Gathering User Requirements

Chapter 3

Data collection techniques

- Observation
- Interview
- Questionnaire
- Document analysis
- Brainstorming

1. Gathering User Requirements

- The first Step – not easy
  - People don’t want to spend time here
    - ✓ SME don’t want to invest time
    - ✓ Developers want to get into the “real work” of coding
    - ✓ Senior Management want to see progress
  - Communicate with everyone
- Use Models
  - Use Case Modeling – behavior and functional tasks
  - Domain Modeling – business domain concepts
  - User Interface Prototyping – screen,

Gathering Requirements cont…

Requirements can be

1. Functional requirements:- Functional requirement is what a system does or expected to do. Eg. Details of inputs into the system from paper form and documents
2. Non functional requirements:- This is how well the system provides the functional requirements. Eg. Performance, ability, security of the system

2. Putting together a requirement gathering team

- Your requirement team will be composed of SME. Who will provide you with detail information.
- These people are responsible for:
  - ✓ Requirement gathering
  - ✓ Documenting
  - ✓ Validation
- Formal requirement team often includes:
  - ✓ **Facilitator** : some one responsible for managing modeling session
  - ✓ **Scriber** :responsible for recording information as it is identified

Cont...

- The first step to put together a requirement gathering team is to identify your project stakeholders.
- Project stockholder are any one who could be materially affected by the implementation of a new system or application.
- Project stockholder includes:
  - ✓ Direct user of the system
  - ✓ The customer
  - ✓ Any one who will affected by the output of the system products.
  - ✓ Any one who will evaluated
  - ✓ Any internal or external user
  - ✓ Any who will operate and support the system

cont. . .

- The Next step to put together a requirement team is to identify who will be on your requirements modeling team.
- You may want to distinguish b/n active members of your team such as full time requirement analysis and key SME and supporting members.

In general: Putting together a requirement team has to be:

1. Choosing Good subject mater experts:

Good SME have the following qualities:

To know the business :they understand one or more aspect of the business.

Cont…

- ✓ They think logically: not only the SME to understand the business but also able to think logically.
- ✓ They can communicate well : will be working as a team to develop the requirement modeling . So, they must have good communication skill.
- ✓ They are willing to invest their time in software development
- ✓ Most, but not all ,of the SME come from the user community : you must have real user of the system who know the business

2. Choosing Good facilitator : Good facilitator have the following quality

- ✓ They have good meeting skills:
- ✓ They understand the requirement modeling process: understand the entire software process, principally

Cont…

3. Choosing Good Scriber:

Good scriber have the following quality:

- They listen well: they listen and record business logics and rules,
- They have good written communication skills
- They have good oral communication skills
- They have an ear for business logic

Cont…

The main aim of this techniques:

- To broaden your understanding of the application
- To determine who to invite to requirements modeling sessions
- To identify new or existing requirements directly for your application

3. Requirement gathering techniques

a. Interviewing

- You usually start defining requirement by interviewing experts, either potential users of the application or people with expertise in technical domain.

Pointers for interviewing

- ✓ Send ahead agenda:
- ✓ Verify a few hours a head of time that interview is still on : because people schedule can change.
- ✓ Thank the person: when you first meet the person, thank for taking time out of busy day.

Pointers for interviewing

- ✓ Ask critical questions As Soon As Possible
- ✓ Ask for missing information or if he wants to add something
- ✓ Don’t Shut down users – “ I have already heard this before” don’t assume you know everything
- ✓ End the interview by summarizing the main points, this helps you to ensure you understand everything .
- ✓ Thank the person again at the end of the interview
- ✓ Listen than talking
- ✓ Inform the interviewee when you will summarize the interview notes and tell him you will send him copy for review. This

b. Brainstorming

- Technique where group of people discuss a topic and say anything that comes into their minds about it.
  - Rules
    - All ideas are good; not judged by the group
    - All ideas are owned by the group, not the individual
    - All ideas immediately become public property, anybody is allowed to expand upon them

