BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INFORMATION SYSTEMS
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

GENERIC DECENTRALIZED SELF-ADAPTIVE CONTEXT-AWARE ARCHITECTURE MODEL

GENERIC DECENTRALIZED SELF-ADAPTIVE CONTEXT-AWARE ARCHITECTURE MODEL

EXTENDED ABSTRACT OF PHD THESIS
ROZŠÍŘENÝ ABSTRAKT DISERTAČNÍ PRÁCE

AUTHOR **Ing. M. MOHANNED KAZZAZ**
AUTOR PRÁCE

SUPERVISOR **Doc. Ing. JAROSLAV ZENDULKA, CSc.**
ŠKOLITEL

BRNO 2019

Abstract

The evolution in information system continuously raises demands for more efficient, effective and adaptive cooperation between system's components to cope with changes in the system and to guarantee its best performance. Two main approaches have been introduced to achieve these requirements. First, the self-adaptation approach which enables information system to adapt to the changes in context information of the system and its surrounding environment based on an adaptation strategy. Second, context-awareness approach which enables to monitor the context information and recognize those changes that can trigger the adaptation process.

In this work we introduce a generic context-aware self-adaptive architecture model to support software system with adaptation functionalities that guarantee system's availability, operation conditions and performance. Moreover, we provide two real-life case studies as a proof-of-concept of the applicability and re-usability of our proposed adaptation approach.

Abstrakt

Vývoj v informačním systému neustále zvyšuje nároky na účinnou, efektivní a adaptivní spolupráci mezi komponenty systému, aby se vyrovnal se změnami v systému a zaručil tak nejlepší výkon. K dosažení těchto požadavků byly zavedeny dva hlavní přístupy. Přístup k adaptaci umožňuje informačnímu systému přizpůsobit se změnám v kontextu informací systému a jeho okolního prostředí na základě adaptační strategie. Přístup ke zvyšování informovanosti zase napomáhá sledovat informace o kontextu a rozpoznat změny, které mohou proces adaptace vyvolat.

V této práci představujeme obecný kontextově orientovaný model vlastní adaptivní architektury pro podporu softwarového systému s adaptačními funkcemi, které zaručují dostupnost systému, provozní podmínky a výkon. Navíc poskytujeme dvě případové studie v reálném životě jako důkaz konceptu použitelnosti našeho navrhovaného adaptačního přístupu.

Keywords

Self-adaptation, software architecture, context model, decentralized control, context awareness.

Klíčová slova

Adaptabilita, softwarová architektura, model kontextu, decentralizované řízení, sledování kontextu.

Reference

KAZZAZ, M. MOHANNED. *Generic decentralized self-adaptive context-aware architecture model*. Brno, 2019. EXTENDED ABSTRACT OF PHD THESIS. Brno University of Technology, Faculty of Information Technology. Supervisor Doc. Ing. JAROSLAV ZENDULKA, CSc.

Generic decentralized self-adaptive context-aware architecture model

Declaration

I hereby declare that the thesis is my own work that has been created under the supervision of doc. Ing. Jaroslava Zendulky CSc. It is based on the seven papers [11, 12, 13, 10, 14, 15, 16] that I have written jointly with my supervisor specialist RNDr. Marek Rychlý, Ph.D. Where other sources of information have been used, they have been duly acknowledged.

.....
M. MOHANNED KAZZAZ
July 4, 2019

Acknowledgements

First, I would like to thank my parents and family for supporting me with their love and prayers all my life. Thank you so much!

I would like to thank doc. Ing. Jaroslava Zendulka, CSs., for his invaluable advice, support and guidance during his supervision of this work. Also, I would like to thank RNDr. Marek Rychlý, Ph.D., who has been a great source of guidance, inspiration and assistance during my doctoral studies.

Contents

Chapter 1

Introduction

The ever-developing nature of the distributed system arises demanding requirements of the design process for an automatic and robust management means. To satisfy these requirements, Self-adaptation has been proposed to support software systems with the mechanism to modify their behaviour and maintain their goals flexibly and robustly through an automatic reaction to information context changes of 1) actor's requirements, 2) surrounding environment, 3) and, the system itself [19, 5]. The reaction is a result of specific monitoring strategy the system should follow. Four functionalities have been defined in [22] as required functionalities in Self-adaptive system, the system must: 1) monitor context information of system and environment, 2) detect changes, 3) decide the adaptation plan to perform and 4) act by executing the chosen adaptation plan. On the other hand, Context Awareness approach has been proposed to solve the problem of information misunderstanding between system distributed components over different domains. It addresses the need to provide a unified model of information context which helps the realization of system adaptation by supporting system information context understanding, monitoring and discovery of context changes in the operational environment.

The self-adaptive context-aware framework for stationary and mobile devices presented in this thesis is a framework for enabling self-adaptation and context-awareness in information system. It utilizes ontology-based model to describe system components including their properties and preferences. Using an ontology-based model enables the adoption of context awareness approach concepts of context modelling, monitoring and context reasoning. Moreover, the framework supports the utilization of a decision-making process to choose the best adaptation scenario based on defined set of criteria. To present the application of the decentralized context-aware adaptive architecture model proposed in this research, we provided two case-studies with a detailed description of system configuration and framework's analysis and evaluation.

The introduced adaptive architecture model provides an answer to the question of how to support information systems with a dynamic response to changes in their surrounding environment. In other words, how to design a formal architectural model that supports information system reconfiguration during runtime based on the changes in system components and the current state of system environment. Moreover, the contribution of this thesis leverages the adoption of system adaptation in service-oriented mobile architectures. Also, it provides researchers with an adaptive architecture model supported with a multi-criteria decision-making process, which facilitates and easies the design and implementation of new adaptation scenarios through the utilization of the provided adaptive architecture model and implemented framework.

In the next section we present the motivation scenario behind this work and our proposed approach to support adaptation in software system and to solve the limitations of current approaches.

1.1 Motivation

Inspired by [29, 23], let us consider a traffic jam detection system as a real-life scenario of utilizing adaptation in car navigation system. The motivation behind utilizing the adaptation is to solve service's loss situation and to help drivers to avoid traffic bottlenecks on roadways.

To design such a system, it is required to enable traffic information exchanging between cars. For that, traffic information should be formally modelled and correctly understood by the navigation system application on each car. On the other hand, the navigation system requires to be context-aware by continuously monitoring and analysing traffic status to detect traffic jam situations that trigger a process of re-planning the route. Moreover, a process of deriving of alternative routes and recommending the best one should be provided. These requirements can be satisfied by defining a unified taxonomy of context terms to describe the system and its components. Context information like car speed, position, traffic status, route details, number of surrounding cars, etc., are pieces of information that should be realized and exchanged between cars during run-time. This information should be noted formally as car context model and published to be discovered by other cars. The usage of car context model facilitates the integration of context-awareness approach by enabling context changes monitoring and discovery. Moreover, it supports the utilization of self-adaptation to traffic information changes (i.e., traffic jam status), by defining adaptation conditions and the utilization of a proper decision-making process used to choose the best adaptation decision.

The importance of our research resides in the following points:

- The introduced context model of system components allows to provide a formal and common understanding of system context information between different application domains.
- The abstract core self-adaptive architecture model can be extended and customized to adopt new adaptation scenarios.
- The proposed framework facilitates system adaptation and service provision in mobile architectures.
- The proposed framework supports the integration of self-adaptation on existing information systems regardless their technical implementations which minimizes the upgrade effort to enable self-adaptation in those systems.

1.2 Thesis Objective

The general goal of this research is to design a decentralized self-adaptive architecture. In this architecture, we want to provide the possibility of implementing self-adaptation not only on the user side (i.e., end user interface) but also on the system itself in the way that allows the system to adapt regarding context's changes (i.e., light, temperature, communication bandwidth, battery status) through a decentralized adaptation which can

minimize the adaptation costs, guarantee the quality of provided services and improve system performance.

The specific objectives supporting the general objective can be summarized as follows:

O1 To provide a formal system context model which enables the understanding of context and context's changes meanings and promotes context awareness in the system.

