Object Constraint Language

Arif Khan

On September 1, 1997, a standard for object oriented analysis and design was submitted to the Object Management Group (OMG). This proposal is best known under its name UML 1.1, the Unified Modelling Language. UML includes common constructs for object oriented modelling like class models, state machines, use cases, and collaborations.

The Object Constraint Language is an extension to the latest version of the Unified Modelling Language (UML). It is a formal specifications language that has an easy understandable syntax and semantics. OCL is defined as the expression language for the Unified Modelling Language (UML). To understand OCL, the component parts of this statement should be examined. OCL has the characteristics of being an expression language, a modelling language and a formal language. Each of these gives its different properties that provide us with a very useful composite product.

OCL is a pure expression language, therefore, an OCL expression is guaranteed to be without side effect. It cannot change anything in the model. This means that the state of the system will never change because of an OCL expression, even though an OCL expression can be used to specify such a state change (e.g., in a post-condition). All values for all objects, including all links, will not change. Whenever an OCL expression is evaluated, it simply delivers a value.

OCL is a modelling language, not a programming language so it is not possible to write program logic or flow-control in OCL. You especially cannot invoke processes or activate non-query operations within OCL. Because OCL is a modelling language in the first place, not everything in it is promised to be directly executable. As a modelling language, all implementation issues are out of scope and cannot be expressed in OCL. Each OCL expression is conceptually atomic. The state of the objects in the system cannot change during evaluation in any way.

OCL is a formal language where all constructs have a formally defined meaning. The specification of OCL is part of the UML specification, and it is available from IBM and the OMG. The need for a formal language is that in object-oriented modelling, a graphical model, like a class model, is not enough for a precise and unambiguous specification. UML is useful as a way of communicating ideas from one person to another. OCL allows us to help this by making the model more concrete and precise. There is a need to describe additional constraints about the objects in the model and although such constraints are often described in natural language practice has shown that this will always result in ambiguities. To write unambiguous constraints so-called formal languages have been developed.

OCL is an improvement over its competitors in that traditional formal languages are useable only to persons with a strong mathematical background, but difficult to use for the average business or system modeller. OCL has been developed to fill this gap. OCL is a formal language, which remains easy to read and write. OCL allows expressions using simple logic operations, first order logic, and also set operations. First order logic allows a wide range of expression allowing universal quantifiers leading to generation of some powerful expressions.

OCL deals in 4 main types of constraints; Invariants, preconditions, post-conditions, and guards.

Invariants are constraints that apply to all instances of a class. They must all always evaluate to true. The simplest form of invariant is a constraint on an instance variable. Placing an invariant on an instance variable of a class will entail that all instances of that class will employ a restriction upon that variable. For example if we have a Customer class that contains a variable for the age of each Customer, we can limit this to only over 18s by simply adding an expression in OCL. Invariants can apply across associated classes as well as on simple classes. With collections of classes that have varying multiplicity set operations are used to specify constraints upon them. Constraint can be applied to the multiplicity of classes as well. Multiplicity does outline how many instances of a class can be created, but through OCL these constraints can be further formalised.

Pre-conditions are conditions that need to be true at the moment the operation is to be executed. A precondition is like a prerequisite to an operation. Preconditions can specify any constraints on the object, or objects in question. The context of any constraint has to be made explicitly clear when its outlined in OCL. Typical pre-conditions are checking if one object is already associated to another before associating the two. We do not want to make an association if one already exists, so a precondition can be specified to check that one does not already exist.

Conversely a post-condition is something that must be true the moment an operations ends. Using again the example of creating an association between two objects, a precondition can be used to check that once the association has been created that the two are associated.

A guard is a condition that must be true before when a state change occurs. These can be added to state transition diagrams in the UML. A set of constraints must be met before an object can shift from one state to another and this set is referred to as a guard. In the case of a vending machine, for instance, we do not want it to go to a state of dispensing the item until the required money has been placed in the machine.

There is a point to be made about constraints and inheritance. In object orientations and hierarchies it is a rule that the classes at the bottom end of the hierarchy, the ones that are the concrete classes, are always more specialised and more solid than the abstract classes at the top. The lower down the hierarchy we go the more precise the description of the class. This principle is continued with constraints, in that a subclass can only strengthen the constraints placed upon its superclass, not weaken them in anyway. In practice subclasses inherit the constraints of the super classes and can have added constraints.

Importantly OCL 1.0 gives no information about what to do if a constraint is broken. Constraints are just specified, and should not be broken, but OCL 1.0 does not allow any action to be taken if a constraint is broken. As OCL is just an addition to a modelling language it could be argued that action does not need to be taken, the model will just move to an invalid state. Kleppe and Warmer, however, proposed including “Actions” to the OCL at the UML conference in 2000.

A submission by a German company, Klasse Objecten, has been made for OCL version 2.0 and it is likely to be included in the UML 2.0 which is currently in production. This was in response to the UML 2.0 Rfp (Request for proposals) made at the 2000 UML conference where they requested people to submit formal proposals for additions to the UML for version 2.0.

The Object Constraint Language is a fairly new creation and a useful addition to the UML toolkit. Its development seems set to continue and it should soon be a formal part of the UML, rather than just a UML extension mechanism.