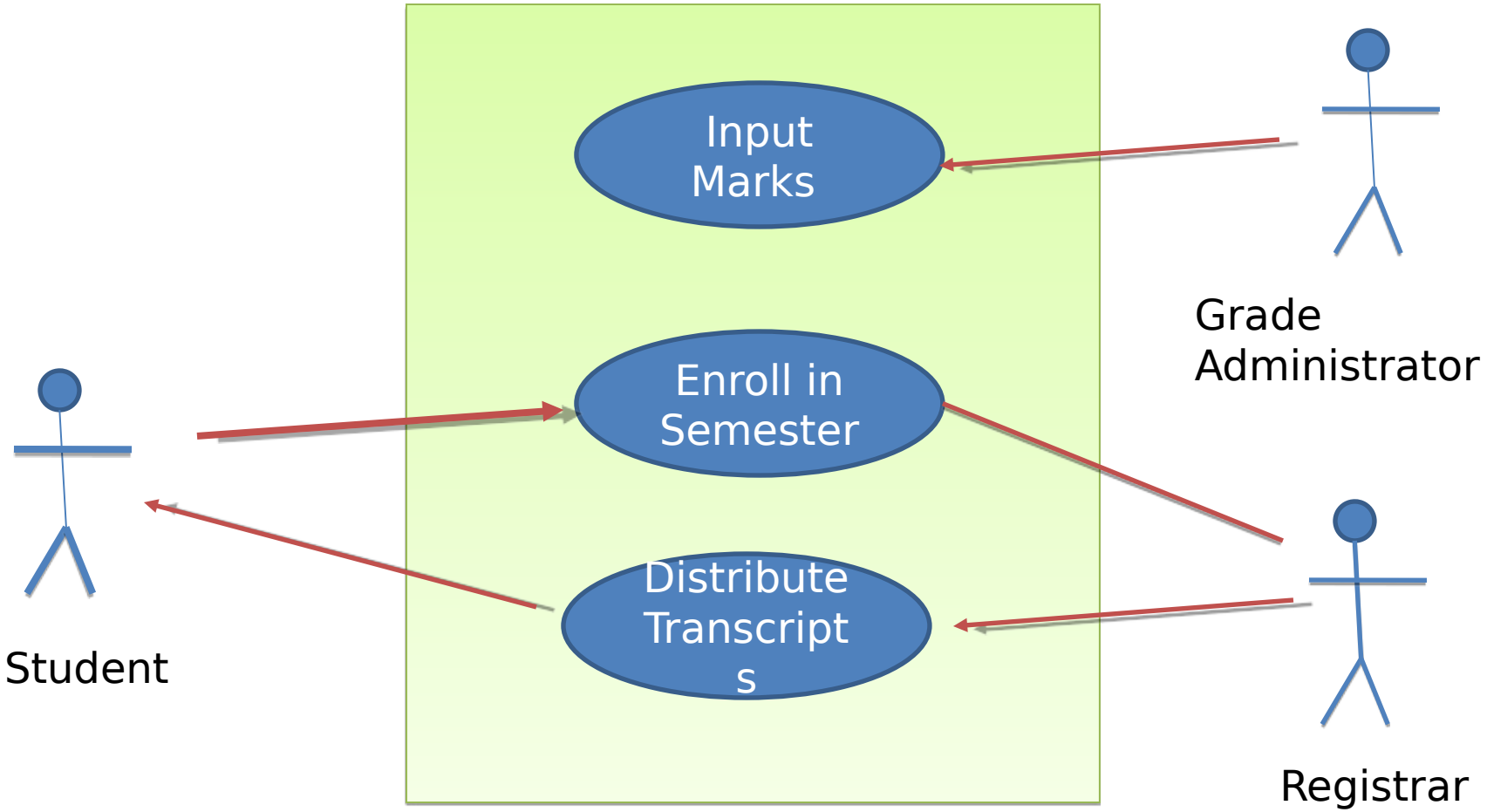
Potential Issues to discuss in Brainstorming

- Who is this system for?
- What will they do with the system?
- Why do we do this?
- Why do we do this the way we do?
- What business needs does this system support?
- What do/will our customers want/demand from us?
- How is the business changing
- What is our competition doing? Why? And how can we do it better?
- Do we need to do this?
- Will the system pay for itself?
- Is the work being performed where it makes most sense?
- How will peoples jobs be affected?
- How can we do this BFC?
- Etc…

4. Essential Use Case Modeling

- To model behavioral requirements
- It is one of the two basic flavors
  - Essential Use Case – Business/Abstract Use Case
  - System Use Case – Concrete/Detailed Use Case
- Intended to capture the essence of problems through technology free, idealized and abstract descriptions
- Flexible

Use Case Diagram for a Simple University



Drawing Use Case Diagrams

- Collection of use cases, actors, their associations, a system boundary box (Optional), and packages (Optional)

- ✓ Horizontal Ellipse – Use Cases
- ✓ Stick Figures – Actors – role players
- ✓ Line – Association and Relationship
- ✓ Arrow – Initial invocation (optional)
- ✓ Rectangle around use case – Scope of the system
- ✓ File Folders – Packages (organized model elements into groups)

Identifying Actors

- Anything or anyone that interfaces with your system
- Including people, external systems, and other organizations
- Never part of the system
- Questions help to find actors
  - Customer?
  - Who obtains information?
  - Who provides information?
  - Who installs the system?
  - Who operates the system?
  - Who shuts down the system?
  - What other system interacts with our system?
  - Does anything happen automatically at preset time?
  - Who will supply, use or remove information?
  - Where does the system get information?

Document Use Cases

- Sections
  - Name – purpose of the use case
  - Description – summary of a use case
  - Actors (optional) – associated with the use case
  - Precondition – conditions that must be met
  - Post condition – effect of a use case action
  - Basic Course of Action – logic followed
  - Alternate Course of Action – Infrequently used path
  - Change History (optional) – when, who, why modified use cases

Identifying Use Cases

- Answer the following questions
  - What are users 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?
  - 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 and inform the system about?
- Use Case Scenario – description of a potential business situation that may be faced by the users of a system

Essential Use Case Description Example

Name: Enroll in Semester

Description: Enroll an existing student in a seminar for which she is eligible

Preconditions: The Student is registered at the university

Post conditions: The Student will be enrolled in the course she wants if she is eligible and room is available

Basic course of Action:

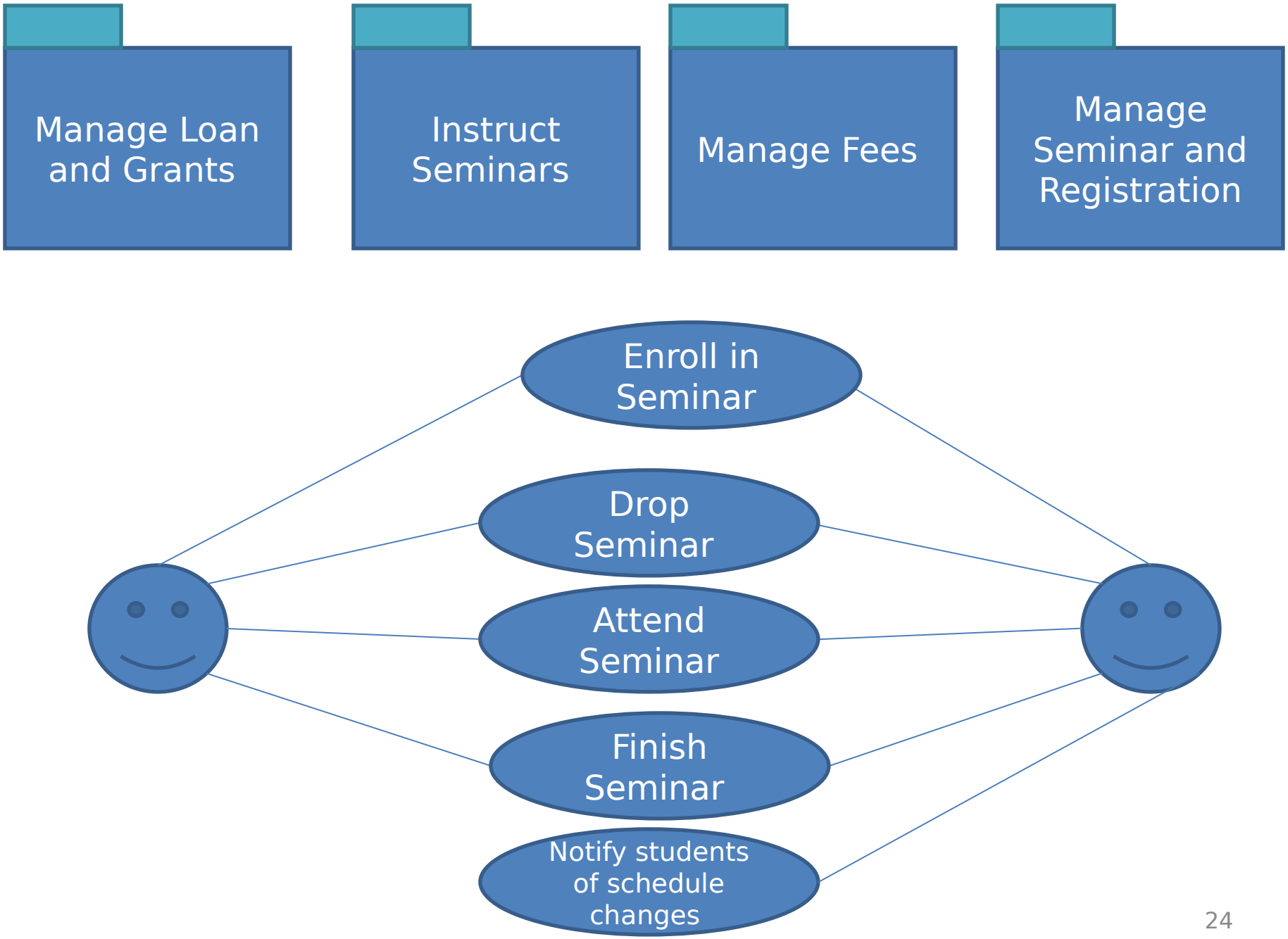
1. A Students wants to enroll in a seminar
2. The Student submits his name and student number to the registrar
3. The registrar verifies the student is eligible to enroll in semesters at the university according to some rule
4. The student indicates, from the list of available seminars, the seminar in which he wants to enroll
5. The registrar validates the student is eligible to enroll in the semester according to the business rule
6. The registrar validates the semester fits into the existing schedule of the student
7. The registrar calculates the fees for the seminar,
8. The registrar informs the student of the fees
9. The registrar verifies the student still wants to enroll in the seminar
10. The student indicates he wants to enroll in the seminar
11. The registrar enrolls the student in the seminar
12. The registrar adds the appropriate fees to the students bill
13. The registrar provides the student with a confirmation that he is enrolled

Things to know about Use Cases

- No time ordering is indicated between use cases
- Customer actors are usually involved in may use cases
- Use Cases are not functions
- No arrowheads are on the associations
- Should be functionally cohesive
- Should be temporally cohesive
- Every actor is involved with one use case and vice versa
- I chose not to include a system boundary box

UML packages

- Represents collection of use cases
- By introducing packages, the use case diagram become simpler and easier to understand
- Each package, in turn, would be documented by another use case diagram.


Manage Loan and Grants
Instruct Seminars
Manage Fees
Manage Seminar and Registration
Enroll in Seminar
Drop Seminar
Attend Seminar
Finish Seminar
Notify students of schedule changes

Alternate Course of Action

- Infrequently used path of logic in a use case
- Exception or error condition that must be handled
- Style issues
  1. Includes the descriptions of actions that must be met to invoke the alternate course
  2. Identification and numbering scheme is used
  3. Each step starts with the letter of the alternate course, followed by the number of the step in the basic course of the use case it replaces
  4. The last step in each alternate course of should indicate either that the use case ends or goes to the last step of the Basic Course of Action

Example of Courses of Actions

Alternate course A: The student is Not Eligible to Enroll in Seminars

A.3. The registrar determines that student is not Eligible to Enroll in Seminar

A.4 The registrar informs the student she is not eligible to enroll

A.5 The use case ends

Alternate Course B: The Student Does Not Have the Prerequisites

B.5 The registrar determines the student is not eligible to enroll in the seminar she chose

B.6 The registrar informs the student she does not have the prerequisite

B7. The registrar informs the student of the prerequisites she needs

B8. The use case continues at Step 4 in basic course of action

Alternate Course C: The student decides not to enroll in an

5. Essential User Interface Prototyping

- Place where user directly interacts with systems
- It represents the general ideas behind the UI, but not the exact details
- Represents user interface requirements in a technology-independent manner
- Beginning point of the user interface prototype for your system

Differentiating facts about essential UI prototyping

- Focus on your users and their usage of the system, not system features
- Simple tools including whiteboards, flip chart paper and sticky notes
- Don’t focus on the design
  - HTML, JAVA, Window Based

