Chapter Three

Gathering user requirements and Reusable technologies

1

What you Gather?

- Functional requirement
- Non-functional requirement
- Use case
- Actors

2

Functional Requirement

- Something which is intended to be used or practical
- Specify what a system do
- What the system does, or.
- Captures behavioral aspects
- Backbone of the software requirement.
- It depends on the complexity of the software system.
- Define functions and functionality.

3

Examples of Functional Requirements:

- **Search** option given to user to search from various invoices.
- user should be able to **mail any report** to management.
- users can be divided into groups and groups can be given **separate rights**.

4

Cont…

- Should comply business rules and administrative functions.
- Software is developed keeping downward compatibility intact.

5

E.g.2: Library Management System (LMS)

- What are functional requirements
  1. Issue a book
  2. Return a book
  3. Membership facility
  4. Reservation of books
  5. Recording member's data
  6. Visiting book status

6

Non-Functional Requirements

- **Implicit expectations from the product**
- **Expected features and specifically documented requirements, known as** "quality attributes" of a system.
- Also "qualities", "quality goals", "quality of service requirements", "constraints" and "non-behavioral requirements".

7

Cont…

- **Qualities—that is non-functional requirements—can be divided into two main categories:**
- **Execution qualities**, such as safety, security and usability, which are observable during operation (at run time).
- **Evolution qualities**, such as testability, maintainability, extensibility and scalability, which are embodied in the static structure of the system.

8

Cont…

- Non-functional requirement are more important than functional requirement

Why ?

Even if you miss one of FR the system can work properly but if you miss one of NFR the system is under problem .

9

Example: Library Management system

- What are non-functional requirements
  1. Security: Security levels (users, librarian administrative)
  2. Usability: all members and users should effectively and easily use the system (user friendly).
  3. Availability: the system gives service 8/24.
  4. Scalability: the system should be enough to accommodate to the changes .
  5. Reliability: the system should be enough to handle those problems
  6. Efficiency: response time

10

Techniques (Methodology) of Data Gathering

- ***Interviewing***
- ***Observation: (Ethnography)*** *open and Close*
- ***Questioner***
- ***Manual(case study)***
- ***Workshop***
- ***Group discussion (Brain storming)***: technique where groups of people discuss a topic and say anything that comes into their minds about it.

11

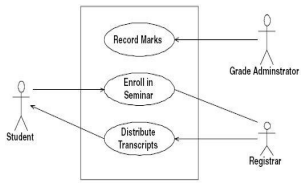
Essential Use Case Modeling/Diagram

- **Use case:** describes a ***sequence of actions*** that provide something of measurable value to an actor and is drawn as a **horizontal ellipse**.
- **Actor:** is a ***person, organization, or external system*** that plays a role in one or more interactions with your system. Actors are drawn as **stick figures**.

- An **association** exists whenever an actor is involved with an interaction described by a use case. It is indicated in use case diagrams by **solid lines**

12

- **System boundary boxes (optional).** indicates the scope of your system. It can be shown by drawing a rectangle around the use cases

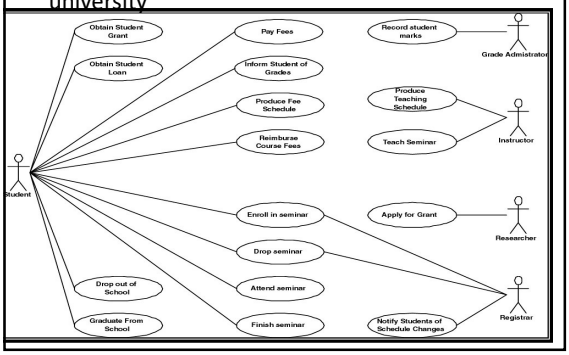


A simple use case diagram for a university

- ***Use case diagrams give you a very good overview of the requirements***.

13

- more complex use case diagram for the same university



- You often draw a use case diagram while you are identifying use cases, actors, and the associations among them.
- There are several interesting things to note about the use cases
  - **These are just preliminary use cases:** overtime you will change existing use cases, break them apart, combine them, introduce new use cases, and remove ones that do not make sense.
  - **No time ordering is indicated between use cases.** For example, you need to enroll in a seminar before you attend it, and you need to attend it before you pass it.
  - **Customer actors are usually involved in many use cases:** This is a normal phenomenon—after all; the main goal of most organizations is to provide services to their customers.

