International Journal of Conceptions on Computing and Information Technology
Vol. 6, Issue. 2, November' 2018; ISSN: 2345 - 9808

Creating a metamodel of UI components in form of model independant of the platform

William Germain DIMBISOA
Laboratory for Mathematical and Computer Applied to the Development (LIMAD)
University of Fianarantsoa
Fianarantsoa, Madagascar
dwilliamger@gmail.com

Thomas MAHATODY
Laboratory for Mathematical and Computer Applied to the Development (LIMAD)
University of Fianarantsoa
Fianarantsoa, Madagascar
tsmahatody@gmail.com

Josvah Paul RAZAFIMANDIMBY
Laboratory for Mathematical and Computer Applied to the Development (LIMAD)
University of Fianarantsoa
Fianarantsoa, Madagascar
jp_josvah@yahoo.com

Abstract— **The design is an inevitable step during the software development. Model Driven Engineering (MDE) is becoming a new paradigm to facilitate the software modeling. The aim of this paper is to define the user interface components (UIC) as a model independent of a specific platform. The Human Computer Interaction (HCI) is the means for the user to communicate with a machine. It includes the visible part called the user interface and the invisible part containing the source codes with services of functionalities. This study is interested in the visible part. Designers of an UI use their own techniques to create software. The result may not match users' requirements and needs. So, it is important to use an approach to make easy and fast the designer task. This study also places the model as the basis for the HCI design and modeling to ensure components reuse. We prefer to use the MDA which is an approach based on the MDE. This paper let us to create a metamodel of the visible part of HCI.**

Keywords- metamodel; component; UI components (UIC); component reuse; specific platform; HCI; MDE.

I. INTRODUCTION

The computer user interacts with an interactive system through an interface. An interactive system is composed of the user interface and functional kernel that provide interaction [1]. UIC and widgets are basic elements for the Graphical User Interface (GUI) design [2] with which the user can interact. These components are usually grouped in toolkits. These Widgets form a complete GUI after the designer has assembled their elements. The good or bad appearance of the HCI depends on the layout and formatting of each Widgets. Designers of HCI do not have the same way to design and generate a GUI [3]. Each of them has their own abstraction level, their own style and aesthetic invention to create a good HCI design [4]. And the final product may not match user's needs and requirements. For this, it is important to use software bricks to facilitate and speed up the creation of UI.

Currently, a new paradigm called Model Driven Engineering (MDE) [5] appears to simplify the software design. This new concept highlights the model as the basis of modeling. Research like [6,7] makes the automatic generation of the HCI without considering its components.

This paper aims to describe the UIC in the form of an independant model of a specific platform according to the MDE approach. UI can be obtained from this component metamodel. Meta-modeling begins with the creation of the UIC metamodel, and ends optionally by obtaining a GUI. We prefer to use the Model Driven Architecture (MDA) approach which is an approach to model transformation. We can obtain a standard type of GUI from a UIC metamodel. But, we can use and integrate other technologies to obtain the web or mobile interface. Firstly, we will describe what concerns the MDA approach. Secondly, we will talk about metamodeling. Finally, we will present the result of the study ending with a discussion. We can test the result using a development environment and required technologies.

II. MDA APPROACH

MDA [8] is a particular variant of the MDE approach. His strategy is based on the transformation of the reality model into an abstract model. Its fundamental principles are to create a model of the existing and transform this model until reaching a model that describes the objective. Transformation techniques and rules are applied to support the transformation of the source model into a target model. In MDA, everything is considered a model. The initiative of this approach is to minimize human intervention during the software design and modeling. In the field of software engineering, the process of creating software is a very complex task for computer scientists. The software designer and the user are not the same language, so MDA facilitates the software design and modeling.

