

OCL (1.4)

- Object Constraint Language
- Official part of UML
- Strongly typed, declarative specification of system properties
- Assertions + collection classes + UML diagram navigation
- Supported by Rational Rose, ArgoUML, Poseidon/Octopus, Enterprise Architect

Why?

- UML diagrams, as a graphical language, are limited in what they can express
 - Structural relationships, behavioral descriptions
 - It needs a mechanism for specifying semantics
- OCL provides
 - Class invariants
 - Operation pre and post conditions
 - Guards on state-machine transitions

OCL Overview

- Pure expression language
 - Focus on values; no side effects expressible
- Declarative not procedural
 - No control flow mechanisms
- Strongly typed
 - Built-in types plus types introduced in diagrams
- Highest level mechanism is the *constraint*

Constraint Format

```
context <identifier> <constraintType>:  
<Boolean expression>
```

- **context** is a keyword indicating the start of a new constraint
- <identifier> is a class or operation name
 - The current position from which relative path names are derived
- <constraintType> is **inv**, **pre**, or **post**
- <Boolean expression> is usually an equation asserting a property

Invariants

- Statement of a property that is always true
 - Except perhaps in the middle of the execution of an operation
- Expresses a key system requirement
- **inv** keyword
- Might be an essential relationship among the values of the attributes of an object
- Might express a relationship between classes
- *The student registration system never allows a student to be registered for two classes that meet at the same time*

Pre and Post Conditions

- UML operation semantics can be expressed using **pre** and **post** condition constraints
 - The **pre** condition says what must be true for the operation to meaningfully execute
 - The **post** condition expresses what is guaranteed to be true after execution completes
 - About the return value
 - About any state changes (e.g. instance variables)
- *The argument to the square root routine must be non-negative*
- *The square of the computed result must equal the argument*

OCL Built-in Primitive Types

type	values	operations
Boolean	true, false	and, or, xor, not, implies, if-then-else
Integer	1, -5, 2, 34, 26524, ...	*, +, -, /, abs()
Real	1.5, 3.14, ...	*, +, -, /, floor()
String	'to be or not to be...'	toUpper(), concat()

Keywords

keyword	description
inv, pre, post	Introduces constraints
if-then-else-endif	Conditional expression
not, or, and, xor, implies	Boolean operators
package, endpackage	Package
context	Modifies use of names
def	Global definition
let, in	Local definition

let Clause

- A `let` clause is a way of introducing an abbreviation
- It has two parts, both expressions
 - The first part between the (`let` and the `in`) indicates one or more names and binds values to them
 - The second part is like a local block; it limits the scope of where the bindings applies

Navigation

- OCL constraints are associated with class model diagrams
 - They can also be associated with statecharts
- Each constraint specifies a context
 - A specific class or method relative to which other diagram elements can be designated
- Navigation is done by walking through a diagram from the context class to another diagram element using intermediary relationship lines and class rectangles
 - Each step adds a name to a list, separated by periods

Multiplicity

- When an association has multiplicity greater than one, the result of a traversal is a collection (set, bag or sequence)
- If an operation is performed on that association, the -> symbol is used, rather than the period, to separate the collection from the operation

Collections

- OCL features four kinds of collections that support associations with many-to-many and one-to-many multiplicities
- The parent class **Collection** provides features that hold true of all collections
 - **size**, **includes**, **count**, **includesAll**, **isEmpty**, **notEmpty**, **sum**, **exists**, **forAll** and **iterate**
- **Set**, **Bag**, (multiset) and **Sequence** (ordered multiset) provide specialized operations

Other OCL Features

- Local (**let**) and global (**def**) definitions
- Enumerations
- Automatic flattening
- Access to the UML metamodel

Changes in OCL 2.0

- Use to express queries on UML models
- Tuple expression for dealing with structs/ records
- Message expression denoting the sending of a signal or the invoking of an operation