Towards Including Layout Properties for Modeling Graphical User Interfaces

Generic Properties for GUI Metamodels

Sarra Roubi[1], Mohammed Erramdani[1] and Samir Mbarki[2]
[1]*High School of Technology, Mohamed First University, Oujda, Morocco*
[2]*Department of Computer Science, Ibn Tofail University, Kenitra, Morocco*

Keywords: Model Driven Engineering, Layout, Transformation, Model, Meta Model, User Interface, Position.

Abstract: Software applications have come to simplify the task for users and offer them automated functionalities. These applications must therefore contain high-performance and efficient user interfaces in order to translate correctly the user's needs. Indeed, several elements contribute to the ergonomics of these interfaces, among them the position and layout of the graphical components which play a very important role to ensure this. However, the design and implementation of such user interfaces for different platform using several programming languages can be tedious and time consuming, especially when the application gathers a large number of interfaces or screens. Since the model driven engineering aims at automating the process of development and raising the level of abstraction, we can use model driven principles to help users choose the right component in the right position on the interface. That is why we present an approach that combines model driven engineering principles and the graphical user interfaces to handle automated layout and position.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

Today, developing high level applications requires an approach to software architecture that helps architects evolve their solutions in flexible ways. Besides, generating application and reuse of code to focus on functionalities rather that code effort is required.

These ideas, among others, were considered central by the Object Management Group (OMG) to address several challenge raised by new software technologies.

The OMG, which is a consortium of software organizations, aims at developing and supporting specifications to improve the practice of enterprise software development and deployment. It encourages efficient use of system models in the software development process and puts the model at the heart of the development process. Such approaches promise improvements in terms of quality and cost by raising the abstraction level of the development.

Moreover, it is through the graphical interfaces that the user can interact with the application and use the functionalities that are offered. Besides, the presentation layer should be ergonomic and well presented. On the other hand, it should face several challenges, among them, the diversity of the interaction devices which certainly involves multiples interaction platforms.

That is why several researches have applied model-driven techniques to the specification of software application and precisely interfaces and user interactions. Among them, the ones focusing on Web interfaces like OOH-Method (Gmez et al, 2001), WebML (Ceri et al, 2002) and RUX-Model (Linaje et al, 2007). Furthermore, some approaches apply model driven techniques for multi-device UI modeling, such as TERESA (Berti et al, 2004), MARIA (Paterno et al, 2009), IFML (Brambilla et al, 2014), (Roubi, Erramdani and Mbarki, 2016). However, there is no complete approach that handles layout and constraints for components and user interface.

In this paper, we propose a model driven approach taking into account the layout and position from the input model of the whole process.

The paper is organized as follows section 2 summarizes related works. Section 3 and 4 present

Roubi S., Erramdani M. and Mbarki S.
Towards Including Layout Properties for Modeling Graphical User Interfaces - Generic Properties for GUI Metamodels.
DOI: 10.5220/0006272505560560
In *Proceedings of the 5th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2017)*, pages 556-560
ISBN: 978-989-758-210-3
Copyright © 2017 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

respectively the proposed approach with examples and Section 5 concludes.

2 RELATED WORK

This work is related to several previous works dealing with conceptual modeling of software applications, especially the graphical aspect. Some of these works focused on the web platform: The Web Modeling Language (WebML) (Ceri, 2002), defined as a conceptual model for data-intensive Web. Also, the OO-HDM method by (Schwabe and Rossi, 1995) presents an UML-based approach for modeling and implementing Web application interfaces. Moreover, WebDSL (Groenewegen et al., 2008) is a domain-specific language consisting of a core language with constructs to define entities, pages and business logic.

Some researches apply model based approaches for multi-device user interface development. Among them we can cite: TERESA (Transformation Environment for inteRactivE Systems representations) (Berti et al, 2003) and MARIA with (Paterno et al, 2009). Also, UsiXML (USer Interface eXtended Markup Language) (Vanderdonckt, 2005). Another related work on applying MDA approach for Rich Internet Applications is found in (Martinez-Ruiz et al, 2006). The approach is based on XML User Interface description languages using XSLT as the transformation language between the different levels of abstraction

Other recent proposals in the Web Engineering field represent the RIA foundations (Urbieta et al, 2007) by extending existing Web engineering approaches. We also find combination of the UML based Web Engineering (UWE) method for data and business logic modeling with the RUX-Method for the user interface modeling of RIAs was proposed as model-driven approach to RIA development (Preciado et al, 2008).