O2 To provide a context-aware self-adaptive architecture model that adapts to the context's changes of system components and its surrounding environment.

O3 To provide a framework that supports adaptation in the system. The framework allows different types of devices to cooperate in centralized controlled orchestrations to solve a problem of context's loss or uncertainty by including new services that do not affect with the changes causing the problem.

O4 To analyse the efficiency of implementing the decentralized adaptation on system side on appropriate case studies.

1.3 Thesis Contribution

According to the proposed objectives, the following contributions are provided:

1. A system ontology to support the usage of common understanding of context information between different domains. A detailed description of the proposed ontology is provided in Section 3.1 and Section 5.2.

2. A formal ontology-based context model that allows to describe system component context model and provides a method to ease the discovery process of new context providers in the system. See Section 3.2 for more details.

3. A framework for distributed context-aware self-adaptive system (presented in Section 5.3), is provided to support an automatic system adaptation to context's changes of the system and environment. The adaptation guarantees the operation conditions to keep a desired Quality of Service (QoS) performance level. Moreover, the distributed mechanism will improve system performance by distributing system tasks of adaptation and context processing over several controllers which helps to avoid system overloads that could happen when utilizing the centralized approach.

4. An extensible adaptation architecture model (see Chapter 3) that can be easily customized by researchers over new case studies. The extensibility of the model eases and stimulates conducting research work on both self-adaptation and context-awareness.

5. A decision-making process to support choosing the best adaptation scenario from a set of alternative adaptations based on set of prioritized criteria. See Chapter 4 For more details.

1.4 Organization

The rest of the thesis extended abstract is divided into five chapters as follows:

In Chapter 2, we review the existing literature implementing the self-adaptive and context awareness approaches. Later, we demonstrate a motivation example using a real-life scenario problem and our proposed approach to solve it.

In Chapter 3, we introduce our proposed context-aware self-adaptive SOA meta-model. We present the ontology-based component context model proposed to describe system components. In Chapter 4, we demonstrate the algorithms developed to support adaptation decision making process in the system.

In Chapter 5, we provide an adaptive SOA based model by applying the meta-model proposed in Chapter 3.

In Chapter 6. we present our framework implemented to support Web service migration in SOA.

In Chapter 7. we provide two case studies to present the application and efficiency of our service migration approach to solve this thesis motivation example provided in Chapter 2.

In Chapter 8, we summaries the thesis approach and highlight its contributions.

Chapter 2

Self-Adaptive Context-Aware Architectures

In this chapter, we review the existing literature implementing the self-adaptive and context awareness approaches in information systems. Later, we introduce an evaluation of the existing researches and we present our proposed approach to solve the limitations found in these researches.

2.1 Adaptive Systems

There have been many researches to introduce a formal or semi-formal architecture model of adaptive systems that can adapt its behaviour or architecture in response to changes in its environmental context [27, 26, 6].

Self-adaptation was first introduced by IBM through the „Autonomic Computing“ approach [17] describing self-managing system using a central controller. The following functionalities have been introduced to define self-managing software system.

- Self-configuration which presents a system's ability to configure itself automatically according to high level policies of its objectives.
- Self-optimization which is achieved by a system through continuously seeking to improve and upgrade itself and its functionality by applying the latest versions of its components.
- Self-healing of adaptive system which is the ability to detect, diagnose and repair its components automatically.
- Self-protection which is the ability of continually predicting and defending system failures or attacks.

Jade [3], an agent development framework that facilitates the development and management of agent-based self-adaptive applications. It provides the tool to define agent platform, containers and agents and their tasks. Agent tasks can be extended by defining new behaviour class together with a behaviour ontology describing the term of this behaviour and then assigning the new behaviour class to the agent object. Jade framework supports the utilization of different ontologies to support different application domains. Using the ontology guarantees the correct understanding of messages between agents. Moreover, the

framework facilitates the integration of Web service through supporting a bidirectional interconnectivity between agents and Web services. Web services can be registered/deregistered in the Universal Description, Discovery, and Integration (UDDI) registry to be discovered and invoked by Jade agents. Jade supports the mobility of its mobile agents between different containers of the same platform. However, moving a mobile agent between Jade containers is only supported by a manual process requiring the definition of platform hops for an agent to visit till it reaches the destination.

Da Silva et al. [26], proposed a generic framework for the automatic generation of processes for self-adaptive software systems so that it can be applied to different application domains. The framework uses workflows, model-based and artificial intelligence (AI) planning techniques to design adaptation plans. They used a standardized AI planning language, the Planning Domain Definition Language (PDDL) [7], to define a system model that is composite of 1) a domain representation stating system's actions (or available tasks that can be used to formulate the adaptation plans), and 2) a problem representation that defines the system initial state and the desired goal. However, the proposed representation does not define relationships between system components and/or system properties. Moreover, the defined system context representation is limited to a static set of terms defined during design time. Which does not answer the question of how to support the automatic usage of newly available resources and tasks in the system during runtime.

FUSION framework for self-adaptive systems is based on self-tuning approach [6]. It uses a technique of analysing system features to define a system model that copes dynamically with the unanticipated system conditions. Feature relationships are used to improve the adaptation planning during runtime. The feature model of the system consists of one core and other features. These features are related by two kinds of relationships: 1) dependency which defines the prerequisites of this feature, and 2) mutual exclusion that helps to enable only one feature from the group of similar features. System context is presented as a *metric* which is a measured value of system properties. Moreover, *utility* refers to a user's context or his/her preference about a specific metric. They presented an analytical method to derive the behavioural model of the system by enabling or disabling system features depending on metrics and utilities.

We see from these studies that system adaptation can be provided through modelling both system's behaviour and architecture explicitly. A formal model representing context information must be defined to facilitate a shared understanding of system context information and their possible changes to be discovered and uniformly understood between all system components.

2.2 Context-aware Systems

Context-aware computing was introduced for the first time by [24] as the ability of a mobile user's application, which is constantly monitoring information about the surrounding environment, to discover and react to changes in this environment.

Context-awareness modelling is presented by two mechanisms, context binding and context triggering. The first one concerns with mapping between the input of service operator and context sources automatically. The second one represents the contextual adaptation of the services according to defined context constraints and a set of actions. The context binding mechanisms enable more possibilities for automatic execution of service.

Dey et al. [1] considered a system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.

CONON [28] is an OWL-based ontology which provides a formal context model and implements a Description Logic (DL) reasoning. A reasoning rules were used to reason over a low-level (explicit) context to derive a high-level (implicit) context based on the proposed ontology and by means of DL and Resource Description Framework (RDF) reasoners.

In [18], the authors use the Web Ontology Language (OWL) ontology and the Semantic Web Rule Language (SWRL) rules to model context in a context-aware system using Rule-Based Inference engine.

An OWL-based device ontology was provided by Bandara et al. [2] to describe devices and their hardware and software components. However, the proposed ontology lacks for full service descriptions as it only provides an initial representation for device's services using a relationship called *hasService* without providing a description of the service's attributes. On the other hand, in [4], an ontology has been used to provide a service's description and preferences and to allow match-making techniques on these descriptions.

2.3 Context-aware Adaptive Software Systems

As the context-awareness contribution is proposed to model, process, and manage context information, self-adaptation approach focuses on the ability of a system to adapt its structure, goals, mechanisms regarding changes in the operating environments. A novel approach was proposed by [8] to apply self-adaptive and context awareness together in software systems. In this survey, the requirements of integrating of these two approaches together have been identified as follow:

1. The context modelling requirements need to be considered from the system-context relationship perspective.

2. Self-adaptivity needs to have a system that can cope with the context/requirements changes (both anticipated and unanticipated), and then the system needs to be designed with adaptation in mind.

3. The requirements for the mechanism that integrates the context-awareness and self-adaptivity needs to be considered.

An abstract architecture has been proposed in a later work [9] and it consists of three layers:

1. *The functional system and its context layer* comprise of 1) the *functional system element* which presents system functionality through the running components and inactive ones. 2) the *context element* that manipulates the system operation and/or adaptation 3) the*interfaces* with the management layer which role is divided to find out the changes in the system or in its context and to apply the decided adaptation plans of the management layers on system functionalities.

