Software Engineering 300:
Introduction to Software Engineering

Software Requirements Modelling

Why model the software requirements?

How can the software requirements be modelled?

What are the strengths/weaknesses of different approaches?

1

Remember: Requirements: The basics

- The requirements for the system define:
  - WHAT the system should do
  - Not HOW the system should do it
- Do they matter?
  - Without them, how do you know what to do?
  - Without them, how do you know when you've achieved the "goal"?

Remember: What is the point of modelling?

- Understanding
  - Getting the point
  - Finding problems
  - Considering how to “do it”
- Explaining

Modelling approaches
a.k.a. specification approaches

- Natural language text
- Structured natural language
  - *structured requirements specification*
- Formal specification
- User stories
- Scenarios
- Use cases

EXTERNALLY VISIBLE BEHAVIOUR

- (Feature lists)

Example

Consider the requirements for the software to run a vending machine …

Natural language text

Our vending machines let customers buy pop by entering Canadian coins and selecting the pop of their choice from a set of options. The total value of all coins entered in the machine and not used in a previous purchase shall be displayed to the customer. When the customer selects a pop and sufficient funds are available, the total will be reduced by the amount of the cost of that pop, the pop will be vended, and the change returned to the customer. If insufficient funds are present, a message will be displayed to that effect. If no pops remain for the selection, a message will be displayed to that effect. A technician will be able to set the price of each available pop, load pops into the machine, and load/empty coins to serve as change. When the machine is open for servicing, other functionality must be disabled, for safety. We are also interested in some sort of "customer loyalty card", whereby customers can prepay amounts and receive discounts on products. And we are also considering having the machines communicating directly with the central office, to tell us when they are full of coins, out of product, etc. And they have to be purple because it's our company colour.

Natural language text: Issues

- Pros
  - Can be read
  - No special training needed
- Cons
  - Poor scalability
  - Mixture of major points and minor details
  - Hard to analyze
  - No organization, structure
  - Hard to navigate & manage

Structured natural language

- Adds (imposes) a hierarchical structure
- Traditional approach

1. The system must accept coins

1.1. Coins that are not valid Canadian currency will be rejected

1.1.1. Rejected coins will be returned

1.2. …

Structured natural language: Example

1. Coin entry
    1. Vending machines shall accept Canadian coins.
    2. Vending machines shall track the total value of all coins entered in the machine not used in a purchase.
2. Pop selection
    1. Customers shall select the pop of their choice from the available kinds.
    2. A pop shall be vended when it is selected and its cost is less than the current total available.
        1. The current total available will be reduced by the amount of the cost of the selected pop.
        2. The remainder shall be returned to the customer.
        3. The pop shall be delivered via the delivery chute.
    3. When the cost of a selected pop is greater than the current total available, it shall not be vended.
        1. An error message will be displayed.
        2. The total available will not change.
    4. When a pop is selected which is currently out of stock, no pop shall be vended.
        1. An error message will be displayed.
        2. The total available will not change.

Structured natural language: Example

3. Display
    1. The currently unused total shall be displayed when other information is not being displayed.
    2. Error messages and informational messages will be displayed temporarily [duration not indicated] and then the currently unused total shall be redisplayed.
4. Service
    1. A technician will be able to set the price of each available pop.
    2. A technician will load pops into the machine .
        1. The totals available will be automatically updated.
    3. A technician will load/empty coins to serve as change.
    4. When the machine is open for servicing, other functionality must be disabled, for safety.
5. Customer loyalty card
    1. Customers can prepay amounts and receive discounts on products.
    2. Prepaid amounts shall be recorded on a card [technology details not specified].
6. Networked communication
    1. Vending machines will communicate directly with the central office, to inform when they are full of coins, out of product, etc.

