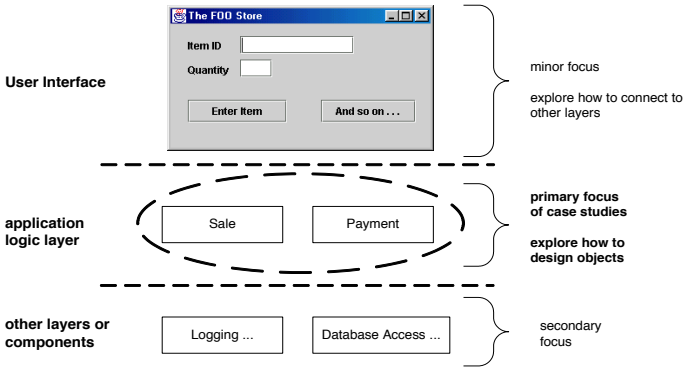
CS 619 Introduction to OO Design and Development

Use Cases

Fall 2012

Define the Problem

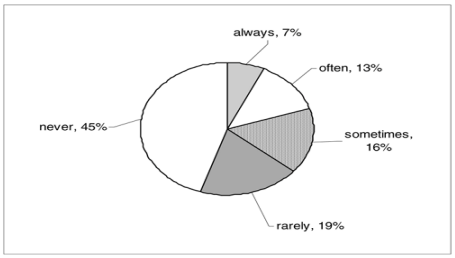
- The most critical question:
- "Is this the right system to make?"



Evolutionary Requirements

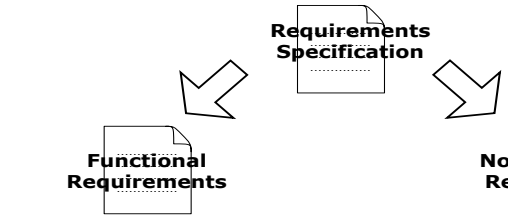
- Requirements are capabilities and conditions to which the system, and more broadly, the project must conform.
- One of the best practices of UP is manage requirements.
  - a systematic approach to finding, documenting, organizing, and tracking the changing requirements of a system.
  - Do an iterative and evolutionary requirement analysis, not a waterfall one.

Figure 5.1. Actual use of waterfall-specified features.

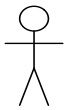


3

Requirements



- functionality as demanded by the end users



- constraints placed on the global system or the development process.
- quality attributes, such as performance, user-friendliness, maintainability, …

4

Types and Categories of Requirements in UP

- In the UP, requirements are categorized according to the **FURPS+ model**
- ***Functional*** - features, capabilities, security.
- ***Usability*** **-** human factors, help, documentation.
- ***Reliability*** - frequency of failure, recoverability, predictability.
- ***Performance*** - response times, throughput, accuracy, availability, resource usage.
- ***Supportability*** - adaptability, maintainability, internationalization, configurability.
- The "**+**" sign means
  - Implementation - resource limitations, languages and tools, hardware..
  - Interface - constraints imposed by interfacing with external systems.
  - Operations - system management in its operational setting.
  - Packaging - for example, a physical box.
  - Legal - licensing and so forth.

5

Requirements Specification Technique

A requirements specification technique must be

- **understandable:** for all parties involved including its users
- **precise***:* so that parties agree what's inside and outside the system

  Can you write a (regression) test for each requirement?
- **open:** so that developers have enough freedom to pick an optimal solution

  Requirements specify the "what", not the "how ".

6

Jan 7, 2009

Use Cases

- Informally, use cases are text stories of some actor using a system to meet goals.
- An example in ***brief format*** use case:

Process Sale: A customer arrives at a checkout with items to purchase. The cashier uses the POS system to record each purchased item. The system presents a running total and line-item details. The customer enters payment information, which the system validates and records. The system updates inventory. The customer receives a receipt from the system and then leaves with the items.

7

Use Cases

- A **use case** describes outwardly visible requirements of the system.
  A use-case is a generic description of an entire transaction executed to achieve a goal (= the use case goal) and involving several actors.
- An **actor** is something with behavior and have responsibilities.
  To carry out responsibilities, an actor sets goals
  • **Primary actor** (= stakeholder) has unsatisfied goal and needs system assistance
  • **Secondary actor** provides assistance to satisfy the goal
- A **scenario** is a specific sequence of actions and interactions between actors and the system.
  also called a ***use case instance***. It is one particular story of using a system, or one path through the use case.

8

Use Cases

- An example in ***casual format*** use case with alternate scenarios:

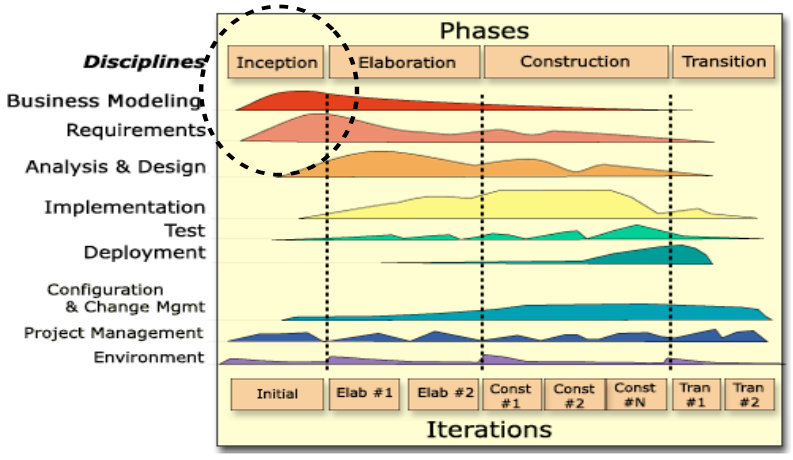
Handle Returns

- 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 customer paid by credit, and the reimbursement transaction to their credit account is rejected, inform the customer and pay them with cash.
  - If the item identifier is not found in the system, notify the Cashier and suggest manual entry of the identifier code (perhaps it is corrupted).
  - If the system detects failure to communicate with the external accounting system, …

A use case is a *collection of related success and failure scenarios* that describe an actor using a system to support a goal.

9

UP: Inception



10

Inception is not the Requirements Phase

- Inception is the initial short step to *establish a common vision and basic scope for the project*.

  Scope Definition + **Risk Identification**

  + **Actors & Use cases** + **Project Plan**

- It will include analysis of perhaps 10% of
  - the use cases
  - analysis of the critical non-functional requirement,
  - creation of a business case, and
  - preparation of the development environment so that programming can start in the following ***elaboration*** phase.

11

Inception: System Scope

During inception you must define the system's scope.

- Used to decide what lies inside & outside the system

Scope

- should be *short*

  (1 paragraph for small projects; 1/2 a page for mid-size projects; 2-3 pages for large projects) long statements are not convincing

- should be *written* down

  later reference when prioritizing use cases

- should have end-user *commitment*

  end-user involved in writing + formally approved by a project steering committee

12

System Scope: Example

(Example from [Schn98a])

- "We are developing order-processing software for a mail-order company called National Widgets, which is a reseller of products purchased from various suppliers.
  - Twice a year the company publishes a catalogue of products, which is mailed to customers and other interested people.
  - Customers purchase products by submitting a list of products with payment to National Widgets. National Widgets fills the order and ships the products to the customer's address.
  - The order-processing software will track the order from the time its is received until the product is shipped.
  - National Widgets will provide quick service. They should be able to ship a customer's order by the fastest, most efficient means possible."

13

Analyzing the Example

The previous example of a system scope description is

- short : quick assessment of what's the system supposed to do
- goal-oriented (track orders): open for various solution.
- includes criteria (quick service, track *all* of the ordering process, ...)
  - will be used to evaluate whether we accomplished the goals
- provides context
  - National Widgets is reseller ⇒ *external* suppliers & shipment
  - some problems are out of scope
- and very importantly … imperfect
  - may be improved when understanding increases, but goal and main criteria should not change once approved

14

Inception: Risk Factors

During inception we must identify the project's *risk factors*

- we do not have control over the system's context and it will change
- projects never go according to plan, identify potential problems early (... including wild success)

Context	Risk Factors	Impact	Likely ?
Competitors	Time to market (too late/too early)		
Market trends	More internet at home		
Potential disasters	Suppliers don't deliver on time		
	System is down		
Expected users	Too many/few users		
Schedule	Project is delivered too early/too late		
Technology	Dependence on changing technology		
	Inexperienced team		
	Interface with legacy systems		

15

Identifying Actors and Use Cases

Following questions may help during the identification process.