These methods focused on generating the application code in general and several elements that handle layout management are not taken into account. Consequently, these methods do not focus on generating the application and its interfaces without taking into account the position and layout manager. This task is done by hand and can be time consuming and needs more rework.

In this paper, we propose an idea to develop which consider completing Meta Models with generic elements that help generating the layout manager while transforming the whole model. Those meta elements should be generic and not related to a specific platform or technology, so it can be used with several methods and approaches.

3 PROPOSED MODEL DRIVEN ENGENEERING

3.1 Proposed Meta Model

In order to automate the process of generating graphical interfaces for Web and Desktop, we proposed a Meta Model as a Platform Independent Model. This Meta Model simplifies the task for users to design their application in terms of major functionalities. Each one of these functionalities is divided into operation and action performed by the user.

The proposed PIM meta model contains the following :

- **Use Case:** describes the main functionality offered by the system.
- **MainOperation:** express the concept of the generic operation performed by the user to interact with the system. This operation is divided into several atomic activities.
- **Task:** represents the atomic task done by the user to handle a part of the main operation; (select an element from a list, input information).
- **Property:** gives further information about the activity, such as if it is a single or multiple choice. This property narrows the translation into graphical component in the PSM meta model.
- **TaskType:** enumeration that lists the basic types that an activity could belong to

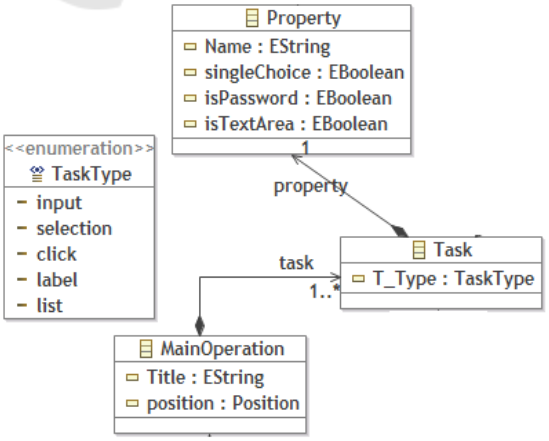

Figure 1: Proposed meta elements for modeling the graphical user interface.

When creating a model as an instance of the proposed meta-model, the graphical interface is hierarchized. Indeed, in the first row, we find the use case with the main functionality of the application. This functionality is divided into one or more major operations, each of which has atomic activities that describe user actions produced by the user towards graphic components.

In order to complete the meta-model, we have added elements to include the choice of the user in terms of positions in the interface. In fact, these elements will be used in the transformation phase to choose the most suitable layout.

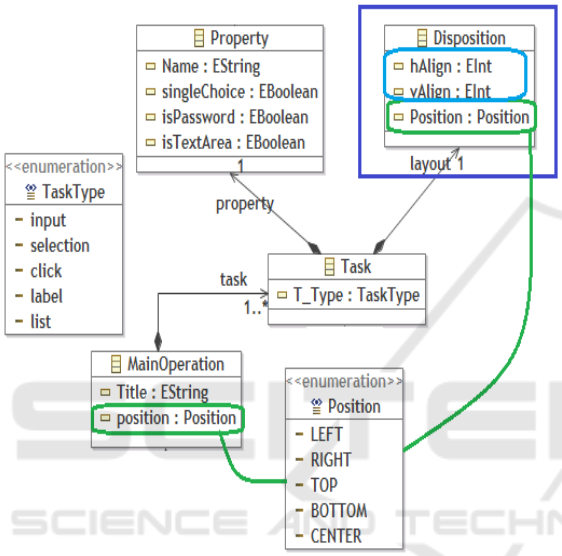

Figure 2: Position elements for the main operation.

First, we added an enumeration with the five positions as explained bellow. Each major operation is placed in the right position.

Second we added the meta element Disposition that describes the vertical and horizontal location besides the relative position itself. The idea is to divide the interface into boxes of a grid and this activity is localized by a peer (V, H) which allows associating the position in the most appropriate layout manager.

3.2 Proposed Process for Component's Position

For the proposed approach, we thought of having two levels of layout in dependence with the two elements MainOperation and Task of the proposed metamodel.

First, the MainOperation is positioned relative to the entire Interface. To do this, we divided the interface into five regions, Top, Bottom, Centre, Left and Right. Each major operation will be placed in one position depending on the user wishes which will be added to the input model. Later with the transformation engine, the whole interface will take a container as a grid with the five positions.