Recently, the Object Management Group (OMG) has stimulated and promoted the adoption of the Model Driven Architecture (MDA) approach for developing large and complex software systems [9]. MDA is divided into three levels of model. The first model called Computational Independant Model (CIM) represents users' needs and

requirements. This first model ensures transformation, it represents the main model of modeling. The second model called Platform Independant Model (PIM) describes the abstract model. This model is obtained from the transformation of the first model, it ensures the automatic or semi-automatic generation of the final application of source code. The third model called Platform Specific Model (PSM) represents the concrete final application. Fig. 1 represents the architecture based on the MDA approach.

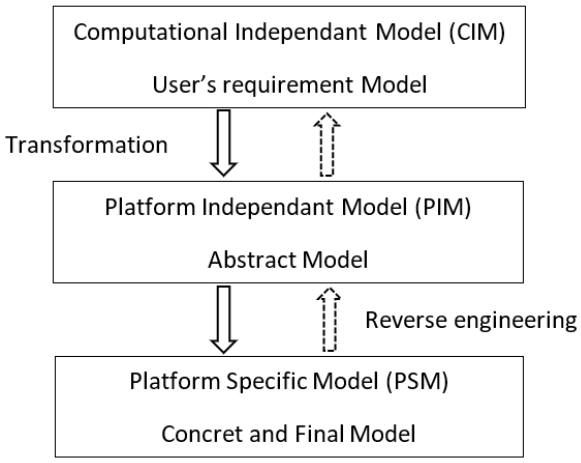

Figure. 1: MDA approach model level

III. META-MODELING

Design and modeling of HCI are fundamental steps for the software creation. The preliminary step of this approach is to model user needs and requirements. This first step reflects a part of the real world as a studied system. A system is defined as a fictional representation of the real world. In the software production company, the principle is based on the coding a software. Recently, MDE approach changes this concept considering that everything can be represented as a model. This model facilitates designer tasks. A model must be representative of designers and users of the system, and ultimately it can be transformed into an execution platform. It is also an abstraction of a system and should be used to answer questions about the modeled system. Presently, models are first class artifacts in a modeling domain where models are capable of providing a rigorous understanding of the system on a more abstract level [10]. Indeed, a model must meet the following two criteria:

- Contemplative model: model that is understandable for the user but remains far from the concept of production;
- Productive model: model that can be interpreted and executed by a computer. It is likely to be automatically transformed between them.

The model is transformed into a metamodel by applying transformation rules and approach. The characteristic of the MDE is to use productive models and metamodels. The metamodel is the representation of a particular point of view on models, it is a description of modeling languages. A metamodel represents an abstract syntax of models, it fully covers the definition of metamodeling concepts. The metamodel is a collection of data based on Meta Object Facility (MOF) framework [10]. It is also transformed into meta-metamodel after transformation. Design and the modeling of an HCI put in general three levels of abstractions: Abstract UI, Concrete UI, and Final UI. To move from one level to another, you need a transformation. For the realization of a UI, it is very important to use Framework to make easy and fast the software realization process. For example, the use of CAMELEON Framework [11] allows to separate models into four main levels: task model, Abstract UI (AUI), Concrete UI (CUI) and Final UI (FUI). The production of CUI's by reification is one approach to the problem: typically, the process starts from high-level descriptions such as task and domain models to produce an AUI, then from an AUI, produces one or multiple CUI's. Alternatively, an existing CUI may be reverse-engineered by means of abstraction to obtain an AUI and/or a task model. These abstract representations are then translated to fit the new target, then reified into new CUI's [11]. This architecture conforms to the MDA approach. For the metamodel creation of UI components, modeling starts at the AUI and CUI. We aim to describe EMF with ECore and FUI components.

A. EMF And ECore

Standardized by OMG, Eclipse Modeling Framework (EMF) [14] is an abstract language and framework for specifying, creating and managing technology which is based on meta-models. EMF is also a standard module for the MDE software engineering concept. It provides an environment for the model creation and transformation. It was focused on highlighting the metamodel based on ECore. Metamodel is a description language of model. EMF model describes a specification of a system and reality. We can generate implementation codes, UI and Eclipse projects or plugins from existing model. EMF provides a metamodel which is a description of ECore. It also offers lots of tooling support within the Eclipse plugin. The main goal of EMF concept is to transform models into efficient, correct, and easily customizable Java code and UI.