Structured natural language: Standardization

1. **Introduction**
   - 1.1 Purpose
   - 1.2 Document conventions
   - 1.3 Intended audience
   - 1.4 Additional information
   - 1.5 Contact information/SRS team members
   - 1.6 References
2. **Overall Description**
   - 2.1 Product perspective
   - 2.2 Product functions
   - 2.3 User classes and characteristics
   - 2.4 Operating environment
   - 2.5 User environment
   - 2.6 Design/implementation constraints
   - 2.7 Assumptions and dependencies
3. **External Interface Requirements**
   - 3.1 User interfaces
   - 3.2 Hardware interfaces
   - 3.3 Software interfaces
   - 3.4 Communication protocols and interfaces
4. **System Features**
   - 4.1 System feature A
   - 4.1.1 Description and priority
   - 4.1.2 Action/result
   - 4.1.3 Functional requirements
   - 4.2 System feature B
5. **Other Non-functional Requirements**
   - 5.1 Performance requirements
   - 5.2 Safety requirements
   - 5.3 Security requirements
   - 5.4 Software quality attributes
   - 5.5 Project documentation
   - 5.6 User documentation
6. **Other Requirements**

Appendix A: Terminology/Glossary/Definitions list

Appendix B: To be determined

Structured natural language: Issues

- Pros
  - Standardization!
  - Details!
  - Can be read
  - No special training needed
- Cons
  - Standardization!
  - Details!
  - Poor scalability
  - Mixture of major points and minor details
  - Hard to analyze
  - Poor organization, structure
  - Hard to navigate & manage

Formal specification

```
\begin{axdef}
amenu: (v_arrow \cross SELECTION \cross selection) \fun SELECTION
\where
        \forall s: selection @ (let n == # (valid s) @
        \forall a: v_arrow; i: SELECTION @ default_selection \leq n \land amenu(a,i,s) \leq n)
\end{axdef}
\begin{schema}{GetMenuArrow}
        EventUnlocked
        \Delta Menu
\where
        input? \in v_arrow
        menu_item' = amenu(input?,menu_item,item)
        Continue
\end{schema}
```

Formal specification: Issues

- Pros
  - Precision
  - Provability
  - Analyzability
- Cons
  - Special skills needed
  - Difficulty to check for "does it make sense?"
  - High cost to produce and to maintain

User stories

- “As a customer, I want to select the pop to purchase.”
- “As a customer, I want to know the price of a pop.”
- “As a technician, I want the vending machine to be deactivated when it is unlocked so that I cannot be injured when servicing it.”

- As a <role>, I want <feature> [so that <reason>].

User stories: Issues

- Pros
  - Focus on user experience
  - Can be read
  - Little special training needed
  - Non-technical people can **WRITE** them
  - Fast & cheap
- Cons
  - Questionable scalability
  - Not all requirements are visible to end users
  - Extra effort needed to manage
  - No particular support for navigation
  - Hard to analyze
  - False sense of simplicity?

Scenarios

"Ming inserts $2 in Canadian coins in the vending machine. He decides which of the available kinds of pop he would like, and selects it. The pop costs less than $2, so the pop is dropped into the delivery chute, and his change is returned through the coin return."

- Simple description of specific interaction sequence
- Multiple scenarios needed for other possibilities

Other scenarios (condensed)

- Ming changes his mind altogether and wants his money back
- Ming doesn't have enough change, and wants his money back
- The machine doesn't have enough of the pop that he chose
- The machine doesn't have enough change
  - When does Ming find out?
- Ming decides that he would like to return his pop purchase

Scenarios: Issues

- Pros
  - Focus on user experience
  - Can be read
  - Little special training needed
  - Non-technical people can **WRITE** them
  - Fast & cheap
  - Good as concrete examples
- Cons
  - Very poor scalability
  - You can get lost in unimportant details
  - Not all requirements are visible to end users
  - Extra effort needed to manage
  - No particular support for navigation
  - Hard to analyze
  - False sense of simplicity?

Use cases

- A use case is an abstraction of one or more scenarios
- Each use case describes **externally visible** behaviour involved in some “task”
- Use cases are usually represented by combination of diagram and structured text

