Automated Prototype Generation from Requirements Model

Yilong Yang

Supervisor: Prof. Xiaoshan Li

Department of Computer and Information Science
Faculty of Science and Technology
University of Macau

May 23, 2019

Contents

Contents

Motivation

- Rapid prototyping is an effective and efficient way for requirements validation.
- However, manually developing a prototype would increase the overall cost of software development.
- It is very desirable to have an approach and a CASE tool that can automatically generate prototypes directly from requirements.

Related Work

- Current UML modeling tools can only generate skeleton code, where classes only contain attributes and operation signatures, not their implementations.
- To generate prototypes, a design model is required, which contains how to encapsulate system operations into classes and how to collaborate objects to fulfill system operations.
- They lack the mechanism to deal with the non-executable elements in the requirements model.
- The generated prototype does not provide the automatic mechanisms in run-time to consistency checking and state observations for requirements validation.

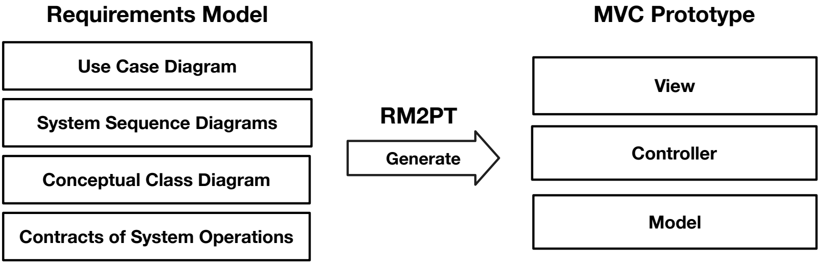
Contribution

We introduce an approach and a CASE tool for generating prototypes automatically, which

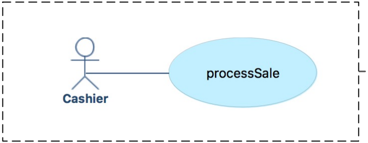
- do not require design models but only rely on a requirements model
- provide a mechanism to identify the non-executable parts of a contract and wrap them into an interface, which can be fulfilled by developers manually or third-party APIs
- contain validity and consistency checking as well as state observation in the generated prototypes

Contents

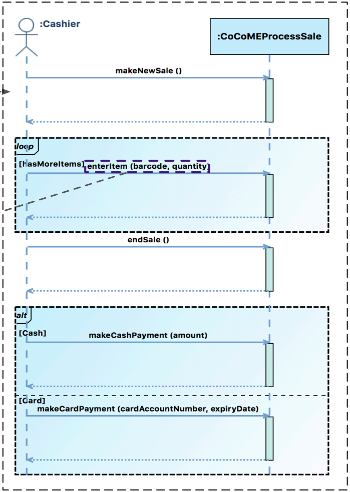
Overview


Requirements Model
Use Case Diagram
System Sequence Diagrams
Conceptual Class Diagram
Contracts of System Operations
RM2PT
Generate
MVC Prototype
View
Controller
Model

