Generating High-Level Interaction Models out of Ontologies

Dominik Ertl, Hermann Kaindl,
Edin Arnautovic, Jürgen Falb,
and Roman Popp
Vienna University of Technology
Institute of Computer Technology
(ertl, kaindl, arnautovic, falb,
popp)@ict.tuwien.ac.at

ABSTRACT
Generating user interfaces out of semantic models is still an issue because of the semantic gap between ontologies and user interfaces. We bridge this gap through semantic model-driven development. More precisely, we show how to automatically generate high-level interaction models (in the form of communication models representing discourses) out of (annotated) ontologies, using model-transformation rules. From these discourse models, user interfaces can be generated (semi-)automatically.

INTRODUCTION
The most important elements of any interactive system are the information it contains and the user interface through which this system communicates with its users. The information may be represented with (formal) semantic models (e.g., based on ontologies), and the user interface is typically created manually on top of such models. This requires a lot of effort, especially if these models are modified and the user interface has to be adapted manually.

In a specific category of interactive systems, such as product recommendation systems, reservation systems or shopping applications, the underlying (semantic) model may strongly influence the behavior of the systems and, therefore, also the interactions to be implemented through the user interfaces. For this category of interactive systems, we address the semantic gap between underlying ontologies and user interfaces. We make use of our *discourse models* [1, 3] for bridging this gap. In this course, a discourse model and a domain-of-discourse model together serve as a high-level interaction model and, as such, as a kind of "intermediate language" between the ontology and the user interface. In addition, such a model can even be used for the (semi-)automatic generation of a user interface [3, 10].

The remainder of this paper is organized as in the following manner. First we give a brief background on our previous work relevant for this paper, and compare it with some of the related work in the field. Then we present our approach for generating interaction models out of (annotated) ontologies. This approach contains two parts: the generation of *discourse* models representing the flow of communication between the user and the computer, and the generation of *domain-of-discourse* models representing what they "talk about". Finally we conclude and provide an outlook of our future work in this direction.

BACKGROUND AND RELATED WORK
Our previous work focused on manual *modeling* of interaction designs [1], where even end users created interaction designs in the form of discourse models using the graphical editor developed for this purpose. These discourse models are based on several theories of human communication [2]. The key parts of our discourse models are *Communicative Acts* as derived from speech acts [11], *Adjacency Pairs* adopted from Conversation Analysis [5], and *RST relations* inherited from Rhetorical Structure Theory (RST) [6]. Communicative Acts are semi-structured messages carrying the intention (e.g., asking a question or issuing a request) and represent basic units of language communication. Adjacency Pairs are sequences of talk-turns that are specific to human (oral) communication, e.g., a question should have a related answer. RST relations specify relationships among text portions and associated constraints and effects, and are organized in a tree structure. In our work, we use RST for linking Adjacency Pairs of Communicative Acts and further structures made up of RST relations. We have also included procedural constructs, to provide means to express a particular order during discourse execution, to specify repetitions or conditional execution of different discourse parts. Since such discourses cast the communication between a human and the computer on a high level, abstracting from technical details, they may even be created without any programming knowledge and experience.

Instead of our discourse models, ConcurTaskTrees from Paterno et al. [7] may be used for bridging the semantic gap between ontologies and user interfaces. ConcurTaskTrees facilitate modeling *tasks*, that are being transformed into a user interface. Our discourse models focus more on the commu-

IUI SEMAIS Workshop 2011.

