Kalpa Publications in Computing

Volume 15, 2023, Pages 15–21

Proceedings of The 2022 International Conference on Digital Transformation and Innovation Technology

Context-Aware Mobile Apps: Identifying Tools for Creation Approaches

Estevan Ricardo Gómez-Torres[1,2*]
[1] Universidad Nacional de la Plata , La Plata, Argentina
[2] University of Army Forces, Quito, Ecuador.
ergomez@espe.edu.ec

Abstract

When developing context-sensitive mobile applications there are several considerations that must be considered such as: being able to express the relevant concepts and subsequently generate the required applications. In this area it is worth mentioning that the technology used must allow expressiveness related to factors such as category, relevance, combination, type of configuration and type of execution and generate a context-sensitive mobile application, framed within the characteristics that are currently in demand and using the most appropriate tools for it. This paper analyzes the particularities of developing a context-sensitive mobile application through a Metamodel creation process, then the generation of a Visual D.S.L with Obeo Sirius, its treatment with Xtext, the generation of a mobile application with Acceleo and finally the definition of the graphical user interface GUI with the help of Ionic and Capacitor, which will allow to exploit the functionalities of the mobile. That is to say, the process to establish the appropriate tools for the creation of context-sensitive mobile applications is shown.

Key Words: MDD, Context-aware mobile applications, Ionic, Xtext, Capacitor, mobile development tools

1 Introduction

A modeling language is also known as domain specific language or DSL (Domain Specific Language), it is used to determine and characterize a model; DSLs can be graphic or textual (García, 2012). One of the advantages of creating a DSL is that when executing it, several instances of the original metamodel can be created, which, particularly in the case of Obeo Sirius (Sirius, 2023), is done easily, since an execution environment or Run time is created, allowing the specification of graphical editors (Visual DSL), and in which it is possible for the user to work on the design of several instances graphically, based on the concepts of the original metamodel. When working with the Obeo Sirius instantiated models, it embeds the plugins that we have previously generated, in such a way that what

is created is an example of our model, the user will be able to use the tool palette and with it, all the elements defined for this purpose.

From a technical point of view, graphical modeling consists of representing a model through diagrams that visually represent the elements of the model and their relationships (DSL). Diagrams are typically made up of boxes and borders: a box representing a model element and a border representing the relationship between two model elements. The boxes can use different shapes and colors, possibly icons, depending on their nature and properties. They can also contain lists or other boxes to represent subitems. Xtext (Xtext, 2023) on the other hand, is an open-source framework for developing IDEs for textual programming languages. As in traditional parser generators, the language is defined in a grammar, from which the parser code is generated. But in Xtext, the grammar also describes how the parser should instantiate a model of the text (Xtext, 2023).

It can be identified that the models can be edited both visually and textually, plus it can be noted that Xtext provides an advantage to create an IDE for your language. Additionally, you can customize the generic defaults, since Xtext relies on dependency injection (DI) to connect your language infrastructure. By changing the DI configuration, you can literally replace each component with your custom implementation. A diagram can be built on the data saved in text files using Obeo Sirius and Xtext. In that situation, you get a graphical representation of the same information and can edit it using the diagram or text editor, as shown at Fig 1.

