Transforming a Discourse Model to an Abstract User Interface Model

Sevan Kavaldjian, Cristian Bogdan, Jürgen Falb, Hermann Kaindl
{kavaldjian, bogdan, falb, kaindl}@ict.tuwien.ac.at
Vienna University of Technology
Institute of Computer Technology
A-1040 Vienna, Austria

ABSTRACT

User-interface design is still a time consuming and expensive task to do, but recent advances allow generating them from interaction design models. We present a model-driven approach for generating user interfaces out of interaction design models. Our interaction design models are discourse models, more precisely models of classes of dialogues. They are based on theories of human communication and should, therefore, be more understandable to humans than programs implementing user interfaces. Our discourse models also contain enough semantics to transform them automatically into user interfaces for multiple devices and modalities. This paper presents a two-step transformation approach with an intermediate abstract UI model. In this paper we concentrate on the first step, transforming discourse models to abstract user interface models by showing transformation rules.

1. INTRODUCTION

In previous work [6], we have already been able to automatically generate usable user interfaces (UIs), even for multiple devices and for real-world applications. We generated such UIs from models, but since these models included finite-state machinery they were more in the spirit of abstract UIs rather than high-level interaction design.

More recently, in the OntoUCP[1] project, we wanted to work with models that are more understandable to and possibly more easily to build for humans. Therefore, we studied several theories of human communication from various fields. Based on insights from some of these theories, we focus on high-level specifications of discourse in the form of models. These models specify discourse in the sense of dialogues, where monologues are embedded and connected.

From our previous work, we inherit the use of *communicative acts* (and references to domain knowledge). Communicative acts are derived from Speech Act Theory and express intentions in the sense of desired effects on the environment.

By integrating communicative acts with some results from *Rhetorical Structure Theory* (RST) and *Conversation Analysis*, we developed a new discourse metamodel. The metamodel defines what the discourse models should look like in our approach.

So, we strive for high-level modeling of discourse, including dialogues. Such a discourse model is inspired by human communication and serves as an interaction design for a traditional information system. Currently we do not support the generation of UIs with direct manipulation.

From such an interaction design, user interfaces for several devices are to be generated automatically. Since we knew already how to generate them from a kind of abstract UI model, we strived for generating an abstract UI from our new interaction design models. We explain our model-driven transformation approach on the basis of our metamodels and self-defined transformation rules.

2. TRANSFORMATION APPROACH

Our approach to fully automated UI generation is a two-step process illustrated in Figure 1. Model-to-model and model-to-code transformations are necessary to transform a discourse model to an abstract UI model and the abstract UI model to multiple concrete UIs for diverse platforms. In the following, we will explain the input (discourse model structure), the output (structure of the abstract UI model) and the transformation rules for the model-to-model transformation step.

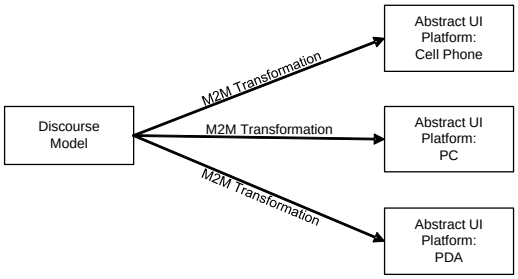

Figure 1: The model transformation steps.

Our discourse models use a self-defined Domain Specific Language (DSL) for specifying the classes of possible dialogues or interactions between the human and the machine. The abstract syntax of the DSL is based on the metamodel shown in Figure 2, which illustrates the used concepts. Every discourse is composed of a tree where leaf nodes are *Communicative Acts* and inner nodes are *Rhetorical Relations* based on RST. The conceptual UML class diagram

[1]OntoUCP (A Unified Communication Platform both for Machine-Machine and Human-Machine Interaction based on Ontologies), partially funded by the FIT-IT Program of the Austrian FFG as project number 809254/9312. We also acknowledge the (financial) support of the PSE division of Siemens AG Österreich.

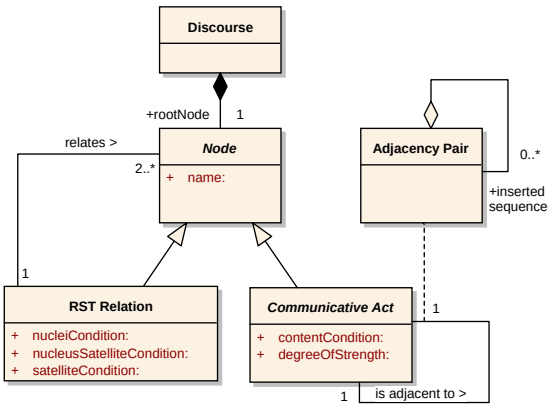

Figure 2: Discourse Metamodel.

shown in Figure 2 is not as restrictive as our interpretation, since it allows to create other kinds of graphs besides tree structures. The association class is needed to model the inserted sequence.

