Fully automatic content presentation specific to intentions

Sevan Kavaldjian, Jürgen Falb, Hermann Kaindl
Institute of Computer Technology
Vienna University of Technology, Austria
{kavaldjian, falb, kaindl}@ict.tuwien.ac.at

ABSTRACT

Programming graphical user interfaces is hard and expensive, while automatic generation is still quite challenging and faces even more usability problems. One of the issues involved in automatic generation is the presentation of content from the domain of discourse according to its purpose in the current state of the human-machine dialogue. We address this issue through including the intention of a given content presentation as indicated by communicative acts, and through generating it specifically according to the type of communicative act. This results in fully automatically generated user interfaces with content presentations specific to intentions.

INTRODUCTION

Manual creation of graphical user interfaces (GUIs) is hard and expensive, so we strive for automatic generation. Instead of generating them from simple abstractions, we let an interaction designer (and even end users) model discourses in the sense of dialogues (supported by a tool) [1]. From such a high-level declarative discourse model, we have been able to automatically generate the overall structure and the "look" of a GUI [3] as well as its behavior [8]. This automatic generation employs model transformations.

Still, we had to deal with the presentation of content of the domain of discourse. In particular, the concrete presentation needs to depend on the intention of the envisaged interaction, since the GUI's usability would clearly not be satisfactory otherwise. In this paper, we present our approach to automatic content generation specific to intentions, which also employs model transformations.

The remainder of this paper is organized in the following manner. First, we sketch our discourse models. Then we elaborate on the model transformations from such a discourse model to a structural GUI model including content presentation. Based on such a derived model, we explain automatic screen generation for the content presentation. Finally, we compare our approach with related work.

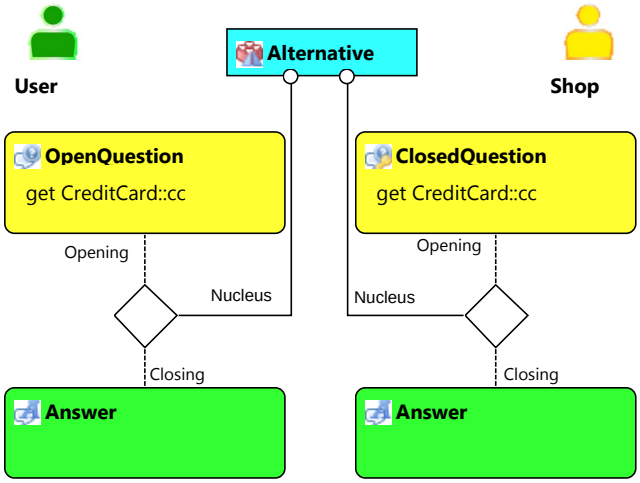


Figure 1. Excerpt of an online shop discourse model.

HIGH-LEVEL DISCOURSE MODELS

The starting point for our automatic GUI generation is a discourse model. According to [2], such a declarative discourse model has the following key ingredients:

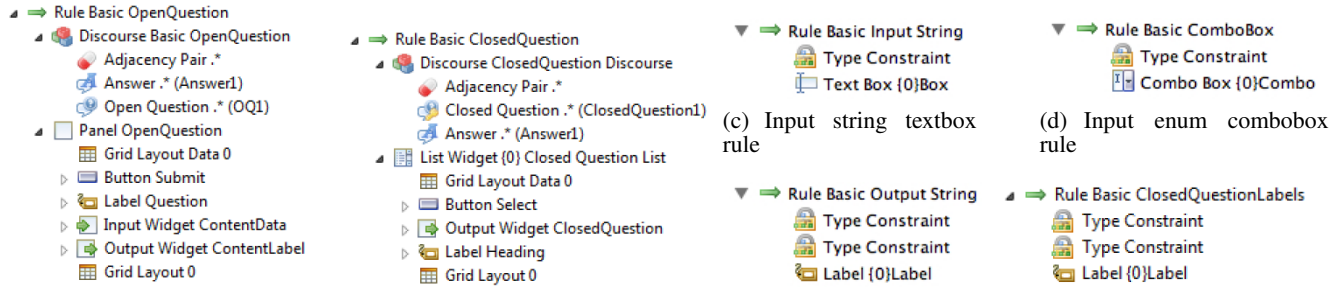
- *communicative acts* as derived from speech acts [10],
- *adjacency pairs* adopted from Conversation Analysis [4], and
- *RST relations* inherited from Rhetorical Structure Theory (RST) [5].

