Extending MoWebA for MobileApps with Functions in the Cloud*

Emanuel Sanchiz, Magalí González, Nathalie Aquino, and Luca Cernuzzi

Universidad Católica “Nuestra Señora d e la Asunción” - DEI, Asunción, Paraguay
{emanuel.sanchiz, mgonzalez, nathalie.aquino, lcernuzz}@uc.edu.py

Abstract. Currently, a growing interest is being caused by the mobile applications and the cloud. In this work, we have focused on mobile applications with functions implemented in the cloud (MobileApps-FC). The improvement related to the portability of these applications among different platforms and different service providers is an interesting challenge. Model Driven Development (MDD) constitutes one of the alternatives to address portability issues. Indeed, in this work we propose MoWebA Mobile, an extension of a MDD approach, called MoWebA, for the design and generation of the MobileApps-FC. Specifically, in this work we have focused on a specific aspect of the mentioned applications, which is the network communication between the mobile applications and their functions in the cloud. Furthermore, as a preliminary validation we present a comparative study with MoWebA Mobile regarding to the traditional development and WebRatio Mobile Platform.

Keywords: Mobile applications · Cloud · Network communication · MDD · Portability

1 Introduction

In recent years, the number of users of smartphones and tablets has been increasing quickly, overcoming the number of users of computers and notebooks.[1] Consequently, the interest for the development of mobile applications has also increased,[2] particularly for native applications.[3] Moreover, it is important to consider the current growing market of the cloud, specially the public cloud,[4] and its role as enhancer of the restricted resources of mobile devices [1,2]. Taking into account both perspectives, mobile devices and the cloud, it is worth noting

* This work has been funded by CONACYT through the PROCIENCIA program with resources from “Fondo para la Excelencia de la Educación e Investigación - FEEI” from FONACIDE. This work has been developed under the project “Mejorando el proceso de desarrollo de software: propuesta basada en MDD” (14-INV-056).

[1] Morgan Stanley, link: `https://goo.gl/Ifznvz`

[2] Mobile is all about application, link: `https://goo.gl/tCnZfN`

[3] Native vs hybrid application, link: `https://goo.gl/JH1u7K`

[4] Growing of public clouds, link: `https://goo.gl/WOVYJW`

the growing convergence between them [3]. Hence, the interest of this study is focused on mobile applications which have at least one module or implementation (e.g., procedure, service, database) running in the cloud. We refer to these applications as mobile applications with functions in the cloud (MobileApps-FC).

On one side, the mobile environment can be based on several different platforms, being iOS and Android the most popular ones. These platforms, again, consider different operating systems, programming languages, tools, etc. In most cases, mobile applications must be developed for more than one mobile platform. And due to differences among these platforms, specific application versions must be built for each one of them. Therefore, more effort and time must be employed, resulting in higher costs in the development process. Since the same implementation can hardly be used for different mobile platforms, there is a portability challenge in the mobile environment [3,4,5].

On the other side, the cloud environment is constituted by several different service providers. In a similar way than previously exposed, each of these providers uses its own specific platform (operating systems, programming languages, libraries, etc.). This causes a portability problem, which in the cloud context is known as vendor lock-in. Therefore, tasks such as building an application that is able to run in clouds from different service providers or migrating an application from the cloud of one service provider to another one constitute interesting challenges, since the same implementation can hardly be used in different cloud service providers [6,7].

Therefore, when considering applications that include, at the same time, the mobile and the cloud environments, the portability problem is highlighted. In this sense, Model Driven Development (MDD) emerges as a possible solution. In fact, one of the main motivations of MDD is related to the improvement of portability [8]. MDD focuses efforts on designing domain models that capture the knowledge of the data and functionality that an application must provide independently of the platform in which it will be run.

Therefore, our goal is to propose a model driven approach for the modeling and generation of the MobileApps-FC. In this case, the scope of the modeling and the generation focuses on one aspect of the applications, which is the network communication. We present our proposal as an alternative for addressing the extra effort caused by the difficulty of platform portability.

The model driven approach adopted for this work is MoWebA [11]. Based on MoWebA, we have identified three aspects that could have a positive impact in the portability problem that is found in the MDD development of MobileApps-FC. These aspects are: i) incorporation of an Architecture Specific Model (ASM) as a new modeling layer, in order to keep the portability of the Platform Independent Model (PIM) regarding the different architectures (e.g., RIA, SOA, Mobile); ii) clear separation of the presentation layer with regard to the navigation and behavior layers, would prevent additional complications to the presentation, since, it is the layer that presents more difficulties related to portability; and iii) definition of the navigational structure according to a function-oriented approach, which prevents the modification of the navigation design caused by implementation changes. Such prevention means a saving of effort, which, finally, contribute to alleviate the effects introduced by the portability difficulty. We have studied the adoption of such aspects in the state of the art, through a Systematic Mapping Study (SMS) [9], which results are summarized in the next section.

