Selection algorithm of contextual software entities for composing adaptive mobile applications

Afrah Djeddar, Abdelkrim Amirat, Hakim Bendjenna, Philippe Roose

Selection algorithm of contextual software entities for composing adaptive mobile applications

Afrah djeddar
Laboratory of Mathematics, Informatics and Systems (LAMIS),
Department of Mathematics and Computer Science,
University of Larbi Tebessi,
Tebessa, 12002, Algeria
afrah-djeddar@hotmail.fr

Hakim Bendjenna
Laboratory of Mathematics, Informatics and Systems (LAMIS),
Department of Mathematics and Computer Science,
University of Larbi Tebessi,
Tebessa, 12002, Algeria
hbendjenna@gmail.com

Abdelkrim Amirat
Laboratory of Informatics and Mathematics (LIM),
Department of Mathematics and Computer Science,
University of Souk Ahras,
Souk Ahras, 41000, Algeria
abdelkrim.amirat@yahoo.com

Philippe Roose
LIUPPA Laboratory,
Department of IUT de Bayonne,
University of Pau et de pays de l'adour,
Pau, 64000, France
Philippe.roose@iutbayonne.univ-pau.fr

***Abstract*— the development of customized mobile applications basing on the composition mechanism (i.e. using existing software entities) has received a lot of attention in the last couple of years. The mobile devices heterogeneity shows that the portability requirements play an important role in the mobile applications developpement domain. Otherwise, mobile applications strongly depend on the execution environment features. Thereby, in order to make sure the correct deployment and the proper functioning of the composite mobile application it is necessary to ensure that their constituents are adaptable to the current context of the mobile device. To cope with this issue and due to the fact that several software entities can be used to implement the identified requirements for a desired mobile application, we propose in this paper a context-driven selection algorithm that aims at selecting the adaptive software entities among all corresponding ones. Also, it targets to determine the different possible composition paths to build customized mobile applications. To achieve this objective, we propose ontology based descriptions to define the context of the corresponding software entities and the execution environment.**

Keywords*— *Mobile applications, Selection algorithm, Context, Ontology, Composition path.

I. INTRODUCTION

Since the recent years, the mobile technology is becoming more and more widespread, and researches in this domain are increasingly so much needed [1]. The available mobile devices in the market such as smartphones and tablets are characterized by several heterogeneous features [2] that are important: battery life, storage capacity, input modes, etc. Furthermore, the massive adoption and the use of the mobile devices to perform our tasks in the daily life is just a show of their growth and of the increasing demand of new mobile applications. So and, in order to meet user's requirements and taking advantages from the existing services, composing new applications submits to several constraints of the mobile environment. The mobile devices heterogeneity and the limited resources offered by them (i.e. the limited computing power, the limitations of networks' connection, small storage capacity, the screen size, etc.) [4] create difficulties and gives to several challenges with composing mobile applications. The most important one is the portability of the obtained applications on the heterogeneous mobile devices [3]. Since it is not economically viable to produce software applications for a few devices (i.e. limit the usability of mobile applications) [5]. To cope with this issue, the mobile developer will have to create several versions of the desired mobile application, where each of them will be specific for a specific mobile environment (i.e. specific context). Thereby, so that to compose the adaptive mobile applications; we always have to deal carefully with the different mobile devices' features precisely when selecting software entities constituents.

Generally, a selection technique is based on certain criteria or metrics. In our research work, we propose to classify these criteria inside two categories: (a) the execution constraints (such as the battery level, the memory size, the availability of some devices (e.g. Wi-Fi, GPS), etc.), so that we select the software entities which can be deployed correctly and function properly on the target device. This kind of criteria represents the necessary conditions for the entity will be able to perform its task. The category (b) is the quality characteristics (such as: the network signal strength in dependence on the mobile device network, the response time in dependence on the CPU speeds, etc.), so that to ensure a better performance and a good quality of the obtained application. Several authors have addressed articles and viewpoints on the selection task in the mobile domain [6] [7] [9] [10], where most of them have focused basically on quality characteristics to perform the selection task, just with for an aim to achieve a better application performance (e.g. network performance [6]). Even though, several important facts should be also taken into consideration, especially and in an important way to assure the correct deployment and the proper functioning of the application such as: the memory size, the screen size, the battery level. For this

