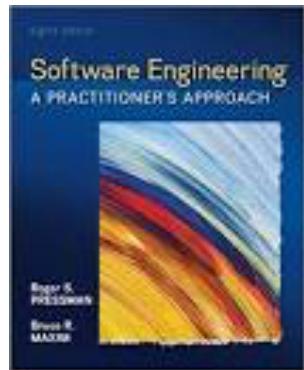


Introduction to Software Engineering

UA.DETI.IES - 2019/20

Resources & Credits



- ❖ Roger S. Pressman, Bruce Maxim,
Software Engineering: A Practitioner's
Approach, 7th Edition,
McGraw-Hill Education, 2015

- ❖ Ian Sommerville,
Software Engineering, 10th Edition,
Pearson, 2016

- ❖ Grady Booch, 'The History of Software
Engineering.' IEEE Software 35.5, 2018



Importance of Engineering Processes



Common Engineers' roles

- ❖ Research and development
 - To enhance current processes and stimulate productivity.
- ❖ Monitoring & troubleshooting
 - To ensure that systems are functioning correctly.
- ❖ Documentation
 - To ensure that all new personnel can understand the best practices and standards of each individual process.
- ❖ Risk assessments
 - To determine risk of malfunction, failure, personal, etc.
- ❖ Management roles
 - May have responsibilities over operations and personnel.
- ❖ Budgeting responsibilities

What is Software Engineering?

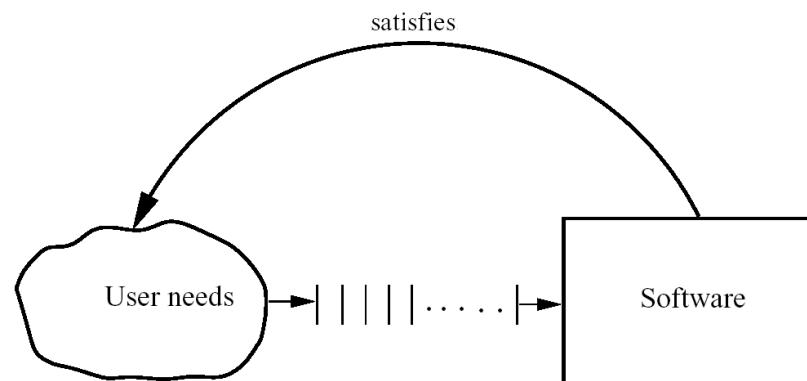
❖ IEEE definition:

- The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software;
- that is, **the application of engineering to software.**

❖ Roger S. Pressman's definition:

- Software engineering is the technology that encompasses a **process**, a set of **methods** and an array of **tools**.

(*Software Engineering: A Practitioner's Approach*)



What is Software Engineering?

Reverse engineering
Software process models Agile software development
Reliability modeling and analysis Formal specifications
Software economics and metrics Agent oriented software engineering
Aspect oriented software engineering

Software Engineering

Software engineering methodologies UML MDA and AADL
Software development tools Component based software engineering
Service-oriented computing
Object-oriented technology Knowledge-based software engineering
Software maintenance Autonomic and self-managed software
Software assurance Domain specific software engineering
Validation and verification Software architecture and design
Software testing Software security engineering
Software architecture Requirements elicitation
Software evolution

<https://wonderfulengineering.com/what-is-software-engineering/>

What is Software Engineering?

