Journal of Computer Science

Original Research Paper

Definition of Object Constraint Language Rules in Models to Support the Development of Android Applications

Jean Carlos Hrycyk, Inali Wisniewski Soares and Luciane Telinski Wiedermann Agner

Department of Computer Science, Mid-West State University (UNICENTRO), Guarapuava, Paraná, Brazil

Article history
Received: 22-12-2017
Revised: 27-01-2018
Accepted: 13-02-2018

Corresponding Author:
Inali Wisniewski Soares
Department of Computer Science, Mid-West State University (UNICENTRO), Guarapuava, Paraná, Brazil
Email: inali@unicentro.br

Abstract: The Model Driven Engineering (MDE) supports software development, promoting quality and speed enhancement in providing new products. The development of more accurate models is obtained through the employment of the Object Constraint Language (OCL). Given the increasing use of mobile devices and the need for new applications, the MDE together with the OCL constraints promote the development of quality mobile applications when the development time is reduced. This research paper focuses on the definition and use of OCL constraints to support the development of Android models in the MDE context.

Keywords: Model Driven Engineering, Object Constraint Language, Mobile Software

Introduction

Today, new Software Engineering approaches to support the development of mobile applications have been searched. This is due to the features and requirements of this type of software, such as: Potential interaction with other applications, sensor handling, existence of native and hybrid applications, applications must support the families of hardware and software platforms, security, user interfaces, complexity of testing, power consumption (Wasserman, 2010). Therefore, the Model Driven Engineering (MDE) constitutes an appealing alternative (Kent, 2002). MDE is used to describe software development approaches in which abstract models of software systems are created and systematically transformed to concrete implementations (France and Rumpe, 2007).

MDE is a Software Engineering approach that considers models as first class entities and every software artifact as a model or model element (Bézivin, 2005). MDE concerns the exploitation of models as the cornerstone of the software development process. It allows both developers and stakeholders to use abstractions closer to the domain than to computing concepts. Thus, it reduces the complexity and improves communication. As the main aim of MDE resides in developing software, this paradigm uses software models as its main component (Gascueña *et al*., 2012).

In MDE, models are created from a combination between drawings and text, which can be defined in either natural or formal language (Kleppe *et al*., 2003). The Unified Modeling Language (UML) (UML, 2015) is a semi-formal modeling language and the most commonly employed in the MDE context, once it provides several graphic representation options that increase the abstraction level of the projects. In this way, the complexity of projects is reduced and thus they become simpler and more intuitive (Agner *et al*., 2013).

Although the UML provides a number of model options to represent a system, such models are not sufficiently complete to describe all details. The OCL is not only used in meta-modeling to supply a precise semantics for UML diagrams, but also in requirements specification (Hähnle *et al*., 2002). As a result, the Object Constraint Language (OCL) is required, once it is textual, based on predicate logic and set theory (OCL, 2014). OCL allows the more accurate specification of all details of a model, to which constraints, rules and operational contract definitions are added (Warmer and Kleppe, 2003). This language allows modelers to define constraints at different abstraction levels and for different model types, as well as verify constraints through a parser or a constraint verification module (Ali *et al*., 2011).

The popularization of mobile devices has significantly increased the need for easy navigation systems, with attractive design and that provide a large number of applications to the users. Currently, there are three platforms that dominate the mobile market: Android (2017), iOS (Apple, 2017) and Windows

Mobile (Microsoft, 2014). A large number of applications that run on those platforms are available in application stores (Perchat *et al*., 2014). In this study, the Android platform was chosen, once it is free and has open source. The Android platform is a software stack for mobile devices that consists of an operating system, middleware and key applications. Android offers many features covering the areas of application development, Internet, media and connectivity (Android, 2017).

The increasing demand for Android mobile applications requires the improvement of their development approaches (Maji *et al*., 2010). Therefore, the use of modeling can facilitate this process, in which software details are abstracted (Freitas and Maia, 2016). Therefore, employing the MDE approach together with the OCL is appealing, in order to create consistent models in mobile systems and supporting the production of these applications. In this research, the development of OCL constraints is investigated so as to support the development of Android applications in the MDE approach context.

This paper is organized as follows. Section 2 introduces some related work. Section 3 describes the materials and methods. Section 4 describes the obtained results and discussions. Section 5 concludes this paper.