978-1-4673-7689-1/16/$31.00 ©2016 IEEE

reason and so to achieve our goal, which is to provide a competent and efficient composite mobile application that works properly in a specific mobile device, we propose an ontology-based description for the context of each of target mobile device and adequate software entities. On the basis of these ontology models, we suggest a selection algorithm which aims to select the contextual software entities, and also to determine the different possible composition paths to obtain a specific mobile application.

The reminder of this paper is organized as follows: Section 2 gives a discussion about the related works. Then, in Section 3, we introduce our research work. In a first place, we introduce the proposed ontology models to define mobile devices' dependent features, and the software entities' dependent execution constraints. After that, we explain our selection algorithm to identify the contextual software entities constituents and also to determine the possible contextual composition paths. This is so to help to compose adaptive mobile applications. Finally, with Section 4, we give a conclusion of our research work.

II. RELATED WORKS

Throughout this section, we present some existing selection approaches in the mobile environment. In [6], the authors have proposed a cross large approach of the service selection. They focused on the way in which the client and the server are paired together, taking into consideration network performance aspect. This approach allows client to switch the better servers as network topology changes (i.e. it groups client with nearby servers), based-on network ad-hoc routing mechanism. They also showed also that the effective selection can improve the network throughput by up 40%. While, the authors in [7] showed that selecting the service provider with the lowest hop-count (i.e. choosing the nearest service providers) is not sufficient to achieve a better system performance. At this level, they proposed an integrated selection which takes a more comprehensive evaluation of each service provider, including the capability of the service provider, the risk of service failure in the network, response time, etc. Authors in [10], proposed the discovery protocol, which adopts two metrics to select a service instance. These metrics are: the hop-count between service requestor and service provider, and the capacity of service (i.e. CoS metric expresses the nominal capacity of a service instance). The authors in [9] showed that the achieved localization of the communications is a factor which leads to improve the network performance. They demonstrated through simulations that triggering re-selection of servers, after detecting changes in network topology, is very reliable in turning down congestion and delays. The approaches mentioned above consider the impact of the service selection mechanism on the client-side without taking into consideration the device target's features. To deal with this issue, several works were developed. Because of the fact that the mobile ad-hoc network devices are characterized by a very resource-constraint, and also the resources situation may change quickly; authors in [8] discuss the service selection for the composite service. They performed the selection task according to the service dependent information and the device dependent information (battery level, current load, network signal strength, etc.). Meanwhile, in [3], a software infrastructure called AppSpotter, which allows the dynamic and automated composition of the software components of mobile application is proposed. In this AppSpotter, a component named "component selector" retrieves from the SPL assets (software component stored) the software components that meet with or fill a set of criteria, which are represented with the mobile device features: platform, screen, input, keyboard and accelerometer. Selection task in both [8] [3] is driven with mobile device features but selection criteria are not the same ones. The following table (cf. Table 1) gives a comparison of the mentioned works according to a set of criteria. These criteria are divided into two categories as explained previously. The first one represents the device target features, while the second one includes a set of characteristics that represent the impact of selection mechanism on the client-side.

TABLE 1. SYNTHESIS OF SELECTION APPROACHES

	Execution Constraints					Quality Characteristics			
	Battery Level	Memory size	Screen size	platform	Wireless connectors	Network performance	Response time	Hop-count	CoS
[6]	-	-	-	-	-	*	-	-	-
[7]	-	-	-	-	-	*	*	*	*
[10]	-	-	-	-	-	-	-	*	*
[9]	-	-	-	-	-	*	-	-	-
[8]	*	-	-	-	-	*	-	-	-
[3]	-	-	*	*	-	-	-	-	-

Most of these selection approaches focus only on quality characteristic. So, no one of them treats some important criteria such as: the availability of the needed wireless connectors or the memory size during the selection task. If a software entity needs to a GPS or Wi-Fi device to perform its task, it is necessary to first make sure that the target device has this wireless connector, otherwise it does not work. Also, the size of the software entity must not exceed the reminder capacity storage of the target device in order to ensure its correct deployment. Our work in this paper comes as a solution to bridge this lack. It aims to take into consideration all the execution constraints to perform the selection task.

