Software Engineering # 1 Introduction

Zakia KAZI-AOUL
zkazi@isep.fr

28/01/2013

5 technical courses in the information systems track

- **II.2403** Advanced Databases
- ***II.2404** Object-Oriented Design and Programming*
- **II.2405** Software engineering
- **II.2406** Advanced WEB Programming
- ***II.2408** Operating Systems and Administration*

6 technical courses in the software engineer track

- **II.2403** Advanced Databases
- ***II.2404** Object-Oriented Design and Programming*
- **II.2405** Software engineering
- ***II.2407** Operating Systems and Programming*
- ***II.2410** System Programming (C Language)*
- ***IG.2408** numerical methods*

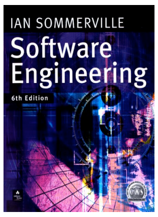
Course global objectives

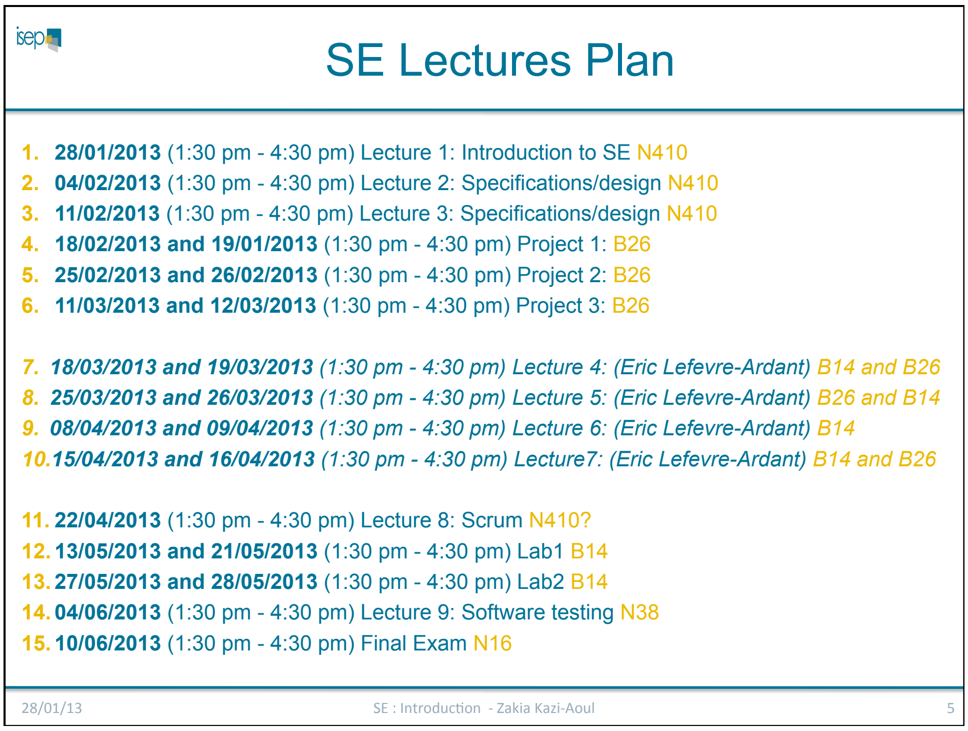
- **Software Engineering** provides:
  - Appropriate tools to conduct a project during the software design phases
  - Design, program, test, deploy, maintain a system
  - Methodologies and best practices in project management

Recommended Course Textbooks

- **[1] Software Engineering: Theory and Practice**, Shari Lawrence Pfleeger Joanne M. Atlee**,** 4th Edition, Pearson Higher Education, 2009

- **[2] Software Engineering**, Sommerville Ian, 7th Edition, Addison-Wesley, Harlow, Essex,UK, 2004



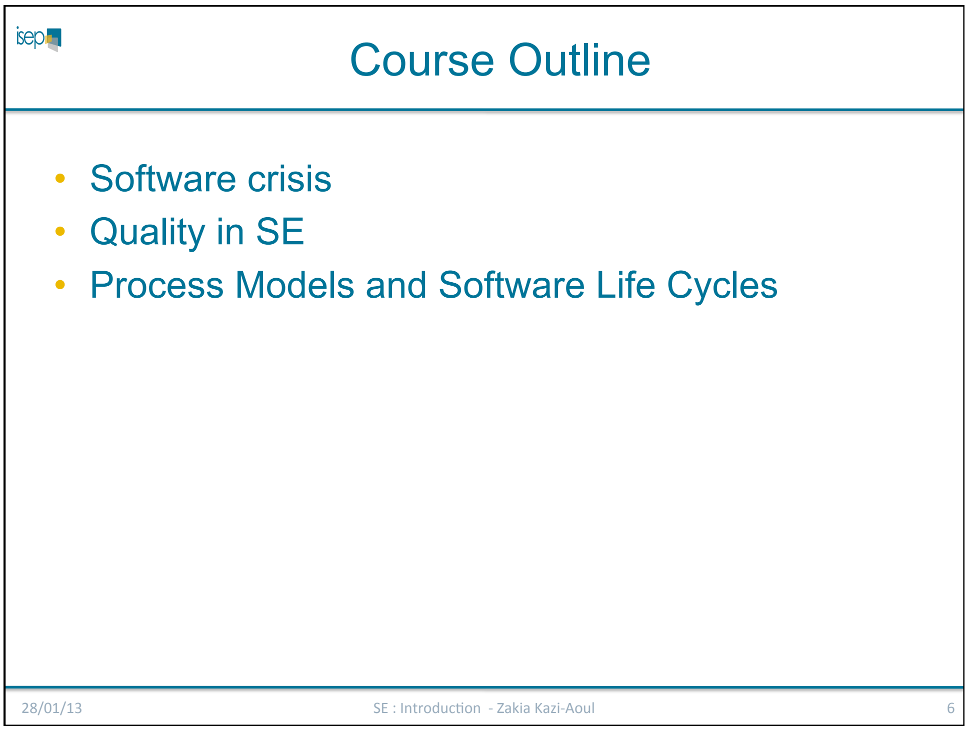


SE Lectures Plan

1. **28/01/2013** (1:30 pm - 4:30 pm) Lecture 1: Introduction to SE N410
2. **04/02/2013** (1:30 pm - 4:30 pm) Lecture 2: Specifications/design N410
3. **11/02/2013** (1:30 pm - 4:30 pm) Lecture 3: Specifications/design N410
4. **18/02/2013 and 19/01/2013** (1:30 pm - 4:30 pm) Project 1: B26
5. **25/02/2013 and 26/02/2013** (1:30 pm - 4:30 pm) Project 2: B26
6. **11/03/2013 and 12/03/2013** (1:30 pm - 4:30 pm) Project 3: B26

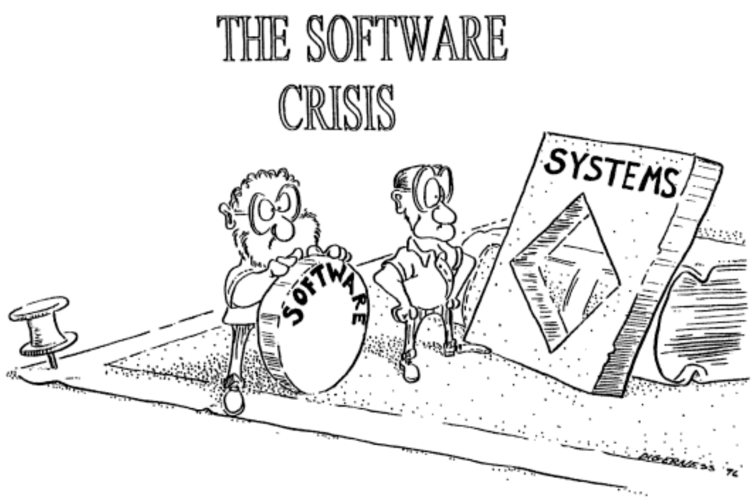
7. ***18/03/2013 and 19/03/2013*** *(1:30 pm - 4:30 pm) Lecture 4: (Eric Lefevre-Ardant) B14 and B26*
8. ***25/03/2013 and 26/03/2013*** *(1:30 pm - 4:30 pm) Lecture 5: (Eric Lefevre-Ardant) B26 and B14*
9. ***08/04/2013 and 09/04/2013*** *(1:30 pm - 4:30 pm) Lecture 6: (Eric Lefevre-Ardant) B14*
10. ***15/04/2013 and 16/04/2013*** *(1:30 pm - 4:30 pm) Lecture7: (Eric Lefevre-Ardant) B14 and B26*

11. **22/04/2013** (1:30 pm - 4:30 pm) Lecture 8: Scrum N410?
12. **13/05/2013 and 21/05/2013** (1:30 pm - 4:30 pm) Lab1 B14
13. **27/05/2013 and 28/05/2013** (1:30 pm - 4:30 pm) Lab2 B14
14. **04/06/2013** (1:30 pm - 4:30 pm) Lecture 9: Software testing N38
15. **10/06/2013** (1:30 pm - 4:30 pm) Final Exam N16



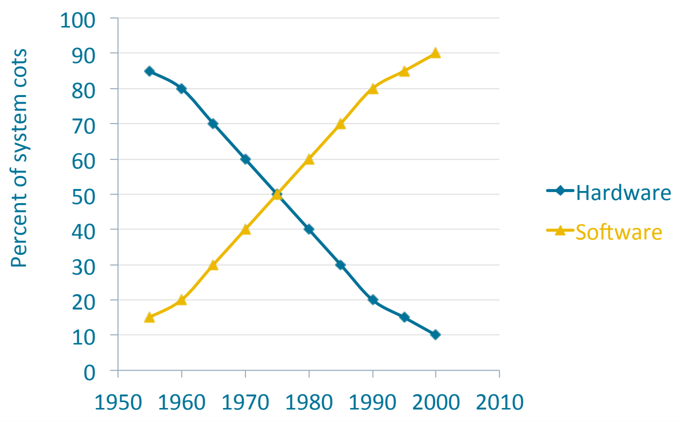
Course Outline

- Software crisis
- Quality in SE
- Process Models and Software Life Cycles



The “Software Crisis” (1)

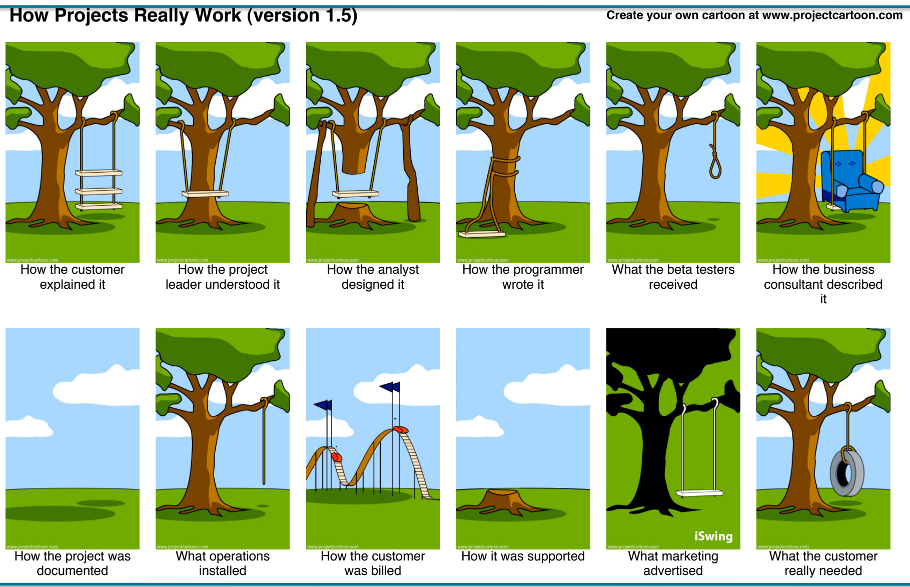
- 1968 : Software Crisis Conference (NATO)



