

ECS505U

SOFTWARE ENGINEERING

MUSTAFA BOZKURT

LECTURER IN SOFTWARE ENGINEERING

Week 6

Domain, Requirements and Use-case Analysis

LESSON OBJECTIVES

- Learn why to perform domain analysis
- Better understanding of software requirements
- Learn different scopes in software projects
- Better understanding of use-case analysis

WHAT IS A DOMAIN?

Berard¹ gives two characterizations:

A collection of current and future (software) applications that share a set of common characteristics.

A well-defined set of characteristics that accurately, narrowly, and completely describe a family of problems for which computer application solutions are being, and will be sought.

1. Berard, E. , Essays on Object-Oriented Software Engineering, Prentice Hall, 1992

WHAT IS DOMAIN ANALYSIS?

A process by which information used in developing software systems is identified, captured, and organized.

WHAT IS DOMAIN ANALYSIS?

The word 'domain' in this case means the general field of business or technology in which the customers expect to be using the software.

WHY IS DOMAIN ANALYSIS IMPORTANT?

- Faster development
- Better system
- Anticipation of extensions
- Reuse

WHY IS DOMAIN ANALYSIS IMPORTANT?

Founder of domain analysis is James Neighbors said:

The key to reusable software is captured in domain analysis in that it stresses the reusability of analysis and design, not code.

DOCUMENTING DOMAIN ANALYSIS

- Introduction
- Glossary
- General knowledge about the domain
- Customers and users
- Environment
- Tasks and procedures currently performed.
- Competing software
- Similarities across domains and organizations

Talk about requirements gathering from Google play

DOMAIN ANALYSIS: REUSE

If similarities across domains and organizations identified successfully we can build:

- The new system/features based on existing ones.
- A more reusable software.

DOMAIN ANALYSIS DOCUMENTATION: EXAMPLE

Introduction:

This document describes background information that has been gathered about events in organizations and how they are handled. This information is to be used to guide the development of software to automate the process of informing people of events.

DOMAIN ANALYSIS DOCUMENTATION: EXAMPLE

Glossary:

Event: A meeting, a social occasion or an activity involving a significant number of employees. Several categories of events have been identified:

- **Open event:** An event that starts at a precise instant but with no predetermined duration. Meetings and celebrations often fall into this category.
- **Fixed event:** An event that starts at a precise instant and with a predetermined duration. Course lectures and seminars are examples of this kind of event.
- **Day events:** An event associated with a particular day without precise start and end times. Birthday, thematic journey are such events.
 - **Recurrent event:** An event that occurs repeatedly on some regular schedule (for example daily, weekly or monthly). The event normally has a starting date and an ending date. Courses and social activities are often recurrent events.
 - **Composite event:** An event composed of several sub-events. For example, a training activity can be composed of a registration period (fixed event), a series of seminars (recurrent events), and a final evening celebration (open event).

DOMAIN ANALYSIS

DOCUMENTATION: EXAMPLE

General knowledge about the domain:

- Most events occur during working days.
- Events are generally associated with a location (where the event is to be held).
- The name of a contact person is often associated with an event. That person is the one that organize the event or that can give complementary information about the event.
- Group of interest are often created to target more precisely peoples that might be interested by a certain event.
- Outdated events are of little interest.
- Each event has a title, a location and the name of a contact person associated with it.
- Events may be seen by anyone within the organization, but there should be some control over posting events to reduce the risk of duplicate postings and other clutter.

DOMAIN ANALYSIS

DOCUMENTATION: EXAMPLE

Clients and users:

Potential clients are medium or large companies whose staff use computers to perform many kinds of daily work. Others impacted by the system will include:

- Employees at all levels have an interest in events and are potential users of event manager software. These employees range from computer novices to sophisticated programmers; however they all have a computer on their desk, have access to the corporate intranet and know how to use a web browser. They have been exposed to and accept new technology but are subject to tight time constraints and have little time to learn and customize new software tools.
- A system administrator normally manages the computer environment.
- Technicians typically install software that must be available to all users.

DOMAIN ANALYSIS

DOCUMENTATION: EXAMPLE

The environment:

The actors all have a computer on their desks; it is most common for this to be MS-Windows based, but a significant minority of potential clients use other platforms.

A wide variety of software is installed on these computers, with each actor having a unique configuration. Some software may be installed on every computer using a site-wide license.

