Requirement Modeling
--Specifying User Requirements in UML Models--

Ji Wu

2019-10-22

Agenda

- User Requirement Capturing with Use Case
  - Identify use cases according to domain analysis
  - Using activity diagram to specify the scenarios
  - Using sequence diagram to specify the interactions between users and system
- Object Modeling
  - Identify key classes involved in requirements
- Dynamic Modeling
  - Specify the behaviors of objects to *realize* the identified requirements

RUCM – Restriction Rules

- RUCM restriction rules are classified into two groups:
  - Restrictions on the use of **natural language** (R1-R16)
  - Restrictions enforcing the use of specific keywords for specifying **control structures** (R17-R26)

RUCM – Restriction Rules (R1-R16)

- Restrictions on the use of natural language (R1-R16)
  - Restriction rules apply only to action steps (R1-R7)
  - Restriction rules apply to all sentences (R8-R16)

RUCM – Restriction Rules (R1-R3)

R1-R3 enforce describing flows of events correctly.

- R1 - The subject of a sentence in basic and alternative flows should be the ***system*** or an ***actor***.—主语是动作发出者
  - The card has been ejected. ==卡已被退出
  - *The system* ejects the ATM card. ==系统退出了ATM卡
- R2 - Describe the flow of events sequentially.—逐步顺次处理
  1. The system ejects the ATM card.==系统退出了ATM卡
  2. The system dispenses the cash amount. ==系统吐出了现金
  1. The system dispenses the cash amount.
  2. The system ejects the ATM card.
- R3 - Actor-to-actor interactions are not allowed.—限制在系统范围内

  The customer gives the teller the ATM card.

  The customer inserts the ATM card into the card reader. ==客户把ATM卡插入到读卡器中

RUCM – Restriction Rules (R4-R5)

- R4—一句话讲清楚一个处理动作
  - Describe one action per sentence. (Avoid compound predicates)
  - Otherwise it is hard to decide the sequence of multiple actions in a sentence.

    …the system cancels the transaction and ejects the card.

    The system cancels the transaction MEANWHILE the system ejects the card.==系统取消交易MEANWHILE系统退出卡
- R5—使用现在时，一般化情况
  - Use present tense only. （中文不适用）

    The system *ejected* the card.

    The system *ejects* the card.
  - Enforce describing what the system does, rather than what it will do or what it has done.

RUCM – Restriction Rules (R6-R7)

R6 and R7 enforce explicitly showing the subject and/or object(s) of a sentence.

- R6--主动语态，理解简单
  - Use active voice rather than passive voice.

    The card *is ejected*.

    The system *ejects* the card.
- R7—明确指出动作发出者和受动者
  - Clearly describe the interaction between the system and actors without omitting its ***sender*** and ***receiver***.

    Customer enters PIN.

    ATM customer enters PIN number *to the system*.==ATM顾客输入PIN码到系统。

RUCM – Restriction Rules (R8-R9)

- R8—声明句，陈述事实
  - Use declarative sentence only. “Is the system idle?” is a non-declarative sentence. 使用陈述句

    Ejects the card.

    The system ejects the card.
- R9--使用一致的术语
  - Use words in a consistent way. Keep one term to describe one thing.

    Customer inserts the ATM card…

    ATM customer inserts the ATM card…

RUCM – Restriction Rules (R10-R11)

- R10--不使用情态动词
  - Don’t use modal verbs (e.g., *might*)

    The system *might* eject the card.

    The system ejects the card.
- R11--不使用副词
  - Avoid adverbs (e.g., *very*)

    The system *very likely* ejects the card.

    The system ejects the card.
- Modal verbs and adverbs usually indicate uncertainty.

RUCM – Restriction Rules (R12-R14)

- R12--简单句，一句话只有一个主语和一个谓语
  - Use simple sentences only. A simple sentence must contain only one subject and one predicate.

    System displays customer accounts *and* prompts customer for transaction type…

    1. The system displays ATM customer accounts. 系统给ATM客户显示账户
    2. The system prompts ATM customer for …
- R13--不使用否定副词和形容词
  - Don't use negative adverb and adjective (e.g., *hardly*, *never*), but it is allowed to use *not* or *no*.

    The PIN number *has never been* validated.

    The PIN number *has not* been validated PIN 码未通过确认
- R14--不使用代词
  - Don't use pronouns (e.g. *he*, *this*)

    …*it* reads the card number.

    …*the system* reads the card number. 系统读取卡号

RUCM – Restriction Rules (R15-R16)

- R15--不使用作为修饰状语的分词短语
  - Don't use participle phrases as adverbial modifier.

    ATM is idle, *displaying a Welcome message.*

    The system is idle. The system is displaying a Welcome message.
- R16—使用"the system" 来指代系统
  - Use "the system" to refer to the system under design consistently.

    "ATM" or "The ATM system"

    the system

RUCM – Restriction Rules (R17-R26)

通过关键词来获得更加准确的语义表达效果

Rule	Description
R17	INCLUDE USE CASE
R18	EXTENDED BY USE CASE
R19	RFS
R20	IF-THEN-ELSE-ELSEIF-ENDIF
R21	MEANWHILE
R22	VALIDATES THAT
R23	DO UNTIL
R24	ABORT
R25	RESUME STEP
R26	Each flow (basic or alternative) has its own post condition.

RUCM – Restriction Rules (R17)

- R17—表达用例之间的Include关系
  - Use keywords INCLUDE USE CASE to describe the include dependencies with other use cases.
  - Grammar
    - INCLUDE USE CASE <included use case name>
  - Explanation
    - The keywords can be used in basic and alternative flows.
  - Example:

    Include Validate PIN abstract use case.