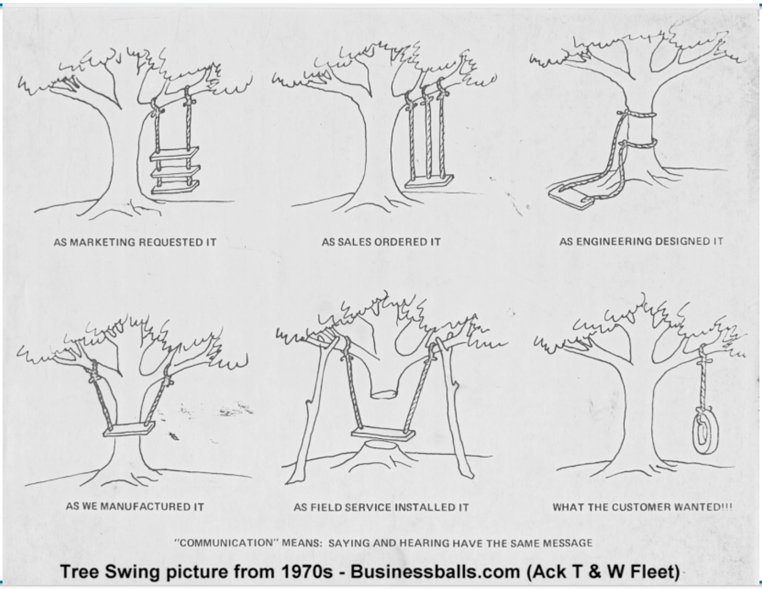
The "Software Crisis" (2)

- Informal software development may cause :
  - Projects running over-budget
  - Projects running over-time
  - Software was very inefficient
  - Software was of low quality
  - Software often did not meet requirements
  - Projects were unmanageable and code difficult to maintain
  - Software was never delivered

The swing project

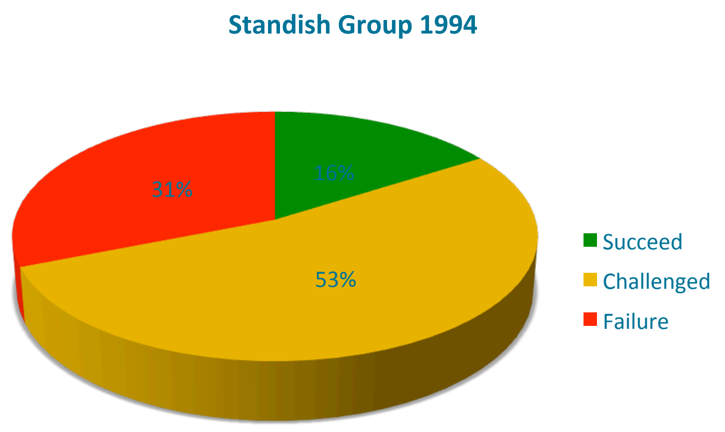

How Projects Really Work (version 1.5)
Create your own cartoon at www.projectcartoon.com
How the customer explained it
How the project leader understood it
How the analyst designed it
How the programmer wrote it
What the beta testers received
How the business consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
iSwing
What marketing advertised
What the customer really needed

http://www.businessballs.com/images/treeswing/tree_swing_70s.jpeg


AS MARKETING REQUESTED IT
AS SALES ORDERED IT
AS ENGINEERING DESIGNED IT
AS WE MANUFACTURED IT
AS FIELD SERVICE INSTALLED IT
WHAT THE CUSTOMER WANTED!!!
"COMMUNICATION" MEANS: SAYING AND HEARING HAVE THE SAME MESSAGE

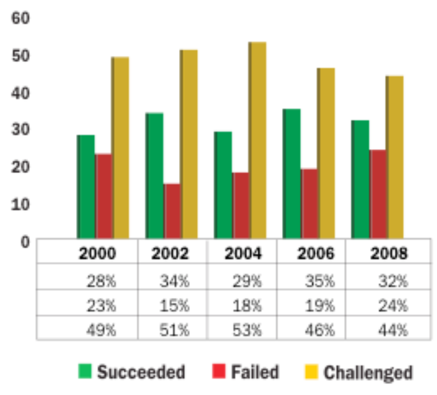
Tree Swing picture from 1970s - Businessballs.com (Ack T & W Fleet)

The Standish Group Chaos Summary 1994



http://www.infoq.com/articles/Interview-Johnson-Standish-CHAOS

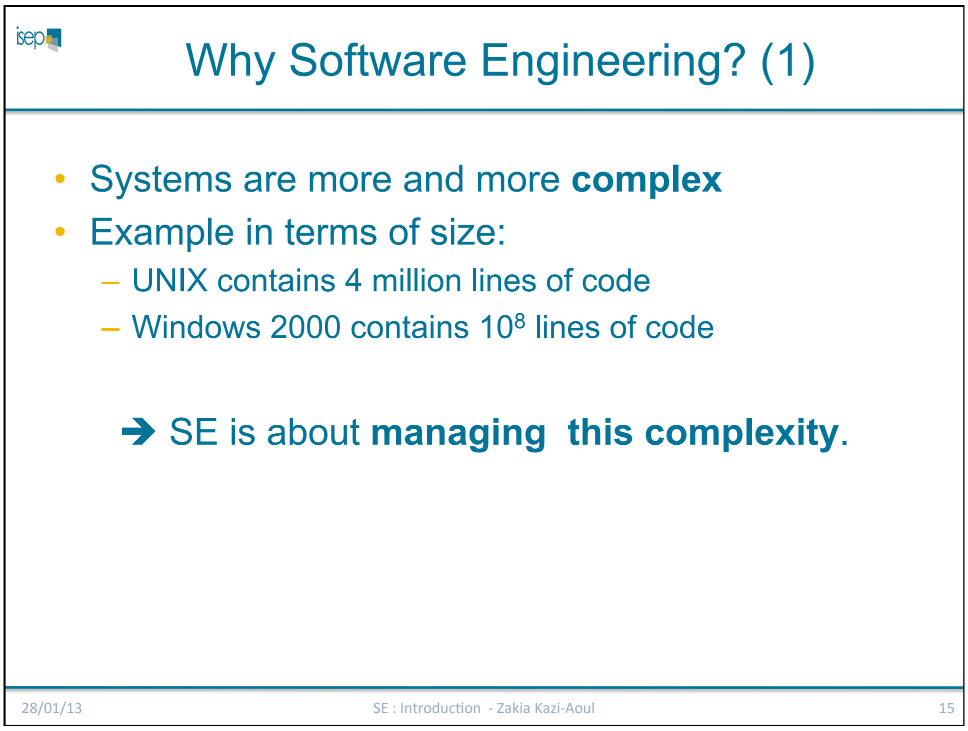
The Standish Group Chaos Summary 2009



2000	2002	2004	2006	2008
28%	34%	29%	35%	32%
23%	15%	18%	19%	24%
49%	51%	53%	46%	44%

Is the Standish study biased?

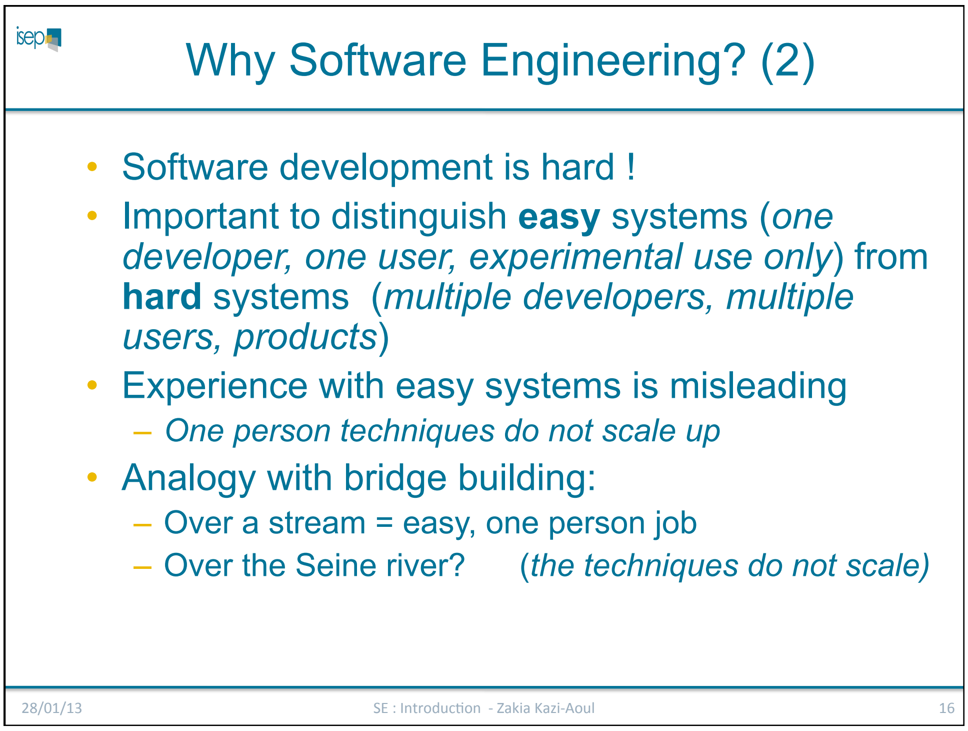
- The Standish report: does it really describe a software crisis?
  - http://www.uio.no/studier/emner/matnat/ifi/INF5180/v10/undervisningsmateriale/reading-materials/p01/glass.pdf



Why Software Engineering? (1)

- Systems are more and more **complex**
- Example in terms of size:
  - UNIX contains 4 million lines of code
  - Windows 2000 contains 10^8 lines of code

➔ SE is about **managing this complexity**.



Why Software Engineering? (2)

- Software development is hard !
- Important to distinguish **easy** systems (*one developer, one user, experimental use only*) from **hard** systems (*multiple developers, multiple users, products*)
- Experience with easy systems is misleading
  - *One person techniques do not scale up*
- Analogy with bridge building:
  - Over a stream = easy, one person job
  - Over the Seine river? (*the techniques do not scale)*

Software Engineering : Definition 1

- Software engineering (SE) is an engineering discipline whose focus is the **cost effective** development of **high-quality** software systems
  - *Source [2]*

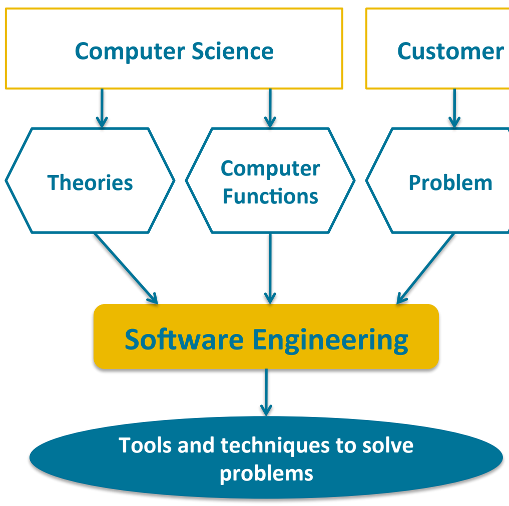
Software Costs

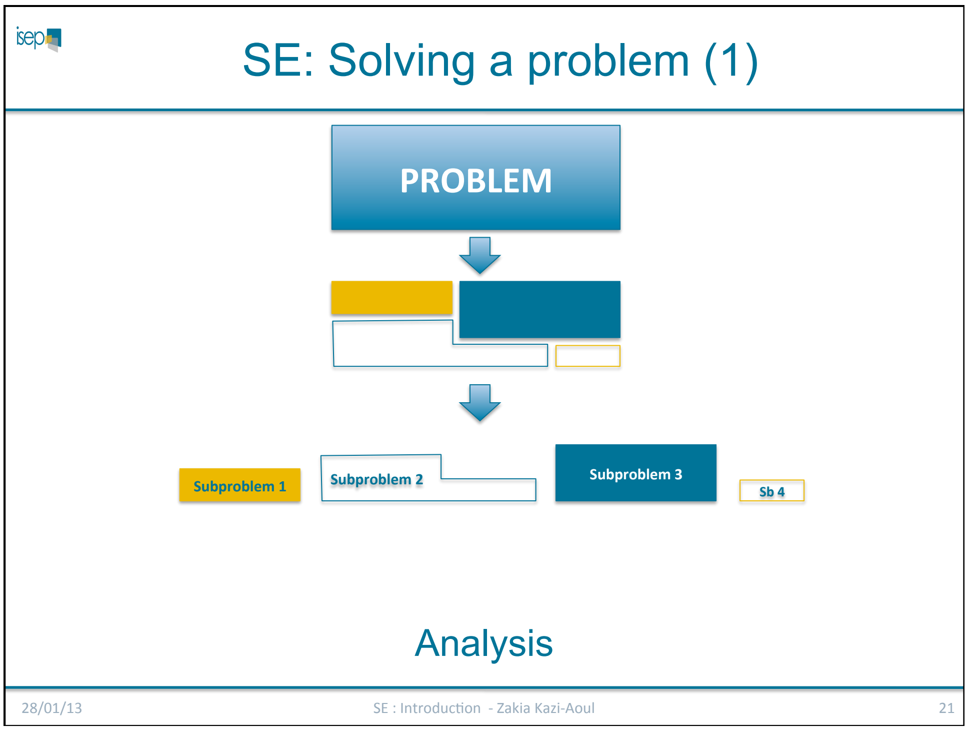
- Software costs often **dominate** hardware costs.
  - The costs of software on a PC are often greater than the hardware cost.
- Software costs more to maintain than it does to develop.
  - For systems with a long life, maintenance costs may be several times development costs.
- Software engineering is concerned with cost-effective software development.