Communicative acts represent basic units of language communication. Thus, any communication can be seen as enacting of communicative acts, acts such as making statements, giving commands, asking questions and so on. Communicative acts indicate the intention of the interaction, e.g., asking a *question* or issuing a *request*. Figure 1 shows such examples in a small excerpt of a discourse model for a simple online shop, which specifies checkout with a credit card.

Communicative acts typically refer to propositional content. In this example, it is about getting the credit card information of the customer, see the text given below the type of communicative act. In fact, it is the *same* propositional content for both communicative acts. However, it should be presented differently, depending on the type of the communicative act, which indicates the intention of presenting this information.

CEUR-WS.org/Vol-439/paper6.pdf

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073



(a) Open question rule (b) Closed question rule (c) Input string textbox rule (d) Input enum combobox rule (e) Output open question label rule (f) Output closed question label rule

Figure 2. Transformation rules for open questions (a) and closed questions (b), for the *ContentData* input widget of the open question depending on the content type string (c) and enumeration (d), and rules for the output widgets of the open question (e) and the closed question (f).

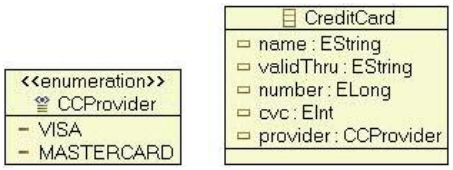


Figure 3. Excerpt of a domain of discourse model.

Propositional content is specified in our approach in a model of the domain of discourse. Figure 3 shows a very small excerpt of such a model in a UML class diagram.[1] It specifies a (logical) CreditCard with its five attributes.

Adjacency pairs are sequences of talk "turns" that are specific to human (oral) communication, e.g., a *question* should have a related *answer*. Figure 1 shows two examples of such adjacency pairs.

RST relations specify relationships among text portions and associated constraints and effects. The relationships in a text are organized in a tree structure, where the rhetorical relations are associated with non-leaf nodes, and text portions with leaf nodes. In our work we make use of RST for linking communicative acts and further structures made up of RST relations. Figure 1 shows an example of an *Alternative* RST relation linking two adjacency pairs.

MODEL TRANSFORMATION TO STRUCTURAL UI MODEL

Our model-driven approach transforms discourse models into structural UI models that are close to the final user interface but still GUI toolkit-independent. It is a process consisting of two interleaved transformation steps:

1. The first step applies rules that generate an overall UI structure based on patterns matched in the discourse model. These rules generate abstract widgets like labels for headings and placeholders for data of the propositional content. They also select the parts of the propositional content to be rendered and associates them with the placeholders.
2. The second step executes specific content transformation rules within the context of the rules of the first step. This allows the selection of abstract widgets depending on the content type, the content's referring communicative act type and the current context the communicative act is embedded in as defined by the enclosing rule.

We explain this process in more detail by means of our running example. For transforming the discourse model excerpt in Figure 1, we need structural transformation rules for transforming the question-answer adjacency pairs, as well as content transformation rules for transforming strings, enumerations and numbers dependent on the communicative act. The two structural rules below are applied first to the discourse model.

Open Question Rule: The rule in Figure 2(a) matches an adjacency pair relating an open question and an answer (upper part of the figure) and transforms it to a panel containing a label for the question text, input and output widget placeholders for the content, and a submit button for submitting the answer. The rule also assigns all attributes of the open question's propositional content type to both placeholders. In our example, these attributes are the ones of a credit card.

Closed Question Rule: The rule in Figure 2(b) transforms each closed question-answer adjacency pair to a list with each list entry consisting of a heading label, an output widget placeholder for the description of one item of the closed question and a button to select it.

The following four rules are used in the online shop example to transform content types depending on the type of communicative act referred from.

Basic Input String TextBox Rule: The rule in Figure 2(c) matches the string content type and is constrained to input widget placeholders. It generates a text box for each string. In our example, it generates for each credit card string attribute associated with the input widget placeholder of the open question a text box. Afterwards the placeholder gets removed.

Basic Input Enum ComboBox Rule: The rule in Figure 2(d) is similar to the *Basic Input String TextBox Rule*, but

[1] At the time of this writing, the specification of UML is available at `http://www.omg.org`.