Related Work

Some studies using OCL rules in the MDE approach have already been developed. A metamodel was defined, extending the UML so as to integrate context attributes in mobile applications that are sensitive to such context. In this study, OCL is used to formally define the semantics of new UML elements introduced and ensure the consistency of such elements (Schefer-Wenzl and Strembeck, 2013).

OCL constraints were built to be employed in MDE models. In this study the authors developed models for mobile applications by treating the Graphical User Interface (GUI). The navigation through the generated GUIs uses semantic links that combine the associations and cardinalities among the conceptual domain entities (Da Silva *et al*., 2014).

An automatic verification method of UML class diagrams using OCL constraints was developed. Such method verifies the conformity of the class diagrams in relation to its several correctness properties, such as weak and strong satisfaction and redundancy of constraints (Cabot *et al*., 2014).

Materials and Methods

In this study, a case study was developed to illustrate the MDE applied concepts and present the OCL rules defined for an Android mobile application, called AppCalagem. This application was chosen given its simplicity, incorporated Android features and simplified class diagram generation, with didactic purposes only. The main goal of the application resides in performing the soil liming calculation, supporting agronomy engineers in the soil analysis. The AppCalagem application was developed by using MDE models. Next, the MDE development stages are presented.

Platform Independent Model (PIM)

The PIM represents the formal specification of the structure and function of a system. In this model, the technical implementation and platform details are abstracted (Kent, 2002). A PIM is built at a high abstraction level so as to ensure its independence and allow its reuse in transformations for different platforms. The higher the PIM model independence level, the easier a future transformation to any PSM type will be. Figure 1 shows the PIM model, a static class diagram for the AppCalagem application.

Definition and Application of OCL Rules in the PIM

After the development of the PIM, some OCL rules were defined for the syntactic verification and static semantics verification of the AppCalagem PIM model. The syntactic verification checks whether the constructs of the structural models are correctly defined. The static semantics verification defines the structural constraints for the models and checks whether they were correctly defined. The models were validated in two stages, as described next.

The first stage consisted in checking if the models were defined in accordance with the UML specification, i.e., if the models are correctly described according to the UML syntax. In order to perform the validation stage, the option Validation in the tool Papyrus (2017) must be accessed. If any inconsistency in the models is found, flags in the model elements are defined and errors described in the tab Problem. When correctly validating a model, a message is shown, indicating the operation succeeded.

In the second stage, OCL constraints were defined for the PIM model and it was checked if such constraints were correctly defined. The indication of the constraints defined in the model as successful means that the PIM models are well formed and valid, i.e., they represent a set of elements that can be mapped for a mobile software, therefore for the Android mobile platform in the context of this paper.

The OCL constraints defined for the models are of the invariant type. Such constraints must be true throughout the entire life cycle of the instance (OCL, 2014). The tool Papyrus defines OCL rule groups that are stored in a file by the OCL editor. Then, this group of rules is named and applied to a selected PIM. It is thus possible to validate and ensure that the constructed model is in conformance with the rules of the considered domain.