Tasks in Essential UI Prototyping

- Explore system usage
  - Use white boards to discuss initial drawings
  - Work iteratively
- Model major UI elements – flip chart paper
  - Potential screens and reports
- Model minor UI elements (input fields, lists and containers) – sticky notes
- Explore the usability of your UI
  - Learnable, increase user productivity, easy to remember, supportable
- Explore the relationship between UI elements
  - User interface flow diagramming

Example for essential UI

Student Information

Student Number

Student Name

Display Requester

Courses the students has taken or is currently taking

Help Requester

roll Students in Seminar

Course Searcher

Course Number

Course Name

Search Requester

Courses list

Details Requester

Course Details

Course Number

Course Name

Course Description

Professor

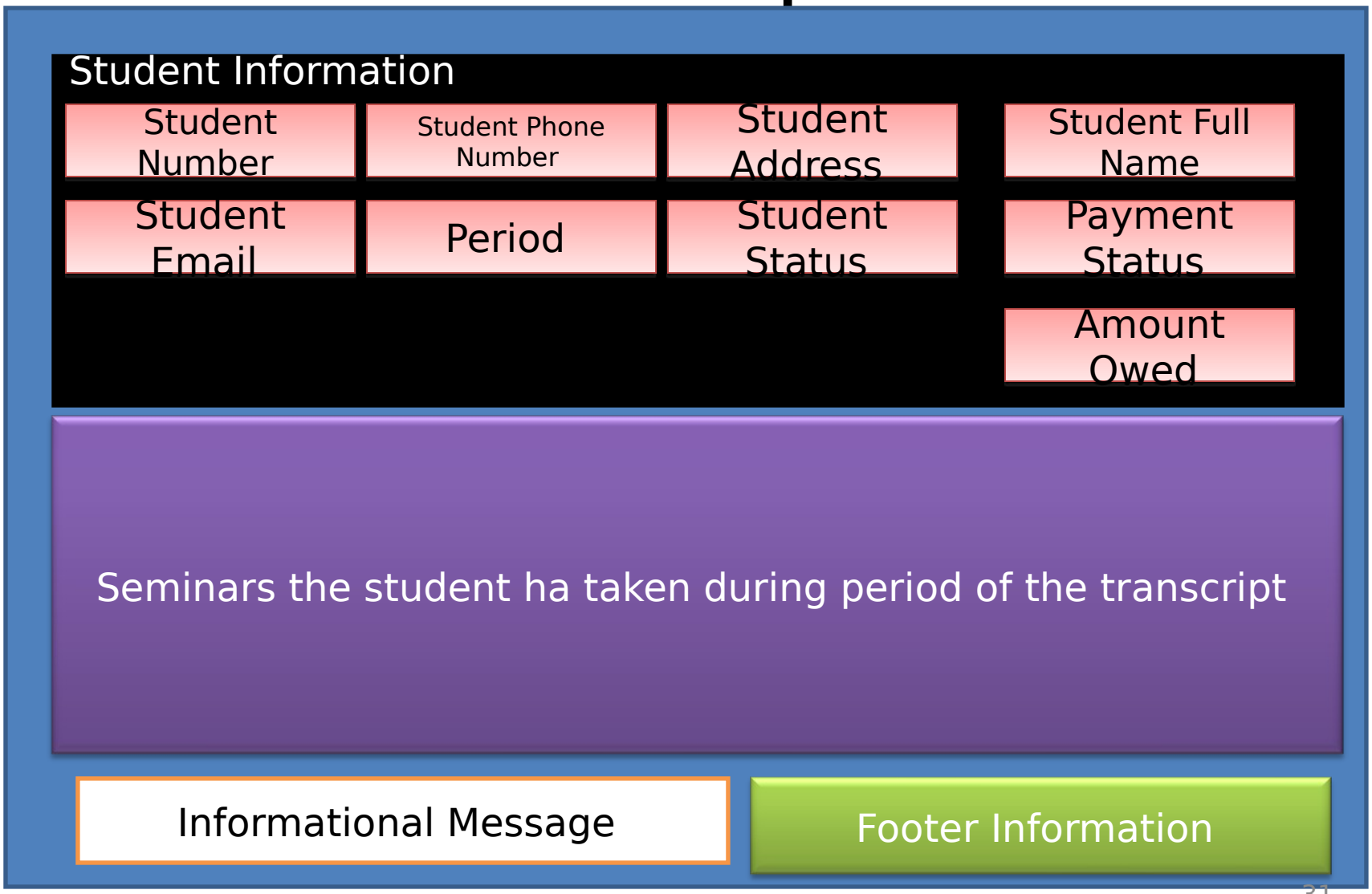
Seminar Location

Seminar Availability

Enrollment Requester

Prerequisite List

Prototype for a Student Transcript



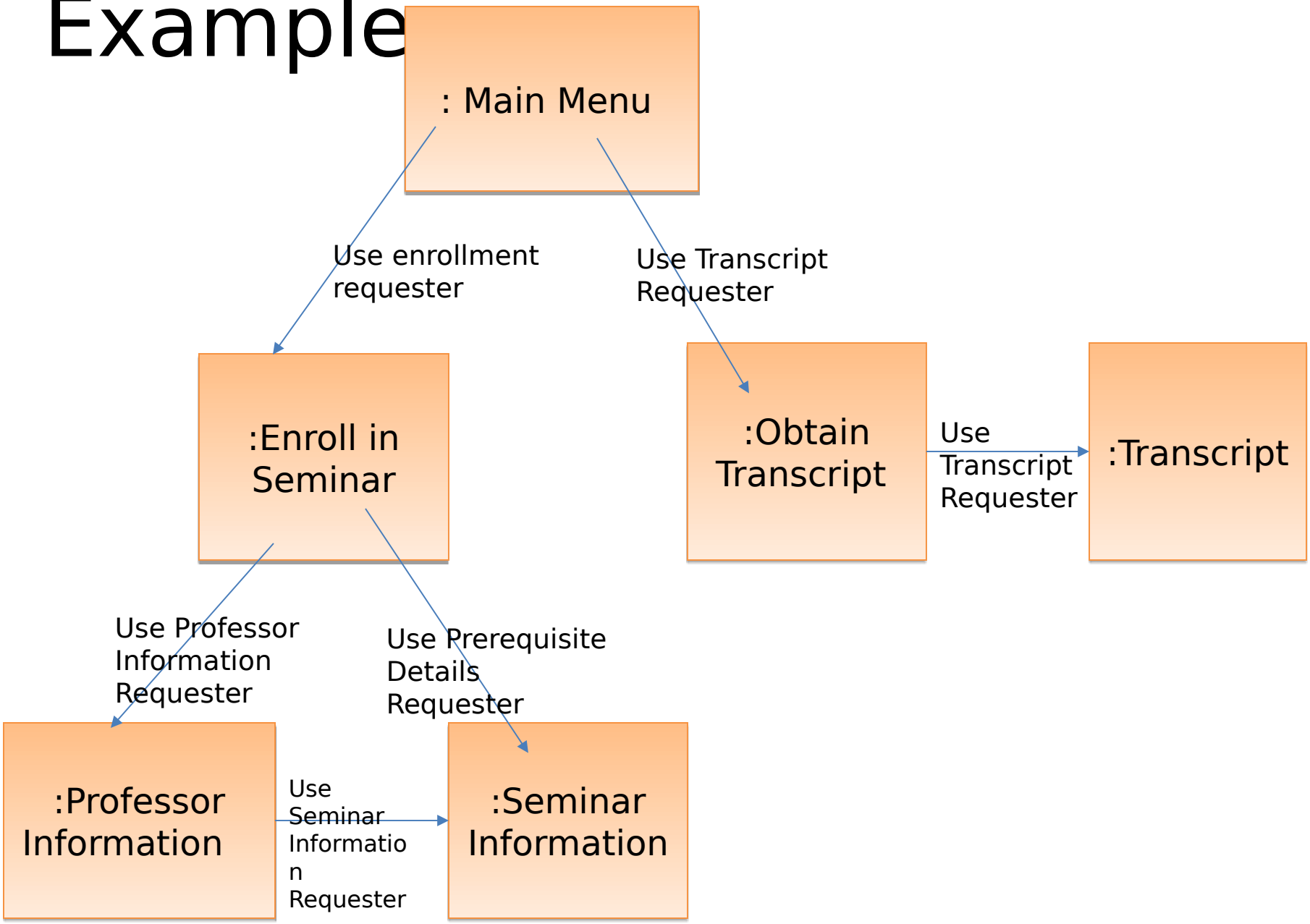
Ensuring System Usability

- Five factors affect usability
  - **Access** : A system should be usable with out help from a person
  - **Efficacy**: System not interfere with or impede use by a skilled worker
  - **Progression**: facilitating advancement in knowledge, skill and facility
  - **Support**: easy, simple, fast and fun
  - **Context** : Real condition and actual environment
- Usability is important because
  - By focusing on use and usability your system can be simpler, smaller and less expensive
  - Best systems give pleasure and satisfaction
  - Unusable systems lead to request for change
  - User are less patient with poorly developed

User Interface Flow Diagramming

- Helpful to see the bigger picture
- High level interaction between the UI elements of your application
- Also called windows navigation diagrams and context navigation maps
- Used for one of the two **purposes**
  - **Model the interactions** that users have with your software, as defined in a single use case.
  - **Gain a high level overview** of the UI of your application.
- Gain an understanding of how the system is expected to work

