**OCL 2011** Discussion session – 29 June 2011

**Participants:**

All the participants in the OCL workshop participated in the discussion. Among others, the discussion included contributions from the following attendees: Achim Brucker, Dan Chiorean, Robert Clarisó, Birgit Demuth, Martin Gogolla, Lars Hamann, Axel Uhl, Class Wilke, Ed Willink and Burkhart Wolff.

**Summary:**

The discussion focused on several topics:

- How can we improve existing tool support for OCL?

  Here the discussion focused on the need for a commonly supported exchange format that allows interoperability and the comparison among tools. Different alternatives were examined (textual syntax, pivot model) and the technical issues for each one were discussed. No consensus was finally reached on the best alternative. However, this was identified as an important issue for the OCL community where further work is required in order to achieve agreement and a working solution.

- What should be the goal of OCL? How can it be improved?

  This discussion considered whether the concepts in the OCL language should be defined in such a way that they are easier to implement, rather than being general enough to support any underlying language or platform. The argument was that this would facilitate and improve tool-support and extend the use of OCL in practice. Another view on this topic was that "closer to implementation" should be achieved through extensions, e.g. component libraries, rather than changes to the core language. As the use of component libraries is not a common practice, this creates an interesting direction for future work.

**Meeting minutes:**

Achim Brucker: Currently, different OCL tools use different formats, they cannot be compared. Comparing tools means remodeling again in different GUIs or IDEs, not practical. The existence of a commonly supported format would be important.

Ed Willink: Currently XMI is not a good solution for interchange. But there is progress on the concrete syntax.

Burkhart Wolff: During the development of HOL-OCL, there were 3 changes to the OCL spec. This meant having three versions of the same tool. Even the concrete syntax had its problems.

Martin Gogolla: The reason why USE describes the class diagram using a textual format is because of the difficulty of finding a common UML exchange format.

Discussion about the relationship among UML and OCL: Two views on the topic were "Collections and OCL can be explained without reference to a class diagram" vs "UML context is important to provide meaning to OCL".

Robert Clarisó: Maybe only a core for UML is required, the minimum required to provide context to OCL.

Ed Willink: This is looking it from the wrong angle. It is model + OCL that gives meaning. Layering of standards is required: we don't need to define a new UNICODE to encode characters; so we already have a standard for describing models (UML) - let's use that.

Dan Chiorean: Constraints are very important. It is necessary to show students (the next generation of developers) that OCL is an integral part of a model, that it can be used for integrity, querying, simulation, …

Summary of the discussion so far: is OCL a "textual annotation language" or "UML add-on"?

Ed Willink: "Complete OCL" is actually working as an interchange format, even though people prefer a textual format. The problem is that UML has no popular textual syntax for UML.

Burkhart Wolff: that would be a very valuable contribution, a syntax which provides **only** access to key items. In USE only basic features such as inheritance and associations are defined.

Ed Willink: Yes, this was my proposal during the talk...

Axel Uhl: Some concepts do not appear in EMOF, e.g. bidirectionality.

Ed Willink: Yes, EMOF throws some information away.

Achim Brucker: We should accept that we will never have a tool that supports 100% of the OCL standard.

Birgit Demuth: What is the right privot model? Which concepts should it support?

Ed Willink: there should be an intermediate model (we can discuss which one is best).

Lack of consensus on this topic (E. Willink: OCL should be aligned with UML – use a pivot model) vs (B. Wolff: a textual format is desirable – XMI is not enough)

CDO (Partial models) – it is available today (E. Willink) / not available now (B. Wolff)

Claas Wilke: there will be an important effort to reach consensus on what should appear in an intermediate format and what its syntax should look like.

Axel Uhl: if the final goal is being able to compare tools, more effort is required: we need to define how the comparison will be performed, common APIs, test cases, …

Ed Willink: everybody is happy with Ecore, except for the corners (A. Brucker: I don't even know any other metamodel facility, e.g. MDR is dead)

Lars Hamann: if OCL was SQL, then we need the equivalent of a “CREATE TABLE”.

Ed Willink: agree, we need to generalize the UML object constructor so that it can create any object.

Discussion by B. Wolff, A. Uhl and E. Willink on potential problems with side-effects when considering the creation of objects.

B. Wolff: what should be the goal of OCL?

E. Willink: OCL is a specification language rather than an implementation language.

Discussion about platform independence: should we define a behavior for OCL which is platform independent as Java? (e.g. define a specific bit-width for integers, use bounded-precision floats instead of reals, …)

B. Wolff: mathematical constructs are already platform-independent.

A. Uhl: Another similar question. Polymorphism not defined precisely, so that different languages and semantics can be accommodated. This is a nightmare from a tools perspective. It is also a reason why it is so difficult to apply OCL in practice and get other people to apply OCL in real-world examples.

E. Willink: OCL is an implementable specification language, with implementation you will reach a limit to the precision of this numbers.

C. Wilke: is it an implementation issue or a specification issue?

Discussion by several: is “the mathematical notion of real numbers” implementable? Is it an important question?

Axel Uhl: The current OCL specification is very unpragmatic. It should be something which is easily implementable.

B. Wolff: If they are required, we should add specific implementations of data types as additional libraries. But the default types are ok as they are.

Subclassing the OCL “Real”s may not be feasible, i.e. it may break the operation contracts.

L. Hamann: should we consider a “pragmatic extension” of OCL with these concepts that facilitate implementation? Kind of like the Calendar class for Java.

Several: there is very little tradition of “libraries” of OCL components.

R. Clarisó: Currently OCL is an idealized constraint language. This is good for expressing constraints, but it seems to be a problem when implementing the concepts (i.e. validating the constraints, generating code, verification, etc.). Shouldn’t we attempt to make the language more usable?

B. Wolff: What we need is a clean mathematical core + a set of machine-dependent concepts implemented as extensions. Otherwise, we need to start “hacking”.