research and application issues for each mentioned step. To evaluate the proposed approach, we plan to use existing software analysis services that are developed in our research group. Software analysis services composition is beneficial for the software developers and architects in several ways. We realized the importance of having analyses composable without any language and platform limitations, and not having to install or configure particular tools. In addition, software analysis services have been used in several studies such as [22]. The existing services infrastructure in our research lab is rich enough to evaluate the proposed approach. Each of the work packages that are mentioned in Section II, will be evaluated for the composition problem shown in Figure 3.

A. Case Study: Software Analysis as a Service (SOFAS)

SOFAS is a distributed and collaborative software analysis platform developed in our research group to enable seamless interoperability of software analysis tools across platform, geographical and organizational boundaries [23], [24]. In SOFAS, the software analysis tools are represented as web services and that can be accessed over the web. In this case study, we plan to use the existing software analysis services and allow domain experts to create composite services to fulfill a certain software analysis task. An interactive user interface will be provided for the domain experts to create composite services at the design level by just dragging and dropping the component analysis services. The BPEL code will automatically be generated hence, the understanding of service composition languages such as BPEL or service description languages such as WSDL is not compulsory. The composite models and the BPEL descriptions are stored separately and later will be used to extract the properties and prepare the case library as described in WP1 and WP2 in our approach respectively.

The properties that are extracted represent many different types of knowledge and that can be stored in many different representational formats. In this particular case study, we plan to store them in textual format focusing on utilizing this textual knowledge in an automatic or semi-automatic way for supporting recommendation through case comparison.
However, a considerable time period is needed to evaluate the recommendation accuracy till the case library gets sufficiently populated.

Figure 3 shows an example composite service consists of five software analysis services. It includes version control service, source code model (FAMIX) service, change coupling service, generic source code metrics service and Java source code metrics service. The version control history service requires the URL of the repository, and the user name and password to access the repository as inputs. FAMIX model service takes the source code file as the input. Also, the user is allowed to specify the threshold value to filter the results according to the user preferences. Results aggregator takes care of analyzing the outputs of change coupling service and source code metrics services to find the design disharmonies in the system [25].

1) Properties Extraction: We plan to evaluate the quality of the extracted case-specific parameters, data flow information and control flow information separately.

Case-specific parameters will be evaluated with the aid of domain experts and a rating algorithm. The main idea is to allow domain experts to manually inspect the extracted case-specific parameters by our system and compare it with exact user requirements and rate them accordingly.

The control-flow correctness will be evaluated using formal methods such as automata or petri-nets. Another candidate would be first order logic based formalism such as event calculus to verify the control-flow correctness. We intend to compare our verification approach with the existing control-flow verification approaches such as the one proposed in [26].

However, the structural correctness of a composite service model does not guarantee that it has successfully captured all the composition requirements. Therefore, analyzing the model of how data are exchanged among the composed services to fulfill the user requirements is essential. Proper data manipulation is a vital factor in ensuring the quality of the service composition thus, data flow based analysis can provide constructive insights in verification and validation. Authors of [27] first mentioned the importance of data-flow verification in workflow processes. That paper recognized a number of possible errors in the data-flow, but no means for checking these errors is provided. We expect to follow the ideas presented in [7] for the validation purpose and this needs to be further studied.

2) Model Representation: We intend to evaluate the model based on its ability to represent the software analysis domain and the ability to represent different attribute types. Another important evaluation aspect is to check the ability of the model to deal with incomplete or missing data. We plan to broaden the evaluation to assess the easiness of managing the model in different problem domains.

Further, we will evaluate RQ2 about using a domain-specific ontology to make the searching process more efficient and precise. To evaluate the efficiency of the searching process, we use widely used information retrieval measures such as precision, recall and F-measure. In case those measures are not capable of capturing the necessary information in textual CBR system for the evaluation, we plan to use other measures. This needs to be further analyzed.

3) Recommendation and tool support: The main goal of this evaluation is to compare our solution with existing state of the art approaches with a particular emphasis on showing the benefits of our solution over the other composition approaches. Specifically, the evaluation will show

1) suitability of this approach for the non-expert users in software analysis domain
2) suitability of this approach for the users who are not familiar with composition languages (e.g. BPEL) as well as interface description languages such as WSDL
3) suitability of this approach in formulating the user requirements

Further, we extend the evaluation by using multi-fold evaluation criteria starting from technically-oriented criteria such as similarity assessment and knowledge representation. These criteria are used to describe the capabilities of a CBR system and, thus, allow the comparison of different systems. We continue the evaluation with domain and application oriented criteria such as theory strength and size. These criteria are used to describe the requirements that arise from a certain domain and application task and, thus, need to be satisfied if an application problem is to be solved. Finally, we intend to evaluate the recommendation accuracy in terms of different indexing schemas and retrieval methods to compare the impact of case-specific parameters, control flow information and data flow information in a composition problem.

In addition, we plan to evaluate the suitability of the jCOLIBRI natural language interface for the software analysis services domain by using the same evaluation criteria used in [8]. Finally, a user study will be conducted with both expert and non-expert users to evaluate the user acceptance of the tool.

V. CONCLUSION

We propose a framework to systematically capture the domain-specific knowledge of previously executed composite services and to effectively reuse it in a new problem scenario. This research will be carried out by splitting it into three
work packages: (1) properties extraction; (2) model representation; and (3) recommendations and tool support. First, we will investigate control flow and data flow information in recommendations together with case-specific parameters such as inputs, output, pre-conditions and post-conditions. Next, the representational model of the extracted features will be modeled by using an ontology to specify the domain. Finally, we will address query generation from natural language requirements and to recommend most suitable composite service workflows.

In addition to SOAP/WSDL based web services, RESTful web services are increasingly attracting the industry's attention. Our approach should be flexible to recommend composite RESTful web service either with WSDL 2.0 support or BPEL for REST extensions [13].

The major contribution of this research is to address three important research questions as stated in Section I. We plan to use SOFAS (the SOFware Analysis as a Service framework① developed in our research group) as a case study to address the research questions and for the evaluation. The three work packages will be evaluated separately for a software analysis services composition problem such as the one shown in Figure 3. We expect to show the following:

1) Domain-specific service composition knowledge of previously executed services can be effectively captured and reused.

2) Introducing domain-specific ontologies make the composite services recommendation process more efficient and precise.

The result of RQ3, i.e. the property, which has the most impact on composite service recommendations among case-specific parameters, control flow, and data flow information, will be a useful information for the research community, especially in correctness verification of web services composition.

ACKNOWLEDGMENT

The author gratefully acknowledges the financial support provided by the Swiss Federal Commission for Scholarships for Foreign Students (FCS), which allows him to pursue his PhD study at the University of Zurich.

The academic advisor of this work is Professor Harald Gall at the University of Zurich, Switzerland, seal.ifi.uzh.ch/gall.

REFERENCES


① seal.ifi.uzh.ch/sofas
A Template-Based Semi-Automatic Web Services Composition Framework
Case Study of Smart Home Management

Abdaladhem Albreshne
PhD Student
Computer Science Department
University of Fribourg, Switzerland
abdaladhem.albreshne@unifr.ch

Jacques Pasquier
PhD Supervisor
Computer Science Department
University of Fribourg, Switzerland
jacques.pasquier@unifr.ch

Abstract—Recently, a range of industry languages and frameworks solutions have been realized to enable web services composition. Among these, we find the Business Process Execution Language for Web Services (BPEL4WS) [1] and the Semantic Markup Language for Web Services (OWL-S) [2]. But there is still a need to propose a system which offers more flexibility to compose and invoke services and which reinforces the human-computer collaboration paradigm. This work aims to explore the challenges of the current web services composition solutions through a case study realized in the field of smart homes environment and to propose a new approach based on generic processes and semantic description in order to discover, compose and invoke services in a changing environment. In other words, it provides an assistance mechanism for the semi-automatic composition of services where the composition is gradually generated by using a declarative generic template process.

Keywords—Web Services; Web Services Composition; Ontology; Semantic Web; BPEL4WS; OWL-S; WSDL; Smart Homes

I. INTRODUCTION

Recently, a range of industry languages and frameworks solutions have been realized to enable web services composition. Among these, the Business Process Execution Language for Web Services (BPEL4WS) [1] is probably the most prominent. It provides a language for web services composition where the flow of processes and the bindings between services are known before the execution. Unfortunately, BPEL4WS does not fully support dynamic reconfiguration of a process, where for example a given service must be replaced by another one with the same functionalities, but a different binding type or a different set of methods signatures. This prevents the user from selecting another service afterwards, when many different services are available to provide similar functional components.

Semantic web services composition constitutes another aspect on which the Web community focuses. In order to be able to describe the services, semantic web languages like the Ontology Language for Web Services (OWL-S) [2], and the Web Services Modeling Ontology (WSMO) [3] have been proposed. They introduce an additional level of abstraction. Instead of a syntactic description of a web service, a declarative description of the service’s functionalities is given. Semantic services description protects the programmer from the complexity of the implementation and thus simplifies the completion of complex tasks using these services. Semantic could also add machine interpretable information to services’ content in order to provide intelligent access to distributed web services. We compare these two approaches and discuss their solutions and limitations regarding the problems of modeling, composing and executing Web services. Additionally, we discuss a new approach to overcome these limitations.

II. MOTIVATING EXAMPLE

A. Home Energy Saving Scenario

In this section, we explore the challenges of the current web services composition through a case study realized in the field of home environment. A smart home is a home equipped with diverse networked sensors such as temperature and movement detectors and devices such as alarms, heating systems, air-conditioners, doors/windows/lights controllers, etc. The interested reader is referred to [4] [5] [6] for a good introduction and a thorough discussion in smart homes themselves, for which an in-depth presentation is out of the scope of this paper.

Jane controls her home environment in a way that allows her to save energy and to adapt her environment to her habits and living conditions. Jane knows that a well-planned home energy control system can reduce the total energy consumption of the home. Now, let us imagine that Jane’s house is equipped with the following web enabled devices and sensors:

- Thermostats - heating/cooling can be adjusted according to the program (for example, to adjust to comfort settings upon arrival at home).
- Door/window controllers - used to open and close doors and windows.
- Temperature sensors - used to measure the internal and external temperature.
- Curtains controllers - used to open and close curtains.
- Movement detection sensor - used to detect objects’ movements.

To understand how these services can be orchestrated to save energy, we define a simplified business process for a home energy saving scenario (see Fig. 1). The process invokes sensor services asynchronously in order to wait for the incoming events. We assume that sensors have the capability to provide asynchronous service invocations. Each time the process receives an event, it starts to execute the necessary tasks in order to save energy depending on the new changing context. Then, the process is waiting for a new event to occur.

