

## Chapter 2

# Generating Models of Recommendation Processes out of Annotated Ontologies

Hermann Kaindl, Dominik Ertl, Roman Popp, Ralph Hoch, Jürgen Falb, Edin Arnautovic, Ada Okoli, and Martin Schliefnig

**Abstract** Creating content- and dialogue-based recommendation processes through manual adaptations requires a lot of time and effort. Therefore, automated generation of such processes is desirable. We present an approach for generating models of recommendation processes out of annotated ontologies. Such product ontologies have to be provided manually, but certain adaptations to them can be discovered from unstructured data (customer-generated content such as blog entries or customer feedback on products in the Web). They are given input for our approach, which applies semantic model-driven transformations to these ontologies for generating discourse-based models of recommendation processes on a high conceptual level first. These generated discourses essentially consist of questions and answers about

---

Dominik Ertl and Ralph Hoch did this work while being at the Vienna University of Technology.

H. Kaindl (✉) · D. Ertl · R. Popp · R. Hoch · J. Falb  
Vienna University of Technology, Vienna, Austria  
e-mail: [kaindl@ict.tuwien.ac.at](mailto:kaindl@ict.tuwien.ac.at)

D. Ertl  
e-mail: [ertl@ict.tuwien.ac.at](mailto:ertl@ict.tuwien.ac.at)

R. Popp  
e-mail: [popp@ict.tuwien.ac.at](mailto:popp@ict.tuwien.ac.at)

R. Hoch  
e-mail: [hoch@ict.tuwien.ac.at](mailto:hoch@ict.tuwien.ac.at)

J. Falb  
e-mail: [falb@ict.tuwien.ac.at](mailto:falb@ict.tuwien.ac.at)

E. Arnautovic  
Vienna, Austria  
e-mail: [edin.arnautovic@gmail.com](mailto:edin.arnautovic@gmail.com)

A. Okoli · M. Schliefnig  
Smart Information Systems GmbH, Vienna, Austria

A. Okoli  
e-mail: [a.okoli@smart-infosys.at](mailto:a.okoli@smart-infosys.at)

M. Schliefnig  
e-mail: [ms@smart-infosys.at](mailto:ms@smart-infosys.at)

those items annotated as important in the ontologies, and their possible sequences. From such a high-level model, transformation rules create a model of an operationalized recommendation process. This model also represents a so-called concrete user interface and consists of both the structure of the process and the course of events, which defines how customers may navigate through the process. From such models, an already given infrastructure can generate running processes including their final user interfaces, which have already been deployed successfully for real-world use.

## 2.1 Introduction

A content- and dialogue-based recommendation process in the Web guides its user interactively to the most suitable products, based on information it asks for and that is provided by the user. In order to reduce the costs of creating such a recommendation process for real-world use, we strived for automation. In addition or even instead of manual work on it by trained people, we investigated automating its creation for various domains through model transformations. For the overall lifecycle see Kaindl et al. (2013).

Figure 2.1 gives a schematic overview on how the transformation process is implemented. Building upon annotated product ontologies, we realize the generation of recommendation process models and a final user interface in two steps. First a model on a high conceptual level is generated as a so-called Discourse-based Communication Model (see, e.g., Popp and Raneburger 2011). This model consists of three parts, the Domain-of-Discourse Model, the Discourse Model and the Action-Notification Model. In the context of this chapter, the Action-Notification Model is not so important, because we only use predefined elements from there, whereas the other two models have to be generated. The Domain-of-Discourse Model generated in Step 1.1 contains the content of Communicative Acts (as derived from speech acts; Searle 1969), which specify the core of the Discourse Model. Properties as well as additional information from the annotated ontology provide this content. In this sense, the Discourse Model generated in Step 1.2 refers to the Domain-of-Discourse Model and defines a sequence of question and answer pairs, which are modeled as Adjacency Pairs (adopted from Conversation Analysis; Luff et al. 1990) of Communicative Acts. These questions and answers are about recommended products and, therefore created from properties from the annotated product ontology as well. Step 2 transforms such a Communication Model into a model of an operationalized recommendation process, which is presented in the Web as a final user interface of the process for customers through an already given infrastructure.

The remainder of this chapter is organized in the following manner. First, in order to make it self-contained, we present background material. Then we present our approach for generating recommendation process models as discourse-based models out of (annotated) ontologies. Based on that, we explain both our approach for generating operationalized recommendation processes and how they are presented to

