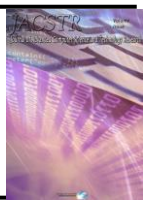


Developing Context-aware Mobile Applications Using Composition Process based-on heterogeneous software entities

Afrah Djeddar[1,a], Hakim Bendjenna [1,b], Abdelkrim Amirat [2,c], Ali A. Alwan[3,d]
[1]LAMIS Laboratory, University of Tebessa, Algeria
[2]LIM Laboratory, University of Souk Ahras, Algeria
[3]Department of Computer Science, Kulliyyah of Information and Communication Technology, International Islamic University Malaysia, Kuala Lumpur, Malaysia
[a]afrah-djeddar@hotmail.fr, [b]hbendjenna@yahoo.fr, [c]abdelkrim.amirat@yahoo.com, [d]aliamer@iium.edu.my

ISSN: 2231-8852

ABSTRACT
Despite the tremendous number of mobile applications (apps) that developed using various implementation forms such as component, service, or app, user's needs are unlike each other. Besides, mobile devices are characterized by heterogeneous software and hardware configurations. Developing customized mobile applications needs to explore and incorporate new entities in the surrounding user context. Besides, involving the existing heterogeneous entities might benefit in developing context-aware mobile apps. Thus, a significant challenge in the development process of mobile apps is the deployment of these applications in the heterogeneous devices available on the market. To tackle these challenges, there is a need for a composition process to reuse and utilize the existing heterogeneous entities to develop mobile apps according to user's requirements. Hence, the behavior of the desired apps can be customized according to the user context information. This paper addresses the issue of discovering, integrating and reusing the existing heterogeneous software entities in developing a customized mobile application. In this paper we propose framework for context-aware mobile apps composition process based-on existing heterogeneous software entities.

Keywords: Mobile apps, Composition process, Heterogeneous software entities, Context-aware, Composition cost

1. Introduction

In the recent couple of years, the interest in developing and using mobile apps is increased exponentially. Developing applications for mobile devices adds new challenges to software engineering process (Sheshagiri, Sadeh & Gandon, 2004; Chakraborty et al, 2005; Rosa & Lucena, 2011; Jeffrey, Michael & Julie, 2015). Mobile platforms features are changing continuously and rapidly. This including diverse capabilities such as GPS, sensors, and input modes. With these new added features, mobile app needs to be adapted and customized its behavior according to the context information surrounding the user. Moreover, these developed mobile applications should work in a seamless way on all mobile platforms. However, despite the huge number of available mobile apps, user's needs and daily-life activities are unlike from each another.

These issues give some need to utilize a composition mechanism when building mobile apps in order to achieve the desired functionalities while considering different context information (i.e. software and hardware characteristics of mobile device) (Sheshagiri, Sadeh & Gandon, 2004). One of the key benefits of using composition process is reuse the existing entities. Hence, composition process ensures fulfilling the reusability concept of software engineering process when designing mobile applications (Hock-Koon & Oussalah 2010a; Hock-Koon & Oussalah, 2010b; Furno & Zimeo, 2012). Exploiting information context is beneficial and might contribute in enabling solution for handling adaptation to customize the solution in a way that best fit user demands (Furno & Zimeo, 2012; Furno & Zimeo, 2014).

To the best of our knowledge, there is no such composition process has been applied on developing mobile apps which designed to consider the heterogeneity of software entities constituents and the composition cost in terms of adaptability. Thus taking advantage from existing works, we propose a composition process for composing context-aware heterogeneous mobile apps while considering the cost of composition. Our propose approach able to compose mobile apps using existing heterogeneous software entities by providing *heterogeneous composition process* in order to satisfy user's needs. Most importantly, consider the context information of the mobile device during the composition in order to develop mobile apps that can sense and adapt to the user context. The work presented in this paper attempt to provide a first step towards a composition process of context-aware apps based-on heterogeneous entities by a metamodeling approach in mobile environment.

The rest of the paper is structured as follows: Section 2 reviews the related works, while Section 3 introduces the motivating scenario and the overview of our proposed framework for composition process with the details description of the steps. Finally, Section 4 gives the conclusion remarks followed by the perspective future work.

