Lecture 7

Use Cases

Requirements Specification & Documentation (II)

SOEN 342, Fall 2017

René Witte
Department of Computer Science
and Software Engineering
Concordia University

Slides Credit

Includes slides by Ian Alexander [Ale03]

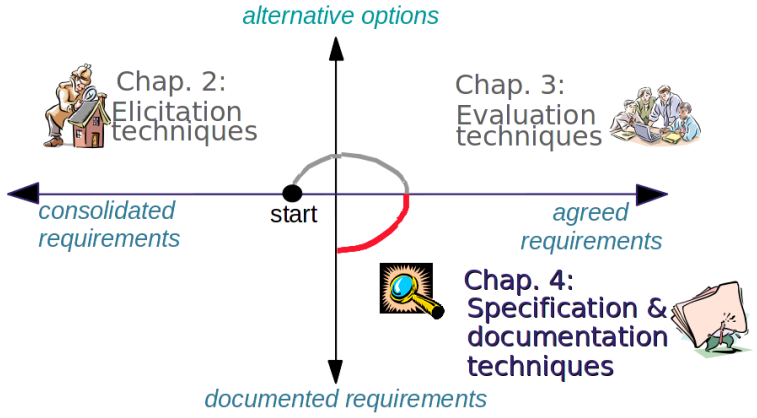
Outline

René Witte

Concordia University

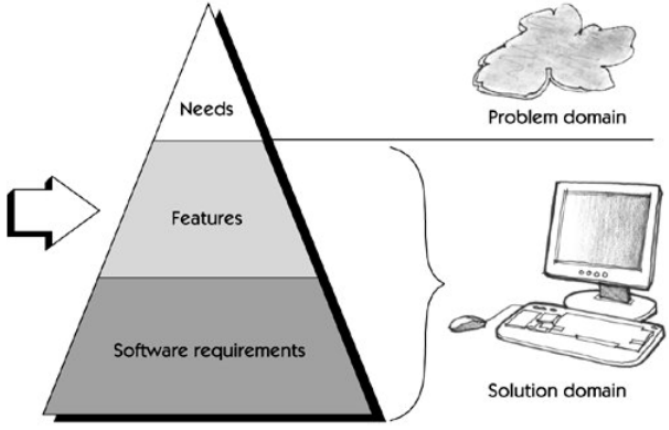
Requirements Specification & Documentation

René Witte



Moving from Problem Domain to Solution Domain

René Witte



René Witte

A Use Case. . .
describes the use of the system in terms of a series of interactions between the actors and the system that yield an observable result of **value** to a particular **actor**.

Use Cases

René Witte
Concordia University

Features

- Are a part of the SRS defined in the UP
- Contain actors
- Describe an observable value delivered by the system to an actor
- Capture functional requirements
- Are derived from the *Vision Document*
- Collect a number of (positive) scenarios, both normal and abnormal

Use cases tell a story about what the system is doing—understandable to both the developer and the user!

Why Use Cases?

René Witte
Concordia University

"Old-style" SRS

Long lists of requirements, structured by sub-system:

```
2.1 Data Entry
2.1.1 Data Entry Subsystem
2.1.1.1 User shall login to system
...
2.2 Report Generation
...
2.2.4.2.1 Reports must be sortable by due date or entry date
```

(commonly associated with Waterfall)

Problem with this?

- After reading dozens [hundreds] of pages of these, you still have no idea what the system is actually *doing!*
- Because they lack context and cohesion

Real-World Example from NASA

René Witte

WIRE C&DH Flight Software Requirements Specification

```
041.1.1 The flight software shall receive data
in the form of CCSDS packets from the Software Bus,
and shall transmit the data over the 1553B bus.

041.1.2 The flight software shall collect data from
the 1553B remote terminals and shall transmit the
data on the Software Bus in the form of CCSDS packets.

041.1.3 If necessary, the flight software shall use
multiple 1553B transactions to collect or transmit
a single CCSDS packet.
```

[you can find this spec in the Internet Archive: https://web.archive.org/web/20111015090659/http://sunland.gsfc.nasa.gov/smex/wire/mission/cdhsw/wirrqtop.htm]

Why Use Cases? (II)

René Witte

Use Cases are like stories

- How can an actor use the system to achieve some result?
- Describe something useful and meaningful
- Have a plot—a sequence of steps
- Have a clear ending—delivering some result

Use Cases do not talk about design or implementation!

Outline

Supporting Artifacts

René Witte
Concordia University

Use Cases alone are not a complete SRS

In the UP, the Use Cases are only one part of the SRS, which includes:

- Vision Document
- Glossary and/or Domain Model
- Supplementary Specification
  - Functional requirements that are not part of the Use Case model
  - Reliability, Performance, Supportability, other NFRs, . . .
  - Design Constraints
  - Legal Notices, Licenses, . . .

All of these artifacts and requirements must be cross-referenced with Use Cases (for traceability, covered later in this course)

Use Cases and "Adding Value"

René Witte
Concordia University

Actor
something with behaviour, such as a person, computer system, or organization, e.g., a cashier

Scenario
a specific sequence of actions and interactions between actors and the system under discussion, e.g., the scenario of successfully purchasing items with cash

Use Case
a collection of related success and failure scenarios that describe actors using a system to support a goal

Use Cases and "Adding Value" (II)

René Witte
Concordia University

Example

UC: Handle returns

Main success scenario:

- A customer arrives at a checkout with items to return. The cashier uses the POS system to record each returned item...

Alternate scenarios:

- If the credit reimbursement authorization is rejected, inform customer and ask for an alternative payment method.
- If item identifier not found in the system, notify the Cashier and suggest manual entry of the identifier code.
- ...

Building the Use Case Model

René Witte

Step-By-Step

Step 1: Identify and describe the *Actors*

Step 2: Identify the Use Cases and write a Brief Description

Step 3: Identify the Actor/Use Case Relationships

Step 4: Outline the Individual Use Cases

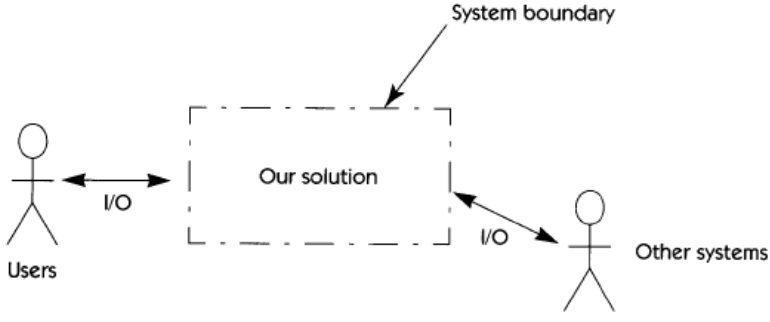
Step 5: Refine the Use Cases

System Boundary is the Context

René Witte

System Boundary

- System Boundary defines the border between the Solution (our system) and the Real World (things that interact with our system).
- World partitioned (i.e., no sharing) in two:
  - Our system
  - Things that interact with our system



How to Identify *Actors*

René Witte

Some helpful questions [LW03]

- Who will supply, use, or remove information from the system?
- Who will operate the system?
- Who will perform any system maintenance?
- Where will the system be used?
- Where does the system get its information?
- What other external systems will interact with the system?

