

# Requirement Elicitation

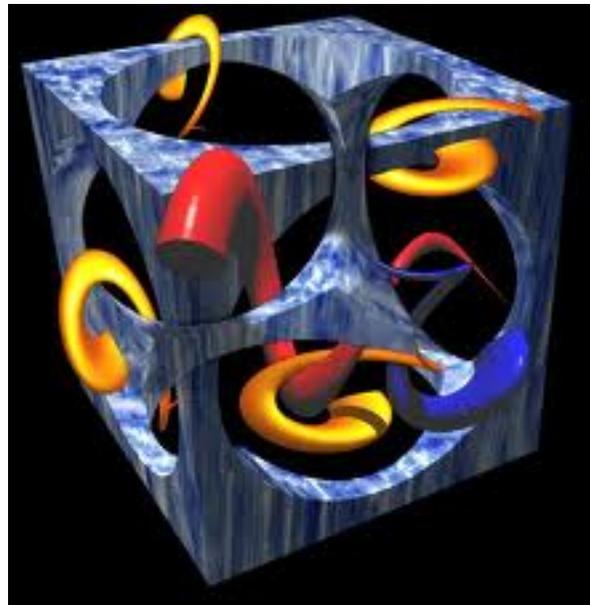
Phase 1

# Hmmm...



# Problem Domain

- The **problem domain** is the area of expertise that needs to be examined to solve the problem
  - Often, as computer engineers, we don't have expertise in that area!!!
  - A learning/exploration-process



Hmm???

# Requirement elicitation

- Requirement elicitation aims to get an understanding of the problem domain...
  - Techniques: Interviews, questionnaires, user observation, workshops, brain storming, **screen mockups, use cases, scenarios, ...**
- ... to be able to **analyze** and **model** the problem
  - Difficulty: Modest to very, very hard...

# Requirement Elicitation Overview

- Purpose
  - Why are we doing this? What are we trying to achieve? Who will use it?
- General characteristics of application
  - What kind of application is this? In what environment will the system be used?
- Scope of the system
  - What should be **in** and what should **not**
- Objectives and success criteria
  - When are we finished?
- What can we do with the system?
  - Functional requirements: Set of inputs, the behavior, and outputs
- Other criteria to judge the operation of a system
  - Non-functional requirements (GUI if required)

# The RAD

- The Requirements Elicitation and Analysis document (RAD) will be used to document the requirement elicitation (and later analysis)
  - **Template on course page**
- Document should to a large extent be understandable for non-computer professionals
  - Could form a basis for the contract
  - Can't be changed without negotiations with customer (in course: your assistant)
    - Customer can (and often do) initiates changes...

# Introducing: "The Monopoly Project"

*(thank's to group "Omen", Axel, Philip, Joanna and Kajsa)*

- As a running "case" we'll implement a prototype of the board game Monopoly by Parker Bros
  - It's an application instance, not universally applicable
  - Abbreviation: **MoPro** (on following slides)



# Problem domain MoPro

- Problem domain known (?) but note...
  - ...there are quite a few rules!
  - ...there are different sets of rules!
  - ...there are possible unspecified situations!
  - ...there are possible hidden rules (hard or impossible in physical world but possible with computers)?
- **Have to find them all....!!!**