Software Engineering : Definition 2

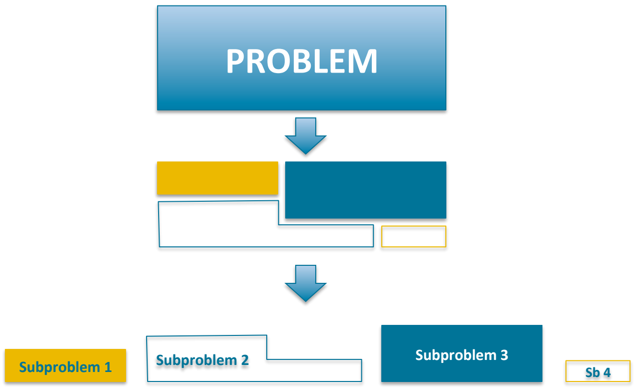
- SE is an **engineering discipline** that is concerned with **all aspects of software production** from the early stages of system specification to maintaining the system after is has gone into use
  - *Source [2]*

Role of SE

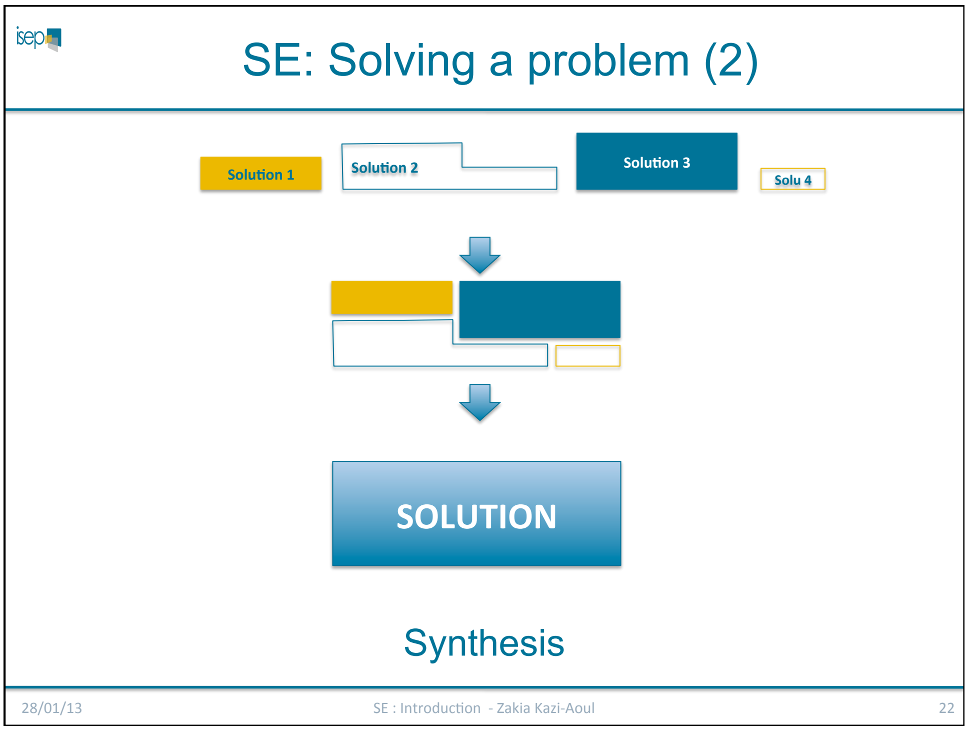




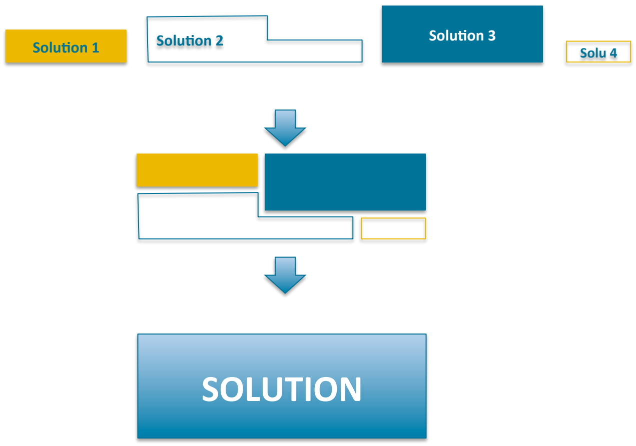
SE: Solving a problem (1)



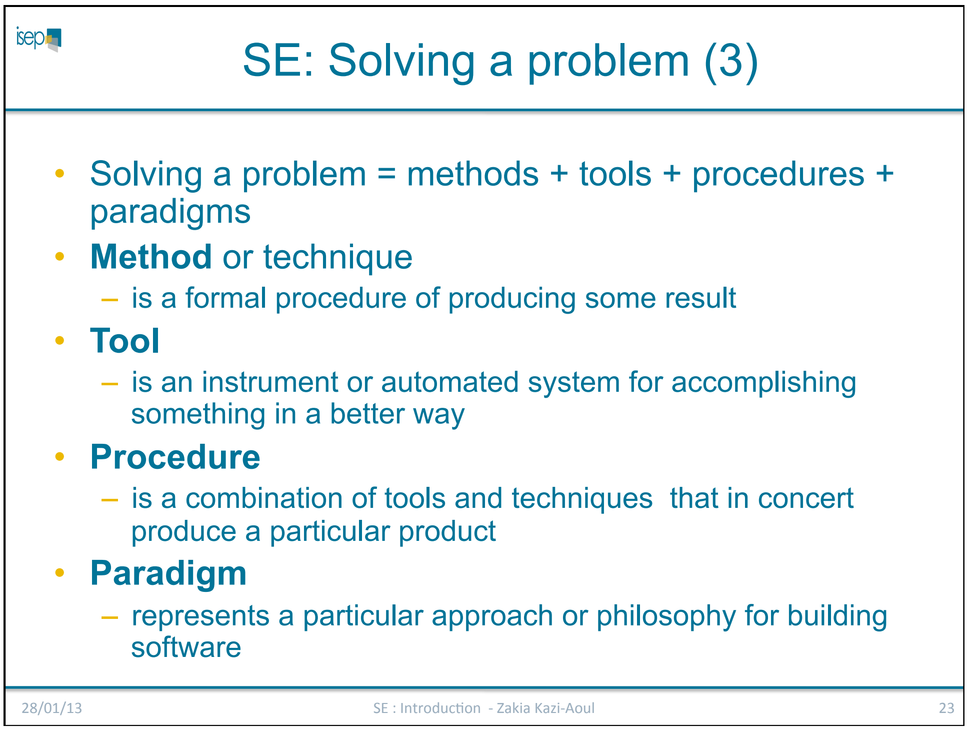
Analysis



SE: Solving a problem (2)

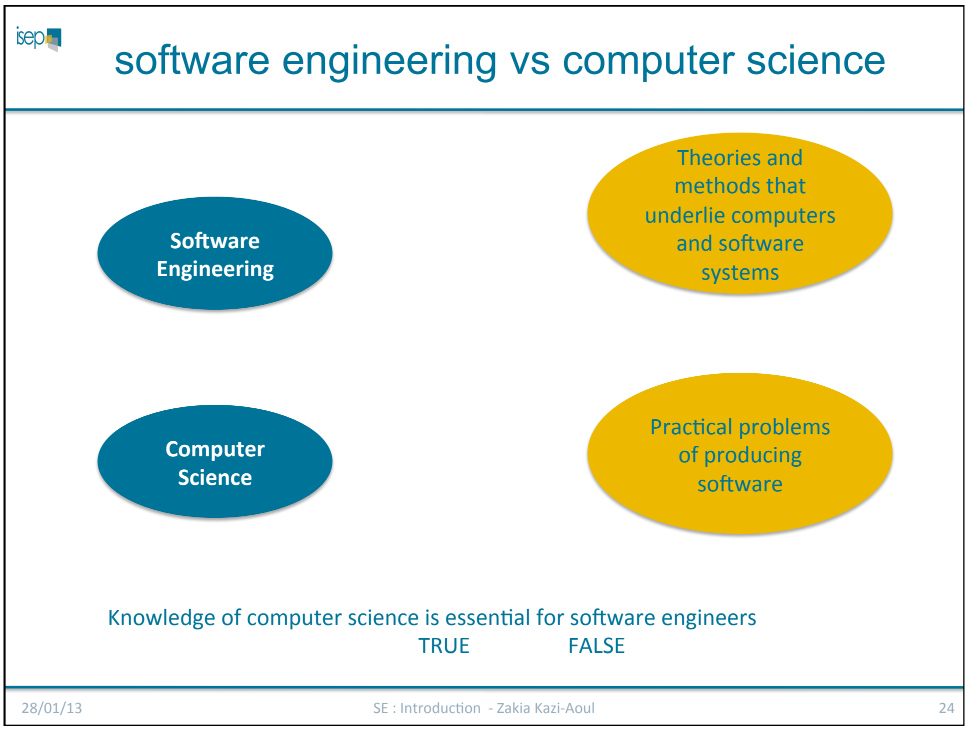


Synthesis



SE: Solving a problem (3)