Actors

- Who uses the system?
- Who installs the system?
- Who starts up/shuts down the system?
- Who maintains the system?
- What other systems use this system?
- Who provides information to this system ?
- Does anything happen automatically at a preset time ?

Use Cases

- What functions will the actor want from the system ?
- What actors will create, read, update, or delete information stored inside the system ?
- Does the system need to notify actors about changes in its internal state ?
- Who gets information from this system ?
- Are there any external events the system must know about ?
- What actor informs the system about those events ?

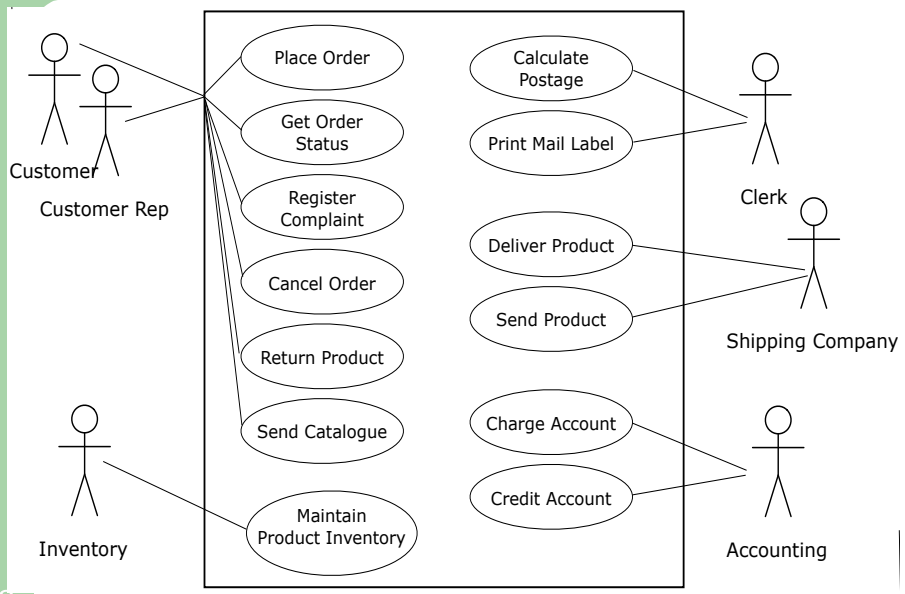
16

Three kinds of actors

- Primary actor
  - has user goals fulfilled through using services of the SuD. For example, the cashier.
    Why identify? To find user goals, which drive the use cases.
- Supporting actor
  - provides a service to the SuD.
    E.g. automated payment authorization service. Often a computer system, but could be an organization or person.
    Why identify? To clarify external interfaces and protocols.
- Offstage actor
  - has an interest in the behavior of the use case, but is not primary or supporting; for example, a government tax agency.
    Why identify? To ensure that all necessary interests are identified and satisfied. Offstage actor interests are sometimes subtle or easy to miss unless these actors are explicitly named.
- *Primary* and *supporting* actors will appear in the action steps of the use case text.

17

Example: Actors and Use Cases

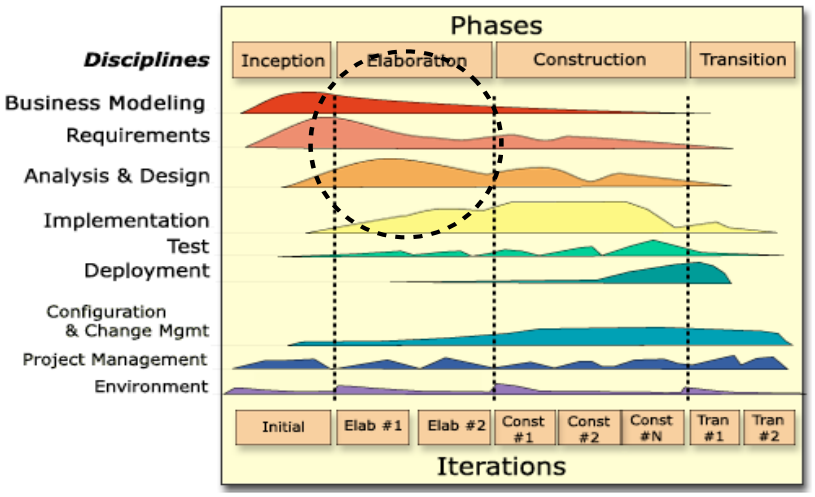


18

Actors or Use Case First?