2. *The system and its context representation layer* provide up-to-date *context model of the* environmental context and a *system model* of its running state. Moreover, it provides the implementation of the operation for applying the actions of the change management layer.

3. *The change management layer* checks any possible system consistency violation, derives the high-level context information and decides the suitable adaptation plans regarding to the context and/or requirements changes.

In this approach, system operation will adapt for both changes of *system model* and *context model*, which could cause extra processing needs especially for modelling and deriving context information and applying adaptation actions on the system.

2.4 Decentralized Self-Adaptive System

Self-adaptation approaches mainly presented adaptation as centralized or hierarchically controlled systems. A new contribution [29] addressed self-adaptation in decentralized managed software architecture. They divided the computational requirements for decentralized controlled system into four groups: 1) coordinated monitoring through sharing locally collected data of the partial system and its synchronization globally in the system, as the monitoring process is managed locally and each partial system has only access to his own knowledge; 2) coordinated analysis to provide a full analysis of each subsystems data to provide full knowledge analysis of system data; 3) coordinated planning between different planning units which could have different private goals that need to be reformed in one adaptation plan avoiding any possible confliction; 4) coordinated execution needed to synchronize and to manage execution plans of each partial system.

Self-healing subsystem is provided to recover camera failures using a self-healing manager component which analyses the monitored data about the status of the cameras. This manager executes a recovery strategy to ensure the consistency of the main system when a camera failure or loss is detected.

This approach presented a framework approach of decentralized self-adaptive subsystems which can avoid the bottleneck in processing of monitored data of participated devices or recovering process in each subsystem. However, the authors did not address the possibility of using other types of devices that can be participated in the proposed organization, which could provide more possible adaptation plans and requires modelling of the acquired data of different devices.

2.5 Discussion

In this section we identify the drawbacks of approaches demonstrated in the previous sections that should be addressed to support self-adaptation in software system. These limitations are listed as follows:

L1 Limitation in system extensibility, as in [26, 29] that provides an adaptive design pattern with a pre-defined set of tasks that can be considered in the design of adaptation plan.

L2 Limitation in context information modelling, as in [27, 26, 25] addressed the adaptation but with a limited context model representation. However, context modelling must be guaranteed to allow common understating of context information between several domains and to support system extensibility. Moreover, context modelling is essential to implement context-awareness and adaptation processes in the system.

L3 Limitation in adaptation strategies, as in [25] that requires a pre-defined adaptation strategy and limits the proposed approach for limited context-aware adaptation scenarios.

L4 Limitation in decision making during adaptation process, as in [27, 26, 29, 6] due to the limitation in the system context model representation or due to the consideration of a goal-oriented approach during system design. The decision-making process should be extensible to adopt new terms of system context model that can be used in making decision to support adaptation process.

2.6 Thesis Approach

Based on our evaluation of current approaches, we see that a context-aware self-adaptive SOA model is required to overcome the existing limitations of current approaches. It is required to provide a generic adaptation model that can promote service reusability and system extensibility in software system through the implementation the SOA principles during the design of system architecture. Moreover, it is required to provide a formal context representation of system components to support context awareness and monitoring of context changes in the system. Finally, we it is required to provide a dynamic decision-making process to support the utilisation of different adaptation strategies and to select the best adaptation plan based on metrics defined in the system context model.

To achieve context modelling, we choose to use the ontology-based approach for context-modelling not only because it describes a system semantically with a proper definition of the relationships between its components, but also regarding to its capability to reason with the Semantic Web. For example, Ontology Based Language (OWL) uses DL reasoner to derive new contextual information about system component which will be used in making the adaptation decision.

By the usage of a decentralized controlled adaptive units, we intend to speed up the adaptation process in the system so that the monitoring and executing adaptation efforts will be distributed over different controllers.

To avoid possible device-type-specific failures regarding some special environmental circumstances, different types of devices (for example, mobile phone, stationary devices) should be considered to participate in the adaptation process. Which also can allow service provisioning on mobile devices and extend the utilization of self-adaptation approach in mobile architectures.

Chapter 3

Context-aware Self-Adaptive Meta-Model

In this chapter, we introduce our meta-model proposed to describe SOA-based system and enable the utilization of context-awareness and self-adaptation. The meta-model is presented in our ontology-based component model schema. The proposed component model allows to describe system components including their properties and preferences. It allows system architects to describe general system operational conditions that should be guaranteed during the run-time. Moreover, it supports the definition of system adaptive behaviour to context changes through a planned self-adaptation strategy.

3.1 System Core Ontology

The core ontology is a set of terms of architecture components and relationships between them. It basically describes SOA-based system as a set of Services and Service Providers. Shown in Figure 3.1, the core ontology is presented using Protégé[1], and it consists of the following classes. The *Service* class, defining *Service* components in SOA, has a relationship with the *ServiceProvider* class called *providedBy*. A *ServiceProvider*, is identified by *hostname* and *protocol* properties. The relationship between *ServiceProvider* and its *Service* is called *provides*.

An object property *hasProperty* is defined in the ontology to describe properties of system components. A component's property object is defined as an instance of ontology class *Property*.

The*Property* class has the following attributes

- *propertyName:* is the name of the *Property*.
- *propertyType*: is the data type of the *Property*.
- *propertyValue:* is the value of the *Property*.
- *criteria:* it states the *CriteriaProperty* that affects the *Property* instance during the decision-making process.

The *CriteriaProperty* has the following attributes:

[1] www.protege.stanford.edu/

- *name*: is the name of criterion that governs the property of system component.
- owner: it has a value of "origin", "destination", and "service", and limits possible owners of the properties which are referring to a particular criterion (for example, a criterion with the owner value set to "service" can be referred only from service context models' properties, i.e., it can be applied only on services, not on service providers).
- *valueWithHighestWeight:* indicates which values of properties referring a particular criterion are considered to be the most important during the decision-making process.
- *valueWithLowestWeight:* indicates which values of properties referring a particular criterion are considered to be the least important during the decision-making process.
- *criteriaPriority:* indicates a general importance of a particular criterion by integer values between 1 and 10. The *criteriaPriority* property of a criterion determines the priority of a preference rule affected by the criterion.