    INCLUDE USE CASE Validate PIN

RUCM – Restriction Rules (R18)

- R18—表达用例之间的extend关系
  - Use keywords EXTENDED BY USE CASE to refer to the extended use case.
  - Grammar
    - EXTENDED BY USE CASE <extending use case>
  - Explanation
    - The keywords can be used in basic and step alternative flows.
  - Example:

    Use case CreateIncident extends the current use case.

    EXTENDED BY USE CASE CreateIncident

RUCM – Restriction Rules (R19)

- R19—明确的流程引用关系
  - Use keyword RFS in a specific (or bounded) alternative flow to refer to a step number (or a lower bound step number and an upper bound step number) of a reference flow step that this alternative flow corresponds to.
  - Grammar
    - RFS <reference flow step #> (specific alternative flow)
    - RFS <reference flow step numbers> (bounded alternative flow)
    - No required notation for global alternative flow.
  - Explanation
    - One specific or bounded alternative flow must correspond to exactly one or more than one reference flow steps.
  - Example:
    - RFS Basic Flow 5 …
    - RFS Basic Flow 5-7, 10, 14 …

RUCM – Restriction Rules (R20)

- R20—分支控制结构
  - Use pairs of keywords of IF, THEN, ELSE, ELSEIF, and ENDIF to describe conditional logic sentences.
  - Grammar
    - IF <condition> THEN <steps> ENDIF
    - IF <condition> THEN <steps> ELSE <steps> ENDIF
    - IF <condition> THEN <steps> ELSEIF <condition> THEN <steps> ENDIF
    - 其中<steps>中每个step占一行，便于阅读
  - Example:

    IF the system recognizes the ATM card

    THEN the system reads the ATM card number ENDIF.

RUCM – Restriction Rules (R21)

- R21—并发表示方式
  - Use keyword MEANWHILE to describe concurrency.
  - Grammar
    - <action> MEANWHILE <action>
  - Explanation
    - It implies that the sentence before keyword MEANWHILE and the sentence after the keyword occur concurrently.
  - Example:

    …the system cancels the transaction and ejects the card.

    …the system cancels the transaction MEANWHILE the system ejects the card.

RUCM – Restriction Rules (R22)

- R22—数据检验是重点（异常处理机制）
  - Use keyword VALIDATES THAT to describe condition checking sentences. VALIDATES THAT means that the condition is evaluated and ***must be true to proceed to the next step***.
  - Grammar
    - VALIDATES THAT <condition>
  - Explanation
    - The alternative case (the condition does not hold) must be described in its corresponding alternative flow.
  - Example:

    … the system checks whether the user-entered PIN…

    …the system VALIDATES THAT the user- entered PIN…

RUCM – Restriction Rules (R23)

- R23—循环控制结构
  - Use keyword pair DO and UNTIL to describe iteration.
  - Grammar
    - DO <steps> UNTIL <condition>
  - Explanation
    - Following keyword DO is a sequence of steps. Following keyword UNTIL is a loop ending condition.
  - Example:

    DO

    action1

    action2

    UNTIL condition

RUCM – Restriction Rules (R24)

- R24—异常处理机制
  - Use keyword ABORT to describe an exceptionally exit action. **An alternative flow ends either with ABORT or RESUME STEP.**
  - Grammar
    - ABORT
  - Explanation
    - Used in alternative flows, iterative, and conditional logic sentences. It ***means the ending of a use case.***

RUCM – Restriction Rules (R25)

- R25—恢复处理机制
  - Use keyword pair RESUME STEP to describe the situation where an alternative flow goes back to its corresponding fork flow(basic flow or alternative flow).
  - Grammar
    - RESUME STEP <fork flow step #>
  - Explanation
    - Used in alternative flows.

任何step动作都不要跨越flow

Example 1

- **PRECONDITION:**
  - Original sentence:
    - ATM is idle, displaying a Welcome message.
  - Rewritten sentence:
    - The system is idle.
    - The system displays a Welcome message.
  - Reason:
    - The original sentence breaks R15, which suggests not using participle phrases as adverbial modifier. Therefore, the original sentence is split into two simple sentence.

Example 2

- **BASIC FLOW STEP 1**
  - Original sentence:
    - Include Validate PIN abstract use case
  - Rewritten sentence:
    - INCLUDE USE CASE Validate PIN
  - Reason:
    - the keyword INCLUDE USE CASE specified as R17, should be used to describe the dependency relationship between use cases *Withdraw Fund* and *Validate PIN*.

Example 3

- **BASIC FLOW STEP 2-4**
- Original sentence: Customer selects Withdrawal, enters the amount, and selects the account number.
- Rewritten sentences:
  1. ATM customer selects Withdrawal through the system
  2. ATM customer enters the withdrawal amount through the system.
  3. ATM customer selects the account number through the system.
- Reason1: the original sentence is **not a simple sentence** (violating R12 and R4); it contains three predicates, which implies three action steps. Besides, it is not clear whether these three actions occur concurrently or sequentially. Therefore, we rewrite the sentence into three sequential action steps: 2-4.
- Reason2: As specified in R9, the **terms** should be used in a **consistent** way; therefore "ATM customer" should be used instead of "Customer: in the original sentence.
- Reason3: R7 says that the interaction between actors and the system should be clearly described without omitting its **sender and receiver**. The original sentence violates R7 and we rewrite the sentence by explicitly saying "…through the system".