The rest of the document is structured as follows. Section 2 analyzes the related works. Then, Section 3 presents the proposal, which is the extension of MoWebA through an ASM for the modeling and generation of the network communication aspect. Section 4, describes a comparative study as a preliminary validation of our proposal. Finally, Section 5 presents the conclusions and future works.

2 Related Works

In this section, due to the space limitation, we present a brief summary of the results of a SMS of the state of the art we have done. More details about the study can be found in Sanchiz et al. [9]. In such SMS, we have identified six model driven proposals for the development of MobileApps-FC that address the portability problem. Such proposals are: 1) WebRatio Mobile Platform [4], a framework based on IFML;[5] 2) SIMON [10], a framework including runtime models based on XML; 3) MD^2 [11]; 4) MobiCloud [12] and 5) Steiner et al. [13], which present a textual modeling language based on the Model-View-Controller (MVC) schema; and, 6) Ruokonen et al. [14], which presents a modeling based on business process models. In Table 1, we summarize the differences between the identified proposals and MoWebA, based on the adoption of the three design aspects: i) incorporation of an ASM; ii) clear separation of the presentation layer with regard to the navigation and behavior layers; and iii) definition of the navigational structure according to a function-oriented approach.

Table 1: Comparison based on the design aspects, more details in [9]

Work	**Aspects**		
	ASM	**Clear Separation of Presentation**	**Function Oriented Navigation**
WebRatio Mobile	Yes	No	Yes
SIMON	Partially	No	No
MD^2	No	Yes	Yes
MobiCloud	No	No	No
Steiner et al.	No	No	Yes
Ruokonen et al.	No	No	No

This review showed us that none of the proposals considered includes all the three MoWebA's design aspects, which could have a positive impact on the

[5] IFML - OMG standard; link: `http://www.omg.org/spec/IFML/`

design portability. Consequently, we were motivated to adopt MoWebA and to extend it for the development of the MobileApps-FC. Specifically, in the present study, we are focusing on the extension for the modeling and the generation of the network communication of the MobileApps-FC.

Furthermore, from the mentioned proposals we also have learned some aspects like: i) the use of standard modeling languages. (WebRatio uses IFML models while SIMON uses XML for textual models); ii) the adoption of the MVC schema, (derived from MobiCloud, MD^2 and Steiner et al.); iii) the adoption of a unified modeling (derived from MD^2, MobiCloud and Steiner et al.); iv) the combination of MDD and the open source approach (derived from WebRatio); v) the use of REST as a uniform and portable communication interface, (derived from WebRatio, MD^2, SIMON, MobiCloud); vi) the generation of native mobile applications (generated by MobiCloud, MD^2, SIMON and Steiner et al.). MoWebA considers natively the aspects i) and ii). While the other ones, iii), iv), v) and vi), we are including into MoWebA Mobile, which is the extended version of MoWebA.

3 Extending MoWebA for the Network Communication

In this section, we present the extension of MoWebA to model and to generate the network communications functions for the MobileApps-FC. Furthermore, we describe the extensions we have applied to one of the PIM models' diagrams of MoWebA, the logic one. From such extension we obtain the ASM which enables the modeling of the network communication of the MobileApps-FC. Similarly, we will present the transformation rules which map the ASM with the code to be generated. Moreover, we will explain the modeling and generation process. Finally, we present an example to show the use of our proposal.

3.1 Network Communication Functions

We have focused on the network communication aspect because its implementation necessarily implies working with different platforms and technologies (iOS, Android and the cloud service providers' platforms) of the MobileApps-FC. In this sense, the platform abstraction which proposes a MDD approach is highlighted on the network communication aspect. Starting from the official documentations of Android[6] and iOS[7] (currently, the most popular platforms)[8] we focused on four types of network communication functions. There are more variants of such functions, but we have focused on such four ones to cover some of the most common cases. Those communication functions set the scope of this work for the design and the generation implementation. Following, we describe the types of such functions: i) *light-data*, where the data exchange does not include files (e.g., images, documents, video or audio). In this sense, the data to

[6] Android, `https://goo.gl/LwMLXn`

[7] iOS, `https://goo.gl/3FlZqW`

[8] Popularity of Android and iOS, `https://goo.gl/ZAu8Ho`

exchange is *light*; ii) *load-image*, to get and to load images in memory for displaying them. Commonly, this function is used for image and video previews; iii) *download-files*, to download files in background; and iv) *upload-files*, to upload files in background.