- Solving a problem = methods + tools + procedures + paradigms
- **Method** or technique
  - is a formal procedure of producing some result
- **Tool**
  - is an instrument or automated system for accomplishing something in a better way
- **Procedure**
  - is a combination of tools and techniques that in concert produce a particular product
- **Paradigm**
  - represents a particular approach or philosophy for building software



software engineering vs computer science

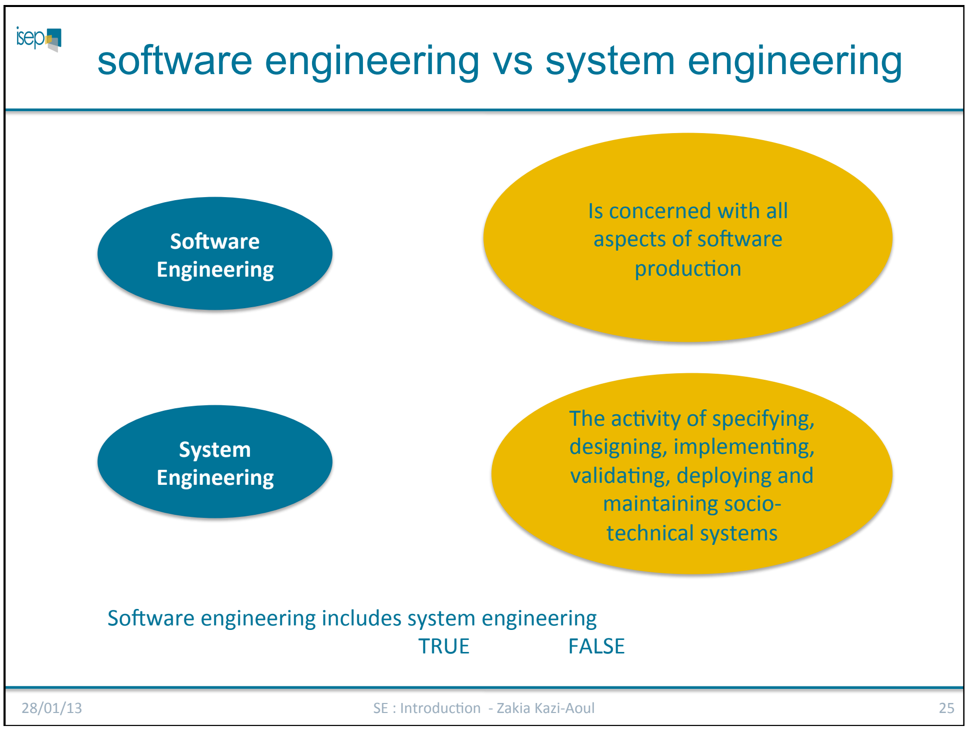
Software Engineering

Theories and methods that underlie computers and software systems

Computer Science

Practical problems of producing software

Knowledge of computer science is essential for software engineers
TRUE FALSE



software engineering vs system engineering

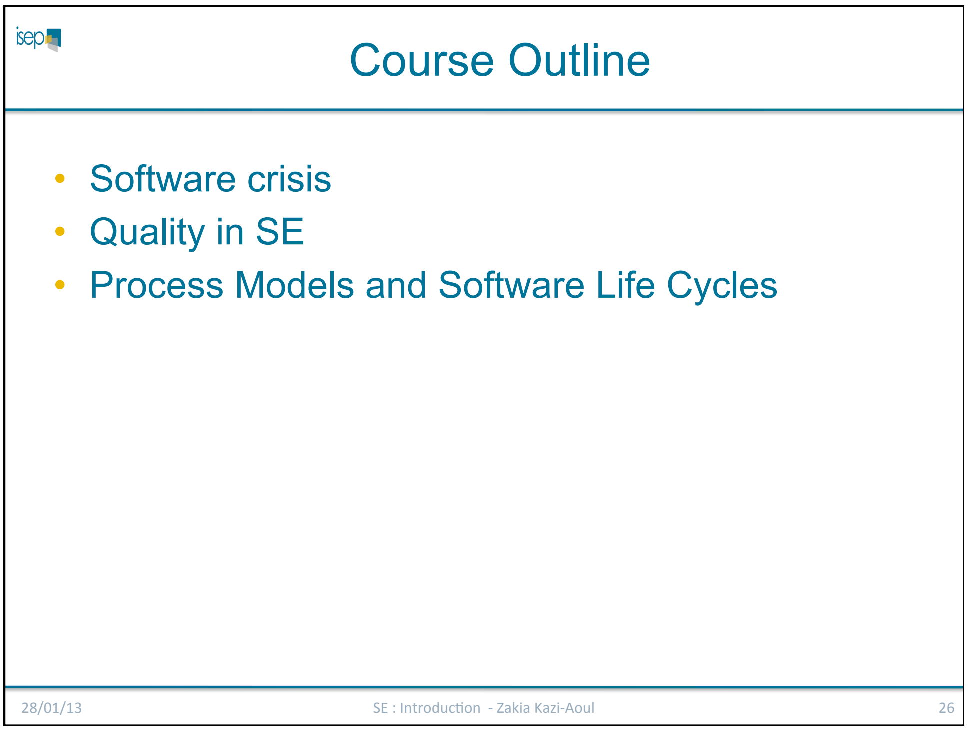
Software Engineering

Is concerned with all aspects of software production

System Engineering

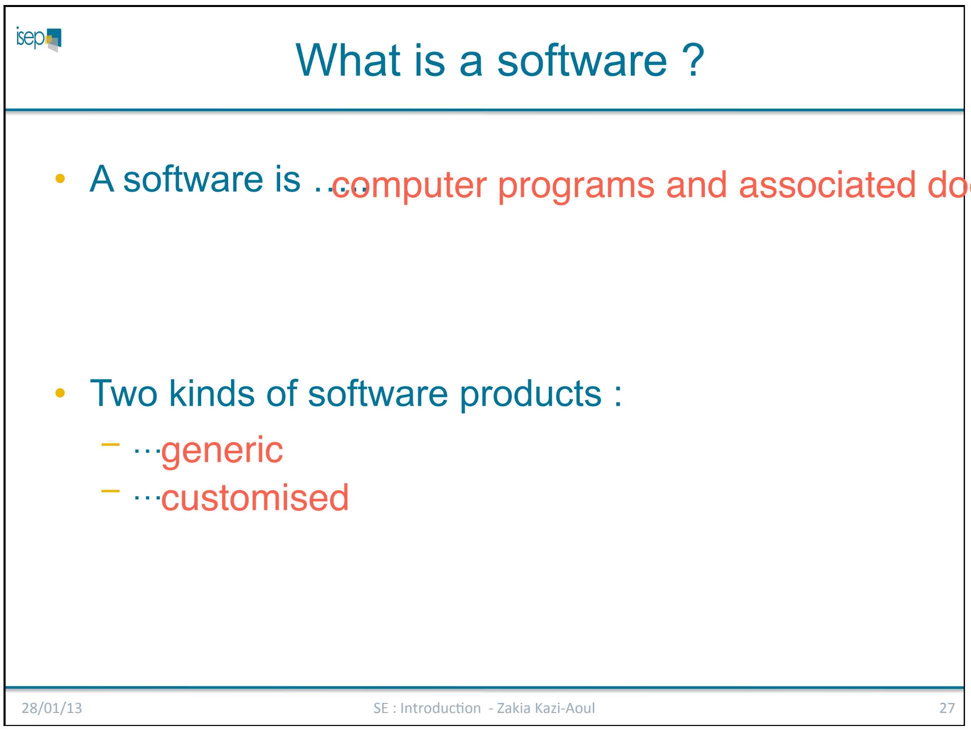
The activity of specifying, designing, implementing, validating, deploying and maintaining socio-technical systems

Software engineering includes system engineering
TRUE FALSE



Course Outline

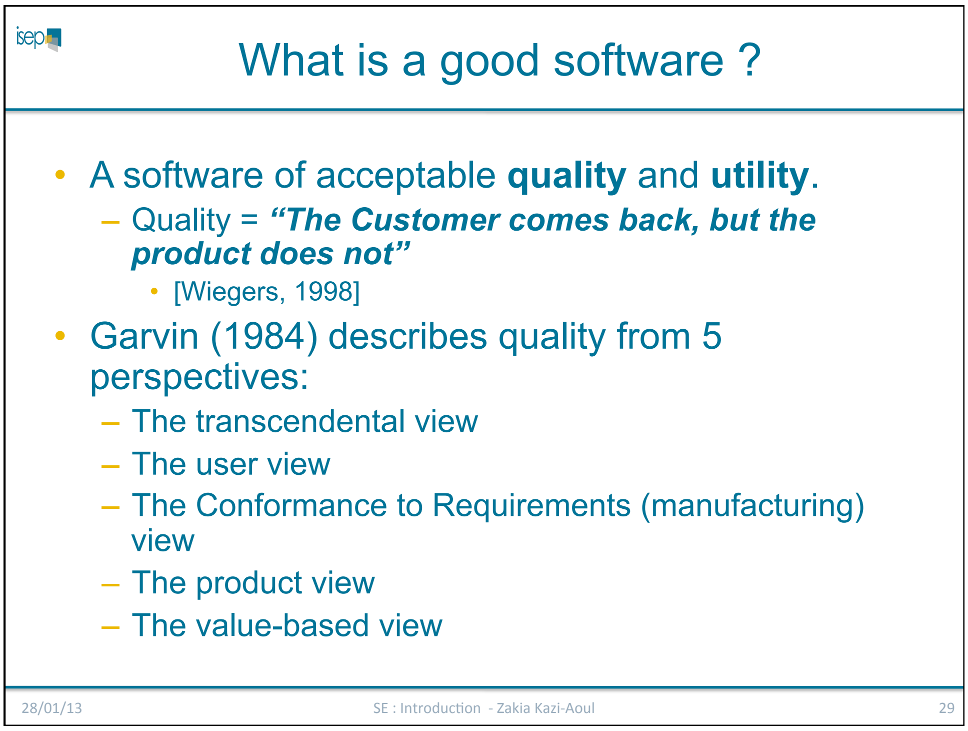
- Software crisis
- Quality in SE
- Process Models and Software Life Cycles



What is a software ?

- A software is ……computer programs and associated documents
- Two kinds of software products :
  - …generic
  - …customised

28/01/13 SE : Introduction - Zakia Kazi-Aoul 27

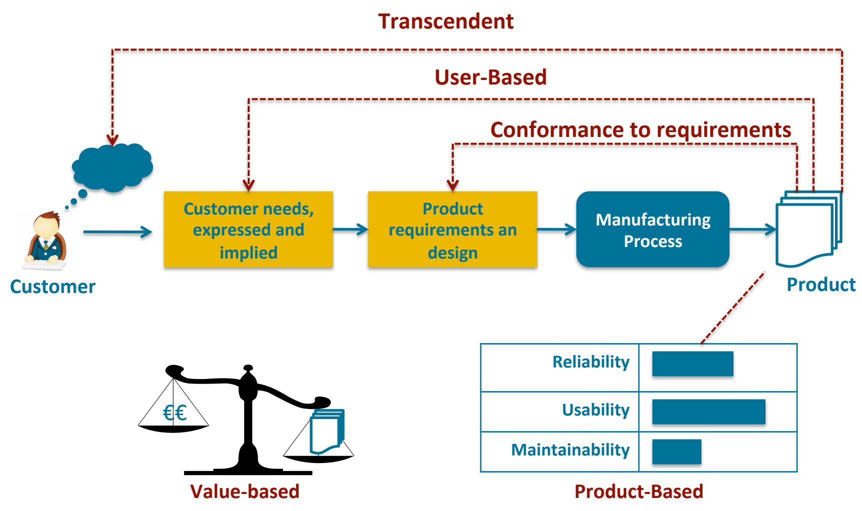


What is a good software ?

