RM2PT: Automated Prototype Generation from Formal Requirements Model

Yilong Yang[1], Xiaoshan Li[1], Zhiming Liu[2], Wei Ke[3], Quan Zu[4], and Xiaohong Chen[5]

[1]University of Macau, Macau
[2]Southwest University, China
[3]Macau Polytechnic Institute, Macau
[4]Tongji University, Shanghai, China
[5]University of Illinois at Urbana-Champaign, USA

Content

- Motivation
- Overview
- Prototype Generation
- Evaluation
- Conclusion

Motivation

- Prototyping is an effective and efficient way for requirement validation

- However, manually developing a prototype would increase the overall cost of software development

- It is very desirable to have a CASE tool that automatically generates prototypes directly from requirements

Related work

- Current UML modeling tools can only generate skeleton code, where classes only contain attributes and signatures of operations, not their implementations.

- Even when design models are provided, only less than 48% source code can be generated from design models successfully

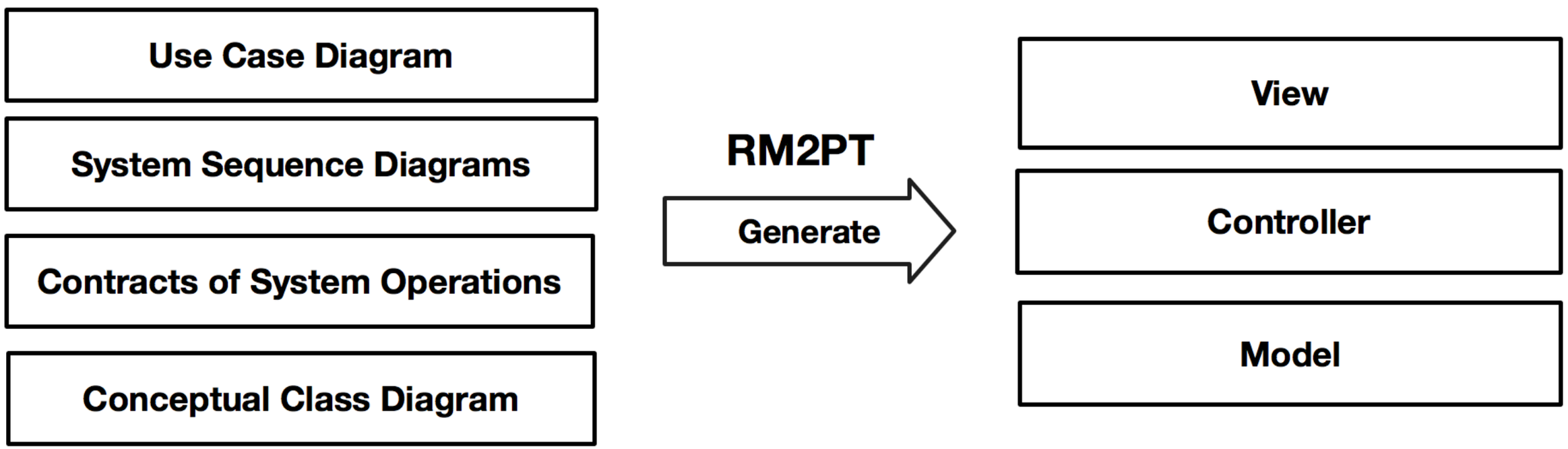
Innovation

- RM2PT can automatically generate prototypes only rely on requirements models without corresponding design models.

- RM2PT can identify the non-executable parts of contract and wrap them into an interface, which can be fulfilled by developers manually or third-party APIs.

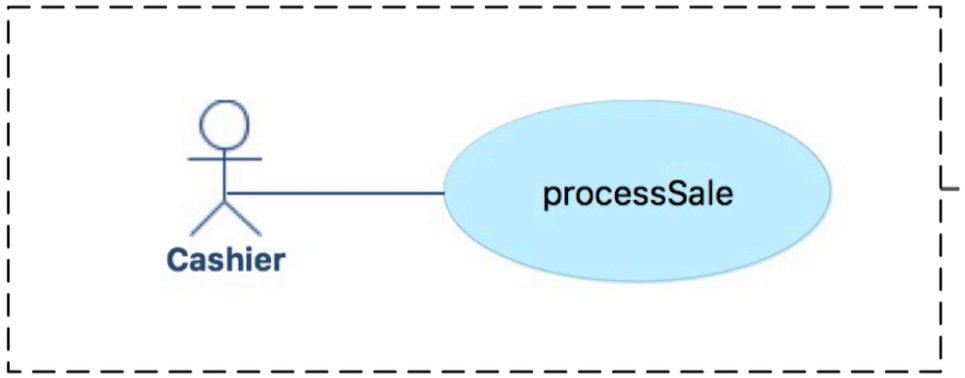
- The prototypes generated by RM2PT provide the functions of a) investigating the execution processes of use cases, b) the state observation, and c) pre-condition, post-condition, and invariant checking.

