An emerging multi-paradigm for representing mobile applications' architectures using heterogeneous conceptual bricks

Afrah Djeddar, Hakim Bendjenna, Abdelkrim Amirat, Philippe Roose, Lawrence Chung

An emerging multi-paradigm for representing mobile applications' architectures using heterogeneous conceptual bricks

Afrah Djeddar* and Hakim Bendjenna

Laboratory of Mathematics, Informatics and Systems (LAMIS),
Department of Mathematics and Computer Science,
University of Larbi Tebessi,
Tebessa 12002, Algeria
Email: afrah-djeddar@hotmail.fr
Email: hbendjenna@yahoo.fr
*Corresponding author

Abdelkrim Amirat

Laboratory of Informatics and Mathematics (LIM),
Department of Mathematics and Computer Science,
University of Souk Ahras,
Souk Ahras 41000, Algeria
Email: 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
Email: philippe.roose@iutbayonne.univ-pau.fr

Lawrence Chung

Department of Computer Science,
University of Texas at Dallas,
Dallas, TX, USA
Email: chung@utdallas.edu

Abstract: The mobile applications have enjoyed explosive growth these last years. Taking advantage from these existing softwares, the constituent software bricks to compose such mobile application can take different implementation forms and manipulate heterogeneous data by dint of user's requirements or its execution context. However, the mobile software developer confronts difficulty to compose already existing software entities because of their heterogeneity. An emerging need is then to have a new modelling space to support the development of heterogeneous mobile applications. In view of this fact, this paper discusses the proposal of a multi-paradigm for representing mobile applications based-on heterogeneous conceptual bricks including their architectural conception and the specification of the necessary adaptation mediators. The proposed paradigm aims to deal with the heterogeneity presented by the constituent conceptual bricks and the execution environment of the final product. A conceptual description of a mobile application baptised *ShopReview* is presented to show the usability of the proposed paradigm.

Keywords: multi-paradigm; architectural description; heterogeneity; mobile applications; conceptual bricks; adaptation mediators.

Reference to this paper should be made as follows: Djeddar, A., Bendjenna, H., Amirat, A., Roose, P. and Chung, L. (XXXX) 'An emerging multi-paradigm for representing mobile applications' architectures using heterogeneous conceptual bricks', *Int. J. Computer Applications in Technology*, Vol. X, No. Y, pp.xxx–xxx.

Biographical notes: Afrah Djeddar is a PhD student in Computer Science at Tebessa University, Algeria. She is a member in LAMIS Laboratory of Tebessa University. She received her Master's

degree in Software Engineering from the University of Souk Ahras, Algeria, in 2013. Her areas of research include software architectures, transformation models, mobile applications, meta-modelling, context description, and adaptive applications.

Hakim Bendjenna is an Assistant Professor in the Department of Computer Science at Tebessa University, Algeria. He received his PhD in Computer Science from Mentouri University, Algeria, and Toulouse University, France. His research interests lie in the general field of software engineering and decision support with special focuses on requirement engineering. He has taught primarily introductory programming courses and courses in software engineering for over ten years. He also frequently serves as a program committee member for various international conferences and workshops. He is a Chair of IT4OD 2014 Conference.

Abdelkrim Amirat received his PhD in Computer Science in 2007, and Habilitation in 2010. Currently he is a Professor of Computer Science at the University of Souk Ahras, Algeria. He is the Director of Mathematics and Computer Science Laboratory and the chief of the software engineering team. His main research concerns are software architectures and their evolution, modelling and meta-modelling. He has served on program committees of several international journals, conferences and workshops.

Philippe Roose is an Associate Professor at LIUPPA Research Lab at University of Pau/France. His research deals with middleware, adaptations, and context-aware applications. Application fields are about smart-* (health, home, city). Since 2016, he has been a Leader of the T2I Research Team. He supervised ten PhD and is involved in many research projects and conferences animations. He wrote two books on history of micro-computers and directed several other ones focused on his research domains. He gave several national and international talks in French, English or Spanish. He is strongly involved in actions with South America.

Lawrence Chung has been working in requirements engineering and system/software architecture. He was the principal co-author of the research monograph 'Non-functional requirements in software engineering', and has been involved in developing 'RE-Tools' (a multinotational requirements modelling project), 'HOPE' (a smartphone app project for people with difficulties), and 'Silverlining' (a cloud computing and big data project). He has been a keynote speaker, invited lecturer, Co-Editor-in-chief for *Journal of Innovative Software*, editorial board member for *Requirements Engineering Journal*, editor for *ETRI Journal*, and program Co-Chair for various international events. He is currently on the faculty of Computer Science at University of Texas at Dallas. He received his PhD in Computer Science from University of Toronto in 1993.

This paper is a revised and expanded version of a paper entitled [title] presented at [name, location and date of conference]. [AQ1]

AQ1: If a previous version of this paper has originally been presented at a conference, please complete the statement to this effect or delete if not applicable.

1 Introduction

The massive adoption of mobile devices to perform our tasks in the daily life proves the exponential growth of mobile applications and their development (Ickin et al., 2012; Jones, 2013; Statista, 2015). These mobile devices are characterised by heterogeneous hardware and software configurations, present limited resources and have specific execution context (Chen et al., 2014). Thus, the constituent software entities of the desired mobile application can take different forms of implementation (e.g. components, services, etc.) or manipulate heterogeneous data owing to the user's requirements. However, this heterogeneity forces software publishers to entirely redevelop their products for each target technology or confront a difficulty to compose the existing software entities in view of their heterogeneity, which generates unbearable additional costs in terms of time and money (Cugola et al., 2014). In view of this fact, MDA (Model-Driven Architecture) (Blanc and Salvatori, 2011) precepts are faced with this issue by producing PIM models (Platform Independent Model) for representing mobile applications in a technologically neutral way (Diaw et al., 2010) through what is called *software architectures* using specific ADLs (Architectural Description Languages) (Medvidovic et al., 1999; Medvidovic and Taylor, 2000). Nowadays, several principles, standards and practices are used for the architectural description of applications using different types of the constituent conceptual bricks. Among the most adopted paradigms dedicated to describe and represent the functional core of applications we find: the approach using services (SOSE) and the approach based-on components (CBSE) (Amirat et al., 2014).

To take advantage of the benefits presented by software components and services (Cai et al., 2000; Papazoglou and Van Den Heuvel, 2007), a combination between the based-components approach and services-oriented approach has appeared necessary. Therefore, several studies have focused on the technological aspect of heterogeneity management whether in terms of the implementation forms of the software bricks or the exchanged data between them where many multi-paradigm systems have thus emerged. The approaches which borrow and combine the conceptual and technical elements derived from

CBSE and SOSE are qualified as *hybrid*. We cite among them those that accurately reflect the heterogeneity concept: SCA (Service Component Architecture) (Beisiegel, 2007) and SLCA (Service Lightweight Components Architecture) (Hourdin et al., 2008).

SCA allows defining architecture of components and services using components to manipulate the orchestration of services and thus create a composite service, while SLCA shows an architectural model for the composition of services based on the assembly of lightweight components. Taking advantage of these multi-paradigms, we propose in this study a new paradigm which enables to combine software entities regardless of their implementation forms while integrating, subsequently, necessary adaptation mediators to ensure the communication between the connected software bricks of different types and the compatibility of exchanged heterogeneous data. Thereby, this proposed multi-paradigm is devoted to compose heterogeneous mobile applications from the architectural perspective.

The different existing software entities offer different forms of implementations and handle heterogeneous data. For this raison, designers/developers are sometimes forced to combine heterogeneous entities to build mobile applications that meet users' needs and that will be adaptable to their execution environments. The existing hybrid paradigms serve either to describe the composition of services using lightweight components (e.g. SLCA) or to manipulate the orchestration of services using components (e.g. SCA) but in some cases we need to describe our desired applications by combining any type of software entities in order to take advantage of their services independently of their implementation details. Our emerging multi-paradigm comes as a solution to fill this lack by allowing the composition of heterogeneous applications using conceptual bricks of different types (e.g. composition of services and components).