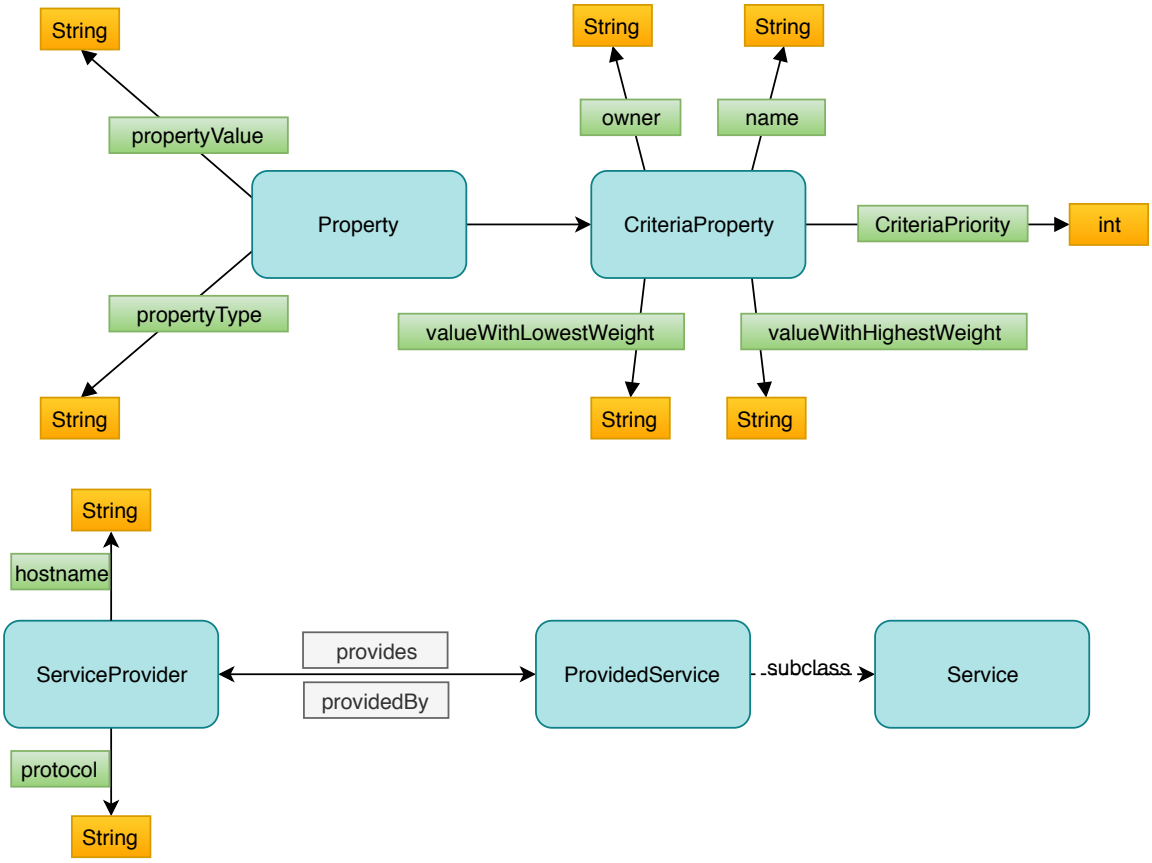

Figure 3.1: **System Core Ontology.** The ontology proposed to describe SOA component.

3.2 Ontology-based Context Model

In this section we demonstrate our context model defined based on the core ontology introduced in the previous section. Using the semantic terms proposed in the core ontology, the

context model defines SOA components and provides a semantic description of the joined service providers and hosted Web services stating their specifications and properties. The context model describes the preferences and conditions that control system's components through semantic-based rules. The full system context model is created by aggregating system components models. A higher-level context is inferred from the final composed system context model through a reasoning process so that new context information can be used in making the adaptation decisions.

The importance of utilizing a dynamically generated model of SOA service providers becomes highly demanding in order to identify and perform the proper reactions to changes in service providers resources. On the other hand, the fixed pre-defined service model can be enriched to contain relevant information about system preferred resources that possibly affect service performance.

A *Service* model holds the context information designed to be used in planning of the adaptation process provided in the system. Similarily, a*ServiceProvider* model contains context information stating its properties, work preferences and conditions that specify the possible hosted services.

{ „name“:„defines the name of system component“,
„*type*“:„contains the type of system component. i.e., “Service” or “ServiceProvider“,
„noPreferenceRules“:„ “true” if there is rules element in the model, “false” if there is not“,
„properties“: {„propertyName“:„states the name of the property“,
„propertyValue“:„states the value of the property“,
„propertyType“:„states the data type of the property, for example, “INT”“,
„criteria“:„ServicePriorityCriterion“},
„rules“:„[RDFS rule: ...]“
}

Figure 3.2: A simplified schema of the system component.

In Figure 3.2, we present this proposed component context model which has consists of the following elements:

- *name*: is a JSON element stating the name of a Service to be identified in the system.
- *type*: is a JSON elements stating the type of related service.
- *properties*: is an array of JSON Objects, describing component's object properties.
- rules contain sub rule tags of string expression. Each rule represents one RDFS-based rule describing component's operation preference possibly related to its defined properties.
- *noPreferenceRules:* is a data property with a value of *'false'* if the service has preferences and '*true'* if it has not.

Chapter 4

The Decision-Making Process

The process of enabling software system to make an adaptation must contain a sub-process of making a decision. Such a mechanism is needed to make the decision that satisfies system's and components' rules and requirements. Usually there will be a set of possible decisions to choose from and several factors affecting the decision-making process. These factors must be considered in order to make more reliable and optimal decisions. To support the decision making process, the Analytic Hierarchy Process (AHP, [21]) method is utilized so that the best decision will be selected from a set of all possible adaptation decisions previously found by ontology reasoning on the context model.

The AHP starts with creating comparison criteria matrix A, which is an $m \times m$ matrix of real numbers where m is the number of considered criteria. In the matrix, a_{ij} is the importance of the i^{th} criterion over the j^{th} one (diagonal entries a_{ii} are set to 1). If the i^{th} criterion has the same importance as the j^{th} one, a_{ij} entry is set to 1, otherwise values of a_{ij} entry range over 3, 5, 7, or 9, which indicate that the i^{th} criterion is slightly more important, more important, strongly more important, or absolutely more important, than the j^{th} criterion, respectively.

Usually, in existing applications of the AHP method, entries of the matrix above are set directly through a user's judgment. In our case, we can automate this process by utilization of *decision making criteria* (and the related properties and preference rules published by services and service providers) to compute individual a_{ij} entries of the matrix. We use the *InitializeCriteriaMatrix* algorithm (see Figure 4.1) to calculate the a_{ij} entries in the upper-right triangular part of matrix A by comparing criteria priorities and in the lower-left triangular part of matrix A as reciprocal values of the symmetric entries in the upper-right triangular part. The algorithm guarantees the consistency of judgments between the criteria and satisfies a consistency ratio condition of comparison matrix A to be less than 10% as required for AHP.

After initializing A matrix, AHP computes matrix $\bar{A}$ by normalizing A entries to make the sum of each column entries equals to 1 through equation

$$\bar{a}_{ij} = \frac{a_{ij}}{\sum_{k=1}^{m} a_{kj}} \tag{4.1}$$

Then, AHP computes weight vector w of criteria by computing the average value of each row of normalized matrix $\bar{A}$ through equation

$$w_i = \frac{\sum_{k=1}^{m} \bar{a}_{ik}}{m} \tag{4.2}$$

Require: $\langle p_1, p_2, \ldots, p_m \rangle$ as values of *CriteriaPriority* of criteria $\langle c_1, c_2, \ldots, c_m \rangle$
Ensure: A is a pair-wise criteria comparison matrix for given criteria $\langle c_1, c_2, \ldots, c_m \rangle$

for $i \leftarrow 1$ **to** m **do**
    $a_{ii} \leftarrow 1$
    for $j \leftarrow i{+}1$ **to** m **do**
        $difference \leftarrow |p_i - p_j|$
        if $difference \geq 8$ **then**
            $judgment \leftarrow 9$
        else
            $judgment \leftarrow 2\lfloor \frac{difference}{2} \rfloor + 1$
        end if
        if $p_i > p_j$ **then**
            $a_{ij} \leftarrow judgment$
        else
            $a_{ij} \leftarrow judgment^{-1}$
        end if
        $a_{ji} \leftarrow a_{ij}^{-1}$
    end for
end for
return A

Figure 4.1: The *InitializeCriteriaMatrix* algorithm to compute a pair-wise criteria comparison matrix for AHP based on *CriteriaPriorities* of individual criteria.

Finally, the *InitializeDecisionMatrices* algorithm (see Figure 4.2) is executed for each considered criterion c_k, where $k = 1, \ldots, m$, of criteria set $C = \{c_1, c_2, \ldots, c_m\}$. The multiple executions of the *InitializeDecisionMatrices* algorithm are used to create $n \times m$ matrix $V = [V^{(1)}, V^{(2)}, \ldots, V^{(m)}]$, where n is the number of possible adaption decisions found before. In the matrix V, each $V^{(k)}$ is a transpose of the weight vector of matrix $S^{(k)}$ obtained by an individual execution of the *InitializeDecisionMatrices* algorithm.

By applying the *InitializeDecisionMatrices* algorithm for all m considered criteria we get m weight vectors of possible decisions, where each vector is related to one criterion. Finally, AHP computes the composite weight vector p of all possible n decisions through equation

$$p = V \cdot w \tag{4.3}$$

where V is the $n \times m$ matrix obtained by multiple executions of the *InitializeDecisionMatrices* algorithm as described before (one execution for each each criterion) and w is the weight vector of criteria from Equation 4.2.