DOMAIN ANALYSIS DOCUMENTATION: EXAMPLE

Tasks and procedures currently performed:

Informing others about events: The organizer of the event (or someone who has heard about it) compiles information concerning an event and posts it so members of the organization can see it.

- One approach is to post the event on a physical bulletin board at a place that most users pass each day. In a large organization, this method is too unreliable due to the sheer volume of events.
- Another approach is to send email and paper leaflets to all members of the staff without knowing exactly who will be interested. In some cases there are mailing lists to make event notification more selective, however mailing lists are often poorly maintained, so some people who might be interested never find out about an event.
- In either approach, there is typically no central listing of available events and the information about an event is not presented in a standard fashion.
- It can be useful to allow anybody to post and remove events, to ensure the information is current. However this can lead to duplication and inconsistency.

Informing oneself about events. At irregular intervals, employees check the list of upcoming events and seek more information about ones of interests.

- When interested in an event, a user must check to see whether he or she has a schedule conflict. This is commonly done by maintaining a calendar. However there is a duplication of effort since all users have to keep their own records.

DOMAIN ANALYSIS DOCUMENTATION: EXAMPLE

Competing software:

Several software tools exist that can manage events. However, since these are usually included as part of a larger system (e.g. agenda software, email software or teamwork software) they are quite complex to use. Hence they are not generally adopted by the entire staff.

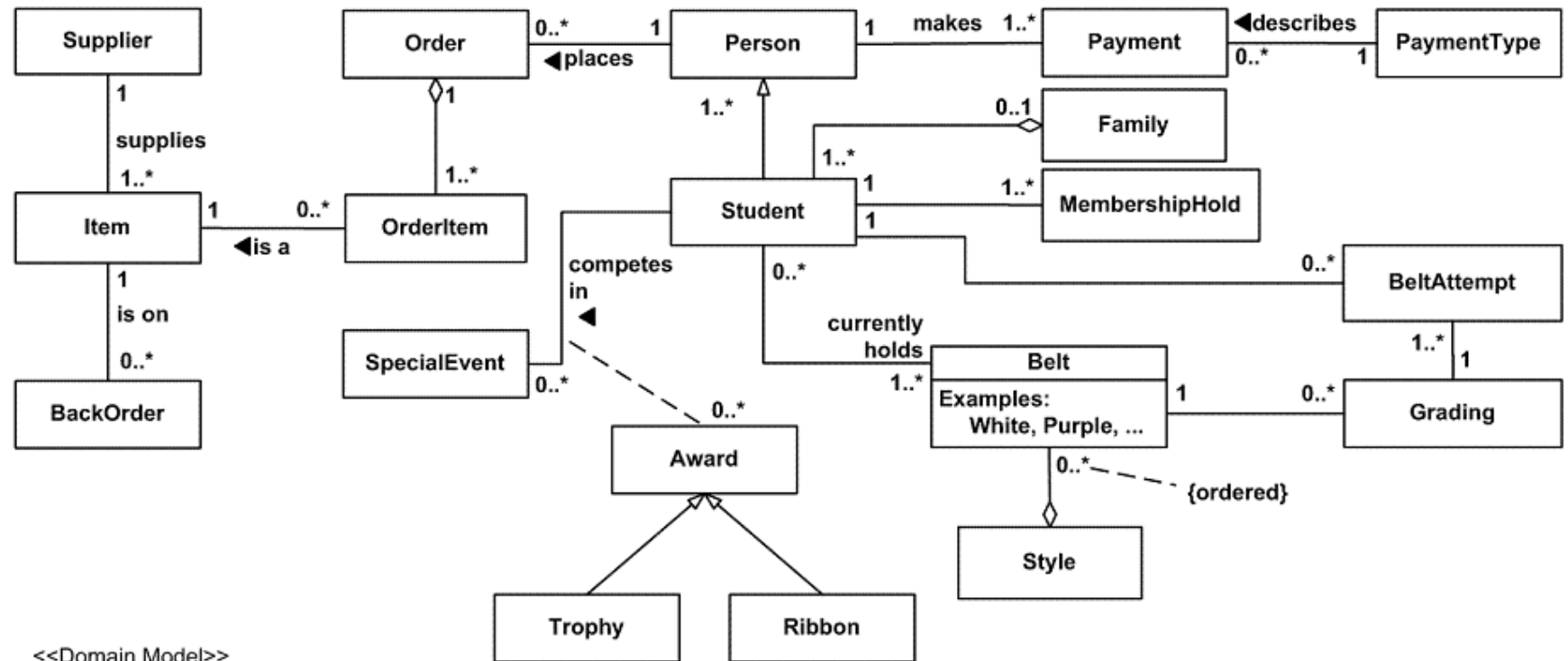
These applications allow you to check for the availability of rooms and to book them. They might also maintain the personal schedules of staff members such that it is possible to find the best schedule for a given event.