The Communicative Acts are used to model the intention of a communication and refer to elements of the domain of discourse. Figure 3 shows a selection of the most important Communicative Acts used in our approach. Two corresponding Communicative Acts, like *Offer* and *Accept*, form a sequence, which is called *Adjacency Pair*. The Adjacency Pairs build up the dialogue structure.

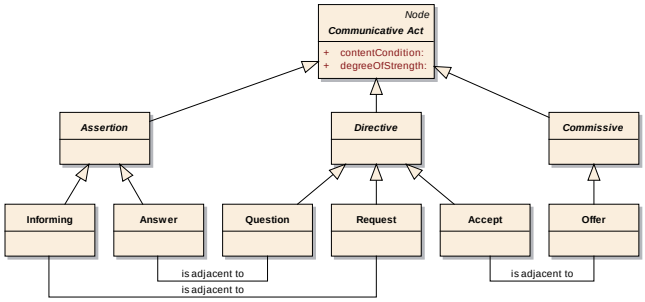

Figure 3: Communicative Act Taxonomy.

The Rhetorical Relations are used to connect Communicative Acts or Rhetorical Relations with each other. They represent the dependencies between single interactions of dialogues. More detailed information about our discourse metamodel can be found in [4] and [1].

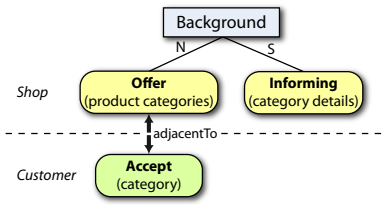

Figure 4: Subtree of an online shop discourse.

Figure 4 shows a small part of an online shop discourse model, that is typical for discourse models and which we will use as a running example throughout the paper. The example describes the interaction between the user and the online shop. The *nucleus* branch N of the *Background* relation conveys the main interaction sequence. The online shop system *offers* a list of *product categories* to the user. The user *accept*s one of them. During the offering process the *satellite* branch S provides background information about the product categories to the user. This part of an online shop discourse model gets transformed to the abstract UI model shown in Figure 5 by applying the rules *Adjacency Pair*, *Offer-Accept* and *Informing* in the listed order. Details on each rule are described below.

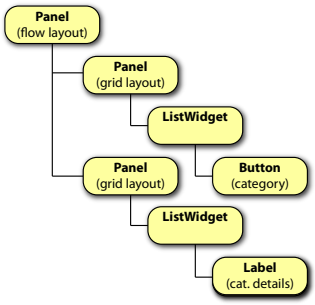

Figure 5: Online Shop Abstract UI Model.

The abstract UI model is basically a tree representing the UI structure. It is not completely independent of the target device, since the device's real estate is considered for building up the abstract UI structure. However, our abstract user interface is completely independent of the considered UI toolkit (e.g. Web, Java Swing, etc.). This tree structure will be transformed to a toolkit-specific concrete UI. The concepts which are used in an abstract UI model are specified in the abstract UI metamodel shown in Figure 6. The most important concept of the metamodel is the *Widget* class. It is specialized into two functional categories, the *OutputWidget*s which have the function of only presenting information and the *InputWidget*s which have the function of presenting and gathering information from the user. Thus, they are actually input/output widgets, but Figure 6 puts the focus on gathering information.

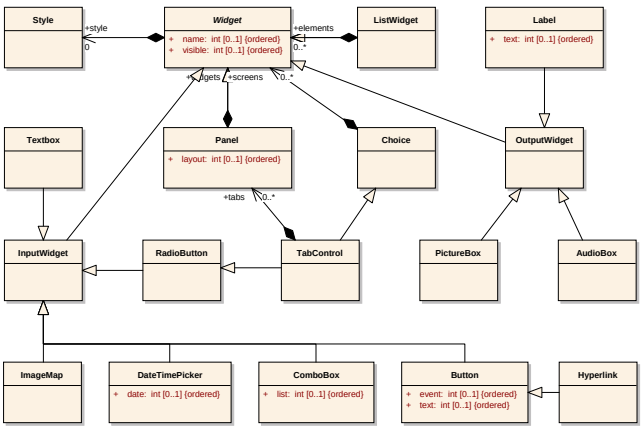

Figure 6: Abstract UI Metamodel.

The main issue that we address in this paper is how to transfer models as exemplified in Figure 4 to user interface models at the abstract widget level. In particular, it means a transformation from a mainly declarative model to a user interface featuring procedural behaviour.

