UCL
Université
catholique
de Louvain

LINGI2255 Software Engineering Project

# Lesson 1

# Course organisation and requirements phase

*Sébastien Combéfis*

*Monday September 14, 2015*

# Software Engineering project

- This project consists of the development of a realistic application, representative of a typical industrial software system, under semi-professional working conditions.
- The topic of the software system to be constructed is proposed by a non-profit organisation who participates in the organisation of this course.
- The project will be carried out by groups of 6 to 8 students.

# Course information

- 1st semester, 6 ECTS (15h + 45h)

  *Master 1 (INFO21, SINF21, SINF2M1)*

- **Language:** English

- **Teacher:** Dr Sébastien Combéfis (sebastien.combefis@uclouvain.be)

  *Compensates Prof. Kim Mens (in 2015–2016)*

- **Assistants:** Christophe Limbrée, John Aoga, Michaël Saint-Guillain, Emery Kouassi Assogba, Hélène Verhaeghe, Parfait Tokponnon

- **Customer**: Victor de Beco (ASMAE ASBL) (debecovictor@gmail.com)

# Course goals

- Work in a team

  *Roles, planning, meetings...*

- Develop a large-scale, industrial-like software system

  *From its initial requirements to its final deployment*

- Follow a model-based development approach

  *Document design decisions and software artefacts*

- Use appropriate tools and frameworks

  *To get a good quality final product*

# Evaluation

- Continuous evaluation
  - 75%: Five intermediate evaluations (15% each)

    *Individual participation and intermediate report*
  - 25%: Final report, delivered system, documentation and tests, presentation and demonstration of the final product
- No possibility to do a second session for this course!

# Disclaimer

- LINGI2255 is a relatively "new" course
  - Third time the course is given in this format
  - No longer tightly coupled to SE course LINGI251
- Changes will be experimented this year
  - Based on course evaluations (with remarks from students)
  - Let us know as soon as possible if anything goes wrong
- Detailed information will be communicated gradually

# Project phases

- Project split in several phases
  - Requirement analysis (2 weeks)
  - Four intermediate phases (every 2 weeks)
  - Final report, project defence
- Intermediate phases depend on development method
  - Waterfall, agile... to be determined in first phase
  - Each phase requires to produce models, code...

# Points of attention

- Database modeling

  *Refactor current database, propose data schema...*

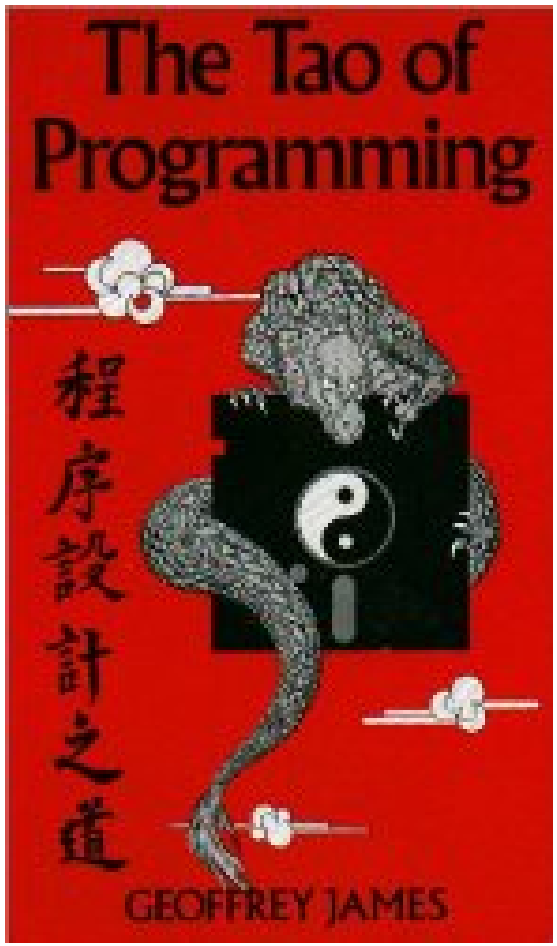
- Algorithmic aspects

  *Two main algorithms to propose...*

- Documentation and UX

  *User and developer documentation, friendly UX...*

# The Tao of Programming

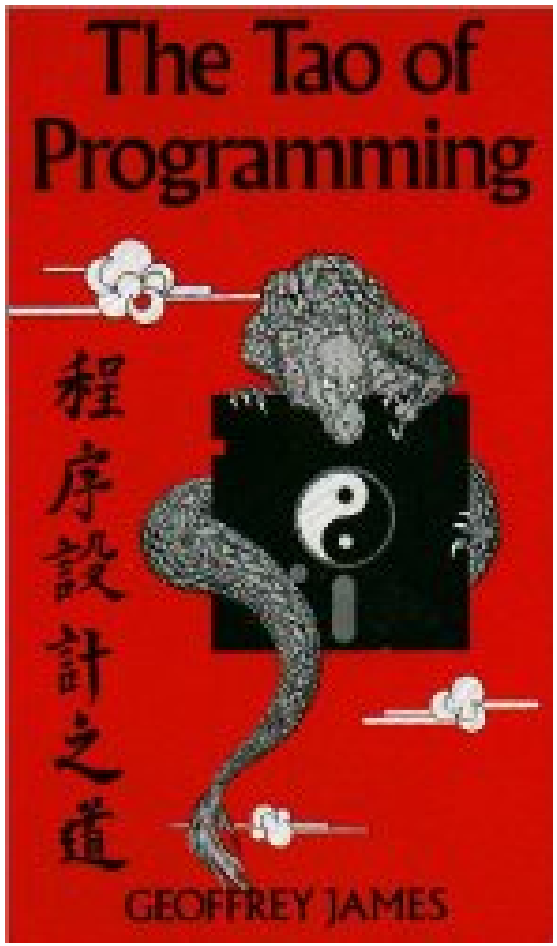



ISBN: 978-0-931-13707-5

**The Tao of Programming**
by *Geoffrey James*

*Book four (Coding)*
*Thus spake the master programmer:*
> *"A well-written program is its own heaven a poorly-written program its own hell."*