Primary vs. Secondary Actors

Primary Actor

- Interacts to achieve required system function and derives the intended benefit from the system
- Works directly and frequently with the system
- (Is usually positioned on the left side of the Use Case diagram)

Secondary Actor

- Supports the system so that primary actors can do their work
- (Is usually positioned on the right side of the Use Case diagram)

René Witte

Step-By-Step

Step 1: Identify and describe the Actors

Step 2: Identify the *Use Cases* and write a Brief Description

Step 3: Identify the Actor/Use Case Relationships

Step 4: Outline the Individual Use Cases

Step 5: Refine the Use Cases

Small Example: Wings Over the World Company

Vision (Objectives and Goals) [Coc01]

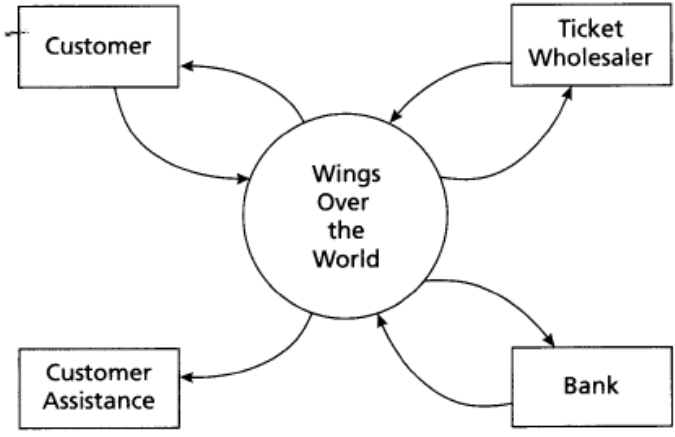
Wings Over the World endeavors to maintain its reputation for innovation by increasing access to our travel services and by offering new and innovative services that are unmatched by our competitors. Specifically:

- Increase brand awareness of Wings Over the World with the creation of a public Web site.
- Increase market share by 15 percent and lower the cost of booking tickets by letting 30 percent of our clients book on-line.
- Open the Wings Over the World travel system to independent travel agents by offering premium booking services and therefore create a new revenue stream.

Wings Over the World Company: Context

Context Diagram

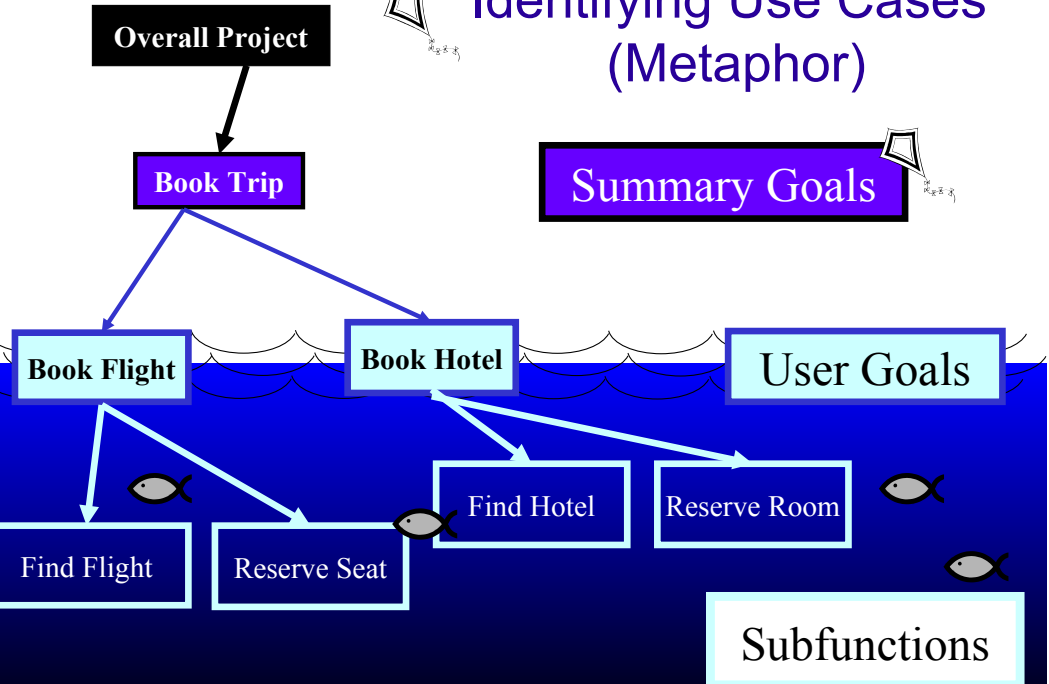
- a possible starting point for building the Use Case set
- (not a part of UML)



Context diagram for Wings Over the World

René Witte

Identifying Use Cases (Metaphor)



René Witte

User-Goal Level Use Cases

Properties

Satisfies a particular and immediate goal of value to the primary actor

Examples

Wings Over the World User-Level Use Cases:

- *Book Flight*
- *Book Hotel*

Writing Rule

Name the Use Case with an active verb phrase that represents the goal of the primary actor

Use Case Brief

Definition

A simple, one-paragraph story describing the main success scenario for the use case

Example

UC: Book Flight

Actor: Agent

The agent retrieves a client's reservation and books the flight. The agent examines the aircraft seat map and selects the client's preferred seats in the aircraft. The agent enters the client's payment information, and the system books and assigns the seats. The system prints the tickets.

René Witte

Step-By-Step

Step 1: Identify and describe the Actors

Step 2: Identify the Use Cases and write a Brief Description

Step 3: Identify the Actor/Use Case Relationships

Step 4: Outline the Individual Use Cases

Step 5: Refine the Use Cases

René Witte
Concordia University

Identify the Actors and their Goals

Actors: What computers, subsystems and people will drive our system?

Goals: What does each actor need our system to do?

- Each goal shows up as a trigger to a usage (use case) of our system.

Result: A list of use cases, a sketch of the system.

- Short, fairly complete list of usable system function.

Wings Over the World Company: Actors and Goals

Actor	Goal
Agent	*Reserve Flight*
	Book Flight
	Cancel Flight Reservation
	Request Upgrade
	Open Passenger Profile
	Close Passenger Profile
Airline	*Cancel Flight*
	Discount Flight

René Witte

Wings Over the World Company: Actors and Goals (revised)

René Witte

Actor	Goal
Agent	*Reserve Flight*
	Book Flight
	Cancel Flight Reservation
	Request Upgrade
	~~*Open Passenger Profile*~~
	~~*Close Passenger Profile*~~
Airline	*Cancel Flight*
	Discount Flight

Actor-Goal List

René Witte

Original

Actor	Goal (Use Case)
User	Login
	Submit report
Administrator	Login
	Submit report
	Add *User*s
	...

Actor-Goal List

René Witte

Refined: User-Goal Level

Actor	Goal (Use Case)
User	Login
	Submit report
Administrator	Login
	Submit report
	Add/Manage *Users*
	...

The “Boss Test”

Applying the Test

Helps to determine if a Use Case is user-goal level or not:

- Your boss asks, *“What have you been doing all day?”*
- You reply: *“Logging in!”*

Is your boss happy?

Failure

If not, Use Case fails the Boss Test (does not deliver measurable value to user).

