Tool-support for Pattern-based Generation of User Interfaces

Jürgen Engel
University of Rostock
Albert-Einstein-Strasse 21
18059 Rostock, Germany
+49 821 5586 3450
juergen.engel@uni-rostock.de

Christian Märtin
Augsburg University of Applied Sciences
Friedberger Straße 2a
86161 Augsburg, Germany
+49 821 5586 3450
christian.maertin@hs-augsburg.de

Peter Forbrig
University of Rostock
Albert-Einstein-Strasse 21
18059 Rostock, Germany
+49 381 498 7620
peter.forbrig@uni-rostock.de

ABSTRACT

This paper introduces the structural and functional architecture of a framework for pattern and model-based automated design of interactive systems. It discusses the various steps and the pattern and modeling resources used by the development process. The framework exploits a broader than usual view on software patterns and provides comprehensive tool support for designing generic and context-specific pattern languages containing pattern categories for all phases and abstraction levels of the life-cycle for the usage-centered design of user interfaces.

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques – *user interfaces*

General Terms

Design

Keywords

Interactive system, user interface, model-driven development, pattern-based development, HCI pattern languages, task-models, UI generation

1. INTRODUCTION

Results from research on model-based development approaches for interactive systems gradually find their way into design environments and tools for web-based applications and other highly interactive domains. In addition specific software engineering life-cycles have been tailored to support the growing demand for the targeted and cost-effective production of high quality interactive applications. These processes are based upon methods and techniques from model-driven and pattern-based development and automated user interface (UI) generation.

In order to maximize the overall benefit for developers and user interface designers of interactive systems we have combined these techniques within a toolset for pattern-based modeling and generation of interactive systems.

The following chapters discuss the functional and structural architecture of the framework and introduce the respective software life-cycle supporting the design of user interfaces.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
PEICS'10, June 20, 2010, Berlin, Germany
Copyright 2010 ACM 978-1-4503-0246-3…$10.00.

2. RELATED WORK

Patterns and pattern languages have a growing impact on the disciplines of HCI and web engineering. Originally pattern languages were constructed for providing standard solutions to architectural and urban planning problems [1]. Later patterns were adopted by software engineers and software architects for promoting the reuse of high-quality design solutions [5]. Recently software patterns have also entered the fields of HCI, usability engineering [7], user experience [10] and organizational workflow [6].

A variety of pattern catalogues and pattern languages have been developed for improving usability aspects and the quality of interface and interaction design, e.g. [9], [11]. Most of these pattern collections, however, lack an appropriate organizational structure in order to facilitate pattern selection and ensure the overall coverage of domain dependent and independent modeling and design problems. In [2] manageability aspects of various existing UI pattern catalogues are discussed and compared. In [8] a structured approach for designing hierarchically organized HCI pattern languages and controlling the selection of the really needed patterns during the software development process is described. This approach has extensively been tested and evaluated for new HCI pattern languages from different domains (online shops, industrial automation, database administration) and has been enriched by tools and components for semi-automated generation of real user interfaces from the selected patterns and for binding them to the application domain code [8].

Pattern-based software engineering has to cover the overall software development life-cycle from early planning down to coding, maintenance and usability evaluation activities. Several pattern-based approaches explore the detailed design aspects for getting from problem definitions to possible solutions, e.g. by using case-based reasoning techniques [12].

3. DEVELOPMENT PROCESS

Our toolset supports a designer or software engineer in planning, specifying and realizing concrete user interfaces. The general workflow of the development process is illustrated in fig. 1.

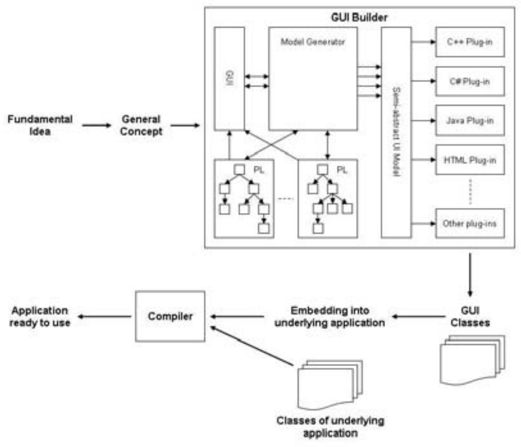


Figure 1. General development approach.

Of course, the process starts with a fundamental idea about which issues should be addressed by the prospective application. Before the toolset can be used a requirements analysis is carried out and a general conception has to be created. This enables the developer to select the appropriate high-level patterns to begin the actual top-down user interface design procedure. The system allows for navigating and browsing through previously constructed pattern languages and select dedicated patterns fitting best for resolving the current design problems. By this the hierarchical structures of the pattern languages are exploited and the more abstract patterns are refined by using more concrete definitions step by step. The work is completed as soon as the lowest level of abstraction has been reached, i.e. the UI element level. The result is a semi-abstract user interface model which is handed over to the code generation module.