# Requirements for MoPro

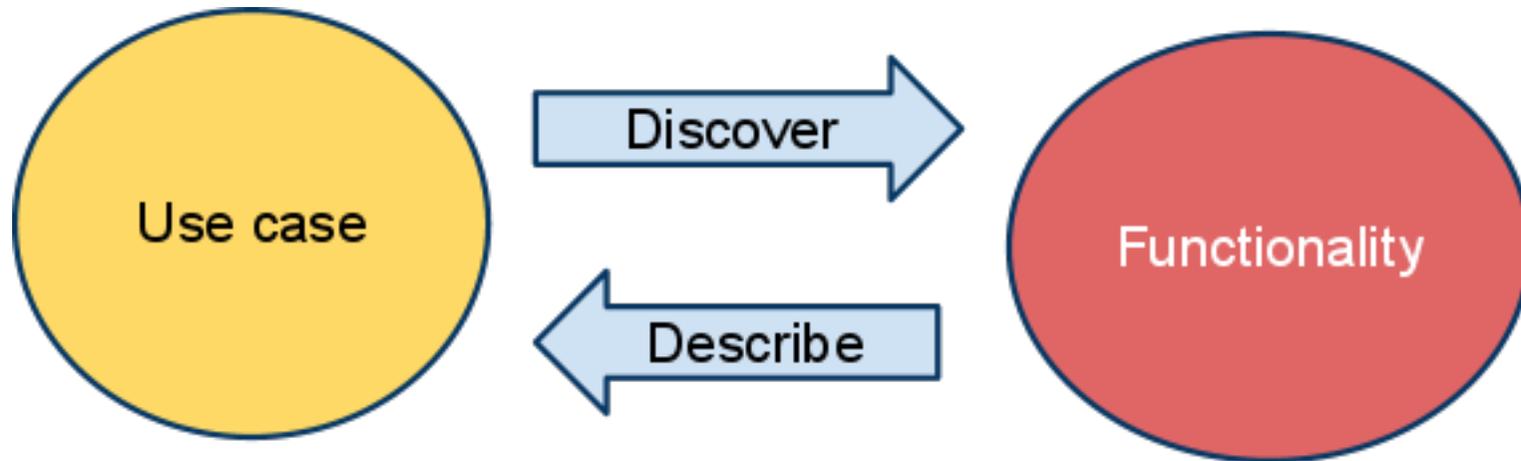
- We'll inspect section 1 and 2.1 of the RAD□ (on course page in near future)
  - Purpose
  - General characteristics of application
  - Scope of the system
  - Objectives and success criteria
  - Proposed application

# Use cases

- A **use case** describes a sequence of actions that provide a measurable value to an actor (user or possibly another system)
- A use case is a text document
  - Often two columns one for user, one for system
  - **Template on course page**
- Use case names begin with a strong verb
- Name use cases using domain terminology
- Use case text uses domain terminology

# Functional Requirements

- To capture functionality we create use cases
- Known or apparent functionality described as use cases
- Start at either side



# Scenarios

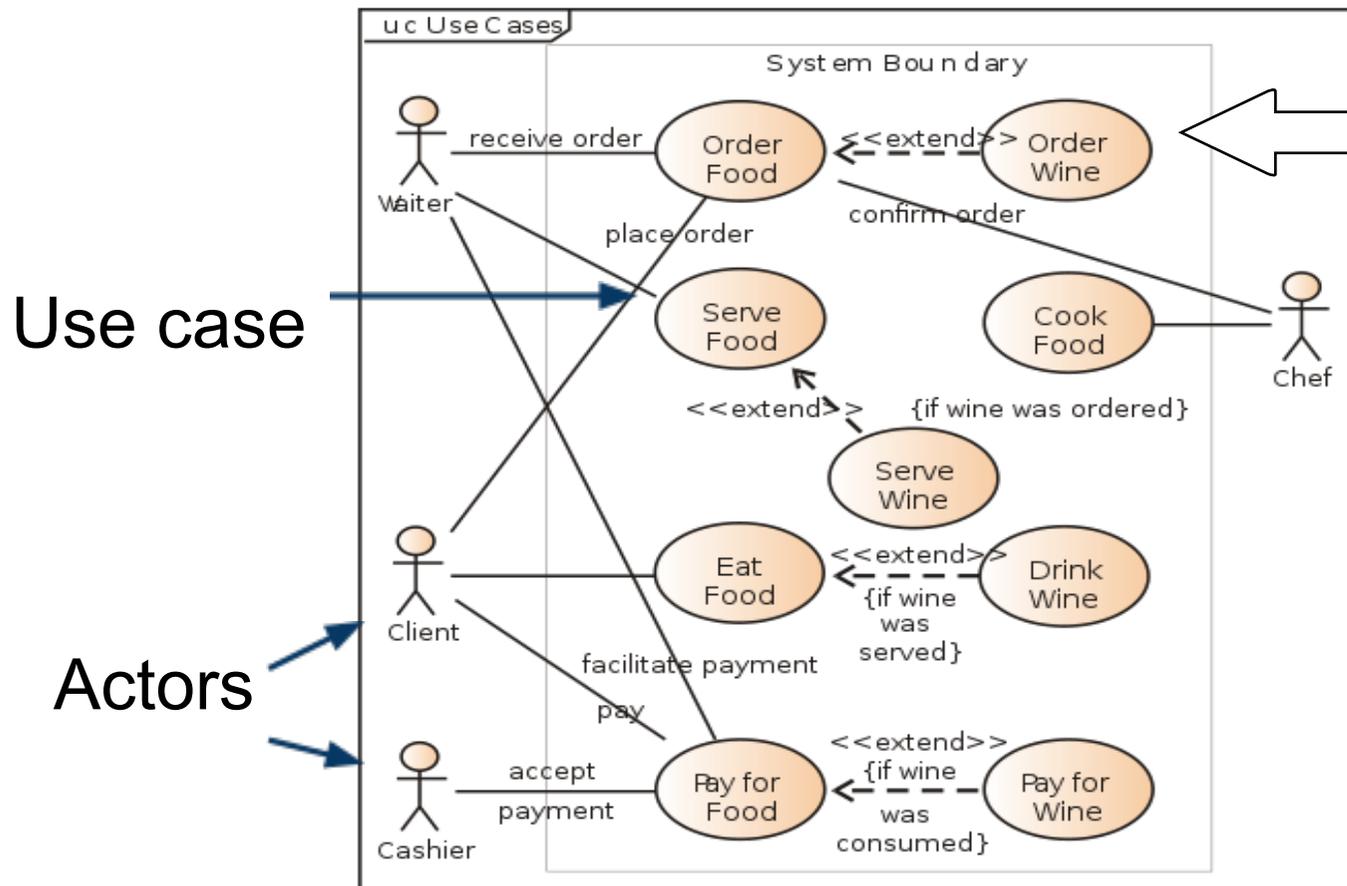
- Finding the general interaction with the system is sometimes too complicated
- A scenario is a special case of a use case (hard coded data)
- Example: Schedule a meeting
  - **Use case:** What is a meeting in general? Location, remote participating, duration, who can participate individuals, groups, roles, rights...who can schedule a meeting .. etc. lot to analyze
  - **Scenario:** Pick an instance. Sven schedules a meeting with Lisa (remote) and all sales representatives at ... thu 10-11...