The EBP Test

Check if UC is an Elementary Business Process (EBP)

- "A task performed by one person in one place at one time, in response to a business event, which adds measurable business value and leaves the data in a consistent state, e.g., *Approve Credit or Price Order*."
- Emphasizes adding business value (similar to the "Boss Test")
- Don't confuse use cases with "functions" (like entries in a menu – add order, delete order, update order, ...)

The Size Test

Check if UC is Substantial Enough

- Typically, a UC has many steps
- In “fully dressed” format (see later), can be elaborated to 3–10 pages or more
- If you end up with only 1–2 steps, this is most likely not a valid use case

Exercises: Apply the Tests

René Witte

Test the following UCs

- Negotiate a strategic company alliance
- Handle returns
- Log in
- Update piece on game board

René Witte
Concordia University

Step-By-Step

Step 1: Identify and describe the Actors
Step 2: Identify the Use Cases and write a Brief Description
Step 3: Identify the Actor/Use Case Relationships
Step 4: Outline the Individual Use Cases
Step 5: Refine the Use Cases

Outline the Individual Use Cases

René Witte

User-Goal Level

- Expand a subset of Briefs and begin to fill in their details (a.k.a. Surface and Dive technique)
- Includes Step-by-step descriptions of the main success scenario

Navigating Goal Levels

Goal

“Place an order” (Customer)

↓ how ↑ why

Step

“Customer prepays for the order”

↓ how ↑ why

Substep

“. . . provides credit card number”

René Witte

For each user goal-level use case. . .

. . .write what the typical goal delivers

- Capture each actor's intent and responsibility, from trigger to goal delivery.
  - Say what information passes between them.
  - Number each line.
- The main success scenario, the "happy day" case.
  - Easiest to read and understand.
  - Everything else is a complication on this.
- Result: readable description of system's function.

René Witte

User Goal Main Success Scenario

René Witte

Example

UC: Book Flight

Level: User Goal

Main Success Scenario

1. This use case begins when a customer calls and requests a flight.
2. The customer describes her flight needs by specifying her origination, destination, travel dates and preferred departure times.
3. The system looks up all flights that match the customer's travel preferences and presents the travel options to the customer.
4. The customer selects a flight.
5. The system builds a flight ininerary for the customer.
6. The system reserves the flight for the customer.
7. The customer provides a credit card number and charges the price of the flight against it.
8. The system issues the ticket to the customer.

Writing Main Success Scenario: Example

René Witte
Concordia University

Dice Game [Lar05]

- A player plays the dice game.
- The dice game includes two dice.
- A player rolls two dice.
- If the total is seven, the player wins; otherwise the player loses.

René Witte
Concordia University

The Dice Game Main Success Scenario

- Write the Main Success Scenario for ‘Play Dice Game’, user goal level use case

Play Dice Game

Main Success Scenario

1. Player indicates to the system that s/he wishes to start a new game. [Trigger]
2. System indicates that the player can take a turn.
3. Player indicates that s/he wishes to take a turn.
4. System simulates rolling two dice and displays the face value.
5. System calculates the total and displays congratulations message.

René Witte

René Witte
Concordia University

How Do You Expect to Use the Dice Game ?

- "Happy end" scenario
  - **Dice Game:** Roll two dice.
  - **System:** CONGRATULATIONS! You won the game.

Failure Conditions

René Witte

Write failure conditions as alternative scenarios

- Usually, each step can fail.
- Note the failure condition separately, after the main success scenario.
- Result: list of alternative scenarios.

René Witte
Concordia University

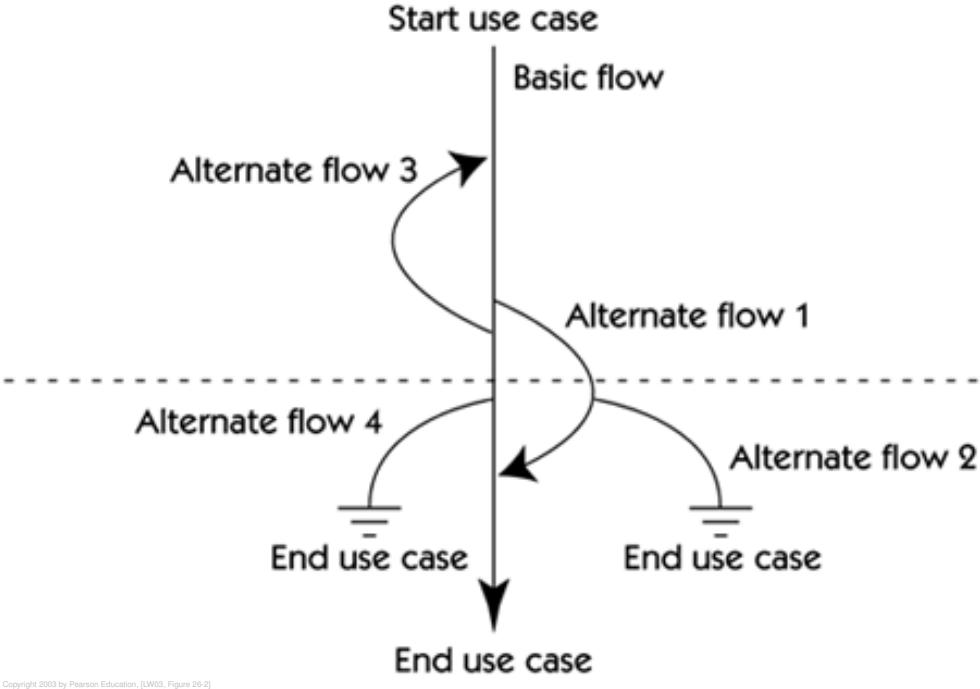
How Do You Expect to Use the Dice Game (II)?

- Not so “happy end” scenario
  - **Dice Game:** Roll two dice.
  - **System:** Loser, try again . . .

Exercise

- write the Alternative Scenario for Play Dice Game user goal level use case

One Use Case, Multiple Scenarios



René Witte

More on Scenarios

Structured narrative keeps the context visible, the value to the user clear.

Compare these specifications:

Unstructured paragraphs:

- "The order entry system has an interface to system EBMS and to a terminal.
- It computes (When? How?) and displays (When? How?) the sum of the order items' cost.
- . . ."

With structured narrative:

- "The orderer (EBMS or an entry person) identifies the name of the customer & the items on the order.
- The system displays the cost of the total order.
- If the items are in stock and the client has sufficient credit, . . ."

More on Scenarios (II)

René Witte

Some Use Case Writing Rules

- Each scenario begins with a triggering action that the system is able to detect:
  - ATM Withdraw Cash Main Success Scenario
    1. Customer inserts the card.
    2. . . .
- Steps should not be too small
  - Stepping up to the sidewalk Main Success scenario
    1. She lifts her foot from the street to the curb.
    2. She arcs the foot through the air, and lowers to the pavement
    3. . . .
  - Keep scenarios to three to nine steps, all at a level of abstraction just below the use case goal

Essential vs. Concrete Style

Essential

Focus on intent.

- Avoid making UI decisions

Concrete

UI decisions are embedded in the use case text, e.g.:

- “Admin enters ID and password in the dialog box (see picture X)”

Concrete style not suitable during early requirements analysis work!