The code generator is implemented in a way that different target languages can be supported by simply adding respective plug-in modules. It produces GUI-related source code files in the format of the corresponding target language, e.g. Java GUI classes. These can be used together with the classes representing business objects and business logic in order to compile and create the intended software application. The toolset concentrates on the user interface design while the development of the core business logic is not in focus.

4. PATTERN ADMINISTRATION

The pattern administration tool PLass (Pattern Language Assistant) allows the framework users to create new pattern languages (PL), and to add, change and delete patterns of existing pattern languages.

The generic idea underlying our HCI pattern language structuring approach is its layering in as many levels of abstraction as needed for covering the requirements of a given problem domain [8]. The lower in the hierarchy, the smaller is the contribution of a pattern to the overall problem domain and its solution space. Low-level patterns cover concrete problem/solution-pairs and may correspond to concrete interaction objects (CIOs), e.g. OK button, or primitive collections of CIOs with their respective relationships and dynamic links. They also may be associated with methods and code fragments that are needed for the functional parts of the application under development (e.g. embedded code for simple or expert search).

Higher-level HCI patterns not only may cover presentation aspects for complete application windows (e.g. community side, portal side etc.), but also describe complex mappings of domain content to appropriate UI solutions (e.g. the mapping of clustered information packages to different classes of multimedia-storytelling approaches). Sub-hierarchies then divide the complex patterns either into alternative or combinable smaller problem/solution chunks.

To find all available patterns for a given development task the application designer starts at the top and follows the hierarchy in a guided workflow that allows for selecting the necessary patterns within tree-sub-hierarchies or combine them through logical links with patterns from other tree-sub-hierarchies.

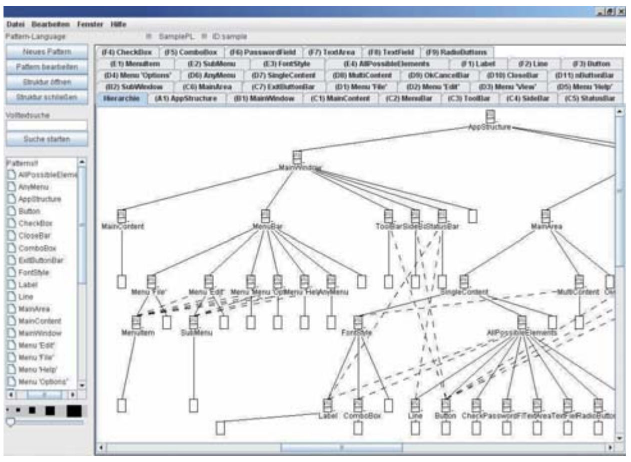


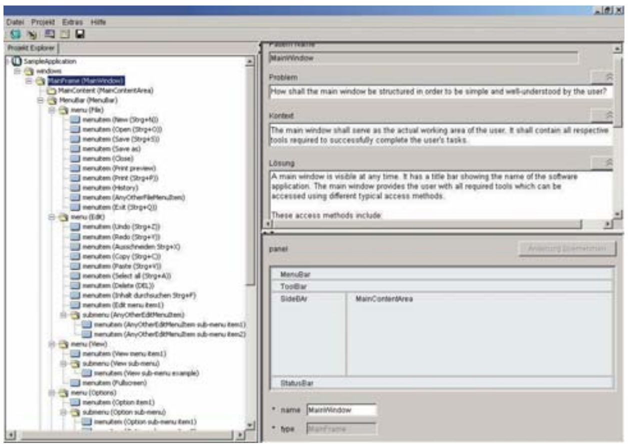
Figure 2. Pattern administration using PLass.

Fig. 2 shows a screenshot of the PLass user interface. Besides title bar and menu bar the upper window area consists of an additional heading bar indicating the name and unique identifier of the PL in progress. Located on the left side of the screen there is a tool and navigation bar providing shortcuts to frequently used administration functions as well as a list of all existing patterns belonging to the selected pattern language. Aside resides the editor panel showing a graphical overview of the pattern hierarchy and relationships. Selecting individual patterns from the list opens up the respective editor tabs providing details of the pattern structure and content. From here the patterns and their relations can be directly modified as required.

The specifications of the patterns of the various abstraction levels include the usual elements, such as pattern name, problem and solution descriptions, context information, and application examples. Additionally patterns consist of special elements incorporating information for user interface generation purposes which are described later in this paper. All relevant information is stored in a XML-compliant format.

5. USER INTERFACE MODELING

The application modeling tool PAgui (Pattern Aided Graphical User Interfaces) provides functions for creating new application model projects and to manipulate existing ones. Application models are based on a previously selected pattern language and might be linked to an individual code generator for preview reasons. Fig. 3 shows the PAgui model of a sample application based on a sample pattern language for general application structures. It is linked to the code generator for *Java 1.6 Swing*.



