Restful-based Mobile Web Service Migration Framework

M. Mohanned Kazzaz
Department of Information Systems
Faculty of Information Technology
Brno University of Technology
Brno, Czech Republic
Email: ikazzaz@fit.vutbr.cz

Marek Rychlý
Department of Information Systems
Faculty of Information Technology
Brno University of Technology
IT4Innovations Centre of Excellence
Brno, Czech Republic
Email: rychly@fit.vutbr.cz

***Abstract*—Nowadays, web service provisioning on mobile devices has become a high demand due to the improvement in mobile device capabilities and web service technologies. In this paper we present a RESTful-based framework for Mobile Web service migration and provisioning on both Android-based mobile devices and Java-based stationary devices in P2P wireless network. The proposed Web service migration framework enables deploying, publishing, discovering, provisioning and migrating Web services to satisfy service providers' and Web services' preferences and improve *QoS* performance.**

***Keywords*-service-oriented architecture; RESTful; Self-Adaptation; Mobile-Hosted Mobile Web Services (MHMWS)**

I. Introduction

In Service Oriented Architecture (SOA) [1], Web Services are provided according to specific context which represents the status of the information system and its software and hardware components during design-time and run-time. This context contains services technical specifications, technical requirements and current status. If one or more of these requirements are violated, an adaptation mechanism to heal the system and remove this violation becomes a major need in order to satisfy service requirements by finding the proper providers to host them in the network. Such a mechanism is provided by implementing Service Migration, which is the ability for services to move between service providers.

In our previous papers [2], [3], Service Migration is proposed as a reaction to context changes and enabled by a continuous monitoring mechanism and an automatic discovery process of any inconvenient runtime context (i.e., low memory, high CPU usage, and low battery power level) that violates system components requirements.

A migration can be implemented for a regular system maintenance to survive urgent breakdowns. Moreover, it is a helpful method to ease the process of system configuration during the design time and also the re-configuration process during runtime implementing new versions of the utilized services. The quality of services (QoS), service availability and other services pre-conditions must be considered and guaranteed through the migration. A migration strategy can also perform a customized service migration that allows services to be provided according specific conditions and configurations.

The need to invest the improvements in mobile devices capabilities and wireless technologies motivates our work to extend SOA adaptation to be performed on mobile devices which allows including mobile devices resources as volatile processing power and memory storage units to system resources. The volatile connectivity of mobile devices presents another motivation to implement Web service migration to improve service availability and web service provisioning on mobile hosts.

We selected Android as an implementation platform for our framework on mobile devices as Android OS is the most dominating operating system for smartphones and mobile devices having by the first quarter of 2016 around 85% share of the global smartphone market share according to the statistics company Statista [4]. On the other hand, Android is an open source development platform which is an advantage to demonstrate work and results on such developing research areas [5].

In previous work [6] we presented our SOAP-based framework for Web services migration between stationary devices in SOA. The migration process is controlled by services and service providers preference rules provided by service manufacturers and system administrators. The system is described by an ontology-based context model which is evaluated and reasoned during the Web service migration discovery process. When a migration is found, web service migration decision selection process is launched to choose one migration from the found possible migrations which has the highest weight calculated by the Analytic Hierarchy Process (AHP) [7]. Each migration process is governed by Jena language[1] based rules describing services and service providers preferences during design-time and run-time. The selection process of the best migration decision is made by implementing the AHP multi-criteria decision-making method.

In this work we provide a software framework that enables web service provisioning on constrained resource devices. The proposed framework enables a hyper-platform web service migration between Android based mobile devices and Java-based stationary devices. Moreover, it empowers mobile devices to

[1]https://jena.apache.org/

978-1-5386-1999-5/17 $31.00 © 2017 IEEE
DOI 10.1109/AIMS.2017.18

manage web service migration to increases service availability in the system.