![Figure 1. Home Energy Saving Scenario](image)

**B. Web Services Composition Requirements and Challenges**

This scheme demonstrates the features necessary for a home energy saving process which are as follows:

- The process should offer **structured activities** (while, if, pick, flow, etc.), which allow for creating complex home control scenarios.
- Since the process is designed to control services and devices which are not known at design time, there is a need for a mechanism that allows for **discovering** services and for involving them into the process.
- Providing **semantic description** for the involved services helps users to configure the environments. In the home energy saving context, the system needs to understand and interpret the environment in order to take decisions about which services can be involved. Furthermore, it must enable interacting with the user in order to configure the process. Semantic description of involved services facilitates the service discovery and configuration by giving meaning to services parameters, inputs, outputs, operation names and non-functional attributes, such as service name, service type or service location.
- Processes should support **events messages** (an event represents the occurrence of something interesting). Event handlers allow processes to respond to the expiration of timers or to events by executing a specified set of operations independently from the rest of the process. Event handlers help a process to react to changing environments. In the home scenario, movement detection or a temperature change can happen at any moment. Providing an event handler mechanism makes the process more aware of the context.
- The final user should be capable of **interacting** with the system to reconfigure the process according to his needs. For example, in a home energy saving scenario, the system needs to know from the user what the interior comfort home temperature is when he is at home or what is the preferred temperature for each air conditioned room.
- The composition implementation should be hidden from the user. This might imply that an **abstract generic process template** is defined to achieve the composition goal. Raising the level of abstraction of composing web services facilitates normal user’s interaction with the system.
- Since there can be many instances of the same process running at the same time, **message correlation elements** or a similar mechanism need to provide a way to decide which process instance a specific message is sent for.
- Finally, **Asynchronous invocations** are extremely useful for home control environments in which a process, such as an energy saving one, must react to many events over a long period of time.

**III. FIRST SOLUTION: WSDL+BPEL4WS**

The role of BPEL4WS is to define a new web service by composing a set of existing services through a process-integration type mechanism with control language constructs. It interacts with external partner services through a Web Service Description Language Interface (WSDL) [7]. A BPEL4WS
process defines the order in which involved Web services are composed. BPEL4WS allows for describing conditional activities [1] [8]. An invocation of a Web service can for example rely on the result of another web service’s invocation. With BPEL4WS, it is possible to create loops, declare variables, copy and assign values as well as to use fault handlers. Additionally, BPEL4WS offers the possibility of asynchronous invocation and supports event messages.

A.  **BPEL4WS Characteristics**

In the following, we briefly describe the main elements of a BPEL process as defined in [1]:

1) **Partner Links Elements** define the interaction of participating services with the process.
2) **Structured Activities** are provided by the `<sequence>`, the `<while>`, the `<if>` ,the `<flow>` (for executing activities in parallel) and the `<pick>` constructs.
3) **Primitive Activities** are provided by the `<invoke>`, the `<receive>`, the `<reply>`, the `<assign>`, and `<throw>` constructs,
4) **Variable Elements** allow for declaring variables in order to receive, manipulate, and send data.
5) **Fault Handlers** determine the activity which the process has to perform when an error occurs.
6) **Correlation Sets** enable several processes to interact in stateful conversations.
7) **Event Handlers** allow the processes to respond to events.

B.  **Home Energy Saving Scenario Implementation Using BPEL4WS**

Let us solve the services composition problem of the energy saving scenario using the BPEL4WS solution. Fig. 2 shows an extract of the code of the process. Since the BPEL4WS process communicates with other Web services, it depends on the WSDL descriptions of the Web services invoked by the process. In the following, we briefly comment the code of Fig. 2:

- **Namespaces**: lines 1-20 define the target namespace to access the partners’ WSDL description files.
- **Import**: lines 21-37 import the WSDL files for all service providers and the process. Services providers must be known a priori.
- **Partner Links**: lines 38-54 specify the partner links which define different partners that interact with the BPEL process. Each partner link is related to a specific `partnerLinkType` that characterizes it.
- **Variables**: lines 55-82 declare the variables which are used to store reformat and transform messages (e.g. temperature, movement detection state, time).
- **Correlation Sets**: lines 83-86 declare the correlation sets which enable partners to interact with the process in stateful conversations. Because there can be many instances of the process running at the same time, message correlation elements provide a way to decide which process instance a specific message is sent for.

- **Process logic definition**: lines 87-259 specify the process main body which defines the order in which the partner Web services are invoked. The work flow structure is divided into two groups: events and services. An event represents the occurrence of a sensor while a service represents the interaction process in response to that event. The first group consists of a pick activity which executes only the first event handler fired. BPEL provides an `<onMessage>` element for event declaration. When an event is fired, the process then executes the second group of structures through different scenarios depending on the events’ received messages in order to save home energy. Structured and basic activities (invoke, flow, if, while, etc.), are implemented in order to define a complex scenario. Finally, the process is waiting through a while loop statement for a new event to occur.

![Figure 2. Code Extract of the Home Energy Saving BPEL Process](image)

IV.  **SECOND SOLUTION: SEMANTIC WEB (OWL-S)**

Most web services provide isolated functions and lack a real capacity to automatically collaborate among each other. To help solving this problem, there is a necessity to use semantic technology. Semantic and ontology facilitate interpreting web services by providing semantic awareness and services filtering capabilities.
Semantic Web Services aim to improve automatic services composition, service discovery and service invocation. In consequence, this assures interoperability and collaboration between different business processes and service partners. A number of solutions have been proposed by the software industry [9]. One of them is OWL-S. It is a framework based on the W3C OWL Web Ontology Language, proposed to help service requestors to search, discover, invoke, compose and monitor Web services. OWL-S allows for describing Web services’ features as well as for providing construct activities such as Sequence, Split, Split + Join, If-Then-Else, Repeat-While, and Repeat-Until. In contrast to BPEL4WS, OWL-S does not provide asynchronous invocation, fault and event handlers [8].

A. OWL-S Characteristics

OWL-S consists of three Models: Service Profile, Process Model and Grounding, used to represent different aspects of a service. The Service Profile Model describes the service features to other services or agents that want to use it. It defines the service with regards to its inputs, outputs, effects and precondition parameters. The Process Model is the essential model of the OWL-S architecture. It specifies how the process is used. Services can be composed using a combination of atomic, simple or composite services. Additionally, the Process model defines the order in which involved Web services are composed, either in sequence or in parallel. It allows for describing conditional activities. With the Process Model, it is possible to create loops and declare variables. Finally, the grounding Model defines how to interact with the service by providing the necessary concrete details related to the transport protocol and message format [2] [10]. In the following, we briefly describe the main elements of the OWL-S process as defined in [2]:

1) The Service Profile describes a service as a function of three basic types of information:
   a) the provider information, which consists mainly of the <serviceName> and the <textDescription> (a brief description of the service);
   b) the functional description, which consists of the <hasInput>, <hasOutput>, <hasPrecondition> and <hasResult> constructs;
   c) the properties description, which allows for the description of a host of properties that are used to describe features of the service for example its category.

2) The Process Model consists of the <composedOf>, the <Perform> (for invoking other processes), the <Sequence>, the <Split> (for parallel processing) and the <If-Then-Else> constructs.

3) The Grounding model refers to specific elements within the WSDL specification by using a set of constructs such as the <wsdlDocument>, <wsdlOperation>, <wsdlInput>, etc.

B. Home Energy Saving Scenario Implementation Using OWL-S

Let us now try to solve the service composition problem of the energy saving plan from a Semantic Web perspective. Fig. 3 shows an extract of the code of the process. It is briefly commented below:

- **Service Profile**: lines 20-64 define the service profile information as mentioned in the previous section.
- **Process Model**: Lines 64-237 specify the process main body which defines the order in which the partner Web services are invoked. The workflow structure starts gathering temperature and movement detection data. Then the process is executed through different scenarios depending on the received data in order to save energy. Finally, the process repeats the previous steps through a while loop statement.
- **Grounding**: lines 237-278 define the grounding model of the process which maps the elements of the WSDL and OWL-S documents. It provides necessary concrete details related to the transport protocol and messages format.

V. SOLUTIONS COMPARISON

BPEL4WS and OWL-S provide different solutions to compose web services. However, many concepts that are implemented in these two approaches are similar. Below is a high level summary comparing the features and characteristics of the two approaches:
- **Primitive and Structured Activities**: Both OWL-S and BPEL4WS allow for invoking, receiving messages or replying to external web services and partners. Structured activities are provided by BPEL4WS as well as OWL-S and they allow for executing services in sequence or in parallel in order to build complex scenarios. The While (BPEL4WS) and the Repeat-While (OWL-S) structures have similar functionalities. OWL-S offers Split+Join and BPEL4WS uses Flow activity for concurrency invocation [10].

- **Service Discovery**: The energy saving scenario shows the necessity of a mechanism which allows for discovering involved services. OWL-S can help to find services that match with a service request based on semantic service description [11]. BPEL4WS just enables composing explicit web services into composite processes without providing capabilities of semantic service discovery.

- **Semantic Description**: Home devices are isolated and provide concrete functions but they lack a real capacity to describe themselves. OWL-S describes the features of the service through its service profile model. The latter contains information such as the service’s name, inputs, outputs, enterprise contacts and category. By describing a service through its semantic features, the OWL-S approach helps web services discovery and composition. Conversely, BPEL4WS does not allow that. There are, however some propositions for increasing the semantic features of BPEL4WS [12].

- **Event Handler**: Events are essential to inform the process about any new change which may occur. The event handler is a BPEL4WS characteristic. OWL-S does not have yet a concept of an event handler. Thus, in our home saving scenario, we had to go around this limitation by using loops for checking permanently the temperature sensors and movement detection services.

- **User Involvement**: In the home energy saving scenario, it is a challenging problem to search and select the concrete services and involve users to adjust them in order to achieve their desired goals. Neither BPEL nor OWL-S does directly support human involvement since the user must tell the system what to do during the development of the composition process (i.e. before its execution).

- **Abstract Process Definition**: The end-users expect a high-level of abstraction in a service composition process. They are not able to use very technical and complex tools. Therefore, the composition implementation should be hidden from them. This implies that an abstract process should be defined to perform the composition goal. This can be achieved using semantic service description and defining generic process templates which contain the involved services as well as the interaction between the user and the system in order to adapt the process to the user’s needs. BPEL4WS only addresses the syntactical aspects of web services, which prevent involving semantic web service description in the process [13].

- **Message Correlation**: Message correlation is the BPEL feature which enables several processes to interact in stateful conversations. OWL-S does not provide such a feature.

- **Asynchronous Invocation**: OWL-S does not support asynchronous web services invocation and process persistence. This limits its ability to manage both atomic transactions as well as long-running business transactions. Conversely, BPEL4WS supports both asynchronous and synchronous invocation mechanisms.