Requirements Model



1. Use Case Diagram

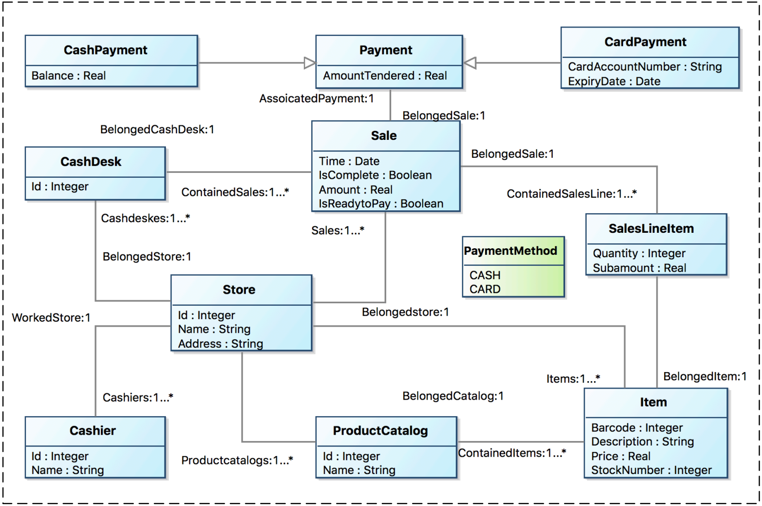


2. System Sequence Diagrams

```
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

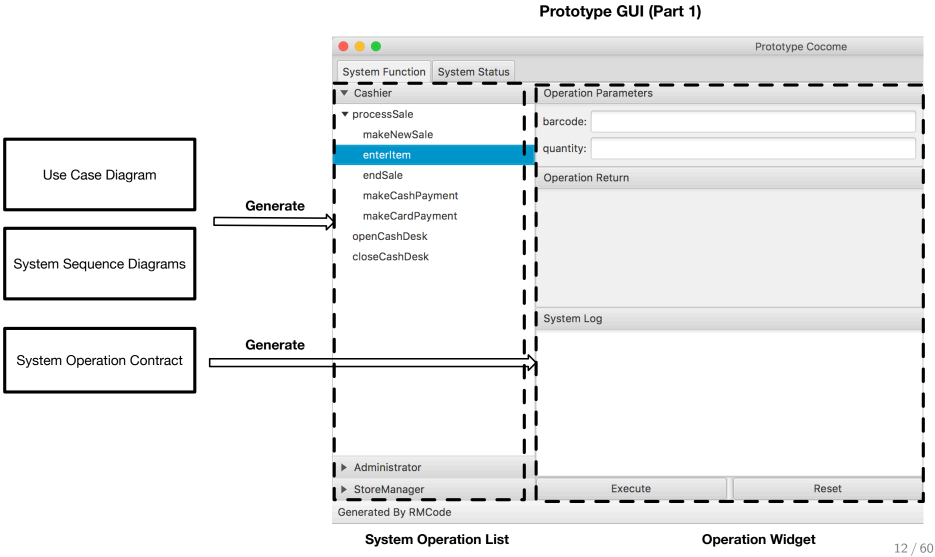
Requirements Model



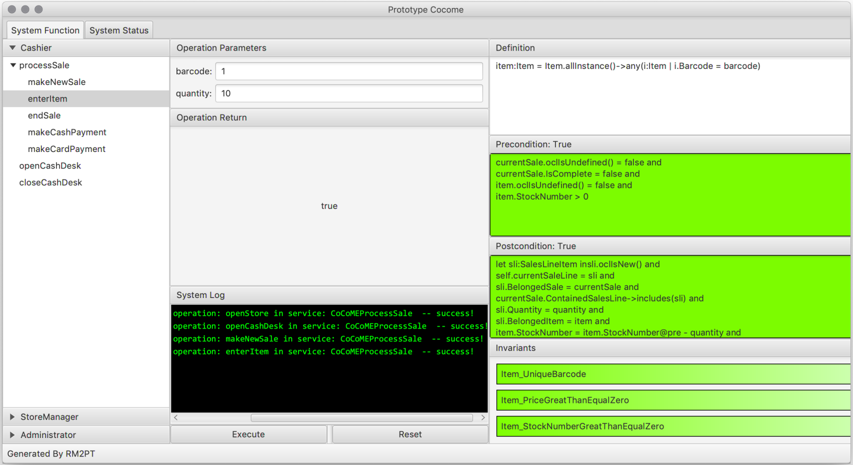
4. Conceptual Class Diagram

Contents

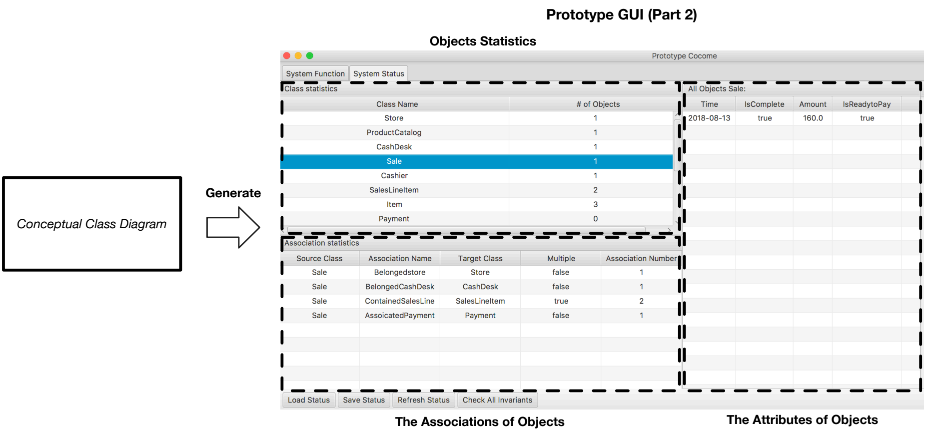
Prototype GUI (Execution)



Prototype GUI (Execution)



Prototype GUI (Observation)

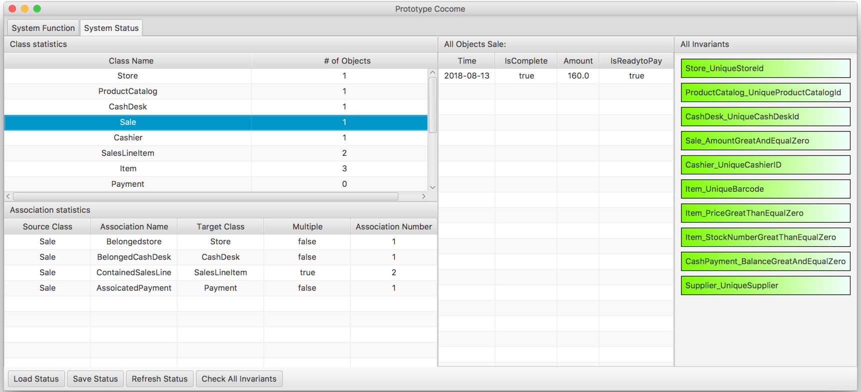


Class Name	# of Objects
Store	1
ProductCatalog	1
CashDesk	1
Sale	1
Cashier	1
SalesLineItem	2
Item	3
Payment	0

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

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

Prototype GUI (Observation)



Terminology

- *System operation.* System operation is an operation that the system executes in response to a system input event in system sequence diagrams.
- *Primitive operation.* Primitive operations are the operations introduced to covers all primitive actions of object-oriented system to manipulate a) objects, b) the attributes of objects, and c) the links of objects.

Primitive Operations

Table 1: 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

System Operation Decomposition

Main taskes:

- Transform the contracts of system operations into primitive operations.
- Encapsulate system operation into classes.

Contract of System Operation