# The Tao of Programming




ISBN: 978-0-931-13707-5

**The Tao of Programming**
by *Geoffrey James*

*A program should be light and agile, its subroutines connected like a string of pearls. The spirit and intent of the program should be retained throughout. There should be neither too little nor too much, neither needless loops nor useless variables, neither lack of structure nor overwhelming complexity.*

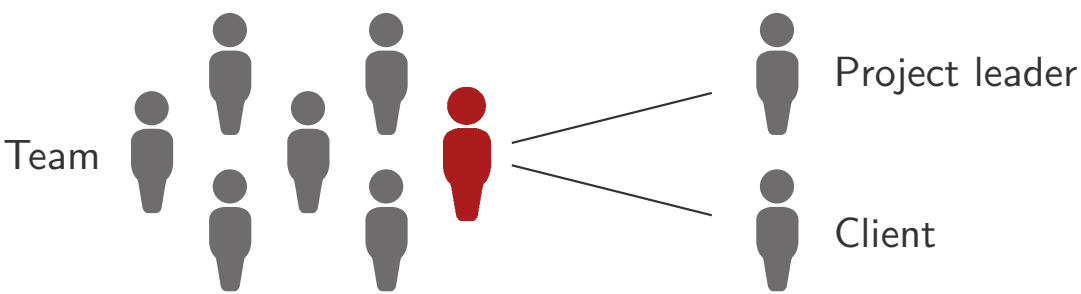
*A program should follow the 'Law of Least Astonishment'. What is this law?*

# Team organisation

- Team with 6–8 developers

  *One of which will be the (internal) project manager*

- Overseen by a project leader (teaching assistant)
  - Weekly meeting
  - Presenting progress and difficulties, assessing alternative options, feedback on intermediate reports, schedule monitoring



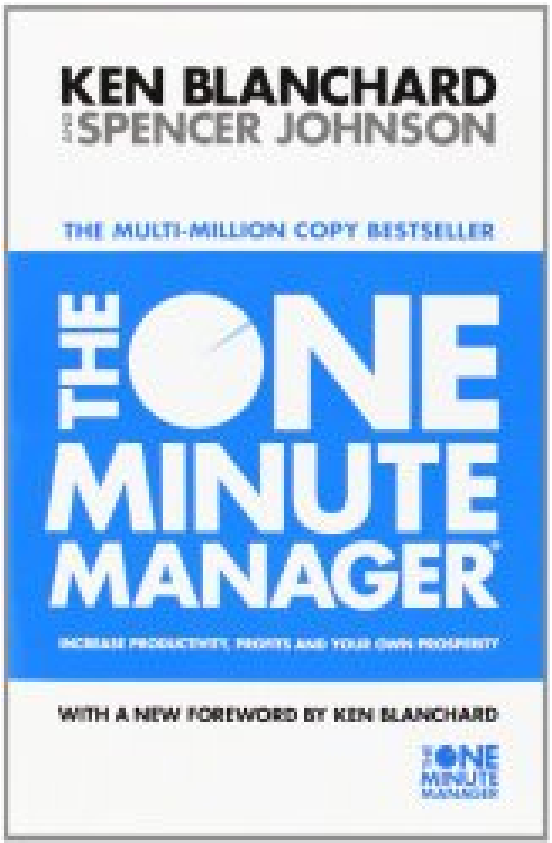
# Teamwork

- Role of the (internal) project manager
  - Spokesperson for the team
  - Delegates responsibilities to team members
- Responsibilities to be mentioned in reports

  *And discussed with the project leader during the weekly meetings*
- Do not exclude newcomers to your team

  *Managing unexpected situations is a skill that needs to be learned*

# The One Minute Manager




ISBN: 978-0-007-10792-6

**The One Minute Manager**
by *Kenneth H. Blanchard* and *Spencer Johnson*

*The brief volume tells a story, recounting three techniques of an effective manager:*

*one-minute goals,*
*one-minute praisings and*
*one-minute reprimands.*

*Each of these takes only a minute but is purportedly of lasting benefit.*

# Project defence

- In the last week before the Christmas holidays
  - On an evening between 6pm and 9pm
  - With the customer, the project leader and the professor
  - Mandatory for **all** team members
- Two parts (45 minutes)
  - 25 minutes of presentation (including questions)
  - 20 minutes of demonstration

# Courses

| Week | Date | Room | Activity |
|---|---|---|---|
| W1 | Mon. 14 Sep.<br>8:30–10:30am | SCES 01 | ■ Presentation of the case by the customer<br>■ Project introduction by the professor<br>■ Introduction to the requirements phase |
| W2 | Wed. 23 Sep.<br>4:15–6:15pm | BARB 93 | ■ Q/A with the customer<br>■ Software architecture |
| W3 | Mon. 28 Sep.<br>4:15–6:15pm | BARB 93 | ■ Test-Driven development (TDD)<br>■ Testing (client/server, interface) |
| W4 | Wed. 7 Oct.<br>10:45–12:45am | BARB 93 | ■ Design patterns, good practices<br>■ Pragmatic programmer |
| W5 | Wed. 14 Oct.<br>10:45–12:45am | BARB 93 | ■ Continuous integration<br>■ Project management tools |