The paper is organized as follows. Section II discusses related work on mobile service provisioning on mobile devices and service migration. In Section III we present the example scenario for the implementation of the proposed web service migration framework. In Section IV, we describe our migration framework for mobile web services. Section VI present our experiments with an example of Web service migration between mobile devices and the efficiency gained over Web service response time and availability by migrating a service to another device through our proposed framework. Finally, we conclude our work in Section VII.

II. Related Work

Several frameworks have been proposed for Mobile-hosted provisioning on mobile devices.

In [8], authors proposed a description based mash-up approach to enable composition and cooperation between of Web services and Web applications hosted on mobile devices. However, web services are only utilized in invoking installed applications on mobile devices enrolled in a planned task.

A mobile-hosted mobile Web service migration framework is proposed in [9]. The framework utilizes SOAP engine to analyse SOAP messages and execute the corresponding service. They utilized a migration policy [10] which only considers Web services preferences without consideration of the service provider preferences. However, their work only considered migration between mobile devices running on mobile windows OS connected via Bluetooth based ad-hoc network. The authors only show the possibility of provisioning and migrating web services on mobile hosts without showing experimental results on the migration decision costs and time measurements.

In [11], authors presented Android-based framework for hosting mobile services using RESTful web services to enable Web service provisioning. The framework utilizes a fuzzy controller to monitor the framework and its devices context, analyses the context and decides which hosted service to be provided.

AlShahwan et al., [12], provided SOAP- and RESTful-based frameworks for distributed execution of mobile Web services. The framework utilizes a FuzzyLogic Module which monitors system resources and activates the offloading strategy by distributing the execution of the web services of an overloaded mobile host on several mobile hosts. Based on the performed tests on both frameworks, the authors found that the REST framework has better performance than the SOAP one in relation to hosting Web services on mobile devices. However, authors did not address the discovery process of service providers but instead they dealt with a pre-defined set of hosts. Moreover no decision making process is proposed to determine the destination host. We present a framework for physical migration of Web service deployable package whereas the authors only use a technique of redirecting the requests to another device hosting the service instead of executing a physical migration of the service to a new host.

In [13], authors provided an agent-based system architecture to improve service availability on mobile hosts by migrating agents onto other mobile hosts as an adaptation against system failures and malicious attacks. However, they did consider service providers preferences in the migration decision process. Contrary to our work, their work only considered a predefined set of participated platforms and did not consider the discovery process of a new participated platforms or mobile agents in the network.

The proposed Restful-based framework enables provider's and service's automatic discovery, system monitoring and adaptive service migration. It supports redistribution of system services between mobile devices running on Android OS as well as stationary devices in wireless P2P network. Moreover, The framework uses an ontology-based description of both devices and web services to support 1) context-awareness in the software system. 2) defining services' and providers' preferences and rules, and 3) service migration decision making process.

III. Example Scenario and Motivation

We propose the following scenario to provide a real life case study for the implementation of our framework. On a tour programme, tourists are subscribing to a travel company service hosted on a mobile phone of a tourist guide. The service provides information about the scheduled tour, information about the sightseeing located around the user position, and also video editing service that allow passengers to edit their videos and publish them on company social webpage. If the tour guide mobile device has a low memory situation, the framework performs a migration of the service onto another device which is located on the travel company bus. Another migration example can happen when passengers leave the bus to a ferry as the service can migrate to a company server located on the ferry and migrated back to the bus after the ferry journey finished and passengers return onboard the bus.

In similar implementation to Facebook internet beaming project[2] proposed to provide internet by solar-powered drones on remote areas, or to Project Loon[3] by Google to connect the world by balloons, we think that our framework can be utilized to provide service migration between drones and balloons due to location preferences or resource vulnerability situation.

IV. Web Service Migration Framework For Mobile and Stationary Devices

In this section we describe the proposed framework for Web service migration running on both standalone and mobile devices. The framework architecture, shown in Figure 1, consists of the controller, context model and a set of migratable Web services. A decentralized service migration decision making process can be performed using our proposed framework by the controller on each emerged service provider.