```
//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
```

Contract of System Operation

```
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
}
```

Transformation rules

Transformation rules:

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

Definition Section Transformation

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

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

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

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

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

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

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

Pre-condition Transformation

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

$$\mathbf{R_9} : \frac{\textit{var.oclIsTypeOf(type)}}{\textbf{StandardOPs.oclIsTypeOf}(\ll \textit{var} \gg,\textit{type}:\textit{String})}$$

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

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

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

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

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

$$\mathbf{R_{15}} : \frac{\textit{ClassName}.\textbf{allInstance}\textit{()} \rightarrow \textbf{isUnique}\textit{(o:ClassName} \mid \textit{o.AttriName)}}{\textbf{StandardOPs.isUnique}(\textit{ClassName}:\textit{String},\textit{AttriName}:\textit{String})}$$

Post-condition Transformation

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

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

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

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

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

$\mathbf{R_{25}} : \dfrac{\textbf{return} = \textit{var}}{\textbf{return} \ll\textit{var}\gg;}$

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

$\mathbf{R_{26}} : \dfrac{\textbf{ThirdPartyServices}.\textit{opName(vars)}}{\textbf{service}.\textit{opName}(\ll\textit{vars}\gg)}$

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

$\mathbf{R_{23}} : \dfrac{\textit{ob.attriName} = \textit{mathExp}}{\textbf{setAttribute}(\textit{ob}{:}\textit{Class},\textit{attriName}{:}\textit{String},\ll\textit{mathExp}\gg{:}\textit{PrimeType})}$

$\mathbf{R_{24}} : \dfrac{\textit{obs}{\rightarrow}\textbf{forAll}(\textit{o}{:}\textit{ClassName} \mid \textit{o.AttriName} = \textit{mathExp})}{\textbf{for } (\textit{ClassName o}{:}\textit{obs}) \{\textbf{setAttribute}(\textit{o}{:}\textit{Class}, \textit{AttriName}{:}\textit{String}, \ll\textit{mathExp}\gg{:}\textit{PrimeType});\}}$

Transformation Algorithm

