Verified Visualisation of Textual Modelling Languages

Fintan Fairmichael[1] and Joseph Kiniry[2]

[1]School of Computer Science and Informatics
University College Dublin

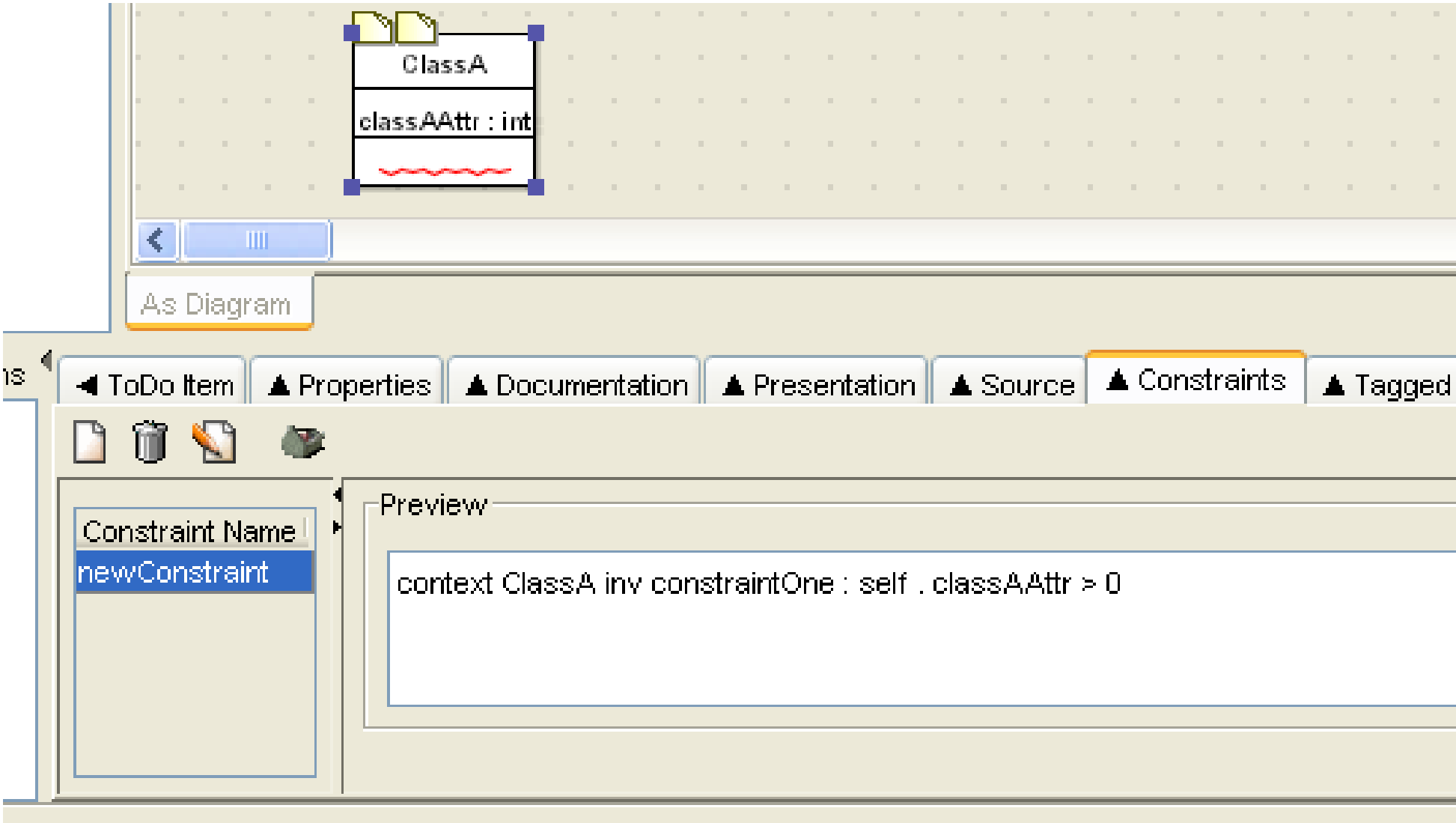
[2]ITU Copenhagen

October 2010 / Textual Modelling

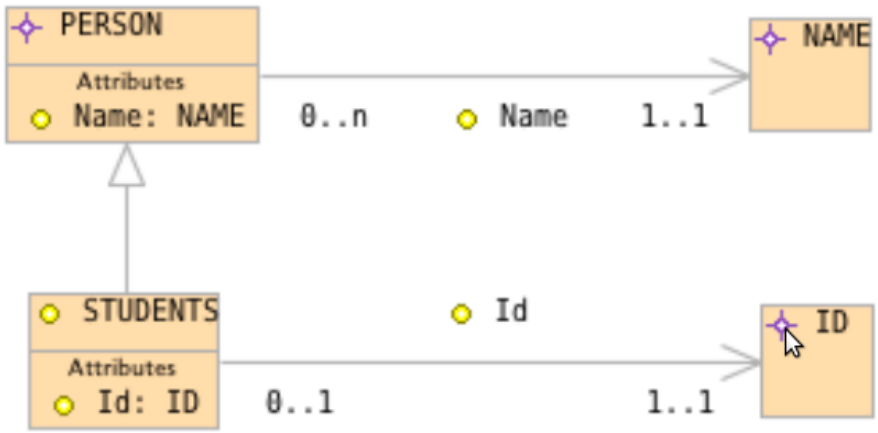
What's it all about?

- There is no shortage of modelling languages or tools.
- Some are purely textual, some purely graphical, and some have both.
- When there are both textual and graphical forms the semantic consistency between these is important.
- Can we allow simultaneous editing of both forms without worrying about consistency?