- **Dynamic Composition**: Composition of web services using languages like BPEL or OWL-S is normally generated off-line. Modifying any part of a process may result in the reconfiguration and redeployment of the whole process. Currently, only BPEL supports (partially) fail-over and dynamical redesign [9] [14].

**VI. PROPOSING A COMBINED APPROACH**

To have a complete and efficient system for web services composition, there is a need to propose a semantic-based framework which offers flexibility to integrate services and reinforces the human-computer collaboration paradigm. We propose a template based composition technique to facilitate the discovery and the semi-automatic composition of web services. It is not up to the user to tell the system what to do but rather to establish and negotiate about goals and how to accomplish them. When possible, the system must involve the user in selecting the appropriate services and actions, which both respect the context (i.e. the available discovered services) and the initial objectives (i.e. the process generic template). This demands understanding the capabilities of those services as well as the conditions and requirements which must be met to accomplish the composition goal.

![Figure 4. Semi-automatic Service Composition Framework](image)

Fig. 4 gives an overview of our approach for semi-automatic services composition. The user can choose a recommended generic process template from the process
repository. The generic process template acts as a configurable module. It defines generic participating activities and the control flow. We are still investigating the best way (language) to describe such templates. Indeed, this question will be one of our major research focuses. It is possible that arbitration will be required between being totally open to all domains or being more adapted to specific tasks (e.g. smart homes).

The Process Generator component captures the generic activities’ characteristics in the process template and sends them to the Service Discovery Engine as services queries. Web services are usually published in registries (Discovery Engine). Consumers can request available services by a keyword-based search engine (e.g. expedia.com, google.com) or by looking it up in a web services registry (e.g. UDDI – Universal Description, Discovery and Integration Registry) [15]. Improving service discovery involves adding a semantic research mechanism [12]. The requestor can provide the inputs and outputs of the required service. Then the discovery process finds any service which matches these semantic requirements. In our framework, service discovery is based on the semantic description of services and of the domain ontology.

After the services have been discovered, the user interacts with the system in order to make the appropriate choices. When a service is put into the composition, the information about inputs, outputs, preconditions and effects (IOPE) is checked automatically to assure that all needed input data are provided; that all operations can be executed; and that all links are established. The process can now be converted into an executable process by the Process Generator. Finally, the Process Execution Engine executes the generated process using an execution language. We are still investigating the best solution for such an engine, but it will most probably be similar to BPEL4WS with the addition of some OWL-S features.

To validate our approach, we intend to develop such a prototypical semi-automatic composition framework in the domain of smart homes. We also intend to base the framework on pluggable generic components (the darkened elements in Fig. 4), which will allow to change its domain (e.g. hospital, university environments).

VII. RELATED WORK

Manual composition approach is generally used in the situation where the requestor has a well-defined process model. Processes are defined using a process execution language like BPEL or OWL-S. The problem with such an approach is that it demands too much knowledge on the part of the user and it becomes more and more difficult with the explosion of web services resources. Automatic composition (without human involvement) is used when the requestor has no process model but has a set of constraints and preferences. Several approaches for automatic service composition have been introduced, including solutions based on Hierarchical Task Network (HTN) [16], Goal Description Languages for Services Composition [17], Artificial Intelligence planning [18] or Rule-Based planning [19]. However, automatic composition is still viewed as a task of high complexity because of the rapid proliferation of available services to choose from and, the heterogeneity of the data formats they offer. Finally, the composition result risks to differ from the user’s original goal. The third approach (closed to our views) is called semi-automatic or interactive composition. In this kind of composition, the system usually helps users to find, filter, and integrate automatically the desired services by matching the users’ requests with the available services. Moreover, it enables end-users to intervene continuously during the composition process. Some efforts like OWL-S, METEOR-S [27] used semantic description of web services to aide in improving discovery and composition processes.

[11] [12] [20] [21] [26] provide mechanisms for semantic annotation of existing services, services discovery and arbitrary service composition. Others approaches have emerged to support specifically end-users in the service composition process. These approaches for example [22] [23] [24] [25] focus mainly on using techniques for involving users to compose services or inform the user about issues to be addressed in the current workflow. These approaches are applicable for some type of implementation. We argue that some problems still exist in current semantic services composition frameworks. In particular, there is a lack of a good generic process template language which involves generic services and user preferences to reach a specific goal.

CONCLUSION

We have explored a few emerging concepts in the area of web services composition, business process and workflow. A set of open, standards-based technologies like BPEL4WS and OWL-S are available for designing and executing interactions between numerous web services. Our home energy saving scenario demonstrates several challenges, which need to be addressed in order to build a flexible and generic web services composition framework. One problem is certainly the definition of a simple, yet flexible, language in order to describe generic process templates for various rich scenarios given specific domains ontologies, (e.g. energy saving scenario or maximum security scenario for a smart homes domain). Another interesting challenge consists in the creation of a process generator engine able to appropriately transform a generic scenario into an executable one with the help of both the end-user and of a powerful semantic discovery mechanism. Finally, the selection (or creation) of an execution engine, which will be able to react to a changing context, represents another non trivial task.

REFERENCES


An Active Negotiation Model for Service Selection in Web Service Composition

Mahboobeh Moghaddam\textsuperscript{1,2}
\textsuperscript{2}National ICT Australia (NICTA),
Australian Technology Park
Sydney, Australia
mahboobe@it.usyd.edu.au

\textsuperscript{1}School of Information Technologies,
University of Sydney
Sydney, Australia
joseph.davis@sydney.edu.au

Abstract— Web service composition (WSC) offers a range of solutions for rapid creation of complex applications in advanced service-oriented systems by facilitating the composition of already existing concrete web services. One of the critical challenges is the dynamic selection of concrete services to be bound to the abstract composite service. In our research, we identify and elaborate on the challenges involved in developing an automated negotiation solution for service selection. We propose an active negotiation model in order to more effectively benefit from the dynamic environment created by negotiation, through an active coordinator. The active coordinator prioritizes the atomic services based on a calibration of the risk of not achieving an agreement in negotiation, and starts the negotiation in ascending order of risk. This strategy enables the coordinator to improve the negotiation result of services associated with higher risk and thus achieves higher rates of negotiation success for the composite service.

Keywords— Web service composition; negotiation; service selection; service level agreement (SLA); quality of service (QoS)

I. INTRODUCTION

Web service composition (WSC) has been an active research area for the past ten years. Its main goal is to compose existing concrete web services together to achieve a new value-added service [1]. Manual composition of web services is time-consuming, error-prone and not scalable [2]. As a result, researchers have been attempting to automate different stages of this process to deal with the high level of complexity involved. In a typical automatic WSC solution lifecycle (Fig.1), the first stage is goal specification. At this stage, the service requester’s goal and preferences are defined. Then, the goal is decomposed into an abstract business process (BP) comprising a set of tasks, i.e. abstract atomic services or briefly atomic services, each with clear functionality, along with the control and data flow among them. The QoS requirements for the BP (end-to-end QoS requirements) as well as for each atomic service are also specified. During the next stage, service discovery, concrete web services that match the atomic services’ functional and non-functional requirements are located by searching a service registry that holds information about the concrete web services. The discovery is performed for all the atomic services in the BP to find a match for each of them. At this stage, it is very likely that more than one candidate web service will be found for each atomic service that, while satisfying the basic required functionality, are offered with different QoS attributes; i.e. different levels of availability, price, etc. Therefore, at the subsequent service selection stage, the service requester has to use complex optimization techniques to rank the candidates and select best concrete web service that matches the specified requirements of the atomic service. After finding the best matches for all the atomic services in the BP and binding the abstract atomic services to the concrete services, the concrete composite service (CS) is created. During service execution stage, a process instance is created by executing the CS. The process instance is continuously monitored for necessary responses toward any failure or change in its status at the final stage of WSC, i.e. service monitoring.

The implicit assumption in the current solutions for service discovery and selection is that the provider would publish the services with a predetermined set of attributes. Therefore, the requester has to search within this set of fixed attributes to find the services that match her requirements. In most cases, as there will be more than one candidate, the service requester has to use complex multi-objective optimization techniques for the trade-off analysis of the QoS attributes. On the one hand, this assumption forces the provider to fix all the service’s attributes based on pre-determined set of values that can be offered generally to all the different requesters. On the other hand, this assumption deprives the service requester of the option of receiving customized services based on her preferences.

To address these problems, this research aims to apply the concept of negotiation for Web service selection during WSC.
Negotiation is the process of searching a space of potential agreements to find a solution that satisfies all the negotiating parties’ requirements [3]. In our case, negotiation enables service requester to receive more customized services and frees the service providers from the obligation of offering general services with predetermined attributes that match every one.

The remainder of this report is as follows: Section 2 discusses the research problem in more detail. Different approaches in applying negotiation for service selection are explained in Section 3. The current passive coordination approach and the proposed active negotiation model are discussed in Section 4 and 5, respectively. Research challenges are presented in Section 6. Finally, Section 7 reports the conclusion and our research plan.

II. RESEARCH PROBLEM

In WSC, service discovery and selection is the process of finding concrete web services that match atomic services in the BP. Therefore, WSC solutions need to define clear search criteria to be executed against the web service registry. What to include in this search criteria varies among different solutions. The matchmaking algorithm proposed in [4] ranks the functionally equivalent services on the basis of their ability to fulfill the service requester’s requirements (functional and non-functional) while maintaining the price below a specified budget. The authors in [5] propose a matchmaking framework based on the context of Web services, defining context as all the information needed for enabling interactions between service requester and providers. Semantic and behavioral information are used in [6, 7] for web service matchmaking during WSC. Web service behavior is the order of the execution of the service operations or the order of message exchange with a service and the constraints governing the operations’ execution order [8]. The selection algorithm proposed in [8] takes into account not only the functional requirements but also the transactional properties, and QoS characteristics of web services; where transactional properties guarantee consistent outcome and correct execution of the composite web service. An information retrieval approach is suggested in [9] for discovering and ranking web services automatically, given a textual description of desired services.

This variation with respect to search criteria arises from the fact that a web service can be defined from different aspects and various perspectives, such as the functionality, QoS attributes, interface, semantics, behavior, and context. Although some aspects are compulsory elements in a web service definition, e.g. the web service interface, what other additional aspects to include in the service definition depends on the application domain and the service user’s request. Note that there is no standard distinct boundary between these aspects and how to define them varies among different researchers. For example [5] suggests that all information about a web service needed for enabling interactions between clients and providers can be defined as its contexts. Context in this definition even includes the functional attributes of a web service, besides the non-functional, domain-dependent, and value-added optional attributes, such as security and privacy. Indisputably, all existing solutions seem to concur that the interface is necessary but not sufficient. Whatever criteria are considered in the search query in order to find the concrete matches of each atomic service, the service registry returns a number of candidate matches after executing the query. Then, the service requester has to use complex optimization techniques to select the one service that best matches the requester’s requirements.