The decision with the highest composite weight entry of the vector p is considered to be the best decision and it is selected to be executed.

Require: criterion k and its attributes $valueWithHighestWeight^{(k)}$ and $valueWithLowestWeight^{(k)}$; $\langle p_1, p_2, \ldots, p_n \rangle$ as values of services or service providers properties that consider k as their criterion

Ensure: $n \times n$ matrix $S^{(k)}$ as a decision comparison matrix based on criterion k

1: **for** $i \leftarrow 1$ **to** n **do**
2: $s_{ii} \leftarrow 1$
3: **for** $j \leftarrow i + 1$ **to** n **do**
4: $range \leftarrow valueWithHighestWeight^{(k)} - valueWithLowestWeight^{(k)}$
5: $diffValue \leftarrow p_i - p_j$
6: $fifthOfDiff \leftarrow range/5$
7: **if** $diffValue > 4 * fifthOfDiff$ **then**
8: $s_{ij} \leftarrow 9$
9: **else if** $diffValue \leq 4 * fifthOfDiff \wedge diffValue > 3 * fifthOfDiff$ **then**
10: $s_{ij} \leftarrow 7$
11: **else if** $diffValue \leq 3 * fifthOfDiff \wedge diffValue > 2 * fifthOfDiff$ **then**
12: $s_{ij} \leftarrow 5$
13: **else if** $diffValue \leq 2 * fifthOfDiff \wedge diffValue > fifthOfDiff$ **then**
14: $s_{ij} \leftarrow 3$
15: **else if** $diffValue \leq fifthOfDiff$ **then** $s_{ij} \leftarrow 1$
16: **end if**
17: **if** $range * diffValue < 0$ **then**
18: $s_{ij} \leftarrow s_{ij}^{-1}$
19: **else if** $range * diffValue = 0$ **then**
20: $s_{ij} \leftarrow 1$
21: **end if**
22: $s_{ji} \leftarrow s_{ij}^{-1}$
23: **end for**
24: **end for**
25: **return** S

Figure 4.2: The *InitializeDecisionMatrices* algorithm to compute an adaptation decision comparison matrix based on a given criterion.

Chapter 5

Web Service Migration-based Adaptive Service Oriented Architecture Model

The migration of a particular service should be considered if its provider is not able to guarantee the functionality or quality of the service and there is no alternative service or service composition that will match a semantic and qualitative description of the original service, i.e., that can provide the same functionality and required quality. In the next section we provide a supplementary study demonstrating specific research work on supporting information system with migration adaptation.

5.1 Service Migration

The service migration itself can start when a particular service is selected to be migrated to a particular destination service provider by a migration controller located in the migration framework hosted on each service provider. An orchestration controller can perform service migration by getting a deployment package of the service and deploying it to the destination. During this process, the migrating service is stopped, and its internal state is stored and sent to the destination provider. All further incoming calls of the service are postponed until the migration is completed, i.e., until the migrated service is initiated in the new location, its internal state is restored, and until the service is able to handle incoming messages.

5.2 Web Service Migration Ontology

To enable service migration with the proposed adaptive approach, we need to accommodate Service and *ServiceProvider* types and their properties. For that, we extend the system core ontology presented in Section 3.1 by defining sub-classes of *Service*, *ServiceProvider* and Property to enable the description of service migration case study.

Demonstrated in Figure 5.1, the new sub-classes are listed as follows:

- ***ProvidedService***: is a subtype Service provided by a *ServiceProvider* of Service.
- ***MigratableService***: a subtype of *ProvidedService* class which possible to be migrated from one *ServiceProvider* to another one.

- ***FrameworkService:*** a subtype of *ProvidedService* class, it is an auxiliary service concerned with managing the migration process.
- ***CandidateForMigrationService***: a subtype of *ProvidedService* class, which represents the service found with violated preference and/or causing violations of its current service provider preferences, so, it is recommended to be migrated to other service providers by classifying it by this subtype.

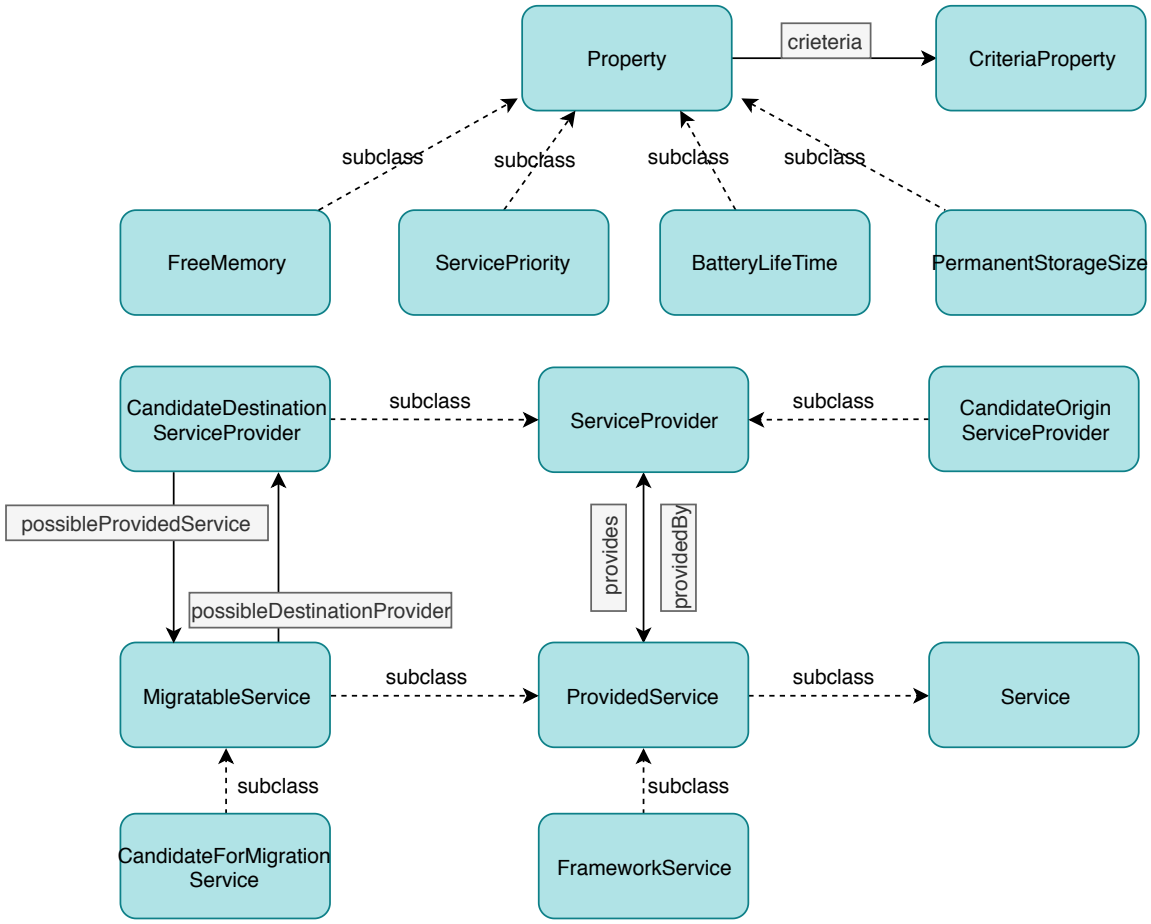

Figure 5.1: **Service Migration Ontology.** The ontology of web service migration system.

- ***CandidateOriginServiceProvider***: is a service provider having the required auxiliary *FrameworkService*-s to send its migratable services to another service provider. A CandidateForMigrationService must be hosted on service provider of *CandidateOriginServiceProvider* to be able to migrate successfully.
- ***CandidateDestinationServiceProvider***: is a service provider that can be a destination for one or more services of the type CandidateForMigrationService.

Additionally, we defined two object properties to express the relationships between the *MigratableService* and *ServiceProvider*.

- ***possibleDestinationProvider***: is the object property defining the relationship between the MigratableService and CandidateDestinationServiceProvider classes as the domain and range respectively.
- ***possibleProvidedService:*** is the object property defining the relationship between the CandidateDestinationServiceProvider and *MigratableService* classes as the domain and range respectively.

