

## Vragen

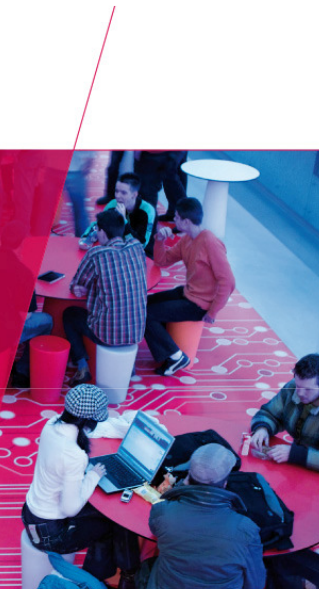
- Waarom zijn we in een continue software crisis?
- Noem een aantal kenmerken van software engineering.
- Geef een aantal software kwaliteitseigenschappen.
- Noem een aantal software ontwikkelmodellen.
- Noem een aantal kenmerken van XP.

## Opdracht

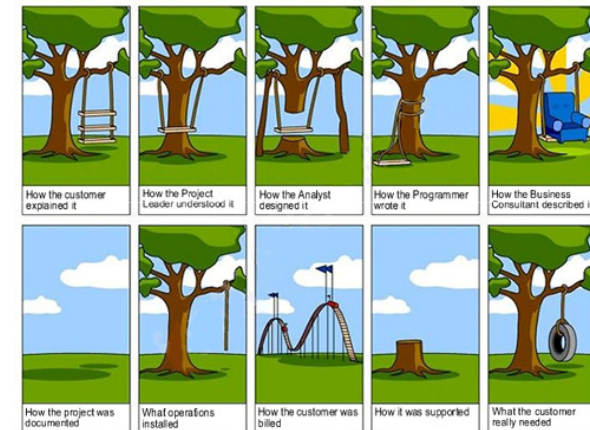
- Inleveren van het URD gemaakt in het Software Engineering Project (2IP45)
- Individueel: analyseren van een van de URDs geproduceerd door een andere groep
- Individueel: het opstellen van een analyse rapport, gebruikt hierbij hoofdstuk 5 “Validatie van requirements”.
  - Omvang max. 5 pagina’s.
  - Deadline laatste week blok D.

## Requirements Engineering

Mark van den Brand



## Requirements Engineering



## Domain Analysis

- The process by which a software engineer learns about the domain to better understand the problem:
  - The *domain* is the general field of business or technology in which the clients will use the software
  - A *domain expert* is a person who has a deep knowledge of the domain
- Benefits of performing domain analysis:
  - Faster development
  - Better system
  - Anticipation of extensions

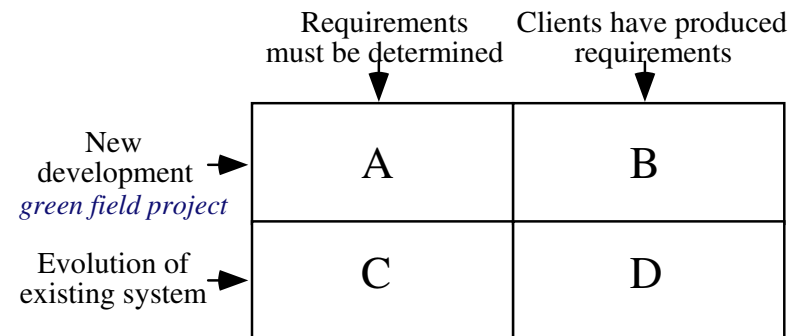
## Domain Analysis document

- A. Introduction
- B. Glossary
- C. General knowledge about the domain
- D. Customers and users
- E. The environment
- F. Tasks and procedures currently performed
- G. Competing software
- H. Similarities to other domains

## Example of domain analysis document

- [http://www.site.uottawa.ca/~laganier/seg3700/cemdo\\_main.htm](http://www.site.uottawa.ca/~laganier/seg3700/cemdo_main.htm)

## Starting Point for Software Projects

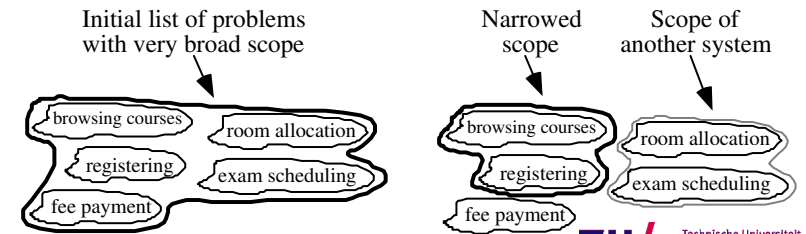


## Defining Problem and Scope

- A problem can be expressed as:
  - A *difficulty* the users or customers are facing,
  - Or as an *opportunity* that will result in some benefit such as improved productivity or sales.
- The solution to the problem normally will entail developing software
- A good problem statement is short and succinct