```
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
```

Example: enterItem()

Signature Generation

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

   definition:

     item:Item = Item.allInstance()->any(i:Item | i.Barcode = barcode)

   precondition:

     currentSale.oclIsUndefined() = false and
     currentSale.IsComplete = false and
     item.oclIsUndefined() = false and
     item.StockNumber > 0

   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

}
```

Rule 3 · Rule 8 · Rule 12 · Rule 16 · Rule 19 · Rule 21 · Rule 23 · Rule 17 · Rule 25

```
public boolean enterItem(int barcode, int quantity) throws PreconditionException {
  Item item = null;
  for (Item i : (List<Item>)EntityManager.getAllInstancesOf("Item"))
  {
     if (i.getBarcode() == barcode)
     {
        item = i;
        break;
     }
  }
  /* check precondition */
  if (StandardOPs.oclIsundefined(currentSale) == false &&
      currentSale.getIsComplete() == false &&
      StandardOPs.oclIsundefined(item) == false &&
      item.getStockNumber() > 0) {

      /* Logic here */
      SalesLineItem sli = (SalesLineItem) EntityManager.createObject("SalesLineItem");
      this.setCurrentSaleLine(sli);
      sli.setBelongedSale(currentSale);
      currentSale.addContainedSalesLine(sli);
      sli.setQuantity(quantity);
      sli.setBelongedItem(item);
      item.setStockNumber(item.getStockNumber()-quantity);
      sli.setSubamount(item.getPrice()*quantity);
      EntityManager.addObject("SalesLineItem", sli);
      return true;
  }
  else {
      throw new PreconditionException();
  }
}
```

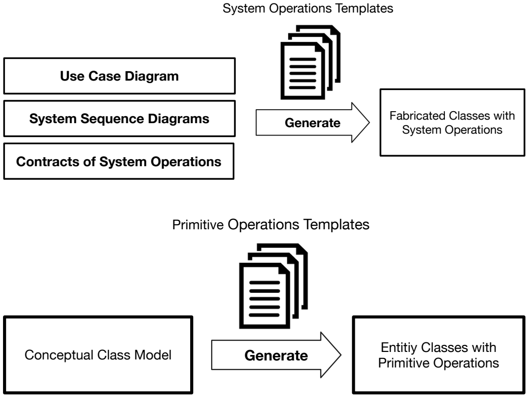
Fabricated and Entity Classes

To indicate the classes are from domain concepts or fabrications in the prototype, we divide classes into two types:

- *Entity class.* Entity classes are Java classes generated in prototypes from conceptual class diagrams, the others are fabricated classes.

- *Fabricated class.* Fabricated classes are the classes not generated from conceptual class diagrams.

Fabricated and Entity Classes Generation



Fabricated Class Generation Algorithm

Input : *ucd* - Use Case Diagram,
ssds - System Sequence Diagrams,
contracts - Contracts,
t_{so} - System Operation Template
Output: Fabrication Classes

```
/* Initialize ucClasses as empty set */
for uc ∈ ucd do
    /* generate fabrication uc class */
    ucClass ← generateClassSkeleton(uc);
    /* find uc related system sequence diagram */
    ssd ← findSSD(uc, ssds);
    for op ∈ ssd do
        /* find system operation contract */
        opCtr ← findContract(op, contracts);
        /* generate system operation sysOp() */
        generate systemOperation() by t_so with opCtr;
        encapsulate systemOperation() to class ucClass;
    end
end
```

System Operation Template

```
/* operation signature */
public «c.opSign.returnType» «c.opSign.name»(

«FOR para : c.opSign.parameters SEPARATOR ','»
   «para.type» «para.name»
«ENDFOR») throws PreconditionException {

/* contract definition */
«IF c.definition != null»
   «c.definition.mapping»
«ENDIF»

/* check precondition */
if («c.precondition.mapping») {

   /* contract post-condition*/
   «c.postcondition.mapping»

/* result return */
«IF c.opSign.returnType != null»
   return «returnName»;
«ENDIF»
else {
   throw new PreconditionException();
}}
```