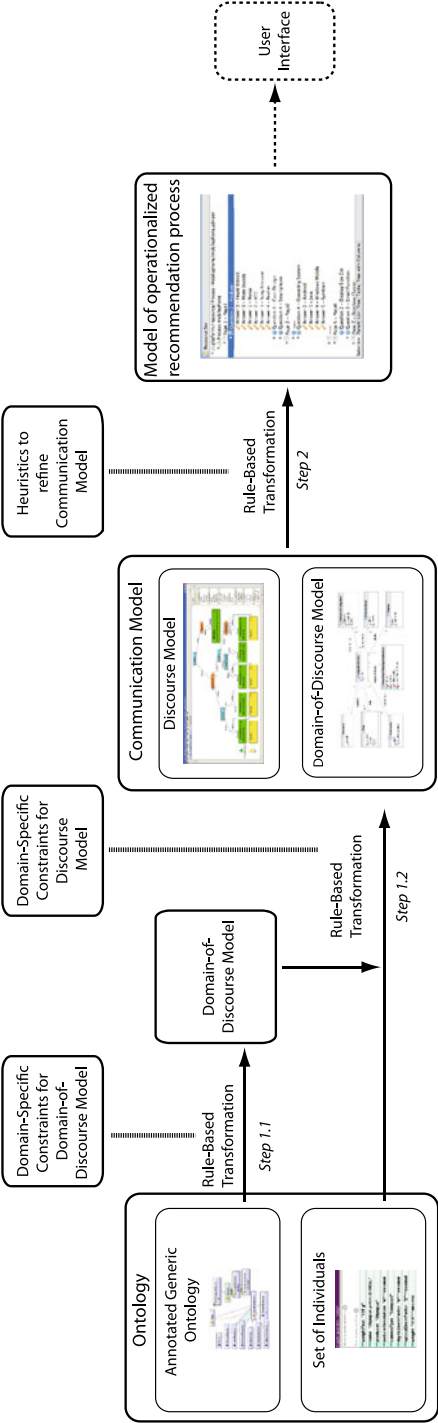


Fig. 2.1 Transformations from annotated ontology to operationalized recommendation process

the customers in the form of user interfaces. For explaining this generation process, we use a mobile phone domain as a running example. A report on our evaluation of such recommendation processes deployed in the real world follows, for the mobile phone domain first, and then for seven other domains as well. Finally, we relate our approach to other work and discuss it more generally.

## **2.2 Background**

We first provide an overview of the complete process lifecycle, which includes also the context of our approach as presented in this chapter. Since the annotated product ontology is the key input, we explain it at a level of detail as required to understand our approach.

### ***2.2.1 Process Lifecycle***

For semi-automatic generation of recommendation processes, we make (indirect) use of knowledge sources on related products in the Web. In fact, this involves a whole recommender lifecycle with the following ingredients and steps. First, customer feedback on the given products can be found as unstructured text reviews of products in the Web, and text mining techniques can be used to extract valuable information. With these processed values it is then possible to adapt a given ontology, including its annotations like recommendation priorities, and thus to influence the order of properties in later steps. Our approach uses the resulting ontology to generate a recommendation process, first on a high level of abstraction represented as a Discourse-based Communication Model. This model already contains the overall sequence as well as metadata information of the question and answer pairs. Using domain-specific heuristics, this process is operationalized in all its details, including a user interface for its actual use of the recommender by customers. When deploying this process in a real-world setting, its results can be compared to other processes (a manually created one first) using A/B-variant tests. More details on this process lifecycle can be found in Kaindl et al. (2013).

### ***2.2.2 Annotated Product Ontology***

To allow transformation from an ontology into a discourse-based model, certain specific model transformation rules are required and thus it was necessary to specify the structure of the ontology as well as additional information fields, such as annotations, and what they represent. As given input for our approach, an annotated ontology has been specified, which encapsulates both our overall structure and

product individuals.<sup>1</sup> It is specified in a way (as a metamodel) facilitating model transformations.

This annotated ontology is designed according to GoodRelations<sup>2</sup> and also contains an individually defined namespace *rdf4ec*. Within this namespace all our custom properties and annotations, such as *rdf4ec:DomainSegment*, are defined and enable to configure properties. For example, Domain-Segments are used to group properties in a semantic and logical way, e.g., *WeightAndDimension*, which classifies all properties that characterize physical dimensions. Domain-Segments are defined as custom individuals in this namespace and new Domain-Segments can be introduced if necessary.

Individuals in the ontology are structured representations of kinds of real-world products, such as different mobile phone models. Note, that these are not the concrete mobile phones to be finally delivered to the customer. These kinds of products can be characterized by their multiple features they have in common, such as *weight*, *resolution*, etc., which are defined through the specified properties in our *rdf4ec* namespace. As the interpretation of these properties may vary by product domain, we use custom ontologies for different product domains.

Properties, as described above, belong to a specific domain segment and are correlated to a *rdf4ec:DomainSegment* via another annotation property that is set for all properties, *rdf4ec:belongsToDomainSegment*, allowing us to arrange properties in groups. This structure enables us in later process steps (compare Sect. 2.3) to handle properties combined that belong to the same logical segment.

For customers, particular properties may be of more interest than others and the ontology needs to support this fact. To facilitate this case, a property annotation *rdf4ec:priority* has been introduced for specifying the relative interest of properties. Properties are defined within a range of 0 to 100, where a higher value means a more important property. Comparing this to real-world products, as an example we refer to mobile phones again. Some properties, such as resolution or screen size, might be considered more important than others, weight for example, and thus would have a higher priority assigned. A higher priority means, if suitable (for a more detailed description see Sect. 2.3), a higher listing in the final recommendation process.

Figure 2.2 shows a schematic presentation of the annotated ontology as a class and an object diagram. The upper part of the figure presents the ontology concepts as a class diagram. The *Property* class defines all values that are necessary to specify a product property as well as additional annotations. Through an annotation *belongsToDomainSegment* it is related to another class *DomainSegment*. A *DomainSegment* can hold several *Properties* but one *Property* belongs only to one *DomainSegment*. The lower part of the figure shows instances of these classes in an object

<sup>1</sup>In the context of object-oriented software engineering and programming, individuals are typically called instances. Since we follow the model-transformation approach from there in addition to building on ontologies, we use these notions interchangeably in the remainder of this chapter.

