Towards a new perspective of building tools for context-aware mobile applications

Estevan Gomez-Torres 1,2 [0000-0002-1171-7256], Cecilia Challiol 3,4 [0000-0001-5140-0264] and Silvia E. Gordillo 3,5 [0000-0001-5724-5058]

1 Universidad UTE, Facultad de Ingeniería, Carrera de Ingeniería en Informática. Quito, Ecuador
2 Facultad de Informática, UNLP. La Plata, Buenos Aires, Argentina
3 LIFIA, Facultad de Informática, UNLP. La Plata, Buenos Aires, Argentina
4 CONICET. Argentina
5 CICPBA. Buenos Aires, Argentina
estevan.gomezt@info.unlp.edu.ar,
{ceciliac,gordillo}@lifia.info.unlp.edu.ar

Abstract. Technological growth has been exponential in relation to mobile devices (such as embedded sensors) that have allowed developing context-aware mobile applications for the market. This growth generates a new challenge about how to develop this kind of application to adapt them to the current user's demand. Nowadays, there are several approaches that could be used to create context-aware mobile applications, but these approaches are designed without considering to provide variability in the kind of generated applications. The aim of this paper is to propose a building tool that has been designed from scratch taking into account variability points in order to generate a wide variety of context-aware mobile applications. The first version of our tool is presented in this paper which is based on UML tools like Eclipse, Sirius Obeo, and JBoss. Finally, a discussion of different aspects is detailed to help designers to have guidelines to select the appropriate development environment for the design of building tools for context-aware mobile applications.

Keywords: Building Tools, Context-aware Mobile Applications, Variability, UML Tools, Eclipse.

1 Introduction

Technological growth has been exponential in relation to mobile devices (such as embedded sensors) that have allowed developing context-aware mobile applications for the market that is described in [1]. Moreover, a new issue has emerged as is presented in [2], which consists of how to provide useful context-aware mobile applications to the users, something that is critical in today's market. This growth generates a new challenge about how to develop this kind of application to adapt it according to the current user's demand.

Context-aware mobile applications are used in different domains, for example, traffic assistance, medical monitoring, tourism, etc. [1,2]; but they could also be used in other potential scenarios, as mention in [3] for future mobile applications for education. Thus, building tools for this kind of applications should be designed to support new requirements. In [4], the authors identify two kinds of new requirements, those that maintain the same goal of the application (associated with adaptation) and those that change the main goal of the application (related to evolution). According to [4], integrating runtime adaptation and evolution is crucial for the sustainability of software systems; and the building tools are not the exception.

Context-aware mobile applications could be created using one of the existing building tools. However, there is not yet an agreement about how to provide a unified solution as is mentioned in [5]. Some of these building approaches are for non-expert users, one of these is detailed in [6]. Besides, others approaches are for experts because they require some technical knowledge, for example, to create models as the tool presented in [7]. Nevertheless, these tools are designed to provide, for example, only some location mechanism such as GPS. Thus, these tools do not support variability in the kind of generated applications. This issue is one of the motivations of this paper.

Potential variability points should be considered from the initial design of the building approaches for context-aware mobile applications. A possible way of handling some variability concept was exemplified in [8] for conceptual building approaches; however, it is not detailed from the building tool's perspective, so this is the motivation for the research presented in this paper.

Therefore, the aim of this paper is to propose a building tool to support variability in while generating applications that have been designed using the taxonomy of variability concepts proposed in [8]. This tool is based on UML standards in order to have more interoperability with other existing tools.

In this paper, the first version of our tool is presented which is implemented following the UML specification described in [9]; and it integrates Eclipse, Sirius Obeo, and JBoss.

We hope that this paper helps designers to know a possible way of handling variability in building tools for context-aware mobile applications. To do so, a discussion in relation to this topic is presented, and details are presented on how to consider the variability in these tools to generate a wide variety of applications. This discussion could be used as guidelines to select the appropriate development environment for the design of these building tools.

The paper is structured as follows. Section 2 presents the methodology related to the design of our building tool; according to that, a survey of the taxonomy of variability concepts and the UML tools used are described. Section 3 presents the first version of our building tool for context-aware mobile applications. Section 4 presents a discussion and the outlook of the paper. Finally, conclusions and future works are detailed in Section 5.