Overview

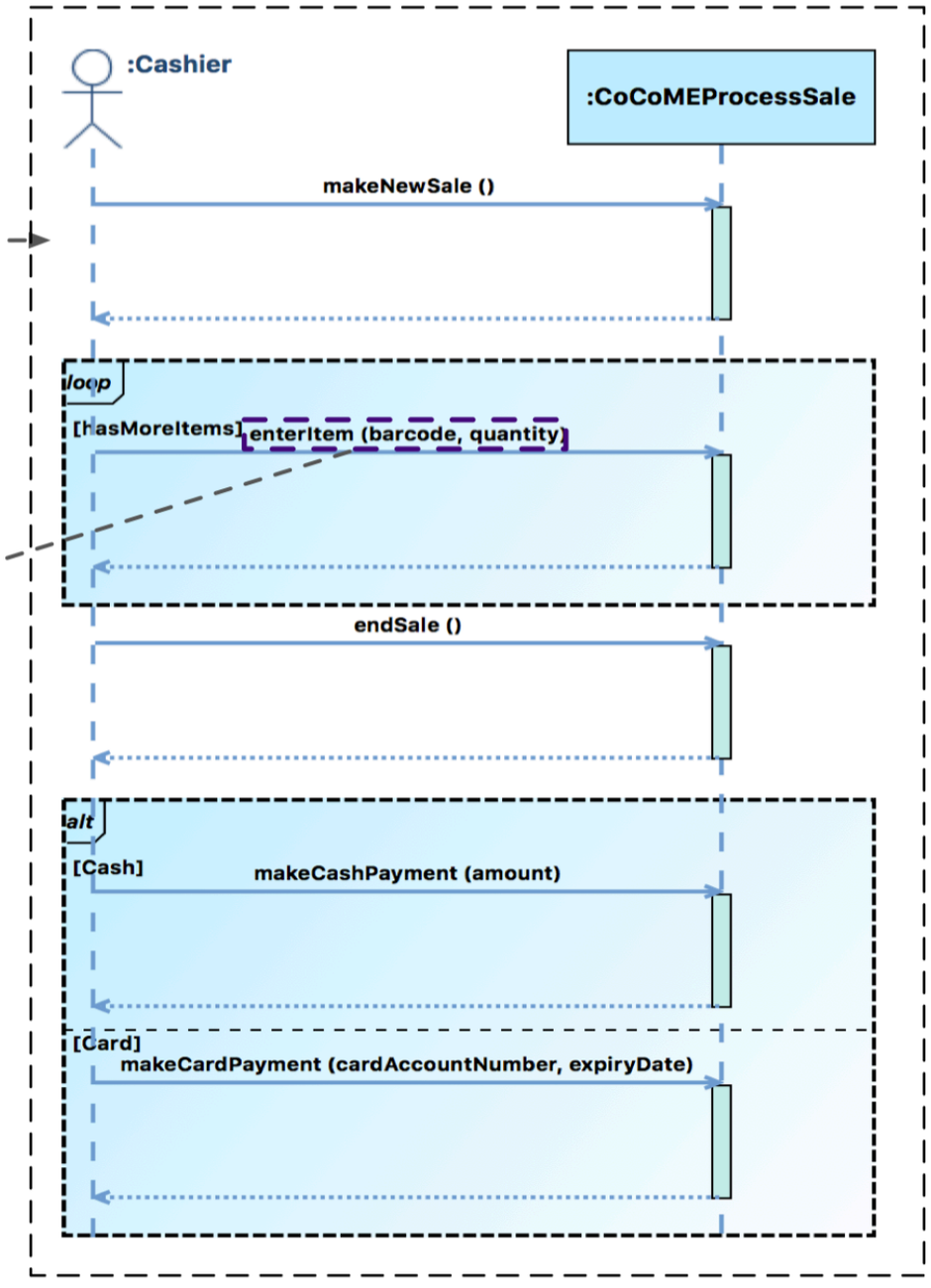


Available as free software: http://rm2pt.mydreamy.net

Requirements Model



1. Use Case Diagram

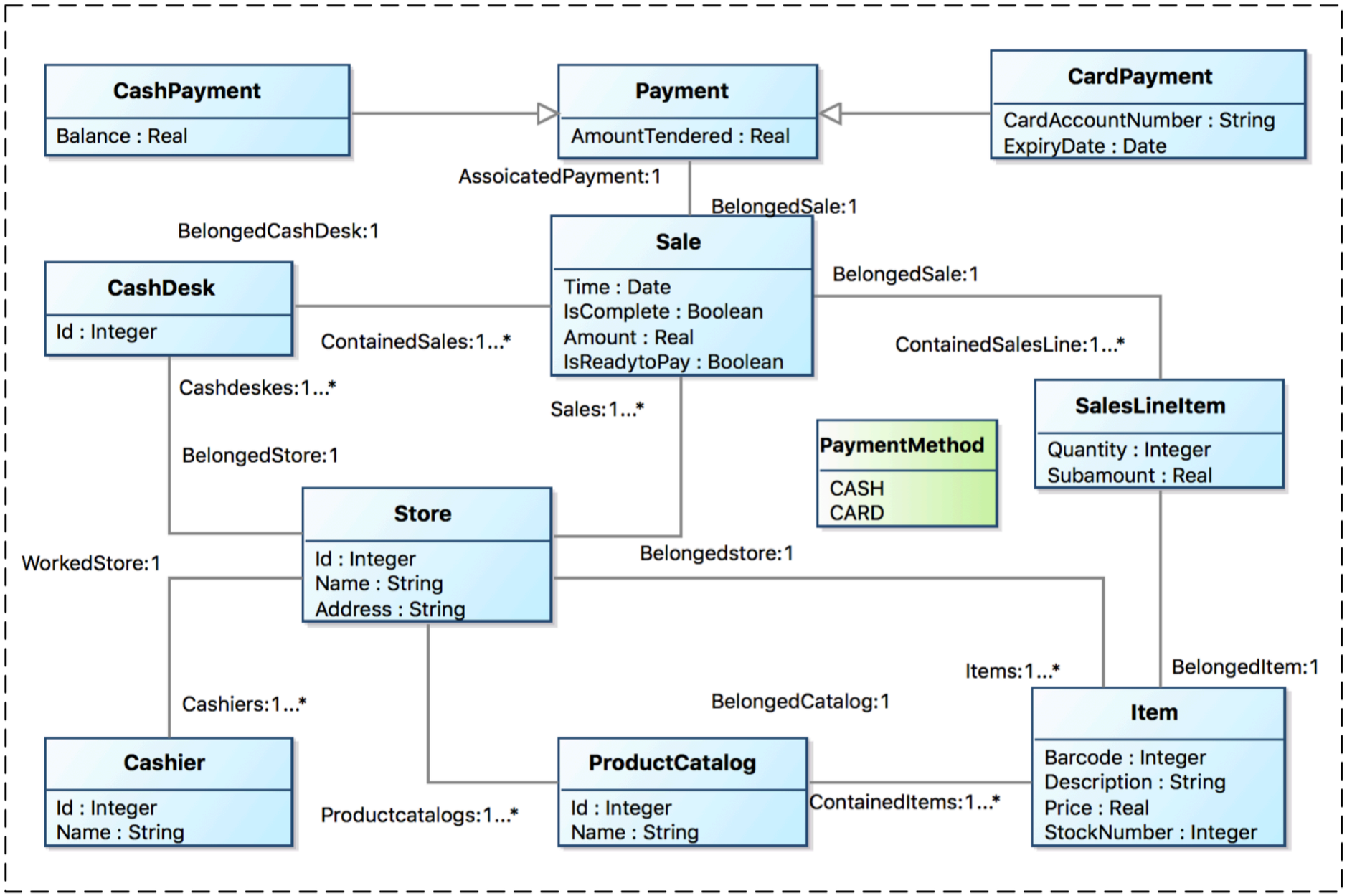


2. System Sequence Diagrams

```
Contract CoCoMEProcessSale::enterItem(barcode : Integer, quantity : Integer) : Boolean {

    /* Definition: find specific Item instance by barcode */
    definition:
        item:Item = Item.allInstance()->any(i:Item | i.Barcode = barcode)

    /* Precondition: there is a sale underway */
    precondition:
        currentSale.oclIsUndefined() = false and
        currentSale.IsComplete = false

     * A salesLineItem instance sli was created (instance creation).
    postcondition:
        let sli:SalesLineItem in
        sli.oclIsNew() and
        self.currentSaleLine = sli and
        sli.BelongedSale = currentSale and
        currentSale.ContainedSalesLine->includes(sli) and
        sli.Quantity = quantity and
        sli.BelongedItem = item and
        item.StockNumber = item.StockNumber@pre - quantity and
        sli.Subamount = item.Price * quantity and
        SalesLineItem.allInstance()->includes(sli) and
        result = true

}
```