[2]http://www.telegraph.co.uk/technology/facebook/11895598/Facebook-ready-to-test-internet-drones-using-AI-generated-population-maps.html

[3]https://x.company/loon/

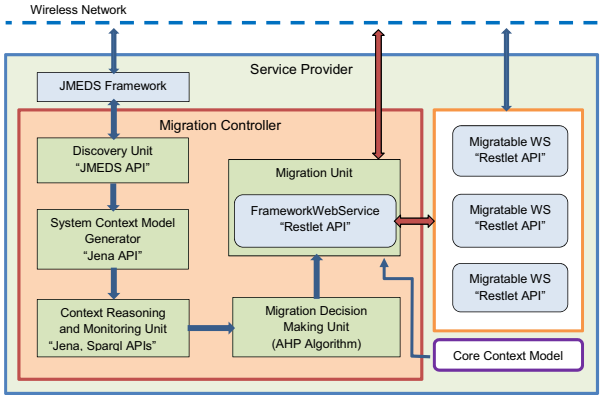

Fig. 1. MOBILE WEB SERVICE MIGRATION FRAMEWORK ARCHITECTURE.

A service provider's Controller periodically performs service and device discovery, system violation monitoring and decision making processes to improve system performance and QoS factor. A controller consists of the following modules:

A. Discovery Unit

The Discovery Unit is responsible for the automatic discovery of the emerging stationary/mobile service providers and their hosted services. This is enabled using the light-weight stack WS4D-JMEDS, that implements the Devices Profile for Web Services (DPWS) build-in services on Java-based and Android platforms.

The discovery unit provides the means to create local list of the discovered devices that can participate in a Web service migration decision making process when a migration necessity is found by means of the Reasoning and Monitoring Module described in Section IV-C.

B. System Context Model Generator

This module is responsible for generating system context which consists of the core context model and all partial context models of services and service providers discovered by the Web Service Provider Discovery Module. The core model is an ontology based description of SOA software architecture components of *Service* and *ServiceProvider* classes including description of their properties, relationships, and subclasses. The partial models define status properties and preference rules of *Service* and *ServiceProvider* instances representing services and their providers currently available in a system, as it is described in Section IV-F.

The Module generates system context model by extracting the context model of each discovered service, that is integrated in the service *WADL* file and by retrieving the context model of each service provider by calling a framework service on the provider that respond with the context model of the related service provider, a description of the aforementioned framework service is provided in Section IV-E.

C. Context Monitoring and Reasoning Unit

After creating the system context model of the discovered service providers and services, the *Controller* performs a context monitoring of the pre-defined rules governing the utilization of the discovered service providers and their services.

Through *Context Monitoring and Reasoning Unit*, the *Controller* uses an ontology based reasoning process using *JENA* reasoner on the context model in order to monitor services and service provider pre-defined rules, find the violated rules and discover the services and service providers affected by the violated rules.

The monitoring process describes the related services as a *CandidateForMigrationService* service and the devices as *CandidateOriginServiceProvider* and *CandidateDestinationServiceProvider*. Moreover, the context reasoning process derives higher level context with new information for the *possibleDestinationProvider* and the *possibleProvidedService* to be hosted on other service providers.

The output of this unit is a list of triple entries stating the candidate service and their origin service provider and the destination service provider. This list of entries is the input of the *Migration Decision Making Unit*.

D. Migration Decision Making Unit

Having a list of possible migrations with different services, origin service provider, and destination service providers, a process to select the best migration to perform is required. We proposed a multi-criteria decision making process using the *AHP* decision-making method. A detailed description of the process of decision making is noted in [6] where one migration process can be performed per time considering the impact on network components following that performed migration.