ArgoUML + OCL


ClassA
classAAttr : int
As Diagram
ToDo Item
Properties
Documentation
Presentation
Source
Constraints
Tagged
Constraint Name
newConstraint
Preview
context ClassA inv constraintOne : self . classAAttr > 0

UML-B



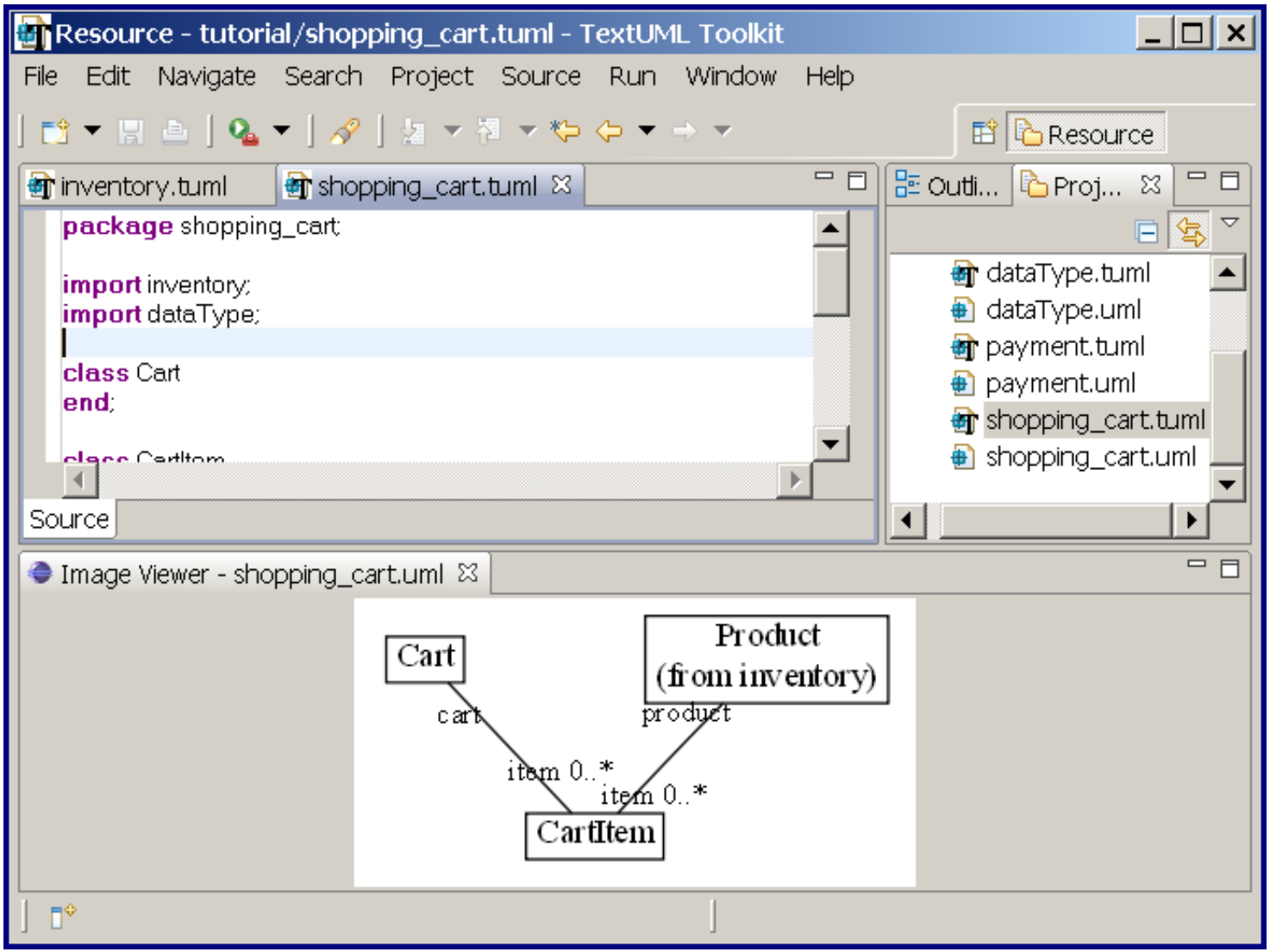

```
context c

constants STUDENTS // classType instances
          Id // attribute of STUDENTS
          Name // attribute of PERSON


sets ID // ClassType
     PERSON // ClassType
     NAME // ClassType


axioms
  @type STUDENTS ∈ P (PERSON)
  @type Id ∈ STUDENTS ⤔ ID
  @type Name ∈ PERSON → NAME
end
```