3. Contracts of System Operations

Requirements Model

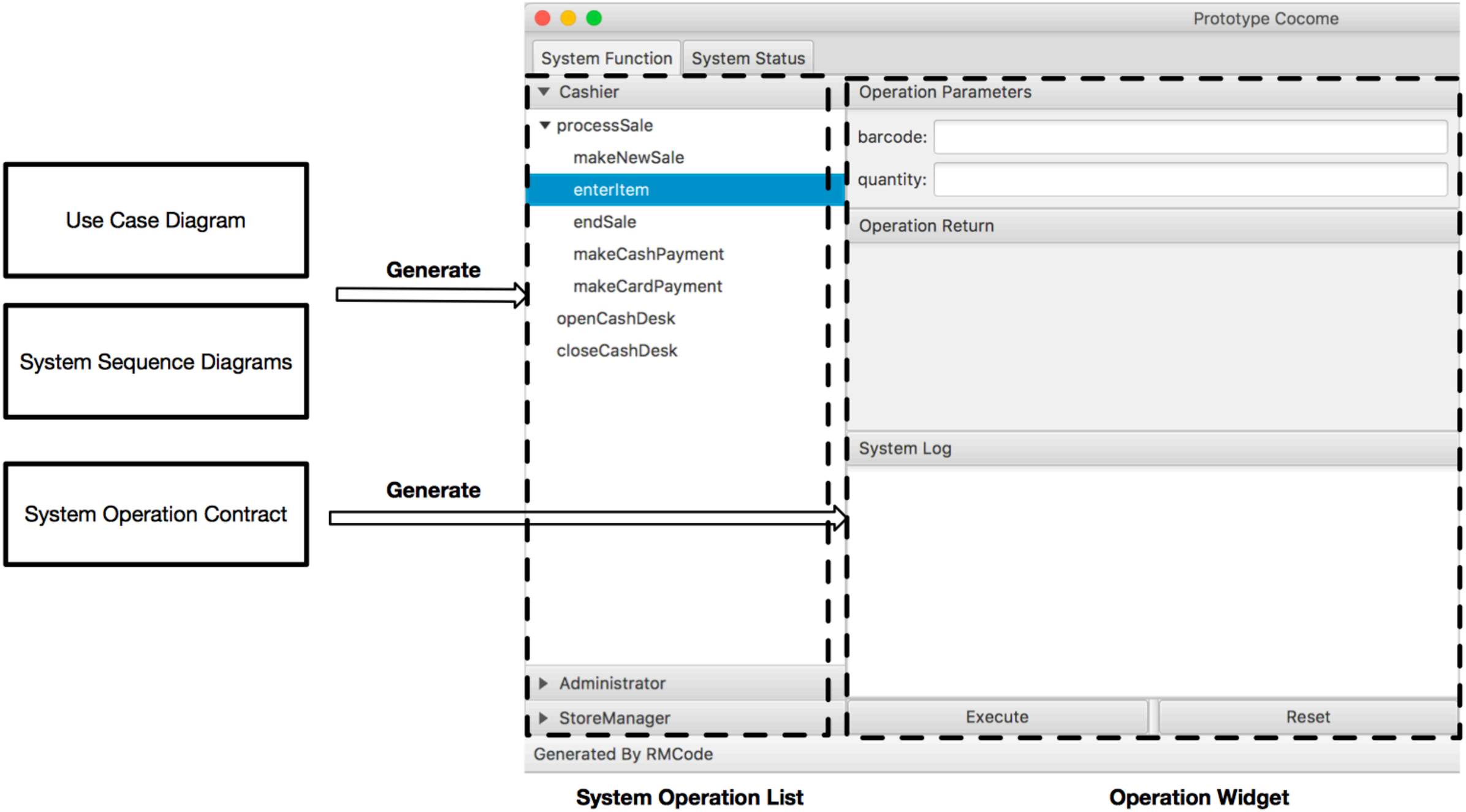


4. Conceptual Class Diagram

Content

- Motivation
- Overview
- **Prototype Generation**
- Evaluation
- Conclusion

Prototype GUI (Execution)



Prototype GUI (Observation)

Conceptual Class Diagram

Generate

Objects Statistics

Prototype Cocome

System Function | System Status

Class statistics

Class Name	# of Objects
Store	1
ProductCatalog	1
CashDesk	1
Sale	1
Cashier	1
SalesLineItem	2
Item	3
Payment	0

Association statistics

Source Class	Association Name	Target Class	Multiple	Association Number
Sale	Belongedstore	Store	false	1
Sale	BelongedCashDesk	CashDesk	false	1
Sale	ContainedSalesLine	SalesLineItem	true	2
Sale	AssoicatedPayment	Payment	false	1

All Objects Sale:

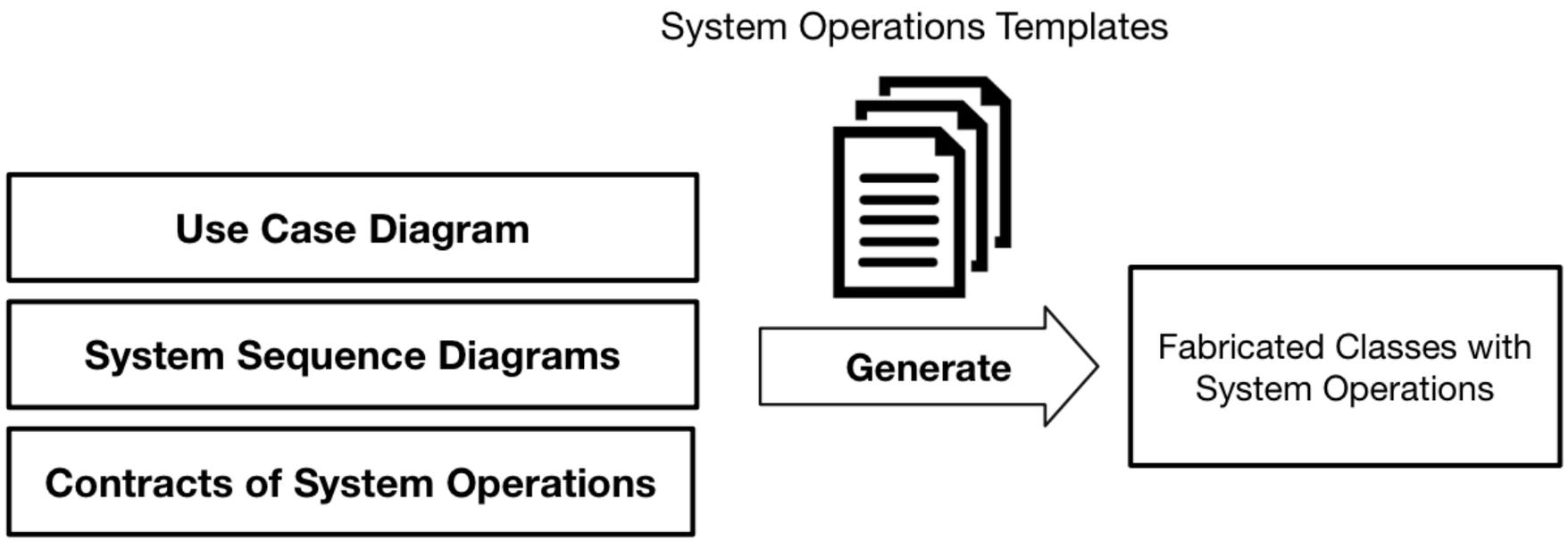
Time	IsComplete	Amount	IsReadytoPay
2018-08-13	true	160.0	true