B. UI Components

To facilitate the UI development, it is very important to use widget components. An interactive system is composed of two distinct parts: the functional kernel and the UI. The part of the functional kernel represents the elements of interaction when the user interacts. The functional kernel also groups all features independently of any representation to the user. The part of the UI represents the visual elements.

These visual components can be dynamic interaction elements (clickable) or visual static elements (non-clickable). These components are grouped by the composition [12] strategy, but this study aims to create the model of components for the visual UI elements. In general, a component model consists of: (a) properties having a set of characteristics either

related to communication and assembly (notification of modification, ...) or related to interfacing; (b) interface elements that can be dedicated to an interfacing tool. But, this article aims at the creation of the metamodel of components of the UI constituent elements. HCI can be designed using the model of UI components. The Windows, Icons, Menus, Pointing devices (WIMP) interface type is built by combining a number of components, nested within each other. There are two types of GUI components: (a) basic components: they do not contain other components. These are static and dynamic elements (buttons, text box, label, …) of a user interface; (b) containers: to contain and organize other components (windows) and basic components. They are separated into two categories: intermediate containers which could themselves be contained by other containers and top-level containers which are the main containers of a graphical interface.

These graphic components are organized as follows: the main container also called parent window, the intermediate container and the basic components. When a screen is displayed, components occupy a rectangular area. Wires of a container usually share its area, and there is no overlap between sibling components. Closing the intermediate window no longer leads to the closing of the main window. The component hierarchy is organized and flattened into a set of nested rectangular areas. Fig. 2 shows an example of UI components.

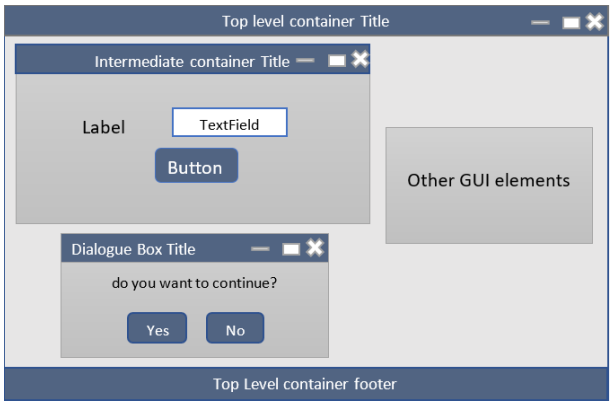

Figure 1. Example of UI components

HCI elements in Fig. 2 are all organized in the main window. In the software development process, GUI model must be returned to users to test their needs. The modification of the final model depends on the validation of the potential user. Adapting plastic HCI [13] to users' requirements is a very complex task. So, MDE concept facilitates its tedious tasks for computer scientists. Each element of the HCI becomes a reusable component model. In this study, we propose HCI standard components model to become a platform independant model.

These components are also linked by internal events to be active or not active. The functional kernel ensures relationships between these events which are not visible in the interface. In this paper, we will describe perennial models for the visible elements of GUI components.

IV. Research Result

Our strategy aims to describe UI components and define the metamodel based on ECore. We create a standard metamodel of GUI components and put this metamodel as an independant model of a specific platform.

A. UI Components based on ECore

The MDE concept highlights the hierarchical level of modeling. By using transformation rules, the model of reality is transformed into an abstract model which answers users' needs. The transformation is carried out step by step from the real world until obtaining the source code. Focusing on the MDA approach and using the EMF metamodel, we create the metamodel based on ECore. We provide a metamodel for the standard GUI components [15]. This paper describes components as a model independant of a specific platform. This independence is very essential for interfacing information and ensures the durability and reusability of HCI components. The creation of the metamodel is inspired by the GUI component metamodel of the Java programming language.