René Witte
Concordia University

Use Case Diagram

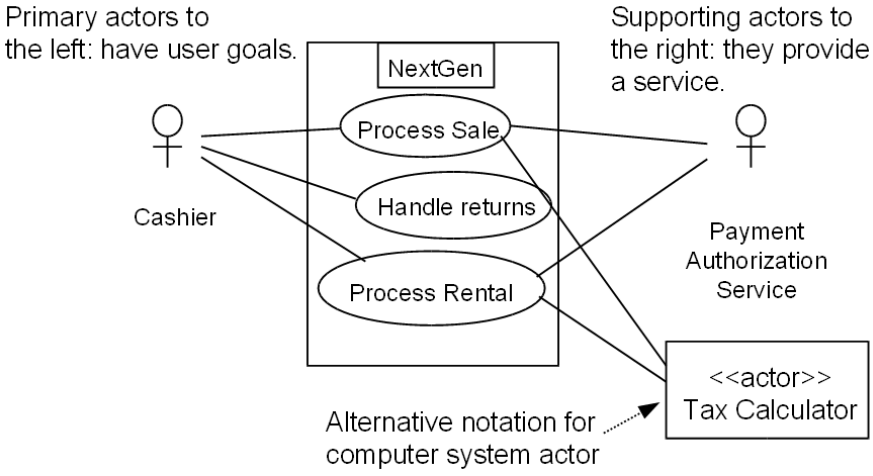
- The UML defines the Use Case Diagram
- It does not explain how to write the actual Use Cases!
- This is actually the least important aspect of Use Cases modeling!

Use Cases vs. UC Diagrams

Use cases are text documents, not diagrams, and use-case modeling is primarily an act of writing text, not drawing diagrams.

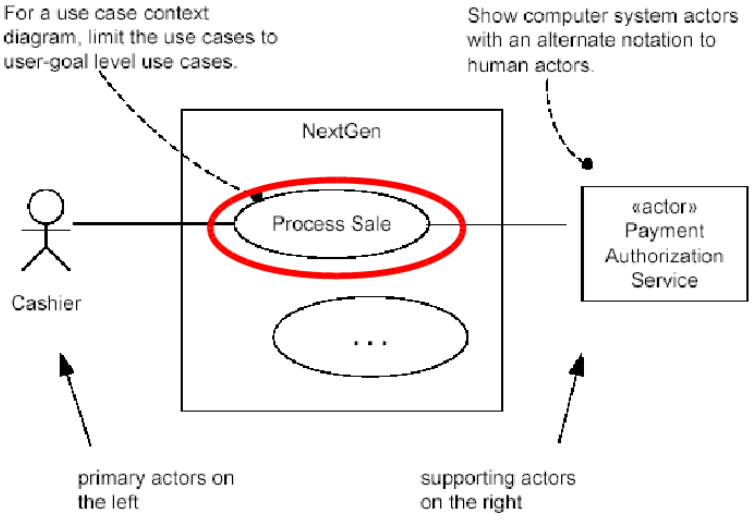
Use Case Diagrams

René Witte

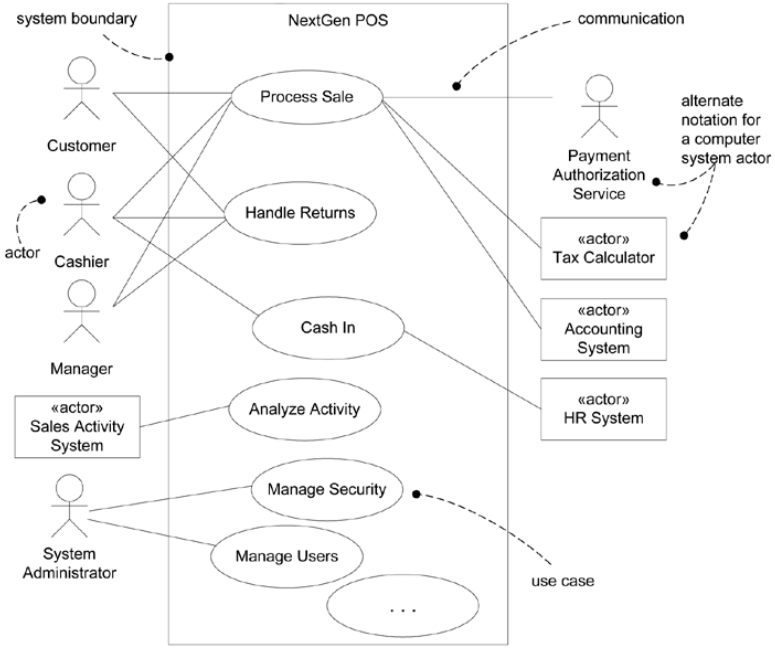


Use Case Diagram as a Context Diagram: POS

René Witte
Concordia University



Use Case Context Diagram



René Witte

René Witte

Step-By-Step

Step 1: Identify and describe the Actors

Step 2: Identify the Use Cases and write a Brief Description

Step 3: Identify the Actor/Use Case Relationships

Step 4: Outline the Individual Use Cases

Step 5: Refine the Use Cases

Relating Use Cases

The include relationship

- partial behavior that is common across several use cases (models required behavior)
- separate it into its own Subfunction Use Case, and indicate its inclusion

The extend relationship

- separate it into its own Subfunction Use Case
- indicate where and under what condition it extends the behavior of some base use case (models optional behavior)

«include» Relationship: POS

UC1: Process Sale
Level: User Goal

Main Success Scenario:

1. Customer arrives at a POS checkout with goods and/or services to purchase.

...

7. Customer pays and System handles payment.

...

Extensions:

7a. Paying by credit: *Handle Credit Payment*.

...

UC2: Handle Credit Payment
Level: Subfunction

Main Success Scenario:

1. Customer enters their credit account information.
2. System sends payment authorization request to an external Payment Authorization Service System, and requests payment approval.
3. System receives payment approval and signals approval to Cashier.
4. ...

Extensions:

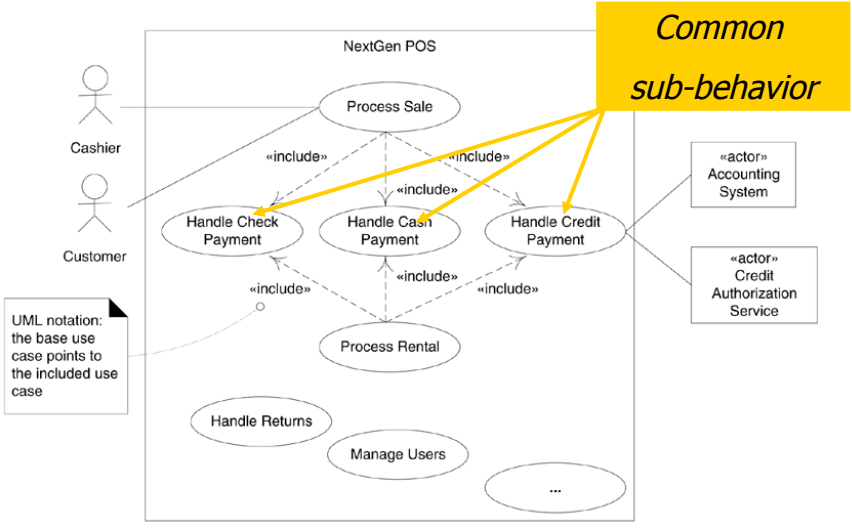
2a. System detects failure to collaborate with external system:
- System signals error to Cashier.
- Cashier asks Customer for alternate payment.