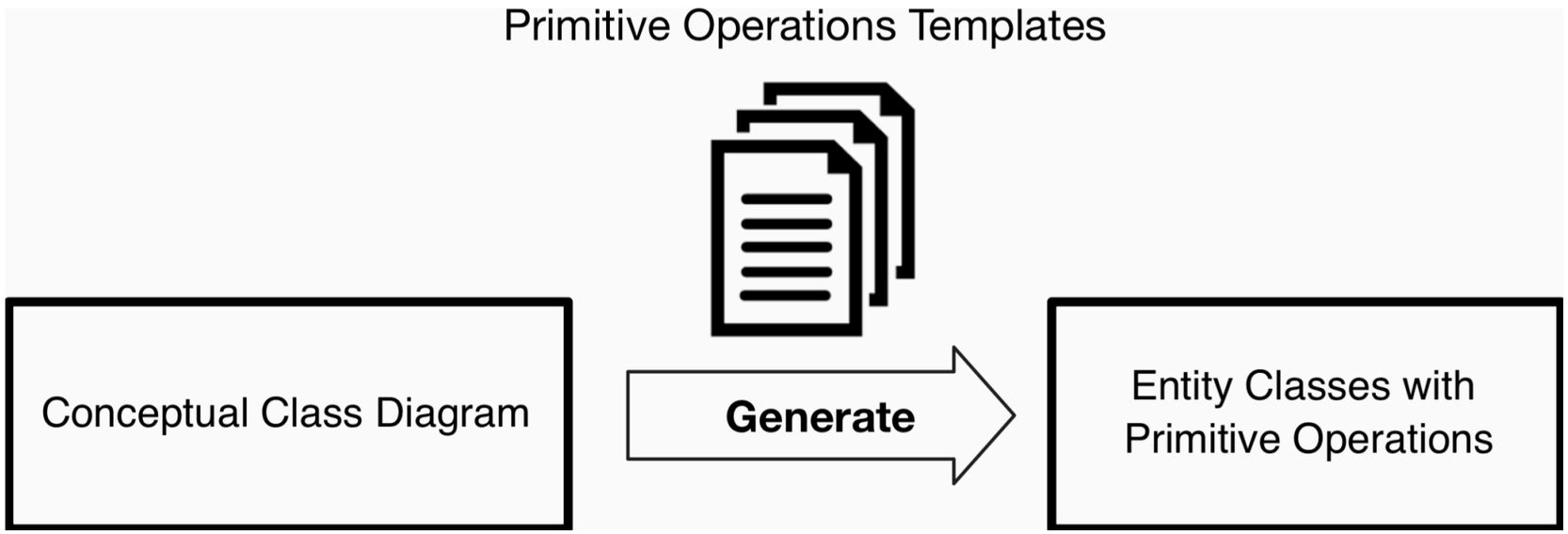
Load Status | Save Status | Refresh Status | Check All Invariants

The Associations of Objects

The Attributes of Objects

Fabricated and Entity Classes Generation





System operation decomposition

- Transform the contracts of system operations into primitive operations

- Encapsulate system operations into classes.

The contract of enterItem()

```
//Signature
Contract CoCoMEProcessSale::enterItem
  (barcode : String, quantity : Real) : Boolean {

//Definition Section
definition:
    //Find Object
    item:Item = Item.allInstance()->any(i:Item |
        i.Barcode = barcode)

//Pre-condition Section
precondition:
    currentSale.oclIsUndefined() = false and
    currentSale.IsComplete = false and
    item.oclIsUndefined() = false and
    item.StockNumber > 0

//Post-condition Section
postcondition:
    //Create an Object
    let sli:SalesLineItem in
    sli.oclIsNew() and
    //Add Links
    self.currentSaleLine = sli and
    sli.BelongedSale = currentSale and
    currentSale.ContainedSalesLine->includes(sli) and
    sli.BelongedItem = item and
    //Modify Attributes
    sli.Quantity = quantity and
    sli.Subamount = item.Price * quantity and
    item.StockNumber = item.StockNumber@pre -
        quantity and
    //Add an Object
    SalesLineItem.allInstance()->includes(sli) and
    result = true
}
```

Primitive Operations

	Primitive Operation	Return Type
Object	***findObject***(*ClassName*:*String*, *condition*:*String*)	Object
	findObjects(*ClassName*:*String*, *condition*:*String*)	Set(Object)
	createObject(*ClassName*:*String*)	Object
	addObject(*ClassName*:*String*, *ob*:*Class*)	Boolean
	releaseObject(*ClassName*:*String*, *ob*:*Class*)	Boolean
Attribute	***getAttribute***(*ob*:*Class*, *attriName*:*String*)	PrimeType
	setAttribute(*ob*:*Class*, *attriName*:*String*, *mathExp*:*String*)	Boolean
Link	***findLinkedObject***(*o*:*Class*, *assoName*:*String*, *condition*:*String*)	Object
	findLinkedObjects(*o*:*Class*, *assoName*:*String*, *condition*:*String*)	Set(Object)
	addLinkOnetoMany(*ob*:*Class*, *assoName*:*String*, *addOb*:*Class*)	Boolean
	addLinkOnetoOne(*ob*:*Class*, *assoName*:*String*, *addOb*:*Class*)	Boolean
	removeLinkOnetoMany(*ob*:*Class*, *assoName*:*String*, *removeOb*:*Class*)	Boolean
	removeLinkOnetoOne(*ob*:*Class*, *assoName*:*String*)	Boolean

Transformation Rules

$$Rule : \frac{OCL\ Expression}{Primitive\ Operation\ in\ Java\ Code}$$

RM2PT contains 26 transformation rules

Definition Section Transformation

$$\mathbf{R_1}: \frac{obs{:}\textbf{\textit{Set}}(ClassName){=}ClassName.\textbf{\textit{allInstances}}()}{List{<}ClassName{>}\ obs\ =\ EM.\textbf{\textit{findObjects}}(ClassName{:}String)}$$

$$\mathbf{R_2}: \frac{obs{:}\textbf{\textit{Set}}(ClassName){=}ClassName.\textbf{\textit{allInstances}}() \rightarrow \textbf{\textit{select}}(o \mid \textbf{\textit{conditions}}(o))}{List{<}ClassName{>}\ obs\ =\ EM.\textbf{\textit{findObjects}}(ClassName{:}String,\ conditions(o){:}String)}$$

$$\mathbf{R_3}: \frac{ob{:}ClassName\ =\ ClassName.\textbf{\textit{allInstances}}() \rightarrow \textbf{\textit{any}}(o \mid \textbf{\textit{conditions}}(o))}{ClassName\ ob\ =\ EM.\textbf{\textit{findObject}}(ClassName{:}String,\ conditions(o){:}String)}$$