The current solutions for service discovery and selection implicitly assume that the service providers would publish the concrete web services with a predetermined set of fixed attributes in a very strict manner of take-it or leave-it. However, there are a number of serious problems with this approach: i) in many cases, the providers are not able to specify all the service’s attributes based on pre-determined values that can be offered generally to all the different requesters, and ii) no mechanism is available by which the requester could receive a customized service based on her preferences. This problem has been referred to as defining the QoS profile in a static non-negotiable, non-configurable manner which is the current trend in process optimization and Web service selection literature [4].

This approach is very different from real world services. Consider a scenario when someone is looking for a specific service, for example a comprehensive car insurance policy for her car. First she searches for the companies who are offering this type of services in a registry, i.e. the city yellow page book. Up to this point, it is very similar to the web service world where the requester tries to find the functional matches for her desired functionality. Afterwards, in the real world, the service requester starts a series of negotiations with different service providers. She may use the information obtained during her first round of negotiation with the provider A, in negotiation with provider B. For example, while negotiating with B, she may claim to have a lower price from A, but as her Third-party insurance is with B, she prefers to buy her comprehensive car insurance policy from B too, only if they can lower their price. Moreover, the provider can customize the offer according to the requesters’ context; i.e. as the service requester is over 30 years with no previous accident history, she can benefit from a special discount.

In computer science, negotiation as a distributed search through a space of potential agreements[10], has been used for many years to solve different problems, e.g. resource allocation in grid computing, or getting agents to cooperate or compete over a common goal in multi-agent systems. In the web service domain, researches have employed negotiation mainly for (semi-)automatic creation of Service Level Agreement(SLA) where service consumer and provider negotiate over QoS attributes to reach an agreement, that satisfies both sides’ requirements.

In the web service composition area, employing negotiation becomes much more interesting by considering the fact that negotiating to find a match for one atomic service is part of a series of negotiations to find matches for other atomic services while they are not independent from each other. For example if the discovery process for one specific atomic service returns only one concrete service, called S1, with a specific response time, it is useless to match the services parallel with S1 in the process, with concrete services which have much smaller
response times, as anyway, SL cannot respond faster than a specific time. Therefore, it is useful to consider this information during negotiation for other services in order to not compromise another important QoS attribute for the response time.

To address these problems, this research aims to develop a novel framework that provides an effective automated negotiation solution for dynamic selection of concrete web services during WSC. More precisely, the intended framework will provide the service requester with an automated negotiation mechanism to negotiate over the desired QoS attributes and preferences, with different service providers for each atomic service. This can eventually lead to effective selection of concrete web services for the BP. Negotiation will enable the service requester to receive customized services based on her specific context and it frees the service providers from the obligation of defining their offered services with a predetermined set of fixed attributes in order to publish them in a registry. More specifically this research seeks to answer the following questions:

1) What are the specific requirements of WSC that distinguish negotiation in this context from negotiation in other close research areas such as grid computing or multi-agent systems, 2) What are the necessary elements for building an automated negotiation solution in the context of service selection during WSC, considering the specific requirements of this context, 3) What are the necessary modifications to the current negotiation protocols and decision making models to effectively address the specific requirements of negotiation in the above mentioned context, 4) How to exploit the specific characteristic of negotiation in the intended context to develop more effective and efficient negotiation techniques (e.g. using different types of dependencies among atomic services in the BP to improve the negotiation process).

A. Progress to date

In our research, we have identified and elaborated on the challenges involved in applying negotiation for service selection during web service composition, including the need for coordination, and the critical design factors associated with it: time limitation, management complexity, and effective use of negotiation information. To address the specific requirements of negotiation in this context, we have extended the negotiation framework proposed in [3] for autonomous agents capable of negotiation, by adding negotiation architecture as an additional element. We have carefully reviewed and analyzed the current approaches toward designing the coordination layer and how they have addressed the critical design factors. Based on this, we argue that the current coordinators take a passive approach toward negotiation; they neither efficiently consider the negotiation space before starting the negotiation, nor do they take advantage of all the available information to improve the negotiation success rate in achieving the agreement for the whole BP. To address these problems, we have proposed an active coordinator which calibrates the risk associated with not achieving the agreement for each abstract atomic service in the BP and prioritizes the atomic service’s negotiation initiation according to the risk. It initiates the negotiation of those with lower risk before the ones with higher risk. This enables the active coordinator to effectively utilize the negotiation result of lower risk services to improve the negotiation of those associated with higher risk and thus totally achieve higher rate of negotiation success for the composite service.

III. NEGOTIATION IN WSC

There are currently two approaches related to how and when to use negotiation for service selection during WSC lifecycle (Fig. 2). a) negotiation as a complementary strategy to optimization techniques for service selection, b) negotiation as the main strategy for service selection. In the first approach, negotiation takes place after the optimization technique, only when it fails to find appropriate concrete services for all the atomic services, due to their not satisfying the end-to-end QoS requirement of the BP[11]. Even though this approach is an improvement on the purely optimization approach, it still faces the problems mentioned in the foregoing, namely, considering only a static set of QoS values for web services and the lack of customization. In the second approach [12-15], negotiation is used as the main technique for service selection. Negotiation makes it possible to dynamically select concrete services for atomic services without the need to statically define a QoS profile for either atomic services or concrete services. Our focus is on the second approach, and whenever we discuss negotiation in the context of WSC, the second approach is intended.

![Figure 2. Different perspectives on how to apply negotiation for service selection during WSC](image)

Basically in this approach, a high-level negotiation process (overall negotiation process) is conceptualized that negotiates for the overall BP. It consists of multiple negotiation subprocesses (briefly negotiation process) each associated with one abstract atomic service in the BP. Each negotiation process in turn, may include multiple negotiation threads to choose the best provider for the specific atomic service that this negotiation process is associated with. Meanwhile, when dealing with negotiation at the BP level, negotiation itself may not be enough for achieving the end-to-end QoS requirement and ensuring a successful overall negotiation outcome for the composite service. Thus, a further management layer referred to in the literature as coordination becomes necessary.

Three important factors affect the design of such a coordination layer: time limitation, management complexity, and effective use of negotiation information. Time limitation is caused by the fact that service selection is one stage in the overall WSC lifecycle, and the time to accomplish it is limited. When applying negotiation as the selection technique, the overall negotiation time depends on the individual negotiation processes’ time. In each negotiation process, all negotiation threads take place concurrently. The negotiation process will
finish when an agreement is formed between the atomic service and one of the available providers or negotiation deadline expires without reaching an agreement. Therefore, the minimum overall negotiation time is achievable if all of the concurrent negotiation processes successfully terminate.

However, the concurrent execution of negotiation processes makes their management exponentially complex, because, a negotiation process which tends to find a match for one atomic service is not independent from other negotiation processes. In fact, if the opportunities brought by negotiation for dynamically specifying the QoS values are about to be used, the coordinator should be able to effectively use one individual negotiation process’ status and final result, in order to improve its dependant negotiation processes status and final result. This can lead to improvement in the overall negotiation performance in terms of reaching an agreement for the whole BP. Nevertheless, how each negotiation process can improve others’ situation in negotiation is determined by the dependencies that exist among the atomic services in the BP. There exist different types of service dependencies in a BP, e.g. regarding the timing, QoS attributes, budget, data flow, or control flow[16]. If a coordinator decides to consider more than one type of dependencies among atomic services, this will further exacerbate the complexity of managing the overall negotiation process with concurrent sub-processes.

IV. PASSIVE COORDINATION

We argue that the current proposals for coordination [13, 14, 17] take a relatively passive approach toward negotiation. The passive approach forces the coordinator to wait till the conclusion of all the negotiation processes to decide upon the necessary action. Then, even if one negotiation process is not successful, the service selection will not be successful. Then, the coordinator should restart negotiation of one or more atomic services, with some new values for negotiation issues. This means that the coordinator does not benefit from any individual negotiation process outcome to improve the situation of other negotiation processes. In [15] however, the coordinator does not wait for all the processes to finish. It receives the result of each negotiation process at its end and acts upon receiving the result.

The following assumptions structure the passive coordinator approach in the current literature on negotiation: a)initial decomposition of the end-to-end QoS attributes takes place regardless of the negotiation space, b)simultaneous initiation of all negotiation processes, independent of the negotiation space, and with no feedback for each other. As a result, the coordinator does not benefit from the information that emerges during the negotiation.

Although there is not much coverage in the literature on the overall agreement rate of such a passive coordinator, we argue that these assumptions make the coordinator’s success rate in achieving an overall agreement for the whole composite service reasonably low, especially if the QoS requirements are very severe. The passive coordinator might have to change the QoS requirements distribution over the atomic services several times to ultimately achieve overall agreement.

V. AN ACTIVE NEGOTIATION MODEL

In order to address the above mentioned problems of current passive coordinators, we propose an active negotiation model. This model follows an active coordination strategy which allows the coordinator to effectively intervene in the negotiation processes in order to increase the agreement rate of the overall negotiation. The ideal case of such an active coordinator is aware of the status of all the concurrent negotiation processes and threads. Thus, it does not just wait to receive the negotiation results to coordinate them, rather it may intervene in the negotiation processes, and can do this in any of them if it assesses the negotiation process to be unsatisfactory (e.g. close to negotiation deadline, or to withdraw without agreement). By getting the status of the concurrent negotiation processes, it may use the information of one process to improve the negotiation in another process. For example if a negotiation process for an atomic service finishes successfully with a final price below the expected budget, the active coordinator is able to assign the remaining budget to another atomic service which negotiation is about to fail due to dissension over price. However, the ideal active coordinator has serious problems in terms of the exponential computational complexity of managing the concurrent negotiation processes and threads with a number of different types of dependencies, as discussed earlier. Management here refers to performing the necessary actions to adjust the overall negotiation process to increase its success probability.

We propose an active coordinator which overcomes the complexity problem by initiating negotiation processes based on their priority. The priority class is decided based on the risk associated with the atomic service for not achieving an agreement. We calibrate such a risk using two factors: the number of potential providers (the lower the number of potential providers for negotiation, the higher the risk of negotiation failure) and the weight associated with it for its importance in the BP in the general term (the more important the atomic service, the more critical its negotiation success). The details of the calibration algorithm are not discussed here due to space limitation. Note that the negotiation space (i.e. the number of potential providers available for negotiation for each individual atomic service, how close is the acceptable interval of a negotiation issue of the atomic service to that of any of the providers, the negotiation deadline of atomic service with regard to those of the providers) and the QoS requirements space (i.e. the severity of limitations over the QoS requirements) affect the result of the overall negotiation process and how successful it can be. However, the only piece of information accessible for coordinator is the number of potential providers (output of the service discovery stage) which is used in our active negotiation model.