...

René Witte
Concordia University

UML: Using «include» Relationship

René Witte



«extend» Relationship: POS

René Witte
Concordia University

UC1: Process Sale
Level: User Goal

Main Success Scenario:

1. Customer arrives at a POS checkout with goods and/or services to purchase.

...

7. Customer pays and System handles payment.
   (gift certificate)

...

Extensions:

7a. Paying by credit: *Handle Credit Payment*.

7b. Paying by gift certificate: *Handle Gift Certificate Payment*.

...

UC3: Handle Gift Certificate Payment
(the extending use case)
Level: Subfunction

Trigger: Customer wants to pay with gift certificate.

Extension Points: Payment in Process Sale.

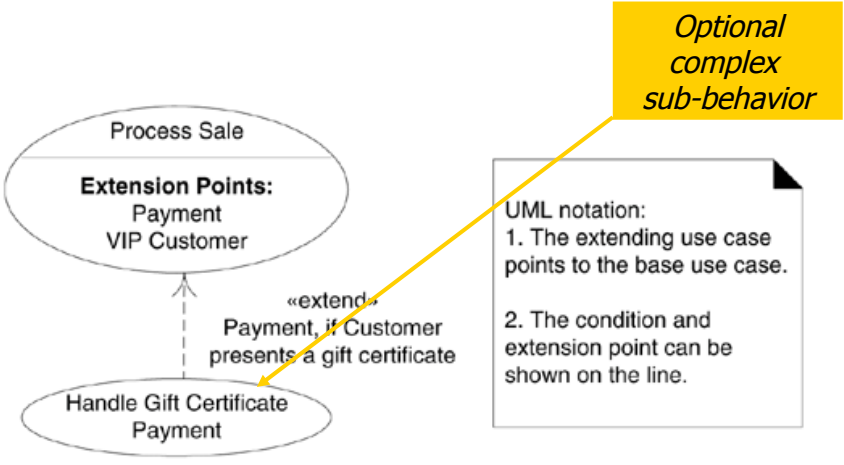
Gift Certificate Main Scenario:

1. Customer gives gift certificate to Cashier.
2. Cashier enters gift certificate ID
   ...

...

UML: Using Extend Relationship (with extension points)

René Witte



Include vs. Extend

Another way to look at include vs. extend:

«include»

- included sub-function shared across several use-cases (duplicate parts!)
- i.e., it was taken out of a use case, once the requirements engineer(s) detected duplicated parts

«extend»

- add new functionality to an existing use case
- original use case remains untouched
- extend sub-function typically specific to a single use-case
- e.g., done in later iterations to add functionality
- extended functionality may be purely optional

Outline

René Witte

Concordia University

Fully-Dressed Use Cases

René Witte
Concordia University

Overview

- Further refinement of a use case
- Usually done in later iterations
  - Pre- and post-conditions
  - Special requirements (non-functional)
  - Success End Condition
  - Failure End Condition
  - Minimal Guarantee
  - Technology and Data Variations
- Details depend on concrete template

Fully-Dressed Use Case Example I

René Witte

Process Sale

Use case UC1: Process Sale

Primary Actor: Cashier

Stakeholders and Interests:

- Cashier: Wants accurate and fast entry, no payment errors, . . .
- Salesperson: Wants sales commissions updated.
- . . .

Preconditions: Cashier is identified and authenticated.

Success Guarantee (Postconditions):

- Sale is saved. Tax correctly calculated.
- . . .

Main success scenario (or basic flow): [see next slide]

Extensions (or alternative flows): [see next slide]

Special requirements: Touch screen UI, . . .

Technology and Data Variations List:

- Identifier entered by bar code scanner, . . .

Open issues: What are the tax law variations? . . .

Fully-Dressed Use Case Example II

René Witte
Concordia University

Process Sale

Main success scenario (or basic flow):

1. The Customer arrives at a POS checkout with items to purchase.
2. The cashier records the identifier for each item. If there is more than one of the same item, the Cashier can enter the quantity as well.
3. The system determines the item price and adds the item information to the running sales transaction. The description and the price of the current item are presented.
4. On completion of item entry, the Cashier indicates to the POS system that item entry is complete.
5. The System calculates and presents the sale total.
6. The Cashier tells the customer the total.
7. The Customer gives a cash payment (“cash tendered”) possibly greater than the sale total.

Extensions (or alternative flows):

2a. If invalid identifier entered. Indicate error.
7a. If customer didn’t have enough cash, cancel sales transaction.

Pre- and Postconditions

René Witte
Concordia University

Overview

- Preconditions are assumptions about the state of the system before execution of the operation.
- A postcondition is an assumption that refers to the state of the system after completion of the operation.
  - The postconditions are not actions to be performed during the operation.
  - Describe changes in the state of the objects in the Domain Model (instances created, associations are being formed or broken, and attributes are changed)
  - Write in past tense: “Inventory was updated”

Process Sale: User Level Use Case Example

René Witte
Concordia University

Pre- and Postconditions

Preconditions:

Cashier is identified and authenticated.

Success Guarantee (Postconditions):

Sale is saved.
Tax is correctly calculated.
Accounting and Inventory are updated.
Commissions recorded.
Receipt is generated.
Payment authorization approvals are recorded.

Fully Dressed Use Cases

Template

Use Case Section	Comment
Use Case Name	Start with a verb.
Scope	The system under design.
Level	“User-level” or “subfunction”
Primary Actor	Calls on the system to deliver its services.
Stakeholders and Interests	Who cares about this use case, and what do they want?
Preconditions	What must be true on start, and worth telling the reader?
Success Guarantee	What must be true on successful completion, and worth telling reader?
Main Success Scenario	A typical, unconditional happy path scenario of success.
Extensions	Alternate scenarios of success or failure.
Special Requirements	Related NFRs.
Technology and Data Variations List	Varying I/O methods and data formats.
Frequency of Occurrence	Influences investigation, testing, and timing of implementation.
Miscellaneous	Such as open issues.

René Witte

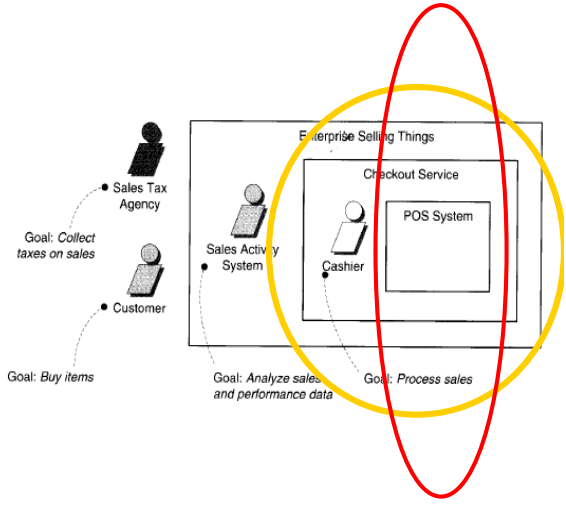
Scope of Use Case

Is the Cashier or Customer THE Primary Actor?