Notes for Your Course Projects

- Specify your use cases with RUCM in the given tool
- Tool usages
  - Use **Chinese**, keywords and variables can be in English
  - Report any technical problems in the TA system
  - You should submit your requirement model file, not snapshots in the system
  - For reviewing, you should post the snapshots and the comments in the TA system

Notes for Your Course Projects

- Don’t forget the most important points when specify requirements by using RUCM
  - **User Language**
  - **Completeness** of a requirement (users get the explicit and complete feedbacks from the system in terms of the expectations of users)
  - Keep the logic as **simple** as possible by using the actions steps, keywords, and restriction rules
  - All inputs from users must be **validated**, otherwise the system must go wrong
  - All VALIDATES THAT sentence must have an alternative flow to handle the **exceptional case**

关于用例规格的常见缺陷

- 涉及actor自身动作
- 不写动作主语
- 未使用关键词
- 未合理使用分支流程
- 分支流与引出分支的语句对应不起来
- 不写前置条件和后置条件
  - 每个流程都有一个后置条件
  - 所有的前置条件和后置条件都必须**逻辑可判定**

Object Modeling

- Transfer to use technical language to specify the logics of the system (analysis of requirements)
- Input
  - Use case model (use case specification)
  - Domain analysis
- Output
  - Class model
  - *Realize* the scenarios (specified in use case) with identified classes (objects) (keep at the requirement level)
- Steps
  - Identify classes from use case specification + domain analysis
    - Classes
    - Attributes, Operations
  - Identify the associations among classes

Object vs Class

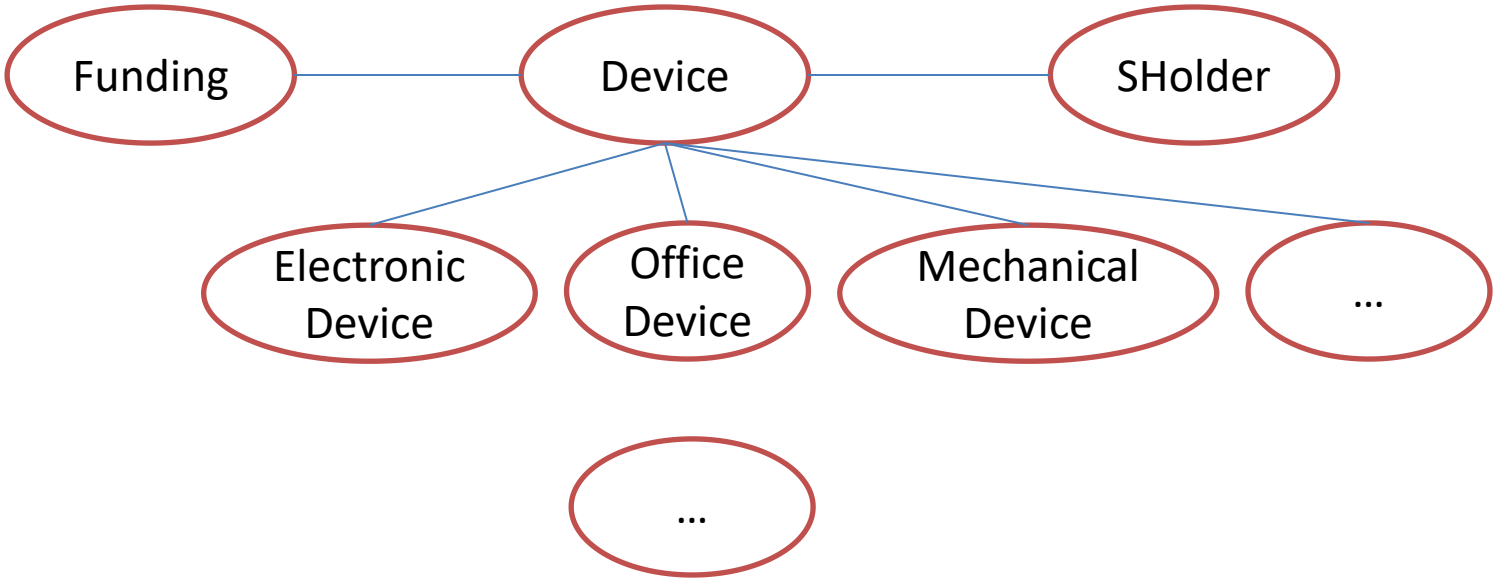
- Object model conforms to class model
  - Class instantiation
  - Association instantiation
  - State machine instantiation
- Objects are concurrently alive
  - Have their own state spaces
  - Independently handle requests

How to find classes?

- Finding objects is a central task
  - Apply domain knowledge to establish domain taxonomy
    - Hierarchy of terms: specialization, composition, connection
  - Find participating objects in the scenarios of use cases
  - Syntactically analyze the *domain analysis* and *use case specification*
    - Nouns for class candidates
    - Verbs for class operation candidates

Establish Domain Taxonomy

- Take Device Mgmt Sys. as an example
  - Device and its categories
  - Stakeholder
  - Type of funding used to buy the device



Finding Participating Objects in Use Cases

- Pick a ***use case*** and look at its ***flow of events***
  - Find terms that developers or users need to clarify in order to understand the flow of events
  - Look for recurring nouns (e.g., *Accident*),
  - Identify real world entities that the system needs to keep track of (e.g., *Field Officer*, *Dispatcher*, *Resource*),
  - Identify real world procedures that the system needs to keep track of (e.g., *Emergency Operations Plan*),
  - Identify data sources or sinks (e.g., *Printer*)
  - Identify interface artifacts (e.g., *Police Station*)
- Some objects are still missing and need to be found:
  - Model the flow of events with a sequence diagram
- Remember to use the domain terms

Example: Lab Device Mgmt

- Scenario
  - 管理员给设备经费负责人和使用负责人发布设备状态核查通知
  - 负责人登录系统，自动获得需要核查的设备列表
  - 系统自动但列出设备经费负责人和使用负责人交叉的设备
  - 负责人安排小组设备管理员进行核查
  - 负责人根据核查结果更新设备状态
  - 通知管理员完成核查
  - 管理员确认，并返回可能的问题

Try to identify possible classes according to the scenario and the taxonomy.