The remaining part of the paper is organised as follows: the next section introduces several representations dedicated to describe the functional core of applications and presents some research works studied in the literature to cope with the heterogeneity problem presented by these paradigms which concerned the exchanged data. Afterwards, the third section describes our multi-paradigm for modelling heterogeneous mobile applications while proposing a metamodel for the architectural description of their functional core. Then, in the fourth section, we implement the proposed metamodel and we explain the functioning of our paradigm, after that we introduce an example to show the applicability of this paradigm. Finally, Section 5 draws the conclusion and future scope of our research work.

2 Related works

2.1 Functional-core representations

Brown and Wallnau (1998) and Cai et al. (2000) attack the components-oriented programming, whose purpose is to build software systems using shelf components (COTS: Commercial Off-the-Shelf) and thus accelerate the software development process. An approach based-on software components (Weinreich and Sametinger, 2001) serves to capitalise the code in software entities called *black boxes* that are reusable and only software interfaces of exchange are known. There are several modelling languages called ADLs (Architecture Description Languages) devoted to describe software architecture of such application using software components. These description languages provide formalisms allowing a designer to model software's specifications and development in a high level of abstraction without forgetting that ADLs are independent of any programming language and execution environment. Therefore, ADLs are a support for the description of the application's structure (Soucé and Duchien, 2002). As examples of description languages for this kind of architectures we quote: Fractal (Bruneton et al., 2006), Wright (Allen et al., 1998), Darwin (Luckham and Vera, 1995), Rapid (Luckham et al., 1995), 2SADEL (Medvidovic et al., 1999), ACME (Garlan and Perry, 1995), and xADL2.0 (Dashofy and Van Der Hoek, 2002).

A second approach for building applications and representing its functional core is based on *Services-Oriented Architectures* (SOA). In 2005, Srinivasan and Treadwell draw attention to the meaning of SOA which refers to the design of a system and not its implementation (Srinivasan and Treadwell, 2005). In 2007, the authors describe SOA as the logical way for designing software systems that provide services where a service is provided by a producer to benefit a customer (Papazoglou et al., 2007). Srinivasan and Treadwell (2005) and Papazoglou and Van Den Heuvel (2007) discuss several features for this type of software entities. These services communicate with their customers through transmitted messages and provided responses and don't require describing the implementation details. The declination of services on internet is *web services* where several languages have been addressed to describe this type of software entities. WSDL (Web Services Description Language) (Christensen et al., 2001) is a language based-on XML dedicated for describing web services which respect the WS * specification. OWL-S (Ontology Web Language for Services) (Martin et al., 2004) is a complement to the WSDL description that aims to add the semantic aspect of services. This language used to describe web services semantically following three parts: service profile, service model, service grounding (Maheswari and Karpagam, 2015).

2.2 Heterogeneous representations: multi-paradigms

CBSE and SOSE are two very similar paradigms which are dedicated to construct applications from existing software entities, components or services (Amirat et al., 2014). CBSE is based at the design phase on the notions *configuration type* and *composite component type* and on the runtime on their instances *configurations* and *composite components*. Nevertheless, the notion of abstract service exists in some approaches (Cavallaro et al., 2009). Most existing studies refer to a service as an entity of the runtime (Stojanovic and Dahanayake, 2005) where the extension of the composite service through the composition mechanism is mainly in runtime.

Serval works are devoted for the composition of software entities of service type (Chemaa et al., 2015; Kalamegam and Zayaraz, 2016). Most of them consider the composite service as the execution, through a composition engine, of a collaborative schema between services where some others (Geebelen et al., 2008) introduced concepts of instantiation of collaborative schema from abstract template that describes them. Hock-Koon in his research work (Hock-Koon, 2011) has chosen to consider this representation similar to the OO (Object-Oriented) with *types of collaborative schema* as entities at the design phase and instances of *collaborative schema* as runtime entities which is the case on our research work. A critical that we bring to the architecture based-on components is its static character after the composition of the system. Indeed, once components selected and coupled are only with difficulty changed during execution (Brel, 2013).

To take advantage of benefits offered by software components and services, a combination of the approach components-based and services-oriented appeared necessary. Among the approaches that borrow and combine the conceptual and technical elements arising from the CBSE or SOSE we can find: SCA (Beisiegel et al., 2005), SLC (Hourdin et al., 2008), FROGi (Desertot et al., 2006)). SCA corresponds to the creation of service-oriented applications based-on SCA components assemblies. Another model classified as hybrid approaches is SLCA. Its main objective is to define a dynamic architecture for the composition of services by leveraging several existing paradigms: web services-oriented architecture, lightweight assembly of components and Events.

2.3 Heterogeneity management of exchanged data

The heterogeneity challenge cannot concern only the software bricks but it can also affect the exchanged data between these constitutive entities. In fact, several studies have focused on the technological aspect of the management of exchanged data heterogeneity.

Hock-Koon treated this issue for its proposed composite service (Hock-Koon and Oussalah, 2010). It brings together the concerns of *invocation* that reflect the triggering of the execution of the constituent services and *mediation* that represent the capacity of the composite to ensure proper understanding of data exchanged between its constituent services.

Kalasapur et al. (2007) propose a method for the composition of services which target the data heterogeneity problems. Services are organised in a graph that represents the set of possible compositions. During data incompatibility between services, the system uses the compositions graph for identifying a succession of services capable of ensuring the necessary changes on the data a priori incompatible.

Derdour in his research (Derdour et al., 2010a; Derdour et al., 2010b) attacked this problem for the multimedia software architectures. He declared that the heterogeneity problem is based on the flow of exchanged multimedia data (e.g. picture, sound, text). In view of this fact, he proposed a metamodel MMSA (Metamodel Multimedia Software Architecture) for multimedia architectures where this metamodel allows describing multimedia systems as a collection of components that handle different types and formats of multimedia data and interact with them via adapters. This model serves to facilitate the adaptation task between media of the same type (e.g. picture to picture), or between different types of media (e.g. text to sound).

3 Proposed paradigm

Software development for mobile environments through the reuse of the existing software entities is headed by user's requirements and the context of the mobile device that will be used to implement the final product. The concrete software entities selected according to the execution context to implement the desired functionalities can be heterogeneous both in terms of their implementation forms or exchanged data between them. Thereby, the resulting application will be an application that reflects a set of software entities of different types and/or the exchanged data between them require some transformations to make them understandable.

If the related software entities cannot communicate directly due to the fact that the exchanged data between them are not understandable, we say that these related entities haven't the same nature. By way of example, the functionality *Acquire Photo*, which serves to acquire an image of a product that we want to buy, provides an image of *jpg* type while the functionality *Read Barcode*, used to extract the product barcode from the acquired image, needs an image of *WebP* type so that it can function properly.

If the related software entities cannot communicate directly due to the incompatibility of their communication interfaces (i.e. heterogeneous interfaces), we say that these related software entities are not of the same type. By way of example, software entity of component type connected with another of service type.

For this reason, we propose in this paper a metamodel baptised HMA-AD (Heterogeneous Mobile Applications-Architectural Description) to describe heterogeneous mobile applications at the architectural level. The following subsections present respectively: a conceptual vision on the proposed description language (i.e. defining HMA-AD metamodel), the specification of the proposed adaptation mediators and the different types of composition treated by our multi-paradigm.

3.1 HMA-AD metamodel: heterogeneous mobile application-architectural description

In this section, we will present the proposed metamodel for modelling the functional core of heterogeneous mobile applications. The aim of this metamodel is to represent any mobile application whatever the implementation details of its constituent software entities. Thereby, it describes a formalism that allows the designer to perform a heterogeneous or homogeneous composition by means of concrete software bricks chosen to implement the required functionalities. A great

advantage of the proposed description language is to use any type of the constituent software entities, i.e. don't restrict the technology choice, while specifying and treating heterogeneity problems in the case of heterogeneous coordinations. So it gives the possibility to associate heterogeneous composition relations with adaptation mediators in order to overcome the encountered heterogeneity issues.