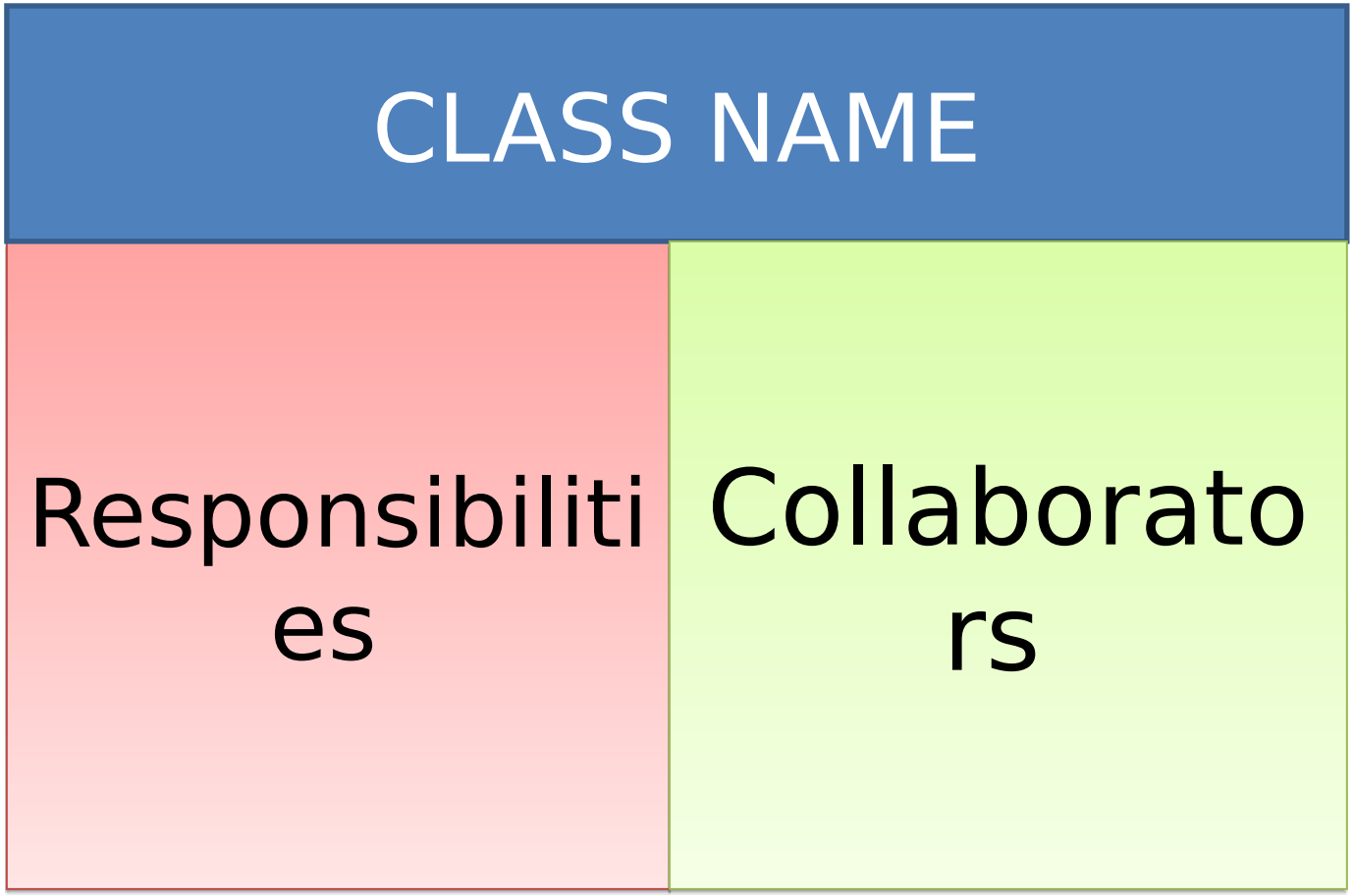
Example


: Main Menu
Use enrollment requester
Use Transcript Requester
:Enroll in Seminar
:Obtain Transcript
Use Transcript Requester
:Transcript
Use Professor Information Requester
Use Prerequisite Details Requester
:Professor Information
Use Seminar Information Requester
:Seminar Information

6. Domain Modeling with Class Responsibility Collaborator (CRC) Cards

- Task of discovering the classes that represent the things and concepts pertinent to your problem space
- A CRC model is a collection of standard index cards that have been divided into three sections
  - CLASS – collection of similar objects
  - RESPONSIBILITY – something that a class knows or does
  - COLLABORATOR - another class that a class interacts with to fulfill its responsibilities

Layout of a CRC card



Domain Modeling with CRC

- CRC models are well suited for domain modeling during requirement gathering
- UML class diagrams are better choice for domain modeling

CLASS

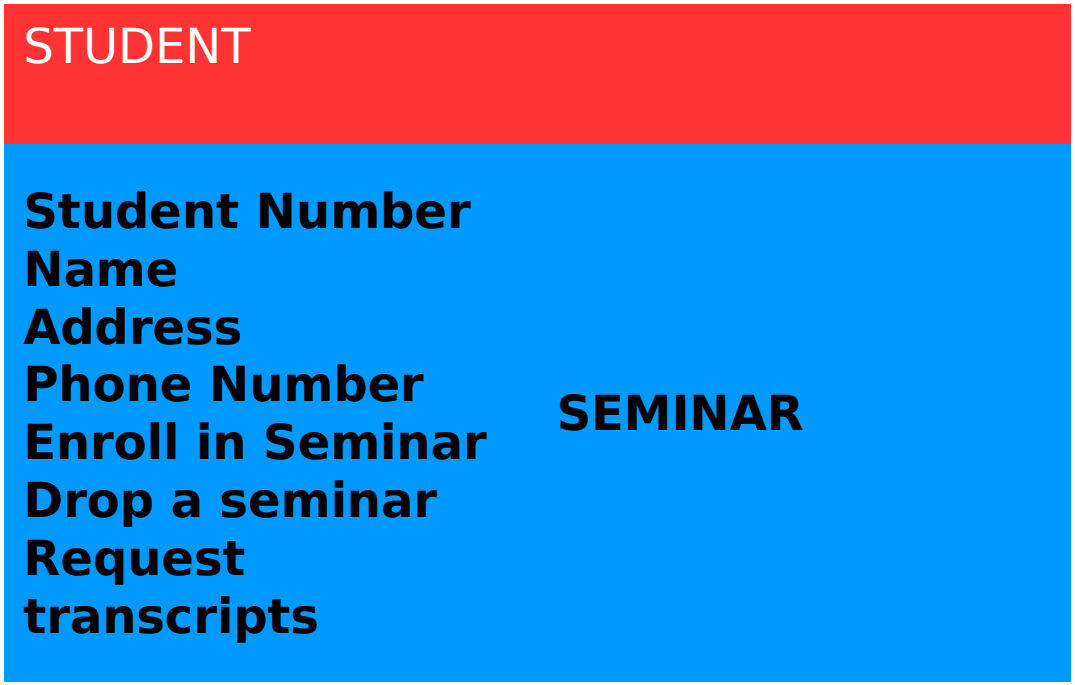
- Represents a collection of similar objects
- Person, Place, Thing, Event or Concept
- Eg. University System
  - Students, professors, seminars
- Singular noun or singular noun phrase
- Class names should also be simple

Responsibility

- Anything a class knows or does
- Eg. – Students have names, addresses and phone numbers. (KNOWS)
- Students also enroll in seminars, drop seminars and request transcripts (DOES)
- Classes update their own attributes and nobody else’s

Collaboration

- A class may not have enough information to do its responsibilities, need information from other classes
- Takes one of two forms
  - A request for information or
  - A request to do something.
- EG. – The card “Student” requests an indication from the card “Seminar”
  - A request for information
  - A request to do something