The mentioned functions include implementations in both sides, the mobile and the cloud, respectively. Such implementation are based on the REST architecture [15].

3.2 MoWebA Mobile: the Network Communication Proposal

The definition process for extending MoWebA for different architectures consists in the following steps [16]: i) define the metamodel using MOF;[9] ii) define the corresponding UML[10] Profile; iii) define transformation rules for elements that can be obtained in an automatic way; iv) apply transformation rules in order to obtain the first version of the ASM from the PIM; v) make necessary manual adjustments to complete the ASM model; vi) generate the target code applying transformation rules; and, vii) include manual adjustments, if necessary.

The scope of this work has focused, on one side, on the steps i) and ii) for defining the ASM metamodel and profile and, on the other side, on the step iii) and vi) for defining model to text transformation rules and to apply them to get the target code from the ASM. It was not necessary the step vii). We have left for future works the automatic transformation from the PIM to the ASM, which includes the steps iv) and v). We illustrate the scope of the extension we have made in Figure 1.

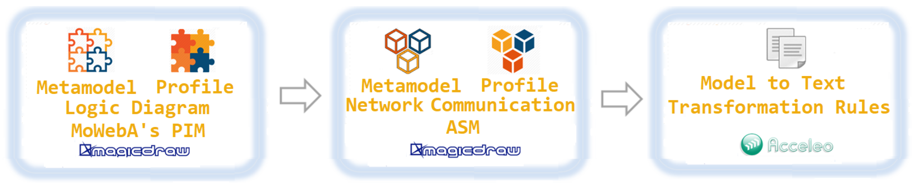

Fig. 1: The extension includes a metamodel, a profile and transformation rules

In this work, we have defined the metamodel and profile for obtaining the network communication ASM. Such metamodel and profile extend the logic diagram of MoWebA. Such logic diagram enables the definition of logic processes. In fact, we consider the network communication as a logic process. The logic diagram contains *TProcesses*, which are the logic processes defined. Such processes include *Services*, which are procedures doing a specific task. Also, the logic diagram has *ValueObjects*, which group attributes of entities and enable the access to the entities' data.

[9] MOF, link: `http://www.omg.org/mof/`
[10] UML, link: `http://www.uml.org/`

Following we describe the extensions we made on the logic diagram. The extensions for obtaining the ASM are based on the REST architecture and on the four types of network communication functions, presented in the previous section. Such elements belong to the metamodel showed in Figure 2.

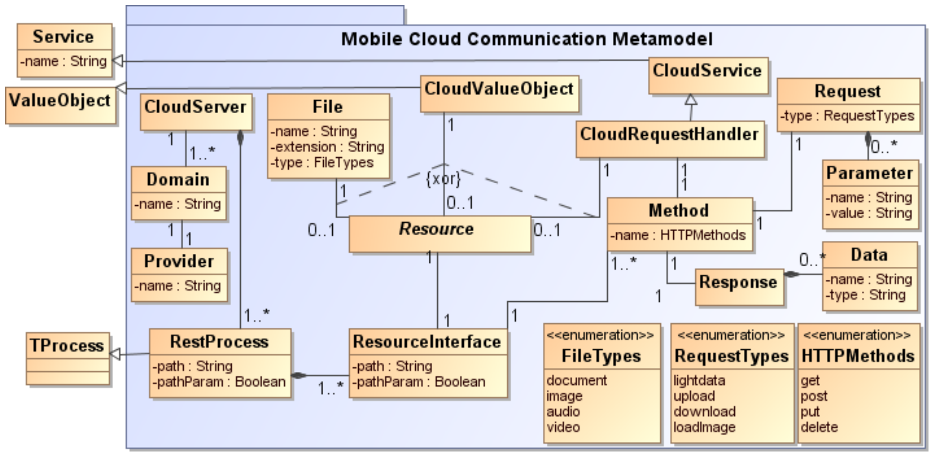

Fig. 2: Metamodel for the network communication ASM

Firstly, it is defined a *CloudServer*, which is accessed through a *Domain*. The *Domain* is provided by a service *Provider*, which enables the recognition of the particular configurations required by each cloud service provider. Then, each *CloudServer* contains a set of logic processes, where each one of them is called *RestProcess*. Such processes modularize the network communication design. Subsequently, each *RestProcess* includes a set of resource interfaces, where each *ResourceInterface* is associated with a *Resource*, one at a time. A *Resource* can be a MoWebA's *ValueObject* called *CloudValueObject* because it is residing in the cloud, a *File* stored or to be store in the cloud or a MoWebA's *Service* called *CloudRequestHandler*, which is executed in the cloud.