5.3 Mobile Web Service Migration Framework Architecture

In this section we demonstrate the framework architecture for service migration presented in Section 5.1. The framework's Controller running on service provider will lead both the context-awareness and adaptation processes. On the first hand, context-awareness process is presented in the system through the following functionalities:

- discovering the connected service providers in the network.
- checking the destination service provider availability.
- monitoring the quality of service after migration and deciding if another migration is required.

On the other hand, system adaptation is presented through the ability of the framework controller on a service provider to migrate a service to a new service provider and assure the availability of the migrated service.
The architecture of proposed framework consists of three modules shown in Figure 5.2.

5.3.1 Discovery Module

The Discovery Module is responsible for a service provider discovery process and retrieves a list of connected service providers in the network. Moreover, it is responsible for the discovery of *FrameworkService*-s and *MigratableService*-s (see Section 5.2) hosted on each discovered service provider so that their models can be requested in order to be considered in the Context Manager Module's reasoning process.

5.3.2 System Context Manager Module

This module is responsible for generating, monitoring and reasoning system context periodically to enable system context awareness. The module creates system core context model and all partial context models of discovered service providers and *MigratableService* model intended for migration. Then, the *Controller* looks for a destination service provider that can host *MigratableService* where the pre-defined rules of both *MigratableService* and the destination service provider can be satisfied.

Through reasoning the system context model process, new context information of *possibleDestinationProvider* for *MigratableService* are generated in the mdodel together with other information of *MigratableService* as a possibleProvidedService.
The output of this module is a list of triple entries stating the *MigratableService*, the source service provider, and the possible destination service provider. This list of entries is the input of the decision-making process performed by the Migration Module.

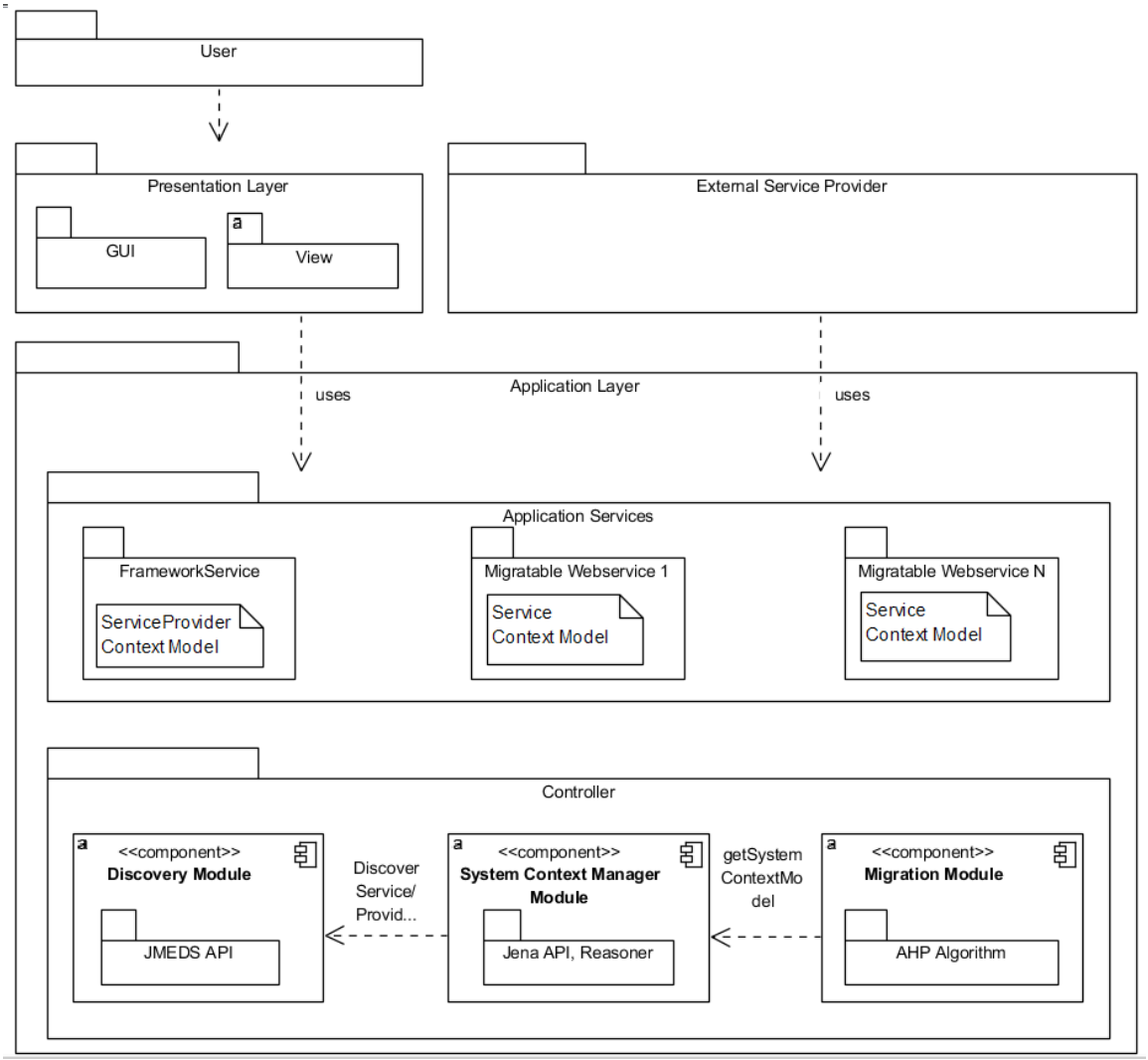

Figure 5.2: Mobile Web service migration architecture.

5.3.3 Migration Module

This unit is responsible for selecting the best migration to perform from the input set of possible migrations. It utilizes the algorithms proposed in Chapter 4 to initiate the required matrices used by the multi-criteria decision-making method AHP to choose the migration with the highest priority calculated based on priority values of the defined criteria.

Chapter 6

Service Migration Framework Architecture

To support the Web service migration process, we designed a generic framework. The framework describes an overall service-oriented architecture supporting the service migration and defines interfaces which can be implemented to adapt the framework to a particular Web service implementation technology. It also provides extension points for user-defined migration decision strategies, i.e., the strategies deciding when the migration of a particular service is needed and how it will be performed. To utilise the framework, demonstrated in Figure 6.1, an implementation of interface MigrationDecisionStrategy and auxiliary classes with interfaces ProviderStatus, ServiceStatus, and ServiceSemanticDescription, representing state and semantic information, is provided.