- A software of acceptable **quality** and **utility**.
  - Quality = ***"The Customer comes back, but the product does not"***
    - [Wiegers, 1998]
- Garvin (1984) describes quality from 5 perspectives:
  - The transcendental view
  - The user view
  - The Conformance to Requirements (manufacturing) view
  - The product view
  - The value-based view

28/01/13 SE : Introduction - Zakia Kazi-Aoul 29

Garvin Views



Quality vs Software quality [IEEE]

- Quality:
  - The degree to which a system, component, or process meets
    - Specified requirements
    - Customer or user needs
- Software quality:
  - The degree to which software processes a desired combination of attributes

Product quality is relative

	Marketing Priorities Usability: **Important** Maintainability: **Neutral**	Development Priorities Usability: **Neutral** Maintainability: **Important**
Product 1 Attributes Usability: **Low** Maintainability: **High**	**Low Quality**	**High Quality**
Product 2 Attributes Usability: **High** Maintainability: **Low**	**High Quality**	**Low Quality**

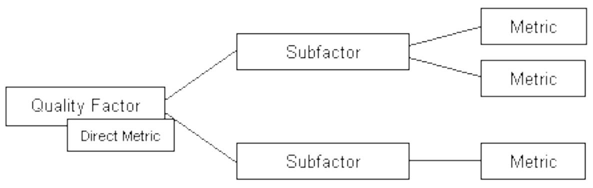
Software quality models (1)

http://programminglarge.com/software_quality_management

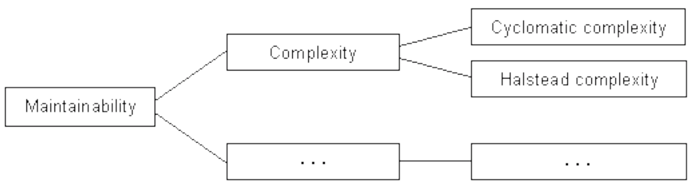
- Provides a framework for expressing and measuring software quality as a collection of **desirable attributes**
- Measures quality in software systems by :
  - Specifying **quality characteristics**
  - Their **relationships**
  - Associated **metrics**
- Many models
  - McCall
  - Boehm
  - ISO/IEC 9126
  - …

Software quality models (2)

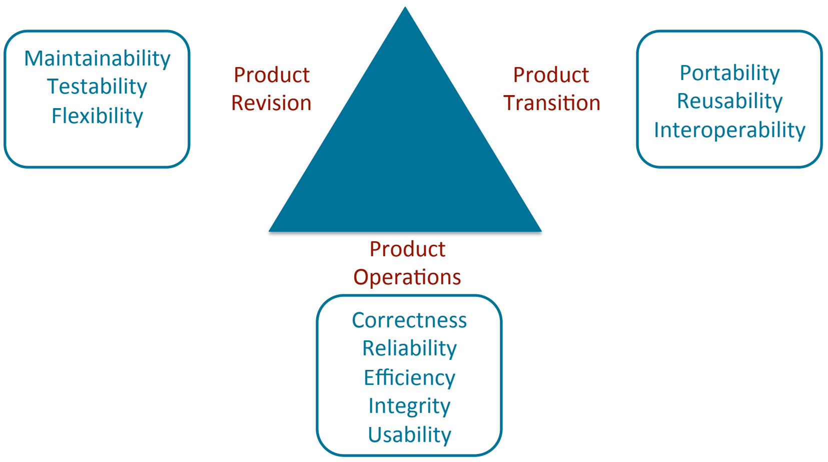
Framework…



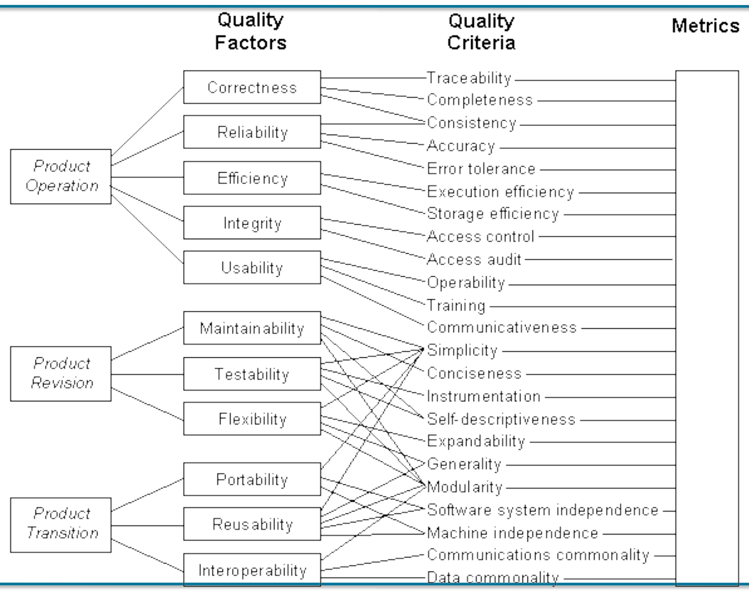
Example…



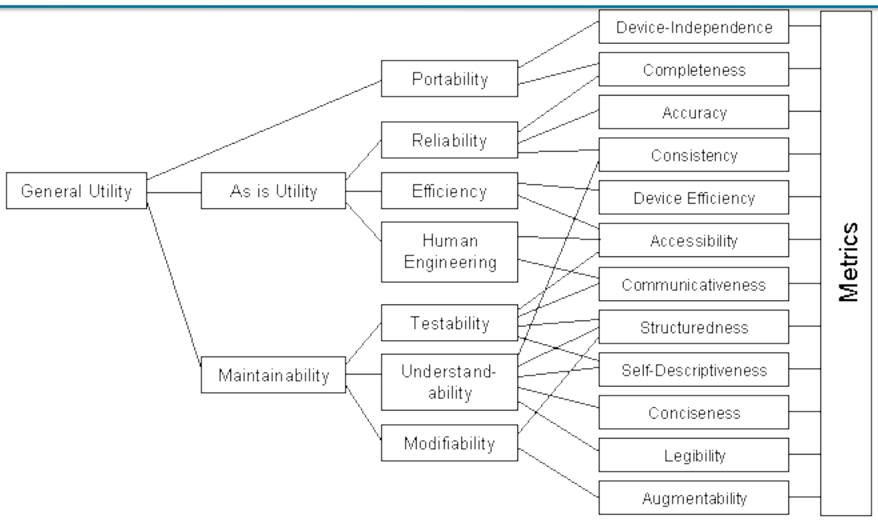
McCall's quality model (1)

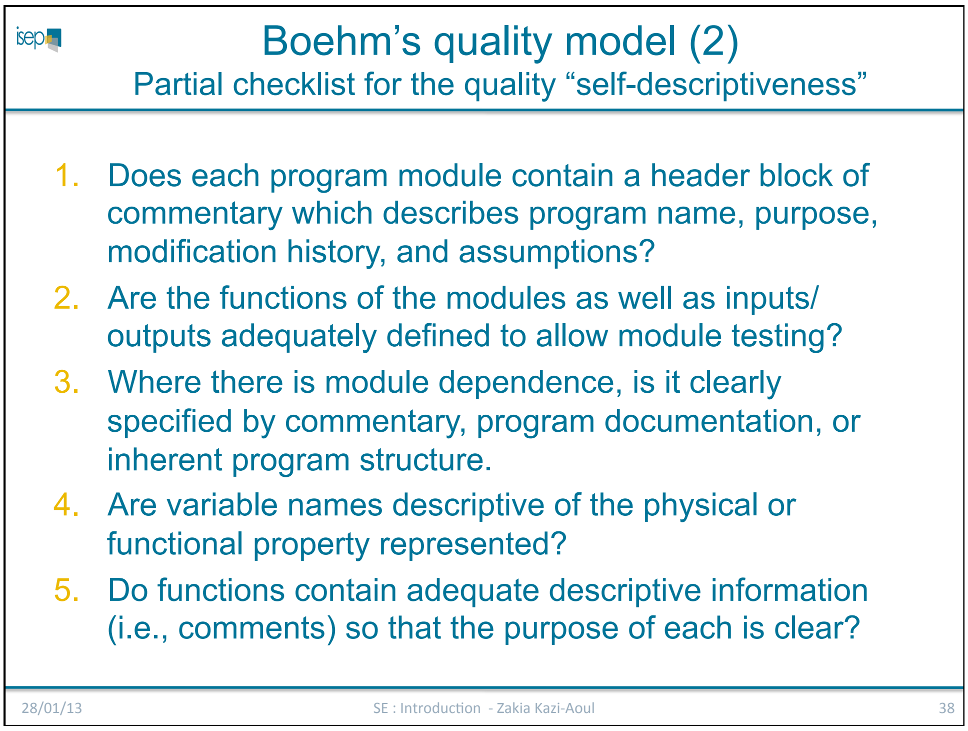


McCall's quality model (2)



Boehm's quality model (1)



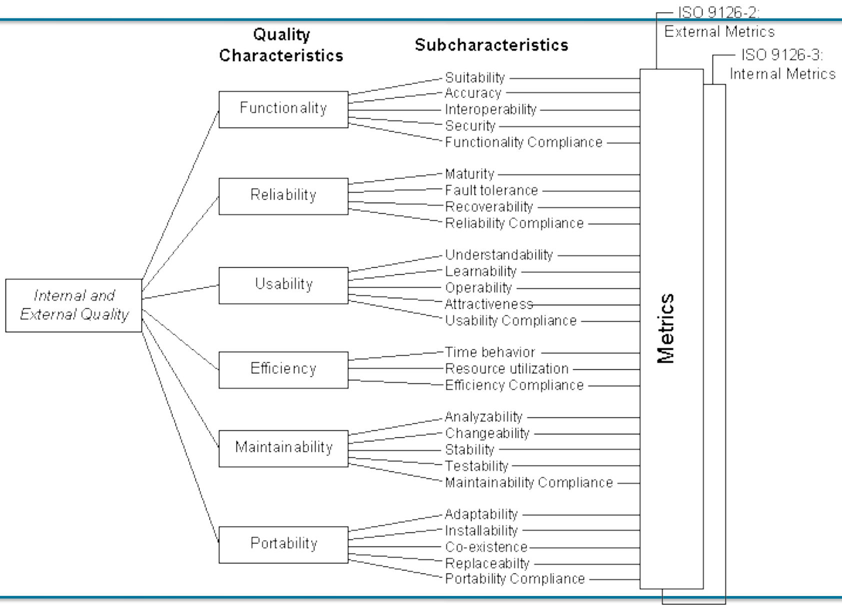


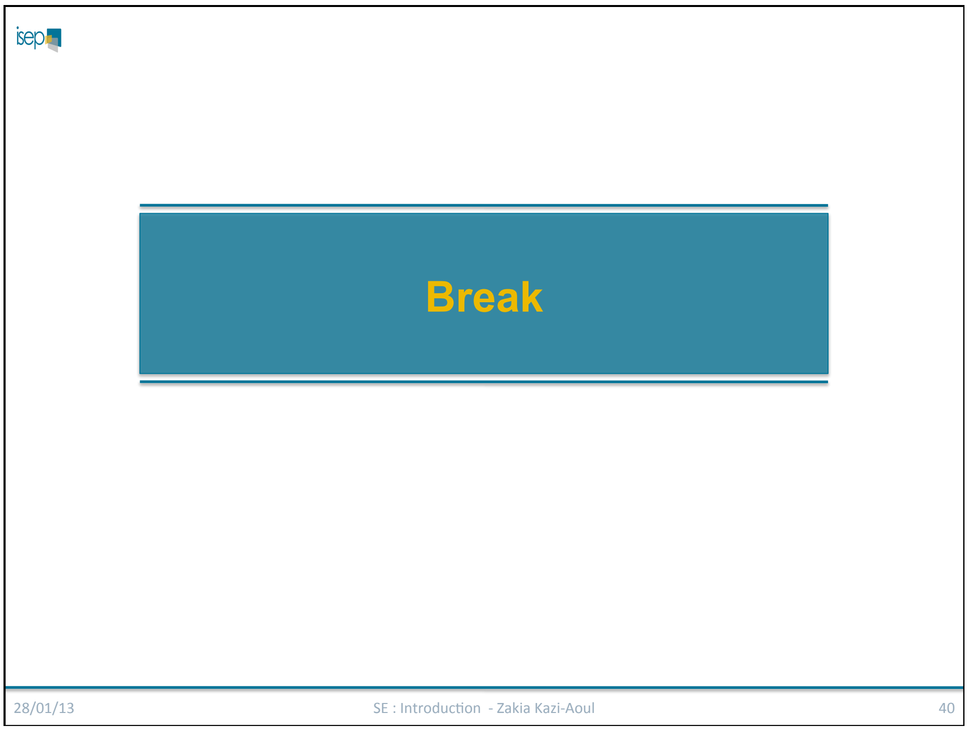
Boehm's quality model (2)

Partial checklist for the quality "self-descriptiveness"