# Deadlines

| Week | Date | Deliverable |
|---|---|---|
| W2 | Fri. 25 Sep. | ■ Requirements<br>■ Development methodology and planning |
| W4 | Fri. 9 Oct. | ■ Phase 1<br>■ Wireframe, mockups |
| W6 | Fri. 23 Oct. | ■ Phase 2 |
| W7 | Fri. 30 Oct. | ■ Report on teamwork |
| W9 | Fri. 13 Nov. | ■ Phase 3<br>■ Link to deployed application with mandatory requirements |
| W11 | Fri. 27 Nov. | ■ Phase 4<br>■ Implementation (almost) finished |
| W14 | Tue. 15 Dec. | ■ Project defence<br>■ Final product demonstration |

# Points of attention

- Regular work is mandatory, respect deadlines

  *75% of the final grade is based on continuous evaluation*

- Organise your team, assign responsibilities

  *Distribute work among team members*

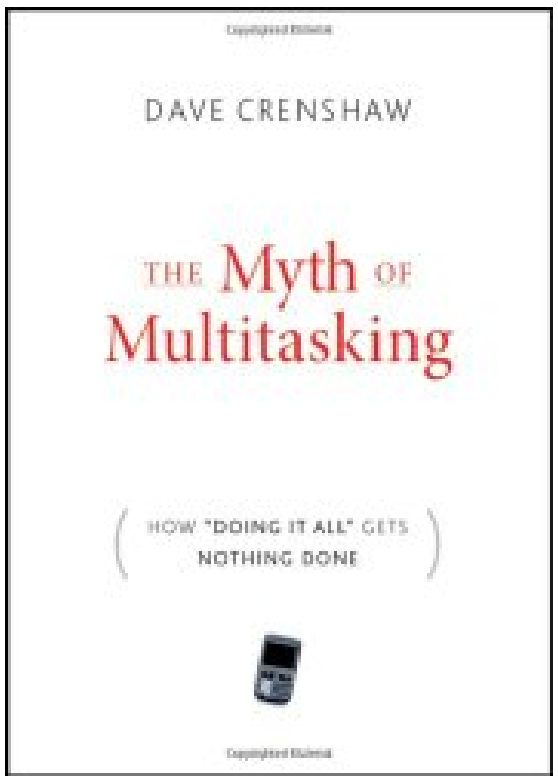
- Reports should contain relevant information for the customer

  *Ensure traceability between reports, take remarks into account*

- **Be critical!**

  *Justify and discuss design decisions and possible alternatives*

# The Myth of Multitasking




ISBN: 978-0-470-37225-8

**The Myth of Multitasking**
by *Dave Crenshaw*

*"Remember this rule:*
*the more responsibility you have,*
*the more hats you wear,*
*the more likely you are to become inefficient."*

# Freedom

- Free to select the development method

  *Waterfall, agile, iterative, lean...*

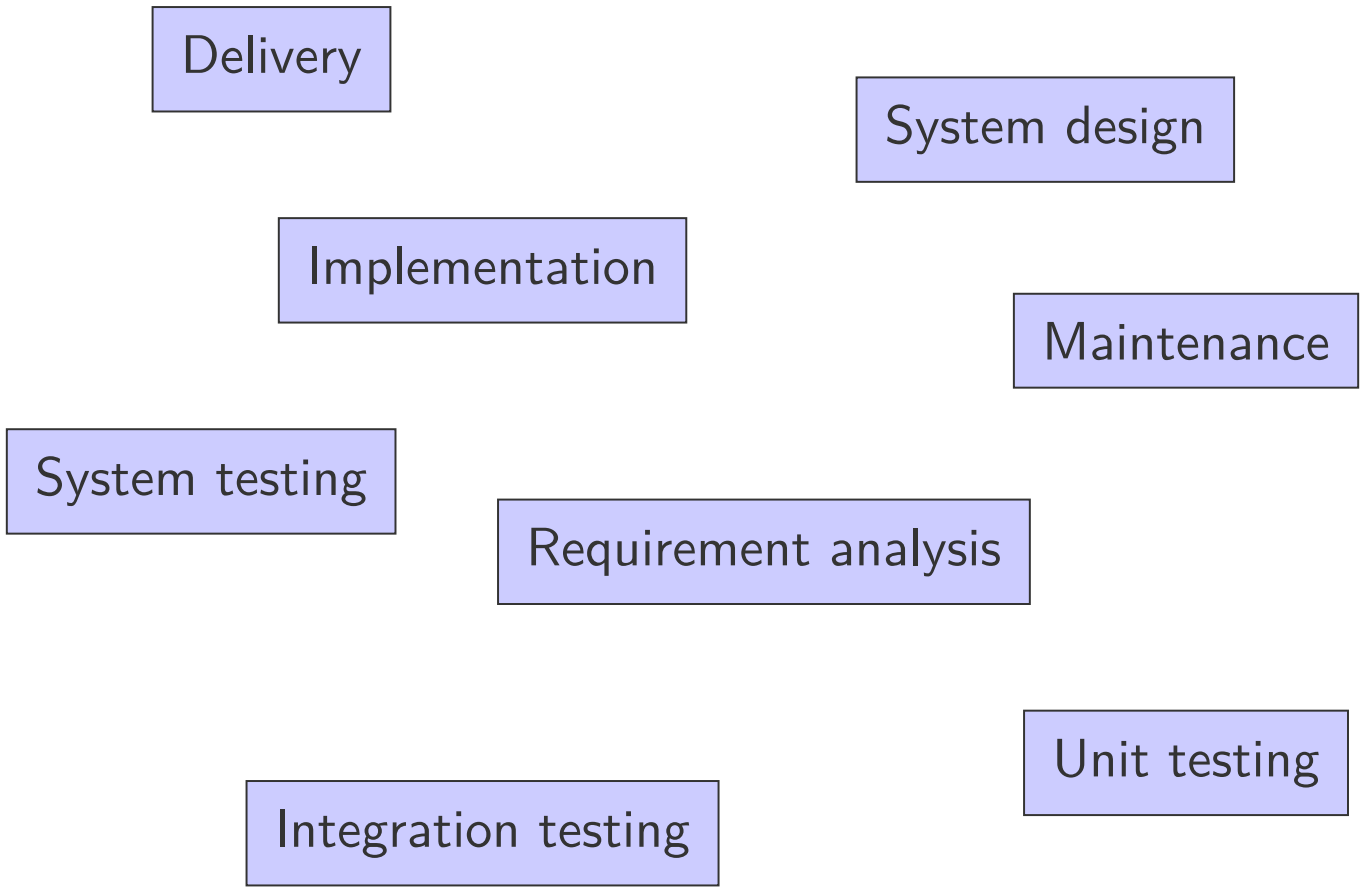
- Free to choose your planning and deadlines