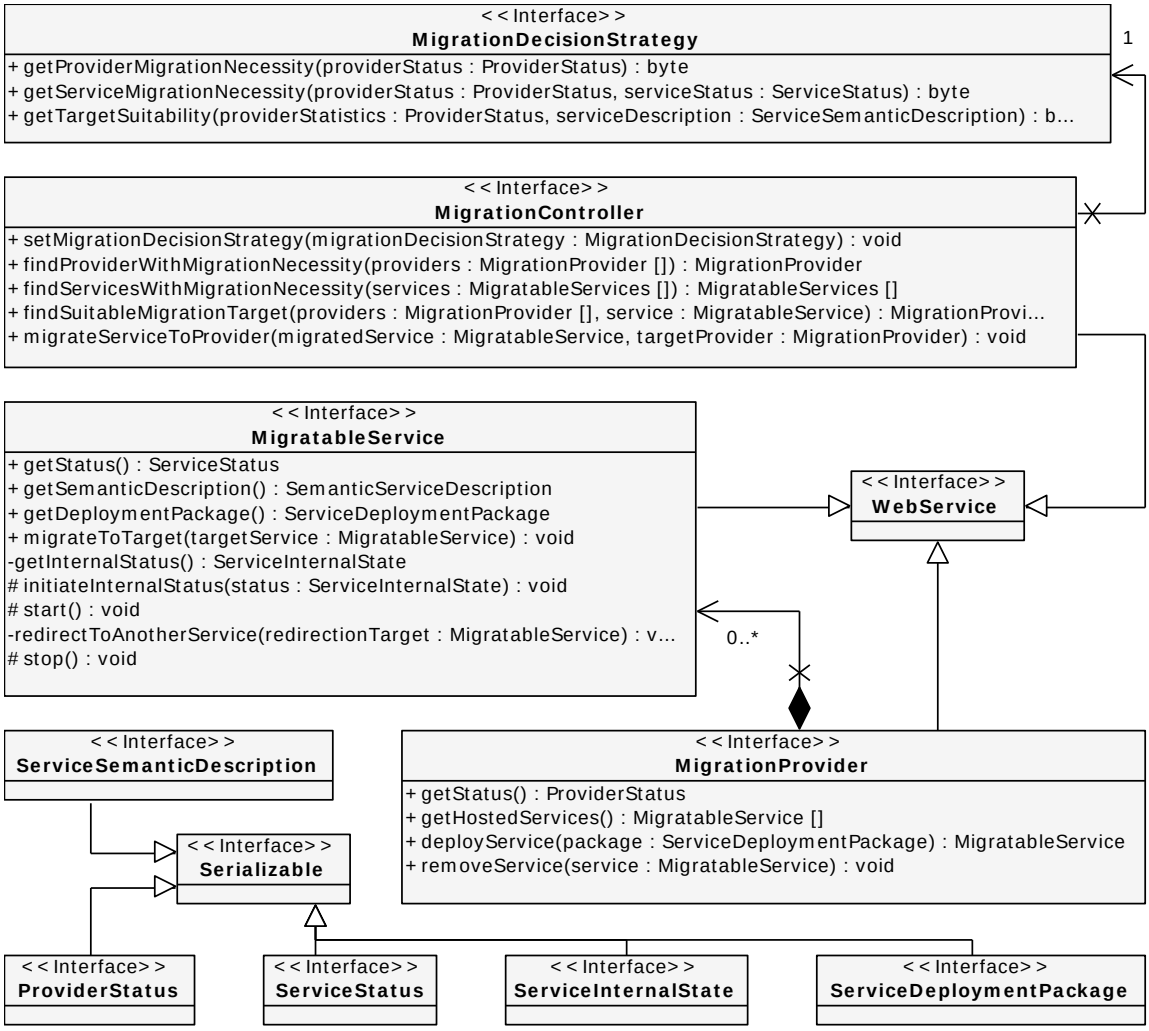

Figure 6.1: The interface of the framework's controller and the interfaces implemented by participating services and service providers to enable the service migration.

Chapter 7

Case Studies

In this chapter we demonstrate the proposed self-adaptive service-oriented architecture for Web service migration between stationary and mobile service providers through two case studies. The case studies have been provided as proof-of-concept of the proposed migration adaptation mechanism for Web service in real-life scenarios. The experiments conducted in this chapter show the applicability and efficiency of the proposed decentralized migration-based service-oriented architecture which is the main goal of the thesis described in Section 1.2.

7.1 Case Study 1 - Traffic Jam Detection Service Migration

We demonstrate a case study of service migration for traffic jam detection using the proposed Web Service migration framework. This case study is inspired by the traffic jam scenario presented in [20] where the migration framework is installed on a group of cooperative cars. The adaptation is performed to support a car navigation application while the location service is down or unreachable. The system controller adapts to this service loss by migrating a Traffic Jam Detection Service to another car to perform the computation needed to get the missed traffic information about a specific area and send it back to the original car.

7.1.1 Results

The case-study provides a proof-of-concept of the applicability of our implemented framework for *TrafficJamSearch* service migration between cooperative cars. Through this case study we present the functionality and usability of the context-aware self-adaptive Web service migration approach proposed in this thesis and the possibility to be customized for different real-life scenarios. The experiment results show the efficiency of the proposed decision-making process demonstrated by the time required to make the decision compared to the total time of the migration process. Moreover, the results show that the framework's performance is seamless and suitable for real-time implementations.

7.2 Case Study 2 - Tourist Video Streaming Mobile Service Migration

We propose a case study of service migration between mobile devices to demonstrate the validity, applicability and efficiency of the self-adaptive migration-based service-oriented architecture approach. We perform experiments to demonstrates the improvement on the *QoS* gained by the migration based on decentralized adaption mechanism. Also, we provide a performance analysis to show the light-weight impact of migration adaptation process and framework on the resources of mobile devices as example of resource restricted service providers.

7.2.1 Results

- Migration Process Time: Based on the performed measurements, we see that the proposed framework enables a seamless adaptation in SOA to redistribute system components.
- CPU Usage Consumption: The measurements show that the framework total CPU usage is 23% in average (18% in User mode and 5% in Kernel mode), while it is 8% and 50% at its minimum and maximum values respectively.
- Battery Consumption: The battery level drop is by 4% higher when the framework is disabled (OFF).

7.3 Conclusion

In this section we describe the contributions that have been provided through this work to solve the limitations of current approaches (see Section 2.5) and to support adaptation in software systems. The contributions of our approach are listed as follows:

1. A SOA-based architecture meta-model has been provided in Chapter 3 to support system adaptation and context-awareness in stationary and mobile architectures.
2. To solve limitation L1, the limitation of reusability and extensibility in the system, we adopt the SOA principles of service reusability and extensibility in designing our architecture model. As presented in our case studies (see Chapter 7), service reusability has been a key factor to support customization of our architecture model to support different adaptation plan.
3. To solve limitation L2, the limitation in context information modelling in the system, an ontology has been provided (see Section 3.1 and Section 3.2) to guarantee a dynamic and sharable understanding of context information between system components, which also supports system extensibility over different domains. Moreover, a generic ontology-based component context model has been introduced to describe system components semantically and to support content information modelling. The provided component context model supports the usage of the OWL-S description of Web service. It enables the utilization of system adaptation and context awareness by providing a method to semantically describe system services and devices including their properties and preferences to be used in planning the adaptation.

4. To overcome limitation L3, the limitation in adaptation strategies, a customizable adaptation approach has been provided in Chapter 3 to support different adaptation plans by extending the system ontology core model and integrating rules stating the adaptation plans. In our case studies (see Chapter 7), we customized our adaptation model to support service migration as example of adaptation plan and demonstrated its application through two real-life scenarios.

5. To overcome limitation L4, the limitation in the adaptation making process, a dynamic multi-criteria decision-making process has been provided to choose the best adaptation to perform from the possible adaptation plans (see Chapter 4). The provided decision-making process is extensible so that the newly ontology-defined terms, metrics, properties and criteria that can be dynamically considered in the adaptation decision making process.

Chapter 8

Conclusion

The research work of the thesis has been conducted based on the objectives presented in Section 1.2. The main objective of the thesis is reached by the proposal of a decentralized context-aware self-adaptive service-oriented architecture approach.

Following the objective O1, we proposed a formal representation of system context using ontology. The proposed ontology enables a common understanding of the semantic meaning of system components and allows to define system components models and realizes changes in these models. A context model states the description of a system component including its attributes and operation conditions to consider in system adaption.

With regard to objective O2, we introduced a decentralized context-aware self-adaptive architecture model that enables SOA-based software system to react to context changes in the surrounding environment and/or the status of system components. We presented a formal description of service migration strategy to demonstrate the behavioural description of service migration-based adaptive SOA.

To reach objective O3, we developed a framework that supports service migration adaptation and allows stationary and mobile devices as service providers to join the adaptation process. The implanted framework enables system component context modelling, context exchange, context reasoning, decision making, service migration adaptation, service provisioning on both Android and Windows platforms.

Finally, regarding to objective O4, we presented two case studies to demonstrate the applicability and efficiency of the thesis approach. The experiment results are provided as proof-of-concept to show the validity of the decentralized context-aware adaptive architecture model. The results provide measurements to presents the efficiency of the proposed approach through showing improvements in system performance and quality of service factor by considering the thesis approach.