DOMAIN ANALYSIS DOCUMENTATION: EXAMPLE

Similarities to other domains:

There is a need to share many types of information in any organization, for example the telephone and office numbers of employees. In most cases, updating this information is the responsibility of one designated employee. This is also very similar to personal agenda, except that in the present case, only events of interest to everyone are added to the system.

DOMAIN MODEL



<<Domain Model>>

Copyright 2004 Scott W. Ambler

PROJECT/PRODUCT SCOPE

Scope involves getting information required to start a project, and the features the product would have that would meet its stakeholders requirements.

PROJECT/PRODUCT SCOPE

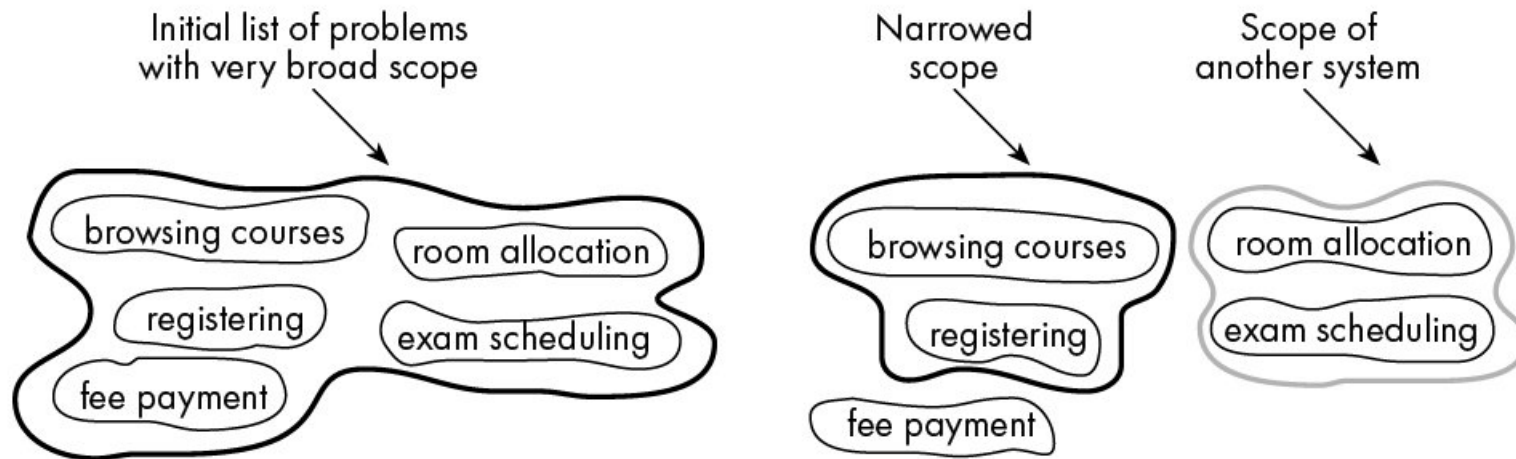
Project Scope

"The work that needs to be accomplished to deliver a product, service, or result with the specified features and functions."

Product Scope

"The features and functions that characterize a product, service, or result."

DEFINING SCOPE



Narrow the scope considering the high-level goal(s)

WHY DEFINING SCOPE IS IMPORTANT?

It's important to define the problem and scope early to avoid having unnecessary requirements for the system.

DEFINING SCOPE

Beware feature creep!

(a.k.a. requirements creep or scope creep)

WHAT IS A REQUIREMENT?

A requirement is a statement describing either:

1. An aspect of what the proposed system must do.
2. A constraint on the system's development.

WHAT IS A REQUIREMENT?