B. UI Components metamodels

HCI components become a standard [16] metamodel. We do not consider services [17] that ensure the operation of each component. But we propose a metamodel of GUI components without dealing with the interactions between these components. Fig. 3 shows the metamodel of GUI components.

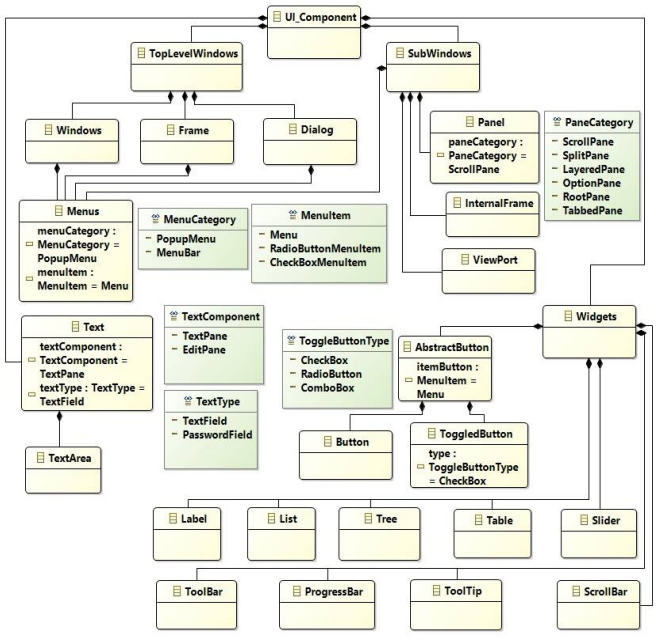

Figure 2. UI components metamodel based on Ecore

This metamodel is a general view of UI components. The concept of his creation is based on the UI metamodel in java. Each class of this model can be customized when designing software. Properties must be assigned to each class, but

methods are optional. The vocabulary used depend on the type of software to be designed.

C. Case of study

We take the example of a window or dialog box for an authentication example. We design elements that can be displayed in a UI. Then, we create participating classes. Main components of an authentication window are: labels, texts and buttons. They are placed in a UI according to users' needs and each of them has its own characteristics. Fig. 4 shows the metamodel of components of an authentication window.

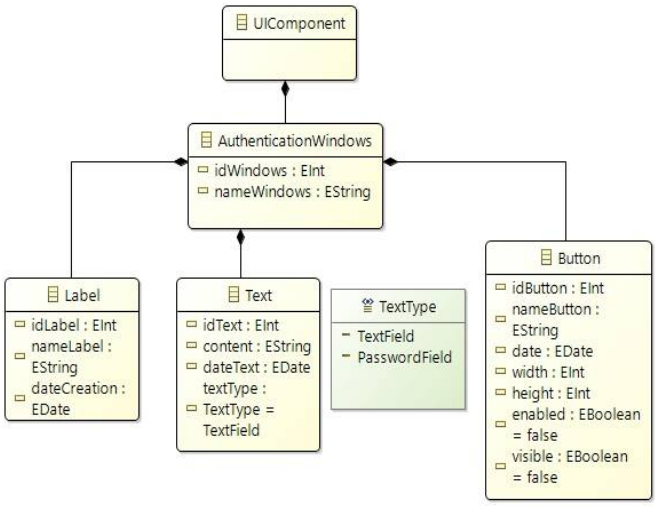

Figure 3. Metamodel of components of an authentication window

Using the realization platform, we can create a View Model Project from the ECore metamodel. There are two models of view: View Editor and View Editor Preview. This second View represents a UI containing widgets or elements of a UI. This interface is used to visualize properties of widgets for interface components. Fig. 5 shows the two views of the metamodel.