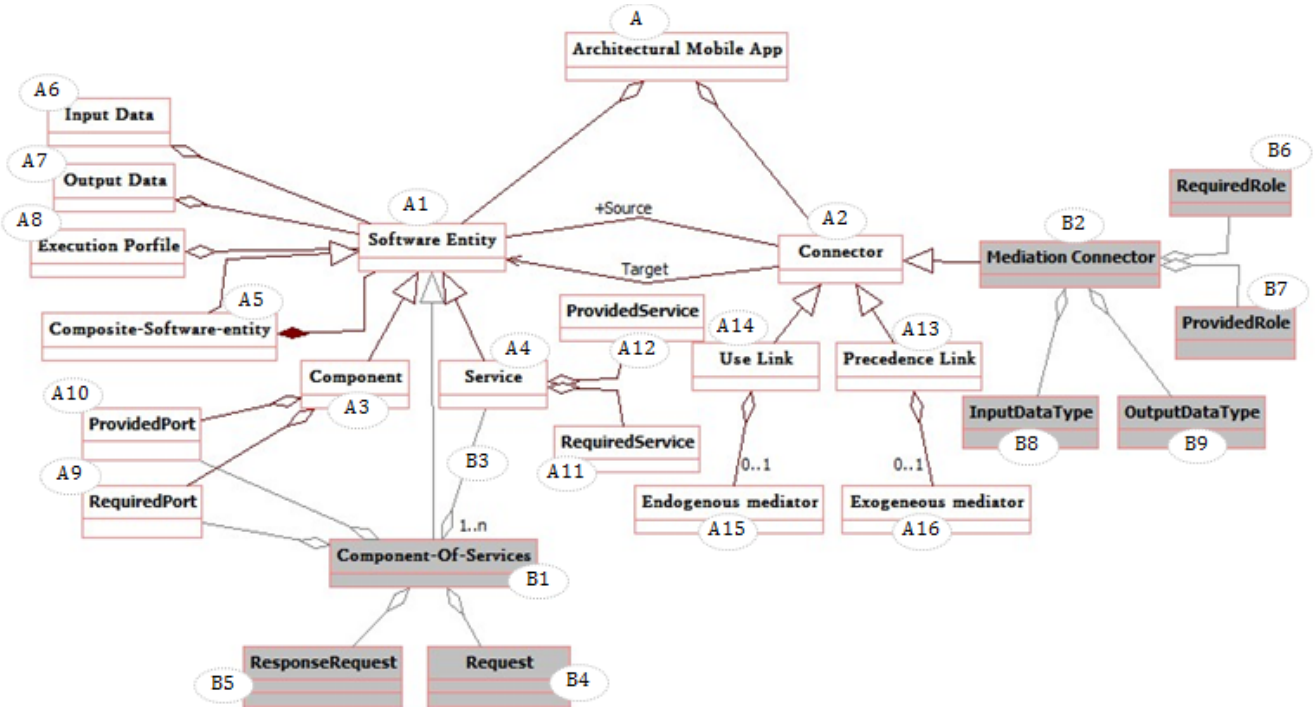
In our research work we consider that a software entity refers to an abstract action which takes as input a set of necessary parameters for its functioning and returns as output the desired result. This action should be executed according to a set of conditions that we have gathered in an execution profile. By way of example, a software entity X needs GPS service so it can function properly and 5 MB of capacity storage for correct deployment on the mobile device to be used. Therefore, we define a software entity as a quadruplet: the function to be carried-out, input data, output data and an execution profile which contains all necessary conditions for its execution.

Figure 1 shows the proposed metamodel. A *mobile application* (A) consists of a set of *software entities* (A1) connected together via *connectors* (A2). A *software entity* may be a *component* (A3), a *service* (A4), or a *composite entity* (A5). A *software entity* has *input data* (A6), *output data* (A7), and an *execution profile* (A8). Data exchange between the related entities is done via *provided ports* (A9) and *required ports* (A10) for entities of *component type* and through *provided services* (A11) and *required services* (A12) concerning entities of *service type*. The different *composition relations* (A2) between the constituent software entities are represented by *precedence links* (A13) and *uses links* (A14), where these connectors will be attached with *endogenous mediators* (A15) or *exogenous mediators* (A16) in the case of heterogeneous coordination.

The *precedence links* indicate the invocation sequence of the identified software entities while the *use links* serve to define the collaboration schema of the exchanged data between them. *Exogenous mediators* will be attached to the *precedence links*, because they carry on the constituent objects of the application, whose objective is to overcome the heterogeneity between related software entities that don't have the same implementation type; while *endogenous mediators* will be attached to the *use links*, because they carry on the data exchanged between the heterogeneity between two software entities of different nature that cannot communicate directly.

Furthermore, the proposed formalism for defining the architectural model of the heterogeneous mobile application is dedicated to be refined in order to integrate the functioning of the proposed adaptation mediators and therefore to get a detailed architectural model (see Figure 1, classes in grey). Each *exogenous mediator* indicated in the architectural model will be replaced by an entity of component type labelled *component-of-services* (B1) which aims to encapsulate entities of *service type* in order to eliminate the heterogeneity between related entities of different types by constructing common communication interfaces. Each *endogenous mediator* becomes a *mediation connector* (B2) which is dedicated to ensure the compatibility of exchanged data. Therefore, the proposed architecture description language relies on the integration of mediators and the description of their functioning to remedy the heterogeneity problems arising during the composition.

Figure 1 HMA-AD metamodel



3.2 Adaptation mediators' specification

After defining the architectural model for the desired heterogeneous mobile application, our paradigm intended to specify more precisely the role of integrated adaptation mediators.

On the one hand, *exogenous mediators* (see Figure 2) are designed to ensure communication between two software entities of different types. Given that the software entities cannot communicate by reason of their heterogeneous implementation forms, *exogenous mediators* are intended to encapsulate these interconnected entities in such a way that they can interact with each other. This kind of adaptation mediators aims to build common and well-formed interfaces for heterogeneous software entities to take advantage of their services but just by manipulating the necessary inputs and outputs regardless of their implementation details. Specifically, in the case of an exogenous coordination between a component and a service, the service will be encapsulated (B3, Figure 1) within the new entity *component-of-services* (see Figure 2). This latter may encapsulate one or more cooperated services where he plays the role of the engine defined in services orchestration. Initially, it aims to trigger the execution of the service that he includes by providing *required data* received through its *required port* by means of a triggered *request* (B4, Figure 1). After, it releases the *obtained result* using its *provided port* by triggering another request of *response* (B5, Figure 1). This ensures the communication between the source and target entities using compatible interfaces.

On the other hand, *endogenous mediators* (see Figure 3) reflect the *mediation services* to be selected for the processing of the exchanged data which are heterogeneous. In this case, each composition relation attached with this type of mediators will be connectors having complex interactions. These *mediation connectors* are designed to convert the exchanged data by calling the appropriate transformation services, while using his required roles (B6, Figure 1) to receive the heterogeneous data and his provided role (B7, Figure 1) to disseminate the transformed data.

Our paradigm represents this connector as *glue* that defines tow functions. The first one serves to search the appropriate mediation service (Search-MD: Searching a Mediation Service) based on the types of the exchanged data Input-data Type (B8, Figure 1) and Output-data Type (B9, Figure 1) in the mediation services' library; while the second function aims to call the found transformation service (Call-MD: Calling a Mediation Service) to ensure the compatibility of the heterogeneous exchanged data.

Owing to the fact that exogenous mediators will eliminate the heterogeneity between two software entities of different types by encapsulating services in a specific software component, the use links will connect only (*component*/*component*) or (*component*/*component-of-services*).