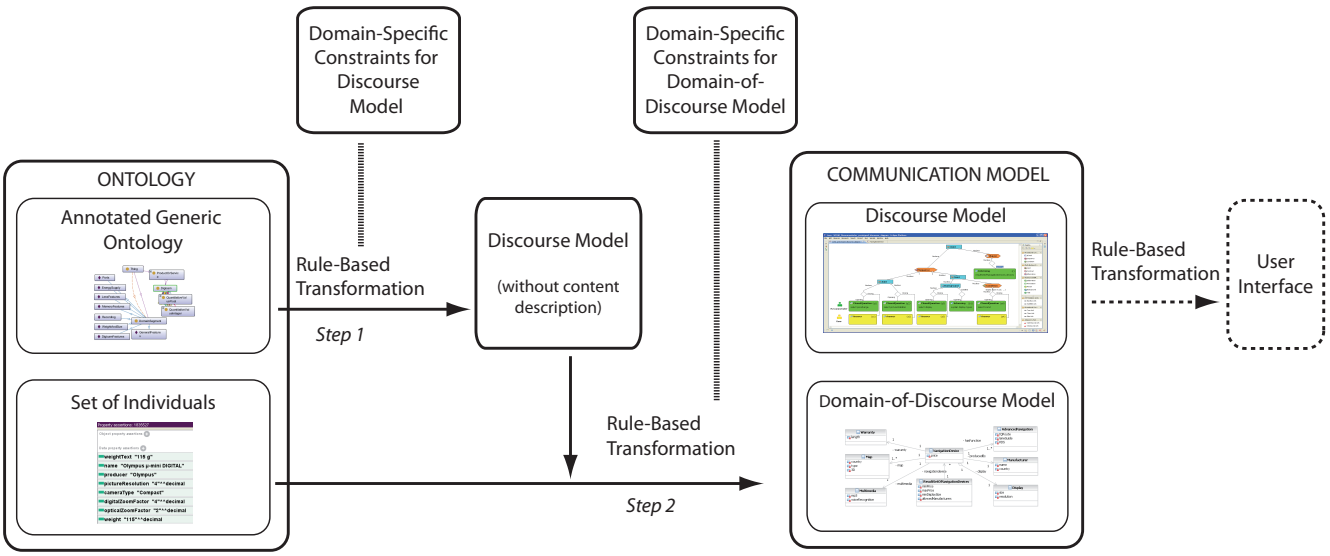

Figure 1. Transformation from ontology to communication model.

nication, and they can be used for machine-machine communication as well [9]. According to our best knowledge, this has not been done with ConcurTaskTrees. We are also not aware of any approach for generating ConcurTaskTrees out of ontologies.

UsiXML [4] is an XML-based specification language for user interface design. It allows describing a user interface at different levels of abstraction, from high-level task models to the concrete code of a user interface. So, it provides an alternative approach to ConcurTaskTrees. Also for UsiXML, we are not aware of any approach for generating UsiXML models out of ontologies.

Paulheim and Probst [8] present a survey about ontology-enhanced user interfaces. They point out that ontologies can be used to improve interaction possibilities, and our approach addresses such a possibility.

FROM ONTOLOGIES TO INTERACTION MODELS

Now let us present our approach to automatically transforming an ontology to a high-level interaction model in the form of a specific communication model by using model transformations. We focus on a small part of an ontology and its corresponding transformations to generate the interaction model of a Product Advisor for digital cameras as a running example. The Product Advisor is designed to ask questions about desired properties of a digital camera to be bought.

Overall Transformation Approach

In Figure 1, we provide an overview of the transformation process, which consists of two steps for generating a communication model from an annotated ontology. We start from such an ontology represented in OWL[1] (illustrated in the left part of Figure 1). While the ontology per se contains the knowledge of the given domain, the annotations contain *meta-knowledge*, e.g., the priority of a given piece of knowledge with respect to the Product Advisor to be implemented. The result is a communication model consisting of a discourse model and a domain-of-discourse model.

We use a GoodRelation[2] ontology for digital cameras as a basis. In Figure 2, we depict selected parts from the Digicam GoodRelation ontology. The top concept *Thing* is specialized by the concepts *ProductOrService* and *DomainSegment*. ProductOrService is further specialized by the *Digicam* concept. DomainSegment groups together properties of a ProductOrService that have a semantic relation with each other.