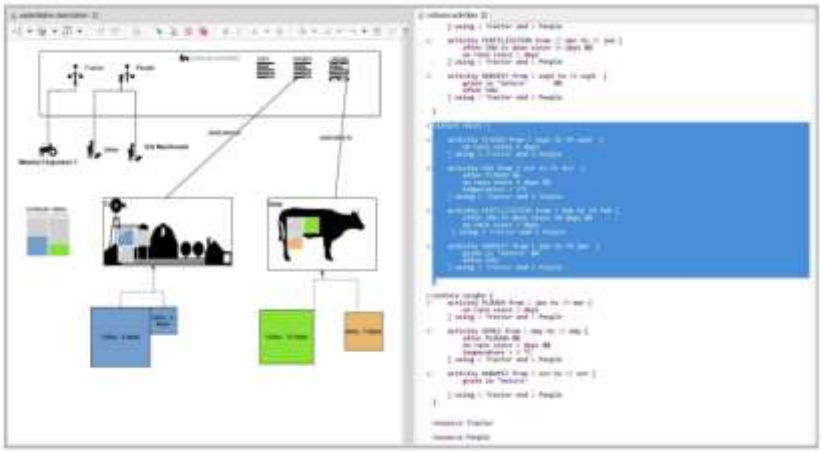

Fig : 1 Edition of Xtext and Obeo Sirius simultaneously

To choose the most appropriate tool, the following considerations were initially made: 1) the tool should be easy to use, to allow, from the specifications of context-sensitive mobile applications and their characteristics, to establish the specific language domain and from this create a meta-model that represents it. 2) With this base, create a DSL in a textual or visual way to facilitate the handling of the characteristics of the future application (context, characteristics, relevance, sensors, etc). 3) Be compatible with the generation of code and the application of MDD in order to be able to create a basic mobile application with the characteristics already defined.

2 Development

With the above described we proceed to detail the technology analysis process. Initially, the possibility of working with Visual Studio (Microsoft, 2023), and creating a DSL was analyzed, since it is generally graphic and designed for a particular purpose, it must be considered that to define a DSL,

a Visual Studio solution must be created to from templates (Microsoft, 2023), which in one way or another restricts development flexibility. On the other hand, the possibility of working with Visual Studio Xamarin (Xamarin, 2023) and Apache Cordova (Cordova, 2023) was analyzed, however, one of the main disadvantages is that elements provided by the platform and some .Net open source resources must be used, that is, there is limited access to open source libraries (Inmmediatum, 2023) .

Subsequently, work was done analyzing the possibility of using Jboss (Jboss, 2023) , which basically presented itself as an Eclipse project, with development capabilities for hybrid application development; that functionality is later redefined and Jboss becomes an open source Java EE application server implemented in pure Java [Mora, 2014]. Although [RedHat, 2023] states that Jboss EAP includes support for Angular Js , jQuery Mobile and Google Web Toolkit (GWT), the architecture gets complicated when trying to install and configure Acceleo for mobile application generation and consequently the application of MDD additionally incorporates a paid subscription model, which further complicates access to development tools. Jboss currently changed its name to WildFly (Wildfly, 2023).

It is also worth mentioning that an analysis of some environments was presented in (Gómez-Torres, Challiol, & Gordillo, 2019), thus selecting Eclipse combined with Obeo Sirius and JBoss as a good choice; further progress was made in a possible solution of the tool in (Gómez-Torres E, 2020), however, as of 2019, RedHat [RedHat] announces the availability of a new version of its JBoss Enterprise Application service platform Platform (EAP). At that certain moment, JBoos changes its way of working to Red Hat Jboss EAP, and these changes are basically aimed at giving it better compatibility with working as a fully managed or self-managing application service in Microsoft Azure and in our case, it generates us to rethink the strategy.

When carrying out a new analysis of environments, it was possible to observe the existence of Eclipse (Eclipse, 2022), which initially worked with a plugin called Sirius, with Eclipse Modeling Framework EMF and Ecore, which is a metamodel, which presents many advantages to when designing our proposal. Then that plugin, which was initially Sirius [Sirius], becomes an environment based on Eclipse and which is called Obeo Sirius, that is, for its operation and when installing and finally other functionalities are developed, establishing itself as the definitive name Obeo Designer.

Considering aspects related to development, we can say that, although JBoss also has templates, they are more limited with respect to the integration environment that it has with Eclipse. As for Visual Studio it is complex to customize the predefined templates. In this case the templates allow variability in the range of applications that can be generated. Then arises the possibility of working in an integrated environment such as Obeo Sirius.

According (Gómez-Torres E, 2020), Obeo Sirius and Acceleo (Acceleo, 2023) , which is an open source code generator from the Eclipse Foundation that allows a model-based approach (MDD) to be used to create applications.