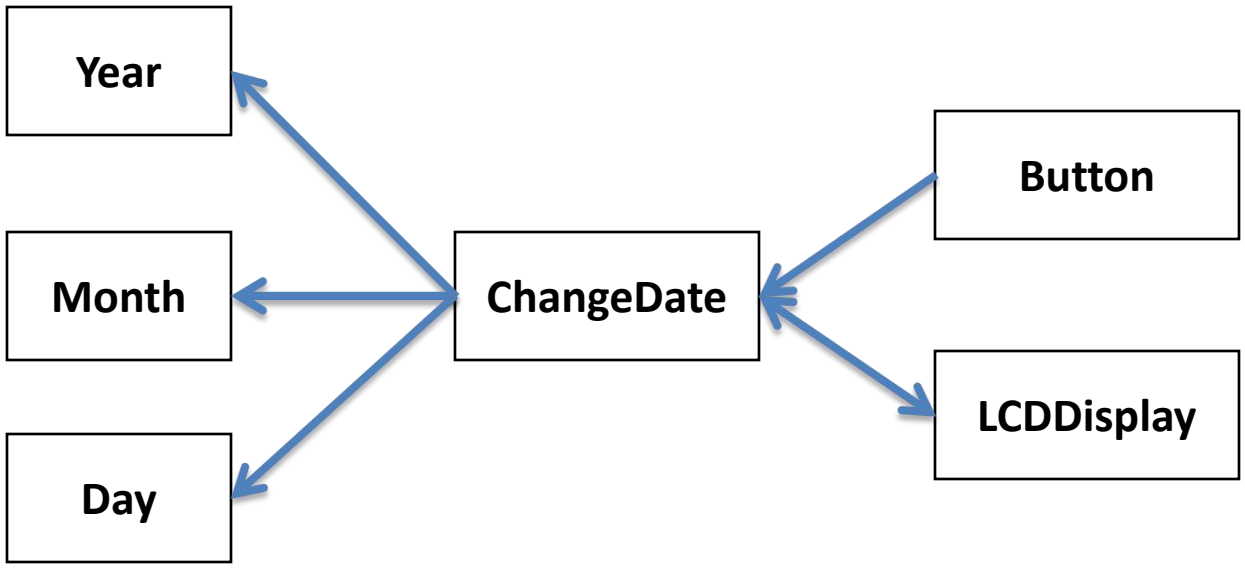
Class Model Abstracts Taxonomy

- Taxonomy simply organizes the terms from domain analysis
- A term does not necessarily to be a class
  - Different state?
  - Different operation/behavior?
- We might have many different devices captured in the domain analysis
  - But we may not differentiate these devices in this system
  - We can simply add a *DeviceType* in the Device class to abstract all kinds of devices

Object Types

- Entity Objects
  - Represent the persistent information tracked by the system (Application domain objects, “Business objects”)
- Boundary Objects
  - Represent the interaction between the user and the system
  - We don’t care how the interactions will be realized, but need to know there is boundary object in analysis model.
- Control Objects:
  - Represent the control tasks performed by the system
  - We don’t care how the control will be realized, but need to know there is control object in analysis model.

Example: 2BWatch Objects



Entity Object

- Entity object usually has persistence requirement (to be managed)
- Usually has interesting states
  - State: identify from application domain
  - Attribute: object state space
  - Operation: trigger of state transition
- Different systems may see different interesting states for the same entity
  - Example: Book in bookstore application, in library application

Boundary Object

- Identify from the interactions between the system and actors
  - Input/Output object
  - Device/mechanism to enable the interaction
- Usually associated with use case constraints
  - Precondition and postcondition
- Example: typical constraints for the order in B2C application

Control Object

- A ‘center’ or ‘mediator/coordinator’ serves for negotiating with multiple boundary and entity objects to realize some functionalities.
- Control object embodies the task independent of specific entity object and boundary object
  - Some task is associated with specific entity or boundary object ---> as object operation
- Example: AssignStock, AssignGoods are typical tasks for B2C application

Control Object

- Think about virus propagation problem
  - A virus can propagate from infected machine to others through several ways (access, program execution, …)
  - Infected machine usually show some specific symptoms (e.g. unexpected communication, heavy load, etc.)
  - Some antivirus tools are effective to detect and kill the virus

Case：Online Examination System

- Domain Analysis
  - (Teacher) can create examination question (with difficulty and score information) at anytime to store in the examination base
  - (Teacher) can specify the criteria to filter the questions
  - (Teacher) can select questions to build an examination
  - (Teacher) can publish the examination to students
  - (Student) can see the examination only within specified time frame
  - (Student) can submit answers through the system
- Identify the three types of objects