Our ontology contains additional annotations that describe characteristics of certain datatype and object properties with respect to the intended Product Advisor. For example, the annotation *priority* specifies how important for the Product Advisor a specific object or datatype property is compared to other properties. These priorities are a distinguishing feature when the transformation process applies the transformation rules. The priority is an integer value between 0 (low priority) and 100 (high priority). The priorities allow the transformation process to decide which datatype and object properties are of interest for the discourse and the domain-of-discourse.

In the first step, a set of model-transformation rules matches parts of the ontology (including its individuals) and transforms them automatically into corresponding parts of a discourse model (see the middle part of Figure 1). These transformations are subject to domain-specific constraints explained in detail below. The discourse model generated in this step represents only the generic communication flow

[1]Last visited on December 10, 2010: `http://www.w3.org/TR/owl2-overview/`

[2]Last visited on December 10, 2010: `http://www.heppnetz.de/projects/goodrelations/`

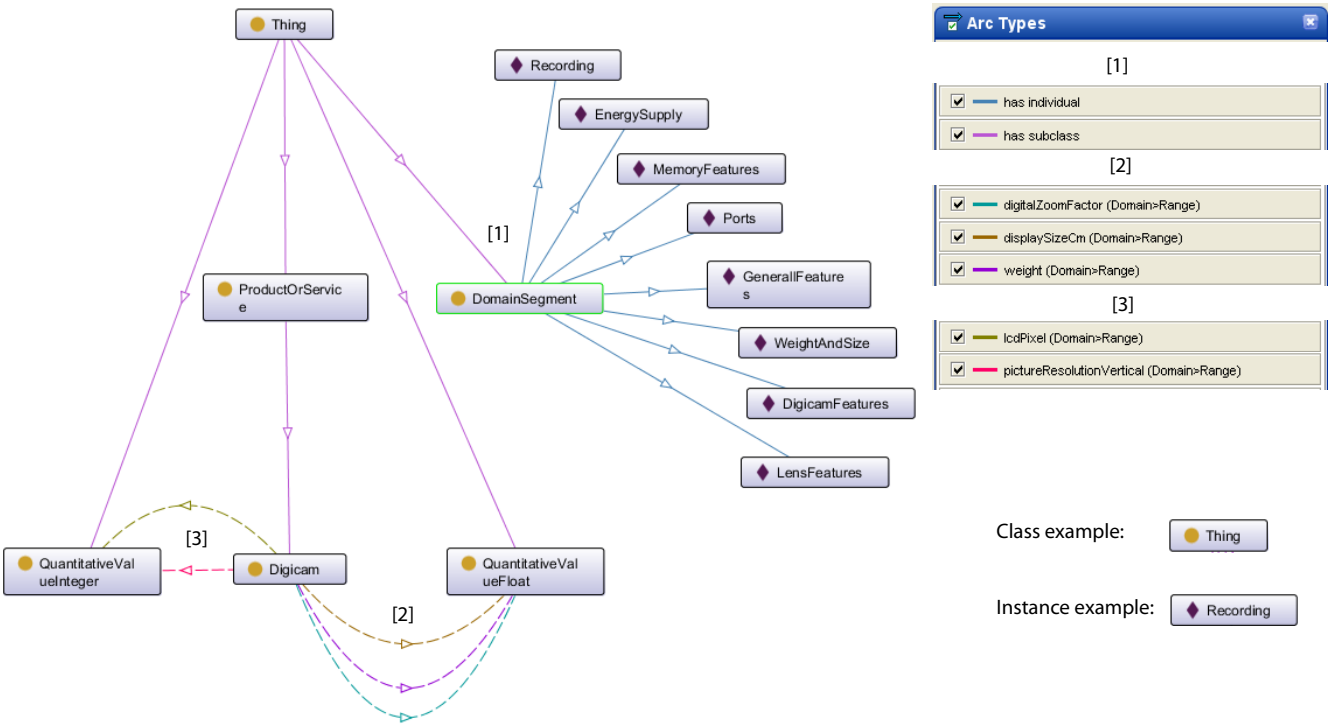

Figure 2. Selected parts from Digicam ontology.

and is incomplete since the *content* of its communicative acts does not (yet) refer to the content of the communication (the domain-of-discourse model).

