

CSC 436, Spring 2017

Use Cases

Ravi Sethi



- **How would you describe the behavior of your system to a non-technical person?**
- **How would you slice your system for iterative development?**

Goals for a description

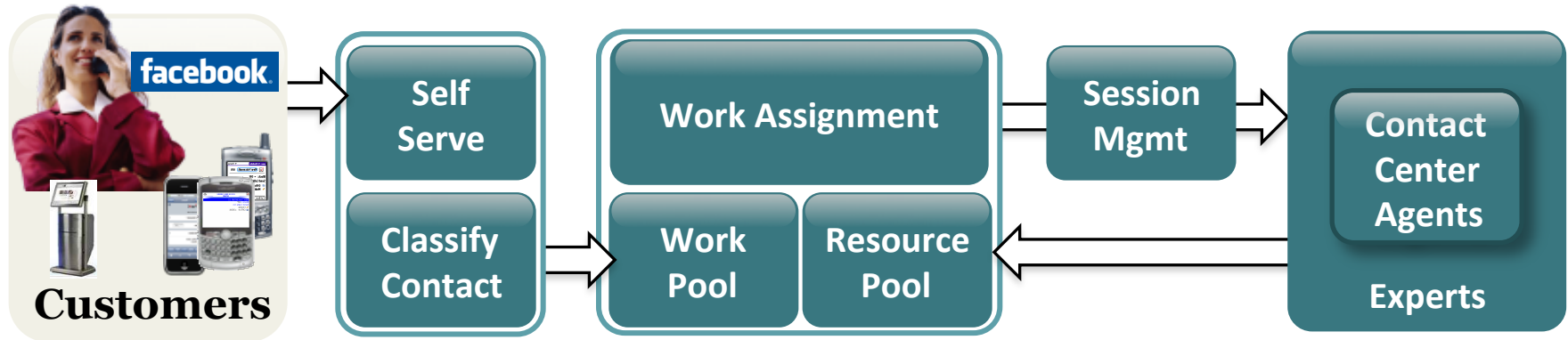
How to describe the behavior of a system?

- **A single description for all stakeholders**
 - Readable enough for customers, users, marketers
 - Specific enough for developers
- **Trace system behavior to user goals**
 - Relate requirements to user goals
 - Relate implementation to requirements and hence to user goals
- **A complete description**
 - Provide an overview of the system behaviors (note plural)
 - Informal notion of completeness

Example: Match Callers with Agents

More generally, match work with resources

- Informal description of a contact center



- Basis for the description:
the flow of actions in response to a customer call

Use Cases

“Between 1987 and 1992, [use cases] evolved and matured, as about twenty customers put [them] to practical use for ... many different kinds of systems: management information; defense (pilot, counter measure, C3I); and telecom (POTS, mobile).”

–Jacobson [2004]

A Use Case Tells a Story

Use Cases Capture Functional Requirements

- **A use case is**

- a sequence of actions that a system performs
- that yields an observable result of value
- to a particular actor

← **Flow**
← **Goal**
← **Actor**

- **Created by Ivar Jacobson in 1986**

- Motivation: model telephone calls

- **Use cases have become part of UML**

- UML (Unified Modeling Language) is not known for simplicity
- “Note, however, that although the UML effort resulted in a much more precise definition of use cases, it didn't do much to evolve them.”

- **Same person can have multiple roles; e.g.,**
 - Manager of a team
 - Employee when it comes to the person's own benefits
- **An actor is a role played by a person or thing; e.g.,**
 - Two actors, same person: manager and employee

Flow: sequence of actions by the system

An Informal Telephony Example

- **Goal**
 - A caller places a telephone call to a callee
- **General Idea (we will get more specific)**
 - Collect the number entered by the caller.
 - Map the number to a destination.
 - Route the call through the telephony infrastructure.
 - Ring the callee's phone.
 - Log the duration of the call, once the callee answers.
 - Release the resources for the call, when either party disconnects.

Flow: Sequence of Actions by the System

Start each action with “The actor ...” or “The system ...”

1. The caller enters the number to be called
2. The system collects the number
3. The system maps the number to a destination
4. The system routes the call through the telephony infrastructure
5. The system rings the callee’s phone
6. The callee answers
7. The system starts logging the duration of the call
8. The caller disconnects
9. The system records the duration of the call
10. The system releases the infrastructure resources for the call

Flow: Sequence of Actions by the System

Flows are the most important part of a use-case description

- **Basic Flow**

- The normal expected sequence of events for the use case
- The sequence that successfully enables the actor to achieve the goal

- **Alternative Flows**

- Alternative Flows capture behaviors that may be essential for success, but are not central to an overall understanding of the system
 - Exceptions & Error Conditions are the most common Alternative Flows; in practice, 60-80% of the text may be devoted to them
 - Other Alternative Flows are for Variations and Optional Behaviors
- For simplicity, keep Alternative Flows independent of the Basic Flow

Basic Flow: Cash Withdrawal from an ATM

Classic Example

1. The customer inserts a card.
2. The system reads the card.
3. The system authenticates the cardholder.
4. The system displays options for bank services.
5. The customer selects Withdraw Standard Amount.
6. The system checks the account for availability of funds.
7. The system dispenses cash and updates the account balance.
8. The system returns the card.

Alternative Flow 1

Classic Example: Customer Withdrawing Cash

• Alternative Flow 1: Authentication Failed

1. The customer inserts a card.
2. The system reads the card.
3. The system authenticates the cardholder.
- 4'. Authentication fails.
8. The system returns the card.

Extension Points: Used to connect flows

Extension points are named points between steps in a flow

- **Withdrawal basic flow with 4 extension points**

1. The customer inserts a card.

{ Read Card }

2. The system reads the card.

3. The system authenticates the cardholder.

{ Bank Services }

4. The system displays options for bank services.

5. The customer selects Withdraw Standard Amount.

6. The system checks the account for availability of funds.

{ Dispense Cash }

7. The system dispenses cash and updates the account balance.

{ Return Card }

8. The system returns the card

Extension Points: Used to connect flows

Extension points are named points between steps in a flow

- **Alternative Flow 1: Authentication Failed**

At { **Bank Services** } if authentication has failed

The system displays “Authentication Required”

Resume the basic flow at { **Return Card** }

- **Alternative Flow 2: Insufficient Funds**

At { **Dispense Cash** } if the account has insufficient funds

The system displays “Insufficient Funds in Account”

Resume the basic flow at { **Return Card** }

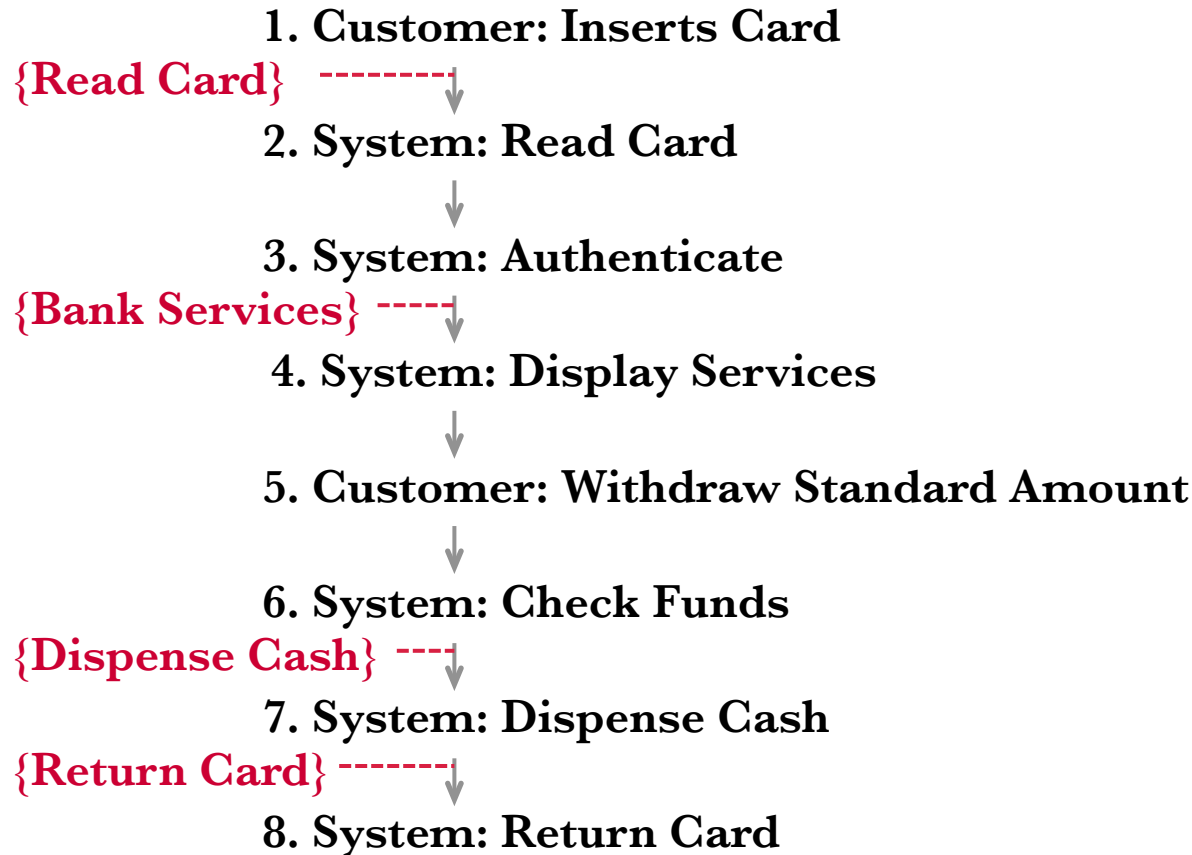
Kinds of Alternative Flows

Specific and Bounded Alternative Flows

- **Specific Alternative Flows**
 - Attach between two named extension points
- **Bounded Alternative Flows**
 - Attach **anywhere** between two named extension points

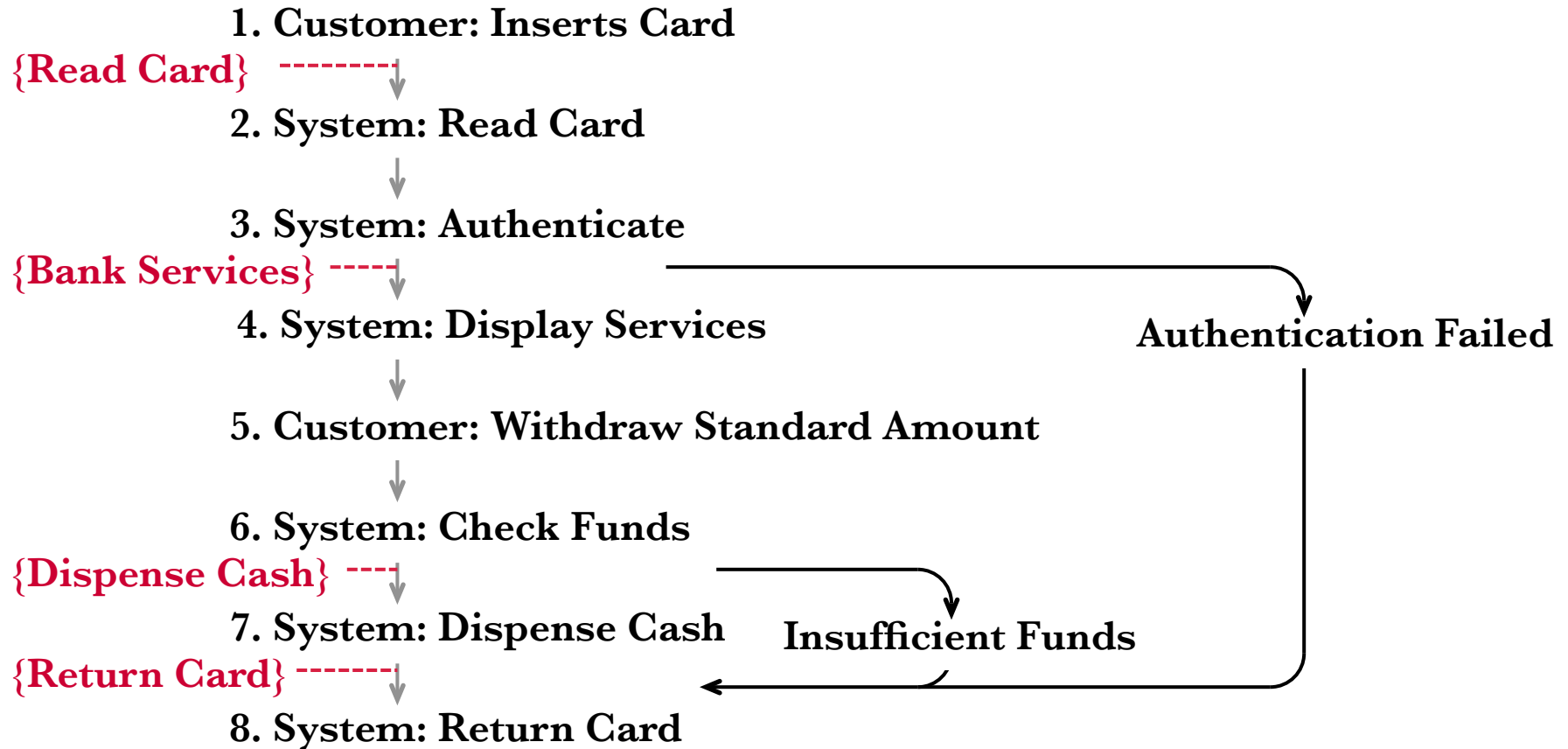
Basic Flow: Cash Withdrawal from an ATM

Graphical View



Specific Alternative Flows

Attach between two named extension points



Bounded Alternative Flows

ATM Example: Note “At any point between ...”

- **Bounded Alternative Flow: Loss of Network Connection**

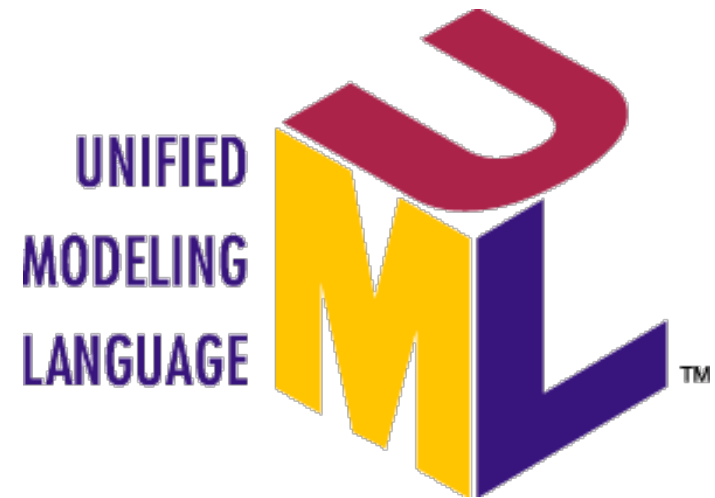
At any point between { **Read Card** } and { **Dispense Cash** }
if the network connection is lost

Display “Sorry, Out of Service”

Resume the basic flow at { **Return Card** }

Use Case Diagrams

*Class Diagrams and
Use Case Diagrams
are part of UML*

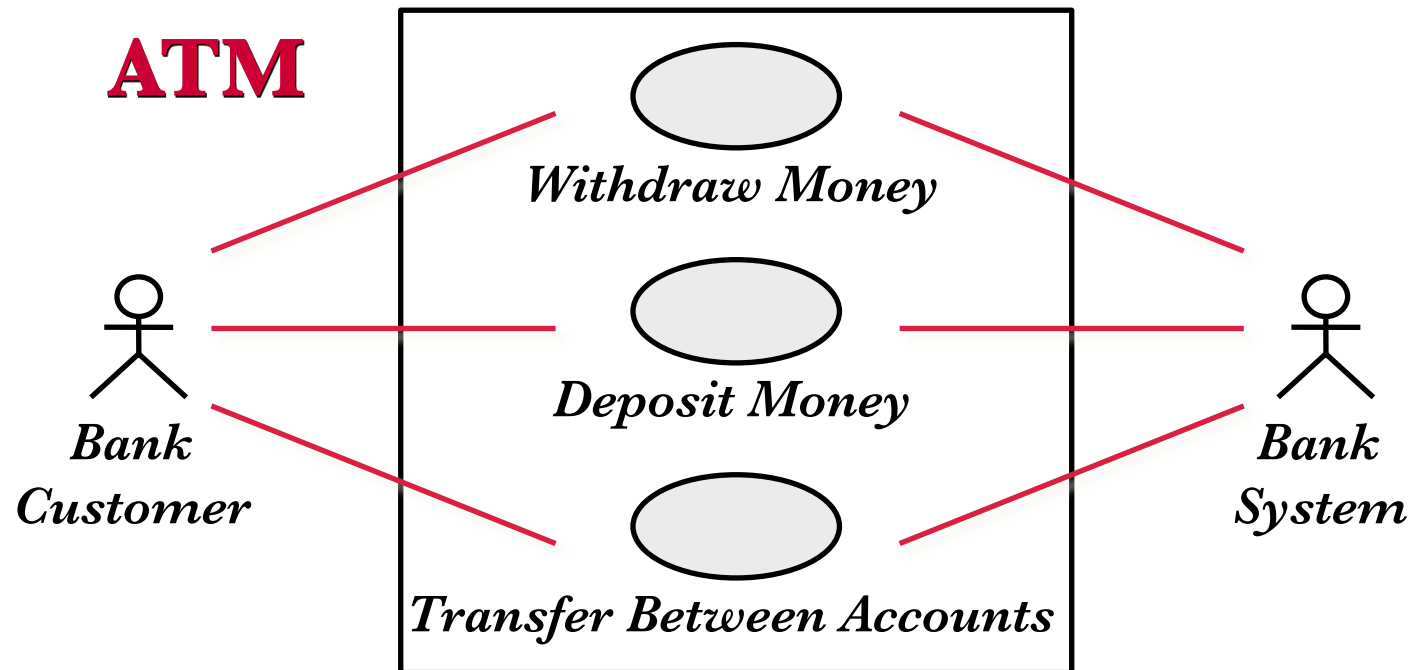


Use Case Diagram

Shows all actors, use cases, and associations

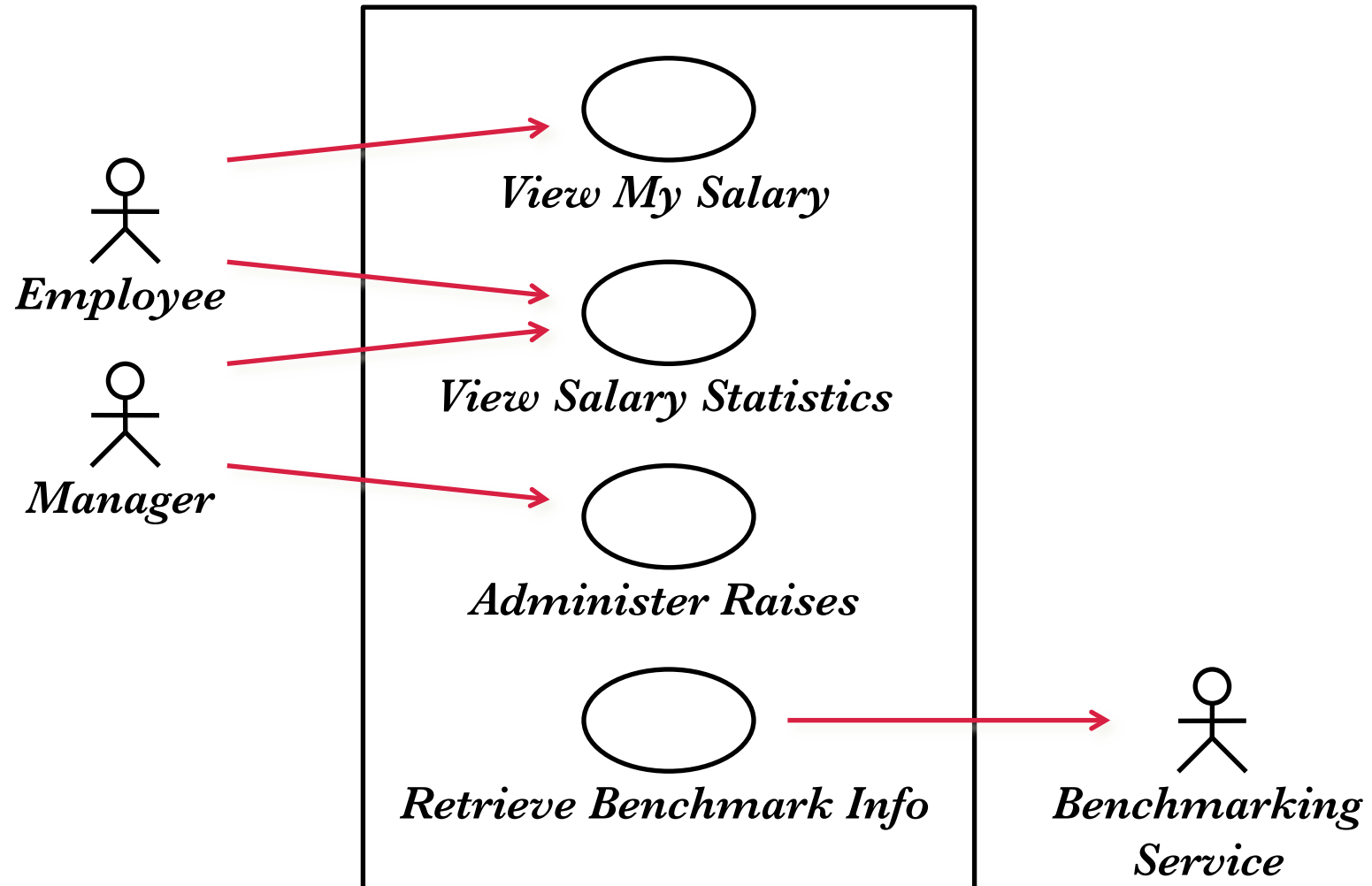
- **Big picture of the system**

- Use cases are represented by ovals; actors by stick figures
- Associations between actors and use cases, are represented by lines
- Arrows, if used, are from the initiator of an action



Use Case Diagram

Salary Administration Use Cases



Use Case Diagrams in Practice

Empirical Study of UML Usage (UML emphasizes diagrams)

- **Based on interviews with 50 professionals**
 - 70%: No use of UML
 - 22%: Selective Use
 - 0%: Wholehearted Use
- **Informal use of Use Cases**
 - “It’s hard to design without something that you could describe as a use case.”
 - “Many described use cases informally, for example, as: ‘Structure plus pithy bits of text to describe a functional requirement. Used to communicate with stakeholders.’ ”

Slices for Iterative Development

Guidelines for Iterative Development

Jacobson et al. lay out 6 “Basic Principles”

4. Build the system in slices

- Flows “slice” through Design, Implementation, Test, Documentation
- Use-case driven development

5. Deliver the system in increments

- Use cases themselves can be implemented in slices/increments

6. Adapt to meet the team’s needs

- Fit the development process to the team and the environment
- Lightweight for small collaborative teams
- Managed/Documented for large distributed teams

Use Case Slices

- **A slice is a set of flows or part of a flow**
 - Flows are from the start to the end of the use case
 - A slice is work that can be done in an iteration
 - If a flow is too big for an iteration, select a part and “stub” the rest
- **Iterative Planning**
 - Start with the basic flow of the most important use case
 - For each iteration, select the most important remaining flows

Writing Use Cases

Jacobson et al. lay out 6 “Basic Principles”

1. Keep it simple by telling stories

- A story, together with test cases that define successful implementation

2. Understand the big picture

- “Without an understanding of the system as a whole you will find it impossible to make the correct decisions about what to include ..., what to leave out, what it will cost, and what benefit it will provide.”
- Use-case diagrams provide an overview of the system by depicting collections of use cases

3. Focus on value

- How will it be used to achieve a specific goal for a particular user
- Start with basic flows, then capture alternative flows

Two columns, one for user actions, one for system response

Actor Action

- 1. Insert Card**
- 2. Enter Passcode**
- 3.**
- 4. Select Withdraw Cash**
- 5. Select Amount**

System Response

- Prompt for Authentication**
- Authenticate Cardholder**
- Display Bank Services**
- Offer Standard Amounts**
- Query Account for Sufficient Funds**

Life Cycle of a Use Case

*Right level of detail depends on the audience:
e.g., users and testers have different needs*

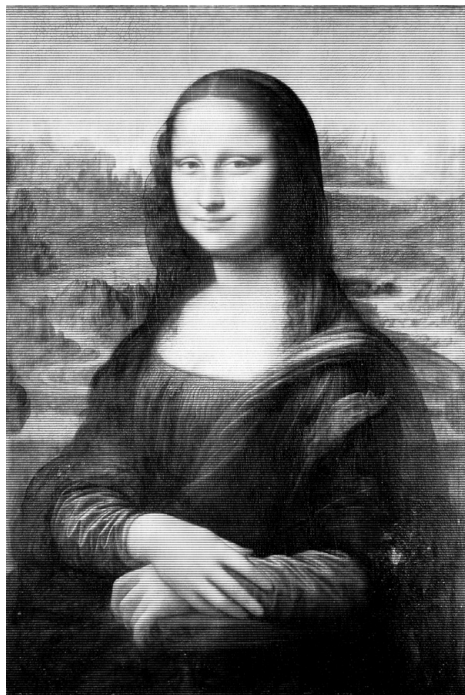
Use Case Life Cycle

Increasing level of detail

- **Start by capturing user intention**
 - Capture as an essential use case (either 1-column or 2-column form)
- **Once intention is solid, capture system interaction**
 - Write a use case
- **Avoid dependence on user interface**
 - Include dependencies on the user interface only as needed

User Intention vs System Interaction

Photo Editing Example



	inches	pixels
Width:	<input type="text" value="1.00"/>	<input type="text" value="300"/>
Height:	<input type="text" value="1.50"/>	<input type="text" value="450"/>
Resolution:	<input type="text" value="300"/>	

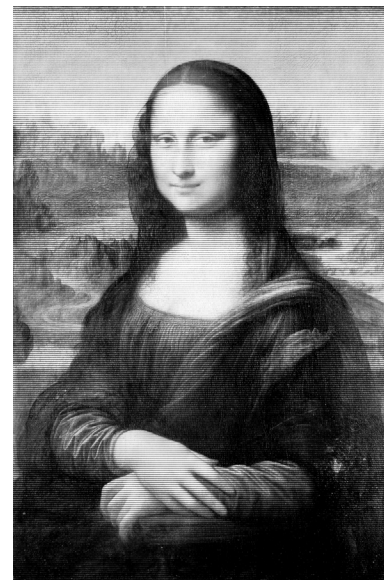
Photo Editing Example

- **Intention**

- User Intention: Resize a photo
- System Response: Display resized photo

- **Interaction 1**

- Actor selects photo
- System highlights photo
- Actor drags a corner
- System displays resized photo

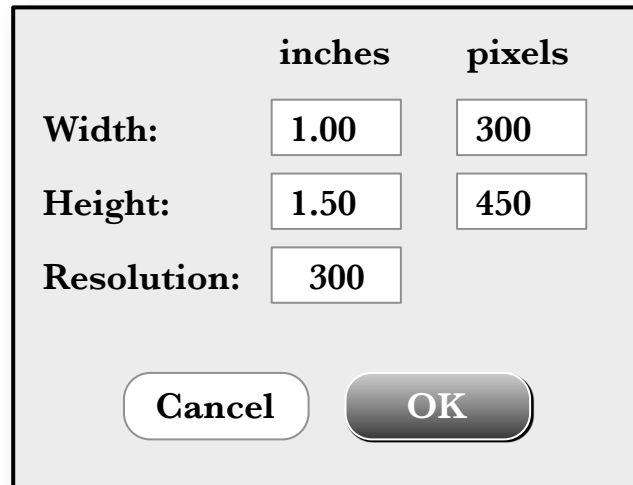


User Intention vs System Interaction

Photo Editing Example

• Intention

- User Intention: Resize a photo
- System Response: Display resized photo



	inches	pixels
Width:	1.00	300
Height:	1.50	450
Resolution:	300	

Cancel OK

• Interaction 2

- Actor selects photo
- System highlights photo
- Actor selects Resize menu
- System displays menu
- Actor enters dimensions
- System displays resized photo

User Intention vs System Interaction

Intention, Interaction, and Interface are at increasing levels of detail

- **Intention: technology-free & implementation-free**
 - Focus on what the customer wants
 - Cash Withdrawal Intent: “It’s me. The usual. Thanks!”
- **Interaction: implementation-free**
 - Focus on system behavior
 - Note: bank card and passcode are a technology for authentication
- **Interface**
 - Focus on design and implementation of layout, colors, fonts, sizes

Writing a Use Case

How much detail?

- **Start by capturing user intention**
 - Capture as an essential use case (either 1-column or 2-column form)
- **Once intention is solid, capture system interaction**
 - Write a use case
- **Avoid dependence on user interface**
 - Include dependencies on the user interface only as needed

Essential Use Cases

Focus on user intention

- **Definition of Essential Use Case**

- Use case that is technology-free and implementation-free

- **Example in conversational form**

User Intention

1. Identify self

2. Enter Passcode

3. Withdraw usual amount

System Response

Authenticate customer

Offer Bank Services

Dispense cash

A Representative Use Case Template

- **Name:** Active phrase conveying what will be achieved for the actor(s)
- **Goal:** Brief description of the purpose of the use case
- **Actors:** Every user, human and automated, involved in the use case
- **Basic Flow:** Sequence of actions – text, readable by stakeholders
- **Alternative Flows:** Variations on the basic flow
- **Extension Points:** Insertion points in the flow for additional behavior
- **Preconditions:** state of system for the use case to operate correctly
- **Postconditions:** state of the system after the use case completes
- **Relationships:** e.g., possible communication with other use cases

General Guidelines for Writing Flows

- Describe how the flow starts and ends
- Include events and data exchanges between the user and the system
- Start every action with “The actor ...” or “The system ...”
- Describe system behavior towards the goal, not the system design
- Make actions specific so they can be tested
- Avoid adverbs, such as very, more, rather, and the like
- Explicitly state it if the order of events is not important

Use Case Example: Place Order

Part 1 of 2

- **Description**

- Process for entering orders into the order processing system

- **Actors**

- Customer

- **Basic Flow**

1. The customer selects Place Order
2. The system displays the Place Order screen
3. The customer enters their name and number
4. The customer enters product codes for items they're ordering
5. For each product, the system adds the price of the item to the total
6. The customer enters credit card payment information
7. The system verifies the payment information
8. The system returns an order id and the use case ends

Use Case Example: Place Order

Part 2 of 2: Expand Alternative Flows as Needed

- **Alternative Flows**

- Shipping address incomplete
- Product code doesn't match actual products
- Payment bad
- Customer pays by check
- Customer sends order by mail
- Customer phones in order
- **Cancel Placing an Order**
 - **Precondition:** The user did not Submit
 1. Customer selects Cancel
 2. The system discards any entered information
 3. The system returns to the previous display
 4. The use case ends