E. Migration Module

The migration module consists of framework web services required to enable the physical migration of the migratable web service package from a service provider to another. Moreover, this module is responsible for replying requests seeking the context model of its hosting service provider, and for deploying a migratable service on the destination service provider. Having an URI for each service provider and using *Restlet* API, a request to derive the context model of device *X* is:

```
http://{X.IP}:{X.Port}/framework/getProviderContext/json
```

and the request to migrate a service from *X* to *Y* and is:

```
http://{Y.IP}:{Y.Port}/framework/download/{X.IP}
/port/{X.Port}/temp/{X.TempFolder}/service/{ServiceWAR}
```

F. Context Model

The Web service migration is performed as a reaction to violation in the pre-defined rules of the services and/or service providers presented in their context models. The context model of each service and service provider is an ontology based semantic description stating their properties/configurations and pre-conditions. Using an ontology based context model of

```
{"name":"S",
"type":"MigratableService",
"properties":    {"propertyName":"ServicePriority",
                  "propertyValue":"50",
                  "propertyType":"INT",
                  "criteria":"ServicePriorityCriteria"},
"rules":"[SPreference: (?service rdf:type core:MigratableService),
        (?origin rdf:type core:CandidateOriginServiceProvider),
        (?destination rdf:type core:CandidateDestinationServiceProvider),
        (?origin core:provides core:S),
        (?destination core:hasProperty ?property),
        (?property rdf:type core:CPUUsage),
        (?property core:propertyValue ?v1),
        le(?v1, \"45\"^^http://www.w3.org/2001/XMLSchema#int) ->
        (core:S core:possibleDestinationProvider ?destination)]"}
```


Fig. 2. AN EXAMPLE OF THE PARTIAL CONTEXT MODEL OF SERVICE *S*.

system components provides a common understanding between service providers and consumers of services and service providers preferences and configurations which enables system adaptivity and as a result guarantees service availability and eases system maintenance.

Considering results of [14], [15], we choose to use *JSON* based representation instead XML-based representation of services and providers context models and in message exchange format between mobile devices due its positive impact on system performance and message exchange especially when used on resource constrained devices. A context model contains *type* and *name* elements to specify the type of the component and its given name in the network. The type can have value of *MigratableService*, *FrameworkService*, or *ServiceProvider*. Also a model contains a *hasProperty* element, to describe a property of a service (i.e., Service Priority) or a service provider configuration (i.e., Battery Life Time) as a Property element which is described by two sub elements *propertyType* and *propertyValue* stating the type and value of the related property respectively. The *rules* section includes Jena-based language rules each one representing a preference rule of a service or a service provider that should be satisfied during design-time and run-time. The *Criteria* section holds the criteria considered by the *Property* during the migration decision making process by AHP. In Figure 2, we demonstrate the *JSON* based schema of the context model of a *MigratableService* called *S* that we use during our experiment and describe in more details in Section VI.

V. Technical Description

In this section we provide implementation description of the proposed framework for Web service migration and provisioning on mobile and stationary devices.

We adopt the Representational State Transfer (REST) [16] architectural style for SOA design due to the lower message payload of REST framework than SOAP framework which makes REST more suitable for mobile devices [17]. For this reason, we utilize *Restlet* APIs[4] to implement our RESTful Web services for the migration framework as Restlet APIs supports both JAVA and Android platforms which enables to run web services on stationary devices presented by Java based host and also on mobile devices presented in Android-based host without any need to modify its source-code.

For the HTTP server we choose I-Jetty project[5] which is a lightweight HTTP server for Android. Each HTTP server hosts one grounding service *FrameworkWebService* which has the required functionalities to package, send and receive Web service WAR packages between system devices. First, The *getWar* method packages the service in a WAR format file. Second, The *getContext* method returns the service provider context model. Last, the *downloadWar* method that is called to download and deploy the Web service WAR package to the destination service provider. Each method are presented in a class file and attached to the device URL and called via http requests to its host using *Restlet* APIs.

On the other hand, each running service has its context model which can be retrieved by its URL. The service OWL/RDF context model is attached to the service's Web Application Description Language (WADL) file by inheriting the *WadlApplication* Restlet class in the main service *Application* class. Then the WADL context model is provided through implementing the *getApplicationInfo* method and stored as an instance of *DocumentationInfo* class of Restlet framework.