2. Related Works

In this section we discuss and examine the existing approaches that focused on composition process for building and developing mobile applications. The work introduced by (Chakraborty et al, 2005) discusses the issue concern on services composition in mobile environment. The focus is given on how to evaluate the criteria that need to be used in order to enable the composition. They have developed distributed service composition architecture for service composition in mobile context. The proposed architecture incorporates a set of composition protocols named *Service Composition Protocols* that utilized to determine the composition process. The composition protocols take into consideration several factors that facilitate the composition process. These factors include, user mobility, dynamic changes of service topology and device resources. However, their work has not concentrate on the application layer and the adaptation capabilities.

The issue of process heterogeneity and data heterogeneity for web service composition has been highlighted by (Wu et al. 2007). Wu et al, (2007) has proposed an automatic planner and data mediator to resolves the issue of process and data heterogeneity for web service applications. They argued that their approach reduces the human effort and only the specification of the task such as initial state and the goal state of the task need to be changed. The proposed planner approach employed GraphPlan (Russell &

Norvig, 2006) planning algorithm to reduce the searching space and automatically generate the control flow of a Web process. The data mediator involves a context-based ranking algorithm to handle different structure and semantics of the web services and select the best element from the source messages if more than one element has acceptable semantics for the target element.

Li, Zhou & Qiu, (2008) proposed an automated Semantic-based approach to compose a semantic web services using data mediator and complete backward tree. The designed model involves some anthologies concept and exploits graph-based and semantic-based approaches to efficiently identify the web service request and the semantic of the service. Besides, the proposed approach also concentrates on reducing the searching space to process the service request with either single or multiple goals and attempt to resolve the issue of data heterogeneities to ensure interoperation between semantic Web services.

Hock-Koon & Oussalah, (2010b) develop a service composition metamodel that relies on reusing the existing entities and merging the available relevant resources which are defined as services. The proposed composite approach involves a homogenous reuse of the available compositions to provide the service composition. Moreover, the composite approach allows the specification of the auto-composition process and dynamically modifies its architecture and its composition logics according to the environmental context. Lastly, they argued that the proposed metamodel able to handle all the impacts on the architectural elements and on the composition logics.

Furthermore, the work in (Rosa & Lucena, 2011), has also concentrates on the issue of the heterogeneity aspects of mobile platform including display size, development libraries, sensors and keypad layout when developing mobile applications. They have presented AppSpotter as software architecture that makes the selection and the composition process of software components to be dynamic and automated when building mobile applications. AppSpotter selects the software components and the composition of them when developing mobile application by taking into consideration the mobile device features. Thus, this will leads into building customized mobile application that best meet user requirements.

Furno & Zimeo, (2014) introduced an automatic composition approach to design context-aware services. The proposed approach employed a semantic model to represent the context information that result into extending the services. Exploiting context information is beneficial and generates context-aware compositions and let the services provided in the model to be explored and composed dynamically. This will tailor a service search space to user needs, preferences and the current situation of the environment where the services have to be executed. Several analyses have been conducted to validate the proposed model and elaborate the improvement in the precision of the automatic compositions.

Elfirdoussi, Jarir & Quafafou, (2014) has discussed the issue of composition process of web services. They proposed an approach to automatically perform the composition process based on the concept of web service popularity. The idea is to develop a web search composition engine that automatically selects the best web services for the selected query based on its popularity. Then, the composition is derived as a result according the BPEL process model. To facilitate the process of extracting data from user, an interface has been developed to define the sequence of the activities with query input.

Han, Lee & Crespi, (2014) designed an automated system for web services adopting SOA paradigm. The proposed system employs the device profile and other context information that need to be forwarded to a composition engine in order to derive the most appropriate services to the user. The service selection process is conducted based on three main entities including, context, composition plan, and predefined set of rules. The composition process in the system consists of six-phases that carried out in sequence to identify the service composition. Furthermore, the composition process has integrated two components to facilitate the process of decision making for the service composition. These components are building ontology as a schema for representing semantic data and data composition plan description language that describe context-based composite services in a form of composition plans.