TextUML

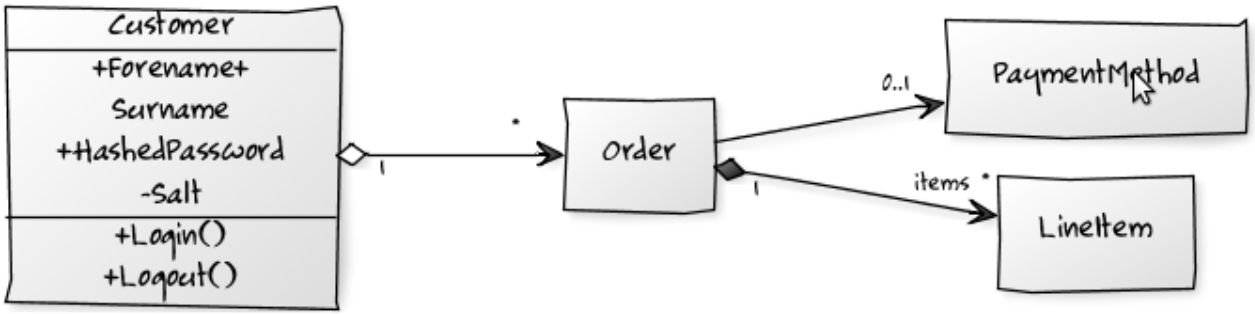

Resource - tutorial/shopping_cart.tuml - TextUML Toolkit
File Edit Navigate Search Project Source Run Window Help
Resource
inventory.tuml
shopping_cart.tuml
Outli...
Proj...
package shopping_cart;
import inventory;
import dataType;
class Cart
end;
dataType.tuml
dataType.uml
payment.tuml
payment.uml
shopping_cart.tuml
shopping_cart.uml
Source
Image Viewer - shopping_cart.uml
Cart
Product
(from inventory)
cart
product
item 0..*
item 0..*
CartItem

yUML.me

```
# Cool UML Diagram
[Customer|+Forename+;Surname;+HashedPassword;-
Salt|+Login();+Logout()]+1->*[Order]
[Order]++1-items >*[LineItem]
[Order]-0..1>[PaymentMethod]
```

Generate the Diagram!