MainWindow

How shall the main window be structured in order to be simple and well-understood by the user?

The main window shall serve as the actual working area of the user. It shall contain all respective tools required to successfully complete the user's tasks.

A main window is visible at any time. It has a title bar showing the name of the software application. The main window provides the user with all required tools which can be accessed using different typical access methods.

These access methods include:

Figure 3. User interface modeling using PAgui.

Below the title bar on top of the modeling component window, the user is provided with a menu bar and an icon bar which allow access to the modeling and generation functionality. The panel on the left side shows the current model in a tree structure format. When clicking on any of the nodes or leaves within this tree structure, all available details of the related pattern are made visible in the upper right area of the screen. If a code generator is linked to the model, the panel below shows a generated preview of the future application as well as editable fields for all possible parameters, such as values, position and size. At the bottom of the window there is a status bar indicating the current state of the system.

When right-clicking nodes of the tree structure, the framework user is enabled to add child elements or even complete sub-tree structures. The system checks the PL definitions and provides a list of all possible successor elements. The user might select and add any of these items and configure the relevant settings. Subsequently the code generation starts automatically and the preview panel is updated accordingly.

6. CODE GENERATION

For user interface generation purposes the pattern definitions are enhanced by the previously mentioned additional special component named <automation>. This description element contains specifications in XML format and is mission-critical for the generation procedures and must not contain erroneous content. Therefore these parts of the pattern definitions are validated using a specific document type definition (DTD).

The <automation> component consists of two mandatory elements, <code> and <children>. While <code> specifies the later appearance of the respective artifacts, the <children> element references to potential child elements. The following code listing of the PAgui DTD gives an overview of the <automation> modeling options. A more detailed description of these capabilities is provided in [3].

```
<!ELEMENT automation (code, children)>
  <!ELEMENT code (element, attribute*, value?)>
    <!ELEMENT element (#PCDATA)>
    <!ELEMENT attribute (#PCDATA)>
      <!ATTLIST attribute name CDATA #REQUIRED>
      <!ATTLIST attribute required (true|false) #IMPLIED>
      <!ATTLIST attribute range CDATA #IMPLIED>
      <!ATTLIST attribute option CDATA #IMPLIED>
      <!ATTLIST attribute fixed (true|false) #IMPLIED>
    <!ELEMENT value (#PCDATA)>
      <!ATTLIST value required (true|false) #IMPLIED>
  <!ELEMENT children (child*)>
    <!ELEMENT child (#PCDATA)>
      <!ATTLIST child ref CDATA #REQUIRED>
      <!ATTLIST child min CDATA #IMPLIED>
      <!ATTLIST child max CDATA #IMPLIED>
      <!ATTLIST child choose CDATA #IMPLIED>
      <!ATTLIST child priority CDATA #IMPLIED>
      <!ATTLIST child relation CDATA #IMPLIED>
      <!ATTLIST child anchor CDATA #IMPLIED>
```

In the following we demonstrate how an exemplary user dialogue can be modeled and generated by using the above described functions of the toolset. According to the 'Registration' pattern [4] a 'User Registration' dialog window is produced providing the user with the possibility to enter personal data, to specify a login name and a password and to opt for a monthly newsletter subscription. Having filled the related fields the user may decide either to submit the data or to abort the process.

The following figure illustrates the pattern hierarchy used for the realization of the exemplary 'User Registration' dialogue.

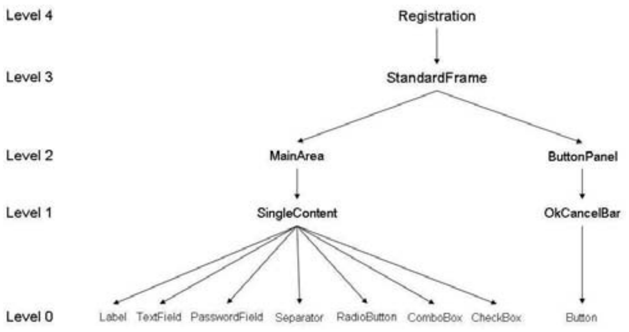
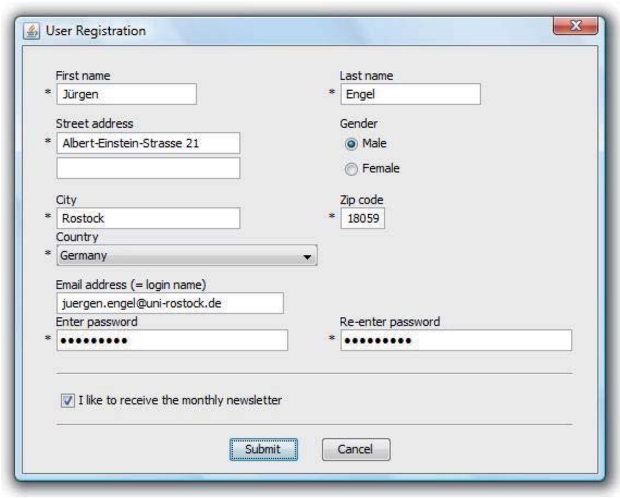


Figure 4. Pattern hierarchy used for 'User Registration' dialog.