It is an implementation of the standard "MOFM2T" [MOFM2T], from the object management group, to perform the transformation from model to text. It is the best option to support variability in the generated applications, achieving greater expressiveness and speeds up instantiation using visual DSLs. The possibility of customizing or extending the environment with plugins makes it possible to support different types of sensors, and to be able to support a wider range of applications in the future. Taking this into account, our tool to support the proposed approach will be built using Obeo Sirius. It should also be considered that Eclipse, and Obeo Sirius, base their working principles on: Eclipse Modeling Framework (EMF) is a modeling framework based on Eclipse and a code generation function to create tools and other applications based on a model of structured data.

Ecore is the central (meta)model at the heart of EMF. It allows expressing other models taking advantage of their constructions. Ecore is also its own meta-model (ie: Ecore defines itself in terms of itself) (Steinberg, 2008).

According to Ed Merks, (Merks, 2023) , EMF project leader, "Ecore is the de facto reference implementation of OMG's EMOF" (Essential Meta-Object Facility). Still, according to Merks, EMOF

was defined by OMG as a simplified version of the more complete 'C'MOF based on the experience of successful simplification of the original Ecore implementation. Using Ecore as a foundational metamodel allows a modeler to take advantage of the entire EMF ecosystem and tools, to the extent that it is reasonably easy to map application-level models to Ecore. This is not to say that it is a good practice for applications to directly leverage Ecore as their meta-model; rather they could consider defining their own meta-models based on Ecore (Steinberg, 2008).

For the development of the mobile application in Android, it must be considered that, as it is a visual DSL, this process generates a code, when using Obeo Sirius intrinsically EMF will be used; which has a complete ecosystem of Model Driven technologies such as: XText, which is used for the creation of textual editors. Obeo Sirius will be used for the creation of graphical editors, while Acceleo and Xtend, used for code generation, (Xtext, 2023).

Another additional aspect that must be considered is that the generated solution will necessarily have to be tested and edited, for which there are several options:

1. We can work directly with Apache Cordova since it is an open-source mobile development framework. It allows you to use standard web technologies: HTML5, CSS3 and JavaScript for cross-platform development. A plugin is a package of injected code that allows the Cordova web view within which the application is rendered to communicate with the native platform on which it runs. Plugins provide access to device and platform functionality not normally available to web-based applications (Cordova, 2023). Applications run within containers intended for each platform and rely on standards-compliant API bindings to access the capabilities of each device, such as sensors, data, network status, etc. The Apache Cordova architecture is shown in Figure 2.

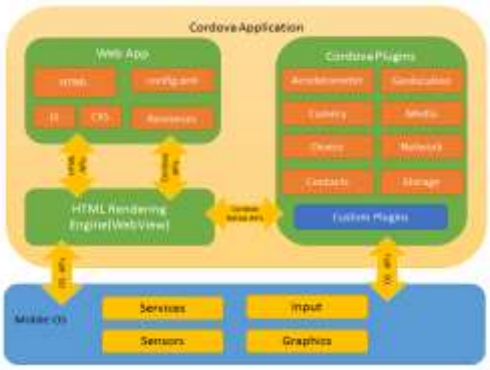

Fig : 2 Architecture of Apache Cordova **(Cordova, 2023)**

2. Work with Ionic Framework which is an open-source frontend SDK for the development of hybrid applications based on web technologies (HTML, CSS, AND JS). That is, a framework that allows us to develop native applications for iOS, Android, and web, from a single code base. It has great compatibility thanks to the implementation of Cordova and its improved version Capacitor and Ionic Native, which make it possible to work with hybrid components. Integrates with major frontend frameworks such as Angular, React, and Vue, although you can also use vanilla JavaScript.

   In our case, we would use Ionic to generate the user interfaces and guarantee that they are hybrid applications, that is, an HTML application that runs on top of a native application. Ionic also offers us a battery of control interface components that has a mobile aspect and is integrated with Capacitor. Another aspect is that it has an adaptive style, that is, it will change

its appearance depending on the platform it runs on, that is, Capacitor will compile the application.

On the other hand, the potential of Apache Capacitor can be used, to enhance the generated mobile application, as well as verify its perfect operation, it must be considered that Ionic with Apache Capacitor uses a WebView, which is the internal browser that the applications have natively. , on top of which will run the HTML5, CSS, and JavaScript application that will be created, as shown in Figure 3.

Fig : 3 Architecture of Ionic and Capacitor

2.1 Context-Aware Mobile Application Development Process:

The development process of context-sensitive mobile applications can be summarized in Figure 4, where from the metamodel, the Visual DSL is generated, interacting with Xtext to edit the generated code, the Acceleo plugin is used to generate the application. android mobile: Considering this generated code and to refine its operation, as well as to improve its user interface (GUI), Ionic and Capacitor [Ionic] will be used, while it will also serve to evaluate the use of plugins or complements, such as status, of the battery, camera, GPS and accelerometer, vibration, among others.

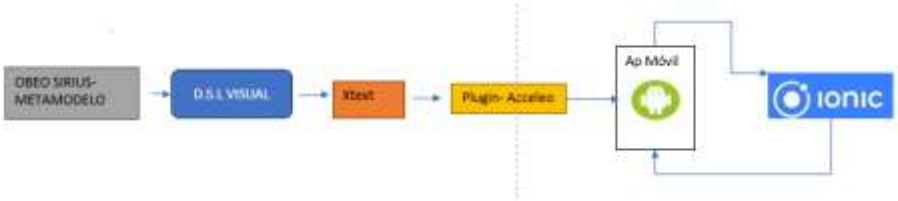

Fig : 4 Process of Mobile Application Development

Additionally, it should be mentioned that: all the main functions of the Cordova API are implemented as plugins or plugins, and there are many others available that enable functions such as barcode readers, NFC communication or custom calendar interfaces (Cordova, 2023).

It is important to note that Capacitor provides a powerful and easy-to-use interface to access the native SDKs and native APIs on each platform. As an alternative to PhoneGap and Cordova, Capacitor offers the same cross-platform benefits, but with a more modern approach to application development, leveraging the latest web APIs and native platform capabilities. Capacitor is the only native runtime element that provides first-class support for Web Apps and Progressive Web Apps.

In Capacitor, traditional native mobile developers can use their choice of programming languages (Swift/Objective-C on iOS, Java/Kotlin on Android) to create UI experiences or business logic, and then expose them to the web layer via of Capacitor JavaScript with native APIs (Lynch, 2018). Additionally, there is a complete agreement between Ionic and Angular so that mobile applications based on Angular are generated and with the use of React, which provides perfect compatibility with Ionic. Out of this comes Ionic React, which is the native React version of the Ionic Framework, the free, open-source SDK that powers millions of mission-critical apps around the world (Ionic, 2023).

3 Results

The possibility of working with a DSL greatly facilitates the work in capturing the development needs of context-aware mobile applications. Since according to the advantages that (Microsoft, 2023) tells us are that: a) It contains constructions that exactly fit the problem space, b) It allows non-developers and people who do not know the domain to understand the general design , c) It facilitates the creation of a prototype of the final application.

On the other hand, we use Acceleo to generate the Android application, taking advantage of the flexibility of code generation through templates (Acceleo, 2023).

We also use Ionic Framework is an open-source front end SDK to develop hybrid applications based on web technologies (HTML, CSS and JS), additionally Its compatibility and, thanks to the implementation of Cordova and Ionic Native, make it possible to work with hybrid components (Ionic, 2023).

As a result, it is intended that the generated applications, being Hybrid, have a high level of compatibility with Android devices when running context-sensitive mobile applications, taking advantage of the functionality of Capacitor to exploit features such as accelerometer, camera, GPS, among others.

4 Conclusions

The compatibility between Eclipse and Obeo Sirius is very important to be able to work in all phases of development of context-aware mobile applications, together with the openness that these tools present through the installation of plugins, they allow adding great functionality through Acceleo and Xtext.

The process allows us, on the one hand, to ensure that a DSL is generated from the metamodel, to use all the plugins and a procedure that uses MDD to generate the context-sensitive mobile application and, on the other, to have the possibility of enriching the result with the use of Ionic and the use of current tools for the generation of hybrid applications such as Capacitor, which will allow us to use a series of plugins or complements, which gives current and versatility to the generated applications..

The ideal would be to be able to generate a Framework that automates the entire process, and that will remain as future work of our application.

References

Acceleo. (2023, 0508). *Acceleo*. Retrieved from https://www.eclipse.org/acceleo/

Capacitor. (2023, 04 20). *Capacitor*. Retrieved from Capacitor: https://capacitorjs.com

Carlisle, D. (2010, April). *graphicx: Enhanced support for graphics.* Retrieved from http://www.ctan.org/tex-archive/ help/Catalogue/entries/graphicx.html

Cordova. (2023, 05 01). *Cordova*. Retrieved from Cordova: https://cordova.apache.org/docs/en/11.x/guide/overview/

Eclipse. (2022, 09 26). *Eclipse Modeling Project*. Retrieved from Eclipse Modeling Project: https://www.eclipse.org/modeling/emf/

García, e. a. (2012). Desarrollo de Software Dirigido por Modelos. In J. García, F. García, V. Palenchano, A. Vallecillo, & J. &. Vara, *Desarrollo de Software Dirigido por Modelos.*

Gómez-Torres E, e. A. (2020). Challenges in Context-Aware Mobile Applications Building Approaches. *Inciscos* (pp. 304-310). Quito: IEEE.

Gómez-Torres, E., Challiol, C., & Gordillo, S. (2019). Variability Features in Building Approaches for Context Aware Mobile Applications. *Inciscos.* Quito: IEEE.

Inmmediatum. (2023, 04 20). *Inmmediatum*. Retrieved from Inmmediatum: https://inmediatum.com/blog/ingenieria/ventajas-y-desventajas-de-apps-desarrolladas-en-xamarin/

Ionic. (2023, 04 20). *Ionic*. Retrieved from http://ionicframework.co

Jboss. (2023, 04 20). *Jboss*. Retrieved from https://tools.jboss.org/downloads/devstudio/

Lynch, M. (2018). *Lynch*. Retrieved from Capacitor vs Cordova-Cuales son las Diferencias entre ellos: https://bit.ly/3KS9FFm

Merks, E. (2023, 04 20). *Eclipse*. Retrieved from https://accounts.eclipse.org/users/emerks

Microsoft. (2023, 04 20). *Modeling how to define a Domain Specofoc Languaje*. Retrieved from https://learn.microsoft.com/pt-pt/visualstudio/modeling/how-to-define-a-domain-specific-language?view=vs-2019

Microsoft. (2023, 05 08). *Visual Studio*. Retrieved from Visual Studio: https://visualstudio.microsoft.com/es/

Mora, M. (2023, 04 20). *Adictos al Trabajo*. Retrieved from Jboss: https://www.adictosaltrabajo.com/2014/09/19/curso-jboss-de-red-hat/

RedHat. (2023, 04 20). *RedHat*. Retrieved from RedHat: https://www.redhat.com/es/technologies/jboss-middleware/application-platform/features

Sirius. (2023, 02 16). *Sirius*. Retrieved 02 23, 2023, from https://www.eclipse.org/sirius/overview.html

Steinberg, D. B. (2008). *Eclipse Modeling Framework.* Pearson.

Wildfly. (2023, 04 20). *Wildfly*. Retrieved from https://www.wildfly.org

Xamarin. (2023, 05 08). *Xamarin*. Retrieved from https://dotnet.microsoft.com/en-us/apps/xamarin

Xtext. (2023, 05 01). *Xtext*. Retrieved from https://www.obeodesigner.com/resource/white-paper/WhitePaper_XtextSirius_EN.pdf