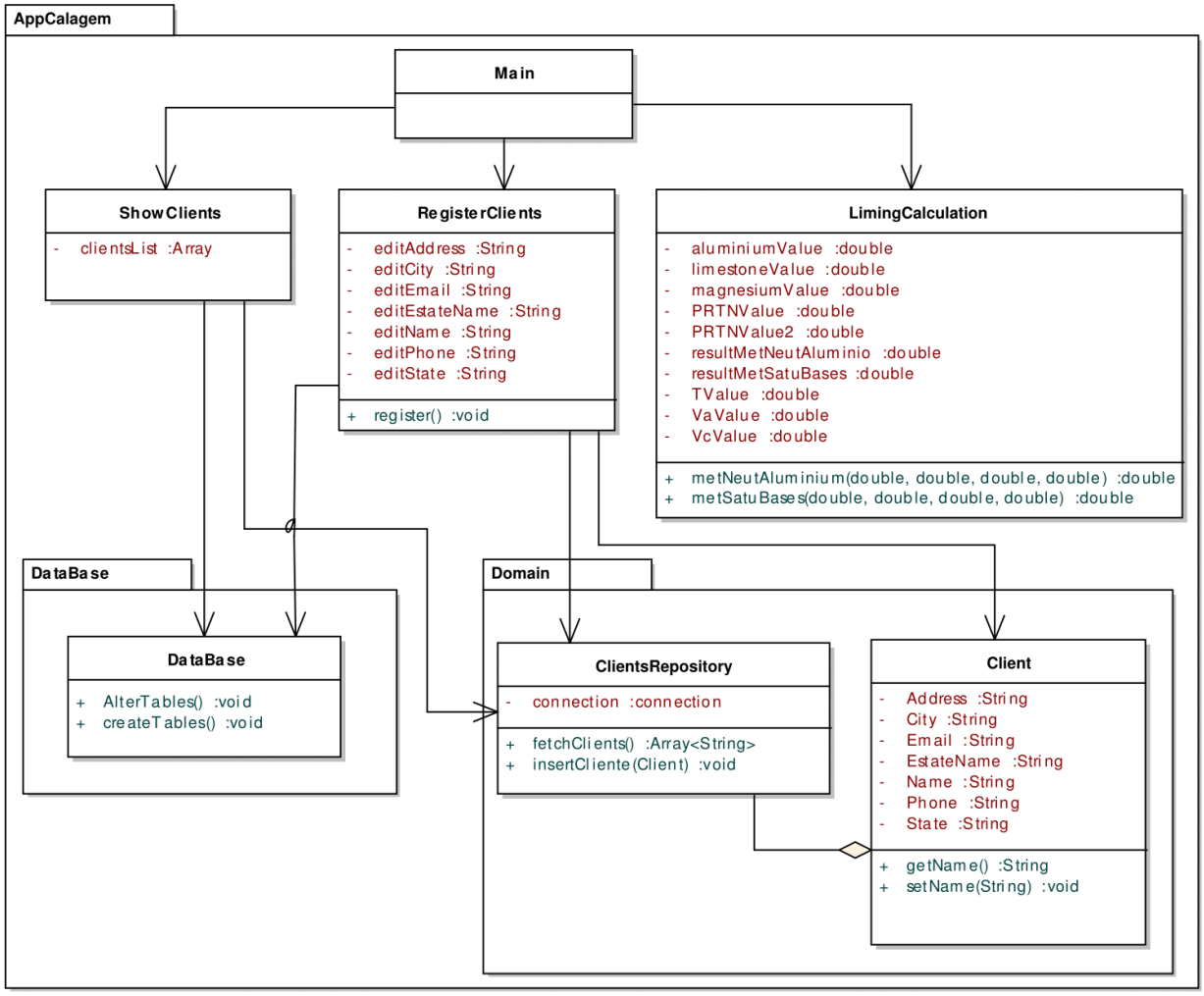

Fig. 1: AppCalagem PIM model

Table 1: OCL Constraints applied to the AppCalagem PIM classes

OCL Constraints	Description in natural language
self.generalization->size()>=1 (context:Class)	Verifies classes with generalization.
not self.name.oclIsUndefined() and self.name <> '' (context:Class)	Checks if the class names are correctly configured.
not self.name.oclIsUndefined() and self.name <> '' (context:Property)	Checks if the property names are correctly configured.
self.nome <> '' (context:Client)	Checks if the property *name* in the class *Client* is different from null.
self.cliente -> forAll(c1.qtdeCalagem-> size > 9 implies c1.tipoCliente == 'Especial' (context:Client)	Verifies if the if the property *qtdeLiming* with a value higher than 9 (nine) has the value '*Especial*' as content.
context Cliente::tipoCliente init: 'Special'	Initializes the property *qtdeLiming* with the string '*Especial'* as content.
def:getClientForType(TypeDesc:String): Set (TypeCliente)=type-> select(name='Special').availableTypeClient->asSet()(context:Client)	Inserts the operation *getClientForType* in the class *Client*. This search operation returns the set of all Types of Client (property called *TypeDesc*) whose content is equal to *Especial*.
self.vlrCalcario <> 0 (context:Liming)	Checks if the property *vlrCalcario* of the class *Liming* is different from 0.
Liming::vlrAluminio init: 0 (context:Liming)	Initializes the property *vlrAluminio* of the class *Liming* with value different from 0.
ClienteIU::editNomeEmail derive:owner.editNome.concat(' '). concat(owner.editEmail) (context:Liming)	Specifies a derived element for the property *editNomeEmail* derived from the properties *editnome* and *editemail* of the class *ClienteIU*.