15

- **No arrowheads are on the associations:** showing arrowheads on use case associations can be confusing for many people
- **Use cases should be functionally cohesive.** A use case should encapsulate one service that provides value to an actor.
- **Use cases should describe something of business value.** Use cases describe something of value to one or more actors.
- **Every actor is involved with at least one use case, and every use case is involved with at least one actor:** If a use case does not provide service to an actor, then why have it? If an actor not involved in a use case exists, why model it?
- **Repetitive actions need not be expressed within a single use case:** For example, a student will typically enroll in many seminars each semester. It is not necessary to express this fact within the use case, unless it makes sense to do so.

16

Identifying Actors

- An actor represents anything or anyone that interacts with your system
  - People, external systems, and other organizations
- To help indentify actors in your system, you should ask yourself the following questions:
  - Who is the main customer of your system?
  - Who obtains information from this system?
  - Who provides information to the system?
  - Who installs the system?
  - Who operates the system?
  - Who shuts down the system?
  - What other systems interact with this system?
  - Does anything happen automatically at a preset time?
  - Who will supply, use, or remove information from the system?
  - Where does the system get information?

17

- Description of the Grade Administrator actor.

Name: Grade Administrator

Description: A Grade Administrator will input, update, and finalize in the system the marks that students receive on assignments, tests, and exams. Deans, professors (tenured, assistant, and associate), and teaching assistants are all potential grade administrators

18

Identifying Use Cases

- ***Identify potential services*** by asking your stakeholders the following questions from the point of view of the actors:
  - What are users in this role trying to accomplish?
  - To fulfill this role, what do users need to be able to do?
  - What are the main tasks of users in this role?

19

  - What information do users in this role need to examine, create, or change?
  - What do users in this role need to be informed of by the system?
  - What do users in this role need to inform the system about?
- **For example, from the point of view of the *Student* actor, you may discover that students**
  - Enroll in, attend, drop, fail, and pass seminars.
  - Need a list of available seminars.
  - Need to determine basic information about a seminar, such as its description and its prerequisites.

20

Cont…

- Obtain a copy of their transcript, their course schedules, and the fees due.
- Pay fees, pay late charges, receive reimbursements for dropped and cancelled courses, receive grants, and receive student loans.
- Graduate from school or drop out of it.
- Need to be informed of changes in seminars, including room changes, time changes, and even cancellations.
- Provide fundamental information about themselves such as their name, address, and phone number.

21

- another way to identify use cases is to ask **your stakeholders to brainstorm the various scenarios**, often called **usage/use case scenarios** which your system may or may not support.
- For example, the following would be considered ***use case scenarios*** for a university information system:
  - A **student** wants to **enroll in a seminar**, but the **registrar** informs him that he does not have the **prerequisites** for it.
  - A **student** wanted to **enroll in a seminar** that she does have the **prerequisites** for and seats are still available in the **seminar**.
  - A **professor** requests a ***seminar list*** for every course he teaches.
  - A **researcher** applies for a **research grant**, but only receives partial funding for her project.
  - A **professor** submits **student marks** to the system. These marks may be for exams, tests, or assignments.
  - A **student** wants to **drop a seminar** the day after the drop date.
  - A **student** requests a **printed copy of his transcript**, so he can include copies of it with his résumé.

22

- You can take either approach, or combine the two, if you like, but the main goal is to end up with a **lot of information regarding the behavioral aspects of your system**.
- The next step is to group these aspects, by similarity, into use cases. ***Remember that a use case provides a service of measurable value to an actor.***

Use Case Discription

- A use case is a sequence of actions that provide a measurable value to an actor.
- An essential use case, sometimes called a ***business use case***, is a **simplified**, **abstract**, **generalized** use case that captures the intentions of a user in a technology- and implementation-independent manner.
- **Business use case** are somewhere in between essential and system use cases.

23

- ***Preconditions*** describe the state or status the system must be in, prior to the execution of a use case. What must be true before the use case can execute?
- ***Postconditions*** describe the state or status of the system as a result of the execution of the use case.