Object Modeling within MDA Context

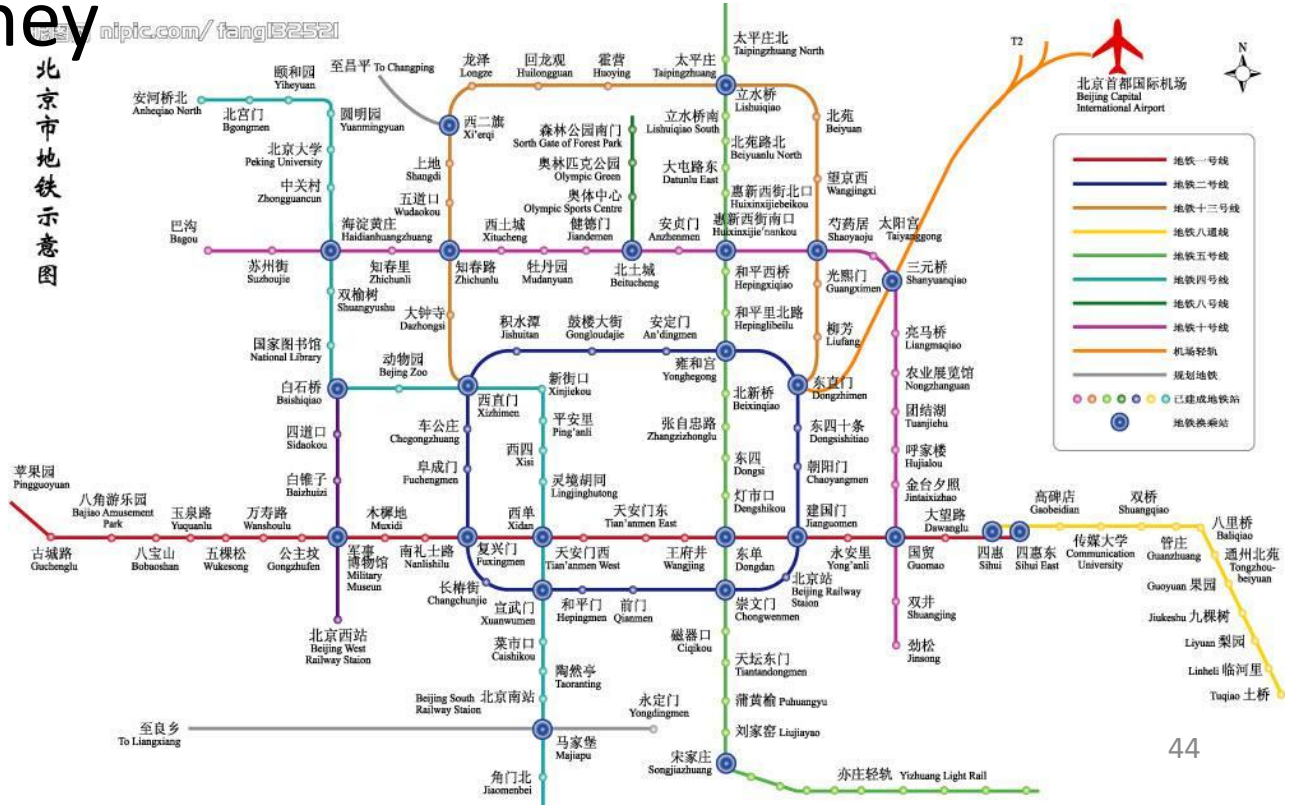
- Entity object and Control object both capture the **platform independent** concept
- Boundary object is strongly associated with specific platform
  - Button, LCDDisplay
- Be careful not to let **platform specific** information leak into PIM
  - Treat boundary object as channel, focus on the information through the channel

Object Modeling within MDA Context

- For 2B Watch
  - Button → operation mode + operation (increase/decrease by one)
  - LCDDisplay → display mode + display methods
- Different system objectives might need different methods to identify objects.

Object Modeling

- Goal: Journey planner / ticketing system
- Line, Station, Transfer Station, End Station
- Ticket, Journey



Class diagram in PIM phase

- Focus on application domain at first
  - Abbr. analysis model
- Then, solution domain class
  - Abbr. design model
  - Without hardware platform specific class: RS232Connection
  - Without software platform specific class: CorbaAgent
  - Can be common technical platform class: HashTable, Queue, Stack, Map, etc

Application domain vs solution domain

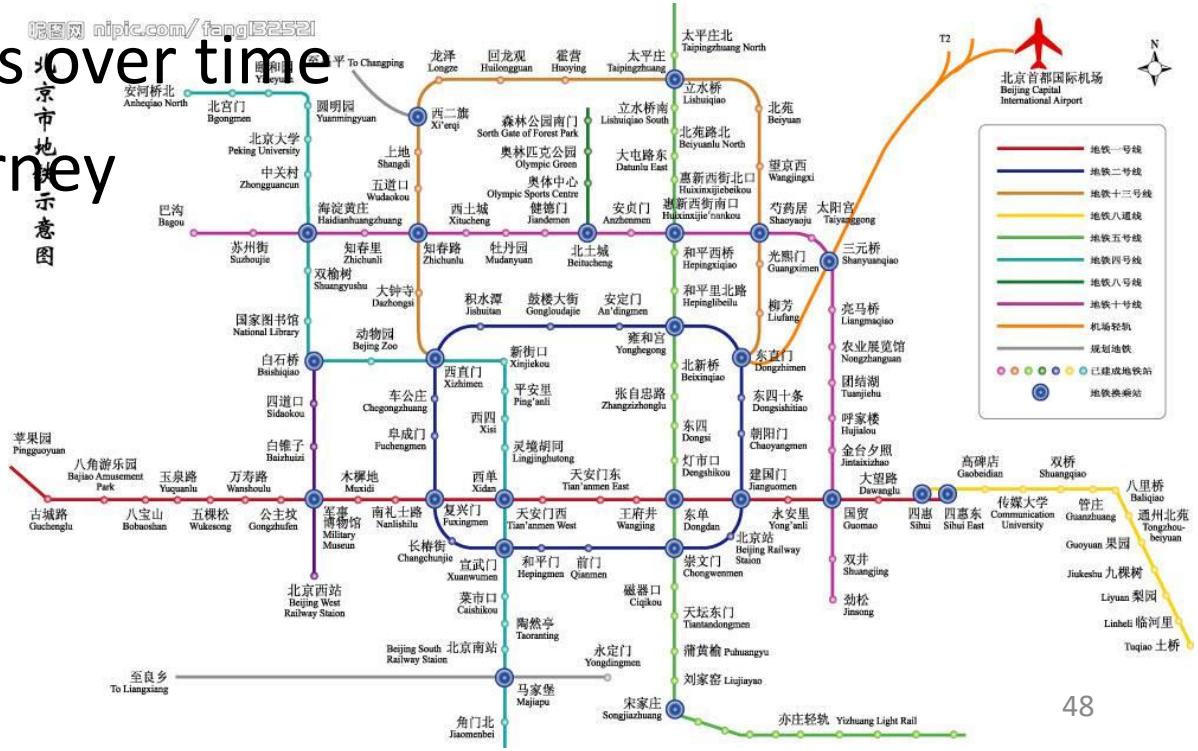
- Application domain:
  - The problem domain (financial services, accident management, architecture, …).
- Application domain class:
  - An abstraction in the application domain, also called business objects.
  - Example: Board game, Course
- Solution domain:
  - Domains in which the solution of problems (tele communication, data bases, compiler construction, operating systems, ….) is specified
- Solution domain class:
  - An abstraction for technical reasons to specify a solution
  - Examples: Tree, Hashtable, Scheduler
- Both domains can be platform independent!

