APPLYING QUERY BY EXAMPLE IN OCL FOR PLATFORM-INDEPENDENT PROGRAMMING

Grzegorz Fałda, Wiktor Filipowicz, Piotr Habela, Krzysztof Stencel, Kazimierz Subieta
Polish-Japanese Institute of Information Technology, Warsaw, Poland

Krzysztof Kaczmarski
Warsaw University of Technology, Warsaw, Poland

Keywords: Database, Object-oriented, Query language, Visual, Query by example, Model driven architecture, UML, OCL.

Abstract: Precise modelling of behaviour is an area where programming meets modelling, and textual syntax competes with a visual one. By developing a UML based platform-independent framework, we aimed to find a visual syntax aid to make the language more approachable to stakeholders, while taking advantage of existing UML syntax intuitions and offering a truly higher level of abstraction. Our solution consists of seamlessly integrated UML Actions and the Object Constraint Language (OCL) as a database query language, featuring both a textual and a visual syntax. In this paper we describe a declarative, Query by Example (QBE)-based approach to visualizing OCL expressions over a UML object-oriented model instance, to be used inside of textual or visual imperative statements. Such visual OCL expressions can also be used as ad-hoc queries. The paper presents a choice of visual syntax and describes its underlying semantics.

1 INTRODUCTION

In the VIDE project (Falda et al. 2007, Falda et al. 2008, VIDE 2008) we aimed at investigating the capability of executable modelling in the spirit of Model Driven Architecture (MDA), basing on standardized, general-purpose modelling languages and applied to the area of enterprise applications involving persistent data sources. This led us to selecting OMG UML 2.1 as the basis of the research and supporting it with more powerful querying capability offered by OCL 2.0. The state of these specifications leaves many questions open. Particularly, no concrete notation for the finer level UML model elements (like Actions and Structured Activities) has been standardized. Furthermore, the use of OCL as a query language inside imperative constructs of UML has not been extensively investigated yet, hence research was needed to provide an answer whether OCL is suitable in this new role. Joining UML and OCL as a database programming language required resolving some ambiguities and shortcomings of those standards.

When extending a modelling language towards the ability to specify detailed behaviour, we enter the area that is usually covered by traditional, textual programming languages. Indeed, introducing diagrammatic notation may turn out to be an overkill for typical programming tasks, where textual code may be sufficiently readable for professionals and - especially – faster do develop (Ambler 2007). On the other hand, the characteristics of the VIDE project required a careful reconsideration of this attitude. The assumed role of the language – that is UML-based platform-independent modelling – puts it halfway between typical programming and traditional UML modelling. Hence, we had a strong incentive to look for visualisations that would correspond to traditional, broadly understood UML diagrams, more so than in the case of traditional programming language development.

In the course of the VIDE project at first we have developed a textual version of the language, incorporating OCL (Habela 2007, Habela 2008) for

FaÅĆda G., Filipowicz W., Habela P., Stencel K., Subieta K. and Kaczmarski K.
APPLYING QUERY BY EXAMPLE IN OCL FOR PLATFORM-INDEPENDENT PROGRAMMING.
DOI: 10.5220/0002791601790182
In *Proceedings of the 6th International Conference on Web Information Systems and Technology (WEBIST 2010)*, page
ISBN: 978-989-674-025-2

its expression part. It became clear that the main source of code complexity is mainly the expressions, which motivated the search for useful visualisations for them. The concept of an object-oriented version of the Query By Example (QBE) approach was chosen due to its intuitiveness and successful adoption of its relational variant, as well as because of the possibility of fitting it gracefully with UML by reusing the instance diagram notation. The visual expressions are intended to be used as ad-hoc queries, and for embedding them into the imperative constructs of the language.

The VIDE software has been implemented on top of ODRA (Adamus 2008), an object-oriented database management system for rapid application development. In this project OCL has been implemented on top of SBQL, an object query and programming language based on the Stack-Based Architecture (SBA). For this reason OCL implemented in VIDE inherits some of the advantages of SBQL, including strong static type checking and powerful query optimization methods.

In this paper we describe a declarative, object-oriented, QBE-based approach to visualizing OCL expressions over a UML model instance, to be used inside of textual or visual imperative statements.

2 VISUAL EXPRESSIONS IN OBJECT QUERY BY EXAMPLE

To improve the approachability and expressiveness of the language and at the same time to exploit the concepts of the well-known UML syntax, we have chosen to follow the QBE paradigm and adopt it to the object-oriented data model of OCL. The language design, called Object Query By Example (OQBE) went beyond the UML/OCL metamodel, providing instead some new, dedicated constructs based on the notion of *example of an object* (accompanied by *link example* and *attribute example*). The actual OCL expression is produced through the transformation of such an expression model. For representing the examples, the UML instance diagram syntax has been adopted.