STUDENT
Student Number
Name
Address
Phone Number
Enroll in Seminar
Drop a seminar
Request transcripts
SEMINAR

Steps in Domain Modeling by CRC cards

1. Prepare to CRC Model
   - Put together a modeling group of SMEs
   - Brainstorm
   - Explain the CRC modeling technique

Iteratively Model CRC
   - Find classes
   - Find responsibilities
   - Define collaborators
   - Define use cases

Finding Classes

- Actors are potential classes
- Identify the customer
- Follow the money
  - Where the money come from?(customer)
  - Products and services being sold
  - Component that makes products and services being sold.
- Concepts and Events are potential classes
- Major UI elements are potential classes
- Look for three to five main classes right away
  - understand the core of the system
  - Eg. University IS:(Student, Course, Teacher, Seminar, Room)
- When you think you have identified a class, create a new card for it immediately
- You are interested in several types of classes

The three types of classes

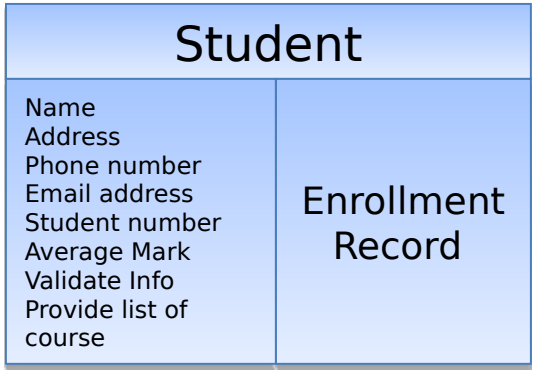
1. Actor Classes – actors that appear in your use case
   - People or organization
2. Business Classes – places, things, concepts, and events that describe what the business is all about
   - Concept – Course
   - Event – Registration
   - Place – Room
3. UI Classes – screens, menus, and reports
   - The Major UI elements. Eg. Student editing screen, registration page, student transcript

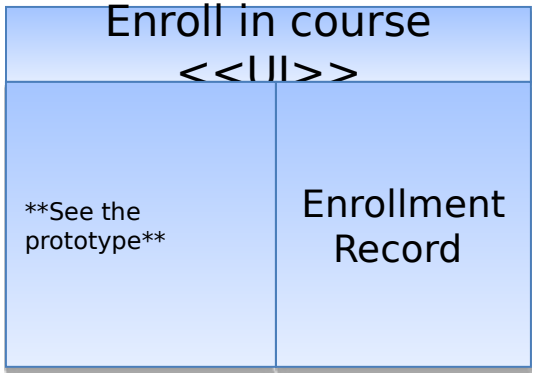
Finding Responsibilities

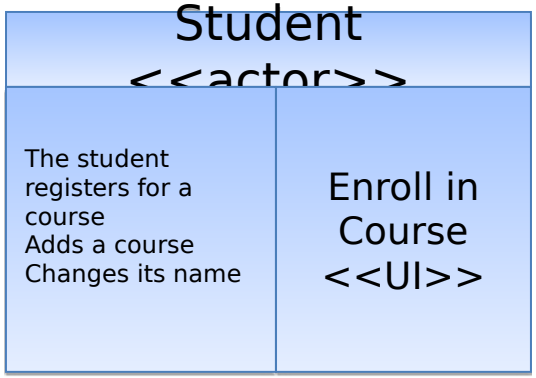
- Ask yourself what a class does
  - Actor Class – Identify the way in which the actor interacts with your system
  - UI Class – accept input from users, retrieve business objects, enable users to manipulate the business objects, support creation, update, deletion, and saving of business objects into the system
  - Business Class – derived from UI classes or other business classes (remember the context of your domain)
- Ask yourself what information you need to record about a class
  - Actor Class
  - UI Class

Defining Collaborators

- Collaboration must occur
  - when a class needs information it doesn’t have
  - When a class needs to modify information it doesn’t have because any given classes can update only the information it knows
- Things to consider
  - There will always be at least one initiator of any given collaboration
  - Sometimes the collaborator does the bulk of the work
  - New responsibilities may be created to fulfill the collaboration


Student
Name
Address
Phone number
Email address
Student number
Average Mark
Validate Info
Provide list of course
Enrollment Record


Enroll in course <<UI>>
See the prototype
Enrollment Record


Student <<actor>>
The student registers for a course
Adds a course
Changes its name
Enroll in Course <<UI>>

7. Supplementary Specification:

- Supplementary specification consists of three things:

1. Business Rule 2. Constraints 3. Non functional Requirements

1. **Business rules often pertain to**
   - access control issues:
     - Professors are allowed to input and modify the marks of the students taking the seminars they instruct
   - business calculations or operating policies :
     - Converting a percentage mark that a student receives in a course to a letter grade
   - principle of your organization:
     - Expel any student who falls more than two courses in the same semester

How Each business rule has identified?

- By Use the format of BR# but you are free to set your own numbering approach
- Enables to refer easily to business rules in other development artifices (class models or use cases)

Supplementary Specification....

Examples

- BR123 Tenured professors may administer student grades
- BR124 Teaching assistants who have been granted authority by a tenured professor may administer student grades
- BR177 Table to convert between numeric grades and letter grades

Supplementary Specification....

Examples:

A fully documented business rule

Name: Tenured professors may administer student grades

Identifier: BR123

Description : Only tenure professors are granted the ability to initially input, modify and delete grade of students receive in the seminar that they only instruct

Example Dr. Rahel, instructor of Database Systems (INSY XXX) may administer the marks of all students enrolled in that course but not those enrolled in "Introduction to Internet Programming (INSY 332)

Source : University Policies and Procedures
Doc ID: U1701
Publication date: August 14, 2010

Related rules BR12 Qualifying to teach
BR200 Modifying Final Student Grades

Revision History: Defined March 2, 2013 by Mr. X
Updated October 10, 2013 by Mr. Y

Supplementary Specification....