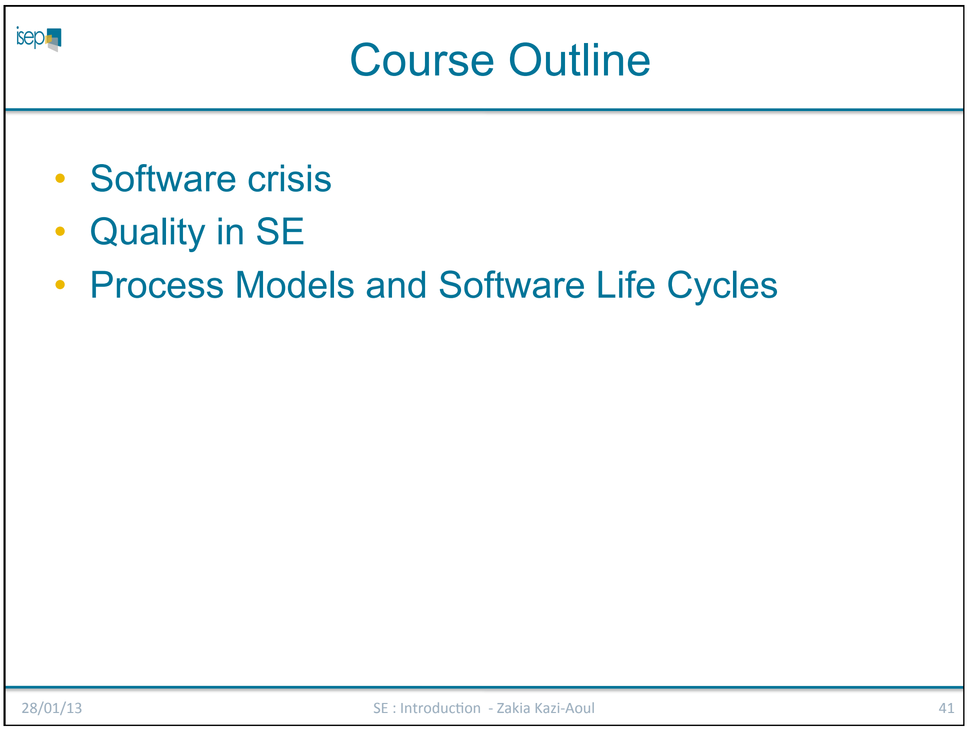
1. Does each program module contain a header block of commentary which describes program name, purpose, modification history, and assumptions?
2. Are the functions of the modules as well as inputs/ outputs adequately defined to allow module testing?
3. Where there is module dependence, is it clearly specified by commentary, program documentation, or inherent program structure.
4. Are variable names descriptive of the physical or functional property represented?
5. Do functions contain adequate descriptive information (i.e., comments) so that the purpose of each is clear?

ISO/IEC 9126 quality model



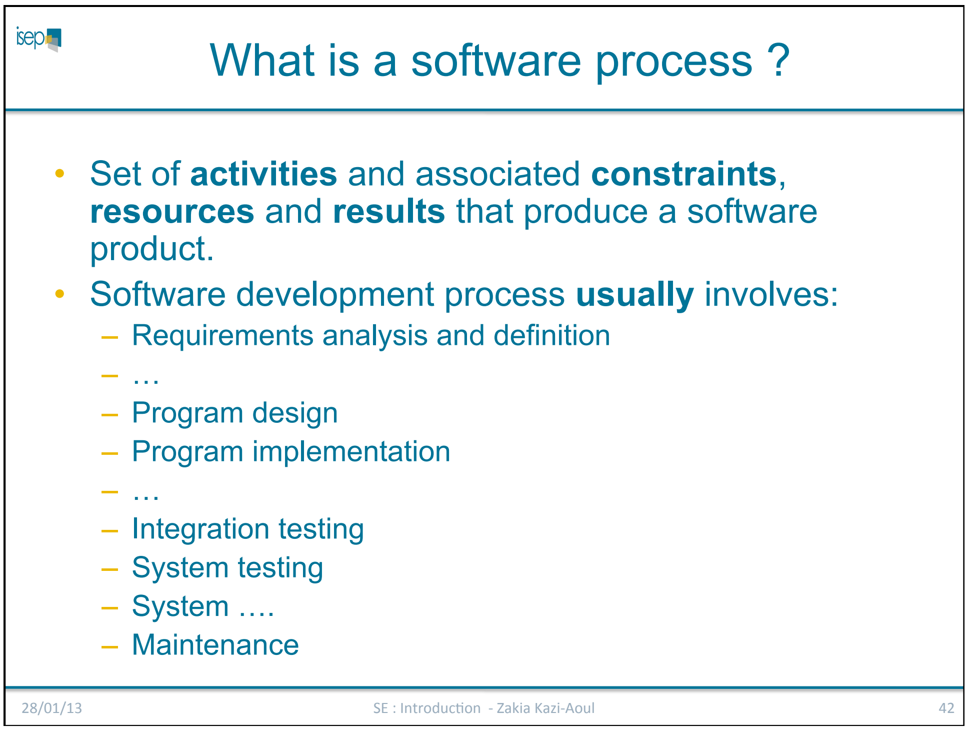


Break



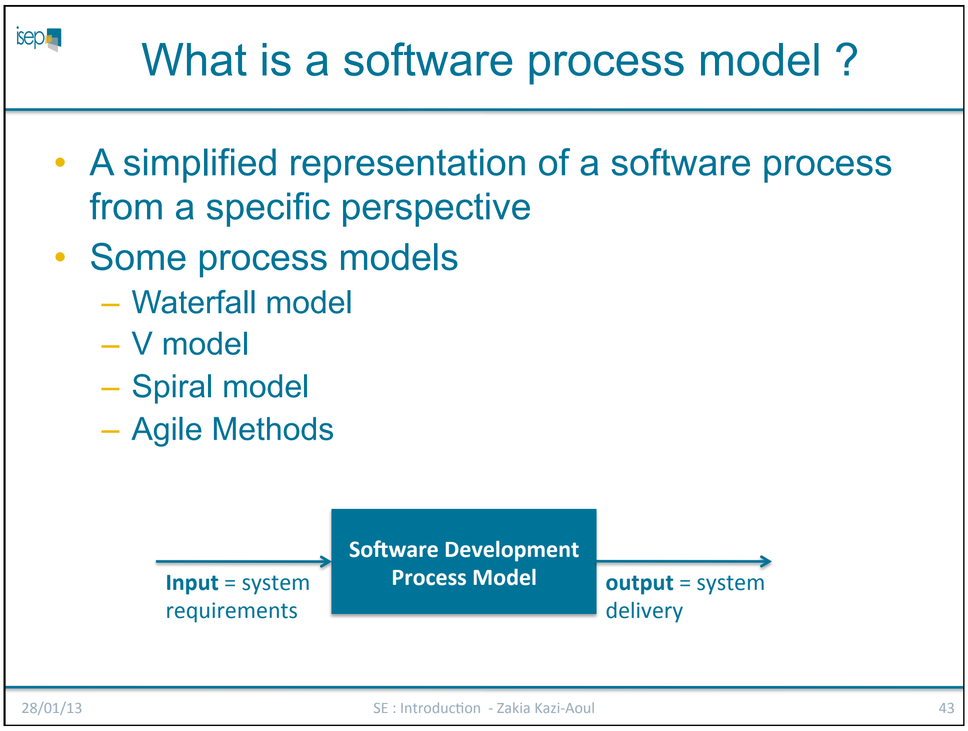
Course Outline

- Software crisis
- Quality in SE
- Process Models and Software Life Cycles



What is a software process ?

- Set of **activities** and associated **constraints**, **resources** and **results** that produce a software product.
- Software development process **usually** involves:
  - Requirements analysis and definition
  - …
  - Program design
  - Program implementation
  - …
  - Integration testing
  - System testing
  - System ….
  - Maintenance



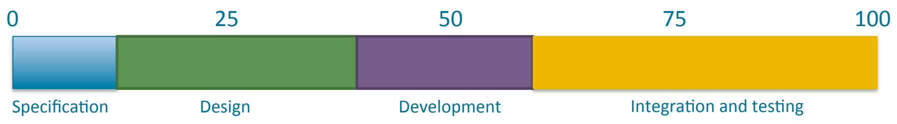
What is a software process model ?

- A simplified representation of a software process from a specific perspective
- Some process models
  - Waterfall model
  - V model
  - Spiral model
  - Agile Methods

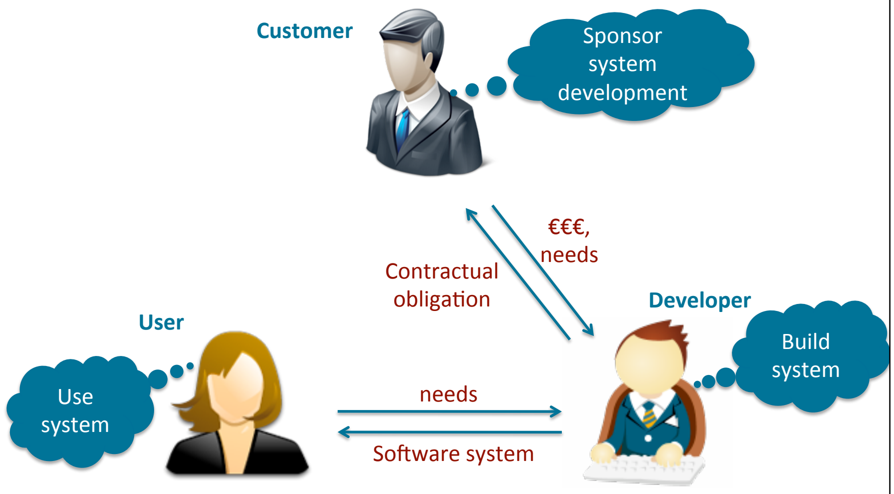
Input = system requirements → **Software Development Process Model** → **output** = system delivery

What are the costs of SE?

- There is no simple answer
  - Costs depend on
    - the process model used and the product
    - the requirements of system attributes such as performance and system reliability
- Example : waterfall model



Who does SE ?

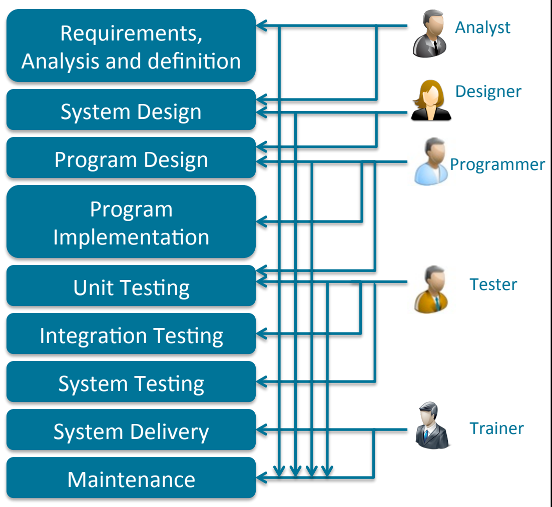


What is CASE ?