# More on Use Cases

- Use case granularity
  - Too large, have to break down
  - Too small, trivial, possible part of other use case
- Use case "extends"
  - Inserting additional action sequences into the base use-case sequence
- Use case "includes"
  - An invocation of a use case by another one
- Use case refactoring
  - Must do! Else possibly end up with duplicate code

# UML for Use Cases

- Use case diagram, not overly useful but can give an overview



There will be a use case text named "OrderWine" to describe in detail

# Use Cases for MoPro

- Lets try to find some...
  - Draw some UML
  - Write an use case text
- ... then we'll inspect the RAD sections 2.2, 2.4.1-2.4.2 and appendix

# Priority of Use Cases

- The use cases should be ordered by priority
  - High, implemented in first iteration
  - Mid, later iterations
  - Low, optional, possible never implemented
- High priority characteristics
  - Significant, central functionality
  - Substantial architectural coverage, stress or illustrate a specific point of the architecture (to be solved)

# Use Cases Signals Exception Handling

- Normal flow and exceptional
  - Star with normal flow
  - Add exceptional cases
- Example: User enables auto reply on mail

**Normal flow:**

User	System
1. Selects auto reply	
	2. System shows a ...
3. User ...	
	4. System...
	n. Incoming mail
	n+1. Auto replies to sender

**Alternative (exception):** Can you think of a (funny) exception?

# Non-Functional requirements

- Usability
  - The ease of use and learnability of a human-made object (in our case GUI)
- Modifiability (Scalability)
- Reliability
- Performance
- Interoperability
- Testability
- ... more later

See [Wikipedia](#) for long list

# User Interface

- During requirement elicitation we also sketch a preliminary GUI
  - Initially envision the system.
  - Enables you to explore the problem space with your stakeholders
  - Enables you to explore the solution space of your system.
  - A vehicle to communicate the possible UI design(s) of your system
  - A potential foundation from which to continue developing the system

# User Interface for MoPro

- Should look like a Monopoly game
  - Flat 2d look for now
- Possible to select different location (London,...Ullared)
  - Must be possible to change texts, etc.
  - Must use internal representation (keys) for text.
- Possible small screen
  - Will use popup for details, dialogs for messages
- We'll create a sketch...

