Object Constraint Language

Martin Nečaský
Dept. of Software Engineering
Faculty of Mathematics and Physics
Charles University in Prague
Incompleteness and Ambiguities

- UML (class) schemas typically do not provide all relevant details (e.g., constraints, pre-conditions, post-conditions)

- more information is required, it can be specified in a form of
  - notes/documentation in natural language
    - ambiguities but easy to understand by average business people or software engineers
  - formal languages
    - unambiguous but usable only to persons with strong mathematical background
Incompleteness and Ambiguities

- What is on your mind when you see this UML class diagram? I.e.:
  - What constraints apply?
  - What constraints are already unambiguously expressed in the diagram?
  - What constraints are not expressed?
    - How would you express them?

Formal Foundations of Software Engineering

```java
class Example_01
Person
+ name: String
+ number: String
+ salary: Int
+ getCurrentWork(KindOfWork) : Document
  + getWorker 1..*
+ project 0..*
+ authoredDoc 0..*
+ author 1..*
+ output 0..*
+ project 1
+ reviewedDoc 0..*
+ reviewer 0..2

Project
+ name: String
+ startDate: Date
+ endDate: Date
+ teamSize: Int
+ publishDocument(Document) : void
  + output 0..*

Document
+ serialNumber: Int
+ pages: Int
+ title: String
+ estimatedWorkHours: Int
+ status: DocStatus
+ project 1
+ reviewDocument(Document) : void
+ increaseSalary(Int) : void
+ isReviewer() : Boolean
```
Incompleteness and Ambiguities

- The start date of a project must be before the end date.
- A document with less than 8 estimated working hours cannot have more than 1 author.
- A person can be either an author or reviewer of a single document but not both.
- A person can be an author of a document only if that document is an output of his or her project.
- The serial number of a document must be unique in a project.
- A document can be published only when it is finished.
Constraints in UML

- UML constraint is a condition or restriction attached to one or more elements expressed in a natural language or machine readable notation
  - boolean expression that restricts the extension of the associated elements beyond what is imposed by the other UML constructs applied to that elements
Constraint Formal Model

- **constraint context**
  - the constraint is evaluated in a given context
  - determines when the constraint is evaluated
    - e.g., operation pre and post conditions

- **constrained elements**
  - all elements constrained by the constraint

- **name**
  - optional
Constraints in UML – Example

- **Person**
  - name: String
  - number: String
  - salary: Int

- **Project**
  - name: String
  - startDate: Date
  - endDate: Date
  + publishDocument(Document) : void

- **Document**
  - serialNumber: Int
  - pages: Int
  - title: String
  - estimatedWorkHours: Int
  - status: DocStatus

«invariant»
{The start date of a project must be before the end date.}

«invariant»
{A document with less than 8 estimated working hours can not have more than 1 author.}
Object Constraint Language (OCL)

- not a procedural language
  - specification and declarative language
- extension to UML
- strongly typed language
  - types defined by UML diagrams
  - predefined types:
    - Integer, Boolean, String, Real, UnlimitedInteger
    - Set, OrderedSet, Bag, Sequence
- functional language
  - no side effects
Kinds of Expressions

- initial values
- derivation rules
- operation pre-conditions, post-conditions, bodies
- invariants
Initial Values

context TypeName::PropertyName: Type
init: -- Expression representing the initial value

defines that the initial value of TypeName::PropertyName is equal to the value of the Expression
- the initial value is the value being assigned at the moment of the creation
- the type of the initial value must conform to Type

PropertyName is an attribute or association end
- if attribute then it must be owned by TypeName
- if association end then it must be owned by TypeName, or TypeName must be the context of PropertyName

NOTE: What is context?
context Document::status
init: DocStatus::New
context Project::output : Set(Document)
init: Set{]
Derivation Rules

context TypeName::PropertyName: Type
derive: -- Expression representing the derivation rule

- declares that the value of TypeName::PropertyName should always be equal to the value of the Expression
  - the derivation rule is a kind of invariant
  - the type of the derived value must conform to Type
-(PropertyName is an attribute or association end
  - if attribute then it must be owned by TypeName
  - if association end then it must be owned by TypeName, or TypeName must be the context of PropertyName
context Project::teamSize
derive: self.worker->size()
context Project::currentReviewer : Set(Person)
derive: output->select(status = DocStatus::Review)
        .reviewer->asSet()
Operation Pre- and Post-conditions

context TypeName::OperName(p1 : Type1, ...): ReturnType

pre: -- pre-condition Expression
post: -- post-condition Expression

- pre-condition must be true when the operation starts its execution
  - otherwise the operation will not be executed
- post-condition must be true when the operation ends its execution
  - otherwise the operation has not executed correctly
  - result – reserved word representing the result of executing the operation
  - @pre – reserved property suffix representing the previous value of the property
Operation Pre- and Post-conditions

Context: Project::publishDocument(d:Document)

Pre: self.output->includes(d) and
d.status = DocStatus::Finished

Post: d.status = DocStatus::Published
**context** Person::reviewDocument(d:Document)

**pre:**
- self.reviewedDoc->excludes(d) and
- self.project.output->includes(d) and
- d.status = DocStatus::ToReview and d reviewer->size()<2

**post:**
- self.reviewedDoc->includes(d) and
- d.status = DocStatus::UnderReview
Operation Pre- and Post-conditions

context Person:::increaseSalary(s:Integer): void
pre:    -- none
post:   salary = salary@pre + s
context Person::isReviewer() : Boolean
pre:   -- none
post: result = (self.reviewedDoc->size() > 0)
query operations can be fully declared by specifying their result in a single expression

- query operation = does not have any side effect, no change to the extension
context Person::getCurrentWork(k: KindOfWork) : Set(Document)

body: if k = KindOfWork::Writing
    then self.authoredDoc->select(status = DocStatus::InProgress)
    else self.reviewedDoc->select(status = DocStatus::Review)
Invariants

context TypeName
inv: -- invariant Expression

- invariant declares a condition which must be true upon completion of the constructor and completion of every public operation
- not necessarily true during the execution of the operations
context Project

inv: self.startDate -> isBefore(endDate)
context Document

inv: self.estimatedWorkHours ≤ 8 implies self.author.size() ≤ 1
context Person

inv: self.authoredDoc -> excludesAll(self.reviewedDoc)
context Person

inv: self.authoredDoc.project
    ->excludesAll(self.reviewedDoc.project)