The process of deriving possible migrations is performed through implementing JENA reasoner on the devices' and services' models with respect to their defined JENA rule preferences. We integrated *Androjena* APIs [6] for Android to enable JENA based reasoning for the generated system context model. Then the migration with the highest weight from the proposed set of migrations is chosen to be executed. We provided a detailed description of the AHP algorithm to calculate the weight of the possible migration based on defined criteria list in [6].

VI. Experiment and Results

We propose a case study of service migration between two mobiles to demonstrate the improvement on the *QoS* gained by the migration in addition to the framework impact on CPU Usage and Battery Level of the mobile device running the migration process.

Two devices are used as service providers in this experiment. Both are Huawei Y560-L01 mobile phones namely: *Destination* and *Source*, with 1.1GHz CPU frequency, 1GB RAM and running Android 5.5.1 APIs. The status properties of both providers are listed in Table I including their preference rules. *Destination* Provider has CPUUsage property equals to 40% and a preference rule defining the least *ServicePriority* of services that can be hosted on *Destination* by 50%. *Source* provider has 85% of *CPUUsage* and no preference rules.

[4]https://restlet.com/
[5]https://github.com/jetty-project/i-jetty
[6]https://github.com/lencinhaus/androjena

TABLE I
VALUES OF THE STATUS PROPERTIES AND PREFERENCES OF THE MOBILE SERVICE PROVIDERS.

Provider	CPUUsage	Preference Rule
Source	85%	NA
Destination	40%	ServicePriority>=50%

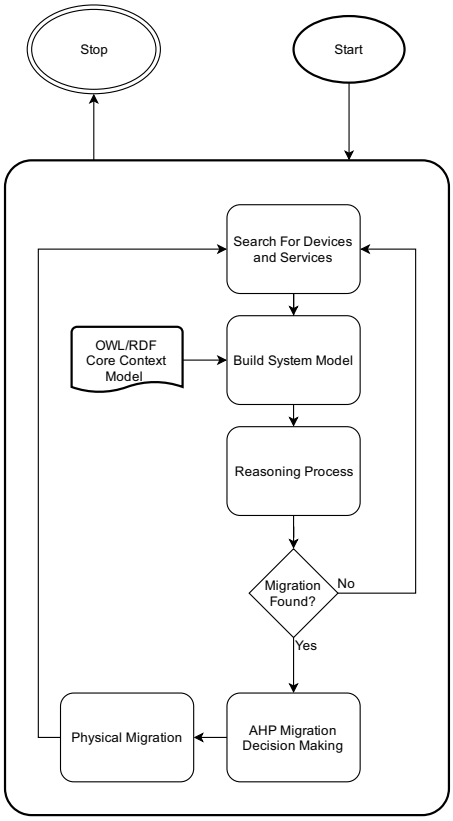

Fig. 3. ILLUSTRATION OF THE MIGRATION PROCESS STEPS.

The subject Web service *S* is a video transcoding service which converts *AVI* video files into *FLV* format. *S* has *ServicePriority* property of value 50% and has one preference rule that allows it to be migrated only to service provider with $CPUUsage < 45\%$. *ServicePriority* and *CPUUsage* both are sub properties of the *Property* Class in the context model. *ServicePriority*'s states the priority of the service by a value of [0, 100] while *CPUUsage* is a service provider property stating the percentage of the device processor in use.

First, we publish *S* on *Source* and we call service *S* to convert an AVI sample video file of 17.1 *MB* size and measured the response times of for service *S* for10 times using the Advanced REST Client API testing tool[7]. The video conversion method of *S* is called the following URI: http://{Host.IP}:{Host.Port}/S/convert/{VideoFilePath}.

[7]https://advancedrestclient.com/

Later, we connect the second mobile service provider *Destination* and run the migration framework application on both mobile phones. The migration process, demonstrated in Figure 3, begins when the *Discovery Unit* on *Destination* by starting the *Search* process to discover the connected devices and services in the network.