Use **NEW Features** on the BETA draw page



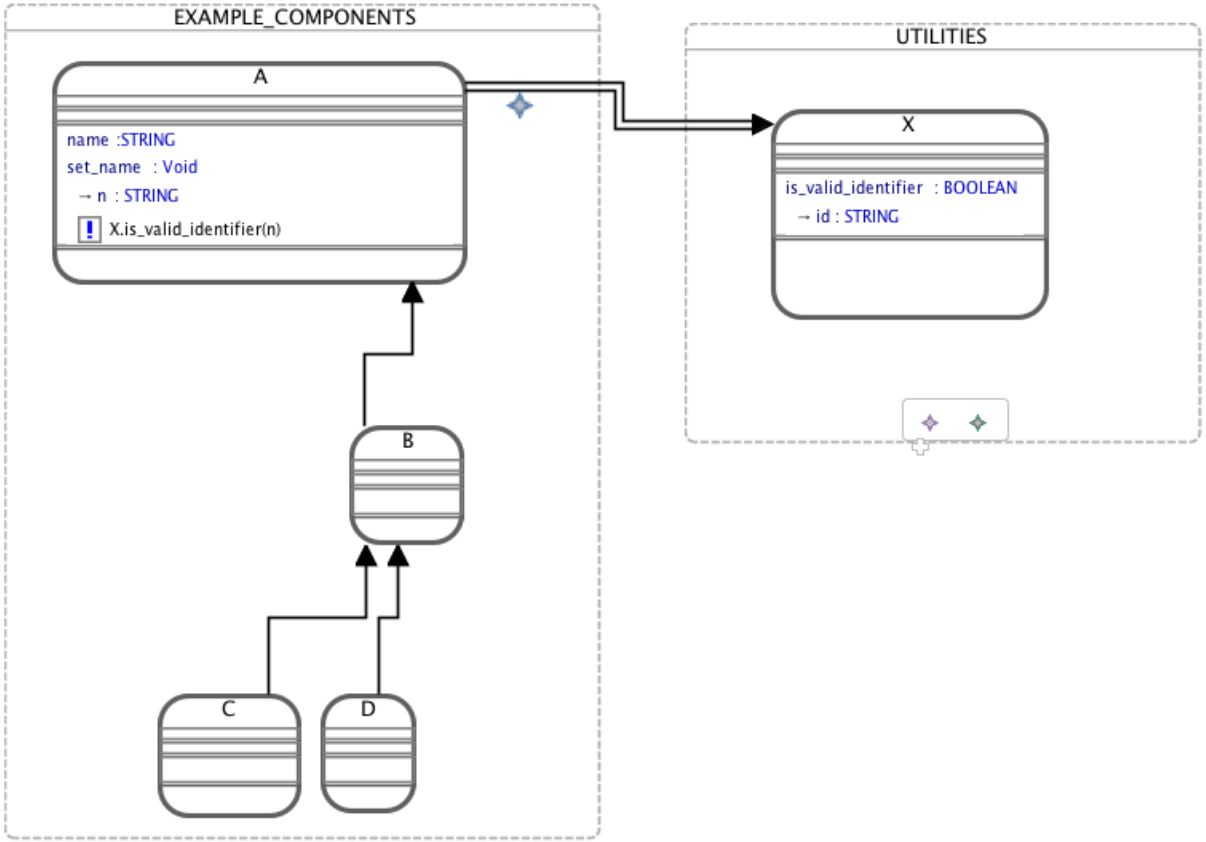
What is the relationship?

- How are the textual and graphical forms related?
- How are changes propagated between the forms?
- How are views on the model related?
- Does what I'm looking at in the graphical view correspond to what I expect in the textual view?

BON

- came from the Eiffel community,
- is used for designing object-oriented systems,
- and has both a textual and graphical form with a one-to-one mapping between them.
- Key concepts: class, cluster, feature, invariant, contract.
- In this work we focused on the static parts of BON.

Graphical BON example



Textual BON example

```
static_diagram EXAMPLE
component
  cluster EXAMPLE_COMPONENTS
  component
    class A
      feature
        name: STRING
        set_name: Void
         -> n:STRING
         require X.is_valid_identifier(n)
        end
    end
    class B inherit A end
    class C inherit B end
    class D inherit B end

    B client UTILITIES.X

  end

  cluster UTILITIES
  component
    class X
      feature
        is_valid_identifier: BOOLEAN
          -> id:STRING
    end
  end
end
```

The formalisation

- Mechanically formalised in the PVS Specification and Verification System
- PVS's interactive proof checker allows proving the correctness of theorems, as well as type-correctness.
- Available for download and inspection.
- Looks something like this:

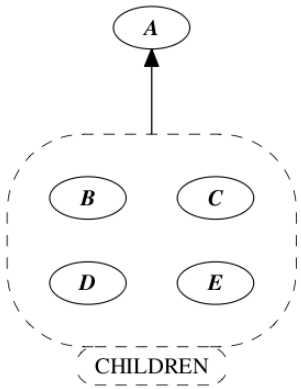
```
ANY: BONClass
inheritance_tree: TYPE =
  { t: Tree[BONClass] | root?(t)(ANY) }
```

Formalisation cont'd

- On the textual side we define type contexts, class and cluster definitions, feature definitions, types, ...
- On the graphical side we define inheritance trees, clustering trees, client graphs, ...
- Next, and most importantly, we define a relation between textual models and graphical models.
- We also define conversion functions from textual models to graphical models, and vice versa.
- These functions are bijective and inverses of each other.

Views

- A *view* is a subset of the elements in a model.
- Each constituent graph (inheritance, etc.) is a subgraph of the corresponding graph in the model.
- We provide a formal defintion of the type of BON model views.
- Examples: outline view, folding, BON compresssions



Model Evolution

- As the textual or graphical form evolves, how do we maintain consistency?
- If we define creating and applying diffs on each form, and a (bijective) translation between these diffs, we can compute a required update without translating the whole model.
- We define functions: graph merge, graph difference, model difference, model diff, and model diff application.
- A model difference has a similar *shape* to a model.

$$apply\_diff(m_1, diff(m_1, m_2)) = m_2$$

To Conclude

- We mechanically formalised the graphical and textual models for (a subset of) BON.
- We formally defined the relationship between these forms, as well as a bijective function for converting between them.
- We also gave a definition for views on a model, as well as functions for diffing and merging BON models
- Plenty of future work: completing the formalisation, full BON coverage, and complete and robust tool support.