Requirement Elicitation [Iteration 1, Phase 1]

Slide Series 2

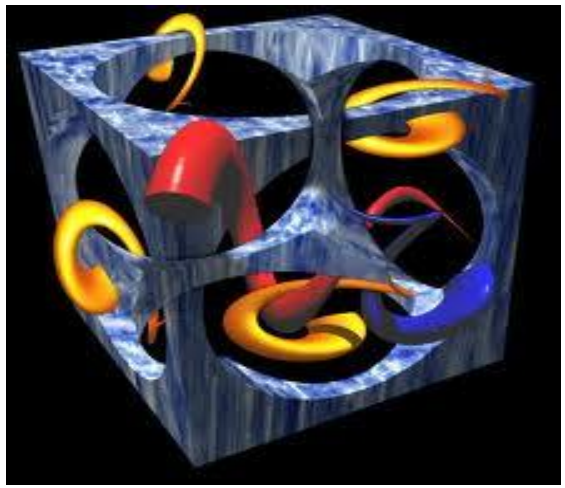
Hmmm...


I'LL GO UP AND FIND OUT WHAT THEY NEED AND THE REST OF YOU START CODING!
SKELTON

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!
  - How do you write "human resource" system for hotels?
- Must explore/learn/understand...



What's the inside of this?

Requirement elicitation

Requirement elicitation is the exploration/learning phase

During RE we aim to get an understanding of the problem domain ...

...and to get a common vision of what to build!

- Else you'll end up building multiple (different) applications inside one application.... (not recommended ;-))!

RE Topics

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 it 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 kind of?)

Introducing: "The Monopoly Project"

As a running "case" we'll implement a prototype of the board game Monopoly by Parker Bros.

- It's an application instance, there are other kind of applications/styles/ways...
- Abbreviation: **MP** (on slides)

Problem Domain MP

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 contradicting rules!
- ...there are possible hidden rules (hard or impossible in physical world but very possible with computers)?

Have to find them all....!!!

Requirements for MP

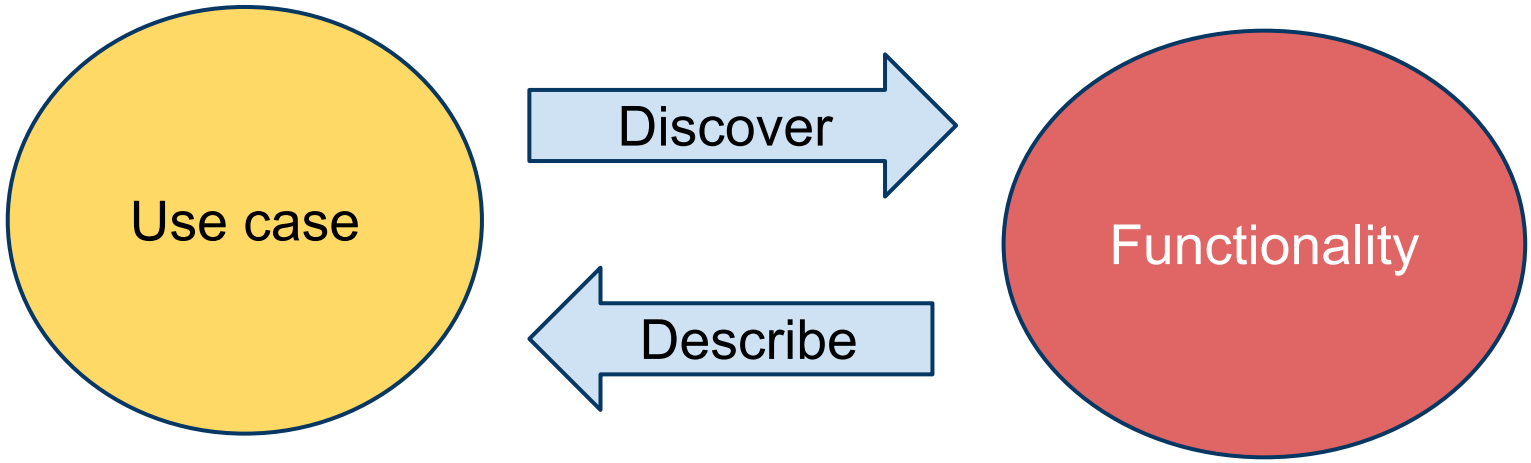
We have no customer so what's our vision for the project?