2. Identifying Non functional Requirements

- Non functional requirement pertains to the technical aspect or system must fulfill, such as:
  - ✓ Performance related issue
  - ✓ Reliability issue
  - ✓ Availability issue
- use format TR#, where TR stands for technical requirements.
- you documented or summarized non functional(technical) requirement in a similar manner as business rules includes:
- Name, Description, An example, Sources, Reference to related TR and revision history
- **Example**:
- TR34: the system shall be unavailable for no less than two minutes in a 24 hours period.
- TR 78: A seminar search will occur with in less than 3 seconds 95% of the time & no more than 10 seconds 99% of the time .

3. Constraints: is a restriction on the degree of freedom you have in providing a solution . constraints can be Economic , Time,

Identifying Change case

Change case should be developed as part of your overall modeling efforts particularly during requirements gathering but also during analysis and design .

Change case is the result of brainstorming by your SME, where questions such as:

- ✓ How can the business change
- ✓ what technology can change
- ✓ What legislation can change
- ✓ What is your computation doing
- ✓ What system will we need to interact with
- ✓ Who else might use the system & how are explored

Documenting change case:

Change case :The university will open a new campus

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

Impact: Large students will be able to register in class at either campus. Some instructor will teach in both campus. Some department such as CS,MIS are

Chapter 4

Ensuring Your Requirements are Correct:

Requirements Validation techniques

Introduction

- ✓ It is not enough to gather requirements; you also need to verify that they are correct
- ✓ A wide range of experience combined with unambiguous models, reduces the chance of misunderstood requirements
- ✓ The most common error made in the dev't of a system are analysis errors: user requirements that are either missing or misunderstanding
- ✓ In many ways developers are good at building system correctly and we are very good at technical aspects of the software dev't. but we are not good at building the right system b/s we are not good at non technical aspects of software dev't such as requirement gathering and analysis.

- What is the implication of analysis error for object oriented development artifacts ?

  With respect to CRC model.

  - ✓ You would be missing user requirements if you are missing few responsibilities for a class, attributes of class or one or more collaborators .
  - ✓ For your essential user interface prototype you would have incomplete screen and report or

- Misunderstood by user requirement are also serious issue.
- Requirement can be misunderstood in three ways:
  - ✓by user,
  - ✓by analyst
  - ✓by designer
- The most serious pbm is when user misunderstood a requirement. This is why when you build a requirement team you want SME from a variety of background. so, you reduce the chance of this sort of misunderstanding.
- You can reduce the risk of both analysis and designers misunderstanding requirements by producing model that is simple and unambiguous.

4.1 Testing Early and often

- If you make an analysis error and find it during:

✓ Requirement: it is inexpensive to fix. You merrily change a portion of your requirement model.

✓ If you do n't find until Design stage :it is more expensive to fix not only you have to change your analysis you also have to evaluate and potentially modify the section of your design based on the analysis faulty analysis.

- ✓ If you don’t find until programming: you need to update **analysis**, **design** and scrap potion of **your code** all b/s of missed or misunderstood user requirements.
- ✓ So, how can you test your requirement model?
- To test your requirement models you can apply one or more of the following techniques
  - Use case scenario testing
  - User Interface walkthroughs

4.2 Use Case Scenario Testing

- Used to test domain models,(either CRC or class model) if it accurately reflects the aspects of the business that you are modeling
- Steps
  1. Perform domain modeling: create either CRC or Class modeling
  2. Create the use case scenarios
  3. Assign classes to your SMEs: one or more CRC model
  4. Describe how to act out a scenario:
  5. Act out the scenarios
  6. Update the domain model
  7. Save the Scenarios

Example:

Name: A student successfully enroll in several seminars and pay partial tuition for them:

Description: a student decides to register in three seminars w/c the student has the prerequisites for w/c still have seats available in them and pay half the tuition at the time of registration.

Steps:

the student prepares to register

A student determines the three

Create Use Case Scenarios

- Conceptually similar to use case: A use case describes the logic including the basic and alternate courses of action, for a single cohesive task
- A use case scenario describes a single path of logic through one or more use cases
- When you describe a use case scenario, you want to give it a name and short description and then a description of the steps to take to fulfill the scenario.
- Helps to catch unusual scenarios
- You can identify new scenarios in several ways
- Consider tasks the system should and shouldn’t be able to handle
  - Explore business rules
  - Do some more brainstorming

Acting Out Scenarios

Steps

1. Call out a new scenario : the facilitator reads the scenario description to the group, the group must decided if this scenario is reasonable(remember: the system can n't handle some scenasrio)
2. Determine which card should handle the responsibility
3. Update the cards whenever necessary
4. Describe the processing logic
5. Collaborate if necessary
6. Pass the ball back when done

Advantages of use Case Scenario Testing

- Use case scenario testing helps you to find and fix analysis errors inexpensively
- Use case Scenario testing provides you with a detailed descriptions of the business logic of the system
- Use case scenario testing is simple and it works
- Scenarios help to define how people interact with the system

- Your SMEs must make the time to do the testing
- Managers often feel real work isn’t being accomplished
- Developers are often skeptical (doubtful)

User Interface Walkthroughs

Similar to use case scenario testing sessions, the only difference being your system’s user interface is being tested instead of your domain model

Requirement Reviews

- The requirement team prepares for review
- The team indicates it is ready for review
- The review facilitator performs a cursory review
- The review facilitator plans and organizes the review
- The reviewers review the package prior to the review
- The review takes place
- The review results are acted on

The End