

T-76.3601 — Introduction to Software Engineering

Course Overview

<http://www.soberit.hut.fi/T-76.3601/>

Casper Lassenius
Casper.Lassenius@tkk.fi



Software Engineering?

1. The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, that is, the application of engineering to software
2. The study of approaches in (1).

IEEE Computer Society



So Why Should I Care?

- Society is dependent on software...
- ...and so is in many cases human life...
- ...and most businesses

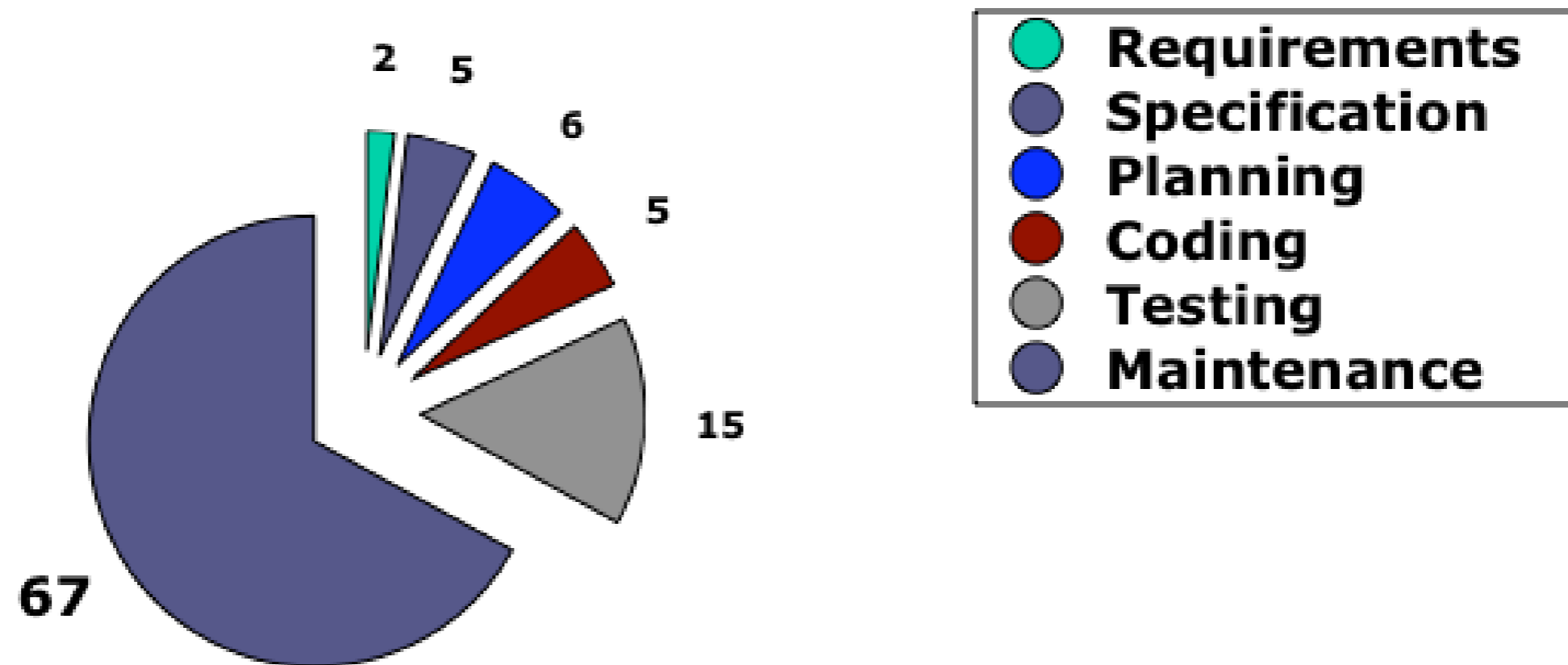
- Would it not be nice if it all just worked?

- Software engineering is about making software systems of various scales, to meet various needs and quality criteria, within real-world constraints

- Much more than coding is involved in this!



Software Life-Cycle Costs



Course Objectives

- Introduce students to software engineering
 - Problems
 - Concepts, models and approaches
 - Industrial practice



After the Course...

...you should...