- Because we have to understand each part of Use Cases, the parts are presented separately.
- But those who create use cases switch back and forth. The text describes use cases substantially before paying attention to actors.
- Typically, both actors and use cases are identified early and then examined to see if more use cases can be found from the actors, or more actors found by examining the use cases.

UP: Elaboration



20

Elaboration: Primary & Secondary Scenarios

During elaboration we must refine the use cases via scenarios from the actors point of view.
- List of steps to accomplish the use case goal

- **Primary "success" scenario**
  Happy path scenario; Scenario assuming everything goes right (i.e., all input is correct, no exceptional conditions, ...)

- **Secondary "alternative" scenarios**
  Scenario detailing what happens during special cases (i.e., error conditions, alternate paths, ...)

21

Three common use case formats

- brief
  - Terse one-paragraph summary, usually of the main success scenario. The prior Process Sale example was brief.

    Process Sale: A customer arrives at a checkout with items to purchase. The cashier uses the POS system to record each purchased item. The system presents a running total and line-item details. The customer enters payment information, which the system validates and records. The system updates inventory. The customer receives a receipt from the system and then leaves with the items

  - When? During early requirements analysis, to get a quick sense of subject and scope. May take only a few minutes to create.

22

Three common use case formats

- Causal
  - Informal paragraph format. Multiple paragraphs that cover various scenarios. The prior Handle Returns example was casual.
  - 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 customer paid by credit, and the reimbursement transaction to their credit account is rejected, inform the customer and pay them with cash.
    - If the item identifier is not found in the system, notify the Cashier and suggest manual entry of the identifier code (perhaps it is corrupted).
    - If the system detects failure to communicate with the external accounting system,
- When? During early requirements analysis, to get a quick sense of subject and scope. May take only a few minutes to create.

23

Three common use case formats

- Fully dressed
  - All steps and variations are written in detail, and there are supporting sections, such as preconditions and success guarantees.
  - Example to be seen shortly.
  - When? After many use cases have been identified and written in a brief format, then during the first requirements workshop a few (such as 10%) of the architecturally significant and high-value use cases are written in detail.

24

A Sample Template of Fully dressed format

Use Case Section	Comment
Use Case Name	Start with a verb.
Scope	The system under design.
Level	"user-goal" 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 the reader.
Main Success Scenario	A typical, unconditional happy path scenario of success.
Extensions	Alternate scenarios of success or failure.
Special Requirements	Related non-functional requirements.
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.

2

Naming Use Cases

Each use case should have a name that indicates what value (or goal) is achieved by the actor's interaction with the system

- Must be a complete process from the viewpoint of the end user.
- Usually in verb-object form
  E.g. Buy Pizza
- Use enough detail to make it specific
- Use active voice, not passive
- From viewpoint of the actor, not the system

Naming Use Case: Examples

- Purchase Concert Ticket
- Purchase Concert Tickets
- Purchase Ticket
- Ticket Purchase
- Ticket Order
- Pay for Ticket

Examples

- Excellent - Purchase Concert Ticket
- Very Good - Purchase Concert Tickets
- Good - Purchase Ticket (insufficient detail)
- Fair - Ticket Purchase (passive)
- Poor - Ticket Order (system view, not user)
- Unacceptable - Pay for Ticket (procedure, not process)

CRUD

- Examples of bad use case names with the acronym CRUD. (All are procedural and reveal nothing about the actor's intentions.)

- C - actor Creates data
- R - actor Retrieves data
- U - actor Updates data
- D - actor Deletes data

Fully Dressed Format Explained

- **Scope**
  - The scope bounds the system (or systems) under design. Typically, a use case describes use of one software (or hardware plus software) system; in this case it is known as a system use case.