However, these examined approaches with theirs features often specialized does not have a global vision of mobile apps composition. Our research intends to clearly express the relevant notions of these existing mechanisms in *heterogeneous composition process* for mobile apps using meta-modeling approach.

3. Proposed Approach

In this section we introduce our proposed composition process, *heterogeneous composition process* that helps developing context-aware mobile applications. Fig. 1 demonstrates the *heterogeneous composition process*. The process comprises five main steps, namely: Defining the abstract functional architecture, Discovering suitable concrete software entities, Selecting context aware concrete software entities, Composing Mobile app and Generating Executable Application. These steps are explained below in furhter details.

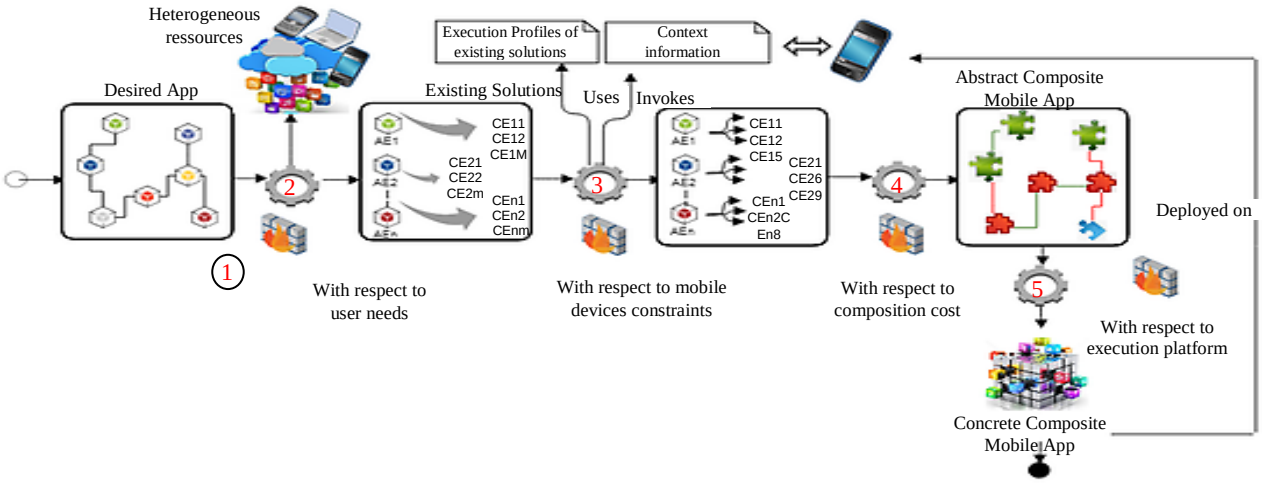


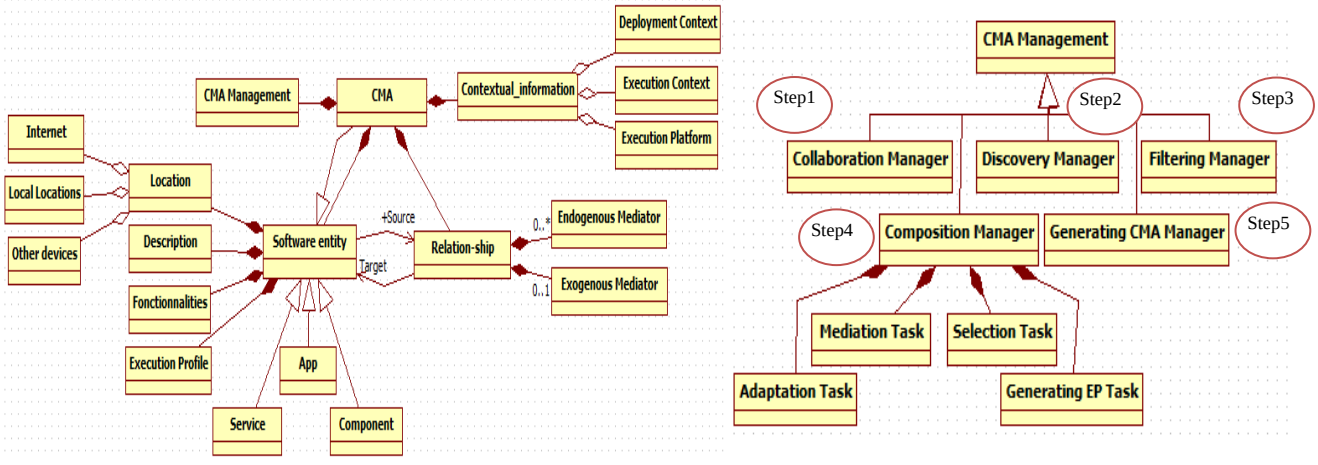
Fig. 1: Heterogeneous Composition Process Overview.