<sup>2</sup>[www.purl.org/goodrelations](http://www.purl.org/goodrelations).

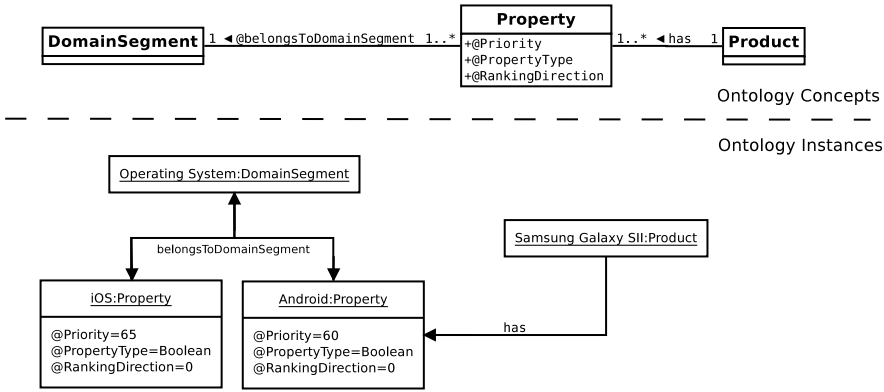


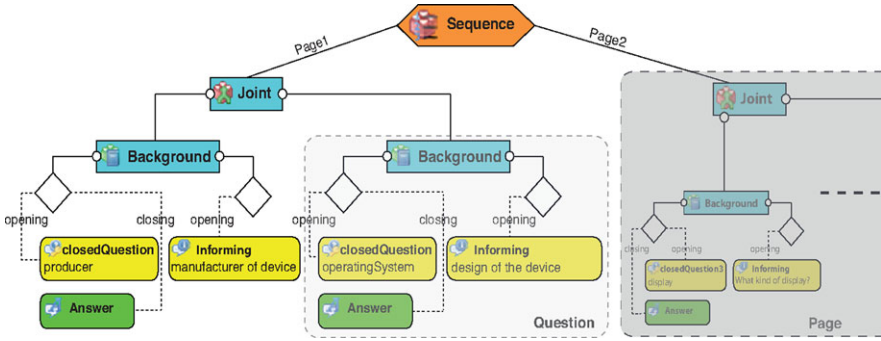
Fig. 2.2 Excerpt of schematic structure of the annotated ontology

diagram. Each class can have multiple instances and each instance has its own custom values. More details on this annotated product ontology can be found in Kaindl et al. (2013).

### 2.3 Generating a Recommendation Process as a Discourse-Based Model

From such a given annotated ontology, we automatically generate a recommendation process on a high conceptual level first. It is represented as a Discourse-based Communication Model (for the background and a definition of such models see, e.g., Falb et al. 2006; Popp and Raneburger 2011). The discourse structure (as explained in detail below) was actually predefined and given to the automated generator as a template. The overall pattern is a sequence of questions and answers related to the recommended products, plus some background information (to be displayed optionally). This template is being filled with information about products to be recommended as given in the product ontology. Its annotations are used by our generator through (manually specified) heuristics to determine what to include in the recommendation process and in which sequence.

Figure 2.3 shows an excerpt of such a Discourse Model. The pairs of related questions and answers are modeled as so-called Adjacency Pairs and shown as diamonds, with opening and closing Communicative Acts (shown as lighter and darker rounded rectangles, depending on whether they are to be executed by the system or the customer, respectively). These Adjacency Pairs are linked with so-called Discourse Relations. In a model of such a recommendation process, only three types of Discourse Relations are used. The first one, *Sequence*, is shown as a hexagon (since it is more specifically a Procedural Construct), the second, *Joint*, and the third one, *Background*, are shown as rectangles (since they are Rhetorical Relations inherited from Rhetorical Structure Theory (RST); Mann and Thompson 1988). While such



**Fig. 2.3** Excerpt of Discourse Model of recommendation process

a model can have many *Joint* relations and many Question-Answer Pairs, only one such element is shown in the figure and a second one is grayed out to indicate the existence of more elements. An earlier version of the representation of a recommendation process through a discourse-based model can be found in Ertl et al. (2011). Note, however, that both the representation and the concrete generation process of the recommender discourse have been changed meanwhile.

The automatic generation of such a model consists of two steps. In Step 1.1 of Fig. 2.1, our model-transformation approach transforms the individuals of the annotated ontology and their concrete datatypes and object property values into a model of the *content* of the communication (the Domain-of-Discourse model). In Step 1.2 of Fig. 2.1, a set of model-transformation rules matches parts of the annotated ontology (including its individuals) and transforms them automatically into corresponding parts of a Discourse Model. This step also defines the content of the Communicative Acts, so that the Discourse Model refers to the Domain-of-Discourse Model.