1 Re Requirements Elicitation	2 Ri Risk Analysis
3 Ra Requirements Analysis	4 Dc Component Design
11 Ar Atomic Requirements	12 Dbd Database Design
19 Rt Requirements Attributes	20 Dp Design Patterns
37 Rr Requirements Reviews	38 Ap Architecture Pattern
55 Tm Traceability Management	56 Lsd Large-scale System Design
87 Rem Management of Requirements Portfolio	88-103 Dn Design Notations
57-71 Agp Agile Methods	72 Pc Program Comprehension
89 Prs Presentation Skills	90 Ts Training Skills
58 Pp Pair Programming	59 Td Test Driven Development
91 Em Empathy	60 Dd Definition of Done
92 Crr Creation of Relationships	61 Cd Continuous Integration
93 Cm Conflict Management	62 Cy Continuous Delivery
94 Ns Negotiation Skills	63 Us User Stories
95 Rh Rhetoric	64 Bam Backlog Management
96 Is Intercultural Skills	65 Sm Stand-up Meeting
97 Crt Creativity Techniques	66 Sp Spike Solutions
98 Ma Marketing Basics	67 Pg Planning Game
99 Lea Leadership Basics	68 No No Overtime
100 Gom Good Manners	69 Co Collective Code Ownership
101 Im Intrinsic Motivation	70 Tl Travel Light
102 Phf Physical Fitness	71 Sr System Metaphor
103 St Stop Talking	
5 Bcs Basic Coding Skills	6 Sa Static Code Analysis
13 Cr Code Refactoring	7 Ut Unit Testing
14 Da Dynamic Code Analysis	8 Rca Defect Root Cause Analysis
15 It Integration Testing	9 At Code Analysis Tools
16 Uid User Interface Design	10 Exm Expectation Management
17 Ct Continuous Integration Tools	
18 Tam Task Management	
31 Cp Code Peer Reviews	32 Vm Volume Metrics
33 Se Service Testing	34 Ua User Acceptance
35 Rt Requirements Management Tools	36 Prb Project Management Basics
52 Ui Usability Labs	53 Ide Integrated Development Environments
54 Est Estimations	
49 Cco Code Comments	50 Cm Complexity Metrics
51 Pt Performance Testing	52 Pt Profiling Tools
80 Pac Parallel Computing	81 Cf Code Format Standards
82 Cc Code Coverage	83 St Stress Testing
84 Tt Test Automation Tools	85 Mo Measurement of Activities
111 Ai Artificial Intelligence	112 Sdp Software Development Process
113 Cr Code Reuse	114 Dea Dependency Analysis
115 Ex Exploratory Testing	116 Mt Modeling Tools
117 Vc Version Control Systems	118 Pc Project Controlling
57 Agp Agile Planning	58 Pp Pair Programming
59 Td Test Driven Development	60 Dd Definition of Done
61 Cd Continuous Integration	62 Cy Continuous Delivery
63 Us User Stories	64 Bam Backlog Management
65 Sm Stand-up Meeting	66 Sp Spike Solutions
67 Pg Planning Game	68 No No Overtime
69 Co Collective Code Ownership	70 Tl Travel Light
71 Sr System Metaphor	

© 2013 by Markus Sprunck, www.sw-engineering-candies.com, v1.0

<https://www.sw-engineering-candies.com/blog-1/periodic-table-of-software-engineering-know-how>

Importance of Software Engineering

❖ Reduces complexity

- SE divides big problems into several small problems, which are solved independently to each other.

❖ Minimizes risks, software cost, and delivery time

- Highly paid professionals are needed to develop software with millions lines of code. Good planning and good practice reduce efforts, costs and delivery time.

❖ Allow managing large projects

- To invest months/years in a product requires lots of planning, direction, testing and maintenance.