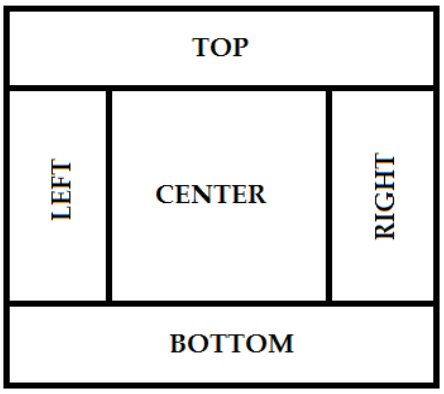

Figure 3: The five region in the graphical Interface.

Second, since each operation contains one or several tasks, these tasks need to be placed in their container properly. At this level, the mainOperation will be the container for the gathered tasks. That's why we integrated the pair (V, H) for each atomic task performed by the user to describe the vertical and horizontal positions. Again, in the transformation process, we will take into consideration these information and choose the proper layout and place the generated component in the right position. These elements are integrated in the input model as described in Fig.4. They are added as properties to both MainOperation and Task elements.

We can say that we still are independent from any platform since we did not use any technical detail or specific layout manager at this stage. This helps the user to define one model that can be used for several platforms and technologies.

These elements are independent from any specific platform or meta modeling elements. So we can add them as an extension to the IFML also, since it is an extensible Meta Model. Indeed, those properties can complete the extension presented in (Roubi et al, 2016) with the Field element extensions.

Moreover, we are working on a more advanced transformation algorithm to choose properly the right layout and expand the range of choices and not be limited to the three types already included. Indeed, we can add the constraints manager for the transformation algorithm without adding more

complex elements to the meta models themselves. Indeed, we should not have too complex Meta Models or related to a specific execution platform.

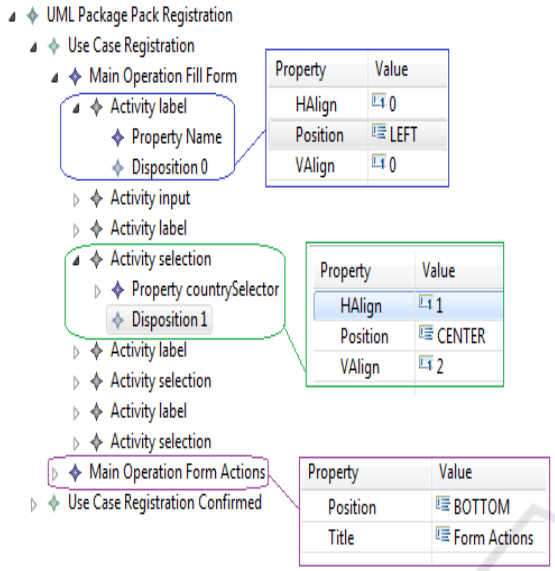

Figure 4: Model Instance of the proposed metamodel with proposed position properties.

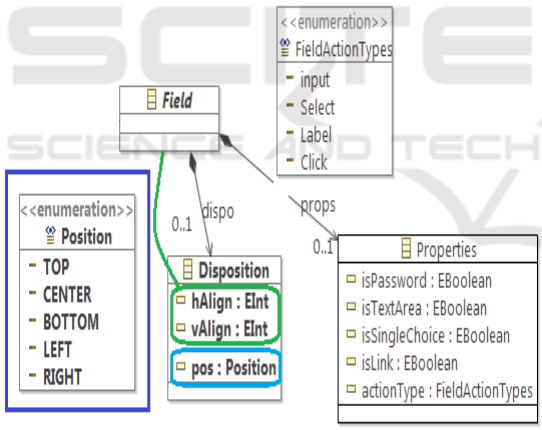

Figure 5: Extension of the IFML by the proposed position elements.

4 TRANSFORMATION PROCESS

In order to automate the whole process, we developed a Model To Model transformation engine that takes into account the elements added for the position of components. Indeed, we used Query View Transformation for the Model To Model Transformation process. We chose the Swing desktop application and Rich internet Applications.