Entity Class Generation Algorithm

Input : *ccd* - Conceptual Class Diagram
t_{ec} - Entity Class Template
t_{po} - Primitive Operation Templates
Output: Entity Classes

for *entity* $\in$ *ccd* **do**
  generate entity class skeleton by t_{ec};
  for *attribute* $\in$ *entity* **do**
    genereate *getAttribute()* by t_{po};
    genereate *setAttribute()* by t_{po};
  end
  for *association* $\in$ *entity* **do**
    if *Is-Multiple(association)* $==$ *true* **then**
      genereate *findLinkedObjects() by* t_{po};
      genereate *addLinkOnetoMany() by* t_{po};
      genereate *removeLinkOnetoMany() by* t_{po};
    else
      genereate *findLinkedObject() by* t_{po};
      genereate *addLinkOnetoOne() by* t_{po};
      genereate *removeLinkOnetoOne() by* t_{po};
    end
  end
end

Entity Class Template

```
/* Class Skeleton */
public class ≪c.name≫

/* Class Inheritance */
≪IF c.superClass != null≫
   extends ≪c.superClass.Name≫
≪ENDIF≫ {

/* Attributes */
≪FOR attribute : c.attributes≫
   private ≪attribute.type≫ ≪attribute.name≫;
≪ENDFOR≫

/* Associations */
≪FOR assoc : c.associations≫
   private
   ≪IF assoc.isIsmultiple≫
      List<≪assoc.class≫> ≪assoc.name≫ =
         new LinkedList<≪assoc.class≫>();
   ≪ELSE≫
      ≪assoc.class.name≫ ≪assoc.name≫;
   ≪ENDIF≫
≪ENDFOR≫

/* primitive operations templates */
}
```

Primitive Operation Templates

```
//Getting Attribute
public «attribute.type» get«attribute.name»() {
    return «attribute.name»;
}

//Setting Attribute
public void set«attribute.name»(«attribute.type» «attribute.name») {
    this.«attribute.name» = «attribute.name»;
}
```

Primitive Operation Templates

```
//findLinkedObjects()
public List<≪assoc.class≫> get≪assoc.name≫() {
    return ≪assoc.name≫;
}

//addLinkOnetoMany()
public void add≪assoc.name≫(≪assoc.class≫ ob) {
    this.≪assoc.name≫.add(ob);
}

//removeLinkOnetoMany()
public void remove≪assoc.name≫(≪assoc.class≫ ob) {
    this.≪assoc.name≫.remove(ob);
}


//findLinkedObject
public ≪assoc.class≫ get≪assoc.name≫() {
    return ≪assoc.name≫;
}

//addLinkOnetoOne() removeLinkOnetoOne()
public void set≪assoc.name≫(≪assoc.class≫ ob) {
    this.≪assoc.name≫ = ob;
}
```

EntityManager Generation Algorithm

Input : *ccd* - Conceptual Class Diagram
t_{em} - EntityManager Template
t_o - Primitive Operation Templates for Object
Output: EntityManager Class
begin
`/* Generate EntityManager Skeleton */`
generate *EntityManager* skeleton by *ccd*, t_{em};
`/* Generation Primitive Operations */`
generate *findObject() by* t_o;
generate *findObjects()* by t_o;
generate *createObject()* by t_o;
generate *addObject()* by t_o;
generate *releaseObject()* by t_o;
end

EntityManager Template

```
/* EntityManager Template */
public class EntityManager {

    /* HashMap Object Records*/
    private static Map<String, List> AllInstance = new HashMap<String, List>();

    /* create object reference list */
    «FOR c : classes»
    private static List<«c.name»> «c.name»Instances =
        new LinkedList<«c.name»>();
    «ENDFOR»

    /* Put object reference list into Map */
    static {
    «FOR c : classes»
        AllInstance.put("«c.name»", «c.name»Instances);
    «ENDFOR»
    }

    /* Get all objects of the class */
    public static List getAllInstancesOf
        (String ClassName) {
        return AllInstance.get(ClassName);
    }
}
```

Primitive Operation Templates for Finding Objects

```
/* find object template */
≪cName≫ target = null; //initialize target object
for (≪cName≫ o:
    EntityManager.getAllInstancesOf(≪cName≫)) {
    //finding the object satisfies the condition
    if (≪precondtion(o)≫) {
        target = o;
        return target;
    }
}
```

```
/* find objects template */
List<≪c.name≫> targets = = new LinkedList<>(); //initialize target object lists
for (≪c.name≫ o:
    EntityManager.getAllInstancesOf(≪c.name≫)) {
    //finding the object satisfies the condition
    if (≪precondtion(o)≫) {
        targets.add(o);
    }
}
return targets;
```