Check Class Model Dynamically

- Instantiate objects from class model
- Realize the scenarios (in use case) with the instantiated objects
- Analyze the change of states of the objects along the scenarios
- Check
  - Whether the class model can still generalize all the objects
  - Whether the constraints in the class model were violated by the objects

Case Study

- View from objects
  - Takes Journey Planner as case
  - Instantiates objects (Line, Station)
  - Object changes over time
  - Check the Journey



Dynamic Modeling

- In requirement modeling phase
  - Object behavior: state diagram
  - Object communication: sequence diagram
  - Use case/system process/flow: activity diagram
- How
  - Transform UCS into sequence diagram or activity diagram -->formalized model for checking the semantics
  - Identify key objects and its state models
- Behavior model in further can help find more attributes and operations

Event and Message

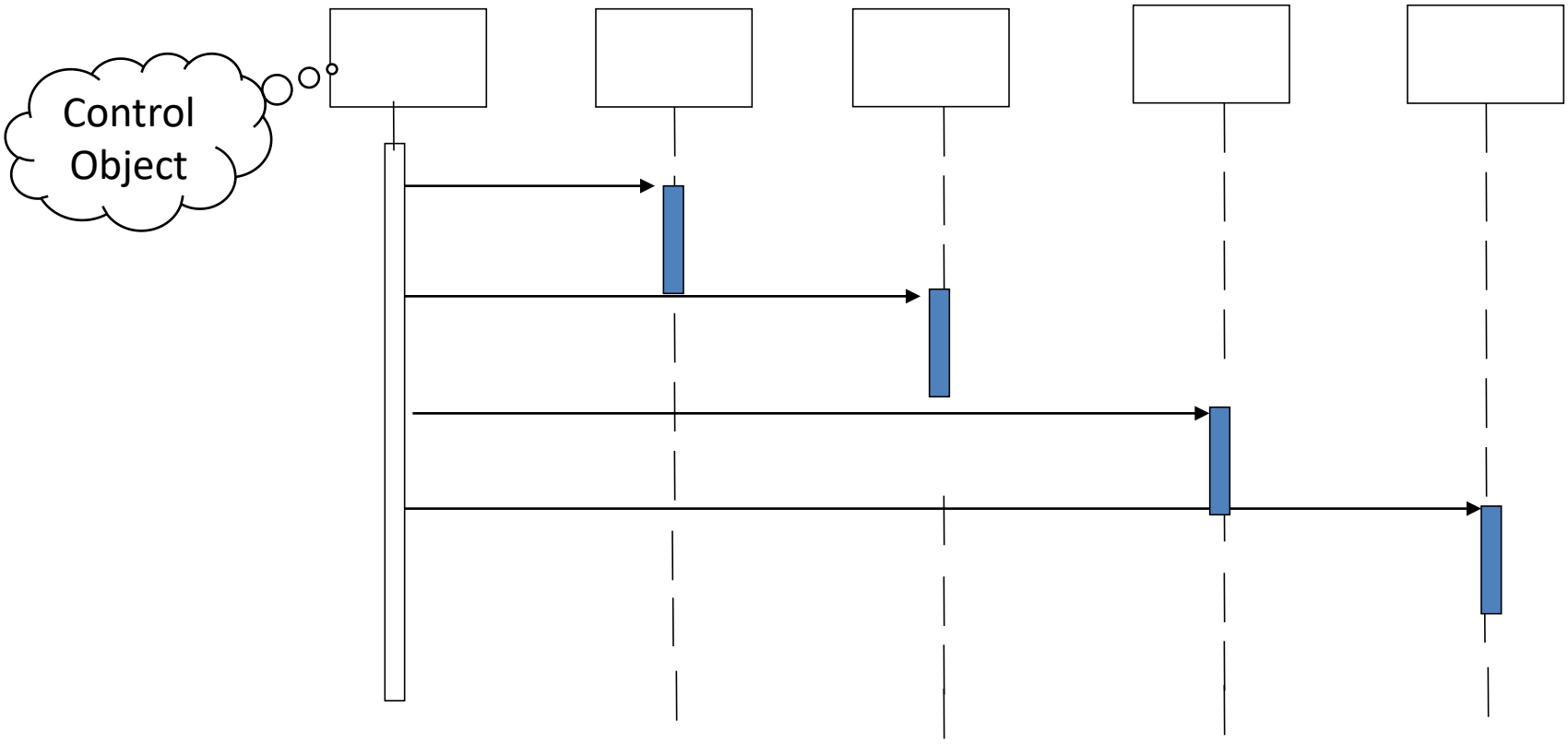
- Event: Something interesting that happens at a certain time point
  - Trigger of state transition
  - Don’t care who delivers the event
- Message: (event+data) with (source, destination)
- Events can have associations with each other:
  - Causally related:
    - An event happens always before/after another event, or together with another event
- UML does not strictly differentiate these two concepts
  - We use ‘event’ in state diagram and activity diagram
  - We use ‘message’ in sequence diagram