- understand and appreciate the **challenges** of software engineering
- be familiar with basic **processes, methods** and **models**, and understand their **applicability** to various situations
- be able to read and understand software engineering **literature**
- understand and appreciate the **gap** between software engineering **theory** and **practice**



Course Overview

Part 1: Introduction	18.1. – 15.2.	<i>Goal:</i> Learn basic concepts and models in SE. <i>Method:</i> Lectures, reading the textbook
Mid-terms	1.3. & 8.3.	Textbook, lectures Two takes: Students can take either or both
Part 2: Practice	14.3. – 25.4.	<i>Goal:</i> Learn how SE is practiced in industry <i>Method:</i> Guest lectures, weekly exercises, class discussions
Mid-terms / Final Exam	8.5. & 15.5.	Textbook, lectures, exercises Two mid-term takes: Students can take either or both OR the final exam.
Feedback deadline	23.5.	Deadline for giving course feedback for students wanting their grade before the summer. If feedback not given by this date, you have to write an essay!



Introductory Part

- Lectures covering the basics of software engineering
- Students get and read the textbook
- Ends with a mid-term exam

- The purpose is to get students quickly up to speed on the basics of software engineering
- It prepares for the advanced part of the course



Advanced Part

- Guest lecturers talk on their topics of expertise
 - Concepts and methods
 - Industrial application
- Students are encouraged to
 - Take notes and actively participate in the discussions
 - Re-read related textbook chapters before the lectures
- Weekly exercises
 - Multiple-choice questions
 - One 500-word essay
 - Questions based upon lecture content **and discussions. Please attend!**



Course Material

- Textbook
 - Pressman: Software Engineering, 6th edition. McGraw-Hill, 2005
 - Available from yliopistokirjakauppa, Akateeminen, Dataclub
 - You use older editions on your own risk
- Lecture slides
 - Available on the course Web-site



Course Material

- Pressman on-line resources: <http://www.mhhe.com/engcs/pressman>
 - Slides
 - Quizzes
- Slides with Pressman's copyright (see below) are used directly/altered based upon Pressman's original slides. This is done with explicit permission by Dr. Pressman (available upon request)

These courseware materials are to be used in conjunction with Software Engineering: A Practitioner's Approach, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005.



Course Language

- All course material is in English
- The lecture language is English
 - Software engineering is international
 - Foreign students
 - You will need to use English in the future
- Exercise essays can be written in English, Finnish or Swedish
- Exam questions will be given in (and can be answered using) English, Finnish, and Swedish



Course Personnel (1/2)

- Lecturer
 - Dr. Casper Lassenius
 - Casper.Lassenius@tkk.fi
 - Office hours: by appointment only (available after the lectures)
 - Overall responsibility for the course



Course Personnel (2/2)

- Assistants
 - Ville Heikkilä, N.N.
 - t763601@soberit.hut.fi
 - Exercise and exam grading
 - Communication with students



Passing the Course

- **Register** by 25.1. (next Friday)
 - Use WWWTopi
- Do sufficiently well in the **weekly exercises** (50% of points required)
- **No extra opportunities** for students not getting this minimum
- Pass the mid-term **exams** OR the final exam
 - 50% of points required
 - Essays, yes/no, multiple choice questions
- Give course **feedback** using the form on the Web



Course Grading

- Students can earn up to 100 points
 - Mid-term exams: 30 points each
 - Weekly exercises: 40 points
 - Final exam: 60 points
- The following **minimum scores** must be met to pass:
 - Mid-term exams: 15 points
 - Weekly exercises: 20 points
 - Final exam: 30 points
- Students not receiving the minimum score on the first mid-term exam will have a possibility to take an augmented final exam
- Students not receiving high enough scores on the weekly exercises will have to **retake** the course next year. There are no additional opportunities or penalty exercises.

Grading scale	
85,00–100,00	5
75,00–84,99	4
65,00–74,99	3
55,00–64,99	2
51,00–54,99	1
0–50,99	0



Unjustly Graded?