A requirement is:

- A statement.
- An aspect of what the proposed system must do.
- A constraint on the system's development.
- Contributes towards adequately solving the customer's problem.
- A negotiated agreement among all stakeholders.

TYPES OF REQUIREMENTS

Two main categories:

- Functional
- Non-functional

FUNCTIONAL REQUIREMENTS

Functional requirements:

- Everything that a user of the system would need to know regarding what the system does
- Everything that would concern any other system that has to interface to this system.

After specifying a cooking method, the user must press 'START' to initiate cooking (input).

FUNCTIONAL REQUIREMENTS

1. The system can be in the following modes (*conditions under which input and output can occur*):

- ❑ 'idle': this is entered when the system is switched on, when cooking is complete or when 'CANCEL' is pressed. This mode is exited when the system starts accepting input.
- ❑ 'accepting input': this is entered if the system was in idle mode and the user presses any button, except 'CANCEL' and 'START'. This mode is exited when the system enters 'cooking' mode, the user presses 'cancel', or the user completes the process of setting the time of day.
- ❑ 'cooking': this is entered if the door is closed, while the system is in 'suspended' or 'accepting input' mode, and the user then presses 'start'. This is exited when the user opens the door or presses 'cancel'.
- ❑ 'suspended': this is entered if the user opens the door while cooking. This mode is exited when the user presses 'cancel'; or closes the door and then presses 'start'.

Textbook Section 4.5, Page 120

NON-FUNCTIONAL REQUIREMENTS

1. Accessibility
2. Audit and control
3. Availability (see service level agreement)
4. Backup
5. Capacity, current and forecast
6. Certification
7. Compliance
8. Configuration management
9. Dependency on other parties
10. Deployment
11. Documentation
12. Disaster recovery
13. Efficiency (resource consumption for given load)
14. Effectiveness (resulting performance in relation to effort)
15. Emotional factors (like fun or absorbing)
16. Environmental protection
17. Escrow
18. Exploitability
19. Extensibility (adding features, and carry-forward of customizations at next major version upgrade)
20. Failure management
21. Fault tolerance (e.g. Operational System Monitoring, Measuring, and Management)
22. Legal and licensing issues or patent-infringement-avoidability
23. Interoperability
24. Maintainability
25. Modifiability
26. Network topology
27. Open source
28. Operability
29. Performance / response time (performance engineering)
30. Platform compatibility
31. Price
32. Privacy
33. Portability
34. Quality (e.g. faults discovered, faults delivered, fault removal efficacy)
35. Recovery / recoverability (e.g. mean time to recovery - MTTR)
36. Reliability (e.g. mean time between failures - MTBF)
37. Reporting
38. Resilience
39. Resource constraints (processor speed, memory, disk space, network bandwidth, etc.)
40. Response time
41. Robustness
42. Safety or Factor of safety
43. Scalability (horizontal, vertical)
44. Security
45. Software, tools, standards etc. Compatibility
46. Stability
47. Supportability
48. Testability
49. Usability by target user community

NON-FUNCTIONAL REQUIREMENTS

According to the book three types of non-functional requirements:

- Quality
- Platform
- Process

QUALITY REQUIREMENTS

Quality requirements concerned with:

- Usability
- Efficiency
- Reliability
- Maintainability
- Reusability

QUALITY REQUIREMENTS

Main categories of quality requirements:

- Response time
- Throughput
- Resource usage
- Reliability
- Availability
- Recovery (from failure)
- Maintainability and extendibility
- Reusability

PLATFORM REQUIREMENTS

Main categories of platform requirements:

- Computing platform
- Technology to be used

PROCESS REQUIREMENTS

Main categories of process requirements:

- Development process (methodology) to be used
- Cost and delivery date

REQUIREMENTS EXERCISE

What information must be stored in the database that travel agents and others access. **Answer**

The system should be designed such that it can be extended to handle a frequent-flier plan. **Answer**.

The system must be available at all times. Only 2 minutes' downtime a week is to be permitted. **Answer**

The system must run on any Linux system. **Answer**

A merge-sort algorithm must be used to sort the flights by departure time.

Answer

CLASSIFYING REQUIREMENTS

Functional and non-functional requirements often easy to identify and classify however **not in all cases**.

USE-CASE ANALYSIS

Use-case analysis is a systematic approach to working **out what users should be able to do** with the software you are developing.