### 2.3.1 Domain-of-Discourse Model Generation

In more detail, Step 1.1 analyses all properties of the ontology and processes them. Each property is stored as a datatype in the Domain-of-Discourse Model. For most of the datatypes, the values used by the individuals are added to the datatype. In case of numeric properties, only the minimum and maximum values are stored as only the boundaries are important. A definable minimum percentage for individuals of properties can be set and only properties that reach this minimum percentage are selected for the recommendation process currently being generated. This means that not all properties from the ontology are taken into account in the course of generating a recommendation process. Nevertheless, all properties are stored in the Domain-of-Discourse Model and the properties that are not to be used are placed in a special container (if a different kind of heuristics will be put in place later, some of these properties might become important).

Each property also stores meta-data information as annotations. These annotations reassemble the annotations of the annotated ontology, especially *Domain-Segment*, which is used to group properties such as *priority*. These enable ordering of properties, and *propertyType*, which gives additional information on how numeric properties are to be rendered. In later process steps, these annotations are used to organize the question-answer pairs.

All further processing in the course of generating a recommendation process is based on the Domain-of-Discourse Model and thus it is necessary to allow a customizable configuration for the generation process. This includes a configurable value for minimum individual values as well as an extensible annotation container. The latter helps to easily introduce new annotations and thus allows adaptation of the generation process for other application areas.

### 2.3.2 Discourse Model Generation

In Step 1.2, a Discourse Model is created. For each property selected before for the currently generated recommendation process, a question-answer pair is created according to the predefined template. This template consists of a *Question* and a corresponding *Answer* Communicative Act. The *Answer* part can have different internal structure to support single-valued (numeric) as well as multi-valued (string, Boolean) answer sets. Furthermore a configuration value specifies if these answers should be mutually exclusive or not. These Communicative Acts are connected with an *Adjacency Pair*.

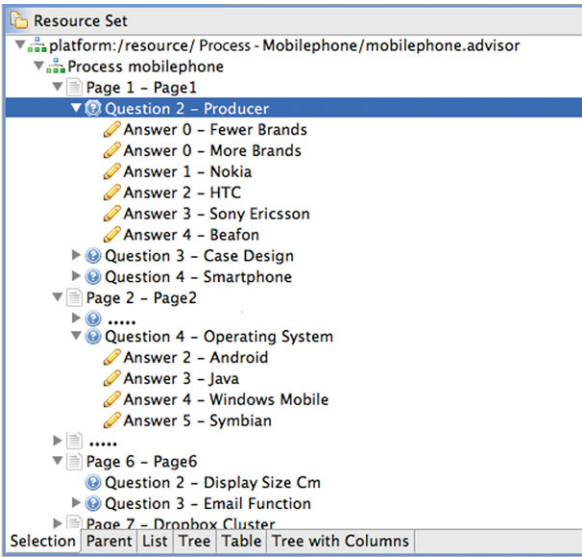
The description property of the transformed property is added to the defined Discourse Model part with a *Background* relation connecting the Question-Answer Adjacency Pair with an *Informing* Communicative Act, which contains the description. It is also possible that some properties of a domain segment are combined into a single question. This combination is applied, if there is more than one property of type Boolean from the same *Domain-Segment*. In our running example, the properties of the domain segment *Operating Systems* are combined. Several individual properties like *android*, *ymbian*, *windowsMobile* have been combined to form a single question.

The generated Discourse Model parts are then sorted according to some heuristics. An example of such a heuristic is, that the question for the producer property is always the first question. Another heuristic is, that questions for properties with a high priority are in front of properties with a lower priority. If the priorities of some questions are the same, we apply another heuristic gained from previous tests of manually created processes. It defines that string questions are asked first, followed by numeric questions and at last combined Boolean questions.

This sorted list is then divided into logical units, which become pages in the resulting user interface. Each of these logical units has a configurable number of questions. In the Discourse Model, each of these logical units is represented with a



**Fig. 2.4** Part of an operationalized recommendation process at the CUI level (reprinted from Kaindl et al. 2013)



*Joint* Relation, which connects the parts created above. The generated *Joint* Relations are connected then with a *Sequence* Relation specifying the order of the logic units.

## 2.4 Generating an Operationalized Recommendation Process and Its User Interface

From such a generated high-level model, model transformations are used to create a model of an operationalized recommendation process. This model also represents a so-called concrete user interface (CUI) (Calvary et al. 2003) and consists of both the structure of the process and the course of events, which defines how customers may navigate through the process. A part of an example of such an operationalized recommendation process can be seen in Fig. 2.4.

### 2.4.1 Generation of Recommendation Process Model

To generate the operationalized recommendation process model, transformations are applied. Manually provided heuristics are used to refine the discourse-based model and to transform elements into concrete representations. To illustrate this task, we present selected specifications of the *Question* element as seen in Fig. 2.4:

- String-Question—a question that has several predefined answer sets: Answers refer to a specific property, e.g., the “Producer” property in Fig. 2.4, as each answer—in this case a brand—is only related to one property.

- **Combined Boolean-Question**—a combined question with several predefined answers to select or not:  
In this case, the answers refer to different properties and belong to a specific domain segment. An example can be seen in Fig. 2.4, where “Operating System” is the domain segment and the answers are distinct properties that belong to this segment.
- **Numeric-Question**—a question referring to a specific numeric property:  
It enables the definition of a value range (e.g., “Display Size Cm”). Boundary values and step-width are defined in the Domain-of-Discourse Model.