We'll inspect beginning of the RAD for MP (on course page)

- Purpose
- General characteristics of application
- Scope of the system
- Objectives and success criteria

Finding Functional Requirements

- To capture functionality we create **use cases** (examples later)
- Known or apparent functionality described as use cases
- Start at either side



Use Cases

A use case describes a sequence of actions that provide a measurable value to an actor (user or possible another system)

- A use case has a name
- Name use cases using domain terminology (board, dice,...not array, randomGenerator ...)
- Use case names begin with a strong verb

A use case is represented as a text document

- Often two columns, one for user, one for system
- Numbered steps (no commonly accepted standard)
- **Template on course page**

Hmm, who started out with this ?

Use Cases: Starting out

The sequence of actions starts with a user invoking some action from the user interface

- To write the UC's we need a preliminary user interface

User Interface

During requirement elicitation we sketch a preliminary user interface

- Needed for use cases
- Tip: Use a paper mock up

Also

- 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 (finding use cases)

User Interface for MP

Should look like a Monopoly game

- Flat 2d look for now
- Possible animations later

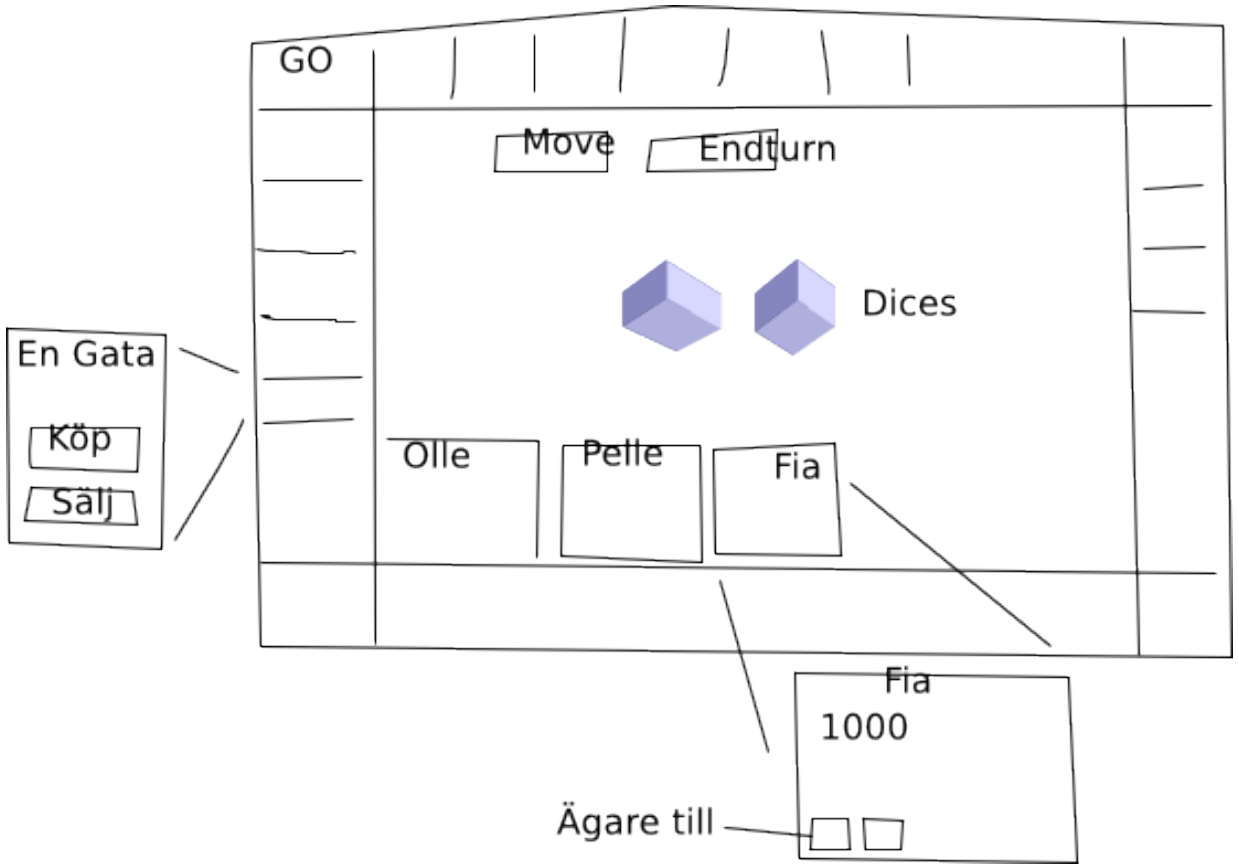
Possible to select different location (Alingsås, Warszawa, Ouagadougou,...)