Writing Alternative Flows

optional, exceptional, or truly alternative behavior

- **Alternative flows are triggered by a condition at some point in the basic flow**
 - **Example: Cancel Placing an Order**
 - **Precondition: The user did not Submit**
 1. This alternative begins when the customer selects Cancel
 2. The system discards any entered information
 3. The system returns to the previous display
 4. The use case ends
- **If it's not conditional, it's not an Alternative Flow**
 - Recall, the intent of alternative flows is to keep the basic flow simple
 - If a flow is not conditional it can be a separate use case or an alternative to some other use case

Structuring Flows

Subflows, Inclusions, Extensions

Flows are the most important part of a use-case description

- **Basic Flows focus on the “sunny day” sequence of events**
 - Capture the primary mandatory behavior in the basic flow
- **Subflows are used to improve readability of flows**
 - Even for the simplest systems, smaller self-contained subsections – each with a well defined purpose – may be better as named subflows
 - Keep subflows atomic; i.e., either all the actions are done or none are
 - For now, keep subflows private to a given flow

Introducing Inclusions and Extensions

- **Inclusion of a use case in another**
 - The basic flow explicitly invokes the inclusion, and then resumes
- **Extensions add behavior; they are a form of inheritance**
 - The basic flow must be complete by itself, without the extensions
 - An extension constitutes a use case that provides value to actors
 - Extensions add behavior that could be mandatory or optional

Examples of Inclusions

From the Place Order use case

- **Original: After the customer enters product codes**
 5. For each product, the system adds the price of the item to the total
- **Modified: Includes “Give Product Info”**
 5. For each product entered
 - 5.1. Include the Give Product Info use case
 - 5.2. The system adds the price of the item to the total

Examples of Inclusions

From the Place Order use case

- **Original:** After the customer enters **payment info**
 8. The system returns an order id and the use case ends
- **Modified:** Includes “**Update Account**” and “**Save Order**”
 - 8.1. The system verifies the payment information
 - 8.2. Include the Update Account use case
 - 8.3. Include the Save Order use case
 - 8.3. The system returns an order id and the use case ends

Examples of Extensions

Extensions add behavior without changing the basic flow

- **Example: Surveillance System**

- The basic flow monitors an area to detect intruders
- The extension, a use case in its own right, notifies authorities
- Detect Intruder is extended to Detect Intruder and Notify Authorities

- **Example: Social Extension of Place Order**

- The basic flow is unchanged: the customer enters product information
- The extension adds recommendations based on customer ordering data and “friend” data from the Facebook platform

- **Inclusions: The Basic Flow Initiates**

- The basic flow explicitly instructs the use case instance to perform the inclusion use case

- **Extensions: The Basic Flow is Unaware**

- Extensions points are labels in the basic flow
- The labels can be explicit labels (we have not discussed them so far)
- The points before numbered actions in flows are potential extension points
- The extension use case specifies an extension point in the basic flow where it (the extension use case) is to be performed

Review of Use Cases

Use Case Review Questions

Completeness

- Are any actors missing?
- For each step in the use case, who are the actors?
- Are there any missing triggers?
- Are the identified triggers correct?
- Are all terms defined?
- Is it clear how to proceed from one step to the next in the basic flow?
- Are all likely alternative flows identified?
- Can you give an example of an alternative flow that should be described but is not?
- Does the precondition include all cases that must be true for the basic flow to execute?
- If the precondition is violated is there a corresponding exception?
- Does the basic flow set up the postcondition?

- **Map from Use Cases to Prototypes and vice-versa**
 - If it's not clear what or how to do some step in the use case, prototype
 - e.g., what privileges should a user have?
 - e.g., how accurate is face recognition with the current algorithm?
 - When a prototype clarifies an issue, the use case may need to be revised
 - e.g., three different types of users, with different privileges, must be able to use the system
 - e.g., face cannot be recognized when travelers are moving; travelers must be stationary

Use Cases or User Stories? Or, Both?

They are complementary

- **Why write Use Cases with User Stories?**

- The list of goal names provides a short summary of the functionality
- Basic Flows provide a context for what the system will and will not do
- Extensions in use cases provide lookahead into completeness issues
- Extensions can answer detailed questions that developers might face
- The full use case set shows that every user's needs are addressed

Use Cases or User Stories? Or, Both?

They are complementary

- **Experience with Use Cases**

- Writing of use cases can be staged through the life of the project – they don't all need to be written up front
- With short iterations/sprints, it's not practical to implement an entire use case in just one. This may result in extra work writing user stories.
- Writing good use cases requires thinking. “It is much easier to write user-story tags on index cards and let the project blow up later.”

Use Cases Summary

A use case is a sequence of actions a system performs that yields an observable result of value to a particular actor

- Use cases are initiated by actors and supported by the system
- A use case describes the actions the system takes to deliver to the actor
- Taken together, all the use cases constitute all the ways of using the system