Sequence Diagram

- Heuristic for finding participating objects:
  - A message always has a sender and a receiver/(s)
  - Find them for each message => These are the objects participating in the use case.
  - Chances to identify operations.
- Sequence diagrams can be derived from use cases
  - But not limited, you can use it to model threads or components collaboration
- The structure of the sequence diagram helps us to determine how decentralized a system is
- Two structures for sequence diagrams
  - Fork Diagrams and Stair Diagrams (Ivar Jacobsen)

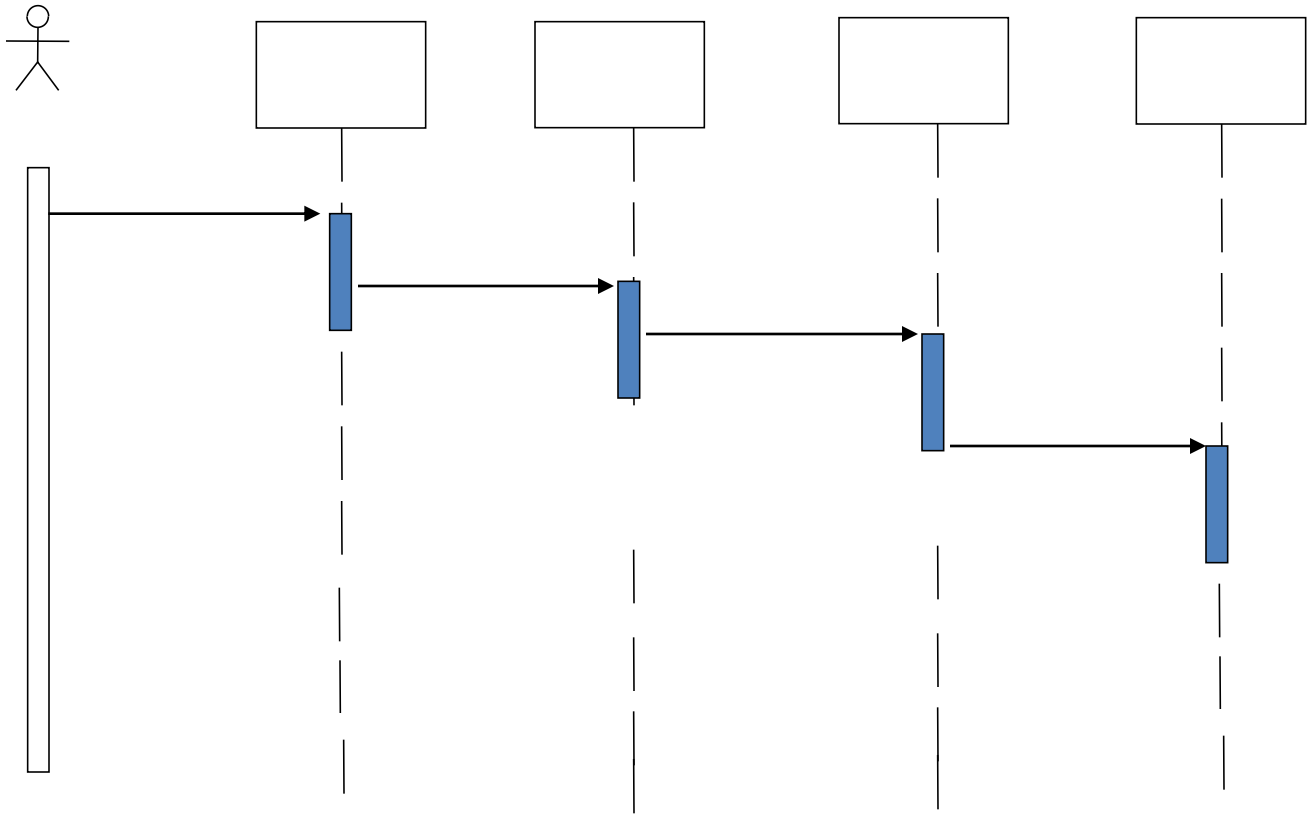
Fork Diagram

- The dynamic behavior is placed in a single object, usually a control object
  - It knows all the other objects and often uses them for direct questions and commands



Stair Diagram

- The dynamic behavior is distributed. Each object delegates responsibility to other objects
  - Each object knows only a few of the other objects and knows which objects can help with a specific behavior