In order to represent the desired composite mobile app (CMA) at architectural level we propose in Fig 2.a the corresponding metamodel that provide a high-level description of the CMA architecture. The idea is to associate the necessary information to a specific architectural element used to compose the desired mobile app. This metamodel aims at defining how the architectural elements of the composite app are related each other. While defining the CMA architecture, we focus firstly on the different functionalities of the desired app (i.e. customer requirements) where each of them refers to an abstract software entity, ignoring how they will be implemented (e.g. components, services, apps). Secondly, we focus on the detailed description of the desired app, thus, this will help us to choose the implementation type of each abstract software entity (i.e. each functionality described in the first architecture) to construct this detailed architecture. This choice is delayed to the time when concrete software entities are selected in step 4 of the proposed process. For this purpose, we need the two following architectures to describe the CMA:

A. *CMA abstract functional architecture:* represents a high-level description of the desired functionalities and their dependencies to accomplish the composite's goal.

B. *CMA abstract detailed architecture:* represents a high-level description of the different concrete software entities that will be used to implement the desired functionalities and the different Mediators that represents the collaboration between reused concrete entities and they will be used to eliminate the heterogeneity among these entities.

CMA management presented in Fig 2.b shows the different composition tasks that need to be performed to composite mobile app. We associate to each role a specific architectural element used to compose the desired mobile app. It is based on the reification of the different properties and functionalities of existing composition mechanisms.



(a): CMA Metamodel **(b):** Composition Tasks

Fig. 2. Composition Process from architectural point view

Step 1: *Defining the abstract functional architecture*

This step aims at abstractly define the desired mobile app and focus on the definition of the different functionalities which are required to compose the future mobile app and provide the dependencies between these desired functionalities. We associate this ability with the collaboration manager as demonstrated in Fig. 2.b. *Collaboration manager* specifies *workflow* and *dataflow* between identified functionalities. *Workflow* schedules the invocations of functionalities (i.e. establishes precedence links) and *dataflow* expresses the data exchanges, inputs to output, between them (i.e. establishes use links). Other than traditional software systems, the development of apps on mobile devices (e.g. smartphones, tablets, etc…) is constrained by their limited resources such as: small memory, a battery powered computing environment, and availability of some devices (e.g. Wi-Fi, GPS, auto-focus camera, etc…) (Zhang et al, 2011). For this purpose, our objective is not only to create mobile apps through the composition of existing software entities according to user needs but also to compose mobile apps that are sensitive to their contextual information (i.e. adaptive to their run-time environment). Thus, it is necessary to identify the context information of the deployment environment. It represents all software and hardware characteristics of the mobile device that will be used to run the desired mobile app. We model this context information in three categories:

a) *Deployment context:* represents hardware characteristics of mobile devices.

b) *Execution context*: represents the current state of available devices.

c) *Execution platform:* represents mobile device operating system.

This different contextual information may be inferred by the system of the mobile device or identified by the developer.

Step 2: *Discovering suitable concrete software entities*

After setting the desired mobile app in abstract level (i.e. according to the architectural representation defined in the first step of the process), it is necessary to connect each abstract entity defined in the *CMA abstract functional architecture* with their corresponding existing concrete entities. We associate this role with the *Discovery manager* illustrated in the Fig 2.b. *Discovery manager* attempts to search and chooses the corresponding concrete entities for each desired functionality by downloading them through *Internet* where some of them are free while others are paying (e.g. line stores such as Google Play, App Store or the Windows Phone Store). Or bringing them from *local locations* or *other devices* (e.g. laptops, mobile devices, etc...). Thus, the *Discovery manager* exploits the abstract functional representation of the desired app to find the different concrete entities that can be better-turned to the user's requirements described in this representation.