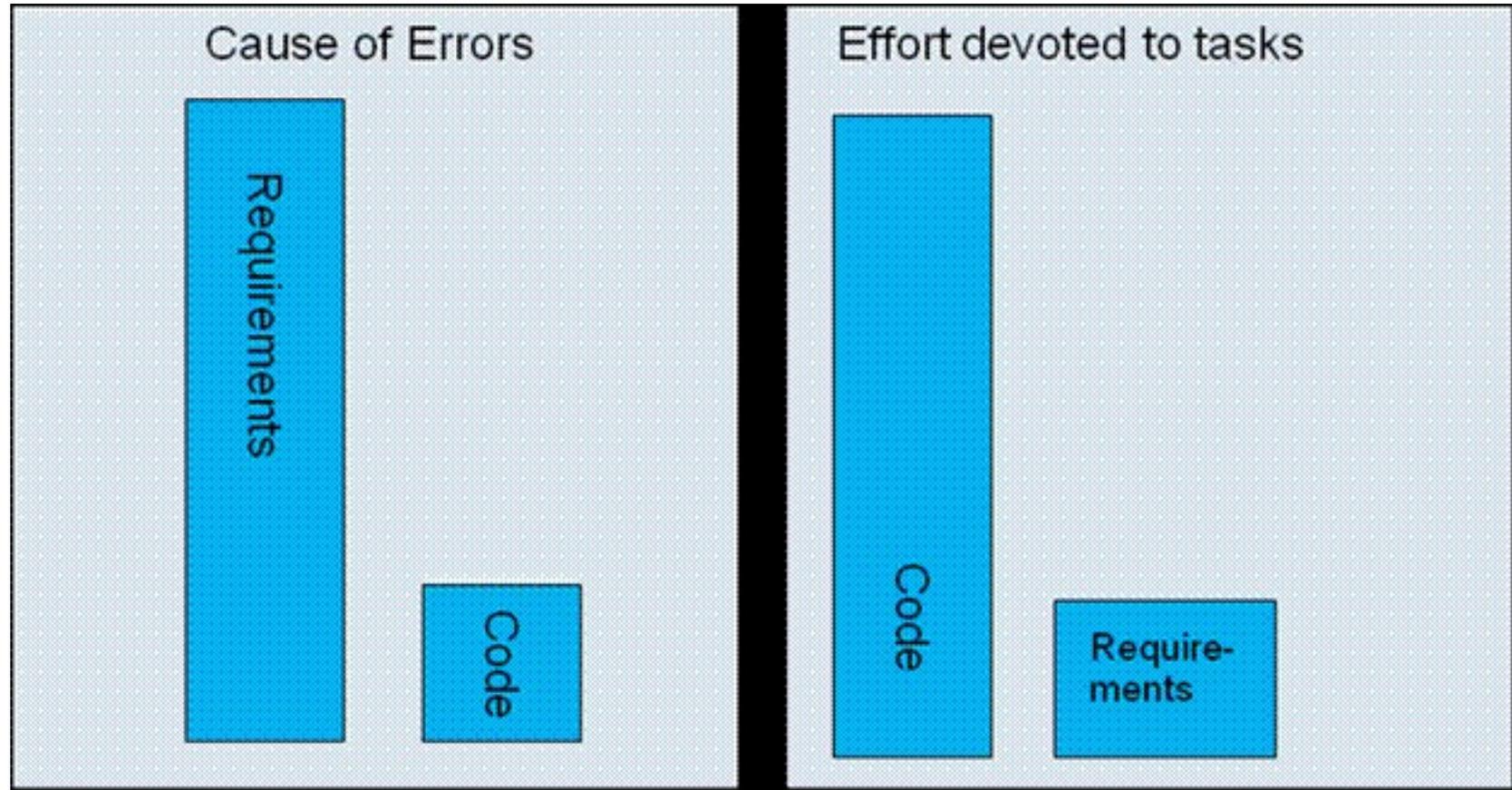
# Non-Functional for MoPro

- Well, what do we have...?
- ...inspect section 2.3 of RAD

# Future directions

- Possible continuations of the project
- Not mandatory
- Collect ideas, brain storm

# Hmmm...



# Summary

- Requirement elicitation focuses on understanding the **problem domain**
  - Functional and non-functional requirements
- Documented in RAD