Applying UI patterns for modeling dialogs

Mathias Kühn and Peter Forbrig

University of Rostock
Department of computer science
Albert-Einstein-Str. 22
18051 Rostock, Germany

{mathias.kuehn, peter.forbrig}@uni-rostock.de

Abstract. Software designs for use on interactive devices can be specified with models. Model-based languages allow layout specifications of UIs on different levels of abstraction. Specifications of dialogs can for instance be made with statecharts. Languages that allow specifications based on statecharts need to be adapted for use together with UI layout models. UI patterns can be applied to user interface layout and behavior designs. Corresponding specifications based on UI models can be created and changed with editors for use with platform-specific interpreters afterwards. The paper focusses the effects of UI pattern application on model-based specifications.

Keywords: user interface modeling, dialog models, user interface patterns

1 Introduction

User interface (UI) models as platform-independent as well as dependent specifications of interactive software surfaces can be used for runtime interpretations based on operations of model-driven engineering. According to the Cameleon Reference Framework (1), these operations are context-dependent transformations that result in final user interface models for platform-specific use. Beside structures, also UI behavior needs to be specified that refers dialog models (7) directly. UI patterns (8,9) can be applied for specifying models that describe how parts of the UI are presented to the user. As result, models can be created as well as changed just by applying patterns (2). In the end, manipulated specifications can instantly be used by interpreters that consider models at runtime. Users directly can see the effects under their platform. However, UI patterns can reduce the effort for designing models. The following sections focus on models for user interface layout and behavior together with UI pattern application. A survey example gives an overview on how to apply the proposed approach. The effects of UI patterns at design- and run-time are shown on specifications together with generated UI instances.

2 Related work

Models for user interfaces can be specified on different levels of abstraction. Languages like UsiXML (5) and MARIA (6) allow abstract and concrete UI specifications. According to the Cameleon Reference Framework (1), task models, abstract-, concrete-, and final-UI models can be transformed into each other. The transformation types are abstractions (concrete to abstract models), reifications (abstract to concrete models), and translations (models into other contexts). However, the framework allows generating UIs depending on various contexts of use, which are specifications of users, platforms, and environments.

Behavior of software can be specified with statecharts (3). Languages like SCXML (4) and XMI (10) allow specifications that consider states, what could be settings of UIs, and transitions, what could be UI events triggered by users. Regarding the Cameleon Reference Framework, statechart specifications also need to be considered on different levels of abstraction. This results in platform-specific abstractions and reifications of events. Statecharts explicitly do not consider the creation of objects. Following this, mentioned languages need to be extended by specific transitions for creating UIs at runtime (7).

Considering dialogs, designers for instance are interested in specifying how users can navigate within the UI. Tidwell (8) and van Welie (9) identified UI patterns for navigation. Some of them are stepwise, hub-and-spoke, pyramid, fully-connected, and multi-level. UI dialogs that are specified with statecharts can be manipulated with transitions and states depending on certain patterns. These automatically can be added to and removed from the statechart.

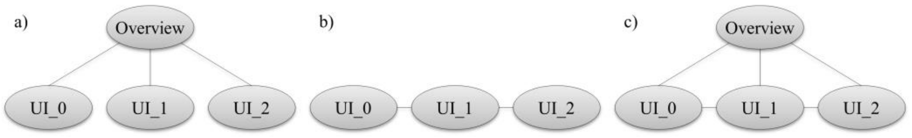


Fig. 1. Visualization of navigation structures for the UI patterns a) hub-and-spoke, b) stepwise, and c) pyramid