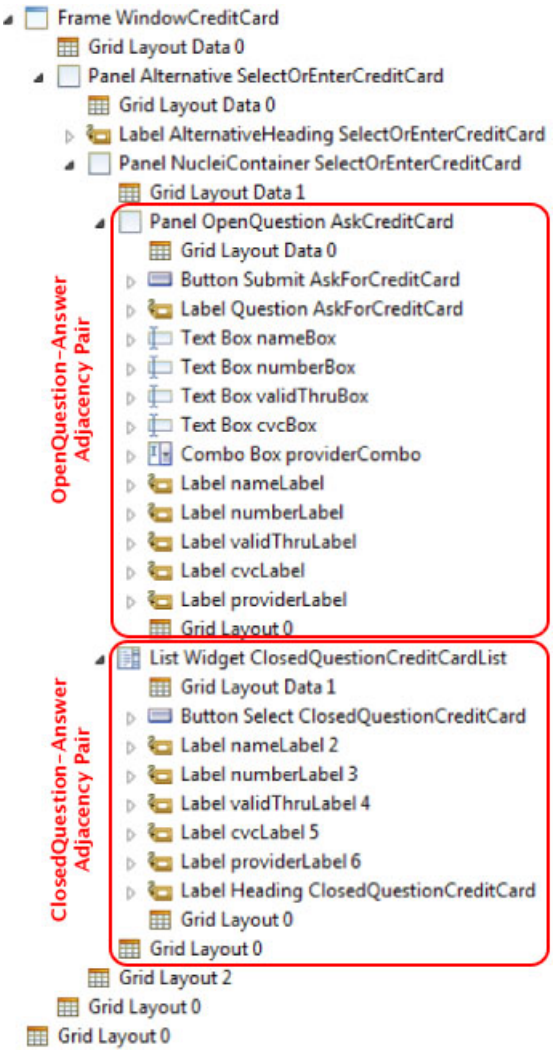


Figure 4. Structural UI model corresponding to the online shop discourse model excerpt.

matches enumerations and creates a combo box with the literals of the enumeration as selection list.

Basic Output Open Question Label Rule: The rule in Figure 2(e) matches any content type of content attributes assigned to output widget placeholders generated for open question communicative acts (specified by the two type constraints). For each matched content attribute, e.g., credit card number, the rule generates a label and sets the label text to the attribute name. This label is used to identify the attribute's corresponding input widget generated by one of the rules described above.

Basic Output Closed Question Label Rule: The rule in Figure 2(f) matches also any content type of content attributes assigned to output widget placeholders generated for closed question communicative acts. In contrast to the *Basic Output Open Question Label Rule*, this rule assigns the actual propositional content to the generated label.

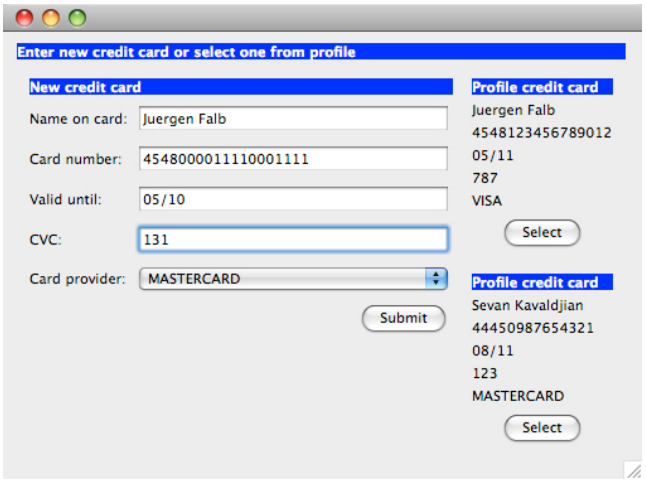


Figure 5. Screenshot of the final UI representing the alternative closed question and open question.

After these rules are applied to our small running example discourse, we get the generated structural UI model illustrated in Figure 4. The resulting structure of the open question and closed question adjacency pairs marked by the rounded rectangle in Figure 4 corresponds to the structures shown in Figure 2(a) and (b) with the input and output placeholder widgets replaced by the application of the other rules. The surrounding structure in Figure 4 corresponds to the Alternative RST relation and is generated by rules not presented in this paper.

SCREEN GENERATION

A structural UI model resulting from our model transformation process, like the one in Figure 4, contains already the complete structure and layout information of the GUI but is still GUI toolkit-independent. Screen generation is our final step that transforms the structural model into GUI toolkit-specific windows and dialogs and generates code for them. Currently, we support the Java Swing[2] and Eclipse SWT[3] GUI toolkits. This screen generation step solves three tasks:

- It maps the abstract widgets of the structural model to toolkit-specific widgets,
- it maps the generic structural UI layout to a toolkit-specific layout, and
- it generates the event handling and the binding to the user interface behavior (represented as a generated finite-state machine as described in [8]).