In the second step, our model-transformation approach transforms the individuals of the ontology and their concrete datatypes and object property values into a domain-of-discourse model. Here we apply domain-specific constraints for a domain-of-discourse model. In effect, this step defines the content of the communicative acts so that the discourse model refers to the domain-of-discourse model. The contents of communicative acts in the case of a Product Advisor are concrete question and answer texts that link to elements of the domain-of-discourse model.

From Ontologies to Discourse Models

Our transformation approach applies several rules to create a discourse and a domain-of-discourse model out of the ontology. Each rule application can be constrained by domain-specific constraints, that are externally configured. We have a logical rule chain (by using Operational Query/View/-Transformation[3] (QVT)) defining the application order of the rules. In principle, a rule that is applied later in the transformation process can influence the outcome of a rule that is applied sooner. In the following, however, we describe two independent rules applied in the transformation process, the *DomainSegmentClusterRule* and the *SingleQuestionRule*.

The first rule explained as an example is the DomainSegmentClusterRule illustrated in Figure 3. It matches the concept *DomainSegment* in the ontology, which has several individuals. For example, the digital camera ontology has the domain segments EnergySupply, LensFeatures, Ports, etc. All datatype and object properties in the ontology that are related to a DomainSegment via the object property *belongsToDomainSegment* are of interest for our transformation process.

The rule DomainSegmentClusterRule creates a *cluster* of questions for all object and datatype properties that belong to the same domain segment. A cluster groups questions that hold a semantic relation (e.g., the ports USB and FireWire belong to the DomainSegment *Ports*). Such a definition of a cluster results in a *Joint* RST relation of a discourse (see the right part of Figure 3). The datatype and object properties are transformed into question/answer pairs that are branches of the Joint relation. In addition to the rule presented above, the following domain-specific constraint applies: Each property needs a *minimum priority* value (e.g., 20) to be included in a cluster. The minimum priority value is configured a priori in the domain-specific constraints.

After the DomainSegmentClusterRule has been applied, all properties with a minimum priority are grouped in the different clusters. For example, USB belongs to the segment *Ports*. Now a rule applies that combines all Boolean properties (like USB) of one domain segment into one question, for optimizing the interaction with the Product Advisor. The left part of Figure 4 shows the datatype property *USB*, which represents the USB port of a digital camera having the priority 85. For this USB property, the *SingleQuestionRule* takes

[3]Last visited on December 10, 2010: `http://wiki.eclipse.org/M2M/Operational_QVT_Language_%28QVTO%29`

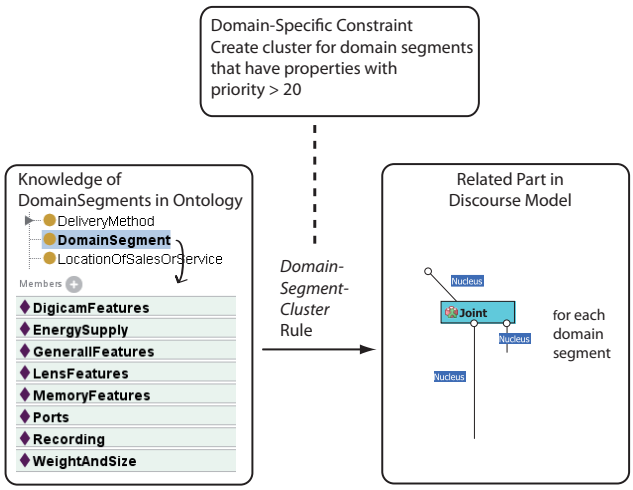

Figure 3. Transformation with DomainSegmentCluster Rule.