The result of this discovery task is a set of suitable concrete entities for each abstract entity (i.e. Suitable-CE-Fun i). The concrete entities mapped to the same abstract entity are functionally equivalent, but may vary in several non-functional aspects. In order to build a functioning mobile app adapted to the mobile device that will support it; we propose to associate each entity with a specific *Execution Profile*

(*EP*) which contains all non-functional aspects that represent the necessary conditions for their execution (e.g. Needed devices such as: Wi-Fi, GPS…etc., size, consumption energy…etc.).

Step 3: *Selecting context aware concrete software entities*

The third step in our proposed *heterogeneous composition process* intents to solve the issue of of mobile devices heterogeneity that can arise when using such mobile device to deploy the desired CMA. The quality requirements of the CMA are represented by its adaptability with the context aware of the mobile device that will be used to run it. To ensure the correct deployment and the good functioning of the composed mobile app we first need to assure that their composed concrete entities are adaptable to the current context of mobile device. This step corresponds to a filtering operation which aims to select among all concretes entities of each abstract entity (i.e. result of step 2) those that are best suited with respect to the deployment and execution context of the mobile device where the composed mobile app will be deployed. We associate this filtering capability with the *Filtering manager* as illustrated in Fig 2.b). The filtering process runs based on *EP* of each concrete entity and the different characteristics of the mobile device to perform the comparison in order to select those that are adaptive to these characteristics.

A concrete entity is conforms to mobile device context information if the comparisons of all execution metrics with all mobile device characteristic are satisfied. The result of this step is a set of concrete entities which are suitable to the deployment and execution context at the same time.

Step 4: *Composing Mobile app*

After connecting each abstract entity with its corresponding context-aware concrete entities, we need to build the final architecture of the application. Composing Moble app step attempts to design and build the final app architecture (i.e. CMA *abstract detailed architecture*) by composing these concrete entities according to the coupling that generates the desired app with optimal composition cost. In this step we ensure a lower composition cost to compose the mobile app by selecting each abstract entity with most appropriate concrete entity with the minimum composition cost. The composition cost of each concrete entity is calculated in terms of adaptability based-on composition constraints explained in this section.

The different existing composition approaches use only one kind of software entities to compose the desired app and any of them was interested by the composition of heterogeneous entities. Typically, they define a composite app as a collection of software entities of the same kind using for example: SOA approach (Erl, 2005) or component-based-approach (Jifeng, Li & Liu, 2005) or other paradigms (Amirat, Hock-Koon, & Oussalah, 2014). Based on the new paradigm XAAS (anything as a service) (Rajasri, Arundurai & Ady, 2013), we try to overcome this limitation by providing a description representing the CMA architecture with: services, components, apps separately, or with heterogeneous entities (i.e. with several software entities types). Thus, our process allows building mobile apps as exogenous or endogenous composition as depicted in Fig. 2.c. Each of constituents' software entities

will provide a specific service. However, heterogeneity issue in the composition mechanism might produce two kinds of heterogeneity problems:

- *Heterogeneous nature of entities:* the composed entities cannot directly communicate because the data which are exchanged between these entities are not understandable (e.g. microphone provides an audio stream and the jukebox needs to string input to perform its task).
- *Heterogeneous type of entities*: represent the coordination of two different types of software entities (e.g. component connected with service).

The proposed metamodel aims to address these heterogeneity issues by proposing two kinds of mediators as shown in Fig 2.a) where (Cimpian, Mocan & Stollberg, 2006):

- *Endogenous mediator* that overcomes the heterogeneity between two entities of different nature, exchange data can require some transformation to be understandable. *Endogenous mediator* represents the *Mediation services* that are selected to ensure these data transformations.
- *Exogenous mediator* this kind of mediators is intended to eliminate the heterogeneity between two entities of different kinds. These entities cannot directly communicate owing to their different type. Exogenous mediator aims to encapsulate related heterogeneous entities, it builds well-formed interface for each of them in order to take advantage of their services but just with manipulating necessary inputs and outputs independently from language implementations of these constituents entities. For this purpose, this composition process treats four type of composition as illustrated in the Table 1.