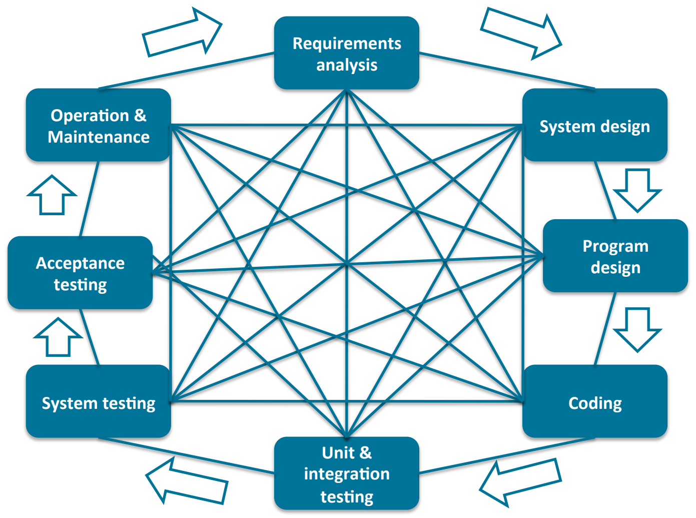
- **C**omputer-**A**ided **S**oftware **E**ngineering
- Software systems which provide automated support for software process **activities**
- Often used for method support (such as UML)
- May also generates code from models
- Upper-CASE tools support requirements gathering and design activities
- Lower-CASE tools support implementation, debugging, and testing activities

Members of a development team

- Developer = software engineer
- Each software engineer may specialize in a particular aspect of development

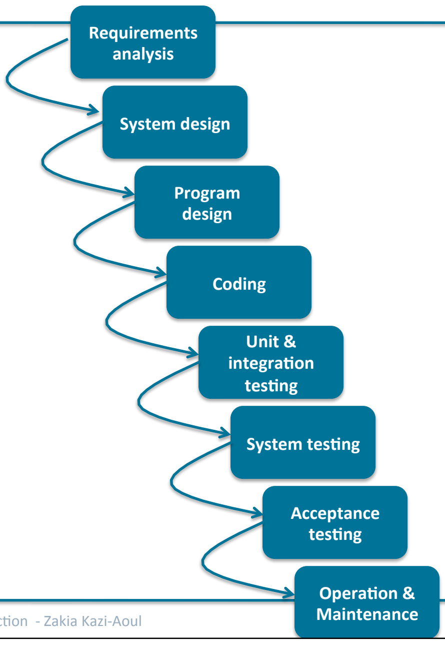


Software Development process could be …



Waterfall Model

- One of the first model proposed (Royce 1970)
- Manufacturing and construction industries
- A stage should be completed before the next begins

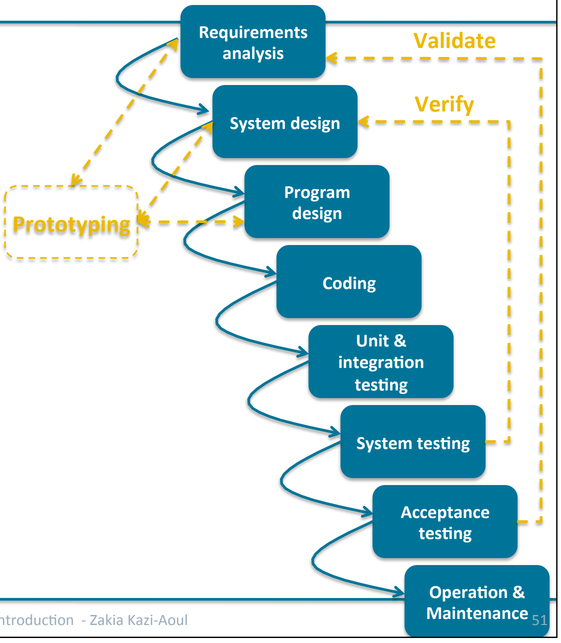


Advantages / Drawbacks of the waterfall model

- **Advantages**
  - Documentation (requirements, design, code, etc.)
- **Drawbacks**
  - Complete and frozen specification document up-front often not feasible in practice
  - Customer involvement in the first phase only
  - Sequential and complete execution of phases often not desirable
  - The product becomes available very late in the process

Waterfall Model with prototyping

- Prototype= partially developed product
  - Customers and developers examine some aspect of the proposed system and decide if it is appropriate for the finished product



Validation / Verification

Validation

Ensures that the system has implemented all of the requirements, so that each system function can be traced back to a particular requirement in the specification

Each function works correctly

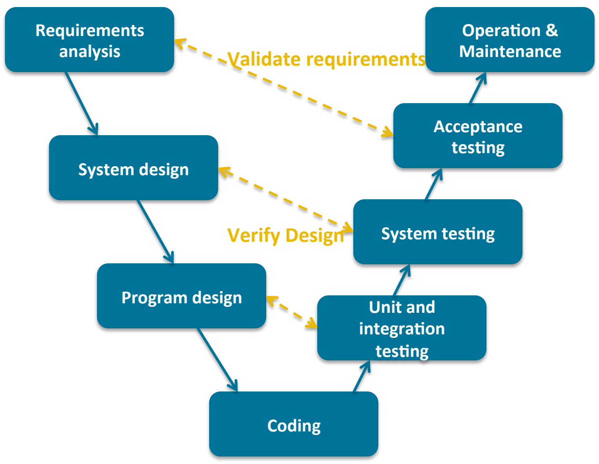
Verification

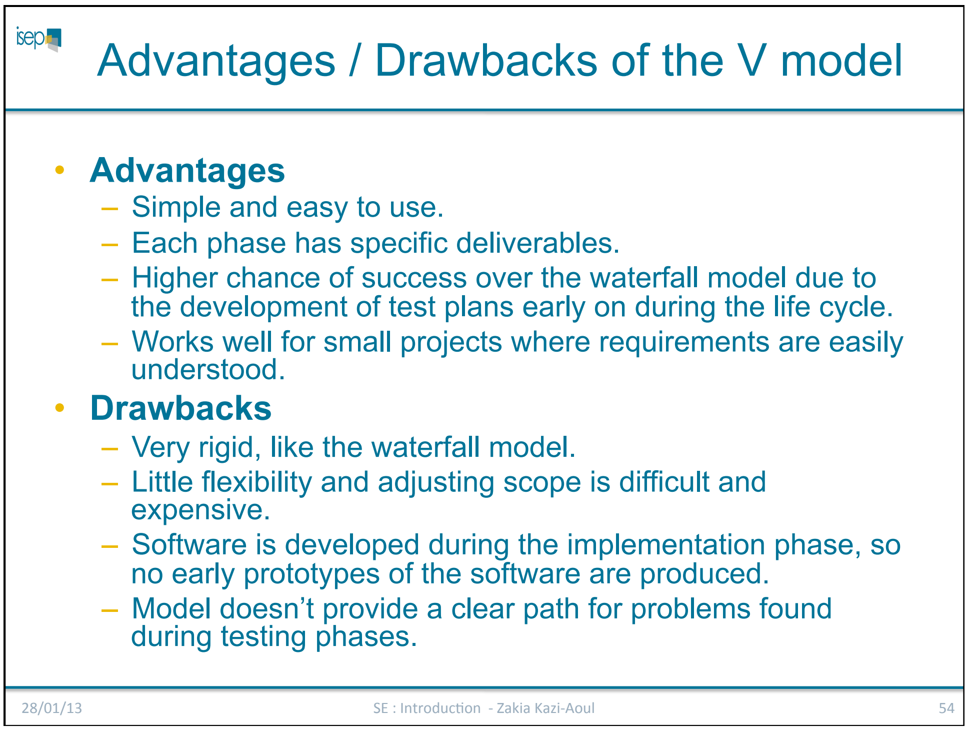
Checks the quality of the implementation

Ensures that developer is building the right product (according to the specification)

V Model

- Variation of the waterfall model
- Focus of Waterfall
  - Documents and artifacts
- Focus of the V model:
  - Activity and correctness

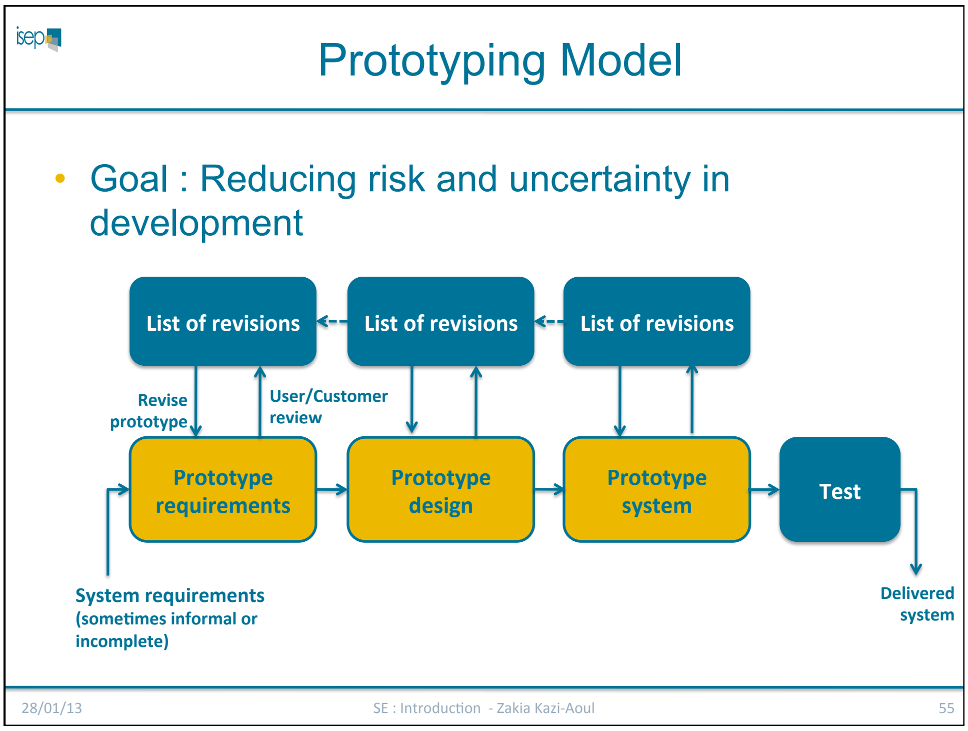




Advantages / Drawbacks of the V model

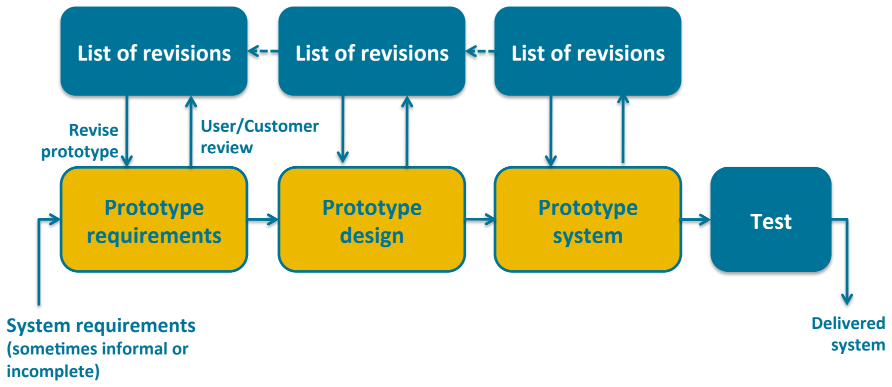
- **Advantages**
  - Simple and easy to use.
  - Each phase has specific deliverables.
  - Higher chance of success over the waterfall model due to the development of test plans early on during the life cycle.
  - Works well for small projects where requirements are easily understood.
- **Drawbacks**
  - Very rigid, like the waterfall model.
  - Little flexibility and adjusting scope is difficult and expensive.
  - Software is developed during the implementation phase, so no early prototypes of the software are produced.
  - Model doesn't provide a clear path for problems found during testing phases.

28/01/13 SE : Introduction - Zakia Kazi-Aoul 54



Prototyping Model

- Goal : Reducing risk and uncertainty in development

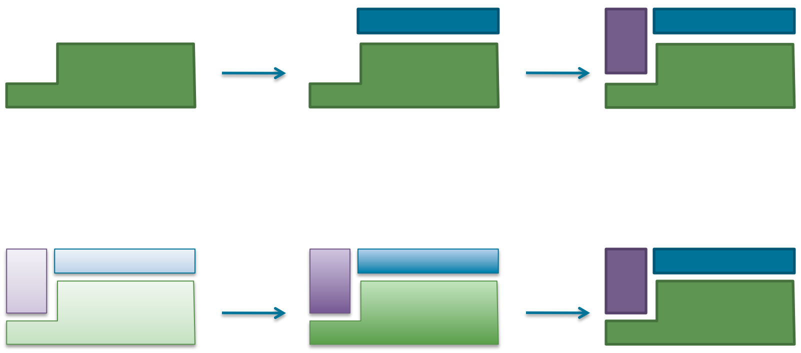


28/01/13 SE : Introduction - Zakia Kazi-Aoul 55

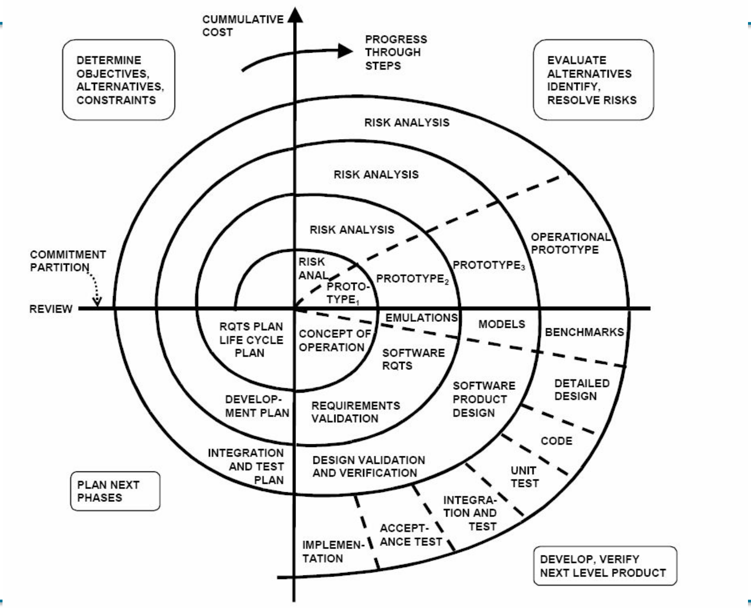
Phased development

- Goal : reduce the cycle time
- System is delivered so that it can be delivered in pieces
  - Users have some functionality while the rest is being developed
  - 2 systems in parallel :
    - Production system
    - Development system
- Two popular approaches:
  - Incremental development
  - Iterative development