2 Methodology

Our building tool for context-aware mobile applications has been designed using the taxonomy of variability concepts described in [8]. This section describes a survey of the proposed taxonomy [8], in order to help the reader to understand better how our tool handles variability. In addition, our building tool is based on the UML standard to have more interoperability with other existing tools. Therefore, in this section, UML tools are also detailed to understand the interdependence with the tool.

2.1 Variability concepts used to design our building tool

We have been working in the area of mobile applications. For example, in [10] a model approach has been proposed for handling variability in context-aware mobile applications. This approach is based on the concept of separation of concern, so it is modelled the concepts of aware-objects, context-feature and sensors in a decoupling way.

We have been creating some authoring tools for this kind of application. For example, in [11] is presented an authoring tool to create in-situ location-based experiences; this tool has been extended in [12] in order to provide support for location-based educational activities. Besides, in [13] is presented an authoring tool to in-situ co-design location-based applications inside indoor spaces.

Based on our experience, an initial taxonomy version of variability concepts for building approaches for context-aware mobile applications has been presented in [14]. The taxonomy presented in [14] defines the following concepts: relevance, combination, categorization, precision and accuracy's margins, configuration type, and execution type. These concepts have been selected in order to help designers to identify the potential variability points to obtain more flexible building approaches. In [14], it is described briefly each concept of this taxonomy without going into details on how it should be handled in building approaches.

In [8] a possible way of handling each variability concept of [14] was exemplified for conceptual building approaches. In [8], each concept of the taxonomy is described using a pattern-based format, detailing how and what could be considered in the design phase of this kind of approach. However, in [8] is not detailed from the building tool's perspective, so this is the motivation for the research presented in this paper.

Note that, when a building tool is designed from scratch considering the variability concepts presented in [8] and [14]; this eases the generation of a wide variety of context-aware mobile applications. Our previous works [8,14] analyses variability concepts only describing them without the details of a concrete implementation.

The main contributions of this paper, in contrast with our previous works [8,14], is to provide a building tool which handler variability concept to generate a wide variety of context-aware mobile applications.

A survey of the variability concepts presented in [8,14] is presented below. This survey helps the reader to understand better the variability concepts handled by our building tool (which is presented in Section 3).

- *Relevance:* each context feature has its own relevance in each application; for some applications, some of them are critical and others optional. A building tool should provide a scale of relevance in order to choose, for example, the most important context-features to generate reduced versions of applications. In addition, it is required to handle the heuristics of what actions to take based on selected context features. This last functionality can be complex to implement from a building tool.
- *Combination:* context-aware mobile applications could require the combination of various context-features to trigger some services or actions. The complexity of implementing this combination in a building tool is to define clear rules of how to react to each value that these context-features could take.
- *Precision and Accuracy's Margins:* each physical sensor used in context-aware mobile applications has its precision and accuracy's margin. When a tool generates an application, it should define how to react to each sensed value from these physical sensors; but also how to consider the precision and accuracy of them in order to trigger the correct action.
- *Categorization:* there are different aware-objects in context-aware mobile applications, for example, the user or the environment. These aware-objects should have different treatment in the generated applications. Thus, the building tool should define the heuristics to handle the behavior based on each category of aware object.
- *Configuration Type*: the physical sensors used in context-aware mobile applications have a default, passive or active configuration type, according to [8]. The building tool should handle each configuration type. The active configuration type is more complex to handle because it requires automatic learning.
- *Execution Type:* the physical sensors used in context-aware mobile applications have a passive or active execution type, according to [1]. The building tool should handle each execution type. The passive execution type is more complex to handle because it requires details of how the application will interact with the user; as well as, how it will react to each possible interaction.

Note that, the variability concepts described above are related to different aspects of the context-aware mobile application. Categorization is associated with aware-objects, Relevance and Combination are related to specific context-features of aware-objects; and the rest of the variability concepts (Precision and Accuracy's Margins, Configuration Type and Execution Type) are associated with physical sensors. A possible way to represent the variability concepts is presented in [8] which is based on the concepts presented in [10].

Using the representation presented in [8], in Fig. 1 is detailed the variability concepts used to design our building tool. The figure shows that some variability concepts are related to the concrete sensor, but others to the aware-objects or context-features as it has been mentioned.