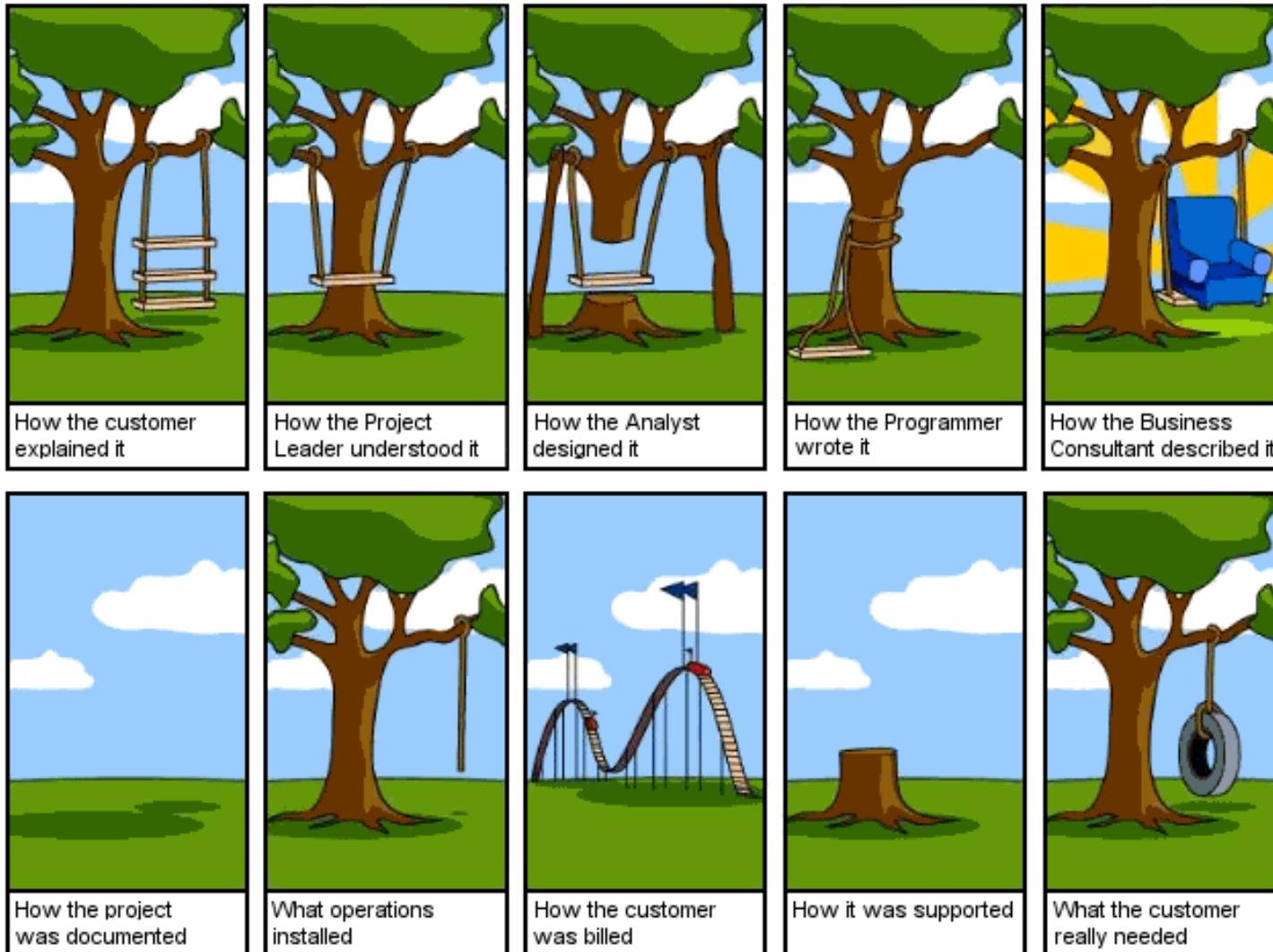
❖ Enforces development of reliable software

- Testing and maintenance are key aspects addressed in SE

❖ Standardize processes

- Software standards are critical to produce more effective software.

Importance of Software Engineering



<http://projectcartoon.com>

Importance of Software Engineering

Academic

- ❖ Developer is the user
 - Works most of the time
 - Bugs are tolerable
 - UI not important
 - No documentation
- ❖ SW not in critical use
- ❖ Reliability not important
- ❖ No investment
- ❖ Don't care about portability
- ❖ Most of the cases: "present and forget"

Industrial

- ❖ Others are the users
 - Works robustly
 - Bugs not tolerated
 - UI very important issue
 - Documents needed
- ❖ Supports business
- ❖ Reliability is key
- ❖ Heavy investment
- ❖ Portability is a key issue
- ❖ Evolve the product through years

Software is Expensive?

- ❖ Rough cost estimate...
 - Productivity = 500 LOC/PM (Lines Of Code / Person Month)
 - Cost to the company = 5K€/PM
 - So each line of delivered code costs about 10€

- ❖ A simple application for a business may have 10KLOC to 50KLOC
 - Cost = 100K€ to 500K€
 - Can easily run on less than 10K€ hardware
 - So HW costs <<< SW costs