René Witte
Concordia University

Depends on the scope: boundary of the system under development

- **Case 1:** the checkout service is viewed as an aggregate system => **Customer is a primary actor**
- **Case 2:** the boundary of the system is POS and the system services the goal of a trained cashier to process the customer's sale => **Cashier is the primary actor**



When is Your Use Case Model Complete?

Ask the following questions:

- Have you identified and documented all actors and goals?
- Has the customer, or someone representing the customer, acknowledged that the use case set is complete, and that each use case is readable and correct?

More Use Case Writing Rules

René Witte
Concordia University

Use cases do not collect formulas, state, cardinality

Bad Examples:

1. Order sum = order item costs * 1.06 tax. (design issue)
2. Promotions may not run longer than 6 months. (business rule)
3. Customers only become ‘Preferred’ after an initial 6 month period. (business rule)

Write in an UI-free style

Bad Examples:

1. Systems displays the “edit users” window (see Figure 1). (design issue)

Focus on intent

Bad Examples:

1. Administrator enters ID and password in dialog box.
   (Better: 1. Administrator identifies self.)

Use Case Model Quality Checklist

1. **Actors**
   - 1.1. Are there any actors that are not defined in the use case model, that is, will the system communicate with any other systems, hardware or human users that have not been described?
   - 1.2. Are there any superfluous actors in the use case model, that is, human users or other systems that will not provide input to or receive output from the system?
   - 1.3. Are all the actors clearly described, and do you agree with the descriptions?
   - 1.4. Is it clear which actors are involved in which use cases, and can this be clearly seen from the use case diagram and textual descriptions? Are all the actors connected to the right use cases?
2. **The use cases**
   - 2.1. Is there any missing functionality, that is, do the actors have goals that must be fulfilled, but that have not been described in use cases?
   - 2.2. Are there any superfluous use cases, that is, use cases that are outside the boundary of the system, do not lead to the fulfilment of a goal for an actor or duplicate functionality described in other use cases?
   - 2.3. Do all the use cases lead to the fulfilment of exactly one goal for an actor, and is it clear from the use case name what is the goal?
   - 2.4. Are the descriptions of how the actor interacts with the system in the use cases consistent with the description of the actor?
   - 2.5. Is it clear from the descriptions of the use cases how the goals are reached and do you agree with the descriptions?
3. **The description of each use case**
   - 3.1. Is expected input and output correctly defined in each use case; is the output from the system defined for every input from the actor, both for normal flow of events and variations?
   - 3.2. Does each event in the normal flow of events relate to the goal of its use case?
   - 3.3. Is the flow of events described with concrete terms and measurable concepts and is it described at a suitable level of detail without details that restrict the user interface or the design of the system?
   - 3.4. Are there any variants to the normal flow of events that have not been identified in the use cases, that is, are there any missing variations?
   - 3.5. Are the triggers, starting conditions, for each use case described at the correct level of detail?
   - 3.6. Are the pre- and post-conditions correctly described for all use cases, that is, are they described with the correct level of detail, do the pre- and post conditions match for each of the use cases and are they testable?
4. **Relation between the use cases:**
   - 4.1. Do the use case diagram and the textual descriptions match?
   - 4.2. Has the include-relation been used to factor out common behaviour?
   - 4.3. Does the behaviour of a use case conflict with the behaviour of other use cases?
   - 4.4. Are all the use cases described at the same level of detail?

René Witte

René Witte
Concordia University

Writing good use cases is a creative task

- You are describing a systems that does not yet exist!
- This requires vision and creativity
- Don't start to analyze structure or implementation
- Synthesize instead of Analyze!
- Take time to understand the domain
- Understand the software development involved
- Assign people to writing Use Cases with good technical writing skills

René Witte
Concordia University

Overview

- Move from Problem Domain to Solution Domain
- Focus on Understanding a set of features that address the stakeholders' needs
- Very challenging activity
- Requires collaboration of people with different backgrounds:
  - **User with application domain knowledge**
  - **Developer with implementation domain knowledge**
- Bridges the gap between user and developer

Outline

René Witte

Misuse Cases

Also known as "Use Cases with Hostile Intent"

- Risk Analysis
- NFR Analysis
- Following slides by Ian Alexander [Ale03]
  http://www-dse.doc.ic.ac.uk/Events/BCS-RESG/Aleksander.pdf

What Happens if You Always Look on the Positive Side?

- At least you relax and are happy
- But you aren't ready for problems when they come up
  - In **business**, there are people who want you to fail
  - On **projects**, there are many types of cockup
  - In **systems**, there are threats and hazards all round
  - In **software**, there's a bug in every module

"If anything can go wrong, it will"

(The REAL Captain Murphy, USAAF)

So, It Pays to Think Out 'Negative Scenarios'

#1 Either you think out what could happen
– and what you mean to do about it

#2 Or you wait until it happens
– and you find out whether it's too late to do anything about it.

Here are some techniques for approach #1.

Understanding Negative Scenarios

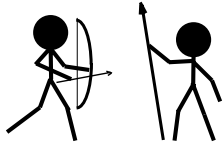
- A **Scenario*** is a sequence of actions leading to a Goal desired by a person or organisation

- A **Negative Scenario** is a scenario whose Goal is
  - desired Not to occur by the organisation in question
  - desired by a hostile agent (not necessarily human)

* This is a different usage from '*four possible future business scenarios*'

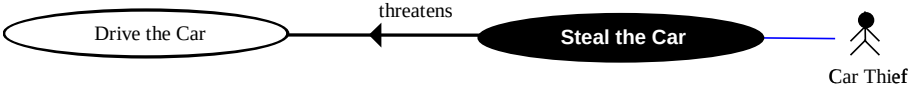
Negative Scenarios are Not New

Montignac Caves, Dordogne, France



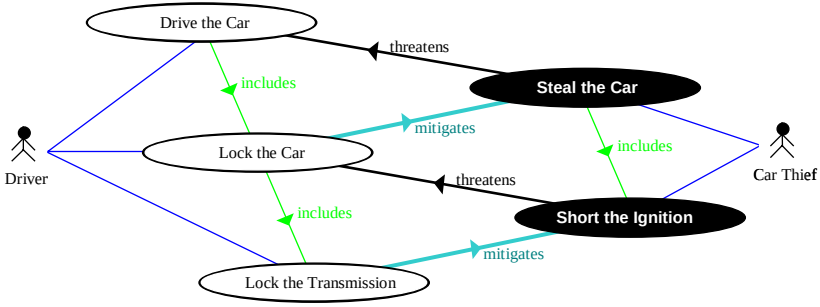
'Suppose it turns and charges us before it falls into the pit'

Misuse Cases



- Guttorm Sindre and Andreas Opdahl, 2000
- Actor is a Hostile Agent
- Bubble is drawn in inverted colours
- Goal is a Threat to 'Our System'
- Obvious Security Applications

A MiniMax Approach to Security

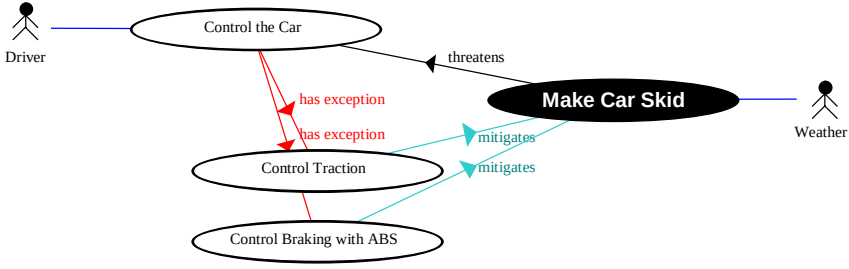


Use Cases for 'Car Security'

