Automated Optimization of User Interfaces for Screens with Limited Resolution

Sevan Kavaldjian, David Raneburger, Roman Popp, Michael Leitner, Jürgen Falb, Hermann Kaindl
Institute of Computer Technology
Vienna University of Technology, Austria
{kavaldjian, raneburger, popp, leitner, falb, kaindl}@ict.tuwien.ac.at

ABSTRACT

More and more devices with different screen resolutions are used to run the same application. In order to reduce usability problems, user interfaces (UIs) specific to resolution are needed, but it is time consuming and costly to implement all the different UIs manually. Automated generation of UIs has the potential to reduce time and costs in case of many such devices. Still, model-driven generation of UIs may not be flexible enough to include optimization for various resolutions.

We extended the straight-forward approach to model-driven generation by including optimization according to maximum usage of available space for a given resolution, minimum amount of clicks, and minimum scrolling. For these optimizations, we also use automated layouting and calculate the space needs of the possible variants. In effect, our new approach generates UIs optimized for screens with limited resolution, in order to reduce related usability problems.

INTRODUCTION

Automated generation of UIs has certainly advanced in recent years, especially based on model-driven approaches. Still, such generated UIs pose many usability problems. We think that this is partly due to insufficient flexibility of the current generation approaches.

In particular, straight-forward model-driven generation only allows for matching a single transformation rule for each source pattern. We extend this approach by taking up means from rule-based programming, that have been around for a long time. We allow matching of several transformation rules for any source pattern, and we use so-called conflict resolution to determine which rule to apply (fire). Based on that, we implement a simple form of optimization in the context of model-driven UI generation.

It allows us to maximize the amount of information to be displayed on a screen with limited resolution, to minimize the number of navigation clicks, and to minimize scrolling. All this is important for reducing usability problems. Since more and more devices with different screen resolutions are used to run the same application, we can automatically optimize the generated UI for the given (limited) resolution.

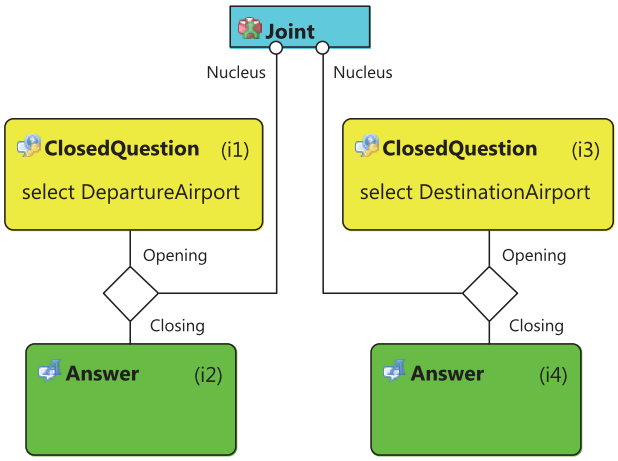


Figure 1. A discourse model excerpt

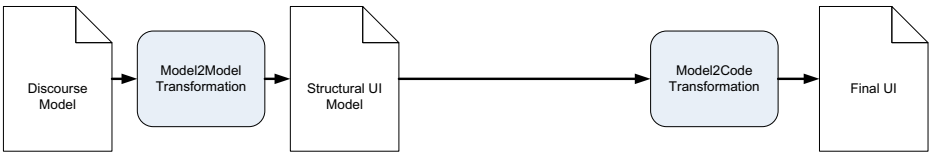
BACKGROUND

The input for our UI generation approach is a discourse model [2]. Such a discourse model serves as an interaction design on a high level of abstraction based on concepts of human language theories. A small excerpt of a larger discourse model for flight booking is shown in Figure 1. We use this discourse model as a running example throughout the remainder of this paper.

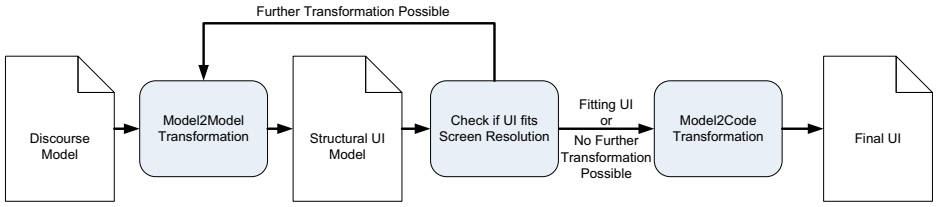
Our Discourse Models

The main ingredients of our discourse models are communicative acts derived from speech acts [7]. A communicative act is represented as a rounded rectangle and models an utterance of one of the communication partners. In our example, the application asks the customer *closed question*s, while the customer provides *answer*s to the questions. In our example, the yellow (or light gray) communicative acts are uttered by the application and the green (or dark gray) ones are uttered by the customer.