The general principle behind our approach is that the abstract UI model is made up of "presentation" units that are set visible when the logic of the interaction with the user so requires. Once this principle is established, our problem can be specified as follows:

- Given a discourse tree with communicative acts as leafs, generate the possible set of presentation units, and the transitions between these presentation units. Since a presentation unit has to be a coherent discourse, it corresponds to a subtree of the overall discourse tree. As such, we call this problem *the discourse tree partitioning problem*. This problem and a solution to it is described in [1].
- Given a presentation unit as a discourse subtree, generate an abstract UI model based on heuristic rules. Since this effectively "pre-renders" a discourse tree into an abstract UI model, we call this problem *the pre-rendering problem*.

In the abstract UI model, a complete tree or subtree with a *Panel* as root element represents a presentation unit. Hence, a complete abstract UI model can be a forest. Our example in 4 represents exactly one presentation unit that corresponds to the tree shown in Figure 5.

Figure 7 illustrates that the transformation is fulfilled by mapping elements of the discourse metamodel to elements of the abstract UI metamodel. Both metamodels are based on the Ecore[2] meta-metamodel. Transformation languages like the ATLAS Transformation Language (ATL) support this transformation concept. At the same time a state machine is derived from the discourse model which controls the sending and receiving of Communicative Acts.

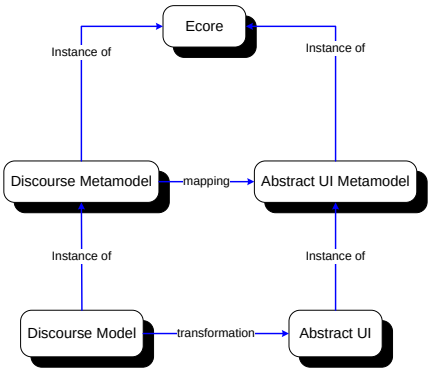

Figure 7: The transformation process.

After having introduced the general transformation principles, we concentrate only on the pre-rendering problem in the remainder of this paper and introduce some rules that are specific to certain structural patterns occurring in the discourse models. We have found many such patterns during our modeling experience, and we continue to find new ones. Due to limited space, we only exemplify five rules which we believe illustrate the principle.

Heavy Background Rule: Figure 8 shows a rule for a "Heavy Background" relation, relating a large satellite subtree with a nucleus subtree. The "nuclear" part is rendered directly, but if there is no space for its background information (which is presumed to be heavy for this rule to apply), the background information is rendered in a separate presentation unit, to which a link is presented.

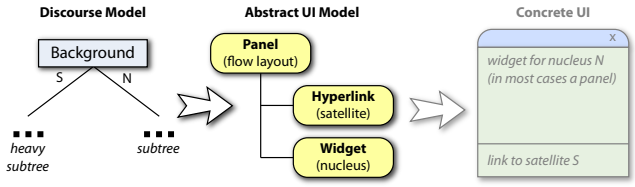

Figure 8: Heavy Background Rule.

Light Background Rule: Figure 9 shows another rule for a `Background` relation on an interface. The satellite is rendered on the right side of the presentation unit, while the nucleus occupies the left area. In accordance to the rule above, the "most nuclear part" takes the interface space that is of highest surface and most central to the user focus. This rule is used in our example to generate the basic tree structure of Figure 5. The Light Background Rule can also be localized (adapted), e.g., for cultures that write from right to left, where it may be more suitable to place the satellite at the left side.

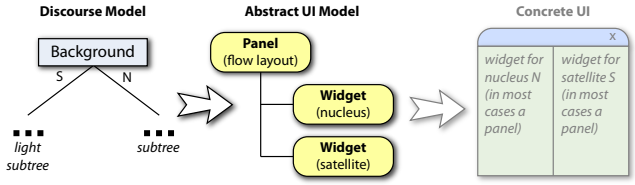

Figure 9: Light Background Rule.

Adjacency Pair Rule: Every adjacency pair is transformed to a *Panel* element of the abstract UI model containing widgets according to the actual related communicative acts. In our example, the first panel on the second level in Figure 5 results from the Offer-Accept adjacency pair. If a communicative act does not take part in an adjacency pair, as it is the case with the Informing in Figure 4, a *Panel* element is also created for the communicative act.

Offer-Accept Rule: Every *Offer-Accept* adjacency pair is transformed either to a *Button* element or to a *ListWidget* element containing a *Button* element depending on the number of content elements offered. Because our example offers more than one product category, the *ListWidget* element is needed to model an undefined number of categories. Since the acceptance of an *Offer* requires a user action, a *Button* element is used.

Informing Rule: Every *Informing* communicative act is transformed either to a *Label* element or to a *ListWidget* element containing a *Label* element, depending on the number of content elements offered. This rule assumes that the information will be forwarded in textual form, otherwise, e.g.,

[2]Essential MOF like core meta model of the Eclipse Modeling Framework (`http://www.eclipse.org/emf/`)

a *PictureBox* or *Audio* element will be used. In the online shop example, *Label* elements are used.