- White's Best Move … is to find out Black's Best Move, and counter it
- Seems natural to me to introduce 'threatens', 'mitigates'
- Economical use of types of relationship (UML stereotypes)

René Witte
Concordia University

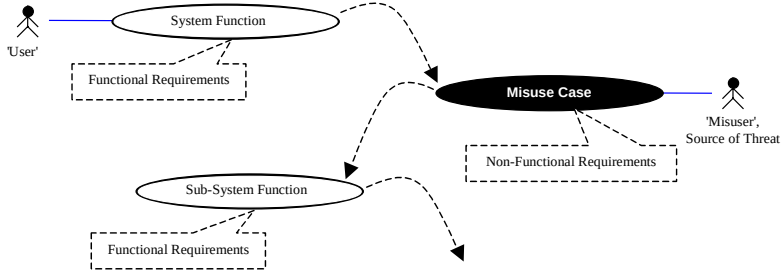
Anthropomorphize … for Safety



- UML's stick-man looks like 'human agent' but can be of any type (robot, system)
- Anthropomorphizing Forces of Nature is useful: it enables us to use our Social/Soap Opera Brain to reason about threats to our systems
- Misuse Case helps to Elicit Subsystem Functions

René Witte
Concordia University

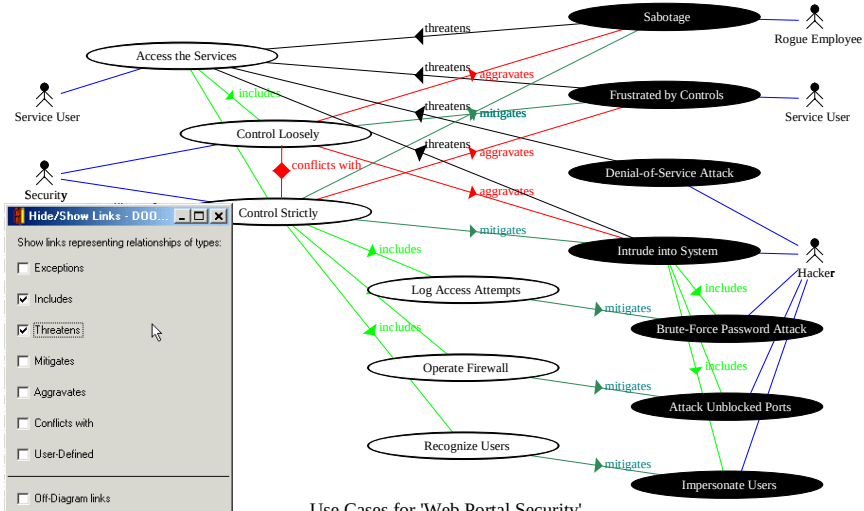
Misuse Cases Identify NFRs



Interplay of Use & Misuse Cases with Functional & Non-Functional Requirements

- Use Cases are weak on NFRs
- Misuse Cases naturally focus on NFRs, e.g. Safety
- Response is often a SubSystem Function, possibly to handle an Exception

Design Trade-Offs



Use Cases for 'Web Portal Security'

Conflict Analysis builds upon Use/Misuse Case Modelling with additional relationships *'aggravates' and 'conflicts with'*

René Witte

Benefits of Misuse Cases

- Open a new avenue of exploration
- Contribute to searching systematically for exceptions, directed by the structure of the scenarios
- Offer immediate justification for the search and indicate the priority of the requirements discovered
- By personifying and anthropomorphizing the threats, add the force of metaphor to requirements elicitation
- Make the search enjoyable and provide an algorithm for the search. Obvious parallel here with Cost/Benefit analysis
- Make the reasoning behind affected requirements immediately comprehensible

Applications of Misuse Cases

- Eliciting **Security** Requirements (see Sindre & Opdahl)
- Eliciting **Safety** Requirements (see Allenby & Kelly)
- *and possibly* **other kinds of NFRs** (?)
- Identifying **Exceptions** (not unlike Anton & Potts)
- Identifying **Test Cases** (seems obvious, future scope)
- Design **Trade-offs** (see Alexander)

Summary

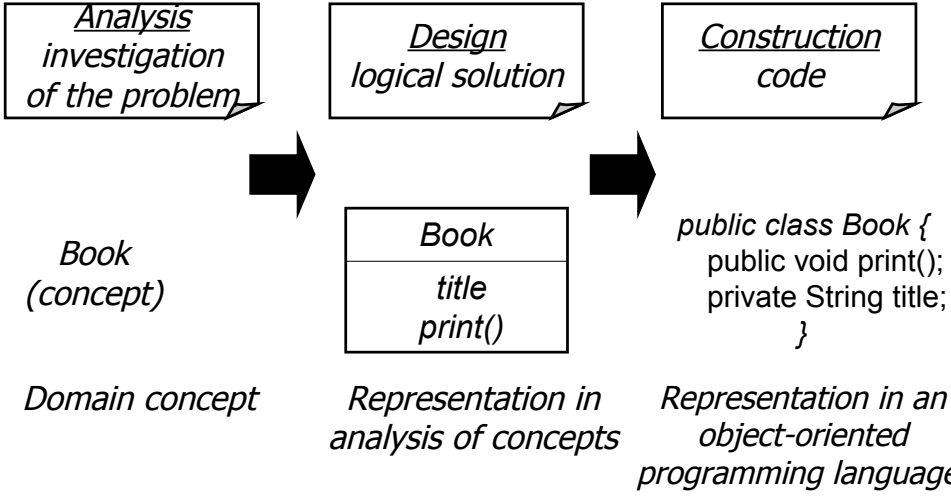
- Misuse Cases are a new form of an old technique
- Planning has always been 'what-if'
- Systems that don't plan to handle Exceptions are planning to fail at once
- Misuse Cases greatly enhance the power of Use Cases
- The power of visual metaphor (black) and anthropomorphism (stick-men) should not be underestimated
- Misuse Cases are useful throughout System Life-Cycle

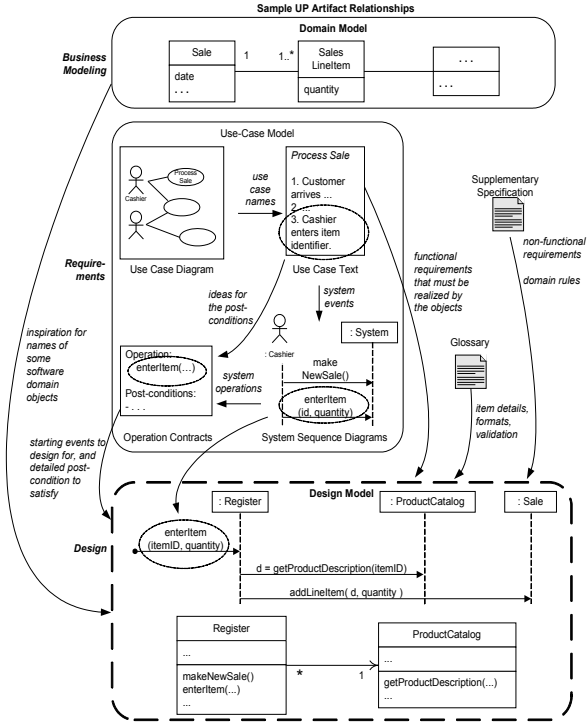
Outline

Connection with Design

From Requirements to Implementation

René Witte
Concordia University





René Witte

Use Case Realizations

Overview

- (R)UP: *"A use-case realization describes how a particular use case is realized within the Design Model, in terms of collaborating objects."*
- Connection between:
  - Requirements expressed as Use Cases
  - Design expressed as Use Case Realizations