  *Choose what to do for each deadlines, within course constraints*

- Free to select any tools and frameworks

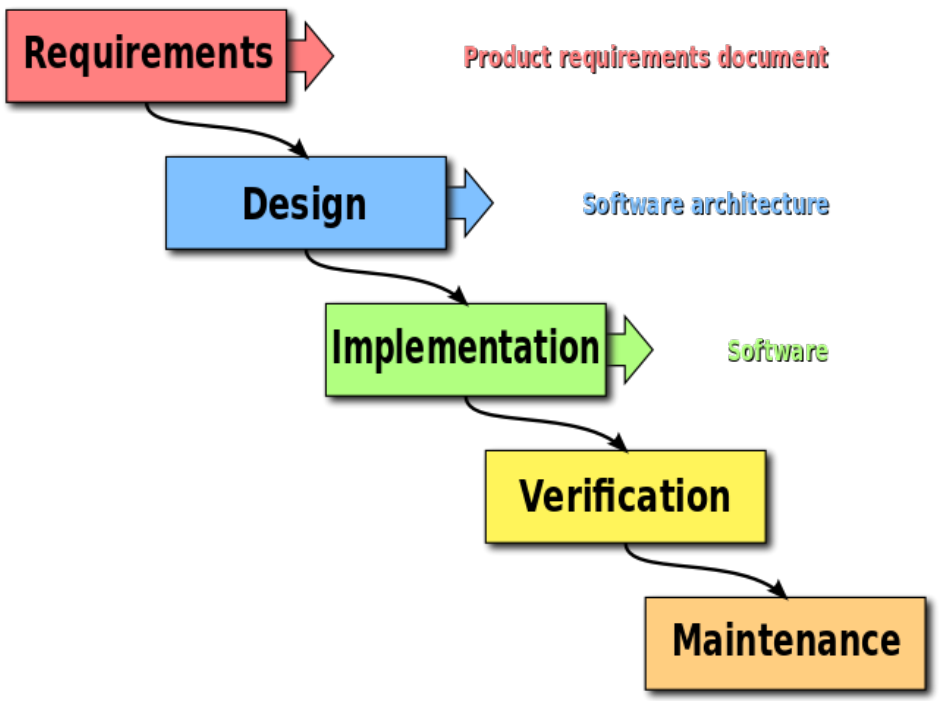
  *Management, development, scheduling... tools*
  *Deployment, testing... frameworks*

# Software development process stages



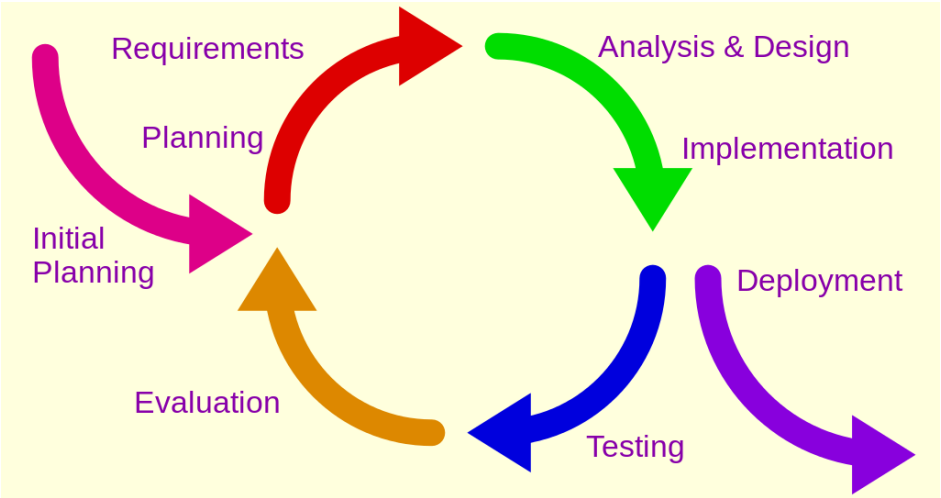
# Waterfall model

- Step-by-step waterfall between various development phases
  - Requirements "frozen" early in the life-cycle
  - Requirements validated too late



# Iterative and incremental model

- More cyclic software development process
- Essential part of other models
  - Rational Unified Process (RUP)
  - eXtreme Programming (XP)
  - Agile software development

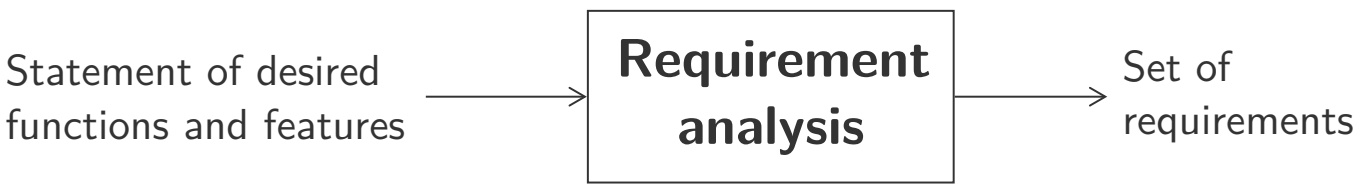


# Requirements analysis

- What are the functional and non-functional requirements?

