A Web-Based Filtering Engine for Understanding Event Specifications in Large Conceptual Schemas

Antonio Villegas, Antoni Olivé, and Maria-Ribera Sancho

Department of Service and Information System Engineering,
Universitat Politècnica de Catalunya – BarcelonaTech,
Barcelona, Spain
{avillegas,olive,ribera}@essi.upc.edu

Abstract. A complete conceptual schema must include all relevant general static and dynamic aspects of an information system. Event types describe a nonempty set of allowed changes in the population of entity or relationship types in the domain of the conceptual schema. The conceptual schemas of many real-world information systems that include the specification of event types are too large to be easily managed or understood. There are many information system development activities in which people need to understand the effect of a set of events. We present an information filtering tool in which a user focuses on one or more event types of interest for her task at hand, and the tool automatically filters the schema in order to obtain a reduced conceptual schema that illustrates all the elements affected by the given events.

Keywords: Large Schemas, Filtering, Event Types, Importance.

1 Introduction

The conceptual schemas of many real-world information systems include the specification of event types. An event describes a nonempty set of changes in the population of entity or relationship types in the domain of the conceptual schema. The sheer size of those schemas makes it difficult to extract knowledge from them. There are many information system development activities in which people need to understand the effect of a set of events. For example, a software tester needs to write tests checking that the effect of an event has been correctly implemented, or a member of the maintenance team needs to change that effect. Currently, there is a lack of computer support to make conceptual schemas usable for the goals of event exploration and event understanding.

The aim of information filtering is to expose users only to information that is relevant to them [1]. We present an interactive tool in which the user specifies one or more event types of interest and the tool automatically provides a (smaller) subset of the knowledge contained in the conceptual schema that includes all the elements affected by the given events. The user may then start another interaction with different events, until she has obtained all knowledge of interest. We presented the theoretical background behind this tool in [2].

S. Castano et al.(Eds.): ER Workshops 2012, LNCS 7518, pp. 383–386, 2012.

2 Events as Entities

We adopt the view that events are similar to ordinary entities and, therefore, that events can be modeled as a special kind of entities [3]. In the UML, we use for this purpose a new stereotype, that we call «event». A type with this stereotype defines an event type. Like any other entity type, event types may be specialized and/or generalized. This will allow us to build a taxonomy of event types, where common elements are defined only once.

The characteristics of events should be modeled like those of ordinary entities. In the UML, we model them as attributes or associations. We define a particular operation in each event type, whose purpose is to specify the event effect. To this end, we use the operation *effect*. The pre- and postconditions of this operation will be exactly the pre- and postconditions of the corresponding event. We use the OCL to specify these pre- and postconditions formally.

3 The Filtering Engine for Events

In this section we describe how the event specifications of a large conceptual schema can be explored by using our tool, which corresponds to the demonstration we intend to perform. The main idea is to extract a reduced and self-contained view from the large schema, that is, a filtered schema with the elements affected by the specification of a set of events.

Our filtering tool is developed as a web client that interacts with a web service following the SOAP protocol. The filtering web service we have implemented uses a customized version of the core of the USE tool [4] to access the knowledge of the large schema the user wants to explore. In our demonstration, we use the schema of the Magento e-commerce system, which contains 218 entity types, 187 event types, 983 attributes, 165 generalizations, and 319 associations [5].

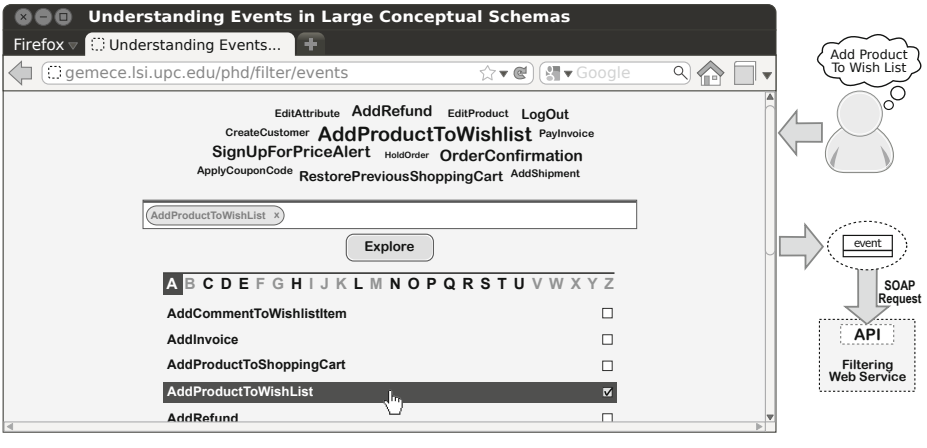

Fig. 1. Request of a user to our filtering tool

Figure 1 presents the main components of the filtering request. First, the user focus on a nonempty set of event types she is interested in according to a specific information need over the event specifications of a large schema. These events conform the input focus set our web-based filtering engine needs to start the process. To help the user on selecting the events of focus, our web client provides a word cloud with the names of the top event types according to their general relevance in the schema, and an alphabetical list with the names of all the event types. The user can select events from both components, or directly write the name of the event in the search bar placed between them. It automatically suggests event names while the user is typing. Once the request is submitted, the web client constructs a SOAP request and sends it to the API of the filtering web service with the focus set of events. As an example, the user focus on the event type *AddProductToWishList* in Fig. 1.