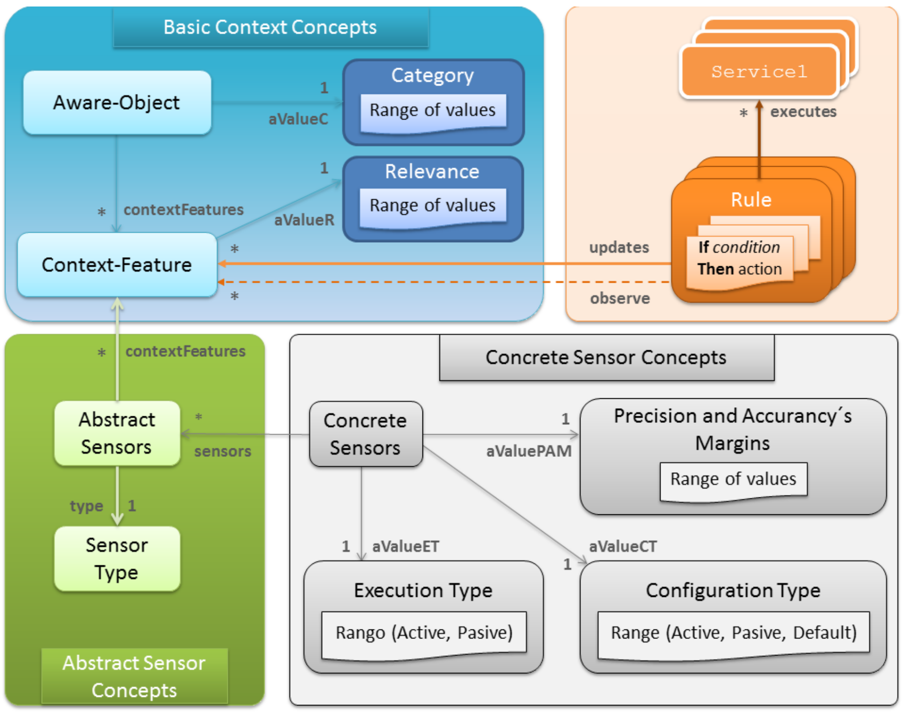


Fig. 1. Variability Concepts used to design our building tool.

2.2 Development environment used by our tool

In this section, the UML tools used by our tool are detailed in order to help the readers to understand why they have been chosen. Note that, there are lots of tools that could be used together to create context-aware mobile applications, but they have not been designed for this purpose. However, we have selected only UML tools to have more interoperability with other existing tools because they are based on the UML standard specification [9]. The selection of these UML tools has been focused on obtaining more flexibility in order to implement each variability concept described in Section 2.1.

There are two key points related to our selection; first, the tools should allow the possibility of representing information associated with each variability concept, and then they should allow the generation of applications using this information. We have considered these key points to choose which UML tools could be used together to design our tool.

The UML tools used to design our tool have been selected considering some research that compares different features of existing tools. For example, in [15] is reviewed the main features, potential advantages and current limitations of the main tools that exist for the development of graphical editors for visual Domain-Specific Languages (DSL). In [16], the authors analyse mobile application tools from the performance features.

These analysed tools are native of Google Android and Apple iOS, and others are cross-platform (such as Appcelerator Titanium, Microsoft Xamarin and Apache Cordova).

In addition, in [17] we had been analysed different UML development environment for the design of building tools for context-aware mobile applications; according to the variability concepts described in [8]. For each UML development environment, we had analysed, for example, the complexity of representing different concrete data, how could be generated mobile applications and the complexity of this, and what sensors could be used and how these could be integrated into the generated applications. In [17] had only analysed different UML development environment; but it had not presented a concrete solution, as it is the proposal in this paper.

Considering that mentioned above, we have chosen to use Eclipse [18] as a development environment combined with Sirius Obeo [19] in order to represent each concept using visual DSL's [20], and JBoss [21] to generate Android mobile applications. Thus, the meta-models are defined with Eclipse, and then these meta-models are used by Sirius Obeo to specify the visual DSLs. This allows specifying values related to each variability concept in a rapid and easy way.

In addition, Eclipse has a lot of plugins to access the device's sensors (such as GPS and Compass) that are compatible with JBoss; to generate mobile applications with sensor's functionalities. Moreover, JBoss allows creating new plugins to specify additional functionalities that are embedded inside the generated mobile applications.

Note that, JBoss works in conjunction with Hybrid Mobile tools [22] and extensive support Apache Cordova multiplatform [23] for the generation of hybrid applications; this lets using plugins for mobile device control and geolocation, among other features.