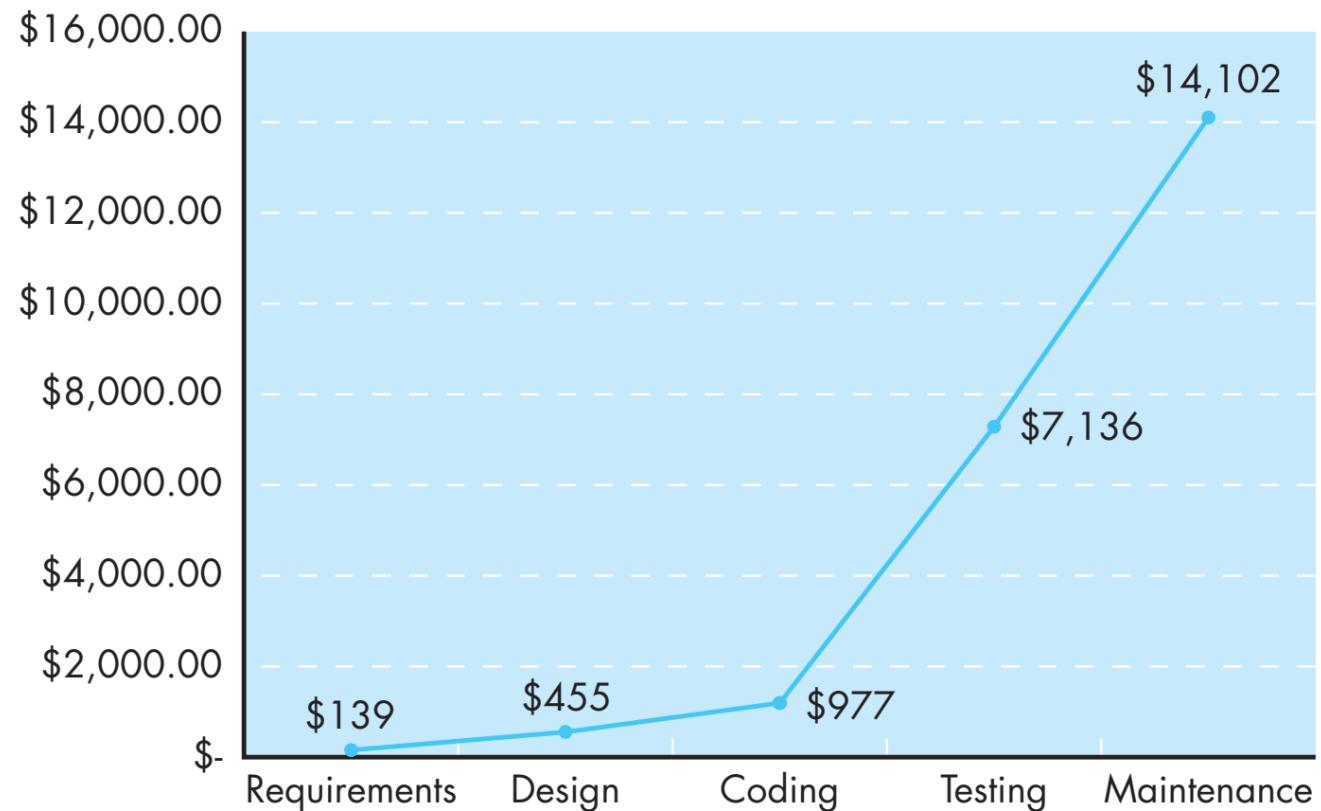
Software is Expensive?

- ❖ SW failures are different from failures of mechanical or electrical systems
 - In software, failures are not due to aging related problems. They occur due to errors introduced during development
- ❖ Once SW delivered, it enters maintenance phase
 - Residual errors requiring corrective maintenance
 - Upgrades and environment changes – adaptive maintenance
- ❖ Over SW lifetime, maintenance can cost more than the development cost of SW

Cost of correction along the lifecycle

FIGURE 19.2

Relative cost of correcting errors and defects
Source: Adapted from [Boe01b].



Software Engineering Methodology

- ❖ SE focuses mostly on processes for achieving goals
 - Process must be systematic
 - Design of proper processes and their control is a key challenge SE faces
- ❖ SE separates process for developing sw from the developed product (i.e the sw)
 - SW process is the equivalent of manufacturing process
- ❖ The development process is typically phased
 - Phases separate concerns with each phase focusing on some aspect
 - Requirements, architecture, design, coding, testing are key phases
 - This phased process has to be properly managed to achieve the objectives
 - Metrics and measurement important for this