Furthermore, a specific representation of each question type is defined. A numeric question, for example, can be represented by a single-slider (minimum or maximum value) or a double-slider (range of values). An example can be seen in Fig. 2.4, where “Display Size Cm” is a numeric question. Slider-Values—like start and end value—are based on the actual minimum and maximum values of the product instances and the specific representation is predefined in the Domain-Of-Discourse Model. Similar definitions apply to the other question types as well.

There are several other heuristics in place for the model transformation. Some of them cover

- text-patterns for questions,
- definition of background information,
- restricted answers per question,
- restricted answer selection per question, etc.

The resulting model consists of the logical layers used during a recommendation process. A root element ‘process’ is in place as a starting point, which contains ordered pages. These pages serve as a container for question-answer pairs, where each question can have multiple related answers. Answers themselves can be restricted and special answer elements can be used to show/hide additional answers (“More/Less” switch). In addition, questions are based on product properties and have varying characteristics, which are rendered differently in the final user interface. For an example see Fig. 2.5, where two different question types and their representations are shown.

Allowing customers to easily navigate through the process and identify the parts of importance is mandatory. Thus heuristics have been put in place to support this. Let us demonstrate it with the *Producer* property. Text-patterns are used to automatically generate meaningful questions that can easily be understood by the customers. Different approaches are used for different question types. In case of a string question, the text-pattern uses the property name to create a comprehensive question. Furthermore, the number of answers is limited and only a fragment is shown, so that customers are not overwhelmed by options. A special switch can be used to show/hide additional answers (“More/Less” switch). Note, that this is a simplified description of the CUI Model as it is possible to parameterize the model transformation, and various heuristics are in place for different requirements.

**Page1** Page2 Page3 Page4 Page5 Page6 Ergebnis

**Which Producer Do You Prefer?**

☐ Nokia ☐ HTC

☐ Sony Ericsson ☐ Beafon

[More Brands](#)

The company that produces the certain good or service for sale. In general the producer is equivalent to the brand.

**What Should Be The Minimum Amount Of Display Size Cm?**

0 10

7

Defines the diagonal length from the upper left to the lower right angle.

**Fig. 2.5** Example of a generated final user interface (reprinted from Kaindl et al. 2013)

Our model of the operationalized recommendation process can be seen as a 1:1 mapping for the final user interface but is still independent from a final implementation. An advantage of this method is that it supports, if necessary, manual adjustment of the process without changes at the implementation level. Adjustments can involve, for example, adding/deleting elements (pages, questions and answers) as well as changing question types. Another advantage of this approach is that it enables the use of different implementations as well as layouts for the final UI.

### 2.4.2 Final User Interface

The final user interface (UI) presented in Fig. 2.5 corresponds to the model on the CUI level in Fig. 2.4. The process together with instance data enables the recommender to give specific recommendations based on the selection of the customer. The combination of actual instance data (e.g., products of a specific Web-shop) is done during runtime by an external engine.

This specific engine provides an interesting feature in addition to what the runtime engine presented by Popp and Raneburger (2011) offers. The engine used for the recommendation process allows the user to redefine her criteria and also to go back in the process. In this case the final UI is generated as an HTML-based Web-shop where various layouts, e.g. customizable color schemes, can be applied.

Elements, such as the sliding controller, are rendered according to the settings on the CUI level.

Figure 2.5 shows an example for such a generated UI. The UI mainly consists of two parts, the top with a bar that allows the customer to navigate between different pages and the question containers. These containers hold one question each (for example *Producer*) and several of them are placed on one page. In our example two representations are shown, one for a numeric question and one for a string question. The string question furthermore holds a ‘More’/‘Less’ switch, which can be used to display additional (hidden) answer options. Such a final UI is the end product of the generation process.

## 2.5 Evaluation

With the aim of evaluating the possible application of such generated recommendation processes within real-world scenarios, we have deployed them in active online shops.

### 2.5.1 Comparison of Recommendation Processes

In these real-world experiments, we have compared and identified differences between semi-automatically generated recommendation processes and manually created ones that have been designed by human domain experts. Both the semi-automatically generated and the manually created product recommendation processes ranked product features according to the computationally inferred relevance for customers. This is based on the underlying assumption that recommendation processes in which the selectable product features are arranged according to their relevance, would perform better than others. Prior to each test, we have set up suitable, automated monitoring services that sent out notifications whenever a decreasing performance (fewer successful recommendations) was detected. As the experiment was carried out in the real-world environment of online shops, these precautionary measures were important to anticipate and prevent any commercial losses due to potentially deficient recommendation processes.

For the comparison, we use the performance measure of the “click-out rat” to measure the success of the recommendation processes. It is a performance metric that is commonly used to measure the success of individual online strategies. In our case, it denotes the fraction of users who used a recommendation process to find a product and followed a recommendation by selecting suggested products to view them in a close-up view or by placing them into an electronic shopping cart (Performed product click-outs/Number of unique clients). This measure is opposed by the “cold exit rate”, which measures the rate of customers that have used a recommendation process, but have left it without following a recommendation.