Templates for Creating, Adding and Releasing Object

```
/* create object template */
public static Object createObject(String cName) {
    Class c = Class.forName("EntityManager");
    Method m = c.getDeclaredMethod("create" + cName + "Object");
    return m.invoke(c);
}
≪FOR c : classes≫
public static ≪c.name≫ create≪c.name≫Object() {
    ≪c.name≫ o = new ≪c.name≫();
    return o;
}
≪ENDFOR≫


/* add object template */
public static Object addObject(String cName, Object ob) {
    Class c = Class.forName("EntityManager");
    Method m = c.getDeclaredMethod("add" + cName + "Object", Class.forName(cName));
    return (boolean) m.invoke(c, ob);
}
≪FOR c : classes≫
public static boolean add≪c.name≫Object(≪c.name≫ o) {
    return ≪c.name≫Instances.add(o);
}
≪ENDFOR≫
```

Templates for Creating, Adding and Releasing Object

```
/* release object template */
public static boolean deleteObject(String cName, Object ob) {

    Class c = Class.forName("EntityManager");
    Method m = c.getDeclaredMethod("delete" + cName + "Object", Class.forName(cName));
    return (boolean) m.invoke(c, ob);
}

≪FOR c : classes≫
public static boolean delete≪c.name≫Object
    (≪c.name≫ o) {
    return ≪c.name≫Instances.remove(o);
}
≪ENDFOR≫
```

Contents

Case Studies

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

Complexity of Requirements Models

Table 2: The Complexity of Requirements Models

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 respectively. INV is the abbreviation of invariant.

Cost of Requirements Modeling

Table 3: Cost of Requirements Modeling

Case Study	UML Diagram	OCL Contracts	Total (hours)
ATM	1.01	1.32	2.33
CoCoME	4.55	4.91	9.46
LibMS	4.64	6.37	11.01
LoanPS	5.51	6.94	12.45
Average	3.92	4.88	8.81

* UML diagram contains a use case diagram, system sequence diagrams, and a conceptual class diagram.

Generation Result of System Operations

Table 4: The Generation Result of System Operations

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.

Results of Requirement Validation

Table 5: Requirements Errors

	Requirements Errors		
Name	Pre-condition	Post-condition	Invariant
ATM	5	12	1
CoCoME	8	23	3
Library	12	26	2
Loan	6	21	2
Total	31	68	8

Results of Requirement Validation

Table 6: Requirements Missing

	Requirements Missing					
Name	Actor	UseCase	SO	Entity Class	Association	INV
ATM	1	3	9	1	2	3
CoCoME	1	11	22	5	10	5
LibMS	4	12	14	11	15	12
LoanPS	2	3	15	4	2	8
Total	8	29	60	21	29	28

Automated Prototyping vs Manual Prototyping

Table 7: Manual Prototyping

Case Study	Implementation	Testing	Debugging	Total (hr)
ATM	6.09	4.63	3.90	14.62
CoCoME	15.08	8.80	8.31	32.19
LibMS	18.28	9.18	7.29	34.74
LoanPS	13.23	8.96	8.79	30.98
Average	13.17	7.89	7.07	28.13

Automated Prototyping vs Manual Prototyping

Table 8: Automated Prototyping

Name	Line of Code	Automated Prototype (ms)	System Operation (ms)
ATM	3897	309.74	2.26
CoCoME	9572	788.99	9.78
LibMS	12017	1443.39	18.22
LoanPS	7814	832.78	5.52
Average	8325	843.73	8.95

Discussion

- Auto-prototyping is much more efficient than manual prototyping (**~1 second vs ~28 hours**) without introducing inconsistency between the requirements model and prototype.
- The spending time of system operations**(~9 ms)** is much less than the prototyping (**~850 ms**).

Scope and Limitation

Our approach has the scopes of application for practical problems.

- The requirements model and the generated prototypes of our approach are object-oriented.
- Our approach suitable for modeling and validating object-oriented information systems, enterprise systems, and interactive systems. The batching systems have heavy internal workloads are not suited for.
- Moreover, our approach focuses on functional requirements but not non-functional requirements such as time, dependability, security, and space. That means the real-time systems, embedding systems, and cyber-physical systems are not suitable for our approach.