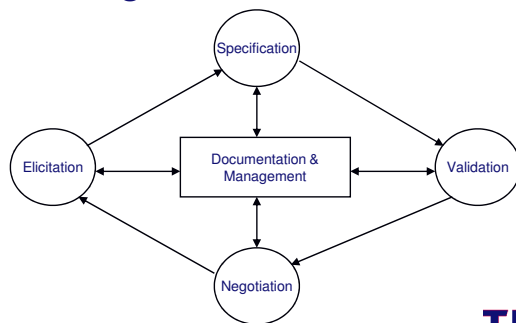
## Defining the Scope

- Narrow the *scope* by defining a more precise problem
- List all the things you might imagine the system doing
  - Exclude some of these things if too broad
  - Determine high-level goals if too narrow
- Example: A university registration system



## Processes in requirements engineering

- Requirements elicitation
- Requirements specification
- Requirements validation and verification
- Requirements negotiation



## What is a Requirement ?

- It is a statement describing either
  - 1) an aspect of what the proposed system must do,
  - or 2) a constraint on the system's development.
- In either case it must contribute in some way towards adequately solving the customer's problem;
- the set of requirements as a whole represents a negotiated agreement among the stakeholders.
- A collection of requirements is a *requirements document*.

## Types of Requirements

- **Functional requirements**
  - Describe *what* the system should do
- **Quality requirements**
  - *Constraints* on the design to meet specified levels of quality
- **Platform requirements**
  - *Constraints* on the environment and technology of the system
- **Process requirements**
  - *Constraints* on the project plan and development methods

## Functional Requirements

- What *inputs* the system should accept
- What *outputs* the system should produce
- What data the system should *store* that other systems might use
- What *computations* the system should perform
- The *timing and synchronization* of the above

## Quality Requirements

- All must be verifiable
- **Examples: Constraints on**
  - Response time
  - Throughput
  - Resource usage
  - Reliability
  - Availability
  - Recovery from failure
  - Allowances for maintainability and enhancement
  - Allowances for reusability

## Use-Cases: describing how the user will use the system

- A *use case* is a typical sequence of actions that a user performs in order to complete a given task
  - The objective of *use case analysis* is to model the system from the point of view of
    - ... how users interact with this system
    - ... when trying to achieve their objectives.It is one of the key activities in requirements analysis
- A *use case model* consists of
  - a set of use cases
  - an optional description or diagram indicating how they are related

## Use cases

- A use case should
  - Cover the *full sequence of steps* from the beginning of a task until the end.
  - Describe the *user's interaction* with the system ...
    - **Not** the computations the system performs.
  - Be written so as to be as *independent* as possible from any particular user interface design.
  - Only include actions in which the actor interacts with the computer.
    - **Not** actions a user does manually

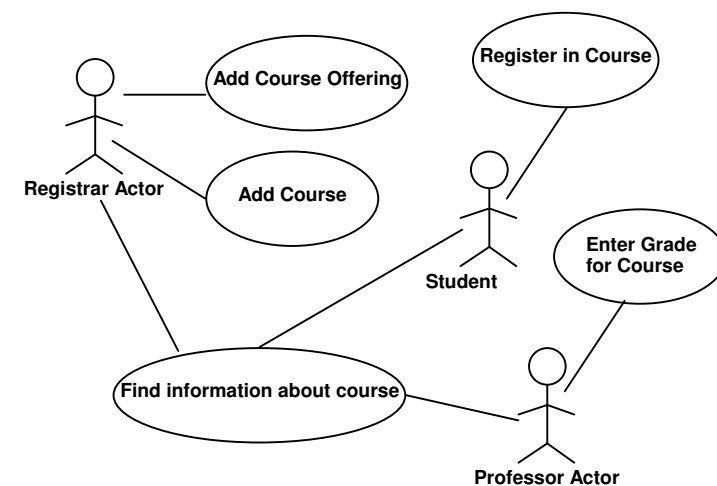
## Scenarios

- A **scenario** is an *instance* of a use case
  - A *specific occurrence* of the use case
    - a specific actor ...
    - at a specific time ...
    - with specific data.

## How to describe a single use case

- Name:** Give a short, descriptive name to the use case.
  - Actors:** List the actors who can perform this use case.
  - Goals:** Explain what the actor or actors are trying to achieve.
  - Preconditions:** State of the system before the use case.
  - Summary:** Give a short informal description.
  - Related use cases.**
  - Steps:** Describe each step using a 2-column format.
  - Postconditions:** State of the system in following completion.
- A and G are the most important

## Use case diagrams



## The modeling processes: Choosing use cases on which to focus

- Often one use case (or a very small number) can be identified as *central* to the system
- The entire system can be built around this particular use case
- There are other reasons for focusing on particular use cases:
  - Some use cases will represent a high *risk* because for some reason their implementation is problematic
  - Some use cases will have high political or commercial value

## The benefits of basing software development on use cases

- They can
  - Help to define the *scope* of the system
  - Be used to *plan* the development process
  - Be used to both develop and validate the requirements
  - Form the basis for the definition of test cases
  - Be used to structure user manuals

## Use cases must not be seen as a panacea

- The use cases themselves must be validated
  - Using the requirements validation methods.
- Some aspects of software are not covered by use case analysis.
- Innovative solutions may not be considered.