Use cases: Actors

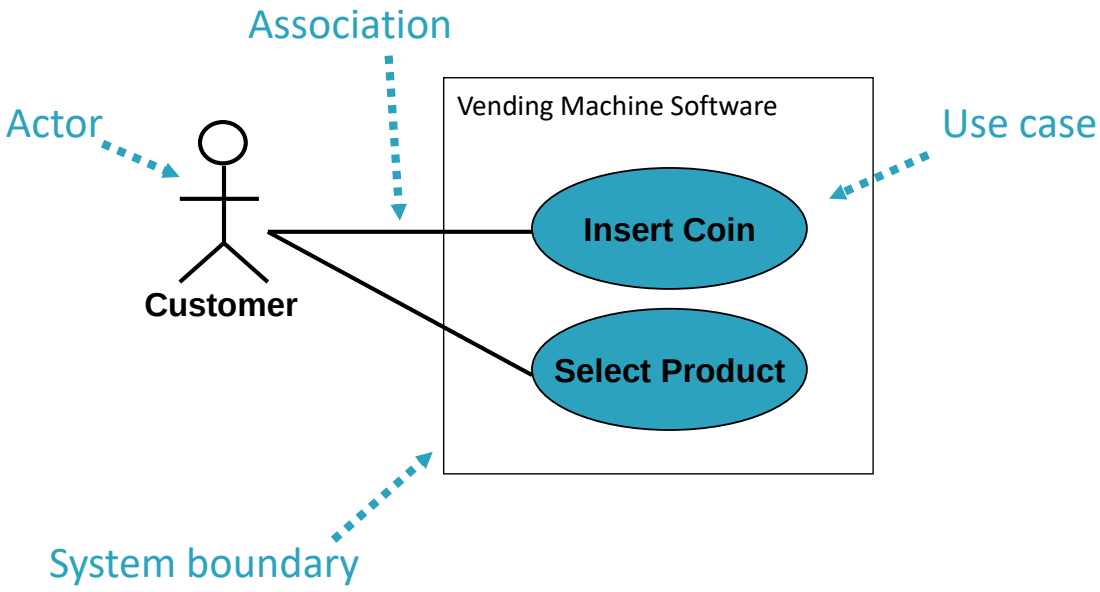
- Every use case must involve one or more **actors**:
  - A person playing a role (e.g., instructor)
  - A physical device
  - External software
  - External organization
- An actor is **NOT** part of the system:
  - It is important to know what the system is and is not!

Use cases: Representation

- Typically, use cases are represented as a combination of …
  - A graphical model, called a **Use Case Diagram**
  - A structured textual description, called a **Use Case Description**

- There **can be** relationships between actors and between use cases

Use case diagrams: Basic



Use case descriptions

- Generic, step-by-step written description of a use case's event flow
  - Describe preconditions (initial system state)
  - List sequence of simple steps
  - Describe postconditions (resulting state)
- Includes interactions with actor(s), and describes what objects are exchanged

Use case descriptions: An example template

Name: Insert Coin

Participating actor(s): Customer

Precondition: none

Main flow of events:

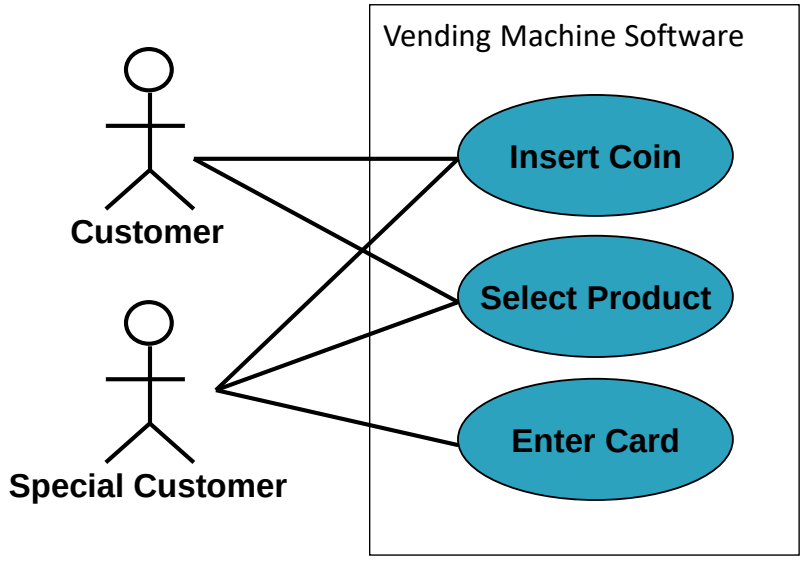
1. Customer inserts a coin
2. Value of coin is added to current total
3. Revised current total is displayed for the Customer

Postcondition: Current total is updated

Alternative flow:

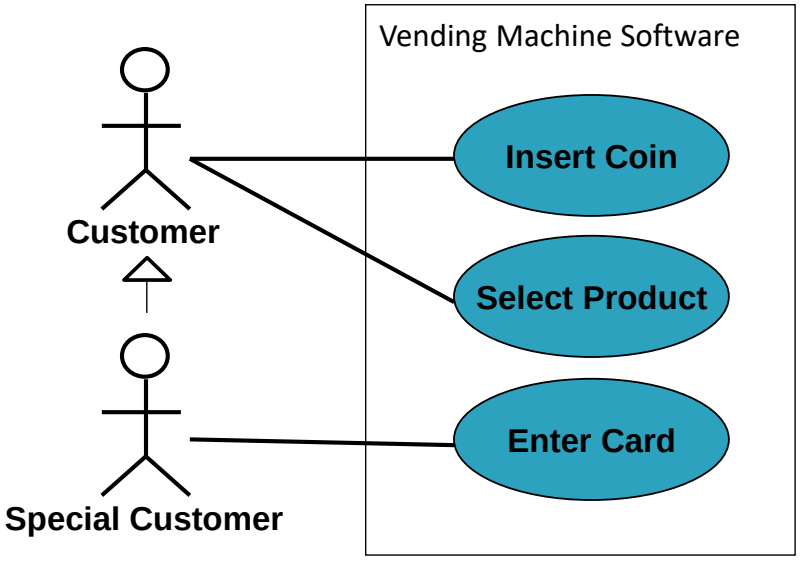
2. Invalid coin is returned to Customer
3. Current total is not revised

Use case diagrams: Actor relationships



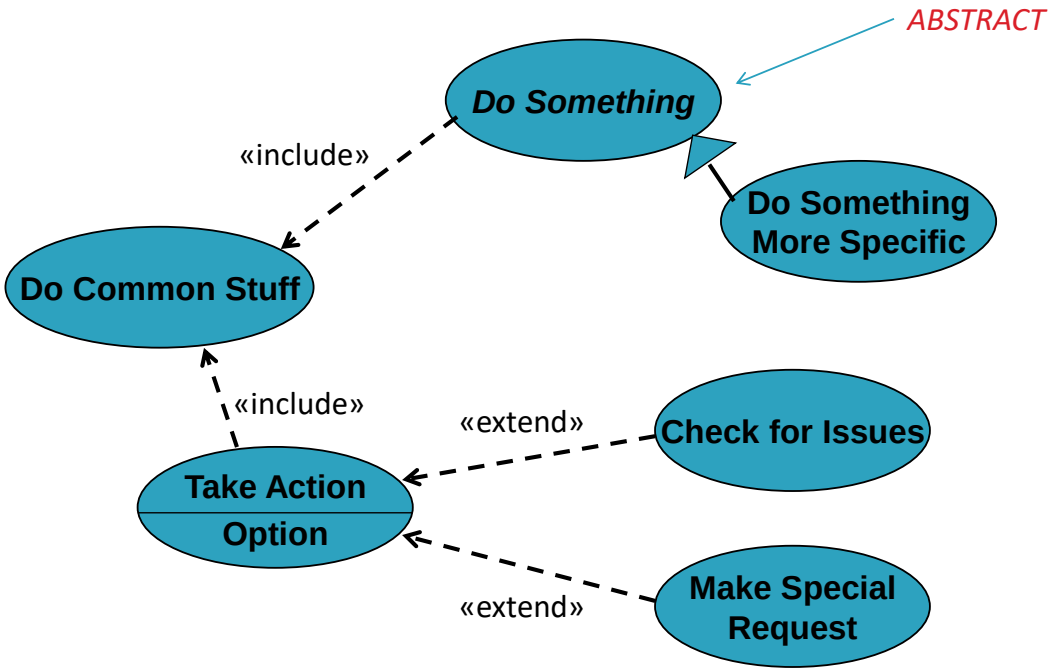
WRONG!

Use case diagrams: Actor relationships



RIGHT!

Use case diagram: Use case relationships



Preconditions versus relationships

- Common mistake:
  - details on diagram that belong just in the descriptions
  - e.g., preconditions
- For example, a login use case will generally not be connected to anything else

Exercise

- What are the use case diagram and descriptions for the vending machine software?

Use cases: Issues

Pros

Cons

At this point and based on what we have figured out up to here, you should be able to figure out what is good and bad about use cases. Try.

Next time

- Software Libraries, Object-Oriented Frameworks, and APIs