  *What the system should do*

- Built with information provided by the customer



# Software requirement

- **Definition:** "Statement about what the proposed system will do that all stakeholders agree must be made true in order for the customer's problem to be adequately solved."
  (Lehtbridge & Laganiere, 2001)

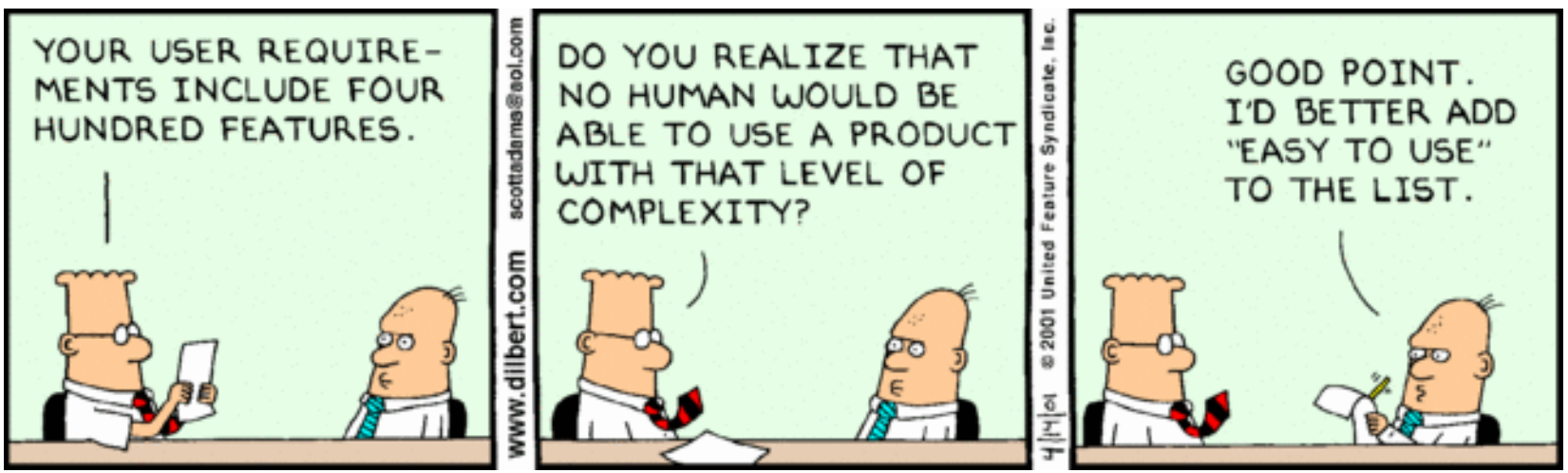
- Important to get the requirements right
  - Wrong requirements main reason of failure of software projects
  - Building the right system VS. building the system right

# Don't be this person!

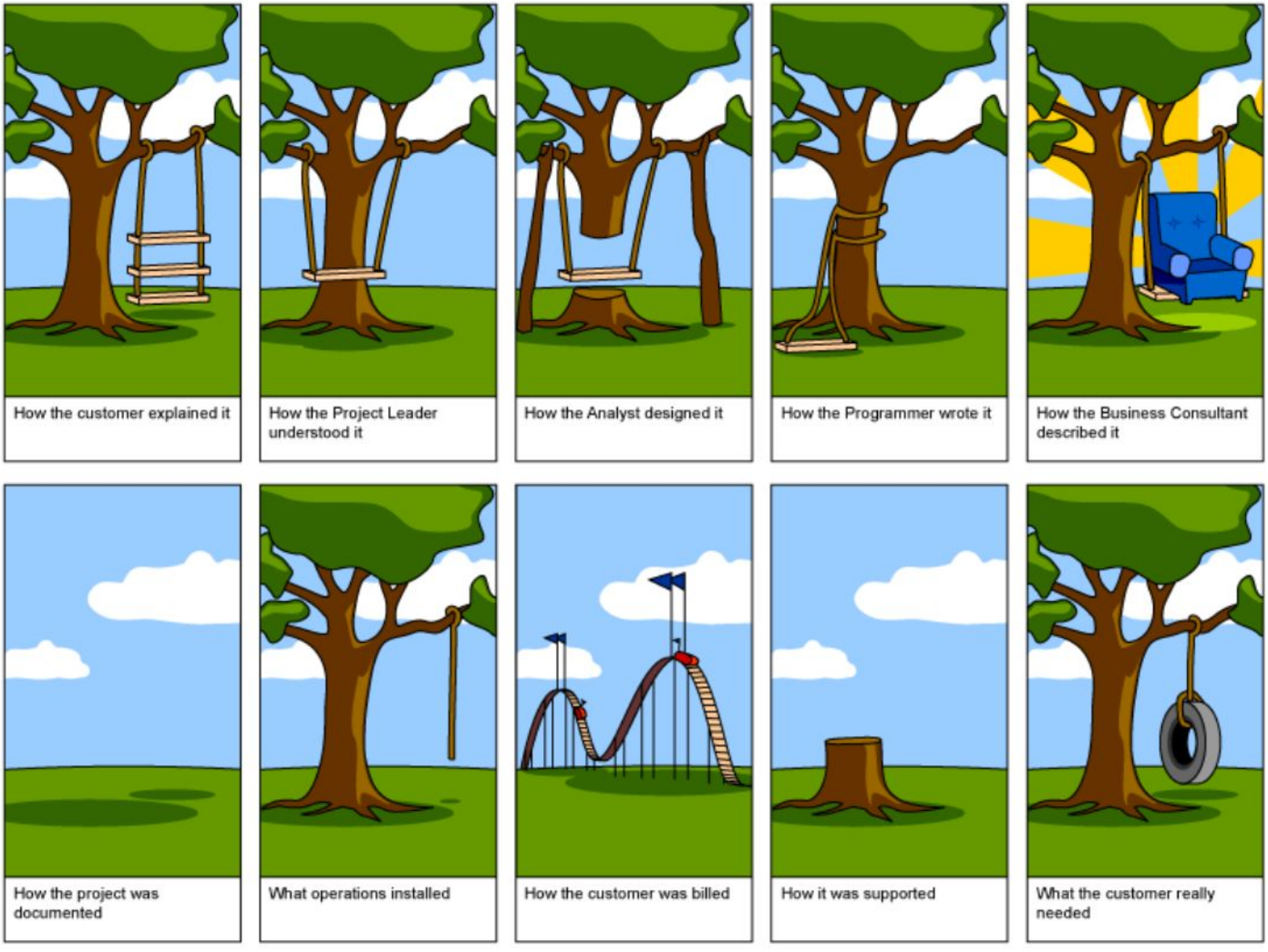
- Hard but important to get the right requirements!

