Journal of Visual Languages and Computing 32 (2016) 1–34

Contents lists available at ScienceDirect

Journal of Visual Languages and Computing

journal homepage: www.elsevier.com/locate/jvlc

User interfaces metamodel based on graphs ☆

Paulo Roberto Lumertz [a,b,*], Leila Ribeiro [a], Lucio Mauro Duarte [a]

[a] Informatics Institute, Federal University of Rio Grande do Sul (UFRGS), Brazil
[b] Quantiza Tecnologia da Informação Ltda, Brazil

ARTICLE INFO

Article history:
Received 30 October 2014
Received in revised form
23 April 2015
Accepted 15 October 2015
Available online 17 November 2015

Keywords:
Metamodel
Graphs
Graph transformation
User interface
User interface patterns

ABSTRACT

Information systems are widely used in all business areas. These systems typically integrate a set of functionalities that implement business rules and maintain databases. Users interact with these systems and use these features through *user interfaces (UI)*. Each UI is usually composed of menus where the user can select the desired functionality, thus accessing a new UI that corresponds to the desired feature. Hence, a system normally contains multiple UIs. However, keeping consistency between these UIs of a system from a visual (organisation, component style, etc.) and behavioral perspective is usually difficult. This problem also appears in software production lines, where it would be desirable to have patterns to guide the construction and maintenance of UIs. One possible way of defining such patterns is to use *model-driven engineering (MDE)*. In MDE, models are defined at different levels, where the bottom level is called a *metamodel*. The metamodel determines the main characteristics of the models of the upper levels, serving as a guideline. Each new level must adhere to the rules defined by the lower levels. This way, if anything changes in a lower level, these changes are propagated to the levels above it. The goal of this work is to define and validate a metamodel that allows the modeling of UIs of software systems, thus allowing the definition of patterns of interface and supporting system evolution. To build this metamodel, we use a graph structure. This choice is due to the fact that a UI can be easily represented as a graph, where each UI component is a vertex and edges represent dependencies between these components. Moreover, graph theory provides support for a great number of operations and transformations that can be useful for UIs. The metamodel was defined based on the investigation of patterns that occur in UIs. We used a sample of information systems containing different types of UIs to obtain such patterns. To validate the metamodel, we built the complete UI models of one new system and of four existing real systems. This shows not only the expressive power of the metamodel, but also its versatility, since our validation was conducted using different types of systems (a desktop system, a web system, mobile system, and a multiplatform system). Moreover, it also demonstrated that the proposed approach can be used not only to build new models, but also to describe existing ones (by reverse engineering).

1. Introduction

Information systems are widely used in all business areas and users interact with them using *user interfaces (UI)*. These UIs are the operational units of the system where all features are made available to users. Normally, users browse through menus to select the one that contains the desired functionality. These systems can be developed for many different platforms, such as desktop, web, and mobile devices, and may also have different patterns of UIs. Customer business areas, where these systems are used, are always evolving, requiring that these systems be rapidly and easily adapted to these changes,

☆ This work was partially supported by FAPERGS and CNPq.
* Corresponding author.
E-mail addresses: paulo@lumertz.com (P.R. Lumertz), leila@inf.ufrgs.br (L. Ribeiro), lmduarte@inf.ufrgs.br (L.M. Duarte).

http://dx.doi.org/10.1016/j.jvlc.2015.10.026
1045-926X/

without losing consistency. It is also important to mention that computing power has grown in recent years for desktop computers as well as for mobile devices. This enables new forms of interaction and the tools for building UIs should take this into account to accommodate the possibility of new elements. Hence, the construction of UIs requires a high level of technical knowledge [27].

Developing and maintaining these UIs is not an easy task, and much effort has to be devoted to it. Companies that develop information systems to automate their clients' business have several challenges, such as: (a) maintaining these systems technologically up-to-date; (b) keeping consistency between UIs of the same system, both in appearance and behaviour, during their development and evolution; (c) keeping the system consistent even after changes in the development teams. New technology implies rewriting large amounts of code to ensure that all the features remain active and according to the original requirements. The modification of technical staff is another problem, since the documentation may be insufficient or out-of-date. A survey conducted by Myers and Rosson [24] concluded that the effort in the construction of UIs can reach almost 50% of the total effort to build a system. It has also been noticed that the design and implementation of UIs rarely have the same level of specification and modeling as the data and the functionalities [28], even though the construction of UIs is recognized as one of the most time-consuming steps of any process of software production.