3 Results

The first version of our building tool is presented in this section which has been developed using Eclipse as the development environment combined with Sirius Obeo (to represent each concept using visual DSLs) and JBoss to generate Android mobile applications. Fig. 2 shows how these technologies are integrated into our building tool. Note that, an Ecore Meta-Model [24] is created using Sirius Obeo and JBoss used both the Ecore Meta-Model and Sirius Obeo models in order to generate Android applications.

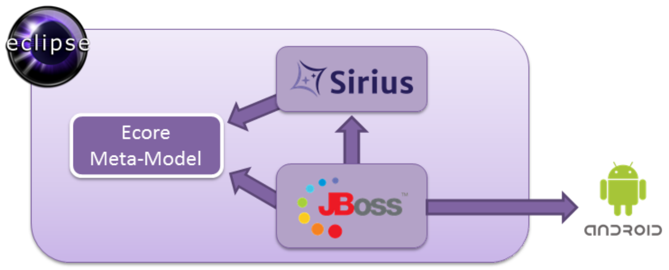


Fig. 2. Our building tool proposal.

The first step creates the Ecore meta-model with each concept described in Fig. 1. This offers the designers the possibility of using this meta-model to facilitate the task of creating concrete mobile applications because they could consume these concepts that have been defined. So, our building tool provides this meta-model as a baseline to generate mobile applications which are shown in Fig. 3.

Using the Ecore meta-model, we have created the visual DSL with Sirius Obeo as it is shown in Fig. 4. The main idea is that the designer gives specific values to the visual concept in a fast and easy way. Currently, we are working to improve the icons related to each concept and look for an easy way that designers could set the value of each concept. Fig. 4 shows an aware-object classified as the user which has a location-feature (that is set by the GPS sensor and has a margin of precision) with a medium relevance. In this case, a rule could be to trigger services according to different location value.

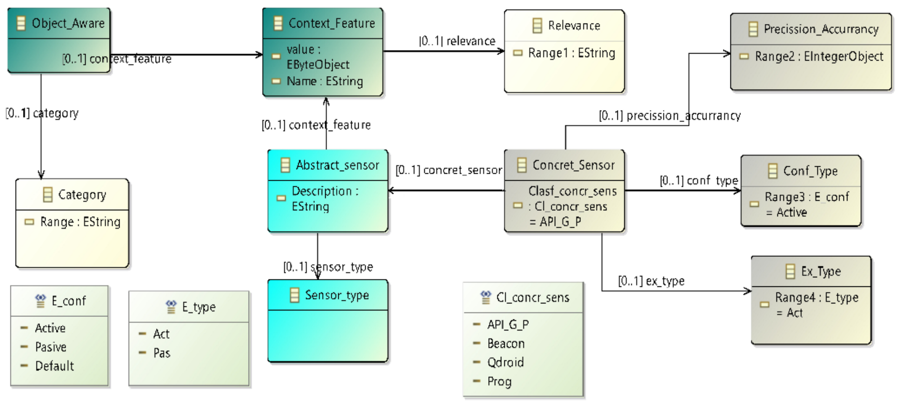


Fig. 3. Ecore meta-model with the basic context-aware and variability concepts.