Use Case Realization: Example

René Witte

System
makeNewSale() *addLineItem(id, quantity)* *endSale()* *makePayment(cashTendered)*

Contract CO1: ***makeNewSale***
Operation: *makeNewSale ()*
Cross References: *Use Case Process Sale.*
Preconditions: *none.*
Postconditions:

- *A Sale instance s was created. (instance creation)*
- *s was associated with Register (association formed)*
- *Attributes of s were initialized*

- A use-case realization describes how a use case is realized in terms of collaborating objects.
- UML interaction diagrams are used to illustrate use case realizations.
- Recall *Process Sale*: from the main scenario we identified a number of system events (operations)

Use Cases in Interative Process Models

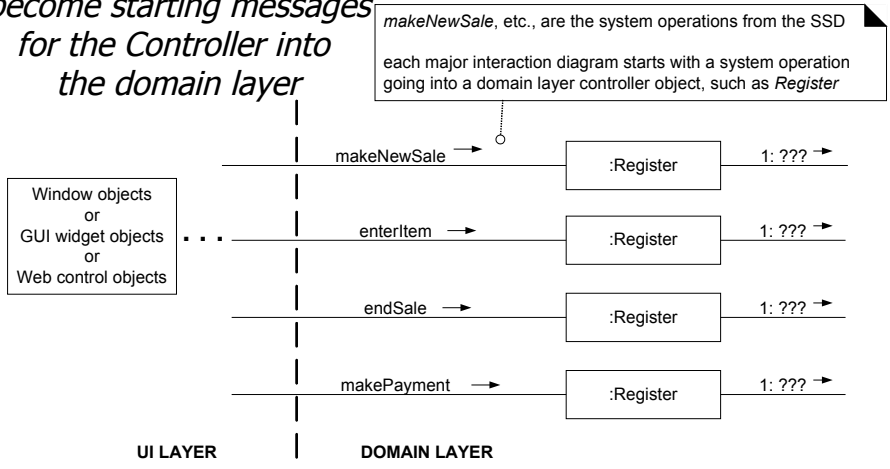
Incremental development for the same use case across iterations

- Not all requirements in a use case (e.g., “Process Sale”) must be handled in one iteration.
- It is common to work on varying scenarios or features of the same use case over several iterations and gradually extend the system to ultimately handle all the functionality required.
- On the other hand, short, simple use cases may be completed within one iteration.

René Witte

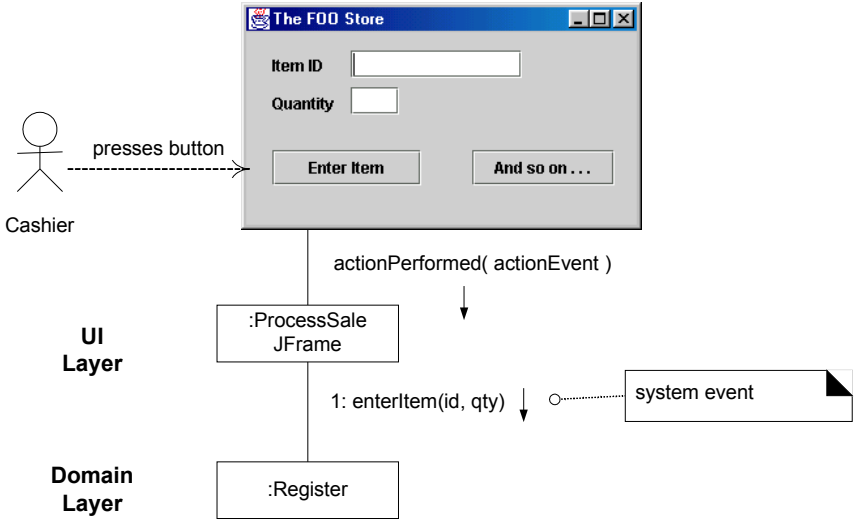
Concordia University

System Operations in SSD become starting messages for the Controller into the domain layer



Connecting UI and Domain Layer

UI Layer vs. Domain Layer

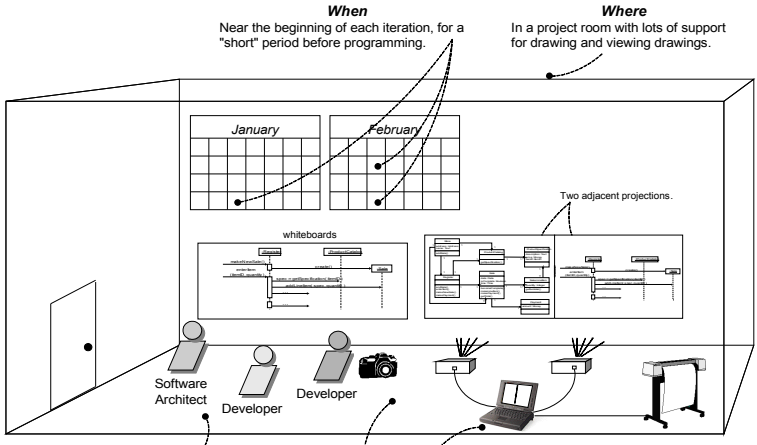


René Witte

Design in UP

When, Where, Who, and How

René Witte
Concordia University



Outline

René Witte

Reading Material

René Witte

Required

- [LW03, Chapters 14, 21] (use case modeling)
- [LW03, Appendix C] (use case template)

Supplemental

- [Lar05, Chapters 6, 30] (use case examples)
- [Ale03] (misuse cases)

Further Reading

- [Coc01] (use case reference)
- [Poh10, Part III.b, Chapters 9–12] (more on scenarios and (mis-)use cases)

References I

[Ale03] Ian Alexander.
Misuse Cases: Use Cases with Hostile Intent.
IEEE Software, 20(1):58–66, Jan/Feb 2003.

[Coc01] Alistair Cockburn.
Writing Effective Use Cases.
Addison-Wesley, 2001.

[Lar05] Craig Larman.
Applying UML and Patterns.
Prentice Hall, third edition, 2005.

[LW03] Dean Leffingwell and Don Widrig.
Managing Software Requirements: A Use Case Approach.
Addison-Wesley, 2003.
Available online at http://clues.concordia.ca/record=b2529323.

[Poh10] Klaus Pohl.
Requirements Engineering: Fundamentals, Principles, and Techniques.
Springer-Verlag Berlin Heidelberg, 2010.

References II

[vL09] Axel van Lamsweerde.
Requirements Engineering: From System Goals to UML Models to Software Specifications.
John Wiley & Sons, 2009.