When *Destination* discovers *Source* and its hosted service *S*, the framework *System Context Model Generator* on *Destination* fetches and adds the partial context models of *Destination*, *Source* and its service *S* to the core context model including their preference rules found in the partial context model of them.

After that, the *Context Monitoring and Reasoning Unit* utilizes JENA reasoner to reason the generated system model and query it for any derived migration suggestions matching the services and devices preferences. The suggested list of possible migration is ranked through the *Decision Making Unit* which selects the migration with the highest priority to be performed.

In this example, *Destination* is suggested to host *S* as a new service provider based on its preference where it has only 40% CPUUsage. On the other hand, *S*'s *ServicePriority* of 50% satisfies *Destination* preference which permits only services with $ServicePriority \geq 50$ to be migrated to *Destination*.

Similarly, measurements of response times of Service *S* are made to convert the same AVI file while *S* is hosted on *Destination*. The measurements show that the average response time of service *S* is 48.6 Seconds when hosted on *Destination* while it is around 134.4 Seconds when hosted on *Source*. The results show that by the migration of Service *S* from Source to be hosted on *Destination* that has CPU Usage less than 45%, the proposed migration framework provides the mechanism that achieves improvements on Service *S QoS* measured by its response time.

We configured this example to be repeatedly executed by the framework Controller of *Destination* for an hour in order to investigate the migration process time and the impact of running the implemented framework on device resources.

a) Migration Process Time: During this test, the controller perform the migration process 634 times. We observed that the average time to perform the migration process from *Source* to *Destination* is 4.836 Seconds. By excluding the time to download and deploy Service *S* WAR file of 2.30 MB on *Destination*, the time spent to take the migration decision is 0.576, 0.449, and 1.309 Seconds at its Average, Minimum and Maximum value respectively. Based on this measurements, we see that the proposed framework enable a seamless adaptation in SOA to redistribute system components.

b) CPU Usage Consumption: We collected the CPU usage samples consumed by the framework application. Figure 4 presents the percentage of time for the CPU usage of the framework during this example. 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 value respectively.

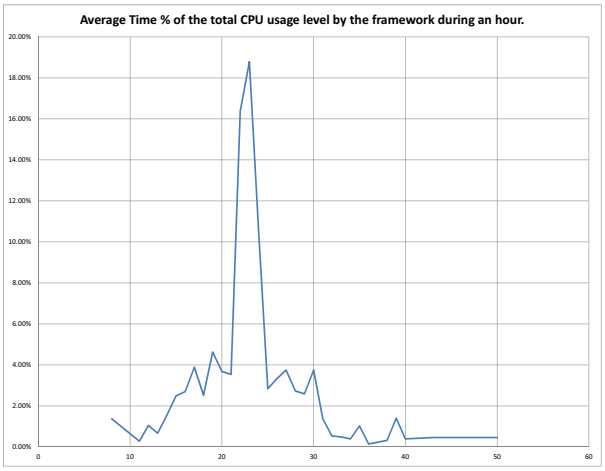

Fig. 4. FRAMEWORK TOTAL CPU USAGE BY ITS AVERAGE LASTING TIME DURING AN HOUR.

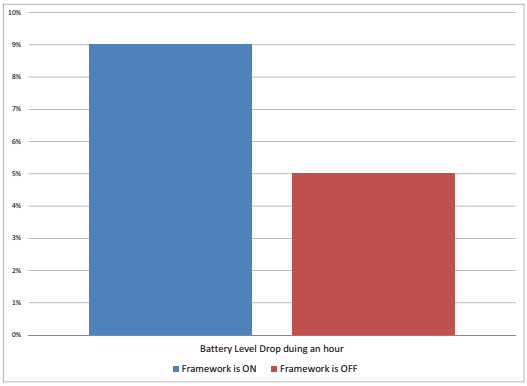

Fig. 5. BATTERY LEVEL DROP DURING THE EXPERIMENTS.

c) Battery Consumption: To investigate battery consumption by the framework, we collected Battery level drop while the migration framework during the test and compare it with battery level drop when the framework is not running. Presented in Figure 5 the results show that the battery level is dropped in 4% during the test more than the battery drop when the framework is OFF.