Each *ResourceInterface* is associated to a set of methods, where each *Method* defines the operation to be performed on a *Resource*. At the same time, each *Method* is associated to the object *Request* and, in some cases, to the object *Response*. On one side, each *Request* can be associated to a set of parameters, which contains each *Parameter* of the request. On the other side, each *Response* is associated to a set of data expected to receive from a *light-data Request*, operated by the *Method get*.

Following, we will explain the attributes of representative elements described so far. The *RestProcess* has a relative *Path*, which makes the process accessible, and a boolean flag which establish if the *Path* is used as additional data in each *Request*. For instance, the *Path* could be a user identification. The same case is for the attributes of the *ResourceInterface*. About *File*, its attributes specify the *name*, *extension* and the file *type*. The *Name* of *Method* defines the type of HTTP method (*get, post, put, delete*). The four HTTP methods considered are the most common ones. The *Request type* (*lightdata, download, upload, loadImage*) defines the call to be done. In case of *lightdata*, the associated *Method* can be any of the mentioned HTTP methods. Nevertheless, *download* (*get*), *upload* (*post*) and loadImage (*get*) have, each one, a predefined *Method*. The *name* and the *value* of the *Parameter* describe the different parameters of the *Request*. Finally, the *name* of *Data* specify the data to be received from the *Request lightdata* and the *Method get*. Such *name* specifies an attribute of the *CloudValue Object* associated to the respective resource interface.

The described elements of the ASM are represented concretely in the respective profile.[11] The profile contains the definition of the elements (stereotypes, tag values and enumerations), which enable the modeling.

3.3 Transformation Rules

We have defined model to text (M2T) transformation rules in order to generate the target code from the ASM. In order to built the transformation rules, we have followed an approach based on templates. We have used the Model transformation language (MTL) of Acceleo[12] for defining the templates. Similarly, we have used OCL for doing queries to the model. Furthermore, we have built a service in Java to extend the functions of the MTL. We have built the transformation rules based on the classes, properties and operations characterized by the respective stereotypes, tag values and enumerations defined in the ASM's profile. In this sense, such rules perform a mapping between the model elements defined and the target code to be generated. Basically, the generation for both side, mobile and cloud, depends on each combination of a *CloudServer*, a *RestProcess*, *ResourceInterface* and *CloudRequestHandler*.

The target code generated consists, on one side, in native mobile code written in Java[13] for Android, in Swift[14] for iOS, and on the other side, in open source code written in Javascript[15] with Node.js for the Openshift and Amazon Web Services platforms. Additionally, our cloud implementation is based on Docker,[16] which is a container where an application runs. Moreover, Docker is an emerging method developed by the open source community for easing the portability of cloud applications.

[11] More details about the profile, link: `https://goo.gl/DZY2yk`
[12] Acceleo, link: `https://goo.gl/jgCZhu`
[13] Java, link: `https://goo.gl/hGBggw`
[14] Swift, link: `https://developer.apple.com/swift/`
[15] Javascript, link: `https://www.javascript.com/`
[16] Docker, `https://www.docker.com/what-docker`

3.4 Modeling and Code Generation with MoWebA Mobile

For modeling we have used MagicDraw,[17] which is a modeling tool of general purpose. Once the model is done, it has to be exported as XMI files. Such files are imported from Acceleo, which is a tool for defining and executing transformation rules. Therefore, in Acceleo, the transformation rules are executed on the imported model. Consequently, the implementation of the network communication is obtained in for every platform considered. See the modeling and generation process in Figure 3.

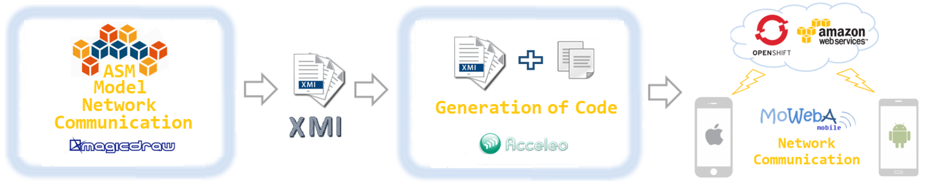

Fig. 3: Modeling and generation process of MoWebA Mobile

The generated code is a running implementation of the network communication. In other words, the generated code is ready to be executed for both mobile platforms, Android and iOS and for both cloud service providers, Openshift and Amazon Web Services.

With the aim of improving the understanding of the design process, we show an example of modeling an application. To see the complete explanation of the example, go to MoWebA Mobile Book.[18]

We have considered a application similar to one presented in Brambilla et al. [4]. The application is a virtual shop for selling products. It is going to be deployed on tablets and cell phones for field agents, i.e., salesman that go to customers for selling the products. This application requires the implementation in mobile and cloud platforms. In this case, the task is focused on the functions of network communication for the data exchange. Basically, the data to be exchanged is about the products (e.g., images, technical sheets, providers). Part of the modeling is shown in Figure 4.