Table 1 describes some examples of rules defined for the AppCalagem case study. These constraints were applied to different elements of a UML model. The reserved word "context" in OCL defines for which UML model element this constraint is valid. For instance, the first two constraints described in Table 1

are applied in the context of a Class. The objectives of these rules reside in verifying if the model classes are generalized and if the class names are correctly configured, respectively.

Definition of the Platform Model

A Platform Model (PM) consists of a set of characteristics and definitions about a certain platform as well as the services it provides. A PM is used to demonstrate all elements of a platform (MDA, 2003).

The PM-MDE defined in Soares *et al*. (2012) was used in the development of the PM, but with adaptations for the construction of mobile applications. Figure 2 shows the PM created to represent some features of the Android platform used in the implementation of the AppCalagem application. As illustrated in Fig. 2, these are the classes that provide Android specific functionalities. Classes SQLiteDataBase and SQLiteOpenHelper provide methods and objects for the platform connection and database management. Classes Activity and AppCompatActivity provide the elements responsible for the creation and interaction with the XML files that form the application graphical interface. The class View is responsible for the design and manipulation of components in the interface. This class contains the components of the Widget package such as buttons (Button), text boxes (EditText), among others, as illustrated in Fig. 2.

The class Context is responsible for presenting global information about the application environment. It is implemented in the android platform and its use allows the access to specific resources of this platform. The class Intent is responsible for storing the application temporary data at a certain point in time, as well as the activity logging. At last, the class Bundle maps the Strings for key values that can be persisted or restored from the disc.

Definition of the Platform Specific Model

A Platform Specific Model (PSM) provides a software view by integrating specific details of the implementation platform and can be created at different levels of detail. In the PSM, specifications already described in the PIM are combined and specificities of the platform on which the system will be executed are added. To do so, a selected PM is used (MDA, 2003). Therefore, the PSM serves as key to the MDE most important principles, that is, the code automatic generation, file systems and identification processes (Heck, 2005).

The Fig. 3 presents the PSM model of the AppCalagem application. As it can be observed, the detailing level of this model is higher is relation to the PIM and as the application is implemented in the Android platform, the PM illustrated in Fig. 2 was selected to define this PSM.