Pre-proceedings of the 5th International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI 2010): Bridging between User Experience and UI Engineering, organized at the 28th ACM Conference on Human Factors in Computing Systems (CHI 2010), Atlanta, Georgia, USA, April 10, 2010.



(a) Our Basic Transformation Process



(b) Our Extended Transformation Process

Figure 2. Our Transformation Process

Additionally, some communicative acts, like *Question* and *Answer* form a so-called adjacency pair which is represented by a diamond in our discourse models and defines the turn-taking and thus the sequence of utterances.

Adjacency pairs can be related with each other to build a tree structure. In our example, two Question–Answer pairs are related by a *Joint* relation. This *Joint* relation states that the Question–Answer pairs in both nucleus branches are of equal importance. Further, it does not imply a temporal order per se. For instance, both pieces of information can be presented in parallel if there is enough space on the screen. Otherwise they can be uttered in sequence.

Our Basic Transformation Process

We have developed a user interface generation process [5] that transforms such discourse models into WIMP-based graphical user interfaces (Windows, Icons, Menu and Pointers). Our basic user interface generation process is illustrated in Figure 2(a) and consists of two steps.

The first step transforms a discourse model into a Structural UI Model [4] by applying transformation rules to discourse model elements. The resulting Structural UI Model represents the user interface's widgets and their structure, but still abstracts from details of the final UI. We do not use a common UI description language (e.g. UsiXML[1]) because our runtime environment is based on the exchange of Communicative Acts. In our running example, the following transformation rules are applied to elements of the discourse model excerpt in Figure 1 for generating a model representing the structure of the final UI in Figure 3(a).

First, a *Joint Rule* gets applied that matches the Joint relation and adds a panel to the Structural UI Model. This panel acts as a container for the Radio Button lists in Figure 3(a), which correspond to the two nucleus branches of the Joint relation.

Second, a *Closed Question Rule* gets applied twice that matches each of the two Question–Answer adjacency pairs. For each adjacency pair a panel containing a label for a heading, a list of radio buttons together with item labels, and a submit button on the bottom is added to the Structural UI Model.

In the second step this Structural UI Model is used to generate source code for a particular target platform, e.g., Java Swing in our running example.

[1] http://www.usixml.org

OPTIMIZED RENDERING FOR LIMITED RESOLUTIONS

The problem tackled by our extended approach is to fit a given amount of information optimally (in the following sense) into screens with limited resolution.

Optimization Objectives and Approach

We assume that the following optimization objectives improve the usability of the generated user interfaces:

- maximum use of the available space for the given resolution,
- minimum amount of navigation clicks, and
- minimum scrolling (except list widgets).

Whenever the given information to be displayed does not fit into a single screen with default widgets, we try to display it with widgets that use less space. If it still does not fit into a single screen, then we split its display to two or more screens. Splitting increases the number of navigation clicks but it minimizes scrolling. We exclude list widgets from this last optimization objective because the number of list entries can vary extremely at runtime and determines whether the list is scrollable or not. This information is not known during our rendering process.

Our Extended Transformation Process

Our basic transformation process looks like straight-forward model-driven generation that only allows for matching a single transformation rule for each source pattern. We are not

aware of any optimization strategy in such a context. Therefore, we extend the straight-forward approach by allowing that several transformation rules may match for each source pattern, and by applying so-called conflict resolution to select which rule to apply (fire) in the next model-to-model transformation. Our extended generation process shown in Figure 2(b) illustrates the resulting possibility of trying out several rules for optimization purposes. In this approach, the rules need not to be specifically designed for a particular screen resolution. It is rather the way the rules are applied that achieves the given optimization objectives.

In order to implement such an optimization, the conflict resolution mechanism needs to select the rules in a certain defined order. For achieving the optimization objectives given above, this selection order is according to the space that the widgets the rule creates occupy in the final UI. Therefore, all rules matching the same discourse element for transformation have to be ranked by the designer according to this space need.