VII. Conclusion

We demonstrate our Restful-based Web service migration framework for dynamic relocation and provisioning of Web services on mobile and stationary devices. The experimental results shows the proposed framework impact on device resources and efficiency gained by utilizing the proposed Web services migration framework to assure services' and devices' preferences and improve QoS in SOA by enabling self-adaptation on mobile devices in P2P network. Our future work will focus on testing the framework on a real case study and measuring the impact of the amount of joined devices on framework performance, more specifically during system context model generating and decision making processes.

Acknowledgements: This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II) project "IT4Innovations excellence in science" LQ1602 and by BUT internal project "ICT tools, methods and technologies for smart cities" FIT-S-17-3964.

References

[1] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*. Upper Saddle River, NJ, USA: Prentice Hall PTR, Aug. 2005.

[2] M. M. Kazzaz and M. Rychlý, "Web service migration with migration decisions based on ontology reasoning," in *Proceedings of the Twelfth International Conference on Informatics-Informatics*, 2013, pp. 186–191.

[3] M. M. Kazzaz and M. Rychlý, "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.

[4] Statista, Inc., "Global mobile OS market share in sales to end users from 1st quarter 2009 to 1st quarter 2016," http://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems, 2016, accessed: 2016-08-14.

[5] K. S. Wagh and R. C. Thool, "Performance analysis of mobile web service provisioning on different mobile host," in *2014 Annual IEEE India Conference (INDICON)*. IEEE, 2014, pp. 1–5.

[6] M. M. Kazzaz and M. Rychlý, "Web service migration using the analytic hierarchy process," in *2015 IEEE International Conference on Mobile Services (MS)*. IEEE, 2015, pp. 423–430.

[7] T. Saaty, *The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation*. RWS, 1990.

[8] K. Prutsachainimmit, P. Chaisatien, and T. Tokuda, "A mashup construction approach for cooperation of mobile devices," in *International Conference on Web Engineering*. Springer, 2012, pp. 97–108.

[9] Y.-S. Kim and K.-H. Lee, "A lightweight framework for mobile web services," *Computer Science-Research and Development*, vol. 24, no. 4, p. 199, 2009.

[10] K. Yeon-Seok and L. Kyong-Ho, "An efficient policy establishment scheme for web services migration," in *International Conference on Convergence Information Technology*. IEEE, 2007, pp. 595–600.

[11] K. Wagh and R. Thool, "Mobile web service provisioning and performance evaluation of mobile host," *International Journal on Web Service Computing*, vol. 5, no. 2, p. 1, 2014.

[12] F. AlShahwan, F. Carrez, and K. Moessner, "Providing and evaluating the mobile web service distribution mechanisms using fuzzy logic," *Journal of Software*, vol. 7, no. 7, pp. 1473–1487, 2012.

[13] Y. Zuo and J. Liu, "Mobile agent-based service migration," in *2015 12th International Conference on Information Technology-New Generations (ITNG)*. IEEE, 2015, pp. 8–13.

[14] C. Rodrigues, J. Afonso, and P. Tomé, "Mobile application webservice performance analysis: Restful services with json and xml," in *International Conference on ENTERprise Information Systems*. Springer, 2011, pp. 162–169.

[15] A. Sumaray and S. K. Makki, "A comparison of data serialization formats for optimal efficiency on a mobile platform," in *Proceedings of the 6th international conference on ubiquitous information management and communication*. ACM, 2012, p. 48.

[16] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, Irvine, 2000.

[17] K. Wagh and R. Thool, "A comparative study of soap vs rest web services provisioning techniques for mobile host," *Journal of Information Engineering and Applications*, vol. 2, no. 5, pp. 12–16, 2012.