Since the 1990s, much research has been developed in order to facilitate the creation of UIs. One example is the research on model-based user interface development (MBUID) [41], where several models were created with the goal of generating the UIs [40]. Many tools have been created for this purpose and Myers [25] introduced a classification for these tools. Extensions of UML with elements for modeling UIs were also proposed, creating the UMLi [5]. Although these approaches were based on simple ideas, the construction of declarative models was not trivial.

Most model-based tools use task, application, and presentation models to generate the executable UIs. This makes the generation of UIs more complex because developers have to deal with different models and this can be more time-consuming than directly programming the UIs. In addition, these models are constructed for each individual UI, which makes it difficult to maintain the consistency across all the UIs of the system.

Model Driven Engineering (MDE) [42] and *reusable components and patterns* [32] have been recently proposed to aid the development and evolution of software systems. The use of patterns is intended to allow the reuse of software, which would make it easier to maintain consistency [29]. Abstract user interfaces can be used to guide the design phase and serve as the basis for automatic code generation [20]. The use of MDE can reduce the effort to build UIs, enabling that they be automatically generated by model transformations. If the UIs are automatically generated, there can be an improvement in usability because they provide greater assurance of consistency of user experiences [26]. However, the practical use of this idea requires the definition of *metamodels*, which correspond to the most abstract descriptions of characteristics of a system. Based on these metamodels, it is possible to incrementally define more concrete models, gradually reaching the implementation level. Each new level of abstraction has to maintain all the characteristics of the more abstract ones. Thus, if the model of one level of abstraction is modified, the changes are propagated to the more concrete models, thereby keeping consistency.

In this work, we propose an approach to tackle the challenges described above. We studied the UIs of 20 information systems to identify patterns. These patterns were classified using UI standards accepted by the scientific community [47,11] and served as basis for a metamodel to build UIs. This metamodel is described as a graph [4,6], where the components of the UIs of a system are represented by vertices and their relationships are described by edges. The choice for a graph structure enables the use of existing tool support for the generation and analysis of UI models as well as to handle their evolution using graph transformations [2,8,19].

To validate the proposed metamodel we constructed models to represent the UIs of one new and four existing systems. These models were constructed from reusable components (patterns) using a modeling tool [10]. The models allow the analysis of UIs and the visualisation of their structure, thus providing a clear idea of their sizes and complexity. Although we currently do not support code generation from the UI models, this is technically possible and will be discussed in Sections 5.5 and 5.6.

This paper is organized as follows: Section 2 presents the basic concepts of users interfaces and graphs that are necessary to understand the rest of the paper. Section 3 presents the proposed UI metamodel, and Section 4 the validation of this metamodel. Section 5 discusses how to use the proposed metamodel to analyse the UIs of system and support their evolution. Section 6 describes related work in the area. Finally, Section 7 presents the conclusions and future directions.

2. Basic concepts

For this work, basic concepts about user interfaces and graph theory are required and are presented in this section.

2.1. User interfaces

A *user interface (UI)* is where a user operates an information system. An interface should provide a "friendly" experience, allowing the user to interact with the software features in a natural and intuitive way. In the first information systems, UIs were quite limited, consisting of textual interfaces. The great development of UIs to reach its current stage happened with the creation of graphical interfaces, which also increased the time and resources required to develop and execute such UIs. The first graphical UIs were created by *Xerox Corporation in Palo Alto Research Center* and the more recent are present in *Windows* 8 by Microsoft, *OS X* by Apple, and also the UIs of mobile devices like IOS, Android and others. All these UIs have common characteristics that were identified in Palo Alto, such as the concepts of windows, icons, menus, and pointers.