III. OUR CONTRIBUTION

Our approach discusses the selection of the adaptive software entities in order to build a specific composite application for the mobile environment. In this section, we firstly aim at modeling the context of the composition. This last reflects the specification of the contextual information of the constituent software entities (i.e. corresponding software entities that can be used to implement the different required functionalities), including its role and all the necessary conditions for its execution. Therefore, all the characteristics associated with the mobile device in which the desired mobile applications will be deployed, and also there execution state. We have chosen as a description language to model the composition context "ontology" [11]. This later provides a semantic description which allows the determining of the behavior of any software entity.

Furthermore, due to the fact that a software entity may be implemented either with a service or a component, etc. the ontology provides a standard description of these heterogeneous entities. So, it supports interoperability of the heterogeneous software entities as a common vocabulary. After, by means of these context descriptions, we propose a selection algorithm that aims to select contextual software entities.

This is to compose an adaptive mobile application for a specific environment, and also to provide all the different possible valid composition paths. Ontologies are intended to represent the contextual information of the software entities, and the target mobile device being interpreted by both the developer of the mobile application and our proposed selection mechanism. The extraction of the needed information will be through the XML descriptions provided by these ontologies.

A. *Ontology based context modeling*

Concerning the software mobile, a successful composition strongly depends on the application environment. Thus, to ensure the correct deployment and the good functioning of the composite mobile applications it must take into account the different context information of the mobile device in which it will be deployed when performing the composition task. Thereby, it is necessary to identify the different deployment environment features. For this purpose, we propose ontology-based description to model the current context of the deployment environment as illustrated in Fig.1. Actually, we can find different devices with different capabilities (i.e. heterogeneous features). Furthermore, mobile device features are temporal and can be changed at any time (e.g. battery level of the user's device becomes low; Wi-Fi is disabled, etc.).Thereby, the context-aware mobile device is represented with all the hardware characteristics, and also their current state.

In our research work, we propose to model this contextual information as follows:

- *Deployment context:* represents the hardware characteristics of the mobile device.
- *Execution Plateform :* represents the plateforme's type installed in the mobile device.
- *Execution context:* represents the current state of the available devices in the user's mobile.

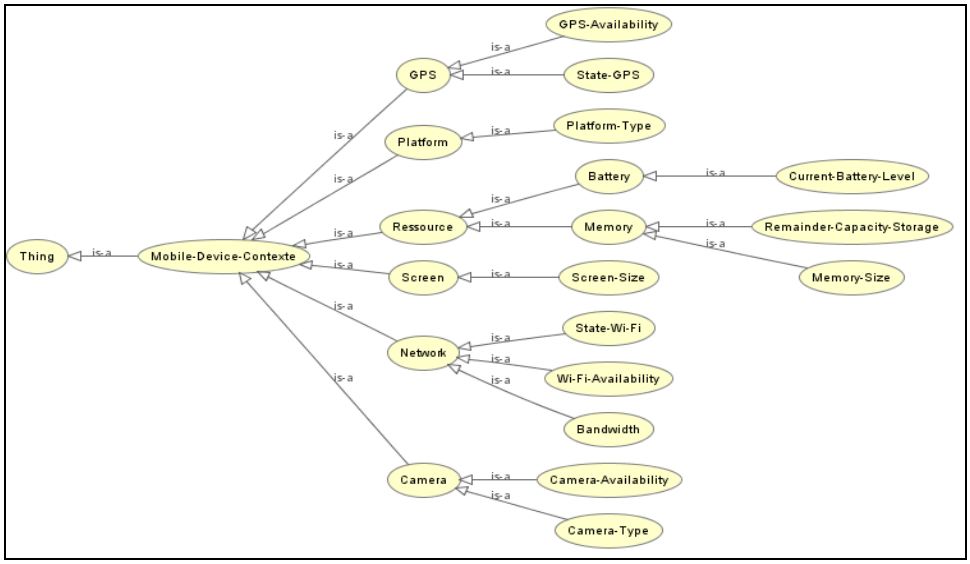

Fig. 1. Mobile device context ontology.

Let us assume that we have a constituent software entity which needs a Wi-Fi service and 3% of energy (i.e. the execution of this software entity implies that the mobile device battery will lose 3% of its capacity): it can perform its task, and the mobile device on which the desired composite mobile application will be deployed has this Wi-Fi service. Nevertheless, its current state is disabled and the mobile device current battery level is 2%. In this case, this software entity would not function correctly while the Wi-Fi is disabled and the battery level is low. Still another example, if we find a constituent software entity which needs to 12Mb of memory and the remainder capacity storage of the mobile device is lower than the needed one, this software entity will not be deployed successfully.

Several software entities may be used to implement the same desired functionality. Therefore, these software entities are functionally equivalents but may vary in several non-functional aspects. Fundamental characteristics related to the software entity are represented with: *Implementation Type* (i.e. service, component, or application), and *Role*. However, in order to select the ones that are conform to the mobile device (i.e. contextual software entities), we propose to associate each entity with a specific *execution profile*. This *execution profile* contains all the non-functional aspects that represent the necessary conditions for its execution (e.g. consumption energy, required platform, needed screen size, etc.). In this way, *Execution profiles* contain various metrics associated with values. These metrics are the criteria that we propose to ensure the proper deployment and the good functioning of the software entity.

Thereby, software entity description concerns not only functional characteristics but also must be rich enough to provide scope awareness to can performing a better selection task. The ontology model presented in Fig.2 describes the functional description and all the necessary conditions for the execution of a software entity.

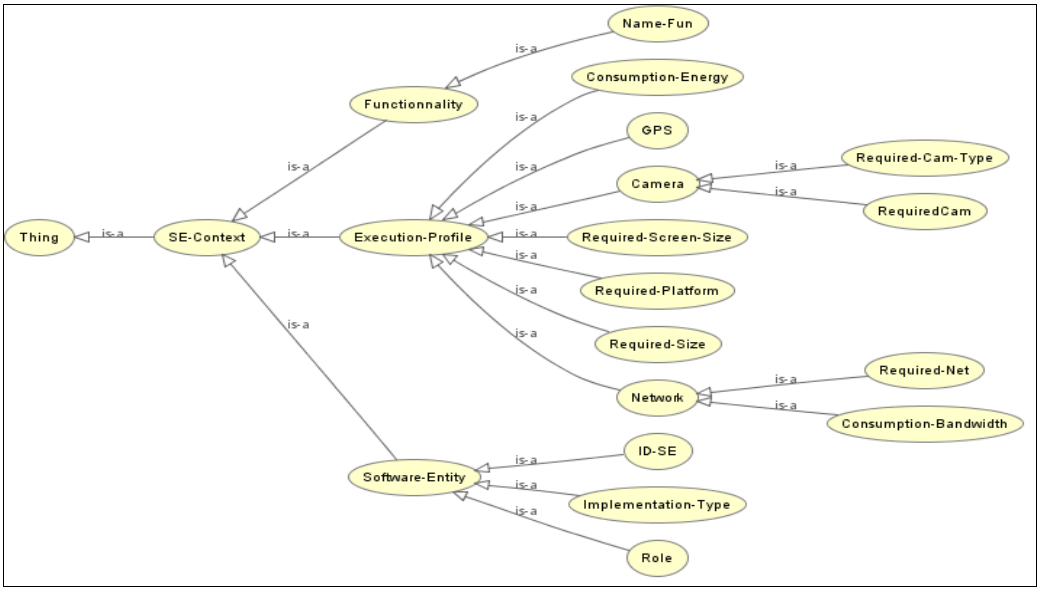

Fig. 2. Software entity context ontology.

By way of example, we assume that we have a desired functionality *Fun1: ReadBarcode* which aims to read the barcode written in the image of a product. This *Fun1* can be implemented with three software entities:

- *SE1*: Service (Remote barcode recognition).
- *SE2*: Component (local barcode recognition).
- *SE3*: Service (http://searchupc.com/).

Each of these software entities must have a set of execution constraints (i.e. list of features required to use a software entity).

For example, *SE1* needs a camera of a fixed-focus type, an activated Wi-Fi device, and 3% of energy to be able to function correctly. Also, it needs to 10Mb memory space to can be deployed properly in the mobile device, etc. Meanwhile, the software entity *SE2* needs to a camera device of an auto-focus type and 1% of energy to function correctly. Also, it needs to 15Mb of capacity storage to be deployed properly in the mobile device, etc. Here, we have not presented the real values but just the relative differences between the non-functional aspects. The contextual description of the *SE1* is represented with the following *XML description* which conforms to the proposed *Software entity context ontology*.

```
<owl:NamedIndividual rdf:about="&untitled-ontology-5;Se1Context">
    <IdSe rdf:datatype="&xsd;int">1</IdSe>
    <NameFun rdf:datatype="&xsd;string">Fun1</NameFun>
    <ImplementationType rdf:datatype="&xsd;string">Service</ImplementationType>
    <RequiredStorageSize rdf:datatype="&xsd;string">10 Mb</RequiredStorageSize>
    <RequiredScreenSize rdf:datatype="&xsd;string">---</RequiredScreenSize>
    <ConsumptionEnergy rdf:datatype="&xsd;string">3%</ConsumptionEnergy>
    <RequiredPlatform rdf:datatype="&xsd;string">Endroid</RequiredPlatform>
    <RequiredCam rdf:datatype="&xsd;string">True</RequiredCam>
    <RequiredCamType rdf:datatype="&xsd;string">Fixed-focus camera</RequiredCamType>
    <RequiredNet rdf:datatype="&xsd;string">True</RequiredNet>
    <consumptionBandwidth rdf:datatype="&xsd;string">10 Mb/s</consumptionBandwidth>
    <Role rdf:datatype="&xsd;string">Read the barecode from the image of the product</Role>
    <GPS rdf:datatype="&xsd;boolean">False</GPS>
</owl:NamedIndividual>
```

Fig. 3. Software entity ontology model.

B. *Contex-Driven Selection Algorithm*

Our proposed algorithm is intended to perform contextual software entities selection after receiving the list of corresponding software entities that match the described user's requirements (see pseudo code). The selection mechanism is based-on both the software entity dependent information and the mobile device dependent information. The mobile device features and their current state may be inferred by the mobile system or obtained by the developer. It will be saved in through an XML file according to the proposed mobile device context ontology. Therefore, an author file conforms to SE context ontology is required to describe all the functional and the non-functional characteristics of the software entities that satisfy the desired functionalities. Selection mechanism aims to retrieve from the SE list path (i.e. set of appropriate software entities) of each functionality belongs to the Fun list path (i.e. set of identified functionalities) those that meet a set of criteria (cf. Fig. 4). Thereby, from the whole of the provided SE, only the software entities that are compatible with the device context (i.e. contextual SE) will be selected to integrate and assemble the desired mobile application. We associate this ability to the *AdaptabilityTest* method which aims to compare the SE execution profile with the mobile device features in order to select the SE that are conform to the mobile device. It returns true value if all the SE execution constraints are satisfied. This method reflects a set of influence rules which aim to extract the needed contextual information from proposed ontology models, in order to perform the comparison task. The following listing represents an example of an inference rule which serve to verify the compatibility of SE (y) which implements the functionality Fun (x) with the target mobile device for the camera feature.

```
PREFIX MD-Context: URI of mobile device ontology model
PREFIX SE-Context: URI of software entity ontology model
SELECT ?Type ?Availability ?RequiredType
WHERE{?x SE-Context: NameFun "x" --the name of the Fun
       ?x SE-Context: ID-SE   "y" --the ID of the SE
       ?x MD-Context: CameraType ?Type
       ?x MD-Context: CameraAvailability ?Availability
       ?x SE-Context: RequiredCam ?RequiredCam
       ?x SE-Context: RequiredCameraType ?RequiredType}
If
EXACT: availability=RequiredCam and Type=RequiredType
Else DISJOINT;
```

Listing 1. Influence Rule

If we assume that we have a mobile device which its current context is: Has-GPS is GPS-disabled, Has-Wi-Fi is Wi-Fi-enabled, Has-Fixed-Focus-Camera, Battery Level is 50 %, and Free Storage capacity is: 500 Mb. However, the result of the selection task according to the example presented in the previous section will be: Contextual SE-Fun1 ←{*SE1*, *SE3*}. Thereby, all specified concepts dedicated to be compared with the mobile device concepts concerning the *SE1* and *SE3* are equivalent (e.g. the *SE1* needs to a fixed-focus camera and the mobile devise in which it will be deployed have a camera of fixed-focus type). Meanwhile, the *SE2* is not adaptable to the current context of the target mobile device because it needs an auto-focus-camera to be able to perform its task while the mobile device has just a fixed one. The proposed selection algorithm ensures that any functionality must have at a minimum one contextual *SE* to implement it. Furthermore, it is designed to provide, after the selection task, the possible composition paths to obtain an adaptive mobile application as illustrated in Fig. 4.

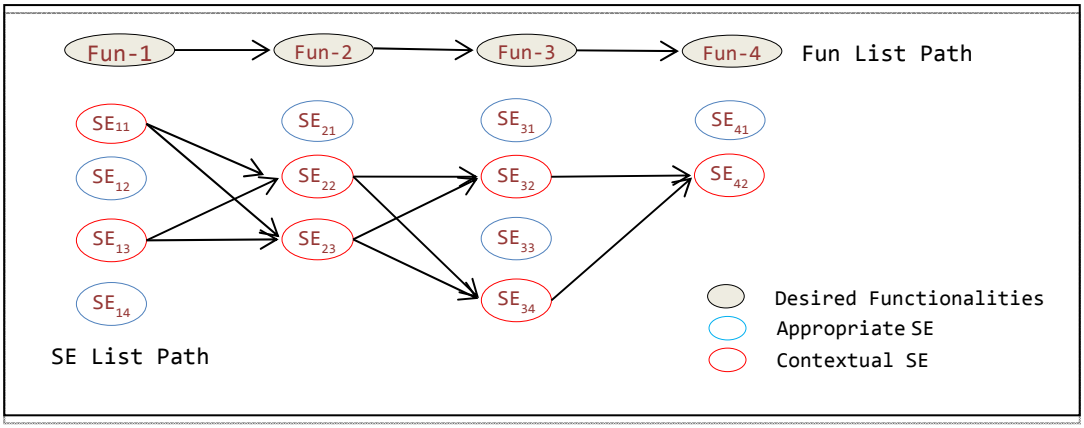

Fig. 4. Selection mechanism.

Pseudo code: Selection Algorithm

```
1.  Fun-List-Path: Desired functionalities.
2.  SE-List-Path: Best suited software entities.
3.  ST1, ST2, ST3: Stack.
4.
5.  --Selecting context-aware SE method—
6.
7.  Boolean Adaptability-Testing (SE) {
8.        Compared SE.execution-profile with mobile-device-characteristics }
9.
10. --Discovery of valid composition path--
11. For each SE € SE-List-Path of the first functionality do
12.       If Adaptability-Test (current SE) return true value then
13.          {AddToGraph (empty node, current SE)
14.          Push (current SE, ST1)}
15. If ST1 is empty= false then
16. {Repeat
17.         For each Fun € Fun-List-Path do ---starting from the second functionality---
18.             For each SE € SE-List-Path of the current functionality do {
19.                   If adaptability-Test (current SE) return true value then
20.               {AddToGraph (Top of ST1, current SE)
21.                 If current Fun is not the last
22.                       {Push (current SE, ST2)}
23.                 Else {Push (current SE, ST3)}    }     }
24.      If (ST2 is empty= false and current Fun is not the last) then
25.         {Pop (Top of ST1),
26.          While ST2 is empty= false do
27.                {Push (Top of ST2, ST1),
28.                 Pop (Top of ST2)}     }
29.      Else {add contextual SE in the list path of the current functionality   }
30.      If (current Fun is the last & ST3 is empty=true) then
31.         {Pop (Top of ST1)}
32.      Else {add contextual SE in the list path of the current functionality   }
33.         Until ST1 is empty = true}
34. Else   {add contextual SE in the list path of the first functionality}
```