  *Questioning the customer helps in this phase*

- Artefacts can be built to communicate requirements



# How project really works



# Rate of software engineering failures

"To err is human, but to really foul things up you need a computer." —Paul Ehrlich

| | |
|---|---|
| Requirements | Very high |
| Specification | Low |
| Design | Low |
| Implementation | Low |
| Installation | High |
| Operation | Enormous |
| Maintenance | Very high |

- Project team delivers the product without having a clear understanding of what the customer wants
- The customer does not like it because the requirements have not been met
- This leads to project failure

http://spectrum.ieee.org/computing/software/why-software-fails

# Activity diagram

- Models computational or organisational processes

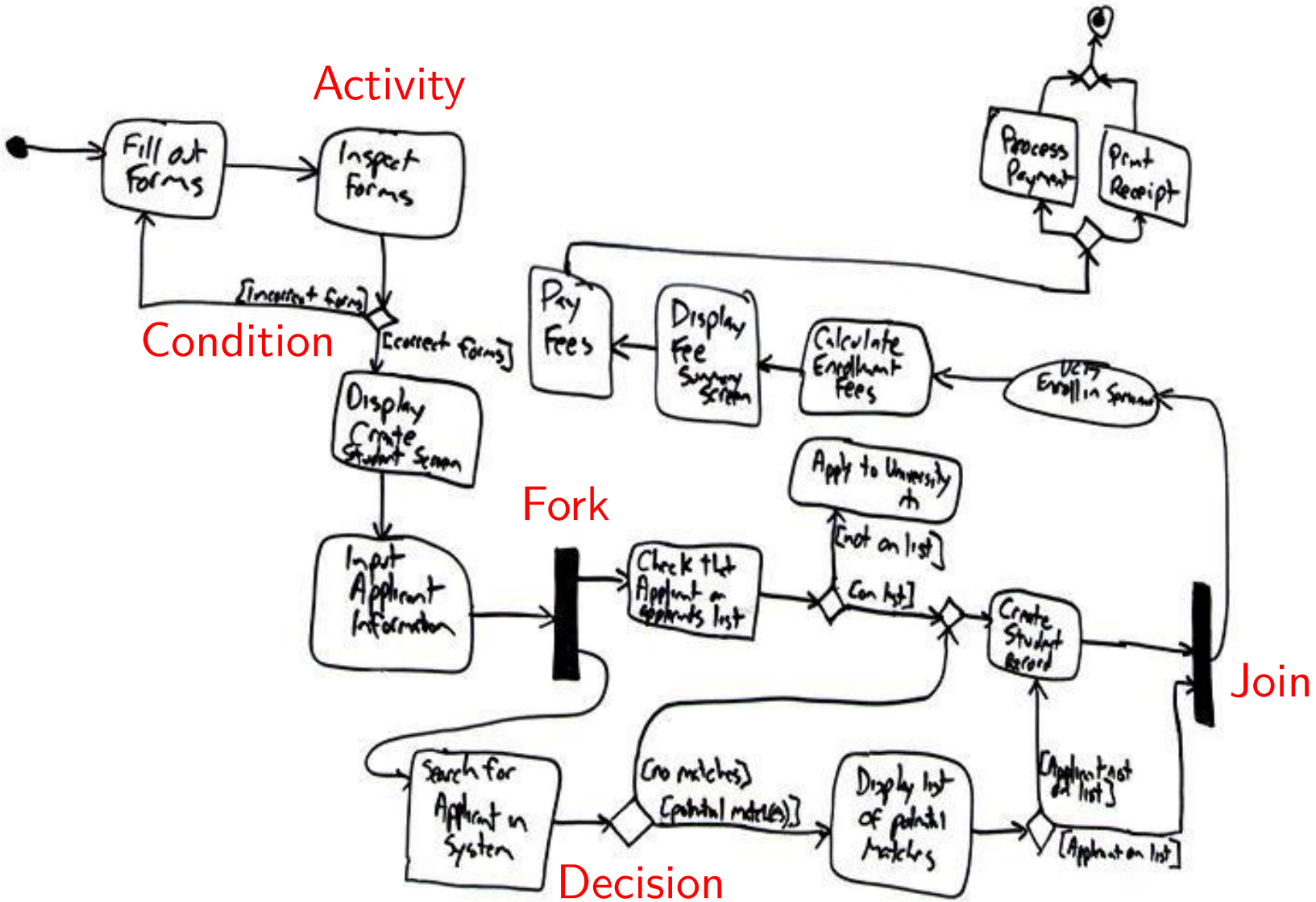
  *High-level view of the behaviour and flow of software system*

