An integrated Multi-View Model Evolution Framework

Volker Stolz



United Nations University

UNU-IIST

International Institute for Software Technology

Problem Description

- rCOS Modeler for use case-driven, component based, development
- Uses UML (+ rCOS profile)
- Development process through refinement/transformation
- Verification (model checking), codegen (Java), test-case generation (MBT)

Where are the models?

- •Possible views:
 - There is no model—at least not until you're finished.
 - There a (too) many models—many models along the way to the result.
 - There is only one model—but not the one you expected!

Modeling Levels: OMG



PIM: sub-divisions







PIM: sub-divisions



Transformations in rCOS

- Class model:
 - "Usual suspects", implemented:
 - Expert Pattern, simple Auto-refine
 - Manual refinement steps
- Abstraction (on the method-level)
- Integration (merge state machine into "guarded designs")
- Component/Use case: splitting into subcomponents/use cases (in progress)

Current setup (1)



<<ExecutionList>> refineAndCheck (from Transformations) This command with integrate the state-machine into the Cashdesk in croce ntuiti-view style. It will apply further refinements to remove CHAOS with RETURN for a robust design. A CSP snapshot is taken before and after, and a CSP model with both definitions for the Cashdesk (before and after refining CHAOS) is produced. The snapshot contains a few interesting assertions, like showing that a true FD-refinement took place.

<<Autorefine>> AutorefineCashDesk (from Transformations)

Current setup (2)



Transformations...

- ...are based on meta-model with classes and associations/attributes.
- ... can be primitive actions (add, delete, update).
- ...have formal parameters.
- ...can be chained: output of one transformation is input for another one.
- ...have prerequisites.
- ...are refinements (proof!).

Updating a model through a transformation

•Workflow:

- User selects transformation and arguments. Example: Expert pattern applied on sub-expression with atton.
 Che Model updated Kodel updated Find Lestructively! Find Lestructively!
- Compute and execute primitive actions.

Alternative



- Primitive actions are singleton transformations, corresponding to traditional editing steps!
- Final model described by (final) "chain" of transformations.

Editing chains of transformations

- Transformation as batch job.
- Achieving the desired output model means changing the transformations.
- Inserting/removing a step influences subsequent step because of dependencies.
- Especially so when there are proof obligations/proofs.

Example: Expert Pattern

package CoCoME_ClassModel

-	Cashdesk
+ e)	xmode : boolean
+ ei	nableExpress ()
+ di	sableExpress ()
+ st	artSale ()
+ fi	nishSale ()
+ ca	ardPay (c : Card)
+ ca	ashPay (a : double, out c : double)
+ e1	nterltem (code : Barcode, qty : int)
	$1 \\ 1 \\ \sqrt{+sale}$
	1 1 +sale Sale
	1 1 +sale Sale + complete : boolean
	1 1 + sale Sale + complete : boolean + total : double

Cashdesk::finishSale() {

Cashdesk	
+ exmode : boolean	
+ enableExpress ()	
+ disableExpress ()	
+ startSale ()	
+ finishSale ()	
+ cardPay (c : Card)	
+ cashPay (a : double, out c : double)	ŝ
+ enterItem (code : Barcode, qty : int)
1 1 + sale	
Sale	
+ complete : boolean	
+ total : double	
+ Sale (c : boolean, d : Date)	



Sale::finishSale() {

[\vdash complete' = true]; var double sum: sum := 0: for (LineItem I : lines) { $[\vdash sum' = sum + I.subtotal]$ }; $[\vdash \text{total'} = \text{sum}];$ end sum 16

[sale != null \vdash sale.complete' = true];

+ finishSale ()

```
var double sum:
sum := 0;
for (LineItem I : sale.lines) {
    [\vdash sum' = sum + I.subtotal]
};
[ \vdash sale.total' = sum];
end sum
```

Things to consider

- Change to the (initial) spec of Cashdesk:finishSale()
 → re-apply transformation.
- Change in the hierarchy of Sale requiring to re-check prerequisites.
- There was no Sale::finishSale() before—now other transformations/editing actions may depend on it.

Some observations

- Refinement-based modeling means developing a program that will transform your model (what about strategies?).
- Model becomes data structure, transformation becomes the model (Composition of transformations).
- Inherits many (interesting) problems and solutions from programming/program analysis; dependency tracking may have huge overhead?
- "Semantic overhead" from refinement setting.
- Presentation issues (GUI just a view; trans-formation should be in-memory instead of on-disk).
- Multi-view: you may not be seeing whole model/entire impact.

A quick word on proof obligations

•Source of proof obligations:

- Prerequisites of transformations (semantic!).
- NOT for refinements themselves...
- ...except for manual refinement steps (which syntactically substitute one design for another).
- May be updated when re-running transformations.
- •Proofs:
 - Stored together with transformation.
 - Re-checked when transformation is re-run.
 - Needs to be updated/re-done when it fails.



Trying out rCOS

0.0	Softwar	e Updates and Add-ons		
	Installed S	oftware Available Software		
ype filter text			0 -	Install)
-				1
Name			version	
COS Modeler - metadata				(Properties)
T III rCOS Modeler				
R rCOS Meta Model Feature			1.2.3	
COS Modeler Feature			1.2.3	Add Site
rCOS Modeler Library Feat	ure		1.2.3	
] 🕨 🖏 The Eclipse Project Updates				(Manage Sites)
				(Refresh)
) 4 1-	
Show only the latest versions of available sof	tware			
D				Close
*CoCoME_ComponentModel.rcos	22 ib			
CoCoME/cocome usecase refineme	nt/CoCoME	ComponentModel.rcosdi		
Coloct		management and a second second		
Margues Compone	ent Model G	oCoME ComponentModel		
maiquee	-	2		
Objects *		< <com< td=""><td>ponent>></td><td>8</td></com<>	ponent>>	8
Service Component				
Process Component		< <servicel< td=""><td>omponent>></td><td></td></servicel<>	omponent>>	
Interface		Proo	rssaue	55
Port			components b	互
- Connections +		~~~	.omponent>>	민
Interface Bealization		< <serv< td=""><td>iceComponent>></td><td></td></serv<>	iceComponent>>	
CashDes.				
Composition	Casi	hDeskif		
Composition	T -	CashPaymentiF	CardPaymentiF	t i
Delegation		D	Test.	1200000000000
Comment *		CashPaymentik	Carg	PaymentIF
🖹 Comment		Ŷ		Y
Comment Link		А		4
		5		<u> </u>
		< <component>> 屯</component>		mponent>> 뇬
		< <servicecomponent>;</servicecomponent>	> < <service< td=""><td>eComponent>></td></service<>	eComponent>>
		CashPayment	Car	dPayment
			_	
	_			
Problems 🕼 Javadoc 🔞 Declara	tion 🖾 Con	sole 🧐 Error Log 🗖 Prope	arties E3	
< <contractinterface>> <interface> Cas</interface></contractinterface>	hDesklF		and the second sec	
Model	rty	Value		
* Contr	act Interface			
Operations (ontract	In << contract>> < Package	ge>_fSaLoKfwEdyy	8P4OYI-5Dg_Contract
Stereotypes				

- Eclipse Ganymede (3.4.2)
- **rCOS Eclipse update site:**
- http://rcos.iist.unu.edu/eclips e/
 - Download the example models





TOPCASED

21