

An Integration Concept for Complex Modelling Techniques

Benjamin Braatz
Technische Universität Berlin

Workshop on Multi-Paradigm Modeling: Concepts and Tools
Genova, Italy

3 October 2006

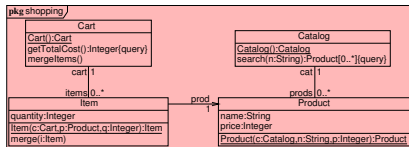
- Contemporary Modelling Techniques: Complex Integrated Techniques
- Need for Integrated Semantics to describe and analyse Interdependencies
- Rich Variety of Multi-Paradigm Behaviour Techniques mapped to Single Semantic Domain
- Semantic Domain should facilitate Verification and Code Generation



CLUML – The Complex Modelling Technique Example – Class Diagram

- Compact, Comprehensive and Constructive UML Profile
- Class Diagrams for Structural Constraints
- OCL Constraints for Method Effects and Query Behaviour
- Transformation Rules for Local Changes
- Flowcharts for Algorithms
- Activity Diagrams for Workflows

Class Diagram: Specification of Structural Constraints



Example – OCL Constraints

Pre-Post Constraint: Descriptive Specification of Method Effect

```

context shopping::Cart:mergeItems()
post: self.items->forAll(i, j:Item)
      i1->i2 implies i1.prod->i2.prod)
self.items@pre.prod->asSet()=
self.items.prod->asSet()
    
```

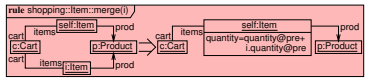
Body Constraint: Complete Specification of Query Behaviour

```

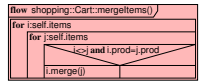
context shopping::Cart:getTotalCost()
body: self.items->iterate(i:Item, sum:Integer=0)
      sum+i.quantity*i.prod.price)
    
```

Example – Transformation Rule and Flowchart

Transformation Rule: Specification of Local Changes



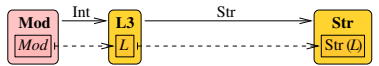
Structured Flowchart: Specification of Algorithms



L3 – The Low-Level Language

L3 – Structure Modelling

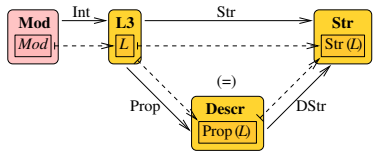
- Common Semantic Domain for CUJML (and possible Extensions)
- Basis for Verification and Code Generation
- Strict Separation of Structure, Descriptive and Constructive Parts



Example – Structure L3 – Property Modelling

Structure Model: Pure Structure without Associations, Cardinalities, etc.

Str	Product
prods: Set(Product) Catalog: Catalog search(n:String): Set(Product)	cat: Catalog name: String price: Integer Product(c: Catalog, n: String, p: Integer): Product
Cart items: Set(Item) Cart(): Cart getTotalCost(): Integer mergeItems()	Item cart: Cart prod: Product quantity: Integer Item(c: Cart, p: Product, q: Integer): Item merge(): Item



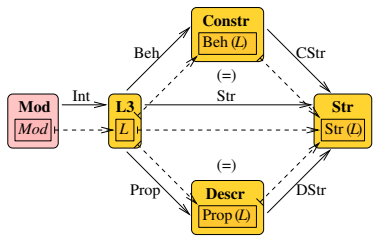
Example – Properties L3 – Behaviour Modelling

Property Model: Associations, Cardinalities, etc. from Class Diagram, OCL Pre-Post Constraints

```

Prop
assoc CarItems: for all i:Item,c:Cart: i.cart=c <<> i in c.items
assoc CatProds: for all p:Product,c:Catalog: p.cat=c <<> p in c.prods
mult Cart: for all i:Items: i.cart defined
mult Prod: for all i:Items: i.prod defined
mult Cat: for all p:Product: p.cat defined
Cart::getTotalCost()
query
Item::Item(c:Cart,p:Product,q:Integer):Item
pre: assoc CarItems,assoc CatProds,
mult Cart,mult Prod,mult Cat
post: assoc CarItems,assoc CatProds,
mult Cart,mult Prod,mult Cat
Item::merge():Item
pre: assoc CarItems,assoc CatProds,
mult Cart,mult Prod,mult Cat,
self.cart=i,self.prod=l,
q1>self.quantity,q2=i.quantity
post: assoc CarItems,assoc CatProds,
mult Cart,mult Prod,mult Cat,
self.quantity=q1+q2
not i in self.cart.items

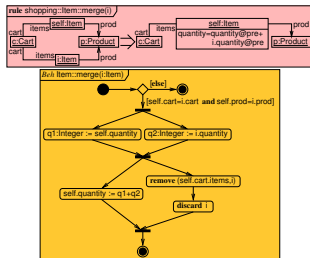
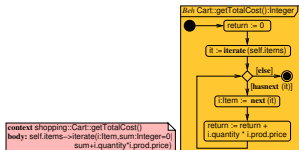
Catalog::search(n:String):Set(Product)
query
Product::Product(c:Catalog,n:String,p:Integer):Product
pre: assoc CarItems,assoc CatProds,
mult Cart,mult Prod,mult Cat
post: assoc CarItems,assoc CatProds,
mult Cart,mult Prod,mult Cat
Cart::mergeItems()
pre: assoc CarItems,assoc CatProds,
mult Cart,mult Prod,mult Cat,
oldits=-self.items.prod
post: assoc CarItems,assoc CatProds,
mult Cart,mult Prod,mult Cat,
for all i1,i2:Item: i1.i2 in self.items ==>
(i1.c>i2.c => i1.prod<=i2.prod)
for all p:Product:
p in oldits <<> p in self.items.prod
    
```



Example – Behaviour for OCL Body Constraint

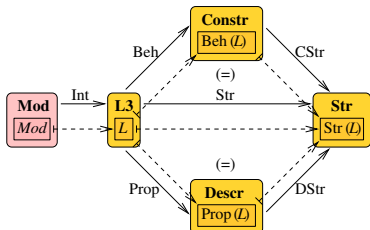
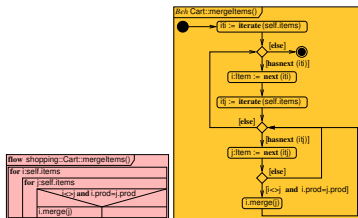
Example – Behaviour for Transformation Rule

Behaviour Model: Executable, Constructive Behaviour for Different Modelling Techniques



Example – Behaviour for Structured Flowchart

Summary – Abstract Integration Concept



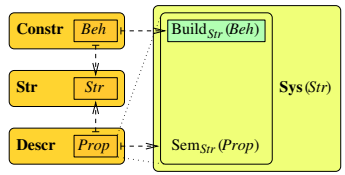
Comparison to Other Approaches Future Work – Semantics

Abstract State Machines and Process Algebras:

- Unintuitive Encoding of OO Concepts necessary
- Detour on the Way to Code Generation
- Advantage: Tools and Theory readily available

Petri Nets:

- No existing Variant supports all OO Features
- New Formalism necessary, as well



Future Work

Formal Syntax:

- UML Profile for CUJML
- Meta Model for L3
- Model Transformation from CUJML to L3

Formal Semantics:

- Behaviour Semantics by Simulating the Execution
- Property Semantics by Satisfaction Relation
- Verification Techniques?

Code Generation:

- Generation of Java Byte Code Classes from Behaviour
- Generation of Test Cases from Properties
- Other Target Platforms: .NET-CIL, ...