- Must be possible to change texts,
- Internationalization,...must use internal representation (keys) for text

Possible small screen

- Will use popup for details, dialogs for messages

GUI Sketch MP



Use Cases: Normal FLow

Normal flow is the most likely (normal) path in the sequence of actions

	User	**System**
1	clicks on destroy button (see GUI sketch)	
2		feels bad
3	clicks ok in confirm dialog	
4		goodbye cruel world...

Use Cases: Alternate flows

Often have alternative path's in the sequence of actions, **alternate flows**

- Depending on outcome of response, or others...
- As noted: No standard numbering system, example later

Use Cases: Exceptional flow

How will the sequence of actions behave if we get an exception?

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

Exceptional flow: Can you think of a (funny) exception?

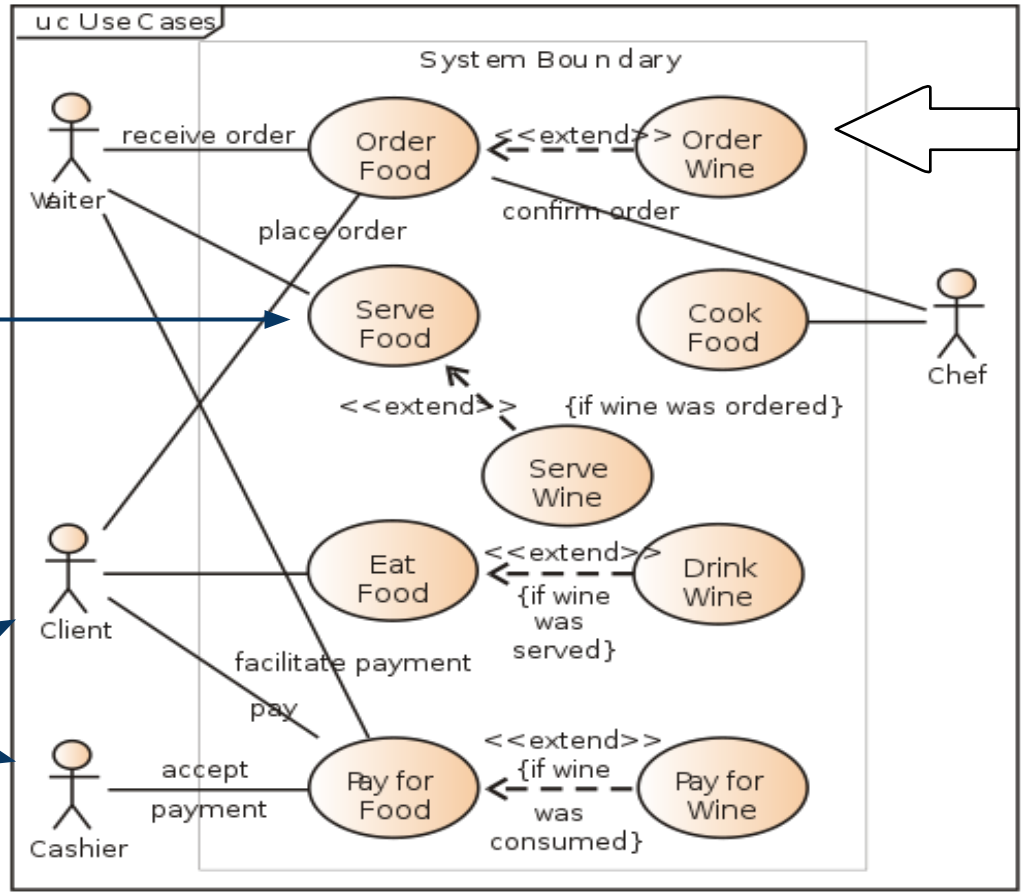
Finding Use Cases

Techniques

- Interviews
- Questionnaires
- User observation
- Literature study
- Workshops
- Brainstorming
- Screen (GUI) mockups

UML for Use Cases

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



Use case

Actors

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

Use Cases: Fine Prints

Use case granularity

- Too large, have to break down
- Too small, trivial, possible part of another use case

Use case "extends"

- Inserting additional action sequences into the base use-case sequence

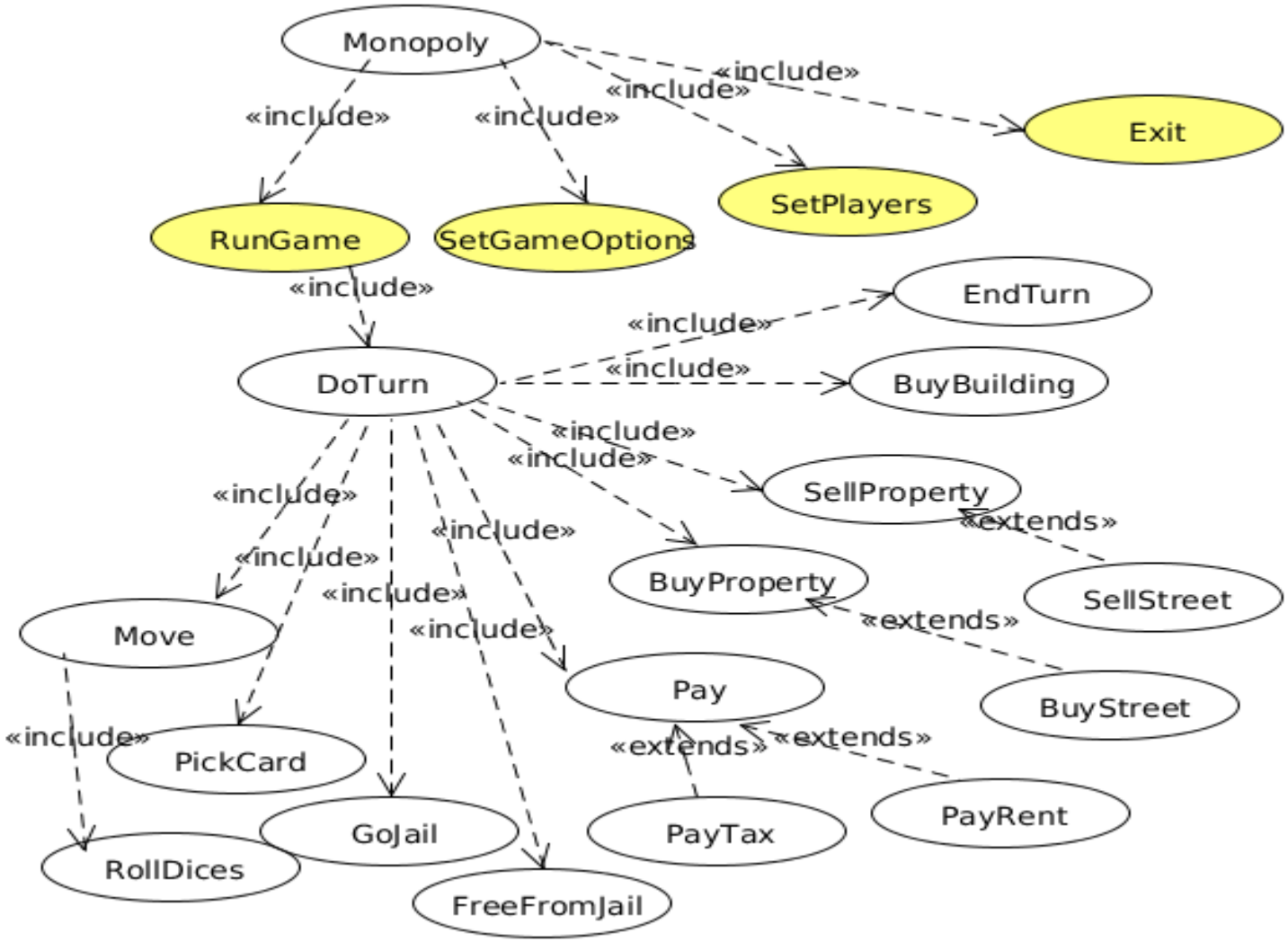
Use case "includes"

- An invocation of a use case by another one

Use case refactoring

- Must do! Else possibly end up with duplicate code

Use Cases for MP



UC Examples for MP

The UC's (on course page)

- Move
- EndTurn

Quality of UC

The quality of the UC's will have impact later

- Let UC text be as focused (short) as possible but try to be precise (missing facts will possibly affect later stages)
- Make it a short play (one person emulating the system, playing really dumb), does it work?

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 coverage of the solution, stress or illustrate a specific point of the solution (to be solved)

UC Priority for MP

Highest
- Move
- EndTurn

Mid
- Buy, sell
- …

Low
- Save and restore game

Scenarios

Finding the general interaction (use case) with the system is sometimes too complicated

- A scenario is special case of a use case (fixed 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 .. lot to analyze
- Scenario: Pick an instance. Sven schedules a meeting with Lisa (remote) and all sales representatives at ... thu 10-11...

Non-Functional requirements

- Usability, the ease of use and learnability of a human-made object
- Reliability, probably NA
- Performance, probably NA
- Supportability
- Testability (yes, implicitly mandatory in course more to come...)
- Implementation (any restrictions, yes, Java in this course)
- Packaging and installation
- Legal

See Wikipedia for long list .. (don't overdo)

Non-Functional for MP

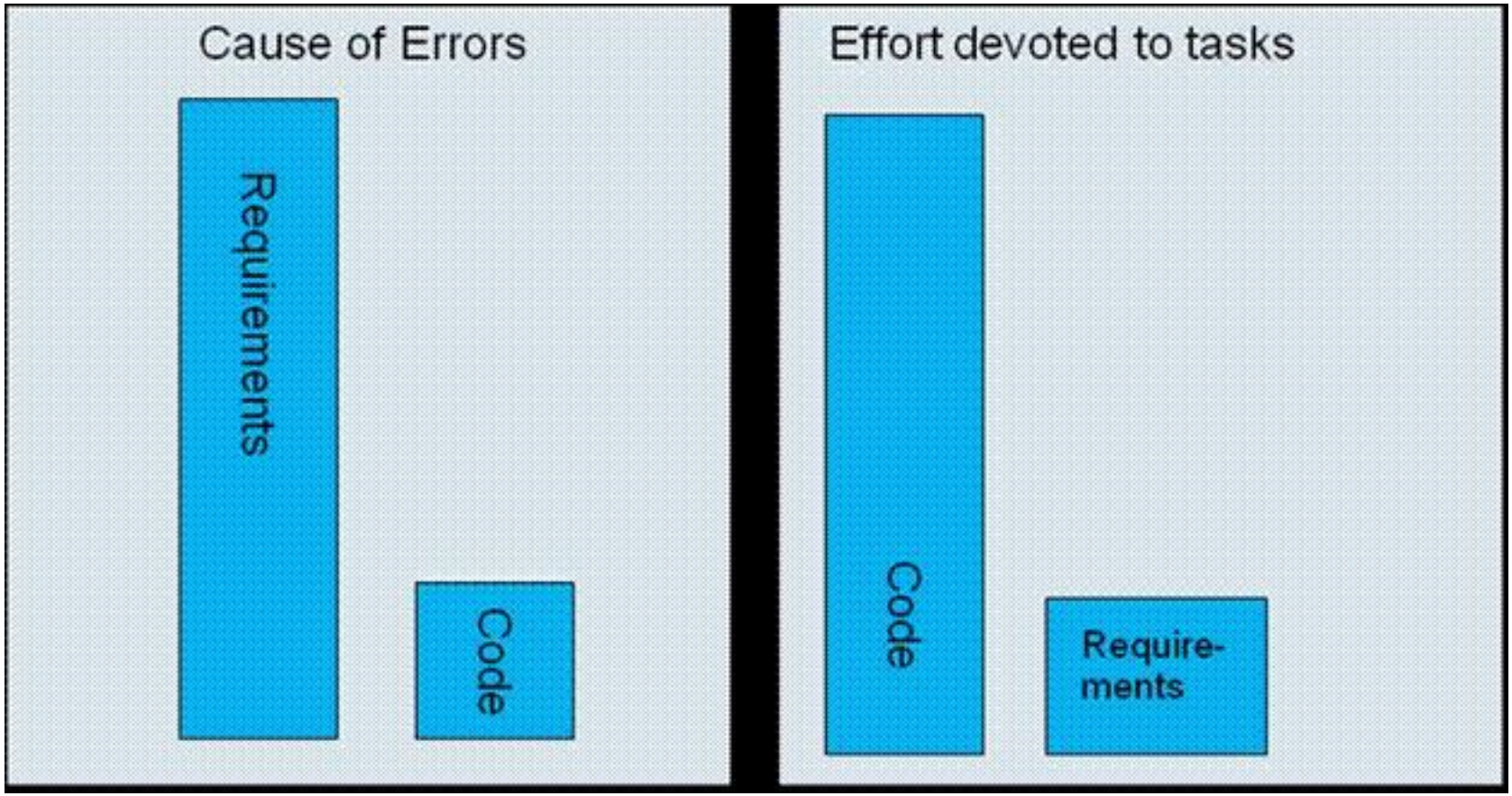
We'll inspect the RAD again

Documenting the RE

All outcome from RE is documented in the RAD

- List of use cases
- Put full UC texts in appendix (not inline)
- Priority of UC's
- Non-functional requirements
- Preliminary user interface

Hmmm...


Cause of Errors
Requirements
Code
Effort devoted to tasks
Code
Require-
ments

Summary

Requirement elicitation focus on

- Understanding the problem domain
- To create a shared vision of the project
- Finding functional and nonfunctional requirements (GUI)
- We tried to do so using the Monopoly project as a case

RE documented as part of RAD

Next: From requirements to the domain model,
i.e. the analysis phase