A. Risk Calibration and Class Determination

We assume that the end-to-end QoS requirements is decomposed over the individual atomic services in the BP either manually or automatically through one of the already existing techniques (Reverse Stochastic Workflow Reduction [20], utility decomposition [8]) and that the potential service providers are identified through a service discovery technique searching in a service registry for semantic and syntactic matches for each atomic service.
Atomic services are once categorized based on the number of potential providers (the two right-hand columns of the Risk Calibration table, in Fig. 3) and another time based on their weight in the BP (the two left-hand columns of the same table). Each categorization will place an atomic service in one of the categories of “low, medium or high number of providers available” and one of the “low, medium, or high importance”. Afterwards, based on the decision table depicted in table 1, atomic services will be classified into three categories: high, medium and low risk atomic services. Each risk class outlines one prioritized class of atomic services.

<table>
<thead>
<tr>
<th>Importance category</th>
<th>Available providers</th>
<th>Low</th>
<th>Medium</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>MR</td>
<td>LR</td>
<td>LR</td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>HR</td>
<td>MR</td>
<td>LR</td>
<td></td>
</tr>
<tr>
<td>High</td>
<td>HR</td>
<td>HR</td>
<td>MR</td>
<td></td>
</tr>
</tbody>
</table>

For example, consider a business process comprising of five atomic services (S1 to S5), each with an identified number of available providers (\(P_{j,k}\) provider available for atomic service \(i\)) and a normalized weigh (\(w_i\) to \(w_5\)). Calibration of the risk of each atomic service and assigning it to a class of priority is depicted in Fig 3.

The classes of priority can be more or less than our suggested three classes. Increasing the number of classes allows the coordinator to more effectively intervene in the negotiation result, while increasing the total time of negotiation for BP. Therefore, based on the number of the atomic services in a composite service and the total acceptable negotiation time, it is possible to raise the number of the classes in order to increase the success rate of the overall negotiation process. As we do not want to augment the total negotiation time dramatically from the current passive approaches, and based on an assumption that only two classes of high and low risk will not give enough opportunity to the active coordinator to exploit the negotiation result for necessary adjustments, we chose three classes for our model.

When the risk level of the atomic service is identified, its negotiation will start concurrently with the other members of the same class of priority. The active coordinator starts the negotiation processes of those in low risk class first, and after they all ended, it starts the medium risk class, followed by the high risk class. At the end of negotiation of each class, the coordinator analyses the result to determine the possible actions for improving the negotiation result of the next priority class.

This prioritized gradual initiation from lower to the higher risk enables the coordinator to effectively utilize the negotiation processes’ results of those which are more likely to have a successful negotiation, to decrease the probability of failure for high risk services. Furthermore, coordinator would be able to assign more resources (e.g. the extra budget resulting from low class negotiation) to the more important (and thus critical) atomic services. This will ultimately improve the overall negotiation process agreement rate, compared to the current passive negotiation models. Active coordinator maintains the complexity of its management as low as the passive coordinators since the strategy toward each class of risk is still passive; i.e. the coordinator does not intervene in the negotiation processes associated with atomic services which are members of the same class of priority. A comparison of the proposed active coordinator with the ideal coordinator and the current passive coordinators is depicted in Fig. 4, along the three dimensions of critical design factors of a coordinator.

<table>
<thead>
<tr>
<th>Atomic service</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Risk Class</td>
<td>MR</td>
<td>HR</td>
<td>LR</td>
<td>LR</td>
<td>MR</td>
</tr>
</tbody>
</table>

VI. RESEARCH CHALLENGES

To develop an automated solution that applies negotiation as the service selection technique during WSC, we face a number of interesting challenges while designing the solution and also its evaluation. The first challenge is related to the
negotiation and its complexity. Negotiation is a very complex process in the general sense, borrowing from different domains such as economics, social science, psychology, artificial intelligence, game-theory, and multi-agent systems. A careful selection of the abstraction level and the boundary of the research problem is a critical success factor affecting the accomplishment of this PhD study. The other design challenge lies in the complexity of effective negotiation for a composite service which involves multiple one-to-many negotiation processes over multi-attribute negotiation objects where the negotiation processes are not independent. We need an automated negotiation technique that helps us manage the complexity without losing its effectiveness in exploiting the information emerging from the individual negotiation processes. This valuable information can be used to improve the overall negotiation process for the BP.

Another challenge involved in this research is associated with the evaluation criteria for the proposed solution. In the research community, it is very common to evaluate the success of the proposed solution by measuring its efficiency or performance (i.e. response time, and latency). However, we think these metrics are not good enough for the context of our research. In fact, it is very probable that applying negotiation for service selection makes the solution slower than the current approaches without negotiation. But we believe it would be much more effective in terms of the ultimate goal of WSC which is addressing the user functional and non-functional requirements including price, performance, response time, etc. Yet, we have to define the necessary metrics for such effectiveness.

VII. CONCLUSION AND RESEARCH PLAN

In this research, we elaborate on the challenges involved in developing an automated negotiation solution for service selection, including: the need for coordinating the negotiation results, the complexity of concurrent negotiation processes, and the efficient use of emergent information from the negotiation.

We have identified the two strategies for the coordination, namely active and passive and discussed the advantages and drawbacks of each of them. To exploit the advantages of the active coordinator without its complexity problem, we have proposed an active coordinator which takes a divide-and-conquer approach to solve the management complexity. It prioritizes the atomic services based on the risk associated with them for not achieving an agreement and starts the negotiation process in an ascending order of risk. This will enable coordinator to consider the necessary actions (i.e. distributing the extra budget, extra time, etc. over other dependant negotiation processes) to increase the possibility of achieving agreement for those atomic services associated with higher risks which in turn will lead to higher rate of achieving agreement for the whole composite service.

Our future work includes a more detailed study on the influential factors over the risk associated with unsuccessful negotiation to apply them in our model. We also intend to improve coordinator performance in terms of the agreement rate by considering more complex analysis of the current negotiation status and other possible reactions. We are in the process of developing a prototype system to evaluate our proposed architecture and demonstrate its overall functionality in providing an effective solution for the dynamic web service selection. The prototype system first will be developed as a limited-version of the proposed active negotiation model. This prototype will allow us to conduct experiments for measuring the defined evaluation metrics and compare the model with the current existing approaches. Some of the evaluation metrics are adopted from the literature [3],[18], [19]. But as discussed earlier in section 6, there is a need for other evaluation metrics that better reflect the effectiveness of the solution in better addressing user requirements. Early results are expected to be published concurrently. At the next stage, we propose to extend the prototype based on the early results of the experiment.

This prototype system will enable us to conduct experiment to evaluate our hypotheses. The central hypothesis is that our proposed active negotiation model will perform at least well as the passive proposals in worst case, while in the other cases it will outperform in terms of the overall negotiation agreement rate.

VIII. REFERENCES


User-controlled data sovereignty in the Cloud

Marc Mosch
Faculty of Computer Science Faculty of Computer Science
Technische Universität Dresden
01062 Dresden, Germany
Email: marc.mosch@tu-dresden.de
Supervisor: Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill (alexander.schill@tu-dresden.de)

Abstract—Cloud computing plays an increasing role in the IT-market. Promised scalability and flexibility attracts enterprises as well as private end users. The loss of data sovereignty is seldom addressed but nevertheless a serious threat to users. The FlexCloud\textsuperscript{1} research group proposes an approach that enables users to profit from public cloud scalability without losing data sovereignty. The proposed dissertation derives from the work within the research group. The goal of the PhD thesis is an architecture design of a gateway for the secure outsourcing of sensitive data.

Keywords—Cloud Computing; Gateway; Storage; Data Security; Privacy; Data Sovereignty

I. INTRODUCTION AND MOTIVATION

In this paper I introduce my PhD intention and propose a way to benefit from cloud computing advantages without suffering from the drawbacks. In this context the term "data sovereignty" in the title refers to the user's self-determined control over his or her data. It means to be sure that important data—once outsourced to distant servers—is still only under user's control and secure from manipulation by third parties. So the meaning of sovereignty is not to be mistaken as sovereignty in terms of geolocating like it is for example understood in [1]. Such territorial assumptions are not of importance for the presented approach.

After the motivation and an introduction of the Personal Secure Cloud (\(\pi\)-Cloud) a scenario shows how the ability to benefit from cloud's scalability without losing the data sovereignty enriches daily life. Afterwards the concept of my thesis points out problems to be solved, lists the extracted research questions and my solution approach. Then related work is discussed followed by a short summary and outlook.

What is cloud computing? There is no unique definition that everybody agrees on. A lot of definitions for example the one by NIST [2] are complex. Cloud computing is nothing completely new. It is a combination of long existing technologies. In short, cloud computing can be understood as a distributed service approach which involves virtualisation of physical servers and their rental, or the rental of services running on them. Because the focus of the thesis is on data security, only storage services are taken into consideration. A distinction can be drawn between four different kinds of clouds: private clouds, where the user is the owner of the infrastructure and in control of the cloud; community clouds, a kind of merged private clouds where the infrastructure is shared among several users with shared concerns; public clouds, which are not owned by the user and which are open to paying customers; and hybrid clouds, which are a combination of the stated types of clouds. End users as well as companies are attracted by the fancy term "Cloud" and the marketing promises associated with public clouds—unlimited scalability and availability. Companies at the management level see cost savings that arise from outsourcing bulky high-maintenance server farms. For end users cloud computing promises a unified database—reachable any time from any place—without the need to manually synchronise smart phone, notebook, tablet, personal computer and other devices. As a welcome side-effect backup-strategies are a matter of cloud providers—no need to care about. What most end users and companies disregard is the high price they pay for the comfort of public cloud services—they pay with the loss of their data sovereignty. Sensitive data once outsourced—whether stored or processed by public cloud services—are exposed to loss, abuse and manipulation. To benefit from public cloud approaches without losing the data sovereignty an open source solution with transparent client side encryption for confidential data has to be established. This is what the FlexCloud project is focused on.

II. INTRODUCING THE \(\pi\)-CLOUD IDEA

A central mediating instance—hereinafter referred to as \(\pi\)-Box (Personal Secure Box)—should encapsulate the variety of devices and thereby form a \(\pi\)-Cloud (Personal Secure Cloud) as it is illustrated in Fig. 1. The \(\pi\)-Box consists of a service platform based on SPACE [3] and the \(\pi\)-Gateway. SPACE is a service broker architecture that features propagation, booking and deployment of services as well as Service Level Agreement (SLA) negotiation.