**Table 2.1** Comparison of ranking of properties in mobile phone recommendation processes

| (a) Manually created recommendation process  | (b) Semi-automatically generated recommendation process   |
|--|---|
| <ul style="list-style-type: none"> <li>• Brand</li> <li>• Price</li> <li>• Multimedia <ul style="list-style-type: none"> <li>– Camera</li> <li>– Video</li> <li>– MP3</li> </ul> </li> <li>• Connectivity <ul style="list-style-type: none"> <li>– WLAN</li> <li>– GPS</li> <li>– Bluetooth</li> </ul> </li> <li>• Operating system <ul style="list-style-type: none"> <li>– Android</li> <li>– Symbian</li> </ul> </li> </ul> | <ul style="list-style-type: none"> <li>• Brand</li> <li>• Price</li> <li>• <b>Display</b> <ul style="list-style-type: none"> <li>– Touch display</li> <li>– Color display</li> <li>– Display size</li> </ul> </li> <li>• <b>Operating system</b> <ul style="list-style-type: none"> <li>– Android</li> <li>– Symbian</li> </ul> </li> <li>• Camera</li> <li>• Connectivity <ul style="list-style-type: none"> <li>– WLAN</li> <li>– GPS</li> <li>– Bluetooth</li> </ul> </li> </ul> |

### 2.5.2 Empirical Results

Based on this way of comparing recommendation processes, let us present empirical results from deploying semi-automatically generated processes. First, we look at the data of one specific comparison in-depth. After that, we summarize results from several different real-world domains and applications.

Let us first provide an example for mobile phones (from where our running example has been distilled as well). Comparing the generated mobile phone recommendation processes (Table 2.1), we found that the semi-automatically generated recommendation process ranked features that were related to the mobile phone *display* and the *operating system* on higher positions, while they were neglected or ranked differently in the manually created version. Also, product features that were related to *Multimedia* (Camera, Video, MP3) and *Connectivity* (WLAN, GPS, Bluetooth) were listed on lower positions in the semi-automatically generated variant. This suggests that these features have lost relevance over time, unrecognized by the domain experts in time. These results reflect the relevance of properties as given in Table 2.1. Here, the product features *touchscreen*, *displaySizeInch* and *resolution-Text* (*Display*) and *android* (*Operating system*) received higher priorities than *video-Function* and *mp3Player* (*Multimedia*) or *WLAN*, *GPS* and *bluetooth* (*Connectivity*) from the given product ontology.

In a real-world experiment, we investigated measurable performance of the pursued approach. As recommendation processes play an increasingly important role in e-commerce, we conducted the experiment within the environment of a large German online retailer. Here, recommendation processes are deployed to support online customers at finding suitable products from vast assortments. The experi-

**Table 2.2** Results of A/B-variant test with (a) manually and (b) semi-automatically generated mobile phone recommendation process instances

| Process variant                           | Unique clients | Click-outs | Click-out rate | Cold-exit rate |
|---|----------------|------------|----------------|----------------|
| (a) Mobile phone rec. process (manual)    | 1,068          | 421        | 0.394          | 0.781          |
| (b) Mobile phone rec. process (generated) | 1,100          | 496        | 0.451          | 0.746          |

ment ran for 14 consecutive days and involved 2,168 uninformed online customers, who accessed the recommendation processes to find suitable mobile phones.

An A/B-variant test, in which the participants were equally distributed to either process variant, was set up to compare the performance of the semi-automatically generated mobile phone recommendation process variant with the manually created variant. Table 2.2 depicts the results of the A/B-variant tests, in which the semi-automatically generated mobile phone recommendation variant (b) was tested against a manually created variant (a). It shows that the semi-automatically generated version (b) led to an increase of the click-out rate by 14 % and a decrease of the cold exit rate by 4 %. The result of this experiment indicates that our new approach may lead to a higher click-out rate and, therefore, may evoke performance increases.

Now let us provide a general evaluation of the performance values for all 8 processes that have been deployed and tested yet at the time of this writing within this experiment series. The 8 recommendation process sets within the categories bluray-player, camcorder, printer, receiver, videoprojector, DVD player, TFT screen, and mobile phone, each comprising a manually created and a semi-automatically generated variant, were tested within the experiment series.

The results are given in Table 2.3. We observe that 5 out of 8 generated product recommendation processes suggest improved data as compared with their manually created counterparts. A statistical analysis of the data is given by Kaindl et al. (2013).

The 3 product recommendation processes that led to decreasing click-out rate data (see Table 2.3c, f, g) had more complex process setups and involved higher interdependencies between the selectable product features. This may be an indicator that our current approach is better applicable for more basic process types.

Overall, the results of our real-world usage experiments provide some empirical evidence that the generation of process recommendation processes can be done semi-automatically with competitive results.

Measuring and comparing the effort of manual creation of recommendation processes with using the semi-automatic lifecycle showed that applying the latter led to a reduction of the manual work by roughly up to 60 %.