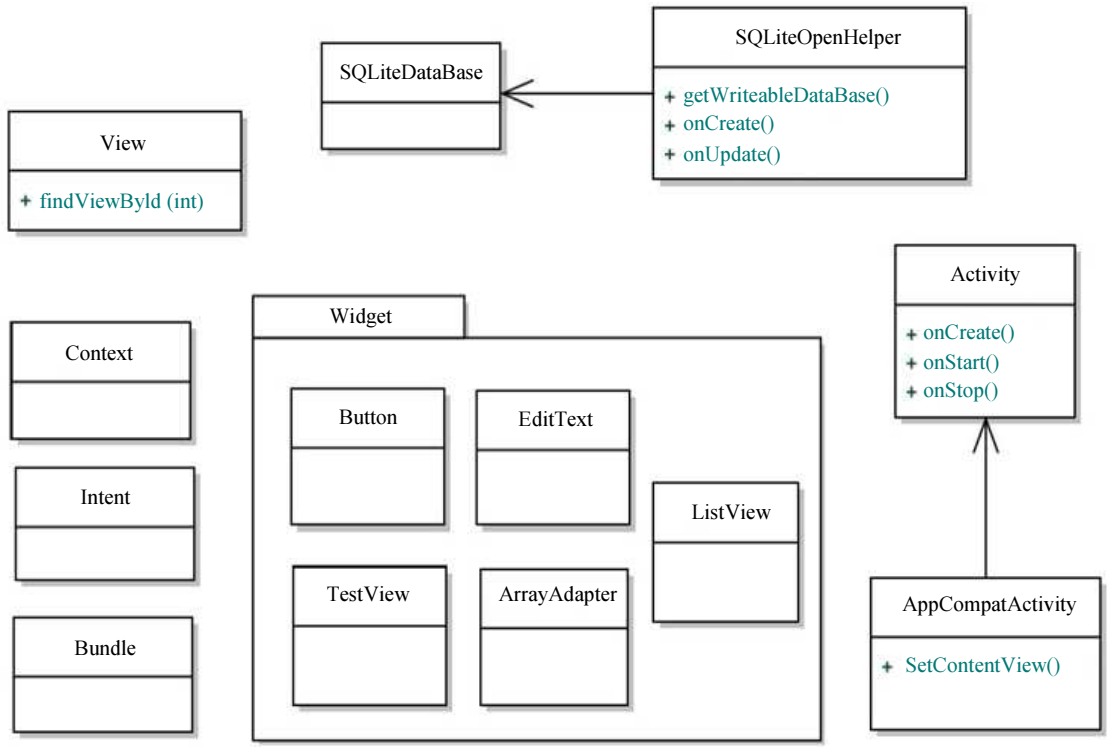

Fig. 2: AppCalagem platform model

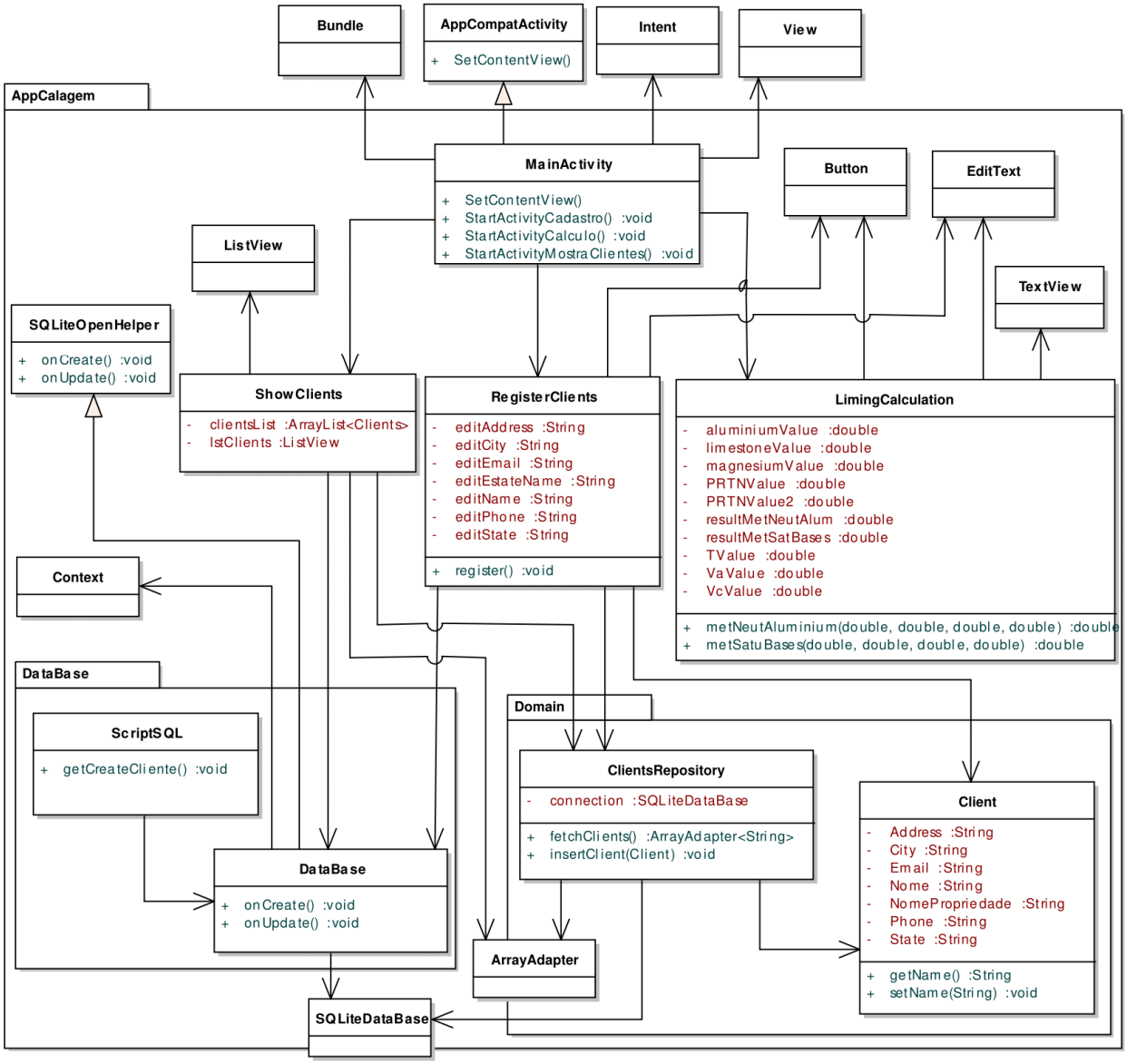

Fig. 3: AppCalagem PSM model

Results and Discussion

Initially, the AppCalagem application static PIM model was defined. When developing the models, the UML modeling language was employed in the MDE approach. OCL rules were defined and applied to the application PIM model.

The OCL can be used by software engineers to define the constraints and semantics of the operations of a model in a more accurate and non-ambiguous way. The OCL represents a crucial aspect in the MDE approach, considering that without a well defined language as the OCL, consistent, coherent and platform independent models could not be defined (Warmer and Kleppe, 2003). The verification ensures that the system was correctly built. Validation allows the system to perform the user requisites (Pressman, 2011). These two activities were performed in two stages; first when validating the PIM model and then defining and applying the OCL rules to the PIM.

The PIM model definition was based on metamodels. Therefore, this model can be validated through validation tools of models and metamodels. The metamodels used in this study were the UML and the OCL, respectively. All models were defined by using the modeling tool Papyrus.

The AppCalagem application was also built by using the traditional development process for this type of

application. As a result, a practical perception of the involved technical aspects on the Android platform internal structure could be attained

Conclusion

Through the use of the MDE approach in the development of an Android application, it was possible to see the importance of modeling for software development in practice. At a first development of an Android application, a lot of effort is required to define the stages of the MDE approach. However, future developments can profit from most of this effort.

The application traditional development was crucial for the understanding of concepts and details referring to aspects regarding mobile applications in the Android platform. The technical expertise achieved helped the construction of PM and PSM models of the MDE approach for the case study in this study.