It is one of the key activities in requirements analysis.

WHAT IS A USE CASE?

According to UML Reference Manual:

“The specification of sequences of actions that a system, subsystem, or class can perform by interacting with outside actors”

A use case is a typical sequence of actions that an actor performs in order to complete a given task.

WHAT IS THE PURPOSE OF A USE CASE?

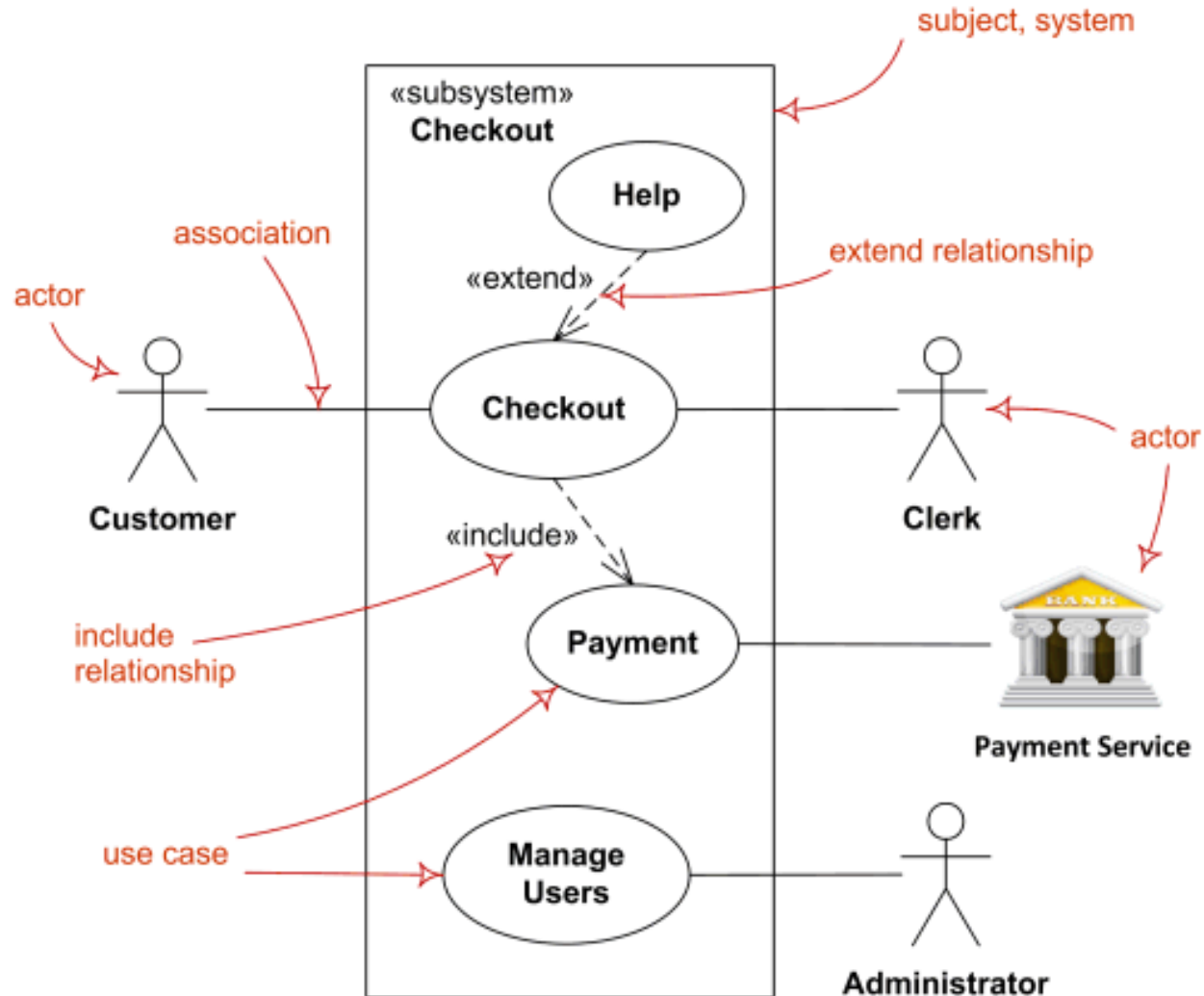
According to UML Reference Manual:

“To define a piece of behavior of a [system or subsystem or class] without revealing the internal structure of the [system]”

USE-CASE ANALYSIS

- Determine types of users
- Determine the tasks for each user

USE-CASE MODEL



USE-CASE MODEL

A use-case model consists of a set of use cases. Use cases often include the following elements:

- Name
- Actors
- Goals
- Preconditions
- Summary
- Related use cases
- Steps
- Post-conditions

USE-CASE MODEL

1 Brief Description

This use case describes how the Bank Customer uses the ATM to withdraw money to his/her bank account.