- If you feel that your **exercises** have been unjustly graded
 - Send an email to t763601@soberit.hut.fi and describe briefly why your answer should have received a different grade compared to the published grading principles
 - or, why the grading principles should be altered
 - attend the exercise complaint session (TBA)
- Complaining about the mid-term and final **exam** results
 - There will be one complaint session for each mid-term exam round (not for every take!)
 - The second complaint session is a joint session with the augmented final exam



Communication Channels

- The lectures
- The course web page <http://www.soberit.hut.fi/T-76.3601/>
- The e-mail list
 - Compiled based upon addresses in WebTopi
- The newsgroup **opinnot.tik.ohjelmistotuotanto**
 - Intended for public questions & answers regarding this course
 - The FAQ on the web pages is updated based upon the discussion
 - Read the Web pages before posting
 - If you absolutely don't want to ask your question in public, send an email to t763601@soberit.hut.fi



Tips for Getting a Good Grade

- Get the book
- Come to the lectures
 - bring the book with you
 - take notes
 - ask questions
- Read the book
- Do the weekly exercises
- Take the mid-term exams (use both takes!)
- Experience shows that getting a good grade does demand work!



Can't Attend the Lectures?

- All lectures are streamed over the internet in real-time using WebEx (soberit.webex.com)
 - Links are available on the course schedule page
 - Interaction (chat, audio, Q&A) is possible
- In addition, the lectures are recorded and can be viewed later



WebEx Demo



Do Not Cheat!

- You must answer all exam questions without consulting any material (not even dictionaries without explicit permission!)
- You may use material when doing the weekly exercises and writing the essay. However, you must provide accurate references, and **may not directly copy text**, e.g., from the course slides or the internet. The essays must be written in **your own words!**
- If you get caught cheating, you will:
 - a) be expelled from the course
 - b) be involved in the department's formal process for dealing with cheating cases. This might lead to you being expelled from TKK.



Questions?



T-76.3601 — Introduction to Software Engineering

Software Engineering in 35 minutes

<http://www.soberit.hut.fi/T-76.3601/>

Casper Lassenius
Casper.Lassenius@tkk.fi



Software Engineering: What?

- A discipline defined as (IEEE Computer Society)
 1. The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, that is, the application of engineering to software
 2. The study of approaches in (1).
- Software engineering deals with the development, operation and maintenance of software in organizations, and has approaches for the individual, team and organizational level spanning all activities involved in software development, such as requirements, design, coding and testing



Why Study Software Engineering?

- Importance of software, as discussed earlier
- Many software development efforts fail on content, quality, schedule, and/or budget
- Most software is developed in teams
- Software engineering is what you need to know in addition to coding
 - coding is only a small part of the picture
- on the other hand...
 - ...to be a successful manager of software development, you must understand the technical aspects as well!



What is Software?

- Software is a set of items or objects that form a “configuration” that includes
 - programs
 - documents
 - data
- “Computer programs, procedures and possibly associated documentation and data needed to operate a computer system” (IEEE 1991)