In this research, OCL constraints were studied, defined and applied to the application structural PIM, more specifically to the UML class diagram. OCL constraints allow increasing the accurateness of the information provided in these models. That results in more robust and complete models in accordance with the system specifications.

The application of OCL rules to Android models, as performed in the case study present in this study, is relevant in the MDE approach development process. Given that one of the main activities developed within the MDE process is the automatic or semiautomatic generation of the source code from the application models, the transformation of a model to generate the source code requires all data in the models to be accurately defined in order to prevent errors in the code or misunderstandings resulting from ambiguous information.

As future works, a transformation of models for Android applications will be built with the support of the OCL constraints defined in this study. Such constraints can also be employed in a model transformation language for the development of Android applications in the MDE approach context.

Funding Information

The authors have no support or funding to report.

Author's Contributions

Jean Carlos Hrycyk: Investigation, writing and edition.

Inali Wisniewski Soares: Writing, edition and supervision.

Luciane Telinski Wiedermann Agner: Writing, edition and revision.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Agner, L.T.W., I.W. Soares, P.C. Stadzisz and J.M. Simão, 2013. A Brazilian survey on UML and model-driven practices for embedded software development. J. Syst. Software, 86: 997-1005. DOI: 10.1016/j.jss.2012.11.023

Ali, S., M.Z. Iqbal, A. Arcuri and L. Briand, 2011. A search-based OCL constraint solver for model-based test data generation. Proceedings of the 11th International Conference on Quality Software, Jul. 13-14, IEEE Xplore Press, Madrid, Spain, pp: 41-50. DOI: 10.1109/QSIC.2011.17

Android, 2017. http://www.android.com

Apple, 2017. http://www.apple.com

Bézivin, J., 2005. On the unification power of models. Software Syst. Model., 4: 171-188. DOI: 10.1007/s10270-005-0079-0

Cabot, J., R, Clarisót and D. Riera 2014. On the verification of UML/OCL class diagrams using constraint programming. J. Syst. Software, 93: 1-23. DOI: 10.1016/j.jss.2014.03.023