More detailed information about the automatic generation that is used as a basis for this approach can be found in [5] and [6].

3. RELATED WORK

Model-based UI design methods developed and published in the nineties including OVID [9], STUDIO [3], Idiom [11] and Point-of-View Analysis [10] focus on creating different kinds of models, like user's conceptual models, task models and interaction models. Unlike our approach, which is *model-driven*, all the mentioned approaches above are *model-based*. That is, they allow expressing an interactive system by abstract models in a first step and use them in an informal process or in a sequence of systematic steps to construct a concrete user interface.

In contrast, UI Frameworks like XUL[3] (XML User Interface Language) are able to generate UIs automatically but they rely on UI models at the abstract widget level, which is on a lower level than our discourse models.

An advanced approach to specifying multi-device user interfaces based on task models instead of discourse models is presented in [8]. The basic approach is to start modeling tasks and to generate user interfaces for diverse devices according to specific device characteristics. In contrast to our approach, some of the transformations between models are done semi-automatically or manually. The transformations are implicitly coded in the system, there is no genuine transformation engine like ATL or ATOMS3.

Florins et.al. describe in [7] transformation rules for pagination of UIs on different levels. In our approach, we support splitting only while transforming the abstract UI model to the concrete UI, but partitioning our discourse model in presentation sets in the first transformation step provides important guidance for pagination [1].

Botterweck shows in [2] a model-driven approach that starts on the abstract UI level, but contains rich procedural UI descriptions together with UI elements. Thus, it requires UI modeling as well as dialogue modeling.

4. CONCLUSION

In this paper, we present a new approach to generating abstract user interface models by applying model-driven transformations to discourse models. Our discourse models are derived from results of human communication theories, cognitive science and sociology and are used for specifying interaction design of human-computer interaction of information systems. Thus, they contain additional metainformation, like the intention of an interaction, which allows us to define sophisticated pre-rendering rules to transform the discourse models to abstract UI models. Our transformation takes already device constraints into account to generate a UI structure well suited for the target device, but the resulting abstract UI models are still independent of UI toolkits. Taking this together with our previous work on automatically generating concrete UIs, this paves the way for automatic generation of concrete UIs from our new interaction design models.

[3] `http://www.mozilla.org/projects/xul/`

5. REFERENCES

[1] C. Bogdan, J. Falb, H. Kaindl, S. Kavaldjian, R. Popp, H. Horacek, E. Arnautovic, and A. Szep. Generating an abstract user interface from a discourse model inspired by human communication. In *Proceedings of the 41th Annual Hawaii International Conference on System Sciences (HICSS-41)*, Piscataway, NJ, USA, to appear 2008. IEEE Computer Society Press.

[2] G. Botterweck. A model-driven approach to the engineering of multiple user interfaces. In *Proceedings of the MoDELS'06 Workshop on Model Driven Development of Advanced User Interfaces*, Genova, Italy, Oct. 2006. CEUR-WS.

[3] D. Browne. *STUDIO: STructured User-Interface Design for Interaction Optimisation*. Prentice Hall, Englewood Cliffs, NJ, USA, 1993.

[4] 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 CHI '06*, 2006.

[5] J. Falb, R. Popp, T. Röck, H. Jelinek, E. Arnautovic, and H. Kaindl. Using communicative acts in interaction design specifications for automated synthesis of user interfaces. In *Proceedings of the 21th IEEE/ACM International Conference on Automated Software Engineering (ASE'06)*, pages 261–264, Piscataway, NJ, USA, 2006. IEEE Computer Society Press.

[6] J. Falb, R. Popp, T. Röck, H. Jelinek, E. Arnautovic, and H. Kaindl. Fully-automatic generation of user interfaces for multiple devices from a high-level model based on communicative acts. In *Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS-40)*, Piscataway, NJ, USA, Jan 2007. IEEE Computer Society Press.

[7] M. Florins, F. M. Simarro, J. Vanderdonckt, B. Michotte, and B. Michotto. Splitting rules for graceful degradation of user interfaces. In *AVI '06: Proceedings of the working conference on Advanced visual interfaces*, pages 59–66, New York, NY, USA, 2006. ACM Press.

[8] 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.

[9] D. Roberts, D. Berry, S. Isensee, and J. Mullaly. Developing software using OVID. *IEEE Software*, 14(4):51–57, July-Aug. 1997.

[10] S. R. Robertson, J. M. Carroll, R. L. Mack, M. B. Rosson, S. R. Alpert, and J. Koenemann-Belliveau. A self-guided, scenario-based learning environment for object-oriented design principles. In *Proceedings of OOPSLA 94*, 1994.

[11] M. van Harmelen. Object oriented modelling and specification for user interface design. In *Interactive Systems: Design, Specification and Verification*, 1994.