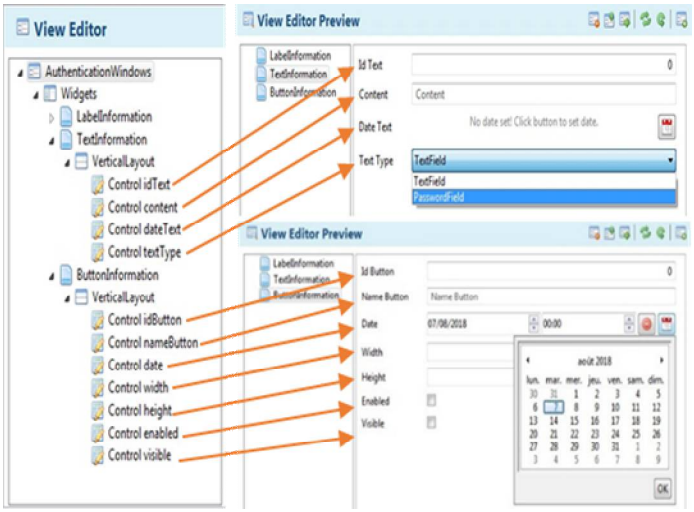

Figure 4. Two view of the UI metamodel

We get an interface with basic elements of an example dialog box for authentication. [6] produces a specific UI from an HCI model. But in this study, the final UI represents the characteristics of UI components.

V. DISCUSSION

The paper [18] aims at the dynamic composition of the HCI by automated planning. [19,20] also describe the UI composition. This work focuses on the use of interface components, considering controller services of UI elements to ensure the own operation. In this article, we create the metamodel of UI standard components. We do not consider controller services of each component and the link between these widgets. So, we highlight HCI as an interface without dealing with user interactions with UI components. We adopt the MDE approach to complete meta-modeling.

MDA is a particular variant of the MDE approach. The concept of the MDA makes it possible to create an Ecore based metamodel using the EMF Framework. MDE also offers modeling and application design at a high level of abstraction. It places the model as the basis of the design process and also generates the code of an application. More advantages of using a model are:

- Abstraction: a model highlights important points while removing unnecessary details;
- Comprehensibility: a model makes it possible to express a complex thing in a form that is easier for users to understand;
- Precision: a model faithfully represents the modeled system;
- Prediction: a model makes it possible to make correct predictions on the modeled system;

Based on MDE, our strategy also provides the following benefits:

- Increased productivity and model reusability;
- Acceleration of software development;
- Design an application by targeting technologies and platforms;
- Durability of designed applications: maintenance, adaptation to changes;
- Target multiple execution platforms from a single design;
- Control, simulation and test to different technologies.

This paper ultimately produces a high-level abstraction model of HCI components. It also describes GUI metamodels. The basic concept is to implement HCI component models as independant platform models. Tools make it possible to complete the realization of metamodels creation. But we adopt the Eclipse implementation environment and EMF to be able to create metamodels based on ECore. An example of a simple interface validates our case study. In this study, we developed the platform independant metamodel.

VI. CONCLUSION AND PERSPECTIVE

In the software development lifecycle, the HCI design is a preliminary step that must be highlighted. In this paper, we have created metamodel of the standard GUI components. This study also describes components in form of a model independant of a specific platform. The MDE approach accelerates the development of software. It also makes it easy to design an application by targeting technologies and platforms.

We use the EMF module for creating the metamodel based on Ecore. Focusing on the MDA, we obtain as a result the metamodel of GUI Widgets components. We can directly generate HCI from the created metamodel. Our approach reduces the tasks of the designer during the realization of a computer project. As a perspective, we will consider creating a platform for UI components so that its components become standard and independant of a specific platform.

REFERENCES

[1] M. B. Lafon, "Designing Interaction, not Interfaces", In Proceedings of the Working Conference on Advanced Visual Interfaces, pp.15-22, 2004.

[2] K. S. Vallerio, L. Zhong and N. K. Jha, "Energy-Efficient Graphical User Interface Design", in IEEE Transactions on Mobile Computing, vol. 5, pp. 846-859, 2006.

[3] Gajos, Z. Krzysztof, S. Daniel. Weld, and O. W. Jacob, "Automatically generating personalized user interfaces with SUPPLE", Artificial Intelligence 174(12-13): 910-950, 2010.