In order to explain the modeling and generation example, we focus on one of the modeled functions, see Figure 5. We consider the load-image function for loading provider logos on the screen. As can be seen in Figure 5, from one model, we can obtain the implementation for every considered platform. The function implies the request on the URL formed from the attributes of the *cloudServer* (the domain server), the relative paths of the *restProcess* and the *resourceInterface*, respectively, and the name of the resource, in this case an image. The function is triggered by a button, when it is touched the image is obtained from

[17] MagicDraw, link: `https://goo.gl/mLPvur`
[18] MoWebA Mobile Book, `https://goo.gl/Qr9367`

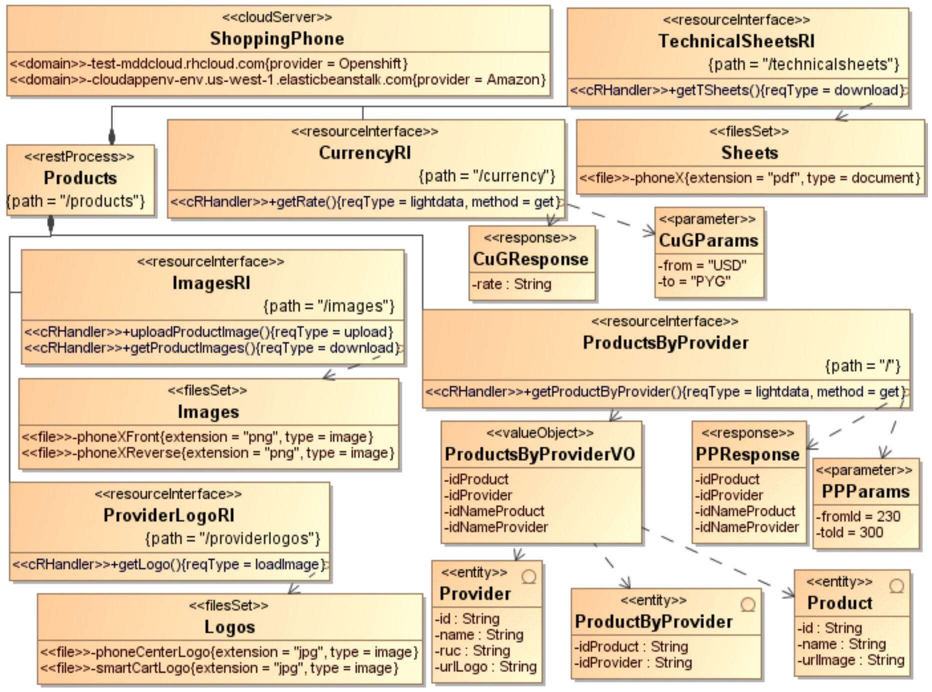

Fig. 4: Example of an application for offering products

the server, then it is loaded in memory and, afterwards, on the screen. The remote function, which handles the requests on the URL is specified in the *resourceInterface*, where it is specified as well, the type of the communication, in this case load-image. The implementation of such request handler function is generated for both cloud service providers' platforms. Similarly, we generate the implementation for every network communication function modeled.

4 Comparative Study with MoWebA Mobile

In this work, we have proposed an alternative to address the portability challenge, which consists in the extra effort introduced in the development of the MobileApps-FC. Specifically, we have focused on the modeling and generation of the network communication aspect. As a preliminary validation of our proposal, we have done a comparative study. In such study, we have measured the effort related to the development of the MobileApps-FC's network communication functions. In order to do the study, basically, we have proceeded as follows: i) we have selected an application as an example to develop its network communication; ii) we have used MoWebA Mobile, WebRatio Mobile Platform (another MDD approach [4]) and the traditional approach to develop independently the network communication; iii) we have measured and registered the development