2 Actors

2.1 Bank Customer

2.2 Bank

3 Preconditions

There is an active network connection to the Bank.

The ATM has cash available.

4 Basic Flow of Events

1. The use case begins when Bank Customer inserts their Bank Card.
2. Use Case: Validate User is performed.
3. The ATM displays the different alternatives that are available on this unit. [See Supporting Requirement SR-xxx for list of alternatives]. In this case the Bank Customer always selects "Withdraw Cash".
4. The ATM prompts for an account. See Supporting Requirement SR-yyy for account types that shall be supported.
5. The Bank Customer selects an account.
6. The ATM prompts for an amount.
7. The Bank Customer enters an amount.
8. Card ID, PIN, amount and account is sent to Bank as a transaction. The Bank Consortium replies with a go/no go reply telling if the transaction is ok.
9. Then money is dispensed.
10. The Bank Card is returned.
11. The receipt is printed.
12. The use case ends successfully.

USE-CASE MODEL

5 Alternative Flows

5.1 Invalid User

If in step 2 of the basic flow Bank Customer the use case: Validate User does not complete this successfully, then

1. The use case ends with a failure condition.

5.2 Wrong account

If in step 8 of the basic flow the account selected by the Bank Customer is not associated with this bank card, then

1. The ATM shall display the message "Invalid Account – please try again".
2. The use case resumes at step 4.



5.3 Wrong amount

If in step 7 in the basic flow, the Bank Customer enters an amount that can't be 'created' with the kind of in the ATM (See Special Requirement WC-1 for valid amounts), then

1. The ATM shall display a the message indicating that the amount must be a multiple of the bills on hand, and ask the Bank Customer to reenter the amount.
2. The use case resumes at step 7.

5.4 Amount Exceeds Withdrawal Limit

If in step 7 in the basic flow, the Bank Customer enters an amount that exceeds the withdrawal limit (See Special Requirement WC-2 for maximum amount), then

1. the ATM shall display a warning message, and ask the Bank Customer to reenter the amount
2. The use case resumes at step 7

USE-CASE MODEL

6 Key Scenarios

6.1 No Response from Bank

7 Post-conditions

7.1 Successful Completion

The user has received their cash and the internal logs have been updated.

7.2 Failure Condition

The logs have been updated accordingly.