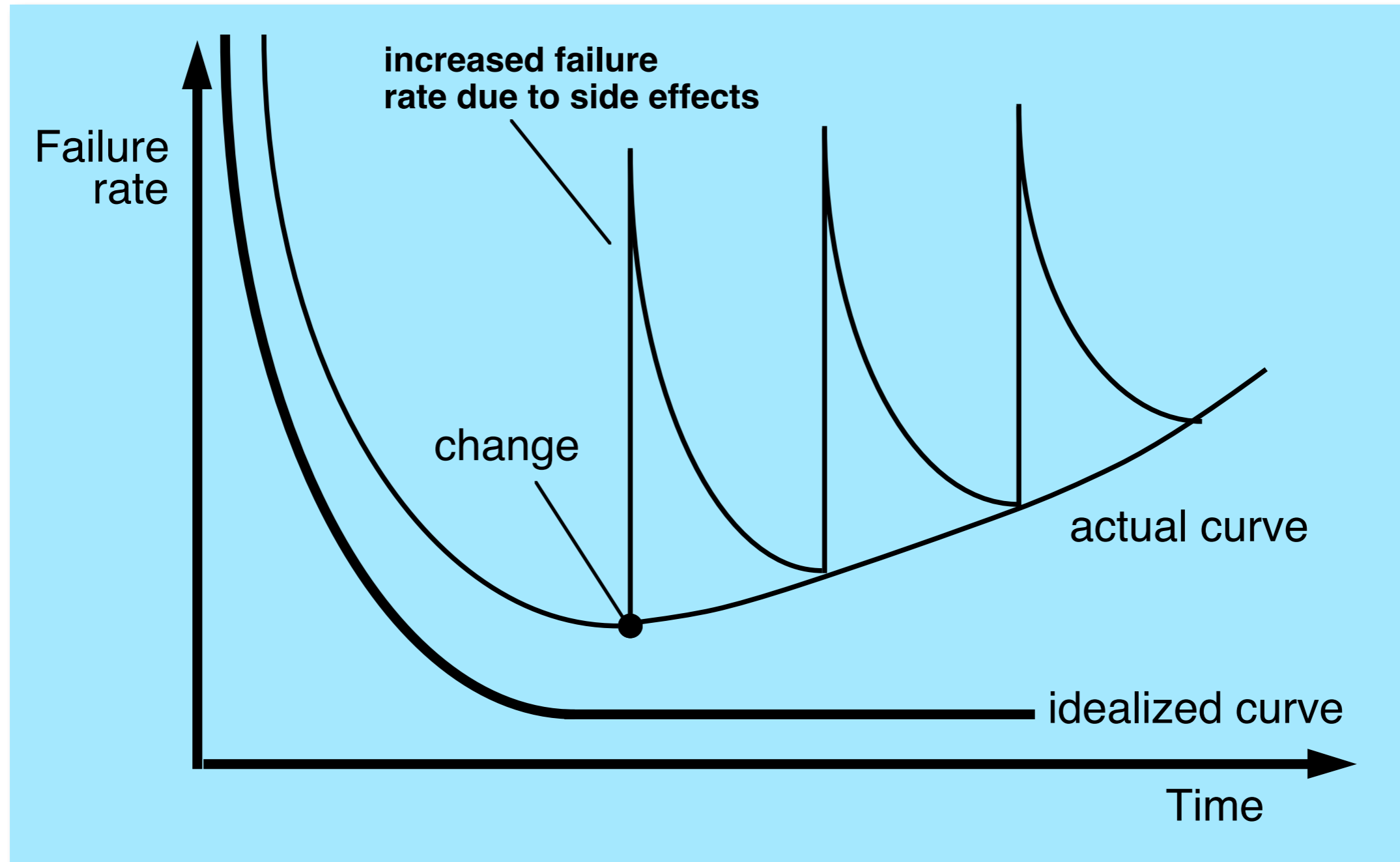


Properties of Software

- Software is engineered, not manufactured
- Software does not wear out
- Software is
 - complex
 - subject to change
 - conformant to environmental change
 - invisible and unvisualizable



Wear vs. Deterioration



These courseware materials are to be used in conjunction with Software Engineering: A Practitioner's Approach, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005.

Software Applications

- system software
- application software
- engineering / scientific software
- embedded software
- product-line software
- Web applications
- AI software
- open-source software