The communication to the outside is coordinated by the \(\pi\)-Gateway in order to protect user's privacy. This is done by distinguishing between private (sensitive) and public (non-sensitive) data and encrypting the former one in a way that only authorized users are able to access

\footnote{\textsuperscript{1}FlexCloud (Flexible Service Architectures for cloud computing): http://flexcloud.eu/. This work has received from the European Social Fund and the Free State of Saxony, Germany, under project number 080949277}
it. The architecture should be designed to fit end user requirements as well as industry needs.

The unified database for the devices could be placed on a home server or an external hard drive connected to the Wireless-LAN-router or a micro- or a plug-server at home. It might actually be part of a virtual machine hosted by a cloud storage provider whom the user trusts. So might be parts of the π-Box. These parts can also be distributed over the devices within the π-Cloud. For example the Data Module—a data analysis module that defines which data might be sensitive and which might be not—could be integrated in every device to enable self-sustaining file exchange.

III. Scenario

The following “smart home” scenario gives a brief overview about what the π-Cloud is aimed at and which benefits are to be expected in daily life. Imagine a household in the near future. Devices are “intelligent” to ease user’s way of living. Therefore they are connected with each other. They form a π-Cloud—mediated by the π-Box. Devices involved in the processes of generating, collecting, storing and editing data like computers, smart phones, printers and notebooks are part of the π-Cloud. So could be household infrastructure like heating, ventilation, washing machines, microwaves and fridges. The latter recognize their content for example by RFID tags. Let us assume there is only one bottle of milk left in a fridge. The number of bottles—for example determined by the number of associated RFID tags—reaches a predefined threshold. This fact triggers the fridge to alter the user’s digital shopping list, which is part of a calendar file. The file is stored in the user’s unified database—reachable via all types of authorized devices. Since there is only one bottle of milk left, the priority for the purchase is set to a high level and is therefore entered for the same day. The examination of the day’s appointments reveals that there is no time left for shopping. It is time for a contingency plan: the status of the milk in the list is set to “delivery” and a query for a delivery service is placed against the service broker module of the π-Box. This module holds two lists—one for services provided within the user’s domain and another for foreign services, provided for example by other π-Clouds or public clouds.

Figure 1. This is an initial architectural layout of the π-Cloud with a coarse grained subdivision of the π-Box into the Service Platform SPACE and the π-Gateway. The latter connects it with other π-Clouds or public clouds.
encrypted photographs and the encrypted decryption key to his friend’s smart phone for example via Bluetooth. If there are no buffered files in the smart phone the user might also hand out links to the files in his unified database to the friend. After receiving the files the friend can decrypt the decryption key with his private key and then decrypt and access the photographs. To enhance the ease of use, this process might be kept transparent by automation.

IV. CONCEPT

The dissertation is derived from the work-package of the FlexCloud project which is dedicated to the transparent service deployment. Within the dissertation the basic architecture of the π-Gateway should be defined. Furthermore a concept shall be created to distribute it and its components among several instances in a redundant and secure way.

A. Problems

The scenario shows applications for a system that features a unified protected database and helps the user sharing data with other persons in a secure manner. In this scenario several problems arise.

The way data are published today lever the shielding effect of common firewalls. Data is stored on distant servers where the user is not able to control for example who gains (physical) access. The outsourcing of the data results in a highly dynamic environment that involves the need for a replacement of common firewalls. This is where the π-Box—especially the π-Gateway—should be applied as a cloud management or cloud control mechanism. The π-Cloud resulting from the encapsulation of the user’s devices is a personal domain, a kind of safe harbour for sensitive data in hostile cloud environments. To be competitive with common cloud solutions data in the π-Cloud have to be available everywhere independent from the device. In order to protect user’s privacy it must be possible to distinguish sensitive from non-sensitive data on the go and share these decisions over several devices to avoid overload by duplication of efforts.

Another problem arises from the fact that in such a highly dynamic environment 100%-connectivity cannot be guaranteed. Imagine the depicted friend from the scenario asked for the photographs before they could be synchronized with the unified database. Then—if the smart phone is not performant enough—no face recognition and no encryption could be done. In this case the smart phone could not hand out encrypted files. This shows the need to provide links in advance in case the synchronization mechanism is delayed and manual user intervention to grant access is undesired.

So the main requirements the π-Gateway has to meet are the encapsulation of one’s devices, the management of a unified database, the ability to distinguish between sensitive and non-sensitive data and the possibility to handle offline situations of single devices.

B. Research Questions

The following research questions arise from the presented scenario and my work at the FlexCloud research group. They result from the need to create a π-Gateway that serves as a kind of firewall in a highly dynamic environment where common firewalls are useless.

1) How does the basic architecture of the π-Gateway have to look like? Should it be a central instance that manages all the other devices in a star topology or should it be a kind of virtual machine or even an application that can be migrated from one device to another?

2) Which criteria are important for the election of the device that serves as π-Gateway/π-Box?

3) Which are the basic functions of the π-Gateway that every device has to provide to host it?

4) How can the encryption and access decisions taken be shared among the devices in order to avoid overload by duplication of efforts?

C. Solution Approaches

The following ordered solution approaches address the research questions and are labelled accordingly.

1) The design of the π-Gateway should be based on a requirement catalogue that has to be developed first. This design has to address the distribution of the π-Box or parts of it. The Data Module—that part of the π-Gateway that distinguishes sensitive from non-sensitive data—might for example either be part of a central π-Box or existent in every device permanently or temporarily by migration. An initial layout of this module can be seen in Fig. 2. If it is distributed over the devices it has to be ensured that the decision base is shared among the devices. So if this information is not solely stored in the π-Box a suitable distribution and synchronization mechanism is needed to share the information, how to treat which data in a consistent and efficient way.

2) The strategies for the election of the device that serves as π-Box have to be developed with the dynamic needs of a mobile environment in mind. If the user leaves for example the house and takes only a smart phone along to a location without network coverage, the π-Box/π-Gateway has to migrate to the phone.

3) A criteria catalogue that has to be created should help defining the basic functions of the π-Box/π-Gateway that every device has to provide to host the π-Gateway. Depending on the device’s performance it might be necessary to outsource some functions to a more powerful device that is always available—like for example a broadband Internet gateway or a trusted cloud service.

4) Overload by duplication of efforts might be prevented if the decisions taken are shared among the devices. This means every device has to synchronize its decisions with the other devices if a connection is established. In times without a connection this
approach ensures that buffered files can be shared with other persons based on access rights already granted before. Thus a potential time-consuming analysis of the files does not have to be repeated on a low performance device like a smart phone. If the synchronisation is delayed and the current device is not able to encrypt the requested files it may nevertheless hand over links that are valid in the future. The files could be identified by metadata like timestamps. The prediction of future link-paths could be achieved by a database which associates these timestamps with link-addresses.

D. Related Work

The \( \pi \)-Cloud idea is focused on cloud storage but also intends to support the trading of services which might go beyond storage. In this context it is important to distinguish two types of data handling. If data are only stored they are called Data-at-Rest. If the data should be processed they are called Data-in-Use. Cloud services intended to process data have to work on plaintext rather than on ciphered one. There are approaches addressing this problem [4]. Unfortunately, at the current technical state the processing of algebraic operations on ciphered text—as it is possible with homomorphic encryption—is far too slow to be used in practice. So at the moment, if data are outsourced user’s privacy can only be ensured for Data-at-Rest. Given these circumstances, the \( \pi \)-Gateway has to deny access to sensitive data for public cloud services that intend to process the data. A deeper look at available cloud storage solutions reveals limitations in the protection of client data. Dropbox [5] and many others encrypt solely on server side which requires trust in the provider since the encryption key is deposited at the provider’s server and the user has no control over it. Only a few providers like Wuala intend encryption on client side. But the software they provide for encryption and upload are not open source. So the weak point is only shifted and the usage of such solutions is a matter of trust in the provider again. Trust provided, Wuala offers an interesting range of features. In addition to the “Cryptree” protocol [6]—a sophisticated key exchange scheme which enables users to share encrypted files with other users—local disk space can be traded for additional storage space in the cloud. Similar approaches are to be found in [7] and [8]. The former emphasize that most internet users have hardware which is much more powerful than needed for standard tasks like web browsing, blogging and chatting. Proceeding from that insight they suggest users might short-lease their resources. Based on similar assumptions the latter introduce the Cloud@Home middleware and a belonging software architecture implementation which should build a cloud based on heterogeneous nodes. In addition to backups in Public Clouds mutual backups between several \( \pi \)-Cloud users are thinkable and might be handled likewise. Based on the work presented in [9] information dispersal algorithms will be used to ensure \( n \)-redundant backups without the need to allocate \( n \)-times the space of the original data.

In addition to the limitations of public cloud storage solutions regarding the protection of client data management issues arise. Today’s cloud storage solutions lack automated management interfaces so that access control involves manual intervention by the user. This complicates the migration and integration of existing infrastructure and lowers the ease of use. The \( \pi \)-Gateway is intended to automate access control as far as possible.

Eben Moglen—professor of law at the Columbia University—tries to make people aware of the threats that are accompanied with technological changes like social networks and cloud computing. According to him, companies like Facebook—whose database reached 21 petabyte in May 2010 [10]—gather data about users to such an extent, that it might be easily used for suppression and the creation of a police state. The FreedomBox foundation [11] is fund-raising for the realisation of the vision based on Moglen’s idea of a decentralized architecture that enables users to share information hidden from commercial or governmental surveillance. This architecture includes that every participant contributes a low power, low cost plug server where everyone’s data are stored encrypted and distributed. The \( \pi \)-Gateway approach is similar to some extent. Started independently from the FreedomBox approach and Moglen’s vision the \( \pi \)-Gateway stands out with its concept of services and their distribution. The approach to encapsulate the user’s devices is also out of FreedomBox’s scope. Another difference can be found in the performance optimization that should be reached by differentiating between sensitive and non-sensitive data, so that not all the data have to be encrypted. The file examination taking place in the Data Module might be based on existing open source data analysis software like Apache Lucene [12] which therefore have to be compared and checked for suitability.

Approaches from different research fields have to be analysed for a proper design of the \( \pi \)-Gateway. Because of the large scale of this task the current status of the thesis
allows only a brief overview over the involved research fields. The architecture could for example be based on hybrid peer-to-peer techniques—as used in [13] for the purpose of resource sharing between organisations—as well as on backup strategies. In the first case one of the devices would be chosen as $\pi$-Gateway/$\pi$-Box—like a kind of super-peer. Grid computing solutions even though not optimised for distributed requests but for the fulfillment of a common goal—might nevertheless also deliver appropriate approaches. Knowledge from mobile agent technology might be of use as well, even though mobile agents are of little importance today. Especially knowledge about transmission of states is of interest for the creation of an architecture for the decision exchange. Today's insignificance of mobile Agents might be substantiated in the inability to protect carried data, code and the runtime environment reliable [14]. This shortcoming is irrelevant for the $\pi$-Box architecture because it can be assumed that only trusted devices exchange states. Backup strategies might provide Token-techniques that could help assigning responsibility to devices. Depending on the importance of migrated parts of the $\pi$-Box transaction control might also play an important role.