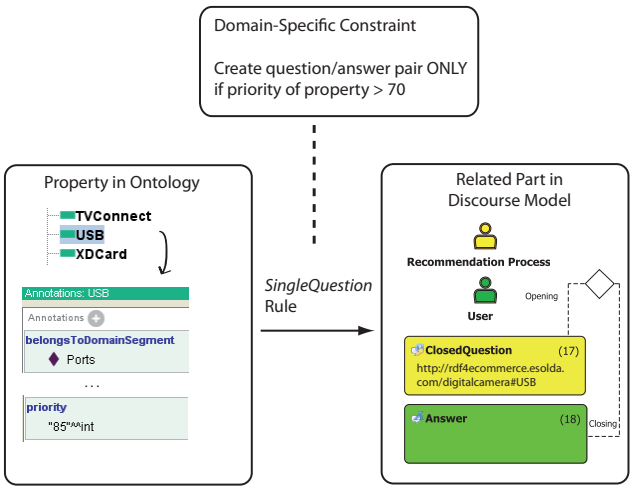

Figure 4. Transformation with SingleQuestion Rule.

effect now. Due to its high priority value (and importance), USB becomes a single ClosedQuestion-Answer pair again. This adjacency pair, shown in the right part of Figure 4, becomes also part of the generated discourse model.

As shown with these example rules and their applications, for each selected property a corresponding question is being generated for the Product Advisor, since this property is considered important for the selection of a camera. While the ontology specifies what exists in the domain, the process being implemented in the Product Advisor contains related questions. This semantic gap is bridged by our approach in the context of the given application.

We show an excerpt of a yet incomplete discourse model in Figure 5, that is the result of the first transformation step depicted in Figure 1. The contents of the communicative acts are URIs that refer to datatype or object properties in the ontology. A *Joint* relation combines one Adjacency Pair and the Background relation connecting two more Adjacency Pairs. In this example, these properties have a high enough priority, so that they have to be grouped together in a special cluster at the beginning of the recommendation process of the Product Advisor. The first question gathers information on the price range, defining the minimum and maximum price that the user is potentially willing to pay. The second question elicits the interest for a USB port on the digital camera. This Boolean question is modeled as a closed question. Moreover, there is an RST relation *Background* intended to optionally inform the human user on additional details about the subject matter, e.g., more information on USB.

From Ontology to Domain-of-Discourse Model

The second step in the transformation from ontologies to our communication models is to generate the *domain-of-discourse model*. This model represents the content of the communication, more precisely the content of the communicative acts within our discourse models. For example, the digital camera's property *hasCurrencyValue* (representing the price of the camera) is the content of the question where the Product Advisor asks the user about his or her preferences (e.g., price range) regarding the camera price (shown at the top of Figure 5). The Product Advisor should only ask for relevant product properties and their values. For example, the prices of all cameras should be within the price range offered for selection. So, the set of individuals of the products is used to generate the possible contents of the communicative acts, e.g., to determine their price range.

So, for the content of each question in the discourse model, a unique *datatype* representing the product property is generated in the *domain-of-discourse model*. Figure 6 shows a small excerpt of such a generated domain-of-discourse model. For product properties representing *numbers* (e.g., price), only the minimum and maximum values are relevant for the Product Advisor (e.g., to generate a slider in the final UI for selecting the preferred value between the minimum and maximum). These values are stored together with the generated datatype. The left part of Figure 6 shows the datatype of the price property realized by a *Float* number. The minimum and maximum values are displayed as an annotation in a *note* below the datatype. For product properties representing *Boolean* values, the concrete individuals do not have to be searched for possible values, of course. As an example of such a Boolean datatype, the USB datatype is shown in the middle of Figure 6. For all *other* properties, an *Enumeration datatype* is generated for storing all possible values. The right part of Figure 6 shows the Enumeration type generated for the *producer* datatype. The values of the enumeration are derived from the set of all camera producer individuals in the given ontology.

The applications of these transformation rules can be influenced by domain-specific constraints specific for the domain-of-discourse model. For example, if no values for a specific property exist in the set of individuals in the ontology or if all of them are same (e.g., if all cameras have a USB interface), then the content of the question would be empty, so that the whole question is deleted from the discourse model.