The graph of an object, link or attribute example makes it possible to declaratively specify an expression that includes a graphical representation for essential query operators: joins, selections and projections. In this sense the intuitions from the traditional QBE solutions are maintained. However, as will be explained, OQBE introduces a number of new important solutions specific to the object-oriented data model and to OCL. These features include:

- Support for both object identity and value comparisons.
- Specifying expressions that return complex results of nested structures (*tuples* in the OCL terminology).
- A dedicated construct for representing the general quantifier (*->forAll* iterator operation in OCL).
- Object retrieval from class *extents* or from variables / attributes visible in the environment of query evaluation.

To illustrate the OQBE constructs, we use a sample model consisting of 7 classes. It represents a simple auction site, including the notions of a User, Auction, Bid, Purchase, Comment and Category. The AuctionSite class with the «module» stereotype represents the application's global data store and functionality. The ad-hoc queries shown in the subsequent part of this section are to be evaluated in that global environment.

Object examples connected with link examples can represent respective data structure examples. If more than one mutually unconnected part of the example diagram exists, they are considered to represent a Cartesian product of their results. An attribute example may play different roles:

- **Predicate** – if the attribute itself or the result of an operation applied to it (including various kinds of comparisons) provides a Boolean value and no specific flag is attached to the attribute example.
- **Output Element** – if an attribute example has the *output* flag attached.
- **Sorting Criterion** – if the *sort* flag with necessary properties (priority, order) is attached (see the figure below for example of its usage).

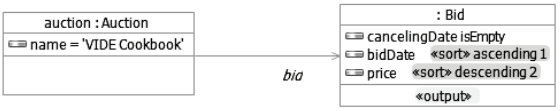

Figure 1: OQBE expression involving selection, navigation, sorting and projection ("get non-cancelled bids for 'VIDE Cookbook' and order them according to date and price").

Figure 1 shows a simple OQBE expression that involves two object examples connected with a link example, including two predicates based on attribute examples and using further two attribute examples to determine the result sorting. The result of the expression is constructed through the projection from the whole example structure onto the element

marked with the output flag. In this case the type of the expression result will be **OrderedSet<Bid>**.

Before we present a detailed algorithm in the next section, let's describe the evaluation of an expression declared in OQBE as a sequence of the following steps (disregarding optimisations):

1. Constructing expressions that retrieve objects represented by the entry points of the graph.
2. Performing joins according to the navigation through link examples.
3. If more than one graph has been drawn, constructing a Cartesian product of their results.
4. Performing selections according to the constraints included in the examples.
5. Sorting the result.
6. Building the final result by performing a projection into one or more fields indicated by *output* flags.

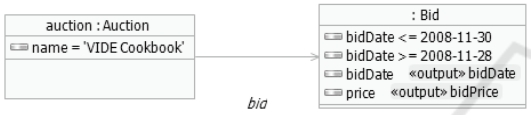

Figure 2: OQBE expression with a tuple result consisting of two named fields ("get dates and prices of bids placed for 'VIDE cookbook' between the given dates").

The *output* flag can be applied to an object example as a whole (in that case it marks a whole object to constitute an element of the result of the expression) or to an attribute. More than one *output* flag can be used in a single expression. In that case however, the result is assumed to be of a *Tuple* type and hence the particular flags need to be accompanied with appropriate tuple field names, as shown in Figure 2.

Figure 3: OQBE expression including an embedded OCL code ("get names and winning bids for auctions active in the period specified").

Since using single names and operations on them can be too limiting for some of the abovementioned applications of an attribute example, the language provides a second similar construct, where instead of an attribute name, an arbitrary OCL code can be embedded. Figure 3 shows this construct used for operation invocation.

An embedded OCL code is evaluated in a way analogous to the plain attribute example names. This feature allows us to overcome some technical limitations of the built prototype tool (as in the above simple case, i.e., by providing a dedicated support for parameter-less method calls), however it can also be useful as a shortcut or when an expression is too complex to be fully visualized (for example, multiplication or adding several values stored in attributes of the same or a different object and returning them as the expression result).

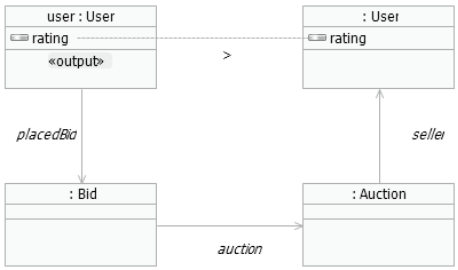

Figure 4: OQBE expression involving the comparator construct ("get users who used to bid for items sold by users with a lower rating").

Although the readability considerations limit the usage of connections between attribute examples, a simple construct called *Comparator* was introduced in order to represent comparisons between attribute or object examples. Its usage is shown in Figure 4.