Incremental vs Iterative



Spiral Model (boehm 1988)



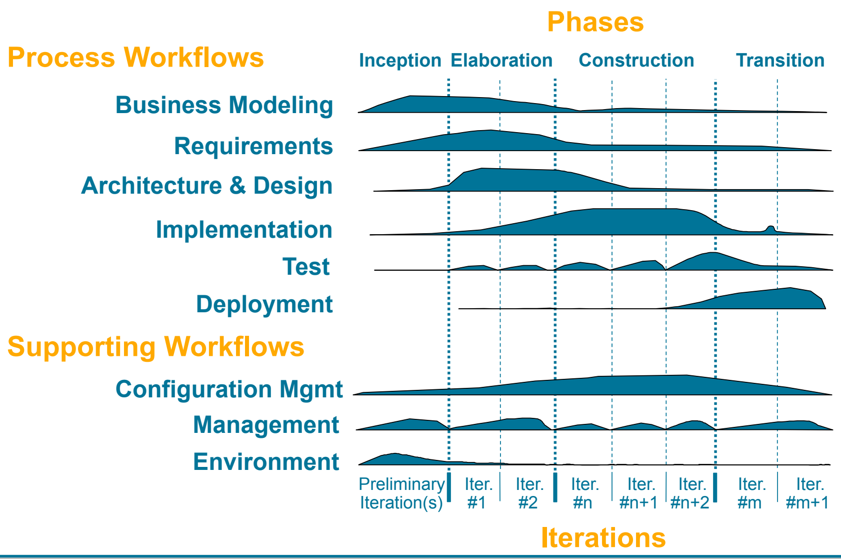
Advantages / Drawbacks of the spiral model

- **Advantages**
  - High amount of risk analysis
  - Good for large and mission-critical projects.
  - Software is produced early in the software life cycle.
- **Drawbacks**
  - Can be a costly model to use.
  - Risk analysis requires highly specific expertise.
  - Project's success is highly dependent on the risk analysis phase.
  - Doesn't work well for smaller projects.

RUP (Rational Unified Process)

- IBM
- A comprehensive process framework that provides industry-tested practices for software and systems delivery and implementation and for effective project management.
- Offers **best practices** guidance suited to a particular development or project need

Overview of RUP



RUP Phases: The time dimension

- Inception
  - Establish the business case for the system.
- Elaboration
  - Develop an understanding of the problem domain and the system architecture.
- Construction
  - System design, programming and testing.
- Transition
  - Deploy the system in its operating environment.

RUP good practice

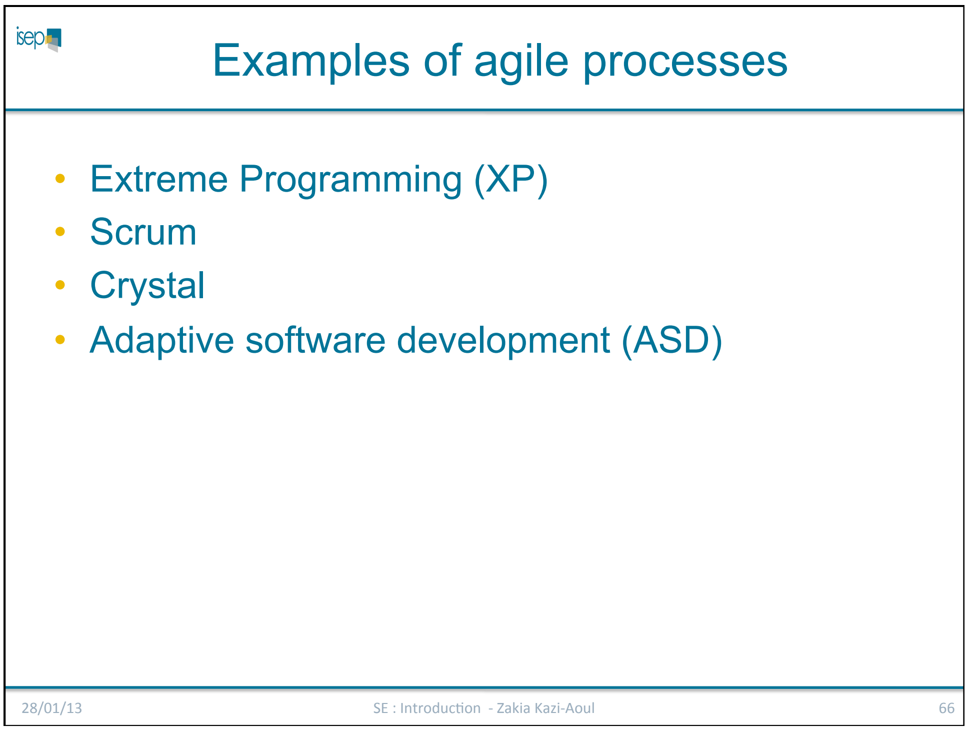
- Develop software iteratively
- Manage requirements
- Use component-based architectures
- Visually model software
- Verify software quality
- Control changes to software

Rational Unified Process – Main Characteristics

- Iterative and incremental
- Use-case-driven
- Architecture-centric
- Uses UML as its modeling notation
- Process framework
  - Comprehensive set of document templates, process workflow templates, and process guidelines
  - Distributed by IBM/Rational on a CD

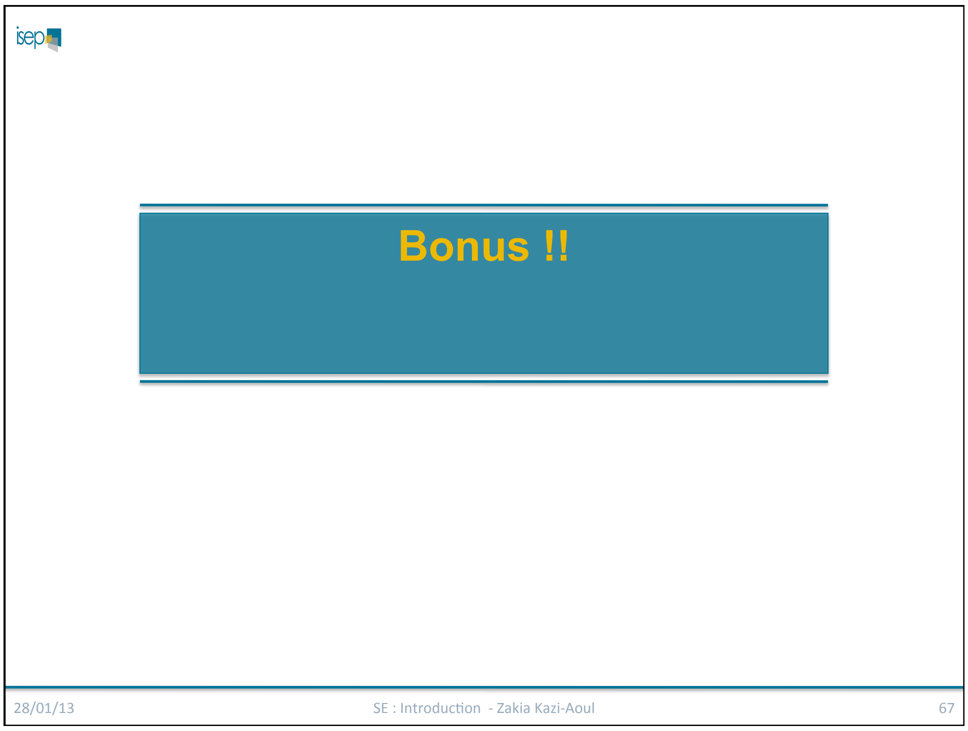
Agile Manifesto

- Four tenets of an alternative way of thinking about software development (Agile Alliance 2001) :
  - Value **individuals and interactions** over processes and tools
  - Invest time in **producing working software** rather than in producing comprehensive documentation
  - Focus on **customer collaboration** rather than contract negotiation
  - Concentrate on **responding to change** rather than on creating a plan
- http://agilemanifesto.org/



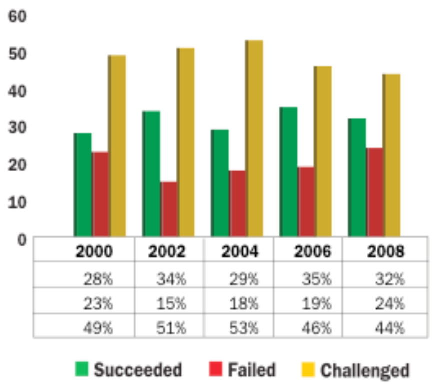
Examples of agile processes

- Extreme Programming (XP)
- Scrum
- Crystal
- Adaptive software development (ASD)



Bonus !!

The Standish Group Chaos Summary 2009



	2000	2002	2004	2006	2008
🟩 Succeeded	28%	34%	29%	35%	32%
🟥 Failed	23%	15%	18%	19%	24%
🟨 Challenged	49%	51%	53%	46%	44%

Some major causes of project failure

- Managerial issues – account for 65% of failure
  - Inappropriate project structure and channels of communication
  - Inappropriate resources (poor skill and knowledge mix)
  - Inappropriate estimates
  - Inappropriate planning methods (poor scheduling and tracking)
  - Inappropriate user buy-in
  - Inappropriate risk management
- Technical issues – account for 35 %of failure
  - Inappropriate software requirements
  - Inappropriate technical design
  - Inappropriate development and testing tools
  - Inappropriate technical documentation
  - Inappropriate technical support
  - Inappropriate technical reviews (and quality audits)

Risk: Definition

- *"A risk is a **problem** that could cause some loss or threaten the success of our project, but which **hasn't happened yet**. These potential problems might have an adverse impact on the cost, schedule, or technical success of the project, the quality of our software products, or project team morale."*
- *"Risk management is the process of **identifying**, **addressing**, and **eliminating** these potential problems **before** they can damage our project."*

(Wiegers 1998)

Common risks

- Common problems typically reported in a software project risk survey include:
  - poor definition and frequent changes to requirements
  - unreliable estimates, unrealistic schedules and inadequate project tracking
  - high staff turnover and shortage of skilled staff
  - lack of project standards and processes
  - lack of design and inadequate documentation
  - inadequate testing and quality procedures

Risk techniques

- **Qualitative** methods:
  - Brainstorming
  - SWOT analysis
  - Maps
  - Checklists and questionnaires
  - Peer interviews
- **Quantitative** methods:
  - Symbolic models
  - Probability analysis
  - Consequence analysis
  - Decision trees
  - Monte Carlo analysis
  - Borda voting method
  - Investment decision making
  - Cost benefit analysis
  - Quantitative market research.

Further reading

- The NATO conferences reports: http://homepages.cs.ncl.ac.uk/brian.randell/NATO/
- Real Software Engineering by Glenn Vanderburg http://confreaks.com/videos/282-lsrc2010-real-software-engineering
  - http://vanderburg.org/Writing/xpannealed.pdf
  - Project for the end of the year
- For fun: Epigrams on programming by Alan Perlis http://www-pu.informatik.uni-tuebingen.de/users/klaeren/epigrams.html