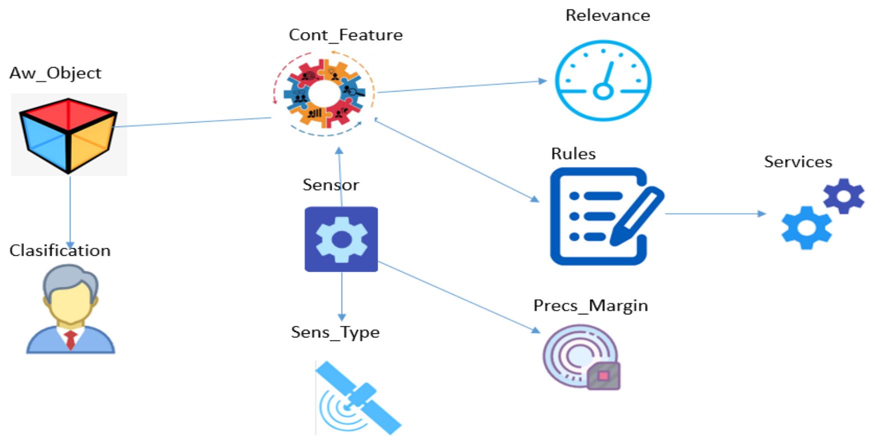
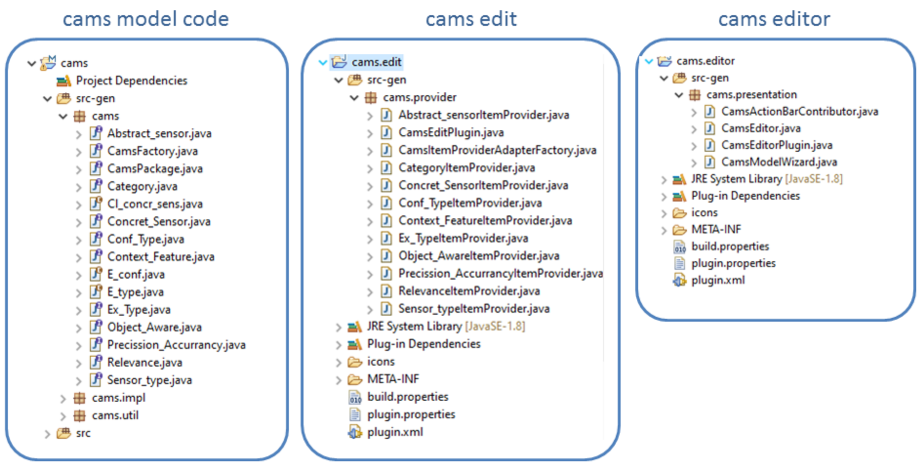


Fig. 4. A first scratch of the visual DSL of our tool.

When the visual DSL has been created three intermediate models are generated: cams model code (classes that represent the meta-model concepts), cams edit (classes associated with edition) and cams editor (classes associated with the visual editor). These intermediate models are shown in Fig. 5.

Currently, we are working on generating Android applications using JBoos integrated with its available sensor plugins. To do that, we have been researching some existing sensor tools, and we are working on an automatic derivation. Besides, we are defining a new plugin with this functionality. We have to emphasize that we want to obtain a version of our tool that allows designers to define and set the values to both basic context-aware and variability concepts in a rapid and easy way, but also to generate functional mobile applications.



```
cams
    Project Dependencies
  src-gen
    cams
      Abstract_sensor.java
      CamsFactory.java
      CamsPackage.java
      Category.java
      Cl_concr_sens.java
      Concret_Sensor.java
      Conf_Type.java
      Context_Feature.java
      E_conf.java
      E_type.java
      Ex_Type.java
      Object_Aware.java
      Precission_Accurrancy.java
      Relevance.java
      Sensor_type.java
    cams.impl
    cams.util
  src
```

```
cams.edit
  src-gen
    cams.provider
      Abstract_sensorItemProvider.java
      CamsEditPlugin.java
      CamsItemProviderAdapterFactory.java
      CategoryItemProvider.java
      Concret_SensorItemProvider.java
      Conf_TypeItemProvider.java
      Context_FeatureItemProvider.java
      Ex_TypeItemProvider.java
      Object_AwareItemProvider.java
      Precission_AccurrancyItemProvider.java
      RelevanceItemProvider.java
      Sensor_typeItemProvider.java
  JRE System Library [JavaSE-1.8]
  Plug-in Dependencies
  icons
  META-INF
  build.properties
  plugin.properties
  plugin.xml
```

```
cams.editor
  src-gen
    cams.presentation
      CamsActionBarContributor.java
      CamsEditor.java
      CamsEditorPlugin.java
      CamsModelWizard.java
  JRE System Library [JavaSE-1.8]
  Plug-in Dependencies
  icons
  META-INF
  build.properties
  plugin.properties
  plugin.xml
```

Fig. 5. Intermediate models associated with visual DSL of our tool.

4 Discussion

This section discusses different aspects of building tools for context-aware mobile applications to help designers to have guidelines in order to handle variability to facilitate them to choose the appropriate development environment.

Note that, this paper is focused on the variability concepts described in Section 2.1, but the variability is a feature of context-aware mobile applications that have been studied from different perspectives. For example, in [25] is analysed different ways of representation of this kind of application, so the authors identified 36 different context-aware models that handle variability at the design level and/or execution time [10]. Considering variability at execution time allows adding context without requiring to be compiled again. Therefore, to handle this variability should be considered in the design of the modelling approach. Context-aware software could handle variability in different ways, as is described in [26]. The authors present in [26] a taxonomy to understand the similarities and differences of each strategy. Therefore, the researches as [10], [25] and

[26] could be used to enrich building tools for context-aware mobile applications with other variability concepts.