## 2.6 Related Work

Ontologies have been used in many areas of software engineering (Coral et al. 2006) and information systems (Guarino 1998). Paulheim and Probst (2010) present an

**Table 2.3** Results and comparison of A/B-variant tests of recommendation process instances in 8 product categories—manually vs. semi-automatically generated process instance

| Domain             | Unique clients | Click-out rate A (manual) | Click-out rate B (semi-automatic) | Click-out increase/decrease of B | Cold-exit rate increase/decrease of B |
|--------------------|----------------|---------------------------|-----------------------------------|----------------------------------|---------------------------------------|
| (a) blurayplayer   | 1,631          | 0.458                     | 0.509                             | +11.14 %                         | −2.40 %                               |
| (b) camcorder      | 438            | 0.444                     | 0.485                             | +9.23 %                          | −2.12 %                               |
| (c) printer        | 746            | 0.683                     | 0.547                             | −19.91 %                         | +9.61 %                               |
| (d) receiver       | 1,363          | 0.316                     | 0.319                             | +0.95 %                          | −2.73 %                               |
| (e) videoprojector | 329            | 0.429                     | 0.462                             | +7.69 %                          | −0.14 %                               |
| (f) DVD player     | 456            | 0.415                     | 0.213                             | −48.67 %                         | +12.19 %                              |
| (g) TFT screen     | 1,019          | 0.455                     | 0.348                             | −23.52 %                         | +3.91 %                               |
| (h) mobile phone   | 2,168          | 0.394                     | 0.451                             | +14.47 %                         | −4.4 %                                |

extended survey on the usage of ontologies for the development and execution of user interfaces. In particular, they focus on user interfaces “whose visualization capabilities, interaction possibilities, or development process are enabled or (at least) improved by the employment of one or more ontologies”. They also propose a classification of this usage. For example, concerning the usage *domain*, ontologies can be used to represent the concepts from the real world (as, e.g., products in our case), IT system and their components, or users and their roles. Ontologies can also be classified according to their role in the lifecycle, whether being used for design or execution (or even both).

For recommendation processes, ontologies can be employed for user profiling as shown by Middleton et al. (2004), where the authors use machine-learning techniques to discover useful patterns in the users’ behavior and to improve the recommendation process. In our current approach, we do not perform any user profiling, but it could be integrated.

Klan and König-Ries (2011) present an interactive approach for service selection. This approach can be compared to our approach, if we assume, that such a service can be seen as a product. This approach also includes some heuristics in the selection process. One heuristic is, that the questions are ranked according to the maximum effect in the presented list. If the answer of one question reduces the list of possible services more than another question, it is ranked higher.

The relationship between domain models and interaction models has been studied in the past. Rosson (1999) presented the integration of task models (which represent the user interactions and are related to our Discourse Models) and object models (related to our Domain-of Discourse Models). Another example of integrating processes with high-level conceptual models is the field of Semantic Web Services (Sheth et al. 2006). Adding semantics to “classical” Web Services could be done with WSDL-S (Web Service Description Language-Semantics). In WSDL-S, Web Services descriptions are annotated with domain-specific (e.g., of one particular industry) and domain-independent ontologies (e.g., for general contracts or agreements).

## 2.7 Discussion

While our approach works and even results in recommendation processes applicable in practice, a few related questions should be discussed. In particular, is it necessary or, at least useful to generate a high-level model of such a process first? Although it seems possible to generate an operationalized process in one shot, we argue that a two-step approach is useful by analogy to compiler construction. Typically, intermediate languages are used on the way from a high-level programming language to machine code. These provide useful levels of abstraction for the compiler developers, and so does our high-level model of recommendation processes.

Still, the question remains, whether other kinds of languages or models could serve the same purpose, and possibly even better. This question cannot be answered with certainty as it stands, since our approach seems to be the very first along these lines.

As our discourse-based models are primarily used for specifying models on a high abstraction level for automated generation of user interfaces, how about the most often used approach for this purpose? Instead of our discourse-based models, task-based ConcurTaskTrees from Paternò et al. (1997) may be used for bridging the semantic gap between ontologies and user interfaces. ConcurTaskTrees facilitate modeling *tasks* and their causal and temporal relations. Such models are also being transformed into a user interface semi-automatically. However, we are not aware of any approach for generating ConcurTaskTrees out of ontologies. UsiXML (Faure and Vanderdonckt 2010) is an XML-based specification language for user interface design. It allows specifying a user interface at different levels of abstraction, from high-level task models (like ConcurTaskTrees) to the concrete code of a user interface. Also for UsiXML, we are not aware of any approach for generating UsiXML models out of ontologies. Since a recommendation process of the kind generated here primarily consists of pairs of questions and related answers, Communicative Acts as used in our discourse-based approach are an excellent fit for modeling them. In contrast, tasks would have to model questions and answers in the sense of corresponding interactions with a specific kind of user interface. So, while this would certainly be feasible, it appears to be less appealing than our approach.

Since this is about modeling processes, also languages for business process modeling may be used. Such languages like Business Process Model and Notation (BPMN) focus on the dynamics of such processes. However, BPMN appears to lack means for specifying the structure of domains sufficient for the content of recommendation processes.

From Discourse-based Communication Models, it is possible to generate general-purpose graphical user interfaces according to Falb et al. (2009) and even optimized ones for devices with small screens according to Raneburger et al. (2011). Contrasting them with the user interfaces generated for recommendation processes as explained above, it is clear that the latter are preferable in terms of usability. As a matter of fact, they have been successfully used for real-world application of these recommendation processes. They are special-purpose, however, and their overall appearance was predefined, while only the content has been generated automatically for the given structure and with many given heuristics.