V. SUMMARY AND OUTLOOK

I have discussed the need for a secure access to cloud computing services which leads to the $\pi$-Cloud idea that was introduced afterwards. Then a scenario which showed the expected benefits of the $\pi$-Cloud idea was presented. In the concept chapter the arising problems where discussed, followed by the resulting research questions, a first solution approach and related work.

The next steps are dedicated to the creation of a requirement catalogue. As mentioned in the related work subchapter many different fields of research have to be observed in order to design the architecture of the gateway properly.

REFERENCES


A Secure Cloud Gateway based upon XML and Web Services

Sebastian Graf
Distributed Systems Group
University of Konstanz
Email: sebastian.graf@uni-konstanz.de
supervised by Prof. Dr. Marcel Waldvogel

Abstract—Storing data in the cloud offers a scalable and easy way to handle large amounts of data guaranteeing availability and scalability by the hosting Cloud Service Providers. The price for the gained availability is uncertainty about the integrity and confidentiality of the data. Even if common approaches provide high availability and end-to-end encryption necessary to achieve Availability and Confidentiality as security goals, other security requirements like Integrity and Accountability are neglected. The key management of those clients for encrypting data to satisfy Confidentiality must furthermore support join-/leave-operations within the client set. This work presents an architecture for a secure cloud gateway satisfying the common security goals Availability, Confidentiality, Integrity and Accountability. Mapping these security goals, XML as storage base is equipped with recursive integrity checks, encryption and versioning based on the native XML storage Treetank. A Key Manager extends this approach to provide the deployment of multiple clients sharing keys to the storage in a secure way. New key material is pushed to a server instance deployed as Platform-as-a-Service (PaaS) propagating this update to the clients. The server furthermore applies integrity checks on encrypted data within transfer and storage. Any communication between client, server and Key Manager relies on fixed defined workflows based upon web services. The proposed architecture called SecureCG thereby enables collaborative work on shared cloud storages within multiple clients ensuring confidentiality, consistency and availability of the stored data.

I. MOTIVATION OF SECURE CLOUD STORAGE

The flexibility of Cloud Based Services offers great possibilities to store any data in a guaranteed available and scalable manner. All data peculiarities ranging from block based byte chunks to Microsoft Word-documents are thereby persisted in the cloud. Cloud Storage Gateways represent convenient applications mapping such interfaces to common Cloud Service Providers. Most Cloud Storage Gateways thereby appear as standalone clients or centralized web applications.

Even if Cloud Storage Gateways enable the power of Cloud Based Services to different kinds of clients, the gained Availability comes at a price. While Availability and Confidentiality are provided by the hosting of the encrypted primary data on Cloud Service Providers, Integrity, Confidentiality and Accountability are not sufficiently considered in existing approaches. Another drawback of current approaches is the dependency of Cloud Storage Gateways to the platform they run on, e.g. they are shipped as complete operating system images, hard coded in routers or delivered as platform dependent applications. Flexible usages representing libraries, server based infrastructures or user specific clients are not supported. A more flexible usage includes collaborative use cases and related distributed environments. Even if Availability and Confidentiality are already provided, the related key management must be adapted to satisfy this usage. Common centralized access controls use the same key for all authorized clients which complicate modifications on the set of authorized clients. Besides the necessary protection of the data from unauthorized access of excluded clients, updated key material must be propagated to the valid clients in a scalable and secure manner.

An XML based architecture as Cloud Storage Gateway named SecureCG satisfies all security requirements, platform independence and flexible key management. Based upon the native XML storage Treetank [3], any data is wrapped on demand into XML[13] and persisted afterwards in the cloud. The underlaying tree structure enables SecureCG to provide easy ways of integrity checks supporting the security goal of Integrity. The Accountability is guarded through the native versioning of XML within Treetank. Encryption on the data satisfies the goal of Confidentiality whereas the underlying tree structure of the XML plus the provided versioning functionality is exposed. SecureCG supports flexible handling of multiple Treetank clients with unique de-/encryption keys using a standalone Key Manager and the VersaKey approach[12]. Equipped with the platform independent, block based interface jSCSI [7], and the adaptive REST based interface JAX-RX[4], the client provides a flexible and secure storage interface. The storage itself is encapsulated by a server deployed as Platform as a Service(PaaS) implementation. The PaaS implementation provides own integrity checks to ensure Integrity within the transmission and the storage of the data. Even if the server only stores encrypted data to ensure Confidentiality on Cloud Based Services, the server also checks and propagates new encrypted keys within the authorized client set. Communication between client, server and Key Manager relies on web services with defined workflows. Within this distributed architecture and as a consequence thereof the scattered functionality, SecureCG fulfill the security requirements even on untrusted storage while working with all different types of data.

Treetank as the base of SecureCG implements already the Accountability and partly the Confidentiality and the Integrity.
Parts of the client were already released to the open source community[3]. Results on local machines show constant access time regarding read- and write-access of any node within any version of the XML which is important since requests on single substructures must scale in the cloud. SecureCG is provided throughout as plain Java implementation which enables flexible usages independent from the underlaying platform.

Current approaches making use of cloud storages either use web services or pull the access to file system level based upon iSCSI or NFS. By utilizing web services as primary communication within a distributed architecture and by offering additional interfaces with the ability to store data on block level based upon jSCSI, SecureCG provides a flexible toolset for secure storage on untrusted third party applications. The distributed architecture consisting out of client, server and Key Manager allows scalable and flexible usage without any restrictions regarding security requirements or functionality.

II. APPLYING SECURITY TO CLOUD STORAGE GATEWAYS

Common Cloud Storage Gateway-approaches claim to store data “secure” on untrusted third party Cloud Service Providers by encrypting it. Since the encryption operations take place either on client side or on trusted applications within the Cloud Storage Gateway providers, the stored data indeed satisfies the requirement of Confidentiality. Hence, the key management offers no adaptive key handling supporting an evolving set of accessing clients. Besides the Confidentiality gained through the encryption of the storage, other common security requirements[10] namely Availability, Integrity and Accountability are not satisfactorily considered. Availability includes the prohibition of unauthorized modifications or deletions of any data whereas it is only partly considered in current Cloud Storage Gateway-approaches. Any secure Cloud Storage Gateway must offer fault tolerance data handling like roll-back-operations of any undesirable data modifications.

Most common cloud storages use only web services as interfaces restricting any error handling. Error handling is however mandatory to achieve Integrity within a system. Integrity is thereby divided into Data Integrity and System Integrity. System Integrity requires constant consistency of the data against any malfunctions regarding the architecture. This requirement can easily be satisfied within hash-based checks of the transferred and stored data. A direct usage of any cloud storage via web services inhibits this requirement since no consistency checks on the transferred data are performed. Additional to System Integrity, Data Integrity tracks the consistency of authorization within each modification on the data. User bound signatures satisfy this requirement if bound to a version and the corresponding modifications.

Extending these goals and thereby the common definition of security, the NIST Definition about Underlaying Technical Models for Information Technology Security[11] adds two additional requirements. Accountability describes the ability to trace any changes within the system. Regarding Cloud Storage Gateways, versioning or logging of any modifications on the data enable the Accountability and thereby support additional security features like non-repudiation.

Assurance, the second requirement defined within the NIST-definition, represents the necessity to define an overall workflow to provide security not only within a system but also within the using environment. This requirement covers social and personal aspects out of focus regarding SecureCG.

Satisfying all requirements denoted above, SecureCG consists out of the components shown in Fig. 1: The server performs computations on the data wherefore direct storages within the cloud like the Amazon S3 are not used. Due to the variety of services in the cloud[8], the storage of the data instead is represented by a server instance deployed as PaaS (e.g. Google AppEngine, Amazon Beanstalk, Windows Azure, ...). The inevitable non-confidential data handling upon Cloud Service Providers requires the stored data to be encrypted. Besides storing the encrypted data, the server furthermore propagates changes regarding encryption keys even though the server is not aware about the keys. Section II-C describes the server component including consistency checks upon encrypted data and propagation of new key material.

The purpose of any client instance is the providing of user centric interfaces (e.g. web services or iSCSI) to access the server within SecureCG. Even if the client can have
flexible representations fitting the requirements of the use cases, it must be deployed in a trusted environment. The trusted environment of the client maps the requirement of Confidentiality within the system. The clients additionally sign the modifications as well to provide Accountability and Integrity. The functionalities within the client are described in section II-A.

To improve the key handling regarding distributed clients and fine-granular access management, the key management is performed within an externalized Key Manager. The Key Manager organizes key material based upon VersaKey[12] for de-/and encryption as well as for signature issues. Since the key propagation within the proposed architecture is performed by the server, the Key Manager can be shutdown while no updates on the key management is performed. A detailed description about the role of the Key Manager is given in section II-B.

The synchronization of key material results in push messages based on REST between the Key Manager and the server. The necessity to use sessions motivates the usage of SOAP for transferring data between the client and the server since the transfer of primary data needs a preceding authorization workflow. The concrete workflows including the different techniques for data transfer are described in section II-D.

A. Outline of the Client

The client within SecureCG pursues the same target like common Cloud Storage Gateways: Different interfaces mapping the requirements of accessing tools are provided within the client. The ability to wrap any content for secure storage in the cloud is archived by using XML as common storage format. As the de-facto “lingua franca” in WWW-environments, most content is not only available as XML, additionally XML has the ability to offer flexible representations in a structured way. Interfaces to web services as well as iSCSI wrap any alien data into XML as denoted in Fig 1. The client relies, for handling XML in a scalable way, on the own implemented native XML database Treetank [3]. Treetank offers different features on node level (including versioning) based upon a layered architecture: Node based operations are provided within the node layer supported by a local node encoding. Independently of the position of nodes within the tree, all nodes are stored in pages representing the page layer. The page layer offers versioning inspired by ZFS[1] and encrypts any data based on the local keys. The de-/serialization process of the pages is independent from the concrete storage backends which include at the moment plain files and the BerkeleyDB. The data is within all storages only appended so no data is deleted based upon the versioning of the page layer.