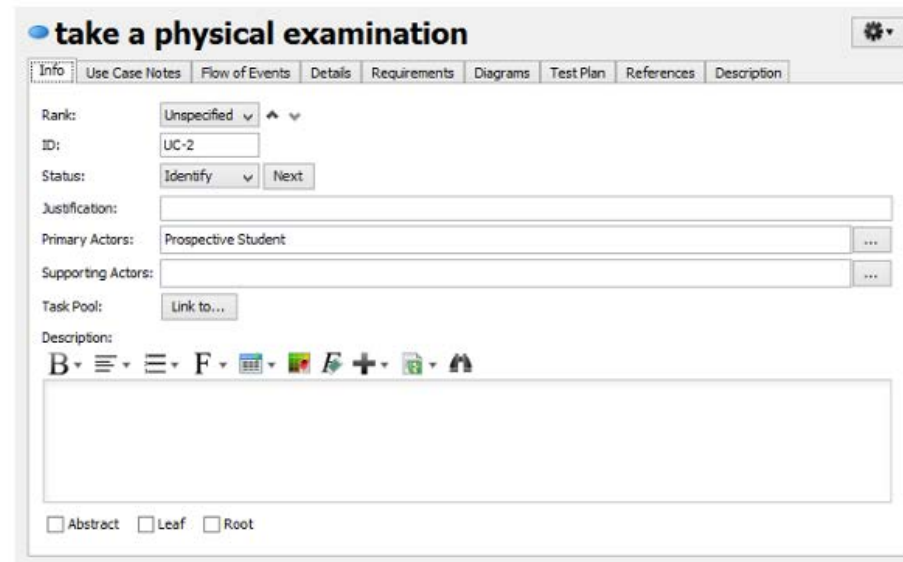
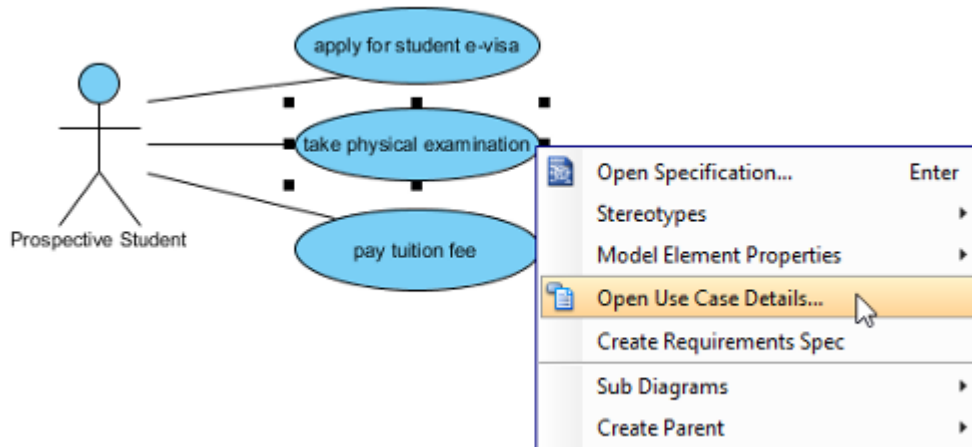
8 Special Requirements

[SpReq:WC-1] The ATM shall dispense cash in multiples of \$20.

[SpReq2:WC-2] The maximum individual withdrawal is \$500.

[SpReq:WC-1] The ATM shall keep a log, including date and time, of all complete and incomplete transactions with the Bank.

USE-CASE MODEL WITH VISUAL PARADIGM



A screenshot of the "take a physical examination" use case details window. The window has a title bar with a gear icon and a tabbed interface with the following tabs: Info, Use Case Notes, Flow of Events, Details, Requirements, Diagrams, Test Plan, References, and Description. The "Info" tab is active, showing the following fields:

- Rank: Unspecified
- ID: UC-2
- Status: Identify (with a Next button)
- Justification:
- Primary Actors: Prospective Student
- Supporting Actors:
- Task Pool: Link to...
- Description: A large text area with a toolbar containing icons for bold, italic, underline, font color, background color, bulleted list, numbered list, link, unlink, and help.

At the bottom, there are checkboxes for "Abstract", "Leaf", and "Root".

USE-CASE MODEL WITH VISUAL PARADIGM

take a physical examination

Info Use Case Notes Flow of Events Details Requirements Diagrams Test Plan References Description

Rank: Unspecified
ID: UC-2
Status: Identify Next
Justification:
Primary Actors: Prospective Student
Supporting Actors:
Task Pool: Link to...
Description:
B. F. + .
☐ Abstract ☐ Leaf ☐ Root

Info Use Case Notes Flow of Events Details Requirements Diagrams Test Plan References Description

Scenario
1.Fill in physical examination form
1.1.Download the physical...

1. Fill in physical examination form
1.1. Download the physical examination from the student visa website
1.2. Double check filled information
1.3. Make a cheque for medical fee
2. Call a physician to reserve an appointment
3. Arrive at the clinic at the scheduled time
4. Follow the instruction of physician and X-ray physician
4.1. Perform general physical examination
4.2. Perform detailed physical examination
4.3. Perform urine test
4.4. Perform X-ray check

Extension:
3.a. For those who are late for physical examination
1. Reschedule appointment with the physician
2. Make an appointment with another physician
<Type here to add an extension without any parent>