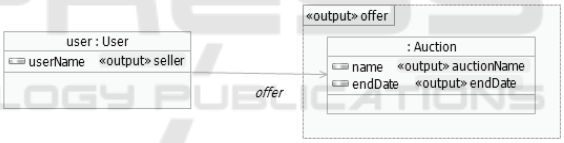

Figure 5: OQBE expression with a nested structure result ("get user names together with sets of names and end dates of their auctions").

One of the features that makes OQBE different from traditional QBE solutions based on the relational model, is the availability of nested structures in the construction of expression results. In the textual OCL this is realized by nesting subexpressions inside an explicit Tuple constructor. In OQBE a tuple is constructed implicitly by providing names to the *output* flags. Hence, the only element that was necessary to achieve nested result structures was a visual region inside which a respective part of the diagram (containing its own output flags) could be nested. Note that the nested output region requires providing the field name for it. The result type of the expression depicted in Figure 5 would be the following:

```
Set<Tuple{ userName : string,
offer = Set<Tuple {name : string,
endDate : Date }> }>
```

3 CONCLUSIONS

In this paper we have outlined the concept of the *object query by example* solution which serves for visualising OCL expressions in the VIDE UML-based action language. Combining OCL with UML Actions and Activities provides a standard-compliant and powerful language for processing object-oriented data structures. Contrary to UML as a whole however, OCL is known by relatively few professionals, and may pose a barrier for a broader adoption of a UML based programming language. Hence we have proposed a QBE-inspired visual language that offers a yet higher level of abstraction compared to OCL. It builds upon UML syntax metaphors and fits the visual syntax proposed for the imperative constructs of the language (actions and activities). The solution includes a concrete syntax inspired by UML instance diagrams, underlying metamodel for object examples, the algorithm for OCL code generation and optimisation by rewriting. We have also developed a prototype implementation within the Eclipse framework. The prototype is a fully-fledged IDE which facilitates designing queries and running them directly on the target platform. In the current version we have chosen, designed and implemented a set of features that is not exhaustive, but on the other hand, avoids the complexity that would undermine the intuitiveness of the syntax and the general usability of the tool.

Several further features have been identified and are considered for enhancing the next release of the prototype.

The evaluation workshop conducted with students during the VIDE project (VIDE 2008) seems to confirm that the syntax is intuitive for inexperienced users who have UML background. We have also observed that OQBE allowed users to construct the queries whose complexity in textual OCL and SQL constituted a barrier.

We are currently approaching a more direct integration of UML-based modelling with our ODRA platform, with visual expression construction with OQBE being a part of the environment. Note that even in case the resulting query code does not need to be reviewed by an application developer, its simplicity and clarity is important for performance reasons. Hence, a work on the optimisation of the generated queries has been also initiated.

The underlying concept of visual querying is also more generally applicable. We are going to take advantage of this fact when designing a generic data exploration tool (involving navigation, querying and intermediate results storage (basket)) for a SQL database which is going to be used for motion data search. Further important details of our solution including the semantics, implementation solutions, intended usage and the design of the diagram-to-OCL code generator, remaining out the scope of this short paper, have been described in other works available at (VIDE 2009).

REFERENCES

Adamus, R., Daczkowski, M., Habela, P., Kaczmarski, K., Kowalski, T., Lentner, M., Pieciukiewicz, T., Stencel, K., Subieta, K., Trzaska, M., Wardziak, T., Wiślicki, J., 2008. Overview of the Project ODRA. In *Proceedings of the First International Conference on Object Databases, ICOODB 2008*. pp.179-197.

Ambler, S., 2007. *A Roadmap for Agile MDA*. Ambysoft.

Falda, G., Habela, P., Kaczmarski, K., Stencel, K., Subieta, K., 2008. Executable Platform Independent Models for Data Intensive Applications. In *Computational Science - ICCS 2008, 8th International Conference, Kraków, Poland, June 23-25, 2008, Proceedings, Part III*. Springer, pp. 301-310.

Falda, G., Habela, P., Kaczmarski, K., Stencel, K., Subieta, K., 2007. Platform-independent programming of data-intensive applications using UML. In *2nd IFIP Central and East European Conference on Software Engineering Techniques, CEE-SET 2007*, Springer, pp. 103-115.

Habela, P., Kaczmarski, K., Stencel, K., Subieta, K., 2007. Implementing OCL as a Database Query Language. In *On the Move to Meaningful Internet Systems: OTM 2007 Proceedings, Part I.* Springer, pp. 17-18

Habela, P., Kaczmarski, K., Stencel, K., Subieta, K., 2008. OCL as the Query Language for UML Model Execution. In *Computational Science - ICCS 2008, 8th International Conference, Kraków, Poland, June 23-25, 2008, Proceedings, Part III*. Springer, pp. 311-320.

VIDE, 2008. VIDE Visualize all moDel drivEn programming. Industrial Use Cases and Examples. http://www.vide-ist.eu/reflib/usecases.html

VIDE, 2009. VIDE Visualize all moDel drivEn programming. *Project website* http://www.vide-ist.eu