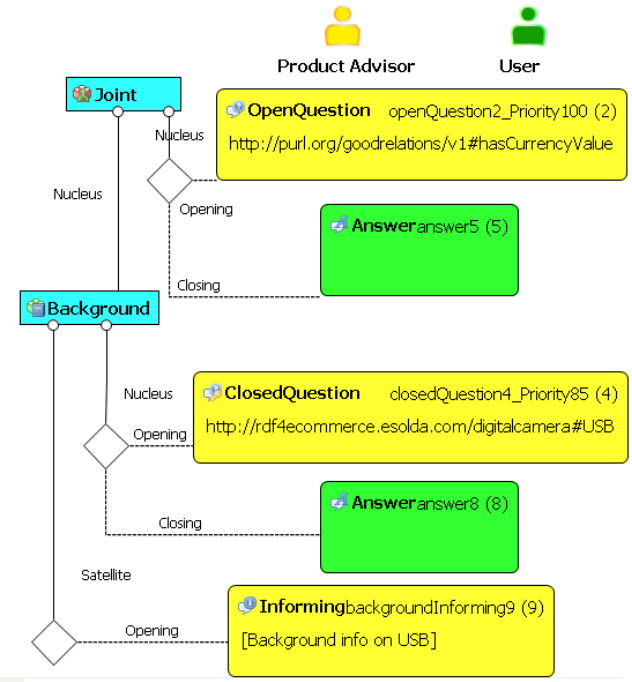

Figure 5. A cluster of questions with high priority.

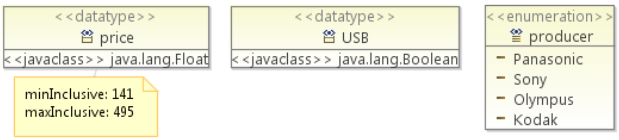

Figure 6. Excerpt of Domain-of-Discourse Model.

CONCLUSION

To ease and speed up the development of ontology-based interactive systems, the automatic generation of their user interfaces would be advantageous. However, due to different perspectives as well as technical and conceptual foci of ontologies used in such systems, the generation of user interfaces directly from ontologies would be hard. We use a high-level interaction model in the form of a communication model based on discourses as an intermediate language. In this paper, we explain the automatic generation of such models out of (annotated) ontologies, and taking application-specific constraints into account.

From such communication models, user interfaces can be generated (semi-)automatically, as we have previously shown already [3]. For small devices, even fully automatic generation leads to usable interfaces through special optimizations of the use of the constrained space [10].

Acknowledgment

This research has been carried out in the SOFAR project (No. 825061), funded by the Austrian FIT-IT Program of the FFG and Smart Information Systems GmbH.

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. ACM Press: New York, NY.

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. ACM Press: New York, NY, 2006.

3. J. Falb, S. Kavaldjian, R. Popp, D. Raneburger, E. Arnautovic, and H. Kaindl. Fully automatic user interface generation from discourse models. In *Proceedings of the 13th International Conference on Intelligent User Interfaces (IUI '09)*, pages 475–476. ACM Press: New York, NY, 2009.

4. D. Faure and J. Vanderdonckt. User interface extensible markup language. In *Proceedings of the 2nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '10)*, pages 361–362. ACM Press: New York, NY, 2010.

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

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

7. F. Paterno, C. Mancini, and S. Meniconi. ConcurTaskTrees: A diagrammatic notation for specifying task models. In *Proceedings of the IFIP TC13 Sixth International Conference on Human-Computer Interaction*, pages 362–369, 1997.

8. H. Paulheim and F. Probst. Ontology-enhanced user interfaces: A survey. *Int. J. Semantic Web Inf. Syst.*, 6(2):36–59, 2010.

9. R. Popp. Defining communication in SOA based on discourse models. In *Proceeding of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications (OOPSLA '09)*, pages 829–830. ACM Press: New York, NY, 2009.

10. D. Raneburger, R. Popp, S. Kavaldjian, H. Kaindl, and J. Falb. Optimized GUI generation for small screens. In *LNCS Volume on Models in Software Engineering: Workshops and Symposia at MoDELS 2010*. Springer, 2011.

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