Table 1. Composition Types

Composition Type	Heterogeneity problems	Proposed Mediators
Heterogeneous Exogenous Composition	Heterogeneous Nature of entities Heterogeneous Type of entities	Endogenous Mediators Exogenous mediators
Homogenous Exogenous Composition	Heterogeneous Type of entities	Exogenous mediators
Heterogeneous Endogenous Composition	Heterogeneous Nature of entities	Endogenous Mediators
Homogenous Endogenous Composition	None heterogeneity problems	None needed mediators

Composition manager aims to handle *CMA abstract functional architecture* in order to derive the *CMA abstract detailed architecture* by replacing each functionality with its appropriate context-aware concrete entity that can implement it with respect to the cost of its composition. *Composition manager* needs to perform four tasks to achieve its objective.

Mediation Task that represents the heterogeneous and/or exogenous composition, it manages *use links* which represent the *dataflow* between desired functionalities and identifies if it is necessary to associate this relationship with services mediation or/and to encapsulate the composed entities. Based-on these composition constraints, *Composition manager* performs the *selection task* by calculating the composition cost of each concrete entity in order to select the best suited one which allow to compose the desired mobile app with the lower composition cost. The composition of adaptable concrete entities is not sufficient to ensure that the composite app itself will be adaptable to the context of the mobile

device. *Composition manager* has the role to verify, after each composition step, if the actual free storage capacity is sufficient for deploy the composed app in this mobile device (*Adaptation task*). Thus, if the consumed energy to handle the composed app does not exceeds the current battery level. If one of these two constraints is not satisfied, *Composition manager* triggers the recomposition of the app using alternative concrete entities that are selected in step 3. Last but not least, it is necessary to generate for each composed mobile app its own *EP* and identify its own properties. *Composition manager* has the potential to fulfill CMA *EP* according to the different characteristics of their constituents (*Generating EPs task*).

Step 5: *Generating Executable Application*

Mobile apps are composed visually with the proposed architectural representation without the need to write any lines of code. The executable model will be generated from the CMA description after searching, filtering, and selecting the most appropriate concrete entities (i.e. *CMA Abstract Detailed Architecture*) as illustrates the instantiation relation presented in the Fig. 2.b. Thus, our approach copes with the heterogeneity of mobile platforms. It is able to support heterogeneous target environments ranging from *CMA* architecture to mobile platforms. From this architectural model, the generation of the concrete CMA (i.e. application code) towards a specific platform is based on MDD mechanisms using transformations: *model to code*. This task is the responsibility of *Generating CMA manager*. This step proves that our process has the potential to deploy and to migrate the same CMA between different mobile platforms (e.g. android, iOS).

4. Conclusion

In this paper we have presented a conceptual framework for our proposed idea to provide *heterogeneous composition process* for mobile apps. This process is defined at architectural level and consists of five main steps, namely: Defining the abstract functional architecture, Discovering suitable concrete software entities, Selecting context aware concrete software entities, Composing Mobile app and Generating Executable Application. The proposed process aimed to meet the needs of users and compose mobile apps that are sensitive and adaptive to the contextual information of the mobile device in which they will be installed, and also to facilitate the task of mobile apps composition while ensuring the efficiency of the generated app. As the future work directions for our work in this paper, we plan to evaluate our proposed framework to investigate its performance and effectiveness with respect to some real life applications.

References

Amirat A, Hock-Koon A., and Oussalah M. C. (2014). Object-Oriented, Component-Based, Agent-Oriented and Service Oriented Paradigms. Software Architecture 1 (ed M. C. Oussalah), John Wiley & Sons, Ltd, Chichester, UK. doi: 10.1002/9781118930960.ch1

Chakraborty, D., Joshi A. , Finin T. and Yesha Y. (2005). Service Composition for Mobile Environments. *Mobile Networks and Applications*, vol. 10, no. (4) (2005), pp. 435-451.