The Software Engineering Sandbox

CMM

CMM

SPICE

RUP

UML

eXtreme Programming

Z

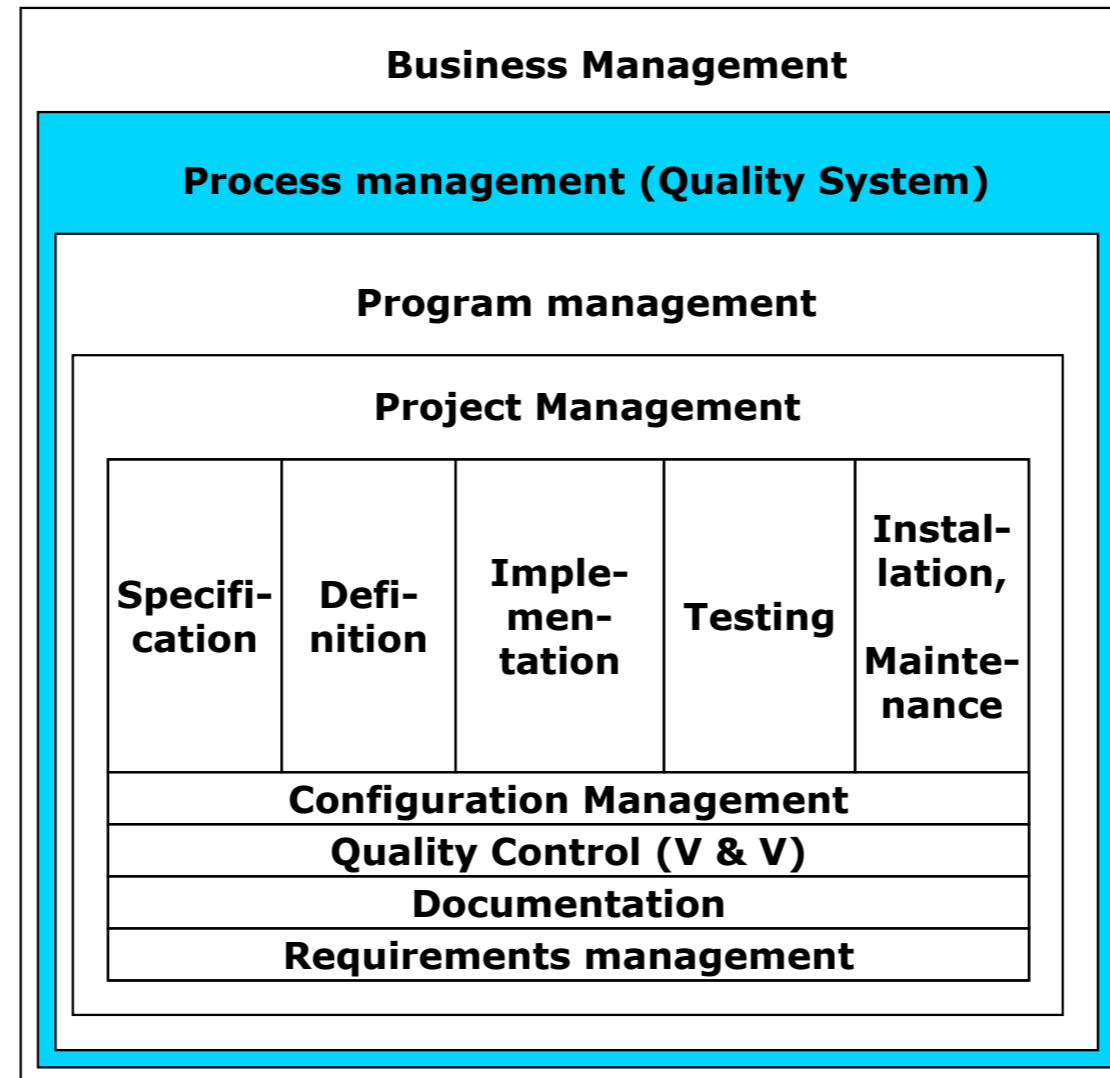
Time Boxing

Object Orientation

SA/SD

USDP

CleanRoom



What: SWEBOK — The Software Engineering Body of Knowledge

- Knowledge areas
 - Software requirements
 - Software design
 - Software construction
 - Software testing
 - Software maintenance
 - Software configuration management
 - Software engineering process
 - Software engineering tools and methods
 - Software quality
- Knowledge areas of related disciplines
 - Computer engineering
 - Computer science
 - Management
 - Mathematics
 - Project management
 - Quality management
 - Software ergonomics
 - Systems engineering



Levels of Software Engineering Study

- Individual
 - Personal Software Process
 - Design & Specification methodologies
- Team
 - Team Software Process
 - Project management
- Organization
 - Process assessment and improvement
 - Process management
 - “Quality system”



Legacy Software

- Why must it change?
 - software must be **adapted** to meet the needs of new computing environments or technology
 - software must be **enhanced** to implement new business requirements
 - software must be **extended** to make it interoperable with other more modern systems or databases
 - software must be **re-architected** to make it viable withing a network environment



The Mythical Man-month

- Effort corresponds to cost but not progress
- Most software engineering tasks are not partitionable
 - 1 woman can have 1 baby in 9 months
 - 2 women can have 1 baby in ? months
 - In software engineering: **communication need**
- The software engineering paradox
 - Adding more people to a project that is running late further delays the schedule

(Brooks, 1974)



“There is No Silver Bullet”

— but many have been proposed...