Figure 2 Exogenous mediator' structure

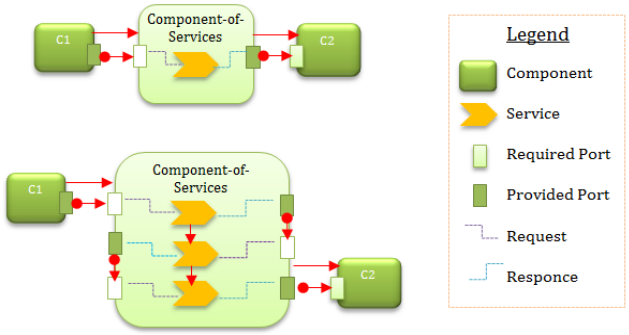
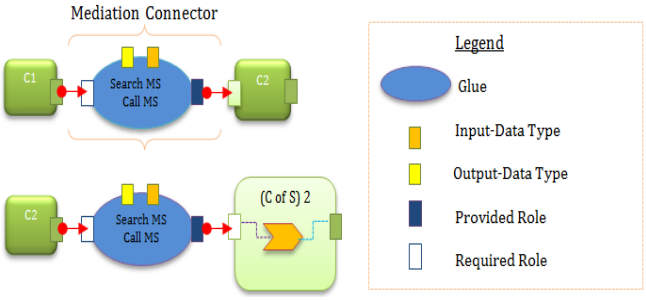


Figure 3 Endogenous mediator' structure



3.3 Types of composition treated by our multi-paradigm

The proposed multi-paradigm treats four types of composition:

a. Exogenous composition: composing software entities of different type (i.e. haven't the same implementation forms).

b. Endogenous composition: composing software entities of the same type.

c. Heterogeneous composition: composing software entities which do not have the same nature (i.e. handle heterogeneous data).

d. Homogeneous composition: composing software entities which have the same nature.

Each exogenous or heterogeneous composition requires attaching specific adaptation mediator in accordance with the type of this composition relation. Our paradigm allows performing *endogenous homogenous composition* without any adaptation and thus obtaining homogeneous mobile applications. We have previously indicated that a composition relation between two software entities is represented by a precedence link to indicate the order of invocation and/or a use link to express the flow of exchanged data between them. For this purpose, the different combinations between software bricks of the desired mobile application can take the following forms (see Table 1).

Table 1 Possible composition relations

Possible composition	*Heterogeneity problems*	*Necessary mediators*
Exogenous heterogeneous composition	Entities of different type/ entities of different nature	Exogenous mediator/endogenous mediator
Exogenous homogeneous composition	Entities of different type	Exogenous mediator
Endogenous heterogeneous composition	Entities of different nature	Endogenous mediator
Endogenous homogeneous composition	Any heterogeneity Problems	Any mediators

4 Implementation

4.1 Implementing the proposed metamodel HMA-AD

Eclipse platform provides graphical tools to facilitate editing EMF models (Eclipse Modelling Framework). EMF aims to describe the domain model (metamodel) under the extension *.ecore. We have relied on this technology to implement the proposed metamodel HMA-AD as shown in Figure 4.

The metamodel shown in Figure 1 reflects just a general vision on the proposed description language to design the architectural model of the desired mobile application. That is why we have used EMF technology to provide the detailed definition of the grammar of this language including all necessary concepts expressed by *classes* and all *composition relations* between these classes in accordance with notations presented by UML as indicated in Figure 4.

The architecture of the desired mobile application includes a set of concrete software entities defined in the abstract level (*Concrete-Entity class*). A software entity can be either a component (*Component class*) or a service (*Service Class*) or other (*Application class*) while the Composite-CE class reflects a composite software entity. Each concrete software entity must be attached by its required data (*Input-Data class*) through the relation represented by *CE2Input-Relation class* and its provided data (*Output-Data class*) using the relation represented by *CE2Output-Relation class*.

The required data of a software component will be represented by required ports (*Required-Port class*) while the provided data will be represented by provided ports (*Provided-Port class*). As well as, the provided and required data for a concrete entity of service or application type will be represented respectively via *Provided-Service class*, *Required-Service class*, *OutputData-App class*, and *InputData-App class*.

The precedence link (*Precedence-Link class*) which connects two entities of different types will be attached by exogenous mediator (*Exogenous-Mediator class*) through the relation expressed by *PL2ExMed-Relation class*. The use link (*Use-Link class*) which connects two entities of different nature will be attached by endogenous mediators (*Endogenous-Mediator class*) through the relation expressed by *UL2EnMed-Relation class*. Each entity is associated with its execution profile (*Execution-Profile class*) through the relation expressed by *ExProfile2CE-Relation class* as well as the desired heterogeneous mobile application must be attached to its own execution profile via the relation expressed by *ExProfile2CMA-Relation class*.

These architectural elements (white classes) are dedicated to define the architecture of the desired heterogeneous mobile application. The obtained architectural model is dedicated to be refined in order to specify the roles of the integrated mediators and thus obtain an architecture that describes the detailed specification for the implementation of the concrete mobile application. The classes in grey illustrated in the HMA-AD.ecore metamodel denote the architectural elements that will be used to replace the endogenous and exogenous mediators that are indicated in the architectural model.

An endogenous mediator that aims to ensure the compatibility of the exchanged data will be replaced by a mediation connector (*MediationConnector class*). This connector has a required role (*RequiredRole class*) and another provided (*ProvidedRole class*). The required role is dedicated to support the data to be transformed which is provided by the source entity whereas the provided role aims to support the transformed data extracted from this mediation connector with the objective to transfer this obtained result to the target entity.

As we have stated previously, this connector needs to know the type of data in transforming and the required type in which the data must be transformed in order to call the appropriate mediation service (i.e. to perform the necessary transformations). These informations will be represented via *InputDataType class* and *OutputDataType class*. A mediation connector aims to compare these types of data in order to execute, in the case of non-compatibility, two methods named respectively *Search-MD* and *Call-MD*. The first method is used to search the necessary mediation service to ensure the compatibility of heterogeneous data and the second one to execute it.

An exogenous mediator aims to provide well-formed and compatible interfaces in order to ensure the communication between exogenous concrete entities. It will be replaced by a new software entity that is represented by *Component-of-services class*. This architectural element encapsulates entities of service type that are connected with software components (see the composition relation between *Service class* and *Component-of-services class*). Therefore, this new entity is regarded itself as a software component (parent entity) which includes a set of services and has provided ports (*ProvidedPort class*) and required ports (*RequiredPort class*). The required and provided services (*RequiredService class* and *ProvidedService class*) of the encapsulated service entity will be transformed respectively into required and provided ports for the parent entity. The input data represented by the required port will be transferred via a request (*Request class*) to trigger the appropriate service.

Consequently, a response (*ResponseRequest class*) for this request will be retrieved and transmitted to a provided port of the parent entity.

AQ2: Reference "Biermann et al., 2006" is not included in the reference list. Please provide the reference details to be included in the reference list, or delete the citation if not required.

4.2 Graphical modelling of architectural elements

After defining the architectural description language of heterogeneous mobile applications, we now aim to provide a graphical representation for the instances of this language. The instantiation of an architectural model reflects the operation of creating a model conforms to the metamodel defined in EMF technology which can be done in two ways:

a. Instantiate the proposed metamodel and have a model in XMI format.

b. Generate a graphical editor from the proposed metamodel using GMF technology (Graphical Modelling Language) (Biermann et al., 2006)[AQ2] to allow the creation of graphical models that conform to this metamodel.

In this research work, we adopted GMF technology to generate specific description palette which allows drawing graphically the architectural model for any mobile application. Figure 5 shows the proposed graphical palette that defines all necessary graphical elements to describe the architectural model of the desired mobile application. The graphic syntax proposed to express the different architectural elements introduced in the HMA-AD.ecore metamodel is indicated in Section 4.4 by Figures 6 and 7.

Therefore, each concept defined in the HMA-AD.ecore metamodel has its equivalence in graphic architectural elements that are dedicated to graphically define the heterogeneous mobile application's architecture.

Figure 4 HMA-AD metamodel.ecore

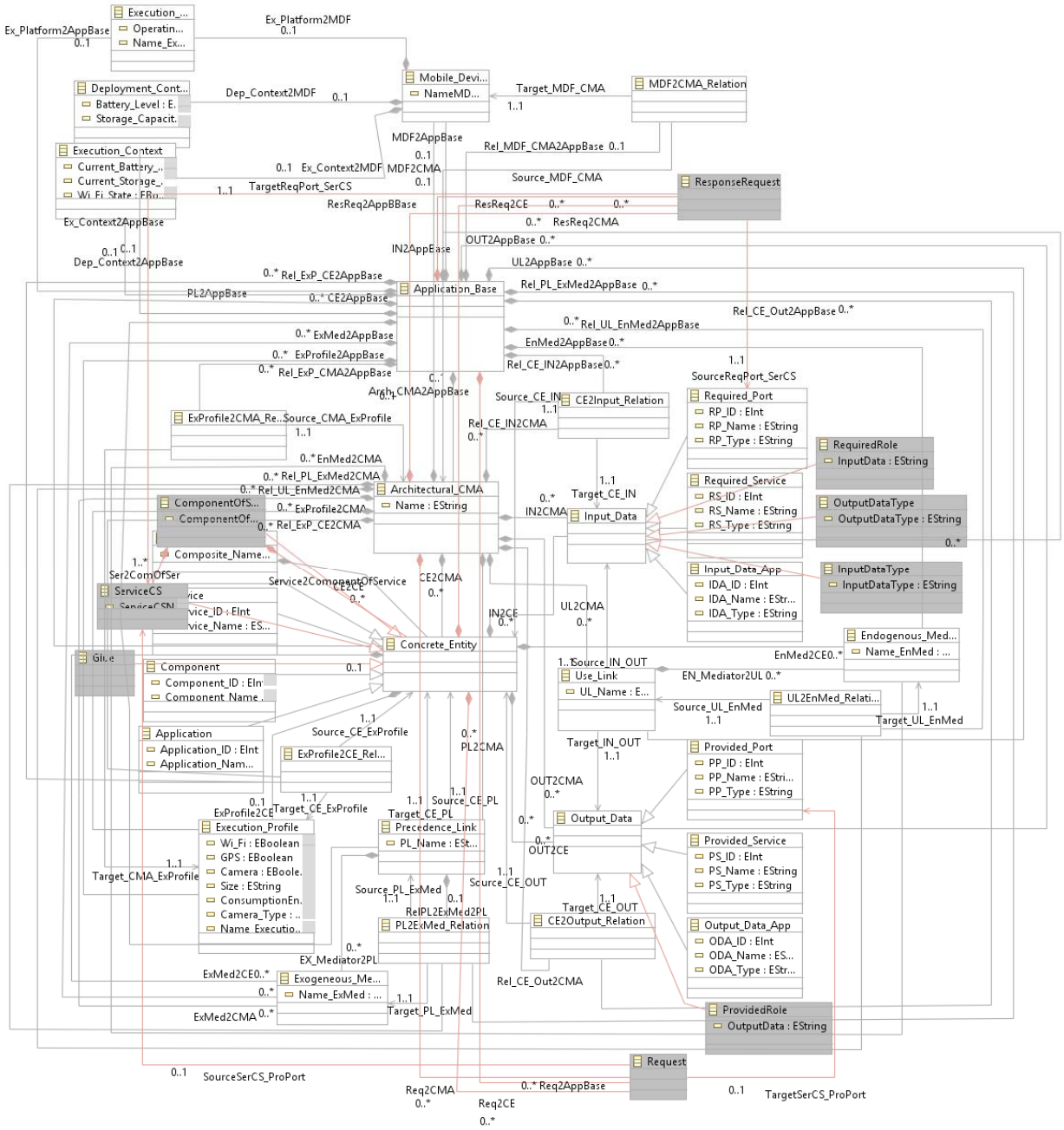
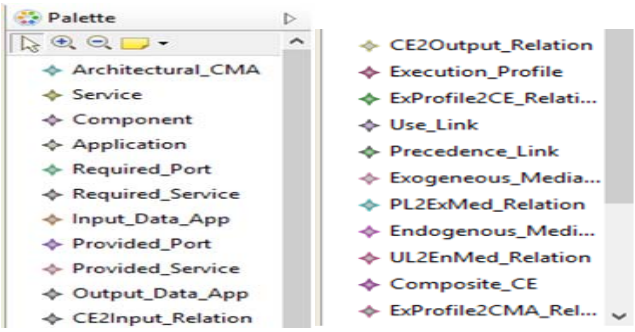


Figure 5 Generated graphic palette



4.3 Crossing algorithm to obtain detailed architectural description

The passage from the architectural model to the detailed architectural description reflects a substitute operation which is performed by running a sequence of transformation rules defined in a specific order while respecting a set of composition constraints. This passage is done through the algorithm shown in Table 2. The proposed algorithm defines an execution strategy of necessary passage rules respecting all composition constraints to have the detailed architectural model. These constraints reflect the specifications that we have proposed to represent the endogenous and exogenous mediators. The proposed algorithm designed to manage precedence links and use links with the objective of detecting those that are associated with mediators where:

- Whenever an endogenous mediator is found, it triggers a set of rules to replace it with a mediation connector; and
- Whenever an exogenous mediator is found, it triggers a set of rules to replace it with a Component-of-service entity. After encapsulated the service entity, our algorithm must perform a test to check if the service entity is linked with another entity of the same type. So it designed to encapsulate all related entities that are of service type until it comes to a composition relation between service/component.

Table 2 Proposed crossing algorithm

	Pseudo code	*Comments*
01	For each PL do {	// Managing Precedence Links (PL)
02	If (PL$_i$ attachedwith ExMed) then {	// If we find an exogenous mediator (Ex-Med)
03	If (cibleSE is Service) then {	// If the target entity of the PL of service type
04	Execute Delete ExMed Rule ;	// Delete the Ex-Med
05	Execute Delete PL(SourceSE, CibleSE) Rule;	// Delete this PL
06	Execute Add-ComOfSer Rule;	// Add a new entity (ComponentOfService)
07	Execute EncapsuleSer (CibleSE) Rule ;	// Encapsulate the service entity in the added parent entity
08	Execute AddPL-Link (SourceSE, ComOfSer);	// Restore the PL between the source entity and the added parent entity
09	For each UL (SourceSE, CibleSE){	// Restore the use links (ULs) that were between the component entity and the service entity for that will be ULs between the component entity and the new added entity
10	ComOfSer-ReqPort :=ReqSer (CibleSE) ;	// Transforming the required service (Req-Ser) of the service entity to a required port (Req-Por) for the parent entity
11	Execute AddReqPort (ComOfSer-ReqPort, ComOfSer) Rule;	// Executing a transformation rule to add the Req-Por to the new entity.
12	Execute AddRequest (ComOfSer-ReqPort, CibleSE) Rule;	// Adding a request between the Req-Por and the service entity in order to trigger the service and transfer the required data for its execution
13	Execute Add-UL (Source-ProPort, ComOfSer-ReqPort) Rule;	// Add UL between the provided port (Prov-Por) of the source entity (component type) and the new Req-Por of the parent entity
14		
15	Execute Delete-UL ;	// Remove the previous UL
16	Execute Delete- ReqSer (CibleSE) ; }	// Remove the Req-Ser of the target service entity
17	While (CibleSE.Cible is Service) do {	// Check if the target entity which connects the service entity is also of service type
18	SourceSE := CibleSE ; CibleSE:= CibleSE.Cible ;	.....
19	Encapsule (CibleSE) ;	// Encapsulating the target entity of the encapsulated service entity because it is also of service type
20	Execute AddPL-Link (SourceSE, CibleSE) ;	// Add a PL between the source service entity and the target service entity
21	For each UL (SourceSE, CibleSE){	// Managing UL between these two entities
22	ComOfSer-ProPort :=ProSer (SourceSE) ;	// The provided service (Prov-Ser) of the source service entity becomes a Prov-Por for the parent entity
23	Execute AddProPort (ComOfSer-ProPort, ComOfSer) Rule;	// Execute the rule that adds this Prov-Por
24	Execute AddResponseRequest (SourceSE,ComOfSer-	// Add a response request between the added Prov-Por
25	ProPort) Rule;	of the parent entity and the source service entity
26	ComOfSer-ReqPort :=ReqSer (CibleSE) ;	// The Req-Ser of the target service entity becomes a Req-Por for the parent entity
27	Execute AddReqPort (ComOfSer-ReqPort, ComOfSer) Rule;	.....
28	Execute AddRequest (ComOfSer-ReqPort, CibleSE) Rule;	.....
29	Execute AddUL (ComOfSer.ProPort, ComOfSer-ReqPort)	// Adding a UL between the Req-Por (which was a Req-Ser of the target
30	Rule;	service entity) of the parent entity and the Prov-Por (that was a Prov-Ser of the entity service source)
31	Execute Delete-UL Rule ;	//Delete old LPs and PLs and add them again according to the new representation
32	Execute Delete-ProSer (SourceSE) Rule;	.....
33	Execute Delete-ReqSer (CibleSE) Rule;}	.....
34	Execute Add-PL (ComOfSer, CibleSE);	.....
35	For each UL (SourceSE, CibleSE) {	.....
36	ComOfSer-ProPort :=ProSer (CibleSE) ;	.....
37	Execute AddProPort (ComOfSer-ProPort, ComOfSer) Rule;	.....
38	Execute AddUL (ComOfSer.ProPort, CibleSE.ReqPort) Rule;	.....
39	Delete UL; Delete ProSer (SourceSE); }}}	.....
40	For each UL do {	// Managing ULs
41	If (UL$_i$ attachedwith EnMed) then	// Check if UL is attached with an End-Med
42	{Execute Delete-UL (SourceSE.ProPort,	.....
43	CibleSE.ReqPort) Rule;	.....
44	Execute Delete-EnMed Rule;	.....
45	Execute Add-MedConnector Rule;	// Replace the End-Med with a mediation connector
46	Execute Add-UL	// Connect the Req-Role of the mediation connector with the Prov-Por
47	(SourceSE.ProPort, MedConnector.ReqRole) Rule;	of the source service entity
48	Execute Add-UL (MedConnector.ProRole,CibleSE.ReqPort	// Connect the Prov-Role of the mediation connector with the Req-Por
49	) Rule; }}	of the target service entity

ATL language (Atlas Transformation Language) represents our technological choice to define the different passage rules and Java technology to invoke and execute them. Our algorithm is based-on XML structures of the architectural models defined by the designer in order to choose suitable ATL rules and trigger them in a specific order for generating the appropriate target models (i.e. detailed architecture). Thus, it aims to extract the necessary information from these XML structures (i.e. browse the architectural model) to perform various tests in order to identify the type of rule to trigger.

4.4 Applicability: describing ShopReview mobile application architecture

Initially, the proposed multi-paradigm gives the hand to the designer to model its mobile application through homogeneous or heterogeneous software entities and to identify, if exist, the various heterogeneous points raised by the coordination of entities of different types and/or manipulate not-compatible data. Figure 6 shows an example of an architectural model of heterogeneous mobile application called *ShopReview*. This application is used to provide the nearby shopping where the same product, that user wants to buy, is sold at a better price (i.e. suitable price). Also, it allows the user to publish the price of the product that he found in some shops selected from those that are close to its current geographic situation (Cugola et al., 2014).

Our multi-paradigm allows firstly describing the architecture of the application by specifying the constituent entities, their invocation orders, and exchanged data between them and also attaching adaptation mediators to the heterogeneous composition relations using the proposed graphic palette shown previously. Table 3 introduces the composition scenario that we proposed to construct the *ShopReview* mobile application.

Owing to the heterogeneity presented by the connected concrete entities and the data exchanged between them, the corresponding composition relations need to be associated with mediators to deal with these problems of heterogeneity.

In our example, the different precedence links that connect the entity *Acquire Photo* and the entity *Read Barcode*, *Get Product-Name* and *Input Price*, *Input Price* and *Search Price*, *Search Price* and *Get Position*, *Get Position* and *Search the Neighbrhood* must be attached with exogenous mediators, while there is one problem of data exchange which is between the entity *Acquire Photo* and the entity *Read Barcode* (jpg≠WebP) and therefore the corresponding use link must be attached with an endogenous mediator.

After obtaining the architectural model of the desired application, our proposed paradigm aims to automatically generate a detailed specification for this application by integrating the functioning of attached adaptations mediators (i.e. resolve encountered heterogeneity problems).

Thereby, our paradigm intended to refine the designed architectural model whose objective is to provide a more detailed description facilitating thereafter the generation of the concrete mobile application. Figure 7 shows the generated detailed architectural model for the mobile application *ShopReview*. We must also draw attention that each composition relation will be attached by the type of treated composition. As an example, we read the composition between the entity *Acquire Photo* and *Read Barcode* an *exogenous heterogeneous composition*.

Table 3 Composition scenario description

	Functionality	*Impl-Type*	*Required data*	*Provided data*
Constituent software entities	*Acquire Photo*	Component	–	Photo of jpg type
	Read Barcode	Service	Photo of WebP type	Barcode of integer type
	Get Product-Name	Service	Barcode of integer type	Name of string type
	Input Price	Component	Name of string type	Price of integer type
	Search Price	Service	Price of integer type Name of string type	Convenient price of integer type
	Get Position	Component	–	Position of string type
	Search the Neighbrhood	Service	Position of string type Convenient price of integer type	Nearby Shops of string type
	Share Price	Service	Nearby shops of string type Convenient price of integer type	–
Invocation orders	*Acquire Photo* → *Read Barcode* → *Get Product-Name* → *Input Price* → *Search Price* → *Get Position* → *Search the Neighbrhood* → *Share Price.*			
Collaboration schema of exchanged data	*Read Barcode* needs *Photo* of *WebP* type, *Get Product-Name* needs *Barcode* of *integer* type, *Input Price* needs *Name* of the product of *String* type, *Search Price* needs *Price* of *Integer* type and *Name* of the product of *String* type, *Search the Neighbrhood* needs *Position* of *String* type and *Convenient Price* of *Integer* type, *Share Price* needs *Convenient Price* of *Integer* type and *Nearby Shops* of *String* Type.			

Figure 6 Architectural model of the *ShopReview* mobile application (see online version for colours)

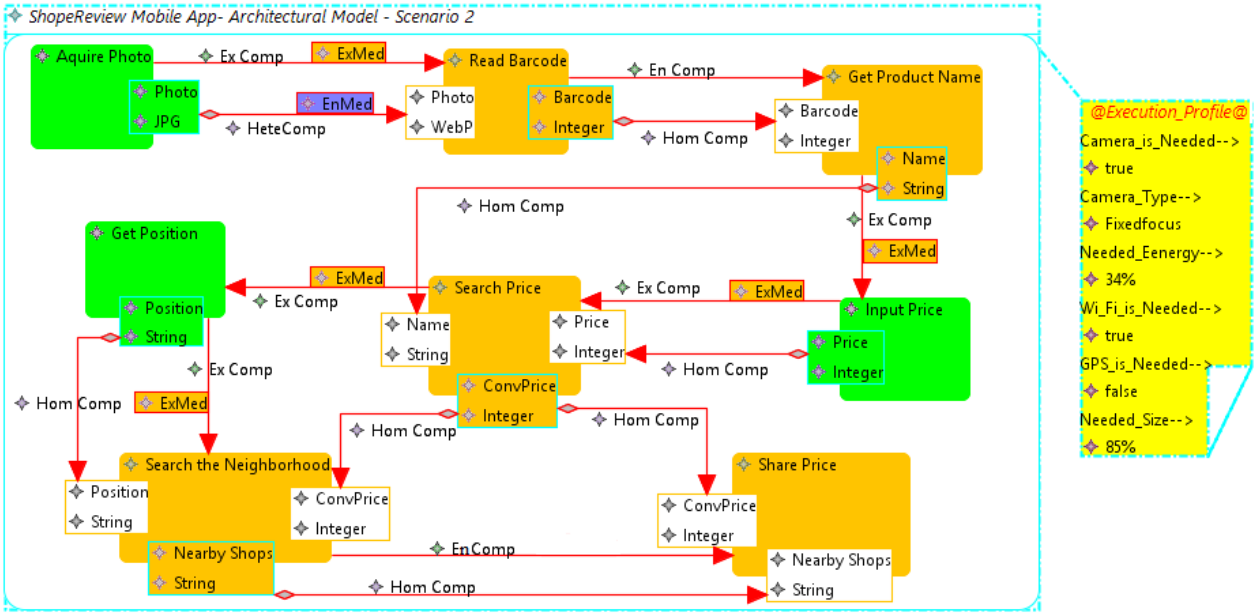
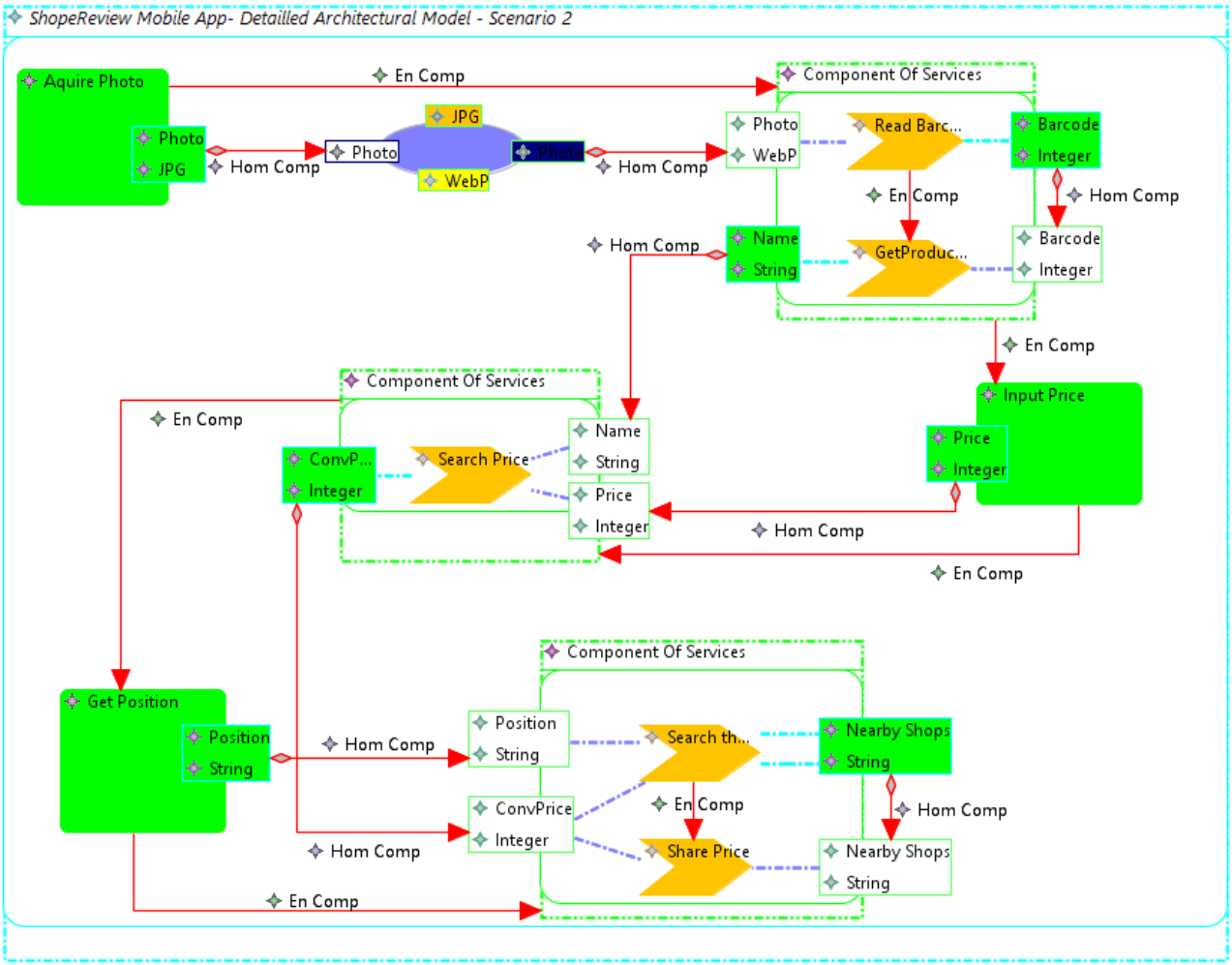


Figure 7 Detailed architectural model of *ShopReview* mobile application (see online version for colours)



5 Conclusion

In this paper, we proposed a new multi-paradigm to facilitate for designers the modelling of heterogeneous mobile applications in order to meet user's needs and to cope with the mobile devices heterogeneity (i.e. obtaining adaptive mobile applications).

Firstly, we proposed a metamodel called HMA-AD which allows modelling the functional core of any mobile application using heterogeneous conceptual bricks. After, a detailed specification for the implementation of the concrete application will be obtained through an algorithm. This latter performs a set of passage rules in a specific order in accordance with the proposed specifications for adaptation mediators required to eliminate heterogeneity problems.

Therefore, we presented in this paper a conceptual regard for the heterogeneous mobile applications including their conception and the specification of the necessary adaptation mediators.

We plan in our future work to realise the proposed exogenous and endogenous mediators by providing the concrete structure of the new entity *component-of-services* as well as the *mediation connector* based on the conceptual structures proposed in this paper.

Reference

Allen, R.J., Garlan, D. and Ivers, J. (1998) 'Formal modeling and analysis of the HLA component integration standard', *ACM SIGSOFT Software Engineering Notes*, Vol. 23, No. 6, pp.70–79.

Amirat, A., Hock-Koon, A. and Oussalah, M.C. (2014) 'Object-oriented, component-based, agent-oriented and service-oriented paradigms in software architectures', in Oussalah, M.C. (Ed.): *Software Architecture 1*, Wiley Online Library, New York, pp.1–53.

Beisiegel, M. (2007) *Service Component Architecture Specification*, Technical Report.

Beisiegel, M., Bloom, H., Booz, D., Dubray, J.J., Colyer, A. and Edwards, M. (2005) 'Service component architecture: building systems using a service oriented architecture', *Whitepaper*, Vol. 1, 31pp. Available online at: http://www.iona.com/devcenter/sca/SCAWhitePaper109.pdf

Blanc, X. and Salvatori, O. (2011) *MDA en action: Ingénierie logicielle guidée par les modéles*, Editions Eyrolles, Paris, 292pp. [In French]

Brel, C. (2013) *Composition d'applications multi-modéles dirigée par la composition des interfaces graphiques*, Doctoral Thesis in Computer Science, 28 June, Université Nice Sophia, Antipolis, 204pp. [In French]

Brown, A.W. and Wallnau, K.C. (1998) 'The current state of CBSE', *IEEE Software*, Vol. 15, No. 5, pp.37–46.

Bruneton, E., Coupaye, T., Leclercq, M., Quéma, V. and Stefani, J-B. (2006) 'The FRACTAL component model and its support in java', *Software: Practice and Experience*, Vol. 36, No. 11, pp.1257–1284.

Cai, X., Lyu, M.R., Wong, K-F. and Ko, R. (2000) 'Component-based software engineering: technologies, development frameworks, and quality assurance schemes', *APSEC '00 Proceedings of the Seventh Asia-Pacific Software Engineering Conference*, IEEE, IEEE Computer Society, Washington, DC, pp.372–379.

Cavallaro, L., Di Nitto, E. and Pradella, M. (2009) 'An automatic approach to enable replacement of conversational services', in Baresi, L., Chi, C. and Suzuki, J. (Eds): *Service-Oriented Computing*, Springer, Berlin, pp.159–174.

Chemaa, S., Bouarioua, M. and Chaoui, A. (2015) 'A high-level Petri net based model for web services composition and verification', *International Journal of Computer Applications in Technology*, Vol. 51, No. 4, pp.306–323.

Chen, D., Zhu, X., Dai, W. and Zhang, R. (2014) 'Socially aware mobile application integrations in heterogeneous environments', *International Journal of High Performance Computing and Networking*, Vol. 8, No. 1, pp.61–70.

Christensen, E., Curbera, F., Meredith, G. and Weerawarana, S. (2001) *Web Services Description Language (WSDL) 1.1*, W3C Note 15 Marsh. Available online at: Http:// www.w3.org/TR/wsdl

Cugola, G., Ghezzi, C., Pinto, L.S. and Tamburrelli, G. (2014) 'SelfMotion: a declarative approach for adaptive service-oriented mobile applications', *Journal of Systems and Software*, Vol. 92, pp.32–44.

Dashofy, E.M. and Van Der Hoek, A. (2002) 'Representing product family architectures in an extensible architecture description language', in van der Linden, F.J. (Ed.): *Software Product-Family Engineering*, Springer, Bilbao, Spain, pp.330–341.

Derdour, M., Dalmau, M., Roose, P. and Ghoualmi-Zine, N. (2010) 'Typing of adaptation connectors in MMSA approach case study: sending MMS', *International Journal of Research and Reviews in Computer Science*, Vol. 1, No. 4, pp.39–49.

Derdour, M., Roose, P., Dalmau, M., Ghoualmi-Zine, N. and Alti, A. (2010) 'MMSA: metamodel multimedia software architecture', *Advances in Multimedia*, Vol. 2010, Article ID 386035, 17pp.

Desertot, M., Cervantes, H. and Donsez, D. (2006) 'FROGi: fractal components deployment over OSGi', in Löwe, W. and Südholt, M. (Eds): *Software Composition*, Springer, Berlin, Heidelberg, pp.275–290.

Diaw, S., Lbath, R. and Coulette, B. (2010) 'Etat de l'art sur le développement logiciel basé sur les transformations de modéles', *Technique et Science Informatiques*, Vol. 29, Nos. 4–5, pp.505–536. [In French]

Garlan, D. and Perry, D.E. (1995) 'Introduction to the special issue on software architecture', *IEEE Transactions on Software Engineering*, Vol. 21, No. 4, pp.269–274.

Geebelen, K., Michiels, S. and Joosen, W. (2008) 'Dynamic reconfiguration using template based web service composition', *Proceedings of the 3rd Workshop on Middleware for Service Oriented Computing*, ACM, Leuven, Belgium, pp.49–54.

Hock-Koon, A. (2011) *Contribution à la compréhension et à la modélisation de la composition et du couplage faible de services dans les architectures orientées services*, Doctoral Thesis in Computer Science: Software Engineering, 28 April, University of Nantes, 205pp. [In French]

Hock-Koon, A. and Oussalah, M. (2010) 'Composite service metamodel and auto composition', *Journal of Computational Methods in Sciences and Engineering*, Vol. 10, No. 1-2S2, pp.215–229.

Hourdin, V., Tigli, J-Y., Lavirotte, S., Rey, G. and Riveill, M. (2008) 'SLCA, composite services for ubiquitous computing', *Mobility '08 Proceedings of the International Conference on Mobile Technology, Applications and Systems*, ACM, Singapore, pp.1–8.

Ickin, S., Wac, K., Fiedler, M., Janowski, L., Hong, J.H. and Dey, A.K. (2012) 'Factors influencing quality of experience of commonly used mobile applications', *Communications Magazine*, Vol. 50, No. 4, pp.48–56.

Jones, M. (2013) 'Developing mobile apps for your users can help them be more productive, but before you start building apps, learn about the different kinds', *Everything You Need to Know about Developing Mobile Apps*, TechTarget. Available online at: http://searchmobilecomputing.techtarget.com/feature/ Everything-you-need-to-know-about-developing-mobile-apps (Accessed on 25 May 2015).

Kalamegam, P. and Zayaraz, G. (2016) 'Requirements driven test prioritisation approach for web service composition', *International Journal of Computer Applications in Technology*, Vol. 54, No. 4, pp.362–370.

Kalasapur, S., Kumar, M. and Shirazi, B.A. (2007) 'Dynamic service composition in pervasive computing', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 18, No. 7, pp.907–918.

Luckham, D.C., Kenney, J.J., Augustin, L.M., Vera, J., Bryan, D. and Mann, W. (1995) 'Specification and analysis of system architecture using Rapide', *IEEE Transactions on Software Engineering*, Vol. 21, No. 4, pp.336–354.

Luckham, D.C. and Vera, J. (1995) 'An event-based architecture definition language', *IEEE Transactions on Software Engineering*, Vol. 21, No. 9, pp.717–734.

Maheswari, S. and Karpagam, G. (2015) 'Enhancing fuzzy Topsis for web service selection', *International Journal of Computer Applications in Technology*, Vol. 51, No. 4, pp.344–351.

Martin, D., Burstein, M., Lassila, O., Paolucci, M., Payne, T. and McIlraith, S. (2004) *Describing Web Services Using OWL-S and WSDL*, Release 1.2 of OWL-S. Available online at: http://www.daml.org/services/owl-s/1.2/owl-s-wsdl.html

Medvidovic, N., Rosenblum, D.S. and Taylor, R.N. (1999) 'A language and environment for architecture-based software development and evolution', *Proceedings of the 1999 International Conference on Software Engineering*, IEEE, Los Angeles, CA, USA, pp.44–53.

Medvidovic, N. and Taylor, R.N. (2000) 'A classification and comparison framework for software architecture description languages', *IEEE Transactions on Software Engineering*, Vol. 26, No. 1, pp.70–93.

Papazoglou, M.P., Traverse, P., Dustdar, S. and Leymann, F. (2007) 'Service-oriented computing: state of the art and research challenges', *Computer*, Vol. 40, No. 11, pp.38–45.

Papazoglou, M.P. and Van Den Heuvel, W-J. (2007) 'Service oriented architectures: approaches, technologies and research issues', *International Journal on Very Large Data Bases (VLDB)*, Vol. 16, No. 3, pp.389–415.

Soucé, J-M. and Duchien, L. (2002) *Etat de l'art sur les langages de description d'architecture*, Technical Report, June 2002, Livrable1.1-2 du Projet RNTL ACCORD, 62pp. [In French]

Srinivasan, L. and Treadwell, J. (2005) 'An overview of service-oriented architecture, web services and grid computing', *HP Software Global Business Unit*, 3 November. Available online at: http://h71028.www7.hp.com/ERC/downloads/SOA-GridHP-WhitePaper.pdf (Accessed on 7 June 2016).

Statista (2015) *Number of Mobile App Downloads Worldwide from 2009 to 2017*, Firme de Recherche Statista, Statistique sur les Applications mobiles téléchargées. Available online at: http://www.statista.com/statistics/266488/forecast-of-mobile-app-downloads/ (Accessed on 22 January 2016).

Stojanovic, Z. and Dahanayake, A. (2005) 'Service-oriented software system engineering: challenges and practices', *Journal of Digital Information Management, Digital Information Research Foundation*, Vol. 3, No. 3, p.210.

Weinreich, R. and Sametinger, J. (2001) 'Component models and component services: concepts and principles', in Heineman, G.T. and Councill, W.T. (Eds): *Component-based Software Engineering: Putting the Pieces Together*, 1st ed., Addison-Wesley Professional, Reading, MA, pp.33–48.