$$\mathbf{R_4}: \frac{o{:}ClassName\ =\ ob.assoName}{ClassName\ o\ =\ EM.\textbf{\textit{findLinkedObject}}(ob{:}Class,\ assoName{:}String)}$$

$$\mathbf{R_5}: \frac{obs{:}Set(ClassName)\ =\ ob.assoName}{List{<}ClassName{>}\ obs\ =\ EM.\textbf{\textit{findLinkedObjects}}(ob{:}Class,\ assoName{:}String)}$$

$$\mathbf{R_6}: \frac{obs{:}Set(ClassName)\ =\ ob.assoName \rightarrow \textbf{\textit{select}}(o \mid \textbf{\textit{conditions}}(o))}{List{<}ClassName{>}\ obs\ =\ EM.\textbf{\textit{findLinkedObjects}}(ob{:}Class,\ assoName{:}String,\ preconditions(o){:}String)}$$

$$\mathbf{R_7}: \frac{o{:}ClassName\ =\ ob.assoName \rightarrow \textbf{\textit{any}}(o \mid \textbf{\textit{conditions}}(o))}{ClassName\ ob\ =\ EM.\textbf{\textit{findLinkedObject}}(ob{:}Class,\ assoName{:}String,\ conditions(o){:}String)}$$

Pre-condition Transformation

$$\mathbf{R_8}: \frac{ob.oclIsUndefined() = bool}{\textbf{\textit{StandardOPs.oclIsUndefined}}(ob{:}Class,\ bool{:}Boolean)}$$

$$\mathbf{R_9}: \frac{var.oclIsTypeOf(type)}{\textbf{\textit{StandardOPs.oclIsTypeOf}}(\text{«}var\text{»},\ type{:}String)}$$

$$\mathbf{R_{10}}: \frac{obs.isEmpty() = bool}{\textbf{\textit{StandardOPs.isEmpty}}(obs{:}Set(Class),\ bool{:}Boolean)}$$

$$\mathbf{R_{11}}: \frac{obs.size()\ op\ \ mathExp}{\textbf{\textit{StandardOPs.size}}(obs{:}Set(Class))\ \text{«}op\text{»}\ \text{«}mathExp\text{»}}$$

$$\mathbf{R_{12}}: \frac{ob.AttriName\ op\ varPM}{\textbf{\textit{getAttribute}}(ob{:}Class,\ attriName{:}String)\ \text{«}op\text{»}\ \text{«}varPM\text{»}}$$

$$\mathbf{R_{13}}: \frac{ClassName.\textbf{\textit{allInstances}}()\rightarrow\textbf{\textit{includes}}(ob)}{\textbf{\textit{StandardOPs.includes}}(EM.findObjects(ClassName),\ ob{:}Class)}$$

$$\mathbf{R_{14}}: \frac{ClassName.\textbf{\textit{allInstances}}()\rightarrow\textbf{\textit{excludes}}(ob)}{\textbf{\textit{StandardOPs.excludes}}(EM.findObjects(ClassName),\ ob{:}Class)}$$

$$\mathbf{R_{15}}: \frac{ClassName.\textbf{\textit{allInstance}}()\text{->}\textbf{\textit{isUnique}}(o{:}ClassName\ |\ o.AttriName)}{\textbf{\textit{StandardOPs.isUnique}}(ClassName{:}String,\ AttriName{:}String)}$$

Post-condition Transformation

$$\mathbf{R_{16}}: \frac{\textbf{\textit{let}}\ ob{:}ClassName\ \textbf{\textit{in}}\ ob.\textbf{\textit{oclIsNew}}()}{ClassName\ ob = EM.\textbf{\textit{createObject}}(ClassName{:}String)}$$

$$\mathbf{R_{17}}: \frac{ClassName.\textbf{\textit{allInstances}}() \rightarrow \textbf{\textit{includes}}(ob)}{EM.\textbf{\textit{addObject}}(ClassName{:}String,\ ob{:}Class)}$$

$$\mathbf{R_{18}}: \frac{ClassName.\textbf{\textit{allInstances}}() \rightarrow \textbf{\textit{excludes}}(ob)}{EM.\textbf{\textit{releaseObject}}(ClassName{:}String,\ ob{:}Class)}$$

$$\mathbf{R_{19}}: \frac{ob.assoName \rightarrow \textbf{\textit{includes}}(addOb)}{\textbf{\textit{addLinkOnetoMany}}(ob{:}Class,\ assoName{:}String,\ addOb{:}Class)}$$

$$\mathbf{R_{20}}: \frac{ob.assoName \rightarrow \textbf{\textit{excludes}}(removeOb)}{\textbf{\textit{removeLinkOnetoMany}}(ob{:}Class,\ assoName{:}String,\ removeOb{:}Class)}$$

$$\mathbf{R_{21}}: \frac{ob.assoName = addOb}{\textbf{\textit{addLinkOnetoOne}}(ob{:}Class,\ assoName{:}String,\ addOb{:}Class)}$$

$$\mathbf{R_{22}}: \frac{ob.assoName = \textbf{\textit{null}}}{\textbf{\textit{removeLinkOnetoOne}}(ob{:}Class,\ assoName{:}String)}$$

$$\mathbf{R_{23}}: \frac{ob.attriName = mathExp}{\textbf{\textit{setAttribute}}(ob{:}Class,\ attriName{:}String,\ «mathExp»{:}PrimeType)}$$

$$\mathbf{R_{24}}: \frac{obs\text{->}\textbf{\textit{forAll}}(o{:}ClassName \mid o.AttriName = mathExp)}{\textbf{\textit{for}}\ (ClassName\ o{:}obs)\ \{\textbf{\textit{setAttribute}}(o{:}Class,\ AttriName{:}String,\ «mathExp»{:}PrimeType);\}}$$

$$\mathbf{R_{25}}: \frac{\textbf{\textit{return}} = var}{\textbf{\textit{return}}\ «var»;}$$

$$\mathbf{R_{26}}: \frac{\textbf{\textit{ThirdPartyServices}}.opName(vars)}{\textbf{\textit{service}}.opName(«vars»)}$$