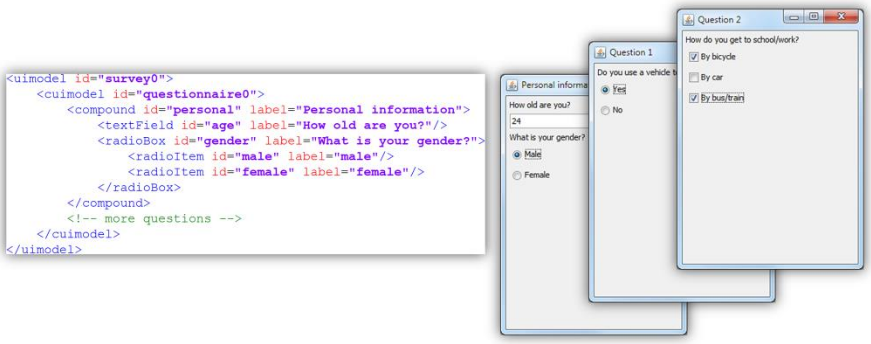
Fig. 1 shows a visualization of some UI patterns for navigation. Ellipses represent user interfaces that can be displayed at runtime. Lines between them represent paths that users can navigate through. Of course, lines also can be directed, but it depends on the context of UI pattern application. However, lines are related to transitions of corresponding dialog specifications.

UI layouts also can be manipulated depending on patterns. For instance, interactors for navigation automatically can be added to specifications. However, design patterns (2) in general are proven solutions for common design problems. Also UI patterns help in finding solutions for UI design problems, what can be related to layout- and dialog-structures, respectively.

3 UI patterns for dialog modeling

Patterns for user interfaces help not only to find well established solutions for recurring UI design problems but also to manipulate existing solutions for implementing changed requirements. For instance, UI patterns for navigation could be used for generating dialog specifications. According to this, changing applied patterns would result in adapting generated specifications. However, UI pattern applications consider needs not only of designers but also of users. For instance, UI patterns for layout can make UIs clearer and more usable in the end.

The following example gives an overview on how to apply UI patterns for dialog specifications of interactive software systems. The example is a survey on using vehicles in everyday life. Participants are asked about personal information (e.g. gender, age) and their preferred vehicles for going to school or to work (e.g. bicycle, car, bus/train). For this reason, an interactive form is prepared that can be used as questionnaire of the survey application. Participants are guided through the questionnaire and can answer questions step by step. Layout and dialog specifications are needed for interactive forms that can be generated on various devices afterwards.



```
<uimodel id="survey0">
    <cuimodel id="questionnaire0">
        <compound id="personal" label="Personal information">
            <textField id="age" label="How old are you?"/>
            <radioBox id="gender" label="What is your gender?">
                <radioItem id="male" label="male"/>
                <radioItem id="female" label="female"/>
            </radioBox>
        </compound>
        <!-- more questions -->
    </cuimodel>
</uimodel>
```

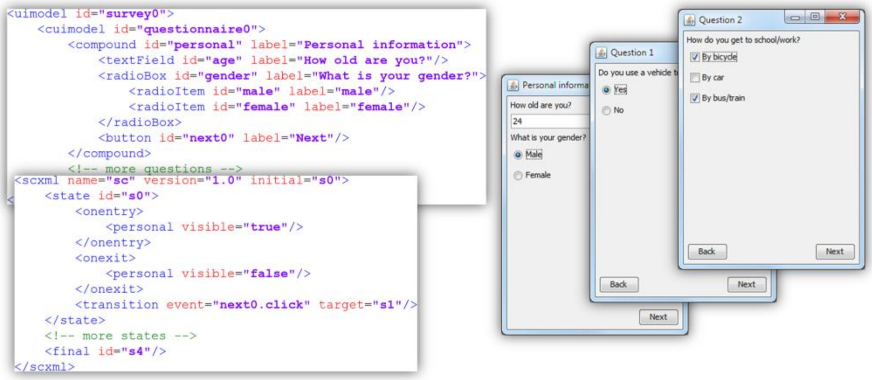
Personal informa
How old are you?
24
What is your gender?
[✓ Male] Female

Question 1
Do you use a vehicle t
[✓ Yes] No

Question 2
How do you get to school/work?
[✓ By bicycle]
By car
[✓ By bus/train]