- Show the overall flow of control

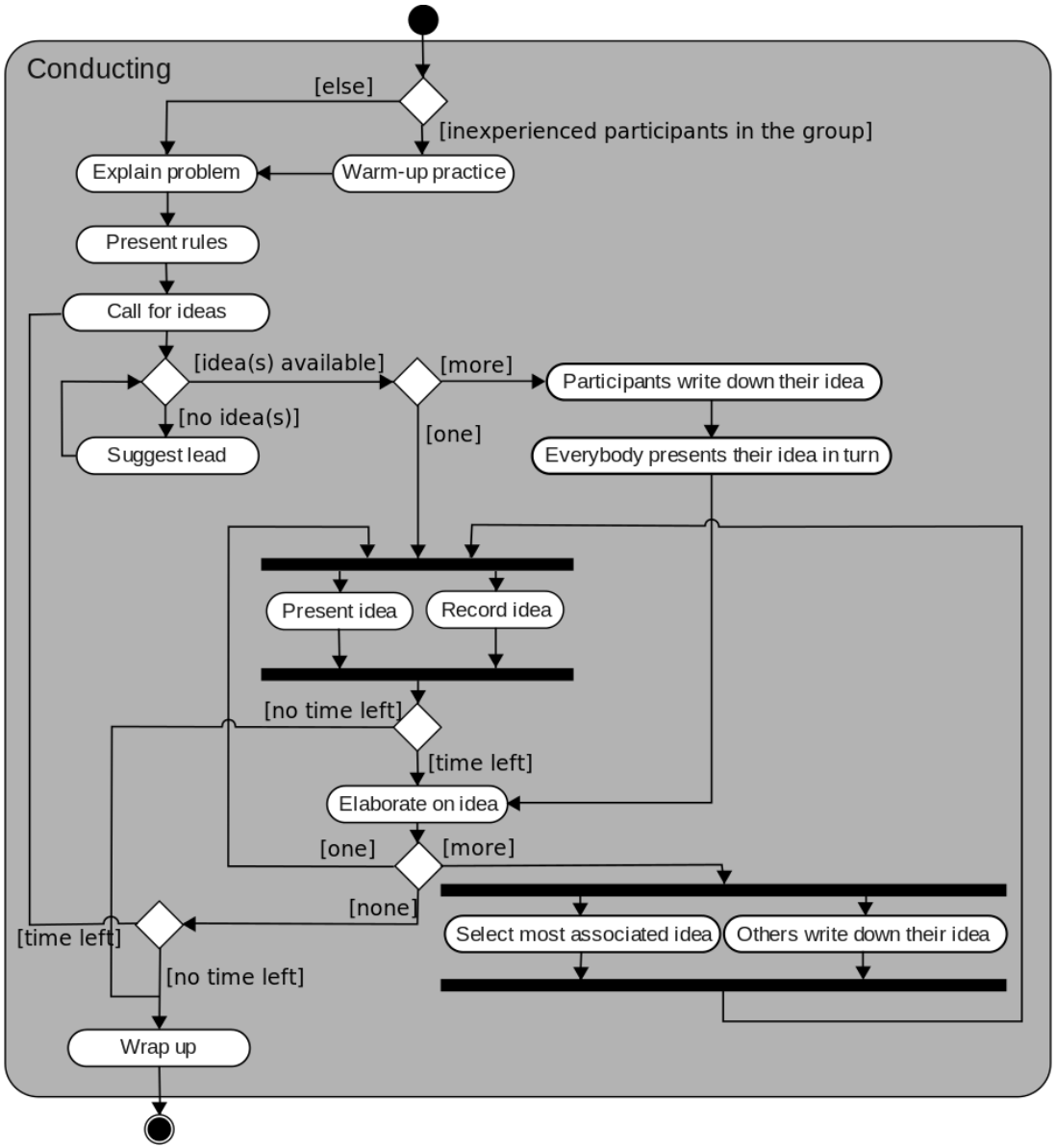
  *Kind of flowchart, with choice, iteration and concurrency*

- Graphical representations of stepwise activities and actions
  - Can be understood by end-users
  - Focus on high-level activities (no implementation details)
  - Show how the control flows from one activity to another