24

•*Enroll in Seminar* as an essential use case

Enroll in Seminar

ID: UC 1

Preconditions:

The student is enrolled in the university.

Post conditions:

None

Actor(s)	**Response**
Student identifies himself	Verifies eligibility to enroll via *BR1 Determine Eligibility to Enroll*. Indicate available seminars
Choose seminar	Validate choice via *BR2 Determine Student Eligibility to Enroll in a Seminar*. Validate schedule fit via *BR3 Validate Student Seminar Schedule* Calculates fees via *BR 4 Calculate Student Fees* and *BR5Calculate Taxes for Seminar*. Summarize fees Request confirmation
Confirm enrollment	Enroll student in seminar Add fees to student bill Provide confirmation of enrollment

25

Base Use Case Example

Use Case ID	UC-100
Use Case	Withdraw Funds
Actors	(P) Customer
Description	Customer logs in, selects withdraw funds, enters an amount and receives cash and receipt
Pre-conditions	Welcome screen is on
Flow of Events	1. Customer slides card in and out 2. Machine prompts customer for password 3. Customer enters password 4. System authenticates customer 5. System presents user with a choice menu 6. Customer selects Withdraw Funds option 7. System asks customer to select account 8. Customer selects account 9. System asks customer for amount to withdraw 10.Customer enters amount 11.System dispenses cash and prints receipt 12.System logs customer out
Post-conditions	Welcome screen is back on
Alternative Flows	Customer may not be authenticated; customer may not have sufficient funds; machine may not have enough cash
Priority	High
Non-Functional Requirements	Cash dispensed within 10 seconds after amount is entered
Assumptions	Customer speaks English
Source	Bank's Operational Procedures Manual

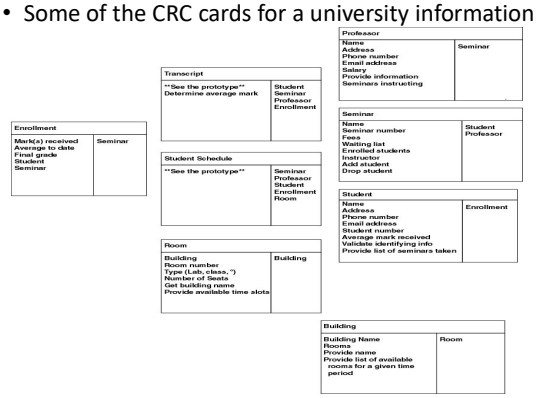
26

Domain modeling with class responsibility collaborator (CRC) cards

- A class responsibility collaborator model is a collection of standard index cards that have been divided into three sections.
  - A **class** represents a collection of similar objects,
  - A **responsibility** is something that a class knows or does, and
  - A **collaborator** is another class that a class interacts with to fulfill its responsibilities
- cards can be identified by **working closely with a project stakeholder** who **understood this part of the domain**

27

- Some of the CRC cards for a university information



Essential User Interface Prototyping

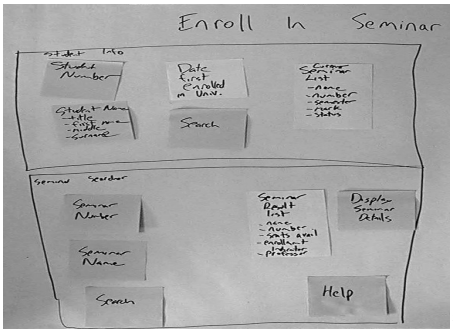
- The UI is the portion of software with which a **user directly interacts**
- It ***is the initial state—the beginning point—of the UI prototype for your system***
- represents the **general ideas behind the UI**, but not the exact details
- Represents and models **UI requirements** in a **technology-independent manner**, evolved through analysis and design to result in the final user interface for your system
- your prototyping tools are simple, including **whiteboards, flip chart paper, and sticky notes**
- Essential UI modeling focus on your **users and their usage** of the system, not system features