Figure 2 presents the main components of the filtering response. The web service of the filtering engine automatically obtains the corresponding pre- and postconditions for the events of the request. The OCL specification of these constraints references the elements from the large schema that are affected by the events of focus. Our service processes the OCL expressions of the previous constraints in order to extract all the elements that appear within their formal specification. Then, it puts them together with the event types of the focus set in order to create a filtered schema with the elements of both sets. The main goal consists of producing a filtered schema of minimum size. To achieve this, our tool projects the referenced attributes and relationship types used in the event specifications that are defined in the context of elements that are not referenced by the OCL expressions to entity or event types in the filtered schema, whenever possible. It also connects the elements of the filtered schema with generalizations according to the knowledge in the large schema [2].

As a result, the web service returns the corresponding filtered schema in JSON format through a SOAP response in less than one second. Then, we use an

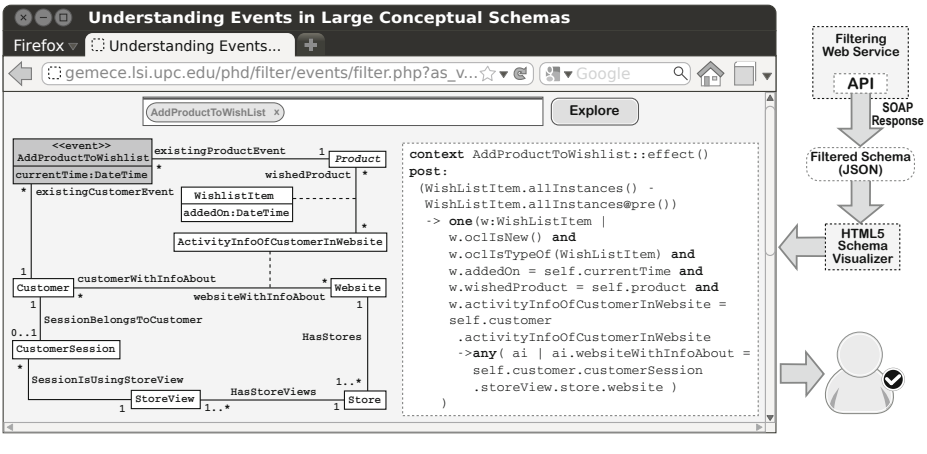

Fig. 2. Response of our filtering tool to the user

HTML5-based schema visualizer to graphically represent the filtered schema in UML, including the textual representation in OCL of the event effects. Thus, the user is able to interact with the resulting fragment of the large schema directly from the web browser. In the example, the user can easily understand that the effect of the event *AddProductToWishList* creates a new instance of *WishListItem* involving the product and the customer associated with the event, in order to represent the addition of a product into a wish list in Magento. The new instance is related to the activity information of the customer that wants to add the product to her wish list. It includes the customer session in the store view of the store owned by the website of the system. Note that the event is marked in gray in order to rapidly identify the focus of the filtered schema. Subsequently, the user can start a new filtering request if required.

4 Summary

We have presented a tool that automatically obtains a reduced view of the schema, which includes the relevant parts for the understanding of the events. Our implementation as a web service provides interoperability and simplifies the interaction with users. A preliminary version of the filtering tool can be found in `http://gemece.lsi.upc.edu/phd/filter/events`.

Our immediate plans include the improvement of our tool by providing traceability between the graphical representation of the elements within the filtered schema and their references in the OCL expressions that specify the effect of the events of focus. As a result, the user will be able to quickly understand the formal specification of the events and their impact in the schema.

Acknowledgements. Our thanks to Jose Maria Gomez for his work in the implementation of the schema visualizer. This work has been partly supported by the Ministerio de Ciencia y Tecnologia and FEDER under project TIN2008-00444/TIN, Grupo Consolidado, and by Universitat Politècnica de Catalunya under FPI-UPC program.

References

1. Hanani, U., Shapira, B., Shoval, P.: Information filtering: Overview of issues, research and systems. User Modeling and User-Adapted Interaction 11(3), 203–259 (2001)
2. Villegas, A., Olivé, A.: A Method for Filtering Large Conceptual Schemas. In: Parsons, J., Saeki, M., Shoval, P., Woo, C., Wand, Y. (eds.) ER 2010. LNCS, vol. 6412, pp. 247–260. Springer, Heidelberg (2010)
3. Olivé, À.: Definition of Events and Their Effects in Object-Oriented Conceptual Modeling Languages. In: Atzeni, P., Chu, W., Lu, H., Zhou, S., Ling, T.-W. (eds.) ER 2004. LNCS, vol. 3288, pp. 136–149. Springer, Heidelberg (2004)
4. Gogolla, M., Büttner, F., Richters, M.: USE: A UML-based specification environment for validating UML and OCL. Science of Computer Programming (2007)
5. Ramirez, A.: Conceptual schema of Magento. Technical report, Universitat Politècnica de Catalunya (2011), `http://hdl.handle.net/2099.1/12294`