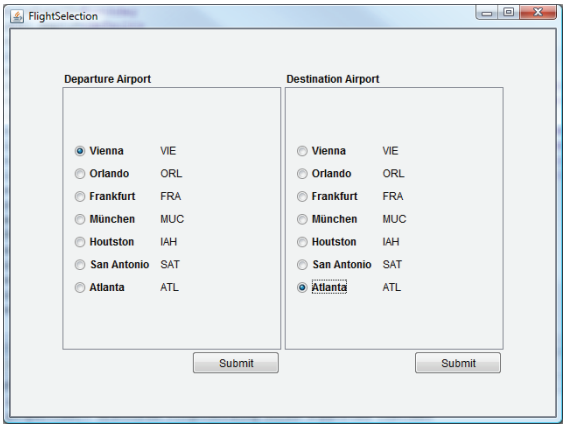
Each target device we render for has an abstract device specification that contains all style data used by the transformation rules. These data specify default sizes for all input and output widgets on the target device that can be overwritten in a transformation rule. They are used to set the size for each final UI element and allow us to calculate the exact size of each container (e.g., panel). For example, we set the size of the list widget explicitly. This makes it independent from the number of entries. If the list widget is not able to display all entries, it becomes scrollable.

After the size calculation we try to layout each generated screen to fit into the given resolution. However, we modify only the arrangement of the widgets that has not been fixed explicitly in a transformation rule. Therefore, we do not change the layout specified by the *Closed Question Rule* (i.e., the layout of the heading label, the radio button list and the submit button in Figure 3(a)). In this example, we modify the position of the complete radio button lists in the panel created by the Joint relation, since the *Joint Rule* does not contain any layout information.

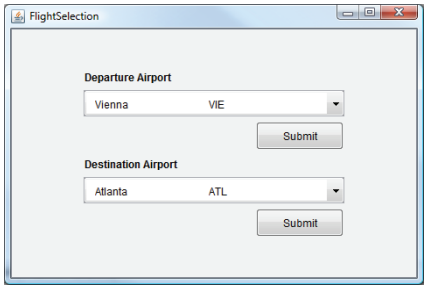
Now let us explain how to apply this approach to automatically generate user interfaces for three target resolutions. As input we use the discourse model excerpt shown in Figure 1.

Our first GUI is rendered for the resolution 640×480. The first cycle of the model-to-model transformation uses the highest ranked rules (i.e., the ones with the highest space need) for each discourse element. These are the same rules that have been applied in our basic transformation process. After the first transformation cycle we calculate the size for each panel in the corresponding Structural UI model. We can place them next to each other without exceeding the screen resolution. So, this is a fitting UI and we trigger the model-to-code transformation. The result is shown in Figure 3(a).

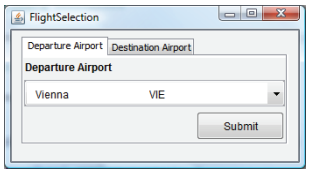
Next we generate a UI for the resolution 480×320. This time, the UI resulting from application of the highest ranked



(a) 640×480



(b) 480×320



(c) 320×180

Figure 3. Generated User Interfaces

rules does not fit. As long as a lower ranked rule can be applied, we initiate another generation cycle. First, the *Small Closed Question Rule* is used in our example. This rule matches the same source element (Question–Answer adjacency pair) as the *Closed Question Rule* but it creates a UI structure which occupies less space on the screen. A combo box element presents the content of the Closed Question communicative act to the user and a submit button is generated to confirm the selection of the user. The user interface shown in Figure 3(b) is the result of two more cycles, because in each cycle only one lower ranked rule is applied. The resulting UI fits and still presents the same information, but using widgets with less space needs (combo boxes instead of radio buttons). However, the list widgets do not fit next to each other and the layouter arranges them vertically.

In a third run, we generate a user interface for the resolution 320×180. Even after all rules for widget selection have been tried out, the generated GUI still does not fit the given resolution. Therefore, we start using rules that split the screen in order to increase the number of navigation clicks before

we make use of scrolling. In our example, this means that in the next cycle the *Small Joint Rule* is applied instead of the *Joint Rule*. The *Small Joint Rule* matches the same source element (Joint relation) but creates a different UI structure (a tabbed pane element instead of a panel). Figure 3(c) shows the outcome for the resolution 320×180. The *Small Joint Rule* and the *Small Closed Question Rule* have been applied and a fitting UI has been generated after a third cycle of rule application. This time no layout modifications are necessary because each tab contains only one panel.