In the first step we select the required patterns on the various description levels. All patterns and their relationships follow the rules and structures as described in [8]. On the top level the 'Registration' pattern translates into a 'StandardFrame' pattern which in turn references two further child patterns, the 'Main Area' and 'Button Panel' pattern. The 'Main Area' pattern links to the 'SingleContent' pattern which in turn references several child patterns of the lowest level, i.e. 'Label', 'TextField', 'PasswordField', 'Separator', 'RadioButton', 'ComboBox', and 'CheckBox' components. The 'Button Panel' pattern defines the

'OkCancelPanel' as child pattern which in turn references the 'Button' pattern.

The most relevant information required for the generation process is covered within the <automation> description elements of the chosen patterns. A detailed introduction to the code generation mechanisms is provided in [3]. When using the generator plug-in for Java Swing the resulting user dialog looks like as illustrated in fig. 5.


User Registration

First name * Jürgen	Last name * Engel
Street address * Albert-Einstein-Strasse 21	Gender [● Male] ○ Female
City * Rostock	Zip code * 18059
Country * Germany	
Email address (= login name) juergen.engel@uni-rostock.de	
Enter password * •••••••••	Re-enter password * •••••••••

[✓] I like to receive the monthly newsletter

Figure 5. 'User Registration' dialog as generated by the Java Swing plug-in on Windows Vista

7. CONCLUSION

The toolset introduced within this paper is used for development and maintenance of pattern languages related to several problem domains including online shops, industrial automation, and database administration. Further it facilitates our research efforts towards testing the potentials and limits of automated software generation.

In one of our current major projects the toolset is extended towards a complete framework in order to seamlessly support the work of user interface designers and developers in the arena of knowledge management and communication. It aims for increasing productivity and effectiveness of interactive system designers by combining several well-accepted and already very valuable software development techniques, model-based (i.e. task models, user models, device model, and environment models) and pattern-based development, automated UI generation, and usability evaluation. It integrates a pattern repository which serves as a structured reservoir of patterns organized in different pattern languages which can be used for generating interactive software applications for a growing number of problem domains. On the other hand the toolset supports our work on generating, evaluating, and comparing alternative designs of interactive applications within the same problem domain and consistently running them on different device platforms.

8. ACKNOWLEDGMENTS

The authors thank Alexander Roski for his major contributions to the tools and the hierarchical pattern language structuring approach.

9. REFERENCES

[1] Alexander C, Ishikawa S, Silverstein M. A pattern language (1977), Oxford University Press.

[2] Deng J., Kemp E., Todd E.G. Managing UI Pattern Collections. *Proc. CHINZ ´05, Auckland, New Zealand*, ACM Press (2005), 31-38.

[3] Engel J., Märtin C. PaMGIS: A Framework for Pattern-based Modeling and Generation of Interactive Systems. *Proceedings of HCI International 2009*. San Diego, USA

[4] Welie.com, Patterns in Interaction Design, available at http://www.welie.com/patterns/ [10 Mar 2010]

[5] Gamma E. et al. Design Patterns. Elements of Reusable Object-Oriented Software (1995), Addison-Wesley

[6] Guerrero Garcia J. et al. Towards a Library of Workflow User Interface Patterns. In *Proc. DSVIS 2008*, Springer LNCS 5136, 96-101.

[7] Marcus A. Patterns within Patterns, *Interactions, vol. 11., Issue 2* (2004), 28-34.

[8] Märtin C., Roski A. Structurally Supported Design of HCI Pattern Languages, Jacko, J. (Ed.) *Human-Computer Interaction, Part I, HCII 2007*, Springer LNCS 4550 (2007), 1159-1167.

[9] Tidwell J. Interaction Design Patterns. In *Proceedings of the Pattern Languages of Programming PLoP98.*

[10] Tiedtke T., Krach T., Märtin C. Multi-Level Patterns for the Planes of User Experience. In *Proc. of HCI International*, Las Vegas, USA, Vol. 4, (2005)

[11] van Welie M., Traetteberg H. Interaction Patterns in User Interfaces. *In 7th Pattern Languages of Programs Conference*, Allerton Park Monticello, USA (2000).

[12] Wentzlaff I., Specker M. Pattern-based Development of User-friendly Web Applications, *Proc. ICWE ´06 Workshops, Palo Alto, July 10-14* (2006), ACM Press.