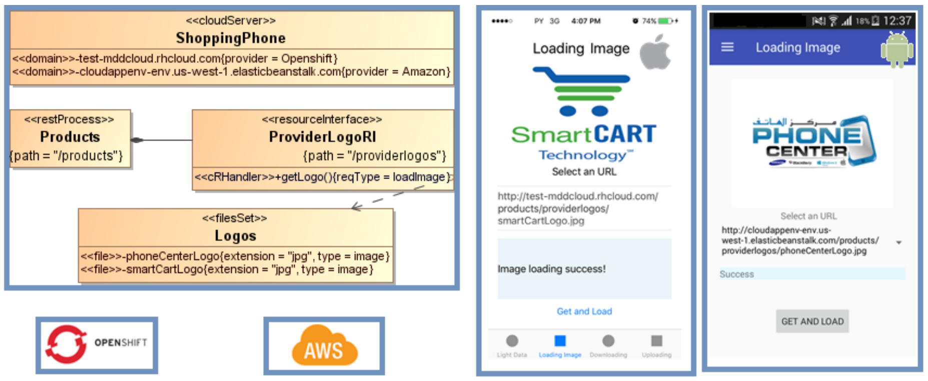
<<cloudServer>>
ShoppingPhone
<<domain>>-test-mddcloud.rhcloud.com{provider = Openshift}
<<domain>>-cloudappenv-env.us-west-1.elasticbeanstalk.com{provider = Amazon}
<<restProcess>>
Products
{path = "/products"}
<<resourceInterface>>
ProviderLogoRI
{path = "/providerlogos"}
<<cRHandler>>+getLogo(){reqType = loadImage}
<<filesSet>>
Logos
<<file>>-phoneCenterLogo{extension = "jpg", type = image}
<<file>>-smartCartLogo{extension = "jpg", type = image}
OPENSHIFT
AWS
Loading Image
SmartCART Technology
Select an URL
http://test-mddcloud.rhcloud.com/products/providerlogos/smartCartLogo.jpg
Image loading success!
Get and Load
Light Data
Loading Image
Downloading
Uploading
Loading Image
PHONE CENTER
Select an URL
http://cloudappenv-env.us-west-1.elasticbeanstalk.com/products/providerlogos/phoneCenterLogo.jpg
Success
GET AND LOAD

Fig. 5: Example of an application for offering products. Load-image function

times taken by each alternative; iv) in case of the MDD approaches, we have registered and analyzed the modeling and generation differences which could affect the development effort. Following, we describe the study in more details.

We have considered the same application presented in Section 3.4 to develop the network communication functions using the three alternatives. The development have been done in an academic environment. The developer was a computer science student in his last year at the university. In this case, we have had only one developer, which is a limitation of the study. Moreover, the resources used to guide and support the development were those available on-line.

As we have mentioned, we have compared MoWebA Mobile against the traditional approach and WebRatio Mobile Platform. In this case, we have assumed the following research question:

RQ1- How much time of development do the traditional approach, WebRatio Mobile Platform and MoWebA Mobile require to obtain the network communication implementation?

Similarly, we have compared exclusively both MDD approaches, MoWebA Mobile (A) and WebRatio Mobile Platform (B). In this case, we have focused on comparing modeling and generation aspects related to the effort, according to the next research questions:

RQ2- What differences in the modeling process between *A* and *B* could affect the effort required to develop MobileApps-FC?

RQ3- Respectively, how many platforms do A and B generate code for (mobile and cloud)?

Following, we present the results by each research question defined.

RQ1: We have compared MoWebA Mobile against the traditional approach and a consolidated MDD tool to analyze the required effort differences. Obviously, a MDD tool will improve the effort needed following a traditional approach (i.e., the manual development). However, in this case, the aim was to understand how big the difference is. WebRatio Mobile Platform, is used in the industry and it is the most representative MDD tool for the development of the MobileApps-

FC [9]. Therefore, we have compared it with MoWebA Mobile to see how much difference of effort exists with such kind of tool.

The development effort, measured through the development time, is presented in Figure 6.

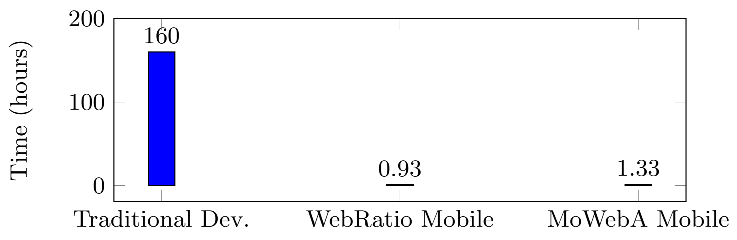

Fig. 6: Comparison of development times

It is worth noting that there exists a substantial difference of effort among the development times of the MDD approaches against the traditional one. For obtaining the same implementation, using WebRatio Mobile Platform and MoWebA Mobile have been necessary 0.93 hours (55 minutes) and 1.33 hours (1 hour 20 minutes), respectively. The traditional approach has taken 160 hours.

On one side, since, the network communication development implies, necessarily, working with several platforms and cloud service providers. In this sense, the developer had to face difficulties like the transition between different development environments (mobile, cloud), the use of different frameworks and programming languages. Such difficulties slowed down the development using the traditional approach.