29

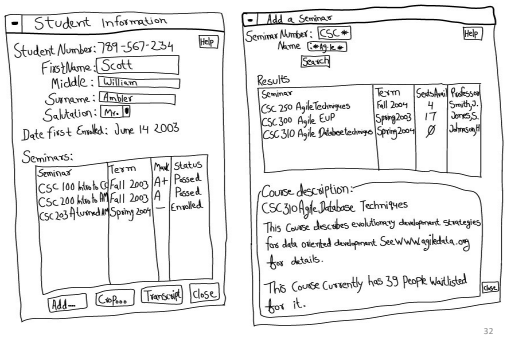
- **Active Stakeholder Participation Is Crucial** to provide requirements, to prioritize them, and to make decisions in a timely manner
- A major UI element represents, potentially a **screen, HTML page, or report.**
- A minor UI element represents, widgets such as **user input fields, menu items, lists, or static text fields such as labels**
- Always remember that your **project stakeholders are the official source of requirements**, not developers
- If you identify some **new functionality** that you think is required you need to **convince your stakeholder** that it is a good idea

30

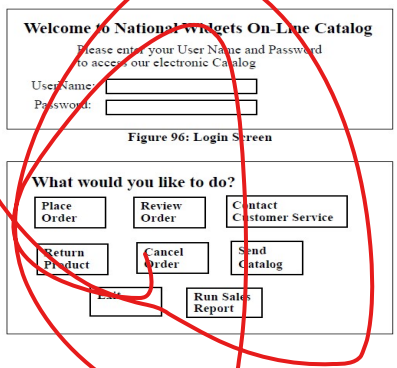
- An essential UI prototype to enroll a student in a seminar



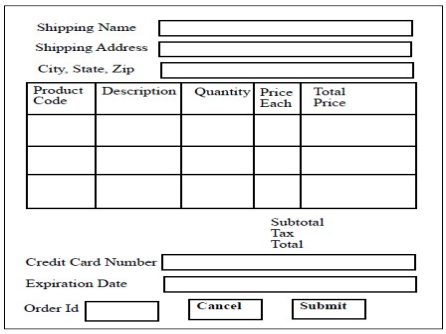
- Revised essential UI prototype to enroll a student in a seminar

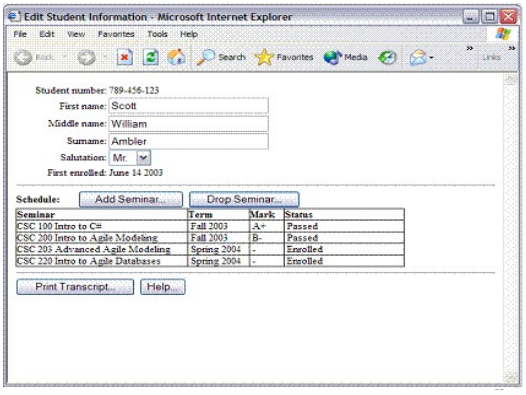


Tommorow ajke



Cont..





Developing a supplementary Specification

- *The goal of requirements engineering is **not to create documentation**; it is to **convey ideas from project stakeholders to developers**.*
- A supplementary specification is a **document that contains requirements not contained directly in your use cases**.
  - **business rules,**
  - **technical requirements, and**
  - **Constraints**
  - **glossary**

Business rules

- Rules that influence the behavior of your business
- focus on access control issues;
- Some business rules focus on the policies of your organization

For example:

- professors are allowed to input and modify the marks of the students taking the seminars they instruct, but not the marks of students in other seminars
- how to convert a percentage mark (for example, 91 percent) that a student receives in a seminar into a letter grade (example, A-).
- each business rule has a **unique identifier**. The convention is to use the format of **BR#**, but you are free to set your own numbering approach.
- **unique identifier** enables you to refer easily to business rules in other **development artifacts, such as class models and use cases**

37

Example Business Rules

- For new customers, we must perform a credit check on purchases over $500.00
- One condition for a house loan to be approved is that the monthly payment of the house loan must amount to no more than 30% of a family's monthly gross income.
- We only write health insurance policies for companies of 10 or more people. We do not provide individual health insurance policies.
- All project assets that describe the working system must be stored for a period of 3 years.
- All mathematical results must be calculated to at least 10 digits of accuracy

38

- BR7 Tenured professors may administer student grades.
- BR8 Teaching assistants who have been granted authority by a tenured professor may administer student grades.
- BR9 Table to convert between numeric grades and letter grades.
- BR6 All master's degree programs must include the development of a thesis.