Figure 5 displays the screen resulting from the structural UI model in Figure 4 by applying the screen generation for Eclipse SWT. In this example, we achieved a one-to-one mapping between structural model and SWT widgets. In some cases, the mapping process is more complex, e.g., our structural UI metamodel contains an image map widget which

[2] `http://java.sun.com/products/jfc/`
[3] `http://www.eclipse.org/swt`

requires, for example, an image, a label widget and the implementation of multiple active areas within SWT.

For transforming the layout information of the structural UI model, we implemented an algorithm that calculates layout data required for the toolkit-specific layout managers. E.g., for the Java Swing *GridBagLayout* we calculate the weight of each grid cell, which is important for resizing of the window, depending on the widget types and the layouting rule applied to the discourse relation. For example, the Alternative relation used in our running example weighs both branches equally, thus the actual resize behavior depends only on the widgets used.

In addition, the screen generation results in toolkit-specific event handlers that collect information from the input widgets, modify content objects provided by the application logic appropriately or generate new ones. Afterwards, the event handlers create input for the application logic and hand them over to the finite-state machine that implements the user interface behavior, which then selects the next screen according to the advancement of the dialogue.

Finally, the screen generation process also internationalizes and localizes the generated GUI, e.g., it externalizes strings and provides translation files with default suggestions derived from the discourse model, the content types and the transformation rules.

RELATED WORK

Our approach to GUI generation is similar to TERESA by Mori *et al.* [6]. Both start from high-level models, but our discourse models have a focus on dialogues and seem to be even on a higher level than the task models in [6].

The UI Pilot approach by Puerta *et al.* [9] is semi-automatic by requiring the designer to specify tasks and a wireframe for the user interface. Afterwards, the tool can suggest widgets for each user interface element. This approach provides more flexibility to the user interface designer than our approach, which allows fully automatic content presentation.

Pederiva *et al.* [7] describe a beautification process that helps a designer to improve a generated user interface via a constrained user interface editor. This editor allows applying beautification operations to specific UI elements, resulting in model-to-model transformation. Since our approach involves content presentation specific to intentions, beautification should be less important for this part of UI generation.

CONCLUSION

We address the problem of presentation of content from the domain of discourse according to its purpose in the current state of the human-machine dialogue. This purpose relates to the intention indicated by the type of the communicative act that refers to propositional content to be presented. Our approach takes this type into account in the course of automatic content presentation. In this way, it leads to generated user interfaces with content presentations specific to intentions.

ACKNOWLEDGMENT

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

REFERENCES

1. C. Bogdan, H. Kaindl, J. Falb, and R. Popp. Modeling of interaction design by end users through discourse modeling. In *Proceedings of the 2008 ACM International Conference on Intelligent User Interfaces (IUI 2008)*, Maspalomas, Gran Canaria, Spain, 2008.

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 *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, pages 754–759, New York, NY, USA, 2006. ACM Press.

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

4. P. Luff, D. Frohlich, and N. Gilbert. *Computers and Conversation*. Academic Press, London, UK, January 1990.

5. W. C. Mann and S. Thompson. Rhetorical Structure Theory: Toward a functional theory of text organization. *Text*, 8(3):243–281, 1988.

6. G. Mori, F. Paterno, and C. Santoro. Design and development of multidevice user interfaces through multiple logical descriptions. *IEEE Transactions on Software Engineering*, 30(8):507–520, 8 2004.

7. I. Pederiva, J. Vanderdonckt, S. España, I. Panach, and O. Pastor. The beautification process in model-driven engineering of user interfaces. In *Proceedings of the 11th IFIP TC 13 International Conference on Human - Computer Interaction — INTERACT 2007*, pages 411–425, Rio de Janeiro, Brazil, Sept. 2007. Springer Berlin / Heidelberg.

8. R. Popp, J. Falb, E. Arnautovic, H. Kaindl, S. Kavaldjian, D. Ertl, H. Horacek, and C. Bogdan. Automatic generation of the behavior of a user interface from a high-level discourse model. In *Proceedings of the 42nd Annual Hawaii International Conference on System Sciences (HICSS-42)*, Piscataway, NJ, USA, 2009. IEEE Computer Society Press.

9. A. Puerta, M. Micheletti, and A. Mak. The UI pilot: A model-based tool to guide early interface design. In *Proceedings of the 10th International Conference on Intelligent User Interfaces (IUI'05)*, pages 215–222, New York, NY, USA, 2005. ACM Press.

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