Cimpian E., Mocan A. and Stollberg M. (2006). Mediation Enabled Semantic Web Services Usage. *In the First Asian Semantic Web Conference, (ASWC 2006)*, Beijing, China, September, 2006, pp. 459-473.

Elfirdoussi S., Jarir Z. & Quafafou M. (2014). Web Service Composition Based on Popularity. Journal of Computer Science and Information Technology. vol. 4, no. (7), Academy & Industry Research Collaboration Center (AIRCC), pp. 251- 263.

Erl T. (2005). Service-Oriented Architecture (SOA) Concepts, Technology and Design. Prentice Hall, 2005.

Furno, A. & Zimeo E. (2014). Context-aware Composition of Semantic Web Services. *Mobile Networks and Applications*, vol. 19, no. (2), pp. 235-248.

Furno A. & Zimeo E. (2012). Context-Aware Design of Semantic Web Services to Improve the Precision of Compositions. *In the 1st International Conference on Context-Aware Systems and Applications, (ICCASA 2012)*, Ho Chi Minh City, Vietnam, November, 2012, pp. 97 – 107.

Han S. N., Lee G. M and Crespi N. (2014). Semantic Context-Aware Service Composition for Building Automation System. *IEEE Transactions on Industrial Informatics*, vol. 10, no. (1), 2014, pp. 752 – 761.

Hock-Koon, A. & M. Oussalah (2010). Expliciting a Composite Service by a Metamodeling Approach. *In the 4th International Conference on Research Challenges in Information Science (RCIS)*,Nice, France, May 2010, pp. 533- 544.

Hock-Koon A. & Oussalah M. (2010). Composite Service Metamodel and Auto Composition. *Journal of Computational Methods in Science and Engineering*, vol. 10, no. (2), (2010), pp 215-229.

Jeffrey S. H. Michael F. and Julie A. A. (2015). The Future Of Mobile Application Development. The Mobile App Development Playbook for 2015, Forrester.

Jifeng H., Li X. & Liu Z. (2005). Component-based Software Engineering. In Theoretical Aspects of Computing–ICTAC, Lecturer Notes in Computer Science, vol. (3722), 2005, Springer. pp. 70-95.

Li R., Zhou Z. and Qiu Y. (2008). Automated Composition of Semantic Web Service using Data Mediator and Complete Backward Tree. *In the 2008 International Conference on Computer Science and Software Engineering, (CSSE 2008)*, Wuhan, China, December 2008, pp. 390 – 393.

Rajasri K., Arundurai R., and Ady K. D. (2013). Qos Aware and Cost Based Optimization of Service Composition Using Genetic Algorithm. International Journal of Computer Science, vol. 1, no. (6), pp. 24- 29.

Rosa R. E. V. S & Lucena V. F. (2011). Smart Composition of Reusable Software Components in Mobile Application Product Lines. *In the 2nd International Workshop on Product Line Approaches in Software Engineering*,Waikiki, Honolulu, Hawaii, USA, May 2011, pp. 44 – 49.

Russell, S. & Norvig, P. (2006). Artificial Intelligence: A Modern Approach, Prentice Hall, 2006.

Schmidt H., Kapitza R. & Hauck F. J. (2007). Mobile Process based Ubiquitous Computing Platform: A Blueprint. *In the 1st Workshop on Middleware-Application Interaction, in conjunction with the European Conference on Computer Systems*, (EuroSys 2007). Lisbon, Portugal, March 2007, pp.25- 30.

Sheshagiri M, Sadeh N. M. and Gandon F. (2004). Using Semantic Web Services for Context-Aware Mobile Applications. *In the MobiSys 2004 Context-Awareness Workshop*, Massachusetts, USA, June 2004.

Wu Z., Ranabahu A., Gomadam K., Sheth A. P. and Miller J. A. (2007). Automatic Composition of Semantic Web Services Using Process and Data Mediation. Technical Report, LSDIS lab, University of Georgia, February , 2007.

Zhang X., Kunjithapatham A., Jeong S. and Gibbs S. (2011). Towards an Elastic Application Model for Augmenting the Computing Capabilities of Mobile Devices with Cloud Computing. *Mobile Networks and Applications*, vol. 16, no. (3), pp. 270-284.