The worst case in our extended generation process occurs if and when no more rules are available and the generated screen still does not fit the given screen resolution. In this case, we stop the optimization loop and rely on scrolling.

RELATED WORK

A transformation system that fits web pages automated and on-the-fly to screens of small devices is presented in [8]. The transformations are performed in order to minimize navigation and scrolling like in our approach. In contrast, however, this process alters an already existing UI.

Declarative user interface specifications are used as input for multi-target UI generation in [3]. The user interface adaption is treated as an optimization problem based on a user- and device-specific cost function. Compared to such user interface specifications, our interaction models are on a higher level of abstraction.

The model-driven approach for engineering multi-target UIs presented in [1] supports switching between predefined presentations during runtime. Our approach, in contrast, is intended to automatically generate presentations for different resolutions from a single discourse model.

An advanced approach for generating multi-device UIs is based on task models [6]. Such a Task Model specifies the temporal relations among tasks and has to be adapted according to the screen space available on the target device. Therefore, any optimization and screen splitting has to be done explicitly during the creation of the Task Model.

We are not aware of any other approach that performs optimization in the course of model transformations. Neither are we aware of any model-driven GUI transformation process that takes the resolution for transformation rule selection into account.

CONCLUSION

Our new and extended approach introduces an optimization technique into model-driven generation of UIs to reduce usability problems. However, we only deal with relatively simple usability aspects (minimum amount of clicks and minimum scrolling). We do not (yet) optimize layout according to, e.g., aesthetic criteria. Therefore, our optimization approach as presented above is not suitable for large screens with high resolution.

Overall we introduce a UI generation process that allows the same rule set to be used for generating UIs for devices with different resolutions. Through the automatic calculation of space need, it may even have an advantage in this respect as compared to a human interface designer. We implemented a simple optimization approach that allows us to optimize generated UIs for devices with limited resolution in such a way as to utilize the given space and to minimize navigation and scrolling. This should pave the way to optimized multidevice UI generation.

ACKNOWLEDGMENTS

This research has been carried out in the CommRob project (`http://www.commrob.eu`), partially funded by the EU (contract number IST-045441 under the 6th framework programme).

REFERENCES

1. B. Collignon, J. Vanderdonckt, and G. Calvary. Model-driven engineering of multi-target plastic user interfaces. In *Proceedings of the Fourth International Conference on Autonomic and Autonomous Systems (ICAS 2008)*, pages 7–14, Washington, DC, USA, 2008. IEEE Computer Society.

2. J. Falb, H. Kaindl, H. Horacek, C. Bogdan, R. Popp, and E. Arnautovic. A discourse model for interaction design based on theories of human communication. In *Extended Abstracts on Human Factors in Computing Systems (CHI '06)*, pages 754–759, New York, NY, USA, 2006. ACM Press.

3. K. Gajos and D. S. Weld. SUPPLE: Automatically generating user interfaces. In *Proceedings of the 9th International Conference on Intelligent User Interface (IUI '04)*, pages 93–100, New York, NY, USA, 2004. ACM Press.

4. S. Kavaldjian, C. Bogdan, J. Falb, and H. Kaindl. Transforming discourse models to structural user interface models. In *Models in Software Engineering, LNCS 5002*, volume 5002/2008, pages 77–88. Springer, Berlin / Heidelberg, 2008.

5. S. Kavaldjian, J. Falb, and H. Kaindl. Generating content presentation according to purpose. In *Proceedings of the 2009 IEEE International Conference on Systems, Man and Cybernetics (SMC2009)*, San Antonio, TX, USA, Oct. 2009.

6. F. Paternò, C. Santoro, and L. D. Spano. Model-based design of multi-device interactive applications based on web services. In *INTERACT (1)*, pages 892–905, 2009.

7. J. R. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, England, 1969.

8. X. Xiao, Q. Luo, D. Hong, H. Fu, X. Xie, and W.-Y. Ma. Browsing on small displays by transforming web pages into hierarchically structured subpages. *ACM Transactions on the Web*, 3(1):1–36, 2009.