# Enroll in University



# Brainstorming process



# Use case

- Interaction between a role and a system to achieve a goal

  *Captures scenarios that the system should be able to handle*

- Brief bullet points list in natural language

  *List of actions or event steps*

- Contains just enough information to get the idea across

# Enroll in University

**Name:** Enroll in Seminar

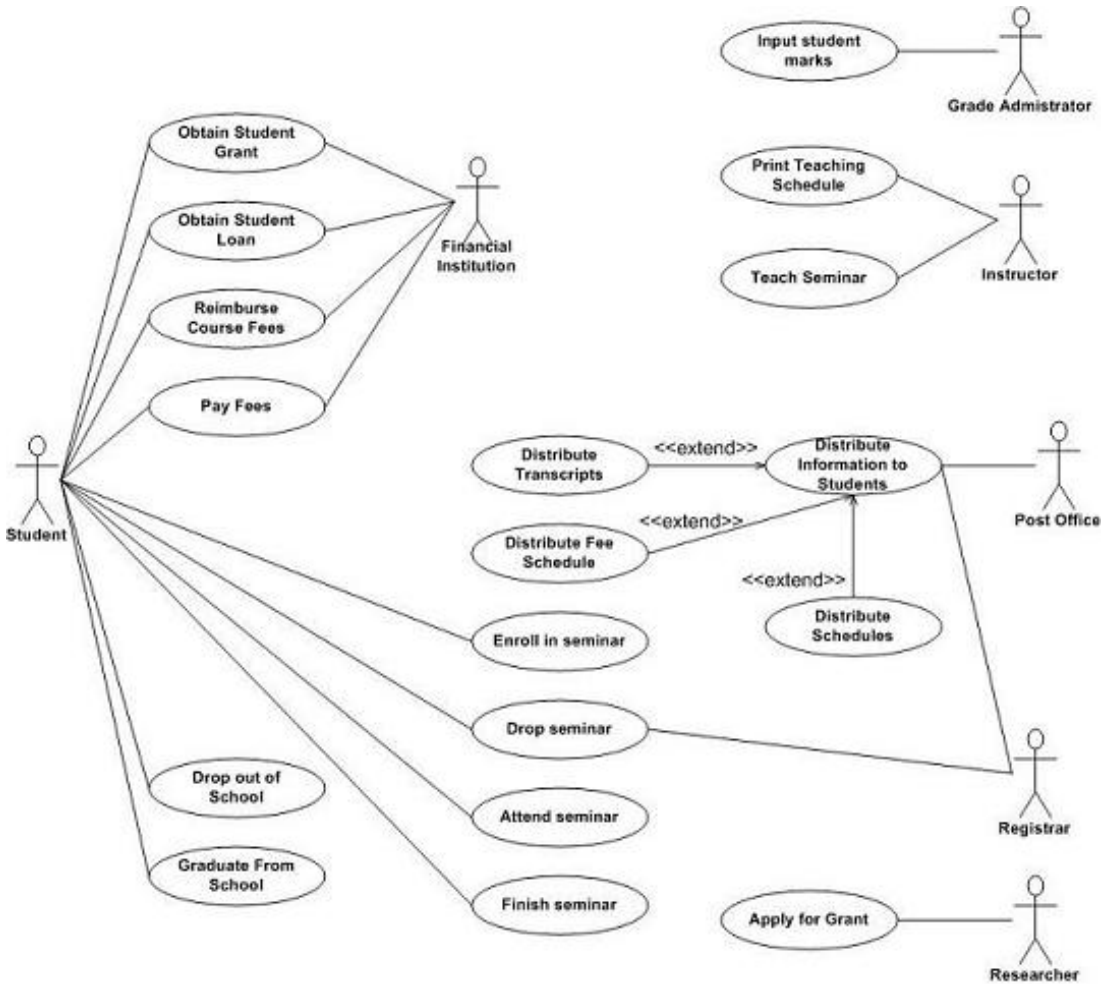
**Identifier:** UC 17

**Basic Course of Action:**

- Student inputs her name and student number
- System verifies the student is eligible to enroll in seminars. If not eligible then the student is informed and use case ends.
- System displays list of available seminars.
- Student chooses a seminar or decides not to enroll at all.
- System validates the student is eligible to enroll in the chosen seminar. If not eligible, the student is asked to choose another.
- System validates the seminar fits into the student's schedule.
- System calculates and displays fees
- Student verifies the cost and either indicates she wants to enroll or not.
- System enrolls the student in the seminar and bills them for it.
- The system prints enrollment receipt.

# Use case diagram

- In conjunction with more textual descriptions of use cases



# Requirements phase

- **Duration:** From September 14, 2015 to September 25, 2015
- **Deliverables:**
  - Chosen software development method
  - Chosen initial planning
  - Report on requirement analysis (functional/non-functional/UI)
- **For the customer:** three use cases for
  - Registration of a player
  - Registration of the owner of a court
  - Creation of a group

# Questions to customer

- Questions about requirements to send to the customer

  *Contact to be taken by the project manager*

- The customer will answer these questions

  *During next course on Wednesday September 23, 2015*

# Todo

- ☐ Register on Moodle

  *LINGI2255 Software Engineering Project*

  http://moodleucl.uclouvain.be/course/view.php?id=7599

- ☐ Form teams, and select project manager

  *Deadline 4pm, Thursday September 17, 2015*

- ☐ Plan weekly meetings with project leader

  *The first one must be before W2 deadline*

- ☐ **Work**, read project specifications carefully

  *Work on requirements analysis phase, contact customer*

# Credits

- Pictures of books from Amazon
- https://openclipart.org/detail/177854/person-icon
- https://en.wikipedia.org/wiki/File:Waterfall_model.svg
- https://en.wikipedia.org/wiki/File:Iterative_development_model.svg
- http://adamstacoviak.com/dilbert-user-requirements-four-hundred-features/
- http://www.w-uh.com/posts/090823-how_projects_work.html
- http://www.agilemodeling.com/artifacts/activityDiagram.htm
- https://en.wikipedia.org/wiki/File:Activity_conducting.svg
- http://agilemodeling.com/artifacts/systemUseCase.htm
- http://agilemodeling.com/artifacts/useCaseDiagram.htm