The node based operations of the node layer, enabled by the local node encoding, include structural integrity checks. Based upon a recursive hash, each node guards the integrity of the corresponding subtree. Figure 2a shows an evolving tree within the insertion of node “67”. The green node is inserted while the yellow ones are updated with the help of reading the blue nodes. The arrows between the nodes represent the pointers stored to each node within Treetank. The insertion of a node updates all nodes on the ancestor-or-self-axis while all siblings on this axis must be read to recompute the new hashes. This functionality can be used not only to provide consistency within the storage but also while the data is in process e.g. regarding consecutive REST requests[5]. Equipped with an incremental hash function[2], the ancestor-sibling traversal becomes redundant which improves the performance of recomputing the hashes.

Next steps include the improvement of this approach by using cryptographic hashes since the incremental hash function is not aware of isomorphic tree structures. Using cryptographic hashes, the recursive integrity check guards the System Integrity within storage and transfer of any data. Extending the page layer with additional checksumming of pages similar to ZFS supports the System Integrity within the storage and the transfer. Regarding the necessity to provide Data Integrity, the hash structures will be extended to sponsor user bound signatures. The data itself will contain thereby the fingerprint of the last authorized user which modified the node and the version.

Security includes non-repudiation whereas Integrity represents a necessary requirement. A second requirement to gain non-repudiation is the Accountability to track changes within the system. Treetank supports, based on the page layer, versioning functionality. Figure 2b shows an evolving page layer within two revisions. All modified nodes are stored in new node pages, denoted by the blue color, created under
the related, red colored revision-root-page representing a new version. The indirect pages, colored yellow, multiply the fanout of the overall root (the uber-page) and the single revision-root-pages. Tag names are stored separately in name pages, colored green within Fig. 2b. All new versions result in new subtrees of pages including the modified nodes. Different versioning algorithms are supported within this hierarchy satisfying the security requirement of Accountability. Access is based upon transactions enabling atomic modifications on multiple nodes within one revision.

Further work to improve Accountability includes the research on adaptive versioning. Whereas incremental versioning offers the write-optimal way to represent changes between two versions, differential versioning represents the read-optimal approach. Since the read- and write-performance should be balanced over the time, a combined approach consuming more storage but resulting in predictable workloads is desirable.

All versioned data stored in the cloud must be encrypted to ensure Confidentiality. This encryption must take place in a trusted environment. Encryption in the “untrusted” cloud would violate the Confidentiality since Cloud Service Providers would have the ability to copy and store the unencrypted data. Treetank offers the ability to encrypt the nodes directly in the page layer. Since the encrypted pages are serialized to the Cloud Service Provider-infrastructure, the data in the cloud can only be accessed within an authorized client. The recursive integrity structure upon XML guards thereby the unencrypted data including any signatures whereas the integrity check on the page layer cares about the System Integrity based on the encrypted pages. The management of the key material for encryption- and signing-operations is organized within an external Key Manager. This supports the access of multiple clients to the same storage as described in section II-B.

The current encryption approach will be extended by a fine-granular access control mechanism respecting the recursive relationships within the tree structure. Based upon the hierarchy, access control on higher level nodes includes automatically the corresponding subtrees. Therefore, access control with different levels of ability must be introduced. A further extension includes the awareness of former versions within changing keys. Older versions should stay accessible within all valid clients at this given time.

Using XML as storage format generates synergies due to the underlying tree structure. The tree structure supports recursive consistency-checks and encryption ensuring Integrity and Confidentiality. Since Treetank consists out of flexible layers supporting versioning and different backends, interfaces to the storage must only wrap XML around any content. Available interfaces include REST [4] whereas extensions will cover even block based communications relying on jSCSI [7]. Treetank offers fine-granular authorization on subtrees[6] to support Confidentiality even on interface level. Available as pure Java implementation, Treetank and its extensions offers multiple possibilities of usages e.g. as third party library or deployed as an own web service within a trusted environment.

B. Standalone Key Manager

If working in a distributed collaborative environment based upon multiple clients, key management becomes a critical issue. Encryption keys must be shared within all clients to access the same data. This influences the security within the system since the keys have to evolve with ongoing modifications within the client set. To support join-/leave-operations of clients, SecureCG externalizes the key management and equips all clients with disjunct keys. An established approach to encrypt network traffic called Versakey[12] supports the joins and leaves of clients. An hierarchy organizes disjunct keys whereas the leafs represent concrete clients, denoted by the green nodes while the root represents the encryption key called DEK represented by the red nodes in Fig. 3. Using an unique key, each client has the ability to access all data. The purpose of the internal nodes is the combination of the keys in the related subtrees. Any leave or join within the client set results in an adaption of all keys on the path to the root. These changes result in encrypted logs named Key Trails. The Key Trails represent the new keys encrypted with the valid former keys of the authorized clients. The distinction between keys and Key Trails results in a distribution of the functionality regarding the management of the keys on the Key Manager and the propagation of changes within the server.

Figure 3 shows an example regarding the key management. Client “1” leaves the architecture consisting out of four clients. The Key Manager generates new keys for 01 → 01’ and DEK → DEK’ where DEK’ is the new encryption key. The new keys are encrypted with the old, valid keys e.g. E0(01’) denotes the new key 01’ encrypted with the key of client “0”. The encrypted keys E0(01’), E01'(DEK’) and E23(DEK’) represent the Key Trails and are pushed to the server in order to be propagated to the clients. Afterwards, all valid clients, accessing the storage, are triggered to update their keys and decrypt the new keys with their former ones. To access this data, each client holds not only its own key but also all keys to the path to the DEK. Client “3” has thus the ability to access the new DEK’ by decrypting the trail E23(DEK’) with its stored key 23.

With this approach and the distributed functionality of adapting keys and propagating changes, flexible joins and
leaves on the client site are possible. The combination of the versioning of the data and the traceability of modifications within the encryption keys based upon the Key Trails enables time aware access policies without re-encrypting any data. Additionally, the Key Trails increase the Availability due to the storage of the encrypted modifications on the server itself.

C. Storing onto any Cloud Service Providers

While common Cloud Storage Gateways map user required interfaces to common web service calls onto the cloud storage directly, the proposed server component of SecureCG consists out of a PaaS implementation since it has to extend the storage of the data with additional functionalities. These additional functionalities act as hooks before and after transferring data guarding Integrity and Confidentiality on the data. Using the hashes of the pages, System Integrity is ensured while the data is in transfer or in storage. With each transfer of an encrypted page, the corresponding hash is delivered. Any difference related to the delivered hash and the one computed on the received page results in a request to resend the data. Any page retrieved from the cloud storage by the server is checked against the deposited hash as well to prohibit any unforeseen changes on the data. Before any transfer of data, the keys are synchronized. In the case of outdated keys, the server propagates the related Key Trails to the client. This results in an on-demand service of the Key Manager increasing security and Availability due to the highly available and encrypted storage of the Key Trails on the server.

D. Communication between Server, Client and Key Manager

Communication within the proposed architecture takes place over common web services. SecureCG uses in this context three different kinds of communication workflows:

1) The first communication workflow is triggered because of changes on the set of clients and includes, besides an external trigger, the server and the Key Manager.

2) The second communication workflow represents the propagation of new key material from the server to the client.

3) The last communication workflow denotes the transfer of primary data between client and server.

The first communication workflow is based upon REST requests shown in Fig. 4a. An external trigger starts the key changing workflow based on a modifying REST request on the authorized client key. Note that the encryption keys are not retrievable over GET as resource. Even though identifiable as substructures, the only valid operations on the keys are DELETE- or PUT-operations on authorized client keys. A deletion request of client “1” results thereby in a new DEK’ and related Key Trails. The Key Trails are afterwards posted to the server to become available for all clients. The only invariant the Key Manager holds within this transmission workflow is the order: Each request coming in is handled in an atomic manner resulting in the generation of the Key Trails followed by the push of the Key Trails to the server instance. The parts communicating with each other within this workflow are identifiable as blue components in Fig. 1.

The second and the third communication workflow are combined as denoted in Fig. 4b. This combination results in a session-based order of requests. An authorization workflow checks if the DEK on the client is up-to-date before any transmission of primary data starts. At the begin of the authorization workflow, the server sends a challenge to the client. The client performs a concatenation of the challenge and the hash of its DEK, H(DEK), and sends back the result to the server.

Since the server also holds the actual H(DEK), the server reproduces the concatenation and compares both results. The server only stores H(DEK) and not the DEK itself since this would violate the Confidentiality.

If the hash differs, the Key Trails are send to the client to be decrypted. The decryption operations suitable to the own keys generate the actual DEK (if the client was not excluded
Treetank is implemented in Java and an ongoing project to provide a secure, versioned XML storage framework. The architecture is designed to support fine-grained, recursive-aware access to XML data while ensuring data integrity and confidentiality. The versioning approach is differential, allowing efficient management of changes to XML documents.

The proposed architecture combines common techniques from XML databases, web services, and security to provide a usable framework for secure cloud storage. Key features include:

- **Confidentiality**: Ensures that data is stored securely and accessed only by authorized users.
- **Accountability**: Tracks changes to the XML data, allowing for accurate auditing.
- **Integrity**: Maintains the correctness of the stored data.
- **Availability**: Ensures that users can access data when needed.

The architecture is designed to be extensible, allowing for the addition of new components without disrupting existing functionality. It supports both incremental and differential versioning, providing flexibility in how changes are managed. The system is designed to handle worst-case scaling scenarios, ensuring that performance is not affected by large numbers of revisions.

**Evaluation**

The performance of Treetank is evaluated using a benchmark that simulates a worst-case scenario involving 1000 revisions. The benchmark shows that the system scales well with the size of the data and the time taken for insertions and retrievals.

**Conclusion and Results**

The proposed architecture combines common techniques from the area of XML databases, web services, and security to provide a usable framework for secure cloud storage. The resulting framework will be extensible regarding interfaces to the client and flexible with focus on different possibilities of appliance. Furthermore, the resulting approaches are applied directly into an open-source reference implementation proving the suitability as well as the scalability.

**Final Steps**

Even if encryption is already implemented within the client, the access control needs further development to offer fine-granular, recursive-aware access on any subtree. This access control must be adapted within Versakey to fit the versioning approach to ensure access on older revisions for former clients. Besides the encryption which respects the versioning, the versioning itself must be adapted to balance between incremental- and differential versioning. This adaption, which might result in intermediate full-dumps of the entire data, must respect the encryption by extending the client based key handling with additional versioning functionality. The current consistency check is either perfromant or not aware of isomorphic subtrees. A combination of both approaches will satisfy the requirements of a cryptographic hash without consuming as much time as the current approach. To ensure System Integrity even in the cloud, a second hash structure will be introduced to guard the encrypted pages. Both hashes are combined with user bound signatures to combine System Integrity and Data Integrity on the XML within the structure (node layer) and the encrypted storage (page layer).

**Acknowledgements**

I would like to thank my supervisor, Marcel Waldvogel, for his guidance. Furthermore I would like to thank Anna Dowden-Williams for her more than valuable input.

**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/