Scope and Limitation

Our approach has the following limitations.

- The first one is that 6.91% system operations cannot be successfully generated without introducing third-party services, but this limitation has been solved by invoking the third-part services.
- The second limitation that is although the formal specification OCL has short learning cure than other formal specification, it still needs time for learning to specify the correct contract.
- The third limitation is the performance of generation, and it can be further optimized in the future.

Contents

Conclusion

This thesis presents an approach and a CASE tool to automated prototype generation from a requirements model.

- The executable parts of the contract are translated into Java source code. The non-executable parts of a contract can be identified and wrapped by an interface, which can be fulfilled by third-party APIs.
- Four cases studies have been investigated, and the experiment result is satisfactory that the **93%** of system operations of use cases can be generated successfully in 1 second.

Future Work

- Improve the current transformation algorithm to cover the more substantial subset of the executable specification.
- Integrate current prototyping tool with our another work on automated translating use case definitions in natural language into their corresponding formal contract in OCL.
- Furthermore, after a system requirements model is validated by prototyping, we plan to generate the prototype into its corresponding real system.

CASE tool - RM2PT

- RM2PT is available as free software: `http://rm2pt.yilong.io`
- Auto-Prototyping Demos `https://youtu.be/rDdpXsjSq8A`
- Requirements Validation Demos `https://youtu.be/Y7GNa57WGfA`

Publication

- **Yilong Yang**, Xiaoshan Li, Zhiming Liu, Wei Ke. "RM2PT: A Tool for Automated Prototype Generation from Requirements Model". presented at the 41th International Conferences on Software Engineering (ICSE'19), Montreal, Canada, May 2019. (**CCF A**).
- **Yilong Yang**, Xiaoshan Li, Wei Ke, Zhiming Liu. "Automated Prototype Generation from Formal Requirements Model". IEEE Transactions on Reliability (**JCR - Q1**).

Publication

- **Yilong Yang**, Wei Ke, Jing Yang, Xiaoshan Li. "Integrating UML With Service Refinement for Requirements Modeling and Analysis". IEEE Access, 7, pp. 11599-11612 (2019). (**JCR** - **Q1**).
- **Yilong Yang**, Quan Zu, Xiaoshan Li. "Real-Time System Modeling and Verification Through Labeled Transition System Analyzer". IEEE Access, 7, pp. 26314-26323 (2019). (**JCR** - **Q1**).
- **Yilong Yang**, Wei Ke, Weiru Wang, Yongxin Zhao "Deep Learning for Web Services Classification". presented at the 11th International Conferences on Web Services (ICWS'19), Milan, Italy, July 2019. (**CCF B**).
- **Yilong Yang**, Jing Yang, Xiaoshan Li, Weiru Wang. "An Integrated Framework for Semantic Service Composition using Answer Set Programming". International Journal of Web Services Research. 11(4), pp. 47-61 (2014) (**SCI Indexed**).

Publication

- **Yilong Yang**, Xiaoshan Li, Nafees Qamar, Peng Liu, Wei Ke, Bingqing Shen, Zhiming Liu. "MedShare: A Novel Hybrid Cloud for Medical Resource Sharing among Autonomous Healthcare Providers". IEEE Access, 6, pp. 46949-46961 (2018). (**JCR** - **Q1**).
- **Yilong Yang**, Zu Quan, Peng Liu, Defang Ouyang, Xiaoshan Li. MicroShare: Privacy-Preserved Medical Resource Sharing through MicroService Architecture. International Journal of Biological Sciences, 14(8), pp. 907-919 (2018). (**JCR** - **Q1**)
- **Yilong Yang**, Zhuyifan Ye, Yan Su, Qianqian Zhao, Xiaoshan Li, Defang Ouyang. Deep Learning for in-vitro Prediction of Pharmaceutical Formulations. Acta Pharmaceutica Sinica B, 9(1), pp. 177-185 (2019) (**JCR** - **Q1**)
- Zhuyifan Ye, **Yilong Yang**, Xiaoshan Li, Defang Ouyang. An Integrated Transfer Learning and Multitask Learning Approach for Pharmacokinetic Parameter Prediction. Molecular pharmaceutics, 16(2), pp.533-541 (2018). (**Co-First** Author and **JCR** - **Q1**)

THANK YOU

Q & A