## 2.8 Conclusion

We show that it is possible, from a given annotated product ontology, to generate dialogue-driven recommendation processes semi-automatically. The key reason is, that such a process primarily consists of questions and answers about exactly the products from such an ontology. Therefore, a template devised by us can be filled based on products in the ontology. In addition, the annotations in this ontology are key to select products for a related recommendation process based on their priority.

The semi-automatic generation requires much less effort than the manual creation. In addition, data from real-world deployment of this new and automated approach provide empirical evidence of its usefulness. For instance, in the real-world application at a large e-commerce shop platform with about 1,500 different customers, this approach increased the rate of customers who followed recommendations by 14 %. So, the generated processes seem to be competitive with processes manually created by human experts (according to the same overall strategy).

**Acknowledgements** This research has been carried out in the SOFAR project (No. 825061), partially funded by the Austrian FIT-IT Program of the FFG.

## References

- Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., & Vanderdonckt, J. (2003). A unifying reference framework for multi-target user interfaces. *Interacting With Computers*, 15(3), 289–308.
- Coral, C., Francisco, R., & Mario, P. (2006). *Ontologies for software engineering and software technology*. Berlin: Springer.
- Ertl, D., Kaindl, H., Arnautovic, E., Falb, J., & Popp, R. (2011). Discourse-based interaction models for recommendation processes. In *ACHI '11: proceedings of the 4th international conference on advances in computer-human interactions*.
- Falb, J., Kaindl, H., Horacek, H., Bogdan, C., Popp, R., & Arnautovic, E. (2006). A discourse model for interaction design based on theories of human communication. In *CHI '06: extended abstracts on human factors in computing systems* (pp. 754–759). New York: ACM.
- Falb, J., Kavalajian, S., Popp, R., Raneburger, D., Arnautovic, E., & Kaindl, H. (2009). Fully automatic user interface generation from discourse models. In *IUI '09: Proceedings of the 13th international conference on intelligent user interfaces* (pp. 475–476). New York: ACM.
- Faure, D., & Vanderdonckt, J. (2010). User interface extensible markup language. In *EICS '10: proceedings of the 2nd ACM SIGCHI symposium on engineering interactive computing systems* (pp. 361–362). New York: ACM.
- Guarino, N. (1998). In *Proceedings of the 1st international conference on formal ontology in information systems*. Amsterdam: IOS Press.
- Kaindl, H., Wach, E. P., Okoli, A., Popp, R., Hoch, R., Gaulke, W., & Hussein, T. (2013). Semi-automatic generation of recommendation processes and their GUIs. In *IUI '13: proceedings of the 2013 ACM international conference on intelligent user interfaces*.
- Klan, F., & König-Ries, B. (2011). A conversational approach to semantic web service selection. In C. Huemer & T. Setzer (Eds.), *Lecture notes in business information processing: Vol. 85. E-commerce and web technologies* (pp. 1–12). Berlin: Springer.
- Luff, P., Frohlich, D., & Gilbert, N. (1990). *Computers and conversation*. London: Academic Press.

- Mann, W. C., & Thompson, S. A. (1988). Rhetorical structure theory: toward a functional theory of text organization. *Text*, 8(3), 243–281.
- Middleton, S. E., Shadbolt, N. R., & De Roure, D. C. (2004). Ontological user profiling in recommender systems. *ACM Transactions on Information Systems*, 22(1), 54–88.
- Paternò, F., Mancini, C., & Meniconi, S. (1997). ConcurTaskTrees: a diagrammatic notation for specifying task models. In *Proceedings of the IFIP TC13 6th international conference on human-computer interaction* (pp. 362–369).
- Paulheim, H., & Probst, F. (2010). Ontology-enhanced user interfaces: a survey. *International Journal on Semantic Web and Information Systems*, 6(2), 36–59.
- Popp, R., & Raneburger, D. (2011). A high-level agent interaction protocol based on a communication ontology. In C. Huemer, T. Setzer, W. Aalst, J. Mylopoulos, N. M. Sadeh, M. J. Shaw & C. Szyperski (Eds.), *Lecture notes in business information processing: Vol. 85. E-commerce and web technologies* (pp. 233–245). Berlin: Springer.
- Raneburger, D., Popp, R., Kavaldjian, S., Kaindl, H., & Falb, J. (2011). Optimized GUI generation for small screens. In H. Hussmann, G. Meixner & D. Zuehlke (Eds.), *Studies in computational intelligence: Vol. 340. Model-driven development of advanced user interfaces* (pp. 107–122). Berlin: Springer.
- Rosson, M. B. (1999). Integrating development of task and object models. *Communications of the ACM*, 42(1), 49–56.
- Searle, J. R. (1969). *Speech acts: an essay in the philosophy of language*. Cambridge: Cambridge University Press.
- Sheth, A., Verma, K., & Gomadam, K. (2006). Semantics to energize the full services spectrum. *Communications of the ACM*, 49(7), 55–61.

Semantic Models for Adaptive Interactive Systems

Hussein, T.; Paulheim, H.; Lukosch, S.; Ziegler, J.;

Calvary, G. (Eds.)

2013, XII, 204 p. 58 illus., Hardcover

ISBN: 978-1-4471-5300-9