Name:	Tenured professors may administer student grades
Identifier:	BR7
Description:	Only tenured professors are granted the ability to initially input, modify, and delete grades students receive in the seminars that they and they only instruct. They may do so only during the period a seminar is active.
Example:	Dr. Bruce, instructor of "Biology 301 Advanced Uses of Gamma Radiation," may administer the marks of all students enrolled in that seminar, but not those enrolled in "Biology 302 Effects of Radiation on Arachnids," which is taught by Dr. Peters.
Source:	University Policies and ProceduresDoc ID: U1701Publication date: August 14, 2000

39

- **Name.** should give you a good idea about the topic of the business rule.
- **Description.** defines the rule exactly.
- **Example (optional).**
- **Source (optional).** The source of the rule is indicated so it may be verified (it is quite common that the source of a rule is a person, one of your project stakeholders, or a team of people).

****Read the principles of writing Business rules****

Technical Requirements

- Focuses on **performance-related issues, reliability issues,** and **availability issues**
- are often called **service-level requirements** or **nonfunctional requirements**

40

- they have a **name** and a **unique identifier** (the convention is to use the **format TR#,** where TR stands for technical requirement)
- document technical requirements in the same manner as business rules, including a **description**, an **example**, a **source**, and a **revision history**. Ex.

- TR34 The system shall be available 99.99 percent of the time for any 24- hour period.
- TR78 A seminar search will occur within less than three seconds 95 percent of the time.
- TR79 A seminar search will occur within no more than ten seconds 99 percent of the time.
-
-

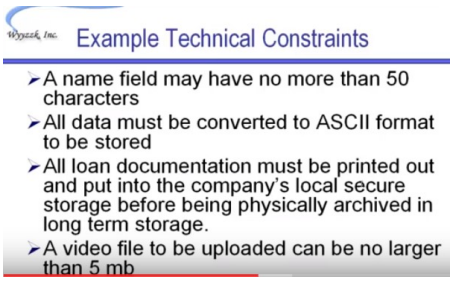
42

Constraints

- a restriction on the degree of freedom you have in providing a solution
- can be economic, political, technical, or environmental and pertain to your project resources, schedule, target environment, or to the system itself
- Like business rules and technical requirements, constraints are documented in a similar manner. Ex.

- **C24 The system will work on our existing technical infrastructure—no new technologies will be introduced.**
- **C56 The system will only use the data contained in the existing corporate database.**
- **C73 The system shall be available 99.99 percent of the time for any 24-hour period.**
- **C76 All master's degree programs must include the development of a thesis.**

44



45

Glossaries

- collection of definitions of both **technical and business** terms which can **improving the communications between developers and users**.
- For example: Terms that stakeholders likely do not know such as **XP, C#, J2EE, Application server** and terms that developers may not know such as **convocation, grant, and transcript** etc.

Identify and documenting Change Cases

- used to describe **new potential requirements** for a system or **modifications to existing requirements**
- Change cases are modeled in a simple manner.
  - You **describe the potential change t**o your existing requirements,
  - indicate the **likeliness of that change occurring**, and
  - indicate the **potential impact** of that change
- can be identified throughout the ***course of your overall development*** and are often the ***result of brainstorming*** with your project stakeholders

46

Change case: Need to support both Linux and Microsoft platforms.

Likelihood: Very likely to happen for application and database servers within six months; medium probability that Linux will be adopted by the school for desktop machines.

Impact: Unknown. Currently application servers for other applications run Microsoft-based operating systems and will likely continue to do so. New servers will likely have Linux installed. Desktop impact is hard to quantify as the Linux market is evolving rapidly. Should be re-evaluated six months from now.

47

Change case: The University will open a new campus.

Likelihood: Certain. It has been announced that a new campus will be opened in two years across town.

Impact: Large. Students will be able to register in classes at either campus. Some instructors will teach at both campuses. Some departments, such as Computer Science and Philosophy, are slated to move their entire programs to the new campus. The likelihood is great that most students will want to schedule courses at only one of the two campuses, so we will need to make this easy to support.

48