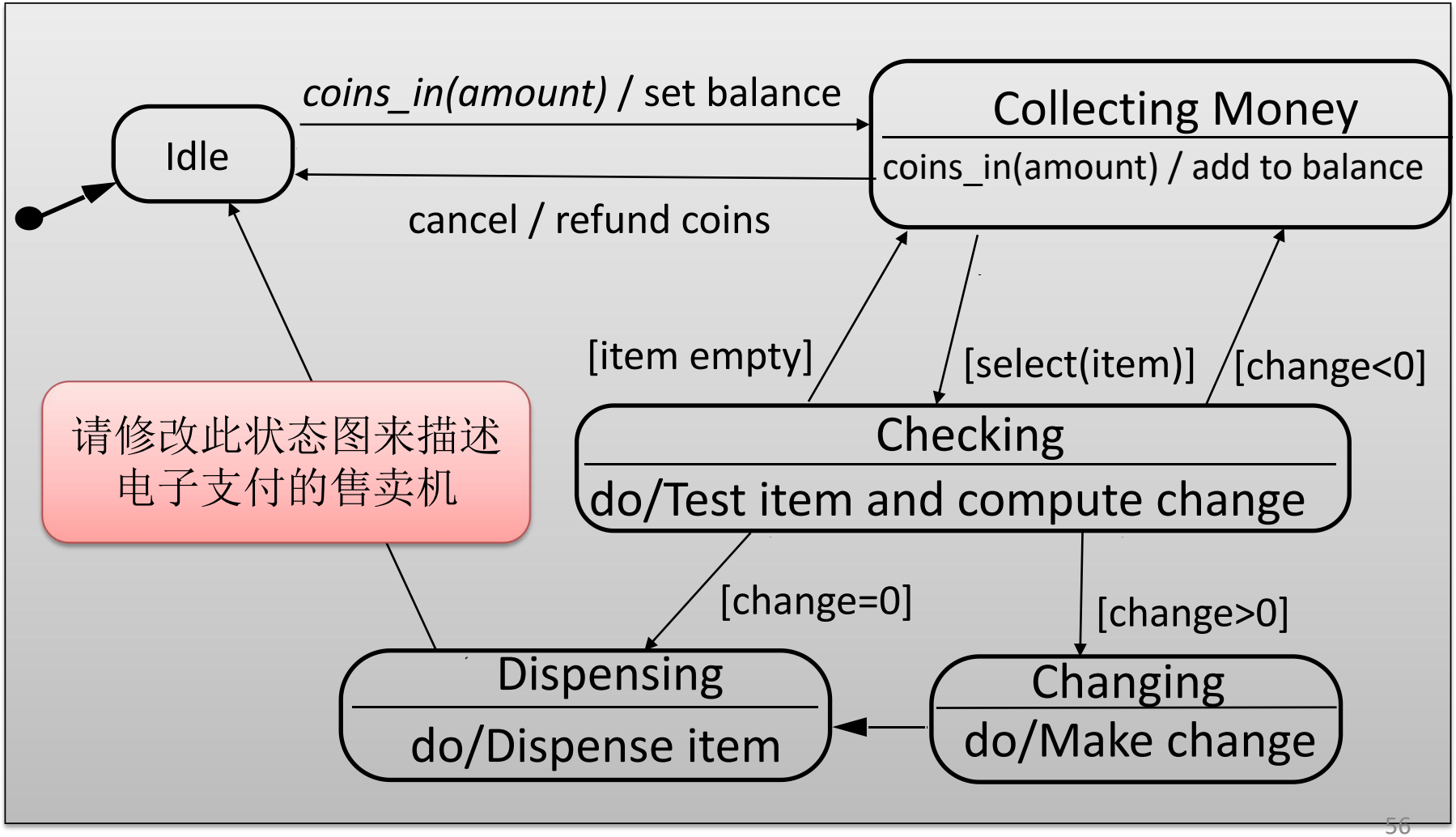
Fork or Stair?

- Object-oriented methodology claims that the stair structure is better
  - Localized or decentralized responsibilities
- Modeling Advice:
  - Choose the stair - a decentralized control structure - if
    - The operations have a strong connection
    - The operations will always be performed in the same order
  - Choose the fork - a centralized control structure - if
    - The operations can change order
    - New operations are expected to be added as a result of new requirements.

State based Modeling

- All objects instantiated from the same class have their own instances of the state machine
  - You can check its current state, state changing trace at anytime
- All live objects concurrently act according to its behavior specification, i.e. the state machine instance
- Interactions among objects in fact are driven by those instances of state machines

Example: vending machine



State

- An abstraction of the attributes configuration of an object
- A state is an equivalence class of all those attribute values and links that do no need to be distinguished
  - Example: State of a mobile phone
- State has duration within which the object keeps unchanged in terms of state

State and Event

- An object should take different reactions to events in different state
- If there is no such an event to differentiate object behavior in two states, the two states should be merged
- Object is both an event producer and an event consumer
  - objects are stimulated to go concurrently

分析模型的主要问题

- 按照图的分工模式
  - 图与图之间没有逻辑关联
- 按照用例的分工模式
  - 每个人都需要构建各种模型图
- 所有模型图都只是对模型的一个观察，必须确保形成模型
  - 图之间具有关联
  - 图之间的逻辑一致性

Cases

- Try to identify the states and transitions of a device in the device mgmt system
- For examination question lib system, figure out the states of an examination question
  - Note: a question can be selected in various exam
- Try to identify the states and transitions of book in library system

Quiz

- 遥控玩具汽车的控制系统
  - 使用电池驱动
  - 能够控制向左、向右拐弯；前进和后退
  - 可以停止和继续前进
  - 控制系统拥有遥控信号接收器、控制拐弯的马达、控制前进或后退的马达、感知轮子是否着地的传感器
- 请识别类及其关系，并分析控制系统和汽车的状态及其迁移关系。

Requirement Modeling Activities

- 2018.10.29：各个组提交经过了内部评审的需求分析模型，开始组间评审。提交物包括：
  - 用例规格(RUCM模型文件)
  - 系统架构图/组成拓扑图(嵌入式系统)
  - 改进的领域分析报告
  - UML模型截图截文字说明
- 2018.11.3：完成组间评审
  - 评分（考验你能发现多少问题）
  - 文字+截图（要指出问题，这是帮助别人）
- 2018.11.5：课堂讨论需求模型评审结果