Software Architect role

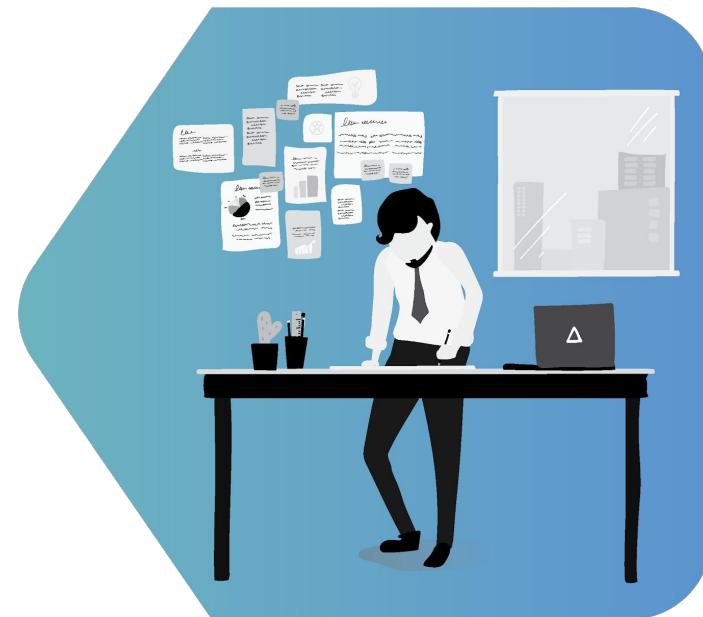
- ❖ Makes high-level design choices and dictates technical standards, including software coding standards, tools, and platforms.



Valentina Sosa, *The Architect Manifest: Principles of design in agile environments*, O'Reilly Software Architecture

Software Architect skills

- ❖ Leader
- ❖ Problem solver and analyst
- ❖ Communicator
- ❖ Technologist
- ❖ Researcher
- ❖ Software Designer
- ❖ Promoter of Good Practices



Software Engineering phases



Matching the organization culture



Working together with the team

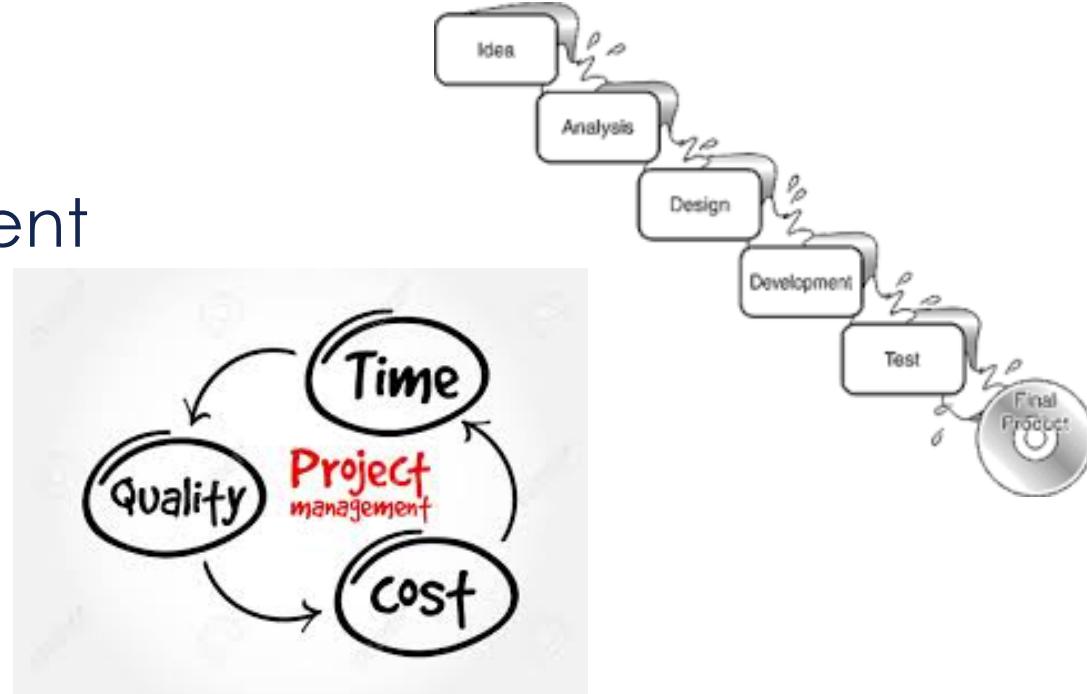
- ❖ Discuss your ideas with the team
- ❖ Be open minded towards new ideas
- ❖ Propose your ideas in a storytelling fashion
- ❖ Your objective is to have a successful project
- ❖ Be prepared to give up your ideas



Other key issues

- ❖ Methods

- ❖ Time management



- ❖ Efficiency

- ❖ Quality

- ❖ Ethics



Summary

- ❖ The problem domain for software engineering is industrial strength software
- ❖ Software comprises programs, documentation, and data
- ❖ Software engineering aims to provide methods for systematically developing high quality products with the high productivity and low costs