On the other side, about the MDD approaches, they have several properties which support such difference of time. Firstly, the abstraction of specific details of the different platforms through the models. We highlight two advantages of such abstraction. On one side, it prevents dealing with the difficulty of working with different technologies. On the other side, it allows developers without specific platform and communication knowledge to get specific platform implementation of the network communication of the MobileApps-FC. Secondly, the automatic generation of platform specific code from the built model, which save most of the development time. Third, the generated code is already tested, so the probability to spend time in fixing bugs decreases.

Therefore, we could say that there is a considerable improvement in saving effort using a MDD approach for the development of the network communication of MobileApps-FC.

Comparing the MDD approaches, there is a difference that favors to one of them. First of all, we have to say that WebRatio Mobile Platform has a specific development environment, which eases, makes simple and faster the modeling comparing with MoWebA Mobile. Furthermore, WebRatio Mobile Platform is a robust and mature platform used in the industry. In contrast, in MoWebA

Mobile the modeling and the generation are made with tools of general purpose which slowed down the development process. Therefore, we suppose that if we build a MoWebA Mobile's specific tool, it will help to make more simple and faster the process of modeling and generation. Such eventual reduction could imply equalizing, or even improving, the network communication development time of WebRatio Mobile Platform. Following, we present further differences which reinforce the possible improvement of development time using MoWebA Mobile through a more specific tool. We refer to MoWebA Mobile as *A* and to WebRatio Mobile Platform as *B*.

RQ2: On one side, *A* prescribe all the modeling and configurations in a unified model, while *B* works with two models and projects. One project for mobile, another project for back-end. In case of *A*, the purpose of working with the same project and model for designing the communication is to abstract the developer from individuals settings and modelings by each side (mobile and cloud). In other words, since the communication implies two sides (mobile and cloud), from a unified model, at the moment of generation, the design and settings are replicated in both sides. Therefore, the developer "works once" instead of twice or more, which is the case of *B*. For instance, if a url is modified, then, the developer does not modify it for the cloud side and for the mobile side, the url modification is made only once in the model. Afterwards, thanks to the transformation rules, the change is replicated in both sides, the mobile and cloud ones. In this sense, *A* saves effort in the modeling process and consequently, in the overall process of developing MobileApps-FC.

On the other side, the main difference of MoWebA regarding other MDD approaches is the inclusion of three design aspects which could help to address the portability challenge. Since, in this work, where we have focused on the network communication aspect, we consider one of such aspects, which is the ASM. The relevance of the ASM is to improve the portability of the PIM regarding the different architectures. Even though, WebRatio considers as well the ASM, MoWebA considers, additionally, model to model (M2M) transformation rules, which enable the semi-automatic transition from the PIM to ASM. In fact, there exist such rules for other architectures.[19] Regarding the network communication aspect, we do not have yet such M2M rules. Nevertheless, once available (such rules) we consider that we could accelerate the modeling of MoWebA Mobile, and, consequently, save more time in the modeling process. Such saving of time implies as well a saving of effort in the process of developing MobileApps-FC.

RQ3: On the mobile side, both, *A* and *B*, generate code for iOS and Android, the most popular platforms. While *A* generates native mobile code, *B* generates code for hybrid applications. On the cloud side, *A* generates code for two providers (Openshift and Amazon). *B* generates a Java application just for one provider, that is its own cloud service platform.[20] With *B*, thanks to its development tools, the application can be automatically deployed in the cloud. In case of A, it generates an implementation to run in two different clouds service

[19] M2M transformation rules for RIA, `https://goo.gl/8Lsy6n`
[20] WebRatio Cloud Plans, link: `https://goo.gl/ByQgMp`

provider's (Openshift and Amazon). Even though, we highlight that the code generated using A, is based on Docker,[21] which is a method developed by the open source community. Precisely, one of the main goals of Docker is to ease the cloud application portability. Therefore, the code generated could be ported, more easily, to other providers which include Docker in their services.

A brief summary of the comparative study is shown in Table 2.

Table 2: Brief summary of the comparative study

Aspects	**Approaches**			**Comments**
	MoWebA M. (*A*)	**WebRatio M. (*B*)**	**Traditional**	
Dev. time	1 h 20 min	55 min	160 hs	*A*'s time can be improved using a specific tool
Modeling diff.	Unified model	One model for mobile, another one for cloud	—	In *A*, the network communication is designed in only one model which saves design effort
	It considers M2M semi-automatic rules from PIM to ASM	It does not consider such M2M rules	—	The semi-automatic M2M rules could help to save effort in the modeling process
Number of Gen. Platf.	2 mobile platforms, 2 cloud platforms	2 mobile platforms, 1 cloud platform	—	The cloud code generated by *A* is based on Docker, which eases the code portability