Fig. 2. Layout specification in UsiXML together with some generated UIs

Fig. 2 shows the first part of the layout specification of questionnaires as interactive forms. These are specified as concrete UI model in UsiXML. Beside the specification are some generated user interfaces. It is assumed that tool support is available that allows designing forms in UsiXML. After designing UI layouts, designers can apply UI patterns for navigation. For example, applying the UI pattern hub-and-spoke would result in generating a distinct UI that is some kind of overview. This overview holds links to the generated UIs. Additionally, all specified UIs would also be added with links for navigating to the overview UI. However, another example would be to apply the UI pattern stepwise instead of hub-and-spoke. The following figure shows instances of the applied pattern.


```
<uimodel id="survey0">
    <cuimodel id="questionnaire0">
        <compound id="personal" label="Personal information">
            <textField id="age" label="How old are you?"/>
            <radioBox id="gender" label="What is your gender?">
                <radioItem id="male" label="male"/>
                <radioItem id="female" label="female"/>
            </radioBox>
            <button id="next0" label="Next"/>
        </compound>
        <!-- more questions -->
```

```
<scxml name="sc" version="1.0" initial="s0">
    <state id="s0">
        <onentry>
            <personal visible="true"/>
        </onentry>
        <onexit>
            <personal visible="false"/>
        </onexit>
        <transition event="next0.click" target="s1"/>
    </state>
    <!-- more states -->
    <final id="s4"/>
</scxml>
```

Fig. 3. Layout/dialog specifications in UsiXML/SCXML together with some generated UIs

Fig. 3 shows the first part of the layout specification of Fig. 2 after applying the UI pattern stepwise. A button `next0` is added for navigating to the next UI. Additionally, the dialog specification is generated that is based on SCXML. Added buttons allow navigating between the questionnaire forms. Transitions consider events triggered by added buttons. Participants interactively can go through the questionnaire forms by using the *Back* and *Next* buttons. However, changing the navigation structure can be made by applying other patterns.

4 Summary and outlook

User interface patterns can be used for designing specifications based on UI models for layout and behavior. The example of a survey on vehicle usage showed how to apply patterns for generating dialogs. Pattern application generates transitions within statechart specifications that are used as dialog of the user interface. Also the UI layout can be manipulated according to the related pattern. The proposed approach uses the model-based languages UsiXML, which can be used for layout specifications of UIs, and SCXML, which can be used for statechart-based specifications. The example showed how to apply the UI pattern stepwise within specifications based on the approach.

Future work need to be done on giving tool support for UI pattern application. For instance, designers can create forms with a common UI designer tool and apply different patterns just by selecting them from a list. Specifications are generated and manipulated accordingly and can instantly be used with platform-specific interpreters on user-related devices. However, model-based languages benefit generating UIs depending on a specific context, what is a goal of corresponding applications.

References

1. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., and Vanderdonckt, J.: A unifying reference framework for multi-target user interfaces. In: Interacting with computers, vol. 15, pp. 289-308 (2003)
2. Gamma, E., Helm, R., Johnson, R., and Vlissides, J.: Design Patterns. Elements of Reusable Object-Oriented Software, 1st ed. Prentice Hall (1994)
3. Harel, D.: A Visual Formalism for Complex Systems. In: Science of Computer Programming 8, vol. 3, pp. 231-274 (1987)
4. Kistner, G., and Nuernberger, C.: Developing User Interfaces using SCXML Statecharts. In: Proceedings of EICS 14 Workshop, pp. 5-11 (2014)
5. Limbourg, Q., and Vanderdonckt, J.: UsiXML: A User Interface Description Language Supporting Multiple Levels of Independence. In: Proceedings of ICWE 2004 Workshop, pp. 339-352 (2004)
6. Paternò, F., Santoro, C., and Spano, L. D.: MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. In: Transactions on Computer-Human Interaction, vol. 16, 4 (2009)
7. Schlungbaum, E., and Elwert, T.: Dialogue graphs: a formal and visual specification technique for dialogue modelling. In: BCS-FACS, pp. 13 (1996)
8. Tidwell, J.: Designing Interfaces, 2nd ed. O’Reilly Media (2010)
9. van Welie, M.: Interaction Design Pattern Library. http://www.welie.com/patterns/
10. XML Metadata Interchange (XMI): http:// www.omg.org/spec/XMI/