Da Silva, L.P., E. Abreu and F. Brito, 2014. Model-driven GUI generation and navigation for android BIS apps. Proceedings of the 2nd International Conference on Model-Driven Engineering and Software Development, IEEE Xplore Press, Lisbon, Portugal, pp: 400-407. DOI: 10.5220/0004715504000407

France, R. and B. Rumpe, 2007. Model-driven development of complex software: A research roadmap. Proceedings of the Future of Software Engineering, May 23-25, IEEE Xplore Press, Minneapolis, MN, USA, pp: 37-54. DOI: 10.1109/FOSE.2007.14

Freitas, F. and P.H.M. Maia, 2016. JustModeling: An MDE approach to develop android business applications. Proceedings of the 6th Brazilian Symposium on Computing Systems Engineering, Nov. 1-4, IEEE Xplore Press, Joao Pessoa, Brazil, pp: 48-55. DOI: 10.1109/SBESC.2016.016

Gascueña, J.M., E. Navarro and A. Fernández-Caballero, 2012. Model-driven engineering techniques for the development of multi-agent systems. Eng. Applic. Artificial Intell., 25: 159-173. DOI: 10.1016/j.engappai.2011.08.008

Hähnle, R., K. Johannisson and A. Ranta, 2002. An Authoring Tool for Informal and Formal Requirements Specifications. Proceedings of the 5th International Conference on Fundamental Approaches to Software Engineering, Apr. 8-12, Springer, Grenoble, France, pp: 233-248. DOI: 10.1007/3-540-45923-5_16

Heck, S., 2005. Model transformation for verification: Building the basis for a generic tool. Bachelor Thesis, Centre Universitaire D'Informatique, Universite de Geneve.

Kent, S., 2002. Model driven engineering. Proceedings of the 3rd International Conference on International Conference on Integrated Formal Methods, (IFM' 02), Springer, Berlin, Heidelberg, pp: 286-298. DOI: 10.1007/3-540-47884-1_16

Kleppe, A., J. Warmer and W. Bast, 2003. MDA Explained: The Model Driven Architecture: Practice and Promise. 1st Edn., Addison Wesley, ISBN-10: 032119442X, pp: 170.

Maji, A.K., K. Hao, S. Sultana and S. Bagchi, 2010. Characterizing failures in mobile OSes: A case study with android and Symbian. Proceedings of the IEEE 21st International Symposium on Software Reliability Engineering, Nov. 1-4, IEEE Xplore Press, San Jose, CA, USA, pp: 249-258. DOI: 10.1109/ISSRE.2010.45

MDA, 2003. OMG. MDA Guide Version 1.0.1. http://www.omg.org/cgi-bin/doc?omg/03-06-01.

Microsoft, 2014. http://www.microsoft.com

OCL, 2014. OMG. Object Constraint Language Specification (OCL). http://www.omg.org/spec/OCL/2.4>

Papyrus, 2017. http://www.eclipse.org/papyrus

Pressman, R., 2011. Engenharia de Software: Uma Abordagem Profissional. 8th Edn., McGraw Hill.

Perchat, J., M. Desertot and S. Lecomte, 2014. Common framework: A hybrid approach to integrate cross-platform components in mobile application. J. Comput. Sci., 10: 2165-2181. DOI: 10.3844/jcssp.2014.2165.2181

Schefer-Wenzl, S. and M. Strembeck, 2013. Modelling context-aware RBAC models for mobile business processes. Int. J. Wireless Mobile Comput., 6: 448-462. DOI: 10.1504/IJWMC.2013.057387

Soares, I.W., L.T.W. Agner, P.C. Stadzisz and J.M. Simão, 2012. A method for the development of platform models in the model driven architecture context. J. Comput. Sci., 8: 1932-1939. DOI: 10.3844/jcssp.2012.1932.1939

UML, 2015. OMG. Unified Modeling Language (UML) Superstructure specification, version 2.5 (formal/2015-15-03). Object Management Group.

Warmer, J.B. and A.G. Kleppe, 2003. The Object Constraint Language: Getting Your Models Ready for MDA. 1st Edn., Addison-Wesley Professional, Boston, ISBN-10: 0321179366, pp: 206.

Wasserman, A.I., 2010. Software engineering issues for mobile application development. Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research, Nov. 07-08, ACM, Santa Fe, New Mexico, USA, pp: 397-400. DOI: 10.1145/1882362.1882443