There are important key points related to select the appropriate development environment, in order to design building tools for context-aware mobile applications as is described below. First, the development environment should allow the possibility of representing information related at least to each variability concept described in Section 2.1, and it is important to mention that our tool provides visual DSLs to do this.

It should be considered that variability concepts are related to basic context-aware concepts. For example, precision and accuracy's margins, configuration type and execution type concepts are related to concrete physical sensors. The relevance and combination concepts are related to context-features and the classification concepts are associated with aware-objects. So, this requires representing basic context-aware concepts in a way that they are decoupled from each other to allow reuse and extensibility. A possible representation to do that is presented in [10]. When decoupling basic concepts to be considered from a building approach, it allows the combination of different layers and thus, to handle the first level of variability. To do that, our tool provides meta-models with the basic context-aware concepts in order to have the first level of variability; and based on this, the designer could choose the value related to each variability concept using visual DSLs.

Another key point is that the development environment should generate applications using the information of basic context-aware and variability concepts. Sometimes, these development environments only bring support to embed physical sensors inside the generated applications such as JBoss. However, there are other ways to set context-feature values, for example, using applications, services or direct user input as it is described in [27]. Therefore, there are different kinds of sensors, in [27] are listed as physical, virtual, direct user input, etc.; each of these has its own configuration and way of execution. So, the development environment should provide a way to represent these different sensors, for example, by using Eclipse and JBoss we could create plugins in order to handle each of them.

The combination variability concept implies to define the heuristics to specify how it behaves when the value of each combined context-features changes. To do this, the development environment should allow the definition of these heuristics; in the case of Eclipse and JBoss, this could be done by creating a new plugin for each heuristic. Something similar occurs with the active configuration type because it requires an automatic learning program which in Eclipse and JBoss could be specified using a new plugin that implements the logic of this automatic configuration.

These discussion points are important when designers need to choose the appropriate development environment to design a building tool for context-aware mobile applications. We have considered these key points in the design of our tool in order to have the flexibility to implement the variability concepts.

5 Conclusions and Future Works

This paper has proposed a building tool to support variability for the generation of applications designed using the taxonomy of variability concepts proposed in [8]. This tool is based on UML standard tools like Eclipse, Sirius Obeo, and JBoss. The first version of our tool and some main features are presented in this paper.

In addition, a discussion and analysis are presented with recommendations that should be considered to select the appropriate development environment for the design of building tools for context-aware mobile applications.

We continue working to improve our tool to provide automatic generation of mobile applications. Besides, we will improve the tool with more plugins to increase functionalities for other sensors such as virtual sensors (using applications and services) and we will analyse how to define rules related to combined context-features easily.

In the future, we will analyse other variability features to enrich our building tool. For example, [26] to handler more variability in the generated mobile applications.

References

1. Alegre, U., Augusto, J.C., Clark, T.: Engineering context-aware systems and applications: A survey. Journal of Systems and Software 117, 55-83 (2016).
2. Alegre-Ibarra, U., Augusto, J.C., Evans, C.: Perspectives on engineering more usable context-aware systems. Journal of Ambient Intelligence and Humanized Computing 9(5), 1593-1609 (2018).
3. Chatterjee, S., Majumdar, D., Misra, S., Damaševičius, R: Adoption of mobile applications for teaching-learning process in rural girls' schools in India: an empirical study. Education and Information Technologies, pp. 1-20 (2020).
4. Weyns, D., Caporuscio, M., Vogel, B., Kurti, A: Design for sustainability= runtime adaptation ∪ evolution. In: Proceedings of the 2015 European Conference on Software Architecture Workshops, pp. 1-7. ACM, New York (2015).
5. Bauer, C., Dey, A.K.: Considering context in the design of intelligent systems: Current practices and suggestions for improvement. Journal of Systems and Software 112, 26-47 (2016).
6. Bales, S.: Build Android apps without Coding: Get started with Android apps using Thunkable-MIT app Inventor. Independently published. ACM, New York (2018).
7. Hamdani, M., Butt, W.H., Anwar, M.W., Azam, F.: A systematic Literature Review on Interaction Flow Modeling Language (IFML). In: 2nd International Conference on Management Engineering, Software Engineering and Service Sciences. ACM, New York, pp. 134-138 (2018).
8. Gómez-Torres, E., Challiol, C., Gordillo, S.E.: Variability Features in Building Approaches for Context-Aware Mobile Applications. Conference on Information Technologies and Communication of Ecuador, LNCS, vol. 1099, pp. 109-123. Springer, Cham (2016).
9. Rumbaugh, J.: Lenguaje Unificado De Modelado Manual De Referencia, Second Edition. Addison-Wesley Iberoamerica (2007).
10. Fortier, A., Rossi, G., Gordillo, S.E., Challiol, C.: Dealing with variability in context-aware mobile software. Journal of Systems and Software 83(6), 915-936 (2010).
11. Alconada Verzini, F.M., Tonelli, J.I., Challiol, C., Lliteras, A.B., Gordillo, S.E.: Authoring tool for location-aware experiences. In: 2015 Workshop on Narrative & Hypertext, ACM, New York, pp. 21-25 (2015).

12. Zimbello, A.M., Alconada Verzini, F.M., Challiol, C., Lliteras, A.B., Gordillo, S.E.: Authoring tool for location-based learning experiences. In: 2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft*)*, IEEE, pp. 211-212 (2017).
13. Mendiburu, F.I., Challiol, C., Gordillo, S.E.: Herramienta de autor para co-diseñar in-situ Aplicaciones Móviles basadas en Posicionamiento. In: Simposio Argentino de Ingeniería de Software 2019 (ASSE) – JAIIO, pp. 29-42 (2019).
14. Gómez-Torres, E.R., Challiol, C., Gordillo, S.E.: Context-Aware Mobile Applications: Taxonomy of factors for building approaches. In: XXV International Conference on Electronics, Electrical Engineering and Computing, pp. 1-4. IEEE (2018).
15. Granada, D., Vara, J. M., Blanco, F.P., Marcos, E.: Model-based Tool Support for the Development of Visual Editors- A Systematic Mapping Study. In: 12th International Conference on Software Technologies (ICSOFT 2017). Science and Technology Publications, pp. 330-337 (2017).
16. Ebone, A., Tan, Y., Jia, X.: A performance evaluation of cross-platform mobile application development approaches. In: 2018 IEEE/ACM 5th International Conference on Mobile Software Engineering and Systems (MOBILESoft). IEEE, pp. 92-93 (2018).
17. Gomez-Torres, E.R., Challiol, C., Gordillo, S.E.: Challenges in Context-Aware Mobile Applications Building Approaches. In: 2019 International Conference on Information Systems and Computer Science (INCISCOS). IEEE, pp. 304-310 (2019).
18. Eclipse homepage, https://www.eclipse.org, last accessed 2020/04/14.
19. Obeo Sirius homepage, https://www.obeodesigner.com/en/product, last accessed 2020/04/14.
20. Viyović, V., Maksimović, M., and Perisić, B.: Sirius: A rapid development of DSM graphical editor. In: IEEE 18th International Conference on Intelligent Engineering Systems INES 2014, IEEE, pp. 233-238 (2014).
21. JBoss homepage, https://docs.jboss.org, last accessed 2020/04/14.
22. Hybrid Mobile tools. Description of Hybrid Mobile Tools and CordovaSim, https://docs.jboss.org/tools/4.1.x.Final/en/User_Guide/html/chap-Hybrid_Mobile_Tools_and_CordovaSim.html, last accessed 2020/04/14.
23. Apache Cordova multiplatform. Description of Apache Cordova multiplatform, https://cordova.apache.org/plugins, last accessed 2020/04/14.
24. EMF, ECore & Meta Model, https://www.eclipse.org/modeling/emft/search/concepts/subtopic.html, last accessed 2020/06/08.
25. Bauer, C., Novotny, A.: A consolidated view of context for intelligent systems. Journal of Ambient Intelligence and Smart Environments 9(4), 377-393 (2017).
26. Mens, K., Cardozo, N., Duhoux, B.: A context-oriented software architecture. In: 8th International Workshop on Context-Oriented Programming, pp. 7-12. ACM, New York (2016)
27. Rivero-Rodriguez, A., Pileggi, P., Nykänen, O.A.: Mobile context-aware systems: technologies, resources and applications. International Journal of Interactive Mobile Technologies 10(2), 25-32 (2016).