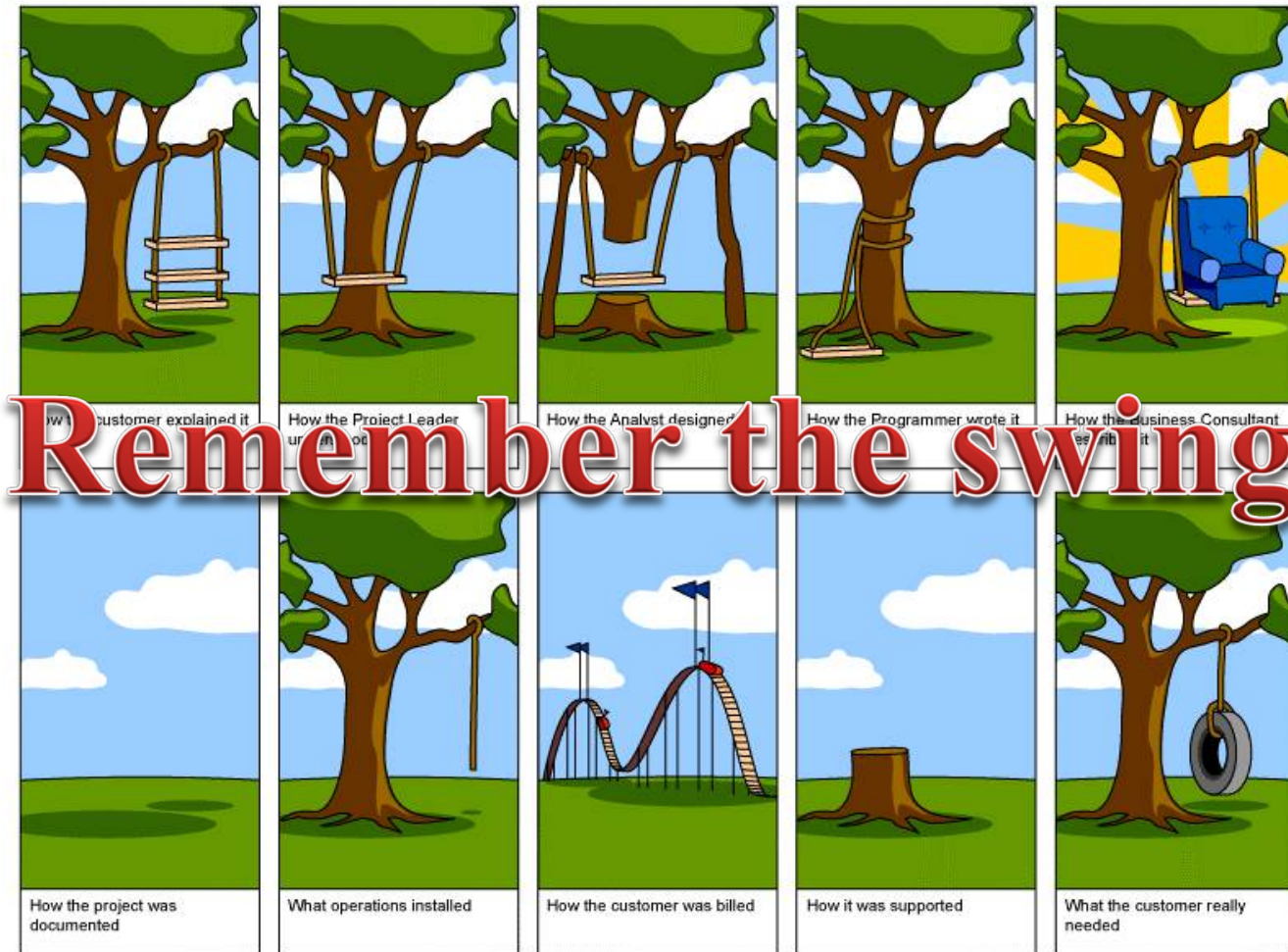
USE-CASE ANALYSIS: BENEFITS

- Define scope of the system
- Plan development process (use cases might indicate size of project)
- Help develop and validate requirements.
- Define test cases.
- Help structure user manuals.

USE-CASE ANALYSIS: DIFFICULTIES

- Misunderstanding and lack of understanding of the domain or the real problem.
- Requirements can change rapidly, resulting in requirements 'churn'.
- Attempting to do too much.
- It may be hard to reconcile conflicting sets of requirements.
- It is hard to state requirements precisely.

SUMMARY



SUMMARY

- Domain analysis is important for gathering requirements and understand the problem
- Never forget to get set the right scope
- Requirements are crucial to the success of the project
- Be precise and leave no ambiguity.