5 Conclusions and Future Works

In summary, we have proposed the adoption of MoWebA for the modeling and generation of MobileApps-FC, based on three design aspects which could improve the portability of such applications. Furthermore, we have focused on extending MoWebA to model and generate the network communication aspect, through an ASM. A as preliminary validation of our work, we have presented a comparative study of MoWebA Mobile versus the traditional approach and the consolidated MDD platform WebRatio Mobile. As expected, we have found a considerable saving of effort in the development of the network communication functions of the MobileApps-FC, using the MDD approaches. Among the MDD approaches, the development time using WebRatio is slightly better than MoWebA Mobile's time. Nevertheless, considering that MoWebA Mobile is implemented through generic tools, we believe that having a MoWebA Mobile's specific and more friendly modeling and generation tool its development time will be improved, which will lead to an additional saving of effort and, consequently, to a better alleviation of the portability challenge effects.

Thus, as future works, we consider: 1) to build a specific tool for making simpler and faster the modeling; 2) to build the transformation rules for the automatic transition from the PIM to the ASM, which includes the steps iv) and v) of the MoWebA extension definition [16]; and, 3) to improve the validation of the proposal with additional rigorous experiments.

[21] Docker, link: `https://www.docker.com/what-docker`

References

1. March, V., Gu, Y., Leonardi, E., Goh, G., Kirchberg, M., Lee, B.S.: Mcloud: Towards a new paradigm of rich mobile applications. Procedia CS **5** (2011) 618–624
2. Sahu, D., Sharma, S., Dubey, V., Tripathi, A.: Cloud computing in mobile applications. International Journal of Scientific and Research Publications **2**(8) (2012) 1–9
3. Abolfazli, S., Sanaei, Z., Gani, A., Xia, F., Yang, L.T.: Rich mobile applications: genesis, taxonomy, and open issues. Journal of Network and Computer Applications **40** (2014) 345–362
4. Brambilla, M., Mauri, A., Umuhoza, E.: Extending the interaction flow modeling language (IFML) for model driven development of mobile applications front end. In: Mobile Web Information Systems - 11th International Conference, MobiWIS 2014, Barcelona, Spain, August 27-29, 2014. Proceedings, Springer International Publishing (2014) 176–191
5. Sanaei, Z., Abolfazli, S., Gani, A., Khokhar, R.H.: Tripod of requirements in horizontal heterogeneous mobile cloud computing. CoRR **abs/1205.3247** (2012)
6. Gupta, P., Gupta, S.: Mobile cloud computing: The future of cloud. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering **1**(3) (2012) 134–145
7. da Silva, E.A.N., Fortes, R.P., Lucrédio, D.: A model-driven approach for promoting cloud paas portability. In: CASCON. (2013) 92–105
8. Pons, C., Giandini, R., Pérez, G.: Desarrollo de Software Dirigido por Modelos. Editorial de la Universidad Nacional de La Plata (EDULP)/McGraw-Hill Educación (2010)
9. Sanchiz, E., González, M., Aquino, N., Cernuzzi, L.: Development of mobile applications with functions in the cloud through the model driven approach: A systematic mapping study. CLEI electronic journal **20**(3) (December 2017)
10. Chondamrongkul, N., Chondamrongkul, N.: Model-driven framework to support evolution of mobile applications in multi-cloud environments. International Journal of Pervasive Computing and Communications **12**(3) (2016) 332–351
11. Heitkötter, H., Majchrzak, T.A., Kuchen, H.: Cross-platform model-driven development of mobile applications with md 2. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing, ACM (2013) 526–533
12. Ranabahu, A.H., Maximilien, E.M., Sheth, A.P., Thirunarayan, K.: A domain specific language for enterprise grade cloud-mobile hybrid applications. In: Proceedings of the compilation of the co-located workshops on DSM'11, TMC'11, AGERE! 2011, AOOPES'11, NEAT'11, & VMIL'11, ACM (2011) 77–84
13. Steiner, D., Turlea, C., Culea, C., Selinger, S.: Model-driven development of cloud-connected mobile applications using dsls with xtext. In: EUROCAST (2). Volume 8112 of Lecture Notes in Computer Science., Springer (2013) 409–416
14. Ruokonen, A., Pajunen, L., Systä, T.: On model-driven development of mobile business processes. In: SERA, IEEE Computer Society (2008) 59–66
15. Richardson, L., Amundsen, M., Ruby, S.: RESTful Web APIs. "O'Reilly Media, Inc." (2013)
16. González, M., Cernuzzi, L., Aquino, N., Pastor, O.: Developing web applications for different architectures: The moweba approach. In: Tenth IEEE International Conference on Research Challenges in Information Science, RCIS 2016, Grenoble, France, June 1-3, 2016. (2016) 1–11