30

Fully Dressed Format Explained

- **Level**
  - Use cases are classified as at the **user-goal level** or the **subfunction level**, among others.
  - A **user-goal level** use case is the common kind that describe the scenarios to fulfill the goals of a primary actor to get work done
  - A **subfunction-level** use case describes substeps required to support a user goal, and is usually created to factor out duplicate substeps shared by several regular use cases (to avoid duplicating common text).

    An example is the subfunction use case Pay by Credit, which could be shared by many regular use cases.

31

Fully Dressed Format Explained

- **Stakeholders and Interests List**
  - The [system] operates a contract between stakeholders, with the use cases detailing the behavioral parts of that contract…The use case, as the contract for behavior, captures all and only the behaviors related to satisfying the stakeholders' interests.
  - The stakeholder interest viewpoint provides a thorough and methodical procedure for discovering and recording all the required behaviors.

32

Fully Dressed Format Explained

- **Preconditions and Success Guarantees (Postconditions)**
  - **Preconditions** state what must always be true before a scenario is begun in the use case.
  - Preconditions communicate noteworthy assumptions that the writer thinks readers should be alerted to.
  - Success guarantees (or **postconditions**) state what must be true on successful completion of the use case - either the main success scenario or some alternate path. The guarantee should meet the needs of all stakeholders.

33

Fully Dressed Format Explained

- **Main Success Scenario and Steps (or Basic Flow)**
  - "happy path" scenario, or the more prosaic "Basic Flow" or "Typical Flow."
  - It describes a typical success path that satisfies the interests of the stakeholders.
- Guideline
  - defer all conditional handling to the Extensions section.
  - always capitalize the actors' names for ease of identification.

34

Fully Dressed Format Explained

- **Extensions (or Alternate Flows)**
  - Normally comprise the majority of the text.
  - They indicate all the other scenarios or branches, both success and failure.
  - Extensions section was considerably longer and more complex than the Main Success Scenario section; this is common.
  - Extension scenarios are branches from the main success scenario, and so can be notated with respect to its steps 1…N.
  - For example, at Step 3 of the main success scenario there may be an invalid item identifier, either because it was incorrectly entered or unknown to the system. An extension is labeled "3a"; it first identifies the condition and then the response. Alternate extensions at Step 3 are labeled "3b" and so forth.

35

Fully Dressed Format Explained

- Extensions (or Alternate Flows)
  - An extension has two parts: the condition and the handling.
- Guideline: When possible, write the condition as something that can be detected by the system or an actor.

  To contrast:
  - 5a. System detects failure to communicate with external tax calculation system service:
  - 5a. External tax calculation system not working:
  - Which one you would prefer?

36

Fully Dressed Format Explained

- Extensions (or Alternate Flows)
  - Extension handling can be summarized in one step, or include a sequence, as in this example, which also illustrates notation to indicate that a condition can arise within a range of steps:
  - 3-6a: Customer asks Cashier to remove an item from the purchase:
    1. Cashier enters the item identifier for removal from the sale.
    2. System displays updated running total.
  - At the end of extension handling, by default the scenario merges back with the main success scenario, unless the extension indicates otherwise (such as by halting the system).

37

Fully Dressed Format Explained

- Performing Another Use Case Scenario
  - 3a. Invalid item ID (not found in the example):
    - 1. System signals error and rejects entry.
    - 2. Cashier responds to the error:
      - 2a. …
      - 2c. Cashier performs ***Find Product Help*** to obtain true item ID and price.

38

Fully Dressed Format Explained

- **Special Requirements**
  - If a non-functional requirement, quality attribute, or constraint relates specifically to a use case, record it with the use case. These include qualities such as performance, reliability, and usability, and design constraints (often in I/O devices) that have been mandated or considered likely.
  - many practitioners find it useful to ultimately move and consolidate all non-functional requirements in the ***Supplementary Specification***, for content management, comprehension, and readability, because these requirements usually have to be considered as a whole during architectural analysis.

39

Fully Dressed Format Explained

- Technology and Data Variations List
  - 3a. Item identifier entered by laser scanner or keyboard.
  - 3b. Item identifier may be any UPC, EAN, JAN, or SKU coding scheme.
  - 7a. Credit account information entered by card reader or keyboard.
  - 7b. Credit payment signature captured on paper receipt. But within two years, we predict many customers will want digital signature capture.

40