[4] C. Kruschitz, M. Hitz, "Human-Computer Interaction Design Patterns: Structure, Methods, and Tools", International Journal on Advances in Software, 2010.

[5] J. Vanderdonck, "Model-Driven Engineering of User Interfaces: Promises, Successes, and Failures", In Proceedings of 5th Annual Romanian Conference on Human-Computer Interaction, ROCHI, 2008.

[6] W. G. DIMBISOA, T. MAHATODY, J. P. RAZAFIMANDIMBY, "Automatically generate a specific Human Computer Interaction from an interface diagram model", 4th International Conference on Computer and Technology Applications Conference Abstract (ICCTA), Istanbul Turkey, 2018.

[7] Kolb, Jens and Hübner, Paul and Reichert, Manfred, "Model-Driven User Interface Generation and Adaptation in Process-Aware Information System", On the Move to Meaningful Internet Systems: OTM 2012: Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2012, Rome, Italy, September 10-14, Proceedings, Part, 2012.

[8] A. A. A. Jilani, M. Usman, Z. Halim, "Model Transformations in Model Driven Architecture", Universal Journal of Computer Science and Engineering Technology, October 2010.

[9] D. Lopes, S. Hammoudi, J. Bézivin, F. Jouault, "Mapping Specification in MDA: From Theory to Practice", Interoperability of Enterprise Software and Applications pp 253-264, 2006.

[10] M. A. Isa, D. N. A. Jawawi and M. Zulkifli, M. Zaki, "A Formal Semantic for Scenario-Based Model Using Algebraic Semantics Framework for MOF", International Journal of Software Engineering and Its Applications, Vol. 7, No. 1, January, 2013.

[11] L. Balme, A. Demeure, N. Barralon, J. Coutaz, G. Calvary, "CAMELEON RT: A Software Architecture Reference Model for Distributed, Migratable, and Plastic User Interfaces", Ambient Intelligence pp 291-302, 2004.

[12] C. Brel, P. R. Gonin, A. M. P. Déry, M. Riveill, "Application and UI composition using a Component-Based Description and Annotations", 38th Euromicro Conference on Software Engineering and Advanced Applications, Izmir, Turkey. pp.204-207, Sep 2012.

[13] J. Coutaz, G. Calvary, "HCI and Software Engineering for User Interface Plasticity", Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications, Third Edition, CRC Press, pp.1195□1220, 2012.

[14] K. Ehrig and G. Taentzer and D. Varro, "Tool Integration by Model Transformations based on the Eclipse Modeling Framework", EASST Newsletter, volume 12, 2006.

[15] T. Amirhosein, L. Frederic, R. Romain, E. Frank, "A Generic Component-based Approach for Programming, Composing and Tuning Sensor Software", the Computer Journal, August 2011.

[16] M. Novak, I. Magdalenic, D. Radosevic, "Common Metamodel of component diagram and feature diagram in generative programming", Journal of Computer Sciences, 2016.

[17] T. Aubonnet, L. Henrio, S. Kessal, O. Kulankhina, F. Lemoine, et al.., "Management of service composition based on self-controlled components". Journal of Internet Services and Applications, Springer, 6 (15), pp.17, 2015.

[18] Y. Gabillon, M. Petit, G. Calvary, H. Fiorino, "Automated Planning for User Interface Composition", International Conference on Intelligent User Interfaces, Feb 2011.

[19] F. Lemoine, T. Aubonnet, L. Henrio, S. Kessal, E. Madelaine, et al.. "Monitoring as-a-service to drive more efficient future system design", EAI Endorsed Transactions on Cloud Systems, 3 (9), pp.1 - 15, 2017.

[20] O. Davidyuk, E. Gilman, I. S. Milara, J. Makipelto, M. Pyykkonen and J. Riekki, "iCompose: Context-Aware Physical User Interface for Application Composition", Central European Journal of Computer Science, (1)4, pp. 442-465, 2011.