In comparison with the related approaches, our approach utilizes context-awareness and self-adaptivity to guarantee and improve system performance. Using the proposed context-aware self-adaptive meta-model allows to provide semantical description for both Web services and service provides. Moreover, the thesis promotes the extension of service-oriented architecture on mobile devices and enables the participation of type different types of context resources and devices independently from their computing platform.

For future work, we consider empowering the proposed approach with other discovery methodologies of service and service provider in large-scale network. Moreover, we intend to extend the implementation work to join new types of devices and services. Another part of the future work is to test the approach through other real-life examples with different adaptation scenarios.

Bibliography

[1] Abowd, G. D.; Dey, A. K.; Brown, P. J.; et al.: Towards a better understanding of context and context-awareness. In *International symposium on handheld and ubiquitous computing*. Springer. 1999. pp. 304–307.

[2] Bandara, A.; Payne, T. R.; de Roure, D.; et al.: An ontological framework for semantic description of devices. 2004.

[3] Bellifemine, F.: Jade-a white paper. *exp*. vol. 3, no. 3. 2003.

[4] Bianchini, D.; De Antonellis, V.; Melchiori, M.; et al.: Lightweight ontology-based service discovery in mobile environments. In *17th International Workshop on Database and Expert Systems Applications (DEXA'06)*. IEEE. 2006. pp. 359–364.

[5] Cheng, B.; De Lemos, R.; Giese, H.; et al.: A research roadmap: Software engeering for self-adaptive systems. In *Schloss Dagstuhl Seminar*, vol. 8031. 2009.

[6] Elkhodary, A.; Esfahani, N.; Malek, S.: FUSION: a framework for engineering self-tuning self-adaptive software systems. In *Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering*. ACM. 2010. pp. 7–16.

[7] Fox, M.; Long, D.: PDDL2. 1: An extension to PDDL for expressing temporal planning domains. *Journal of artificial intelligence research*. vol. 20. 2003: pp. 61–124.

[8] Hussein, M.; Han, J.; Colman, A.; et al.: An architecture-based approach to developing context-aware adaptive systems. In *2012 IEEE 19th International Conference and Workshops on Engineering of Computer-Based Systems*. IEEE. 2012. pp. 154–163.

[9] Hussein, M.; Han, J.; Colman, A.; et al.: An architecture-based approach to developing context-aware adaptive systems. In *2012 IEEE 19th International Conference and Workshops on Engineering of Computer-Based Systems*. IEEE. 2012. pp. 154–163.

[10] Kazzaz, M.; Rychly, M.: Ontology-based context modelling and reasoning in the Web service migration framework. *Acta Electrotechnica et Informatica*. vol. 13, no. 4. 2013: pp. 5–12.

[11] Kazzaz, M. M.: Semantic Services Migration. In *Proceedings of the 18th Conference STUDENT EEICT 2012 Volume*. Brno University of Technology. 2012. pp. 386–390.

[12] Kazzaz, M. M.; Rychlỳ, M.: A web service migration framework. In *ICIW 2013, The Eighth International Conference on Internet*. The International Academy, Research and Industry Association. 2013. pp. 58–62.

[13] Kazzaz, M. M.; Rychlỳ, M.: Web service migration with migration decisions based on ontology reasoning. In *Proceedings of the Twelfth International Conference on Informatics-Informatics*. 2013. pp. 186–191.

[14] Kazzaz, M. M.; Rychlỳ, M.: Web service migration using the analytic hierarchy process. In *2015 IEEE International Conference on Mobile Services*. IEEE. 2015. pp. 423–430.

[15] Kazzaz, M. M.; Rychlỳ, M.: Restful-based mobile Web service migration framework. In *2017 IEEE International Conference on AI & Mobile Services (AIMS)*. IEEE. 2017. pp. 70–75.

[16] Kazzaz, M. M.; Rychlỳ, M.: A Case Study: Mobile Service Migration Based Traffic Jam Detection. *International Journal of Systems and Service-Oriented Engineering (IJSSOE)*. vol. 8, no. 1. 2018: pp. 44–57.

[17] Kephart, J. O.; Chess, D. M.: The vision of autonomic computing. *Computer*. , no. 1. 2003: pp. 41–50.

[18] Lee, K.-C.; Kim, J.-H.; Lee, J.-H.; et al.: Implementation of ontology based context-awareness framework for ubiquitous environment. In *2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*. IEEE. 2007. pp. 278–282.

[19] Oreizy, P.; Gorlick, M. M.; Taylor, R. N.; et al.: An architecture-based approach to self-adaptive software. *IEEE Intelligent Systems and Their Applications*. vol. 14, no. 3. 1999: pp. 54–62.

[20] Riva, O.; Nadeem, T.; Borcea, C.; et al.: Context-aware migratory services in ad hoc networks. *IEEE Transactions on Mobile Computing*. vol. 6, no. 12. 2007: pp. 1313–1328.

[21] Saaty, T. L.: *Multicriteria decision making: the analytic hierarchy process: planning, priority setting resource allocation*. 1990.

[22] Salehie, M.; Tahvildari, L.: Self-adaptive software: Landscape and research challenges. *ACM transactions on autonomous and adaptive systems (TAAS)*. vol. 4, no. 2. 2009: page 14.

[23] Santa, J.; Pereñíguez, F.; Moragón, A.; et al.: Experimental evaluation of CAM and DENM messaging services in vehicular communications. *Transportation Research Part C: Emerging Technologies*. vol. 46. 2014: pp. 98–120.

[24] Schilit, B. N.; Theimer, M. M.: Disseminating Active Mop Infonncition to Mobile Hosts. *IEEE network*. 1994.

[25] Sheng, Q. Z.; Pohlenz, S.; Yu, J.; et al.: ContextServ: A platform for rapid and flexible development of context-aware Web services. In *Proceedings of the 31st International Conference on Software Engineering*. IEEE Computer Society. 2009. pp. 619–622.

[26] da Silva, C. E.; de Lemos, R.: A framework for automatic generation of processes for self-adaptive software systems. *Informatica*. vol. 35, no. 1. 2011.

[27] Tang, S.; Peng, X.; Yu, Y.; et al.: Goal-directed modeling of self-adaptive software architecture. In *Enterprise, Business-Process and Information Systems Modeling*. Springer. 2009. pp. 313–325.

[28] Wang, X.; Zhang, D.; Gu, T.; et al.: Ontology Based Context Modeling and Reasoning using OWL. In *Percom workshops*, vol. 18. Citeseer. 2004. page 22.

[29] Weyns, D.; Malek, S.; Andersson, J.: On decentralized self-adaptation: lessons from the trenches and challenges for the future. In *Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*. ACM. 2010. pp. 84–93.

Appendix A

Abbreviations

AHP	Analytic hierarchy process
CPU	Central Processing Unit
DL	Description Logic
FUSION	FeatUre-oriented Self-adaptatION
JSON	JavaScript Object Notation
OWL	The W3C Web Ontology Language
OWL-S	The Web Ontology Language for Services
PDDL	Planning Domain Definition Language
QoS	Quality of Service
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SOA	Service Oriented Architecture
SWRL	Semantic Web Rule Language
UDDI	Universal Description, Discovery, and Integration

Appendix B

Curriculum Vitae

Personal Information	
Name	M. Mohanned Kazzaz
Title	Ing.
Nationality	Syrian
Date of Birth	18.05.1983
Contact Information	
Address	Halap Al-jaddeda, C5, Aleppo, Syria
E-mail	mohanned.kazzaz@gmail.com
Phone	+420 774 923 358
LinkedIn	www.linkedin.com/in/mohannedkazzaz
Education	
2011 - present	Faculty of Information Technology, Brno University of Technology doctoral study (PhD.) Computer Science and Engineering http://www.fit.vutbr.cz/∼ikazzaz/
2010	Brno University of Technology, Brno. Czech Republic. Master's degree of Computer Engineering and Networks – Education Recognition.
2001-2007	Faculty of Electronic and Electrical Engineering, University of Aleppo, Aleppo, Syria. Bachelor's degree of Electronic Engineering "Computer Engineering„.
Work Experience	
8/2009 - 10/2011	Software Engineer Employer: Aleppo City Council, Aleppo, Syria.