The code excerpt below describes transformation from the input abstract elements to their corresponding ones respecting Rich Internet Application for both the position and the layout.

```
mapping
GUIMVP::Disposition::layoutToPosition(
) : JAVAFXMVP::CDispos {
    result.xPos :=  self.hAlign;
    result.yPos := self.vAlign;

    result.cellPos :=
self.Position.guiPositionToJavaFXPosit
ion();
}

query
GUIMVP::Position::guiPositionToJavaFXP
osition() : JAVAFXMVP::Position {
switch {
case(self=Position::LEFT)
{
   return JAVAFXMVP::Position::LEFT;
}
case(self=Position::RIGHT)
{
   return JAVAFXMVP::Position::RIGHT;
}
case(self=Position::CENTER)
{
   return JAVAFXMVP::Position::CENTER;
}
case(self=Position::TOP)
{
   return JAVAFXMVP::Position::TOP;
}
case(self=Position::BOTTOM)
{
   return JAVAFXMVP::Position::BOTTOM;
}};
```

Afterwards, with the Model To Text transformation, we consider the generated elements to choose the corresponding layout and positions for each component. The result of the generated source code :

```
<[theRoot.position.toString().toLowerC
ase()/]>
<[if(theRoot.Type=RootType::GridPane)]
GridPane[/if]>
<TextField id="[item.value/]"
text="[item.value/]"
GridPane.rowIndex="[widget.position.xP
os/]"
GridPane.columnIndex="[widget.position
.yPos/]"/>
```

5 CONCLUSIONS

In this paper, we presented an approach based on Model Driven Development to design and generate graphical user Interface. We focused on the graphical part of the application by adding new elements to the proposed Meta Model. These elements improve the resulting interface by taking into account the user's wishes for positions and Layouts. However, the algorithm behind transformation is limited to grid layout type. We should improve it and gather more complex layout manager and constraints.

The idea is to stay separated from a specific platform and keep the Meta Models as much simple as possible by reducing the elements and properties added.

REFERENCES

Berti, S., Correani, F., Mori, G., Paterno, F., and Santoro, C., 2004. Teresa: a transformation-based environment for designing and developing multidevice interfaces. *In CHI Extended Abstracts*, pages 793–794.

Brambilla, M. et al., 2014. Extending the Interaction Flow Modeling Language (IFML) for Model Driven Development of Mobile Applications Front End To cite this version : Extending the Interaction Flow Modeling Language (IFML) for Model Driven Development of Mobile Applications Front End.

Brambilla, M., Fraternali, P., et al, 2014. *The interaction flow modeling language (ifml), version 1.0. Technical report*, Object Management Group (OMG), http://www.ifml.org.

Ceri S,., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., and Matera, M., 2002. *Designing Data-Intensive Web Applications. The Morgan Kaufmann Series in Data Management Systems*. Morgan Kaufmann Publishers Inc.

Gmez, J., Cachero, C., Pastor, O., 2001. *Conceptual modeling of device-independent web applications*. pages 26–39.

Groenewegen, D., Zef Hemel, Lennart C. L. Kats, and Eelco Visser, 2008. Webdsl: a domain-specific language for dynamic web applications. *In Gail E. Harris, editor, OOPSLA Companion*, pages 779–780. ACM.

Linaje, M., Preciado, J.C., and Sanchez-Figueroa, F., 2007. A Method for Model Based Design of Rich Internet Application Interactive User Interfaces. *In Proceedings of International Conference on Web Engineering*, July 16-20, 2007, Como, Italy, pages 226–241.

Martinez-Ruiz, F.J., Arteaga, J.M., Vanderdonckt, J., and Gonzalez-Calleros, J.M., 2006. A first draft of a model-driven method for designing graphical user interfaces of Rich Internet Applications. *In LA-Web 06: Proceedings of the 4th Latin American Web Congress*, pages 3238. IEEE Computer Societ.

Paterno, F., Santoro, C., and Spano, L.D., 2009. Maria: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. *ACM Trans. Comput.-Hum. Interact.*, 16(4).

Roubi, S., Erramdani, M. and Mbarki, S. 2016, Extending graphical part of the interaction Flow Modeling Language to generate Rich Internet graphical user interfaces, *MODELSWARD 2016 - Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development*, pp. 161.

Schwabe, D. and Rossi, G., 1995. The object-oriented hypermedia design model. *Communications of the ACM*, 38(8), pp.45–46.

Schwabe, D., Rossi, G., 1995. *The object-oriented hypermedia design model*. pages 45–46.

Urbieta, M., Rossi, G., Ginzburg, J., and Schwabe, D., 2007. Designing the Interface of Rich Internet Applications. *In Proc. LA-WEB'07*, pages 144–153.

Vanderdonckt, J., 2005. A MDA-compliant environment for developing user interfaces of information systems. *In CAiSE*, pages 16–31.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS