Stijn Hoppenbrouwers

Requirements Engineering, Lecture 1:

RE and Use Cases

Radboud University Nijmegen

Course Setup

- Lectures (Stijn Hoppebrouwers, guest lectures), Tue 10.30-12.30
- Textbook
- Syllabus
- Case Project (with Dirk van der Linden), Thu 13.30-15.50

Deliverables:

- Written exam
- Case Project report
- Case Project presentation

- Exam and project count 40%-60% respectively for final mark
- Case Project Report and Case Report Presentation count 75%-25% respectively for Project mark

Communication

- Blackboard
  - Announcements
  - Mailing facility for lecturer
- If necessary, individual e-mails
- Wiki: https://lab.cs.ru.nl/algemeen/Requirements_Engineering
  - Digital files
  - Up-to-date version of syllabus
  - Deadlines, dates, etc.
  - Workshop (“werkplaats”). Every group has their own space, also open to other groups (!).
- Personal: office HG02.611 (preferably by appointment), or at lectures

What I expect you do

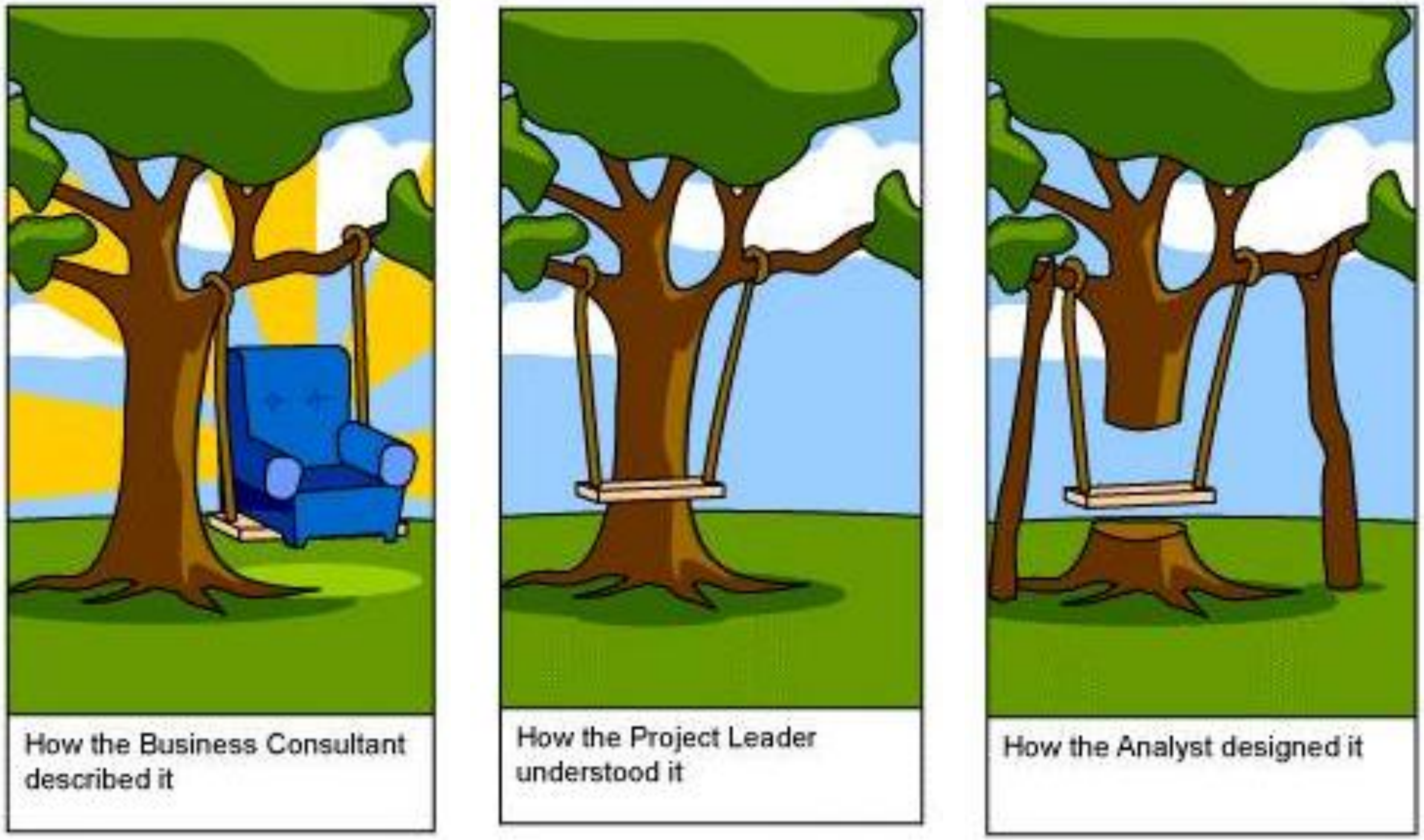
- Read the textbook chapters *in time* (see website for dates)
- Read the syllabus *in time* (ditto)
- Be present at lectures: they do add something
- Participate in Case Project (groups of 4-5)
  - Co-write the report
  - Co-write and possibly co-present the presentation
- Sit the exam

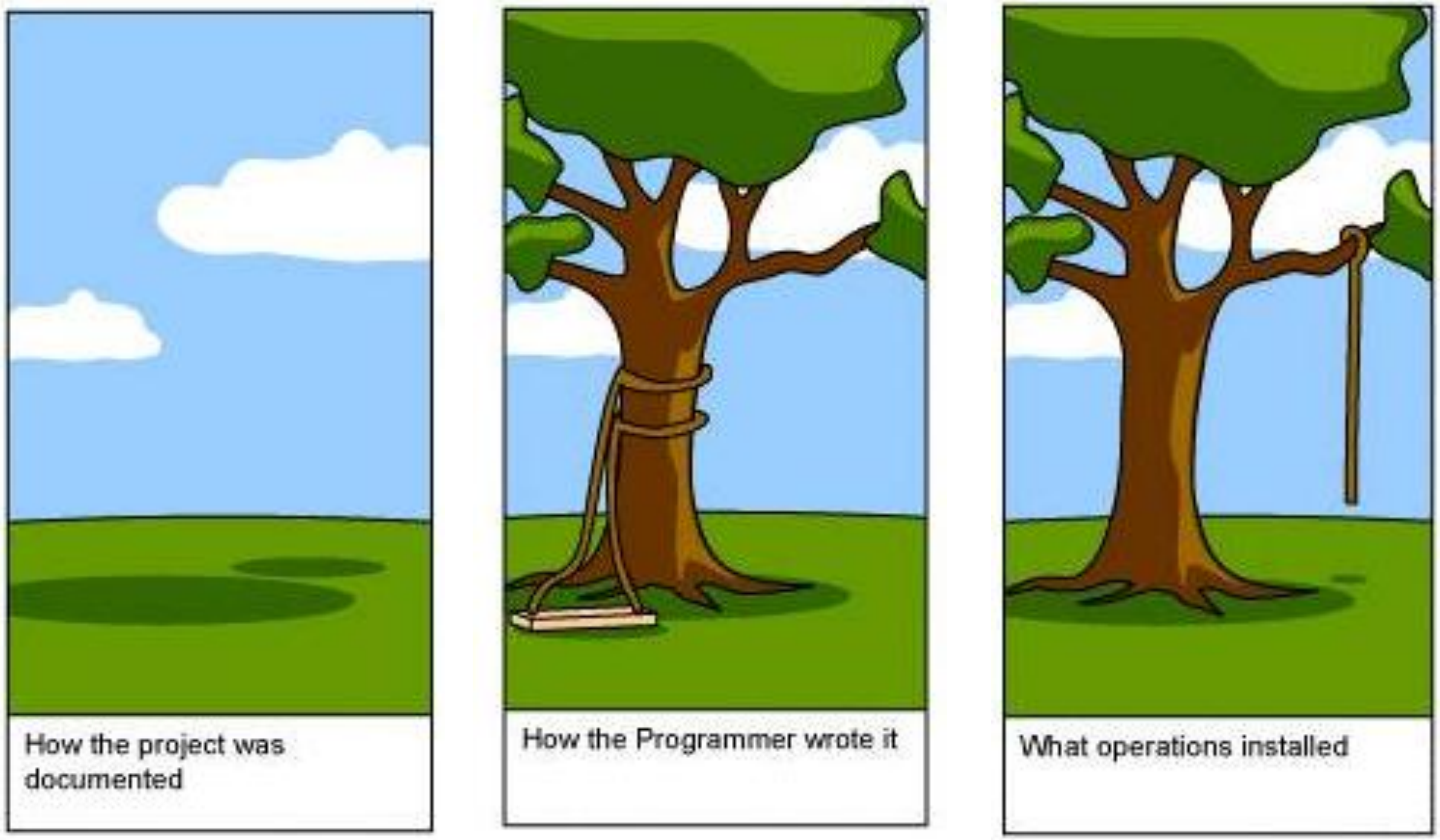
- Try and get the hang of RE activities
- Try and look beyond concrete activities and see what RE is about in view of System Development

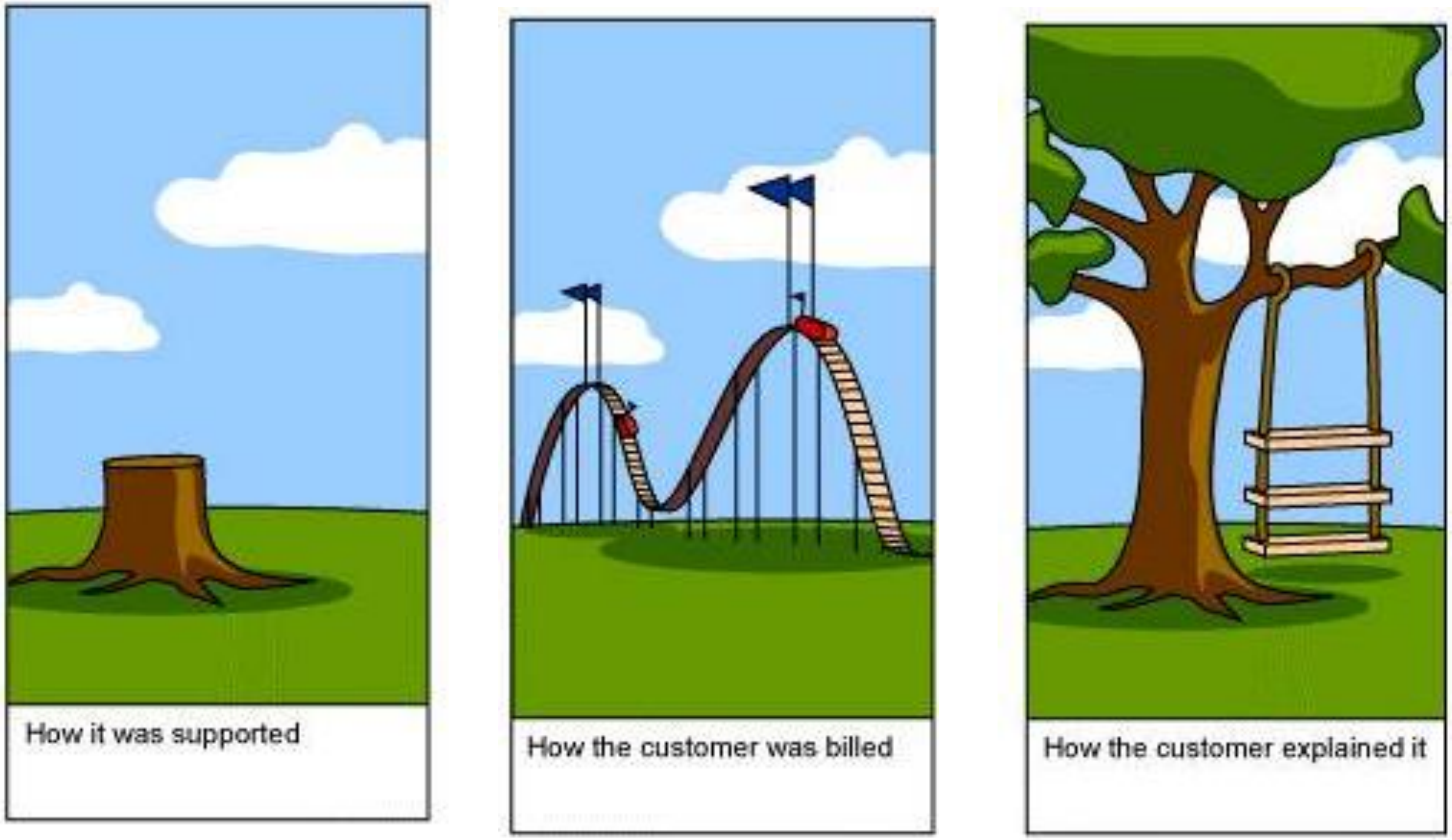
Goals of the course

After sitting the course you should be able to…

- Distinguish requirements from technical design
- Evaluate the quality of a requirements specification
- Gather, specify, and document good requirements, provided you have the information you need
- Explain what the place is of requirements documents within the larger system development process
- Explain how particular items in requirements documentation fit together and fit the broader SD process documentation
- Integrate techniques from domain modeling and business process modeling within the RE process
- Reflect on the RE field process from the perspective of generic SD theory


How the Business Consultant described it
How the Project Leader understood it
How the Analyst designed it


How the project was documented
How the Programmer wrote it
What operations installed


How it was supported
How the customer was billed
How the customer explained it

What the customer really needed

Requirements Engineering

- System engineering
  - WHY (problem, situation)
  - WHAT (essential solution, “black box”)
  - HOW (concrete solution, inside the “white box”)

- There is a logic to this order, but in practice it does not work like that.

- Where to start? It depends…

Kulak and Guiney on Crusade?

- Contract-style requirements lists

- Prototypes

WHAT-HOW

- WHAT – HOW distinction: always hard
- The gnome view on RE
- WHAT before HOW: common sense or dogma?

WHY and WHAT

- “Problem statement” in the documentation makes this explicit. This is a “negative” perspective on “why”.
- WHY before WHAT: again, seems logical, but…

RE & Design

- Requirements **gathering** may not be design, but
- Requirements **specification** *is, inevitably,* partly design!

- “Design” is often viewed as the phase “after” RE; this may cause confusion
- [K&G] are guilty of this: for them “design” = “technical design”

- Better to distinguish between “functional design” (part of RE) and “technical design” (post-RE)

- However, RE = req. gathering + (functional design **– interface details**)

Functional and non-functional requirements

- Functionals: “what users need for the system to work”

- Non-functionals: “requirements hidden from users”
- Misleading: “non-functionals” *do* concern functionality, just not directly related to hands-on use (more general)
- Non-functionals are tricky!
- “-ilities”
- Often a technical flavour, and technical/architectural implications
- Much more in section 4.2.10 [K&G]

The WHATs of RE (*also* of use cases!)

- Find out what users need
- Document users’ needs
- Avoid premature technical design decisions
- Resolve conflicting requirements
- Eliminate redundant requirements
- Reduce overwhelming volume
- Ensure requirements traceability

Use Cases and related items

- Capture essential interactions between users and system
- For “typical” users, system = “black box” (WHAT without HOW)
- Use cases & the UML

- **Use Case Survey**: table of *n* use cases
- **Use Case diagram**: depicts (relations between) actors and use cases, and between use cases
- **Use Case**: type-level, generic textual description of interactions of (outside) actors and the computer system
- **Scenarios**: instance-level, specific textual descriptions of examples of interactions. Use Cases : Scenarios = 1:n

Use case template ([K&G] p42-6)

- Use case name
- Iteration
- Summary
- Basic course of events
- Alternative paths (to avoid IF-THEN-ELSE bog)
- Exception paths
- (Extension points)
- Triggers (when or why does an actor enter the use case)
- Assumptions (ref. “non-formalized assumptions” in B&B)
- Preconditions (ref. “formalized system assumptions” in B&B)
- Postconditions (ref. “formalized system commitments” in B&B)
- Related business rules
- Author
- Dates

Language in use cases and scenarios

- User language only
- Not implementation language!
- But if users are technicians, user language = technical language *of a sort*
- If user/stakeholder language is not coherent or not agreed upon, work on this together with users/stakeholders
- This may actually bring about new questions and insights concerning the domain and the requirements!

Three cycles in “our” RE process

- Façade iteration
- Filled iteration
- Focused iteration

Documentation items beyond use cases

- Introduction
- **Problem statement**
- Stakeholder analysis
- Mission – Vision – (Values): to be provided by initiator
- Statement of work: work plan (p57 [K&G])
- Risk analysis

- **Business rules catalogue**
- **Domain models (ORM), including example ppopulation**
  - **One for each use case**
  - **Preferably, also an integrated one covering all others**
- **Terminological definitions**

Rudimentary Stuff

- Executive sponsor viewpoint: implicit
- Use case tests: implicit

- Business process definitions: optional appendix
- GUI metaphors / storyboards: optional appendix

Overview of deliverables

- On the Wiki you can find an excel sheet showing the required deliverables per phase

Exercise: Use Case Survey, Diagram, and Template

- First read the relevant sections in the book and syllabus (also see planning on website)
- Take as a case a standard **library information system**. You can take the university library system as an example, but please look beyond library clients as users. Also ask yourself: what can *librarians* do with the system?
- Identify the key (i.e. vital) use cases (round about 5 will do nicely)
- Create a matching use case survey and *integrating* diagram
- Fill in the use case template for at least one Use Case (try pick an interesting one)
- **Take your results to the responsiecollege on Thursday!**