IV. Conclusion

In this paper we have proposed a selection technique for composing adaptive mobile applications. We have adopted the ontology notation to specify and model the composition context. We also have proposed two ontology based descriptions, of which the first one aims to describe any information that can be used to characterize the situation of the mobile device; while the second one reflects the specification of the contextual information of the constituent software entities. Based on these descriptions, we have proposed a selection algorithm that aims to select the adaptive software entities and also to discover all the possible composition paths using these selected entities.

As future work, we intend to select the most appropriate contextual software entity in the case of finding several ones for the same functionality. The selection will be carried out according the composition cost of the software constituents in terms of adaptability.

Our selection mechanism focuses basically on a set of criteria that aims to ensure the correct deployment and the proper functioning of a composite mobile application, which is not the case for the existing selection approaches. Currently, a selection mechanism which aims to tackle the execution constraints and the quality characteristics at the same time is still a solution that is not available.

REFERENCE

[1] Economides, Anastasios A., and Amalia Grousopoulou. "Students' thoughts about the importance and costs of their mobile devices' features and services."Telematics and Informatics 26.1 (2009): 57-84.

[2] Zhang,X., et al., Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing. Mobile Networks and Applications, 2011. 16(3): p. 270-284.

[3] Rosa, Ricardo Erikson VS, and Vicente F. Lucena Jr. "Smart composition of reusable software components in mobile application product lines."Proceedings of the 2nd International Workshop on Product Line Approaches in Software Engineering. ACM, 2011.

[4] Hofer, T., Schwinger, W., Pichler, M., Leonhartsberger, G., Altmann, J., & Retschitzegger, W. (2003, January). Context-awareness on mobile devices-the hydrogen approach. In System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on (pp. 10-pp). IEEE.

[5] Alves, V., Santos, G., Calheiros, F., Nepomuceno, V., Pires, D., Costa Neto, A., & Borba, P. (2006). Beyond code: Handling variability in art artifacts in mobile game product lines.

[6] Varshavsky, Alex, Bradley Reid, and Eyal de Lara. "A cross-layer approach to service discovery and selection in MANETs." Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on. IEEE, 2005.

[7] Zhang, Zhuoyao, et al. "An Integrated Approach to Service Selection in Mobile Ad Hoc Networks." Wireless Communications, Networking and Mobile Computing, 2008. WiCOM'08. 4th International Conference on. IEEE, 2008.

[8] Prochart, Guenter, et al. "Fuzzy-based support for service composition in mobile ad hoc networks." Pervasive Services, IEEE International Conference on. IEEE, 2007.

[9] P. E. Engelstad et al., "Service Discovery Architectures for On-Demand Ad Hoc Networks," Int'l. J.Ad Hoc and Sensor Wireless Networks, vol. 2. no. 1, Mar. 2006, pp. 27–58.

[10] V. Lenders, M. May, and B. Plattner, "Service Discovery in Mobile Ad Hoc Networks: A Field Theoretic Approach," Elsevier J. Pervasive and Mobile Computing (PMC), vol. 1, no. 3, Sept. 2005, pp. 343–70.

[11] Ay, Feruzan. "Context Modeling and Reasoning using Ontologies (July 2007)."