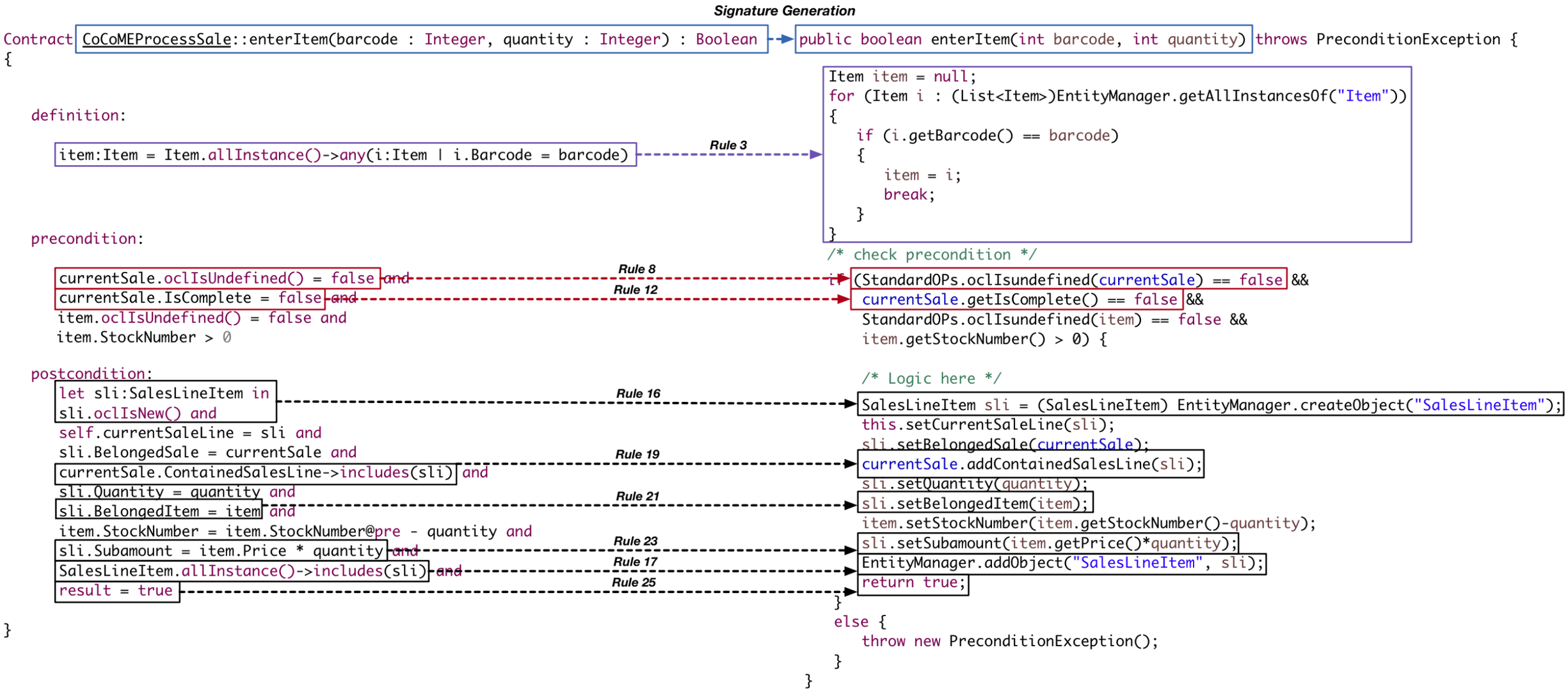
Input : OCLExpression, Tag
Output: Primitive Operations
begin
    rs ← ∅;
    i ← 0;
    sub-formulas ← *parse(OCLExpression)*;
    connectors ← *parseConnector(OCLExpression)*;
    lastn ← *len(sub-formulas)* - 1;
    for *s* ∈ *sub-formulas* **do**
        num ← 0;
        switch *Tag* **do**
            case *definition* **do**
                num ← *matchRule1to7(s)*;
            end
            case *pre-condition* **do**
                num ← *matchRule8to15(s)*;
            end
            case *post-condition* **do**
                num ← *matchRule16to26(s)*;
            end
        end
        if *num != 0* **then**
            r ← *transform(s, num, Tag)*;
            if *tag == "pre-condition" and i != lastn* **then**
                rs.append(r, connectors[i]);
            else
                rs.append(r, "linebreaks");
            end
        else
            rs.append("transformation error for
             sub-formula:", s);
        end
        i++;
    end
    return *rs*;
end

Algorithm 1: Transformation Algorithm

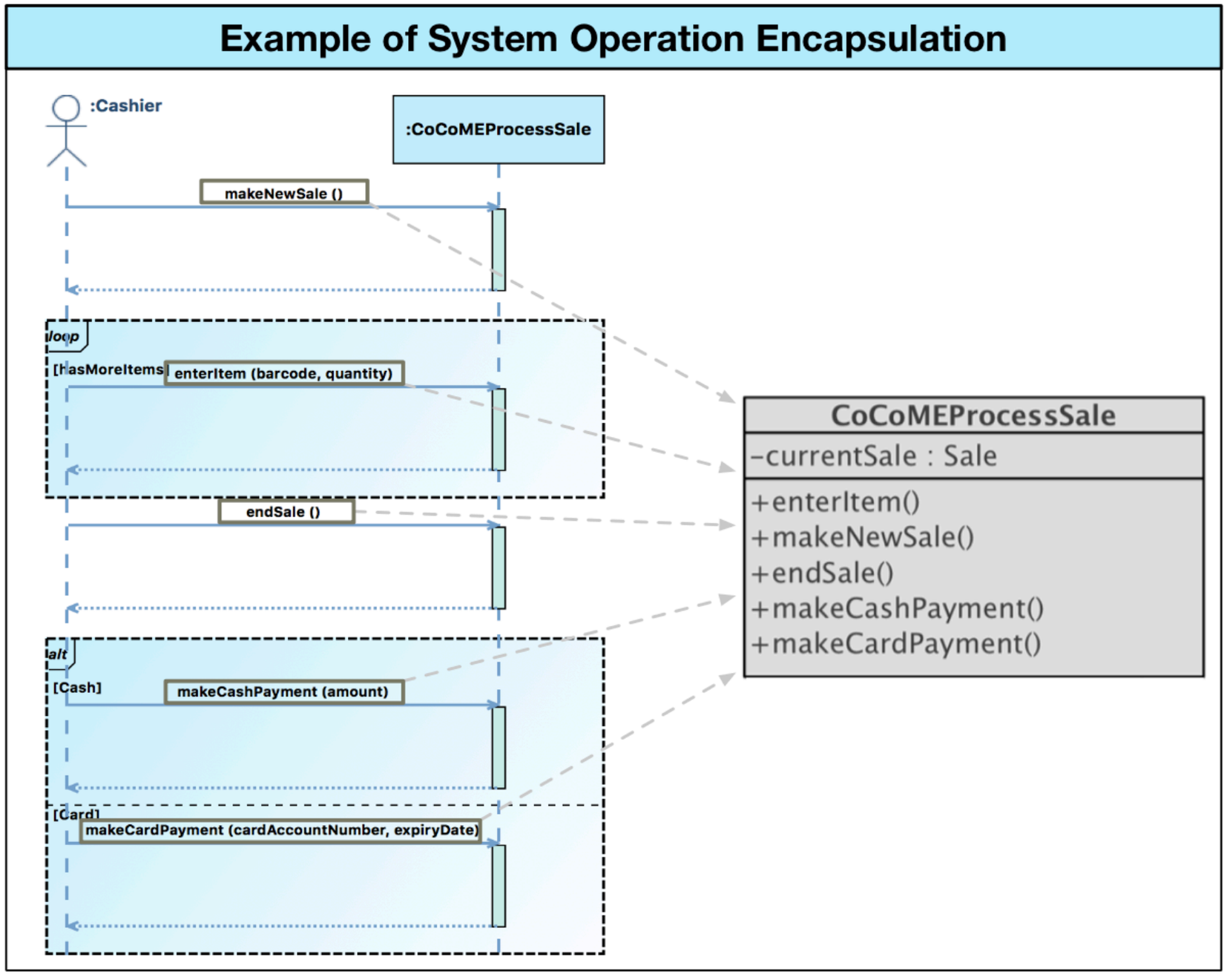
Rule matching

Failed matching

An transformation example for the contract of enterItem()



System operation encapsulation of use case processSale



Content

- Motivation
- Overview
- Prototype Generation
- **Evaluation**
- Conclusion

Four case studies

- ATM - Automated Teller Machine
- CoCoME - Supermarket System
- LibMS - Library Management System
- LoanPS - Loan Processing System

Complexity of Requirements Model

Case Study	Actor	Use Case	SO	AO	Entity Class	Association	INV
ATM	2	6	15	103	3	4	5
CoCoME	3	16	43	273	13	20	10
LibMS	7	19	45	433	11	17	25
LoanPS	5	10	34	171	12	8	12
Sum	17	51	137	980	39	49	52

* Above table shows the number of elements in the requirements model. SO and AO are the abbreviations of system and primitive operations responsibility respectively. INV is the abbreviation of invariant.

Four case studies contain
17 actors, 51 use cases, 137 system operations, 980 primitive operation, 39 entity classes, 49 associations of entity classes, and 52 invariants

Generation Result

Case Study	NumSO	MSuccess	GenSuccess	SuccessRate (%)
ATM	15	15	15	100
CoCoME	43	41	40	93.02
LibMS	45	43	42	93.33
LoanPS	34	30	30	88.23
Average	34.25	32.25	31.75	93.65

* MSuccess is the number of SO which is modeled correctly without external event-call, GenSuccess is the number of SO which is successfully generated, SuccessRate = GenSuccess / NumSO.

Generation Performance

Name	Line of Code	Prototype Time (ms)	SO Time(ms)	Student (hour)	Developer (hour)
ATM	3897	309.74	2.26	8.3	3.2
CoCoME	9572	788.99	9.78	19.2	10.6
LibMS	12017	1443.39	18.22	25.1	14.3
LoanPS	7814	832.78	5.52	16.2	8.4
Average	8325	843.73	8.95	17.20	9.14

In average, the prototypes contain 8325 lines of code, generation spend less than 1 second, students need 17.20 hours and experienced developers require 9.14 hours.

RM2PT is much more efficient than manual prototyping ~1 second vs ~9 hours).

Conclusion

- An approach for automated prototype generation from requirements model

- A case tool: RM2PT

Future Work

- Requirement validation and evolution
- Consistent checking between requirements model and the generated prototype
- Integrating RM2PT with semantics service composition (SSC-ASP, Service Classification, and ML)
- Cloud-native application (MicroServices) generation from the validated requirements model

Thank you !

UM software engineering lab