

Software Engineering

– Introduction & History

L.EIC-ES-2021-22

Ademar Aguiar, J. Pascoal Faria
Rui Neves, André Lago, André Restivo, Bruno Lima, Daniel
Pinho, Filipe Correia, Hugo Ferreira, José Campos

Agenda

- Challenges of large scale software development
- Goals and scope of software engineering
- Brief history of software engineering
- Overview of some popular software processes and process models
- Quiz
- Course structure and organization

Challenges of large scale software development





A portrait photograph of Marc Andreessen, a bald man with a serious expression, wearing a dark suit jacket and a light blue shirt. He is positioned on the left side of the frame, which is divided by a vertical white border from the quote area on the right.

In short, software is eating the world

— *Marc Andreessen* —

AZ QUOTES

SOFTWARE IS EATING THE WORLD

1980

1. IBM
2. AT&T
3. Exxon
4. Standard Oil of Indiana
5. Schlumberger

1990

1. IBM
2. Exxon
3. General Electric
4. Philip Morris
5. Royal Dutch Petrol

2000

1. General Electric
2. Exxon Mobil
3. Pfizer
4. Citigroup
5. Cisco Systems

2010

1. Exxon Mobil
2. Apple
3. Microsoft
4. Berkshire Hathaway
5. General Electric

2017

1. Apple
2. Alphabet
3. Microsoft
4. Facebook
5. Amazon

Mercedes Class S
close to 100 million lines of code, executing in 70 to 100
electronic control unit based on microprocessors, in a
vehicular network



Airbus A380

120 million lines of code (avionics and onboard support systems)



<https://www.computerweekly.com/news/2240233948/Airbus-Interview-research-innovation-and-the-future-of-aviation>



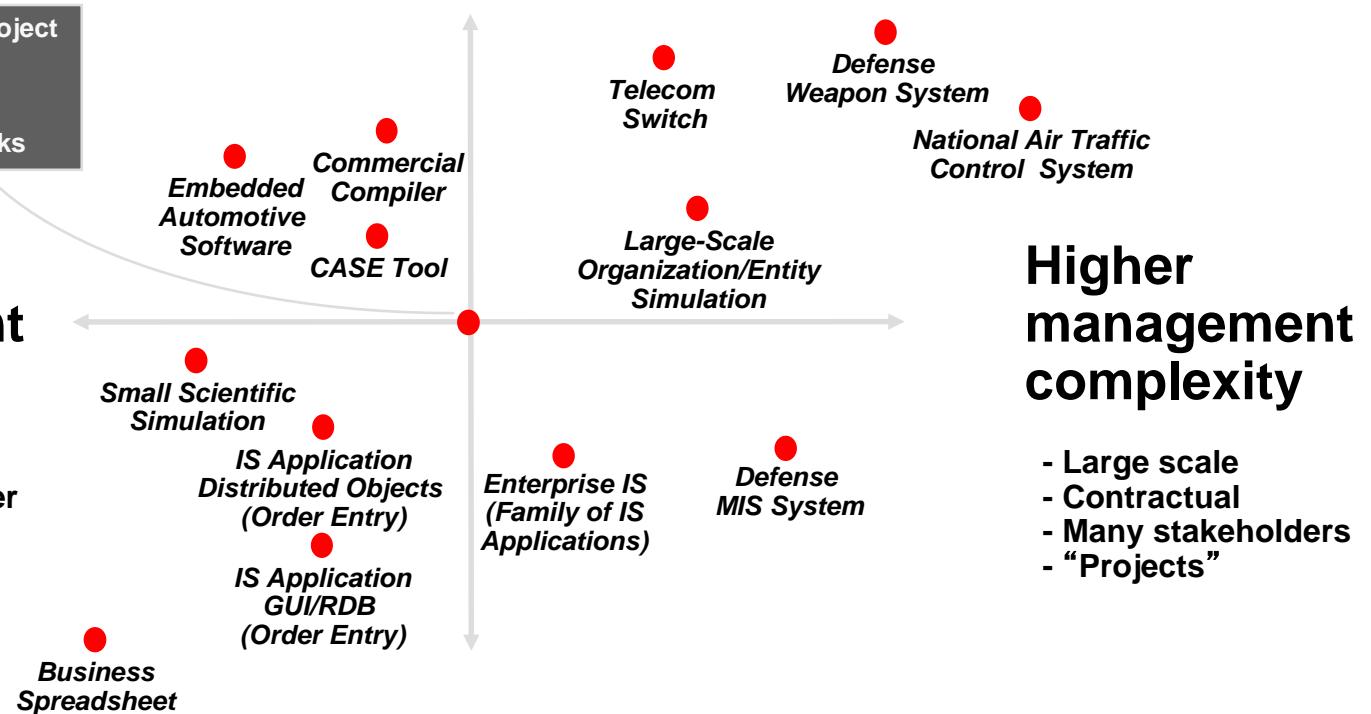
Higher technical complexity

- Embedded, real-time, distributed, fault-tolerant
- Custom, unprecedented, architecture reengineering
- High performance

An average software project
- 5-10 people
- 3-9 month duration
- 3-5 external interfaces
- Some unknowns & risks

Lower management complexity

- Small scale
- Informal
- Single stakeholder
- “Products”



Lower technical complexity

- Mostly 4GL, or component-based
- Application reengineering
- Interactive performance

Walker Royce

Historical episodes of software issues

- [1983 Soviet nuclear false alarm incident - G8](#)
- [Air Control Failure - G7](#)
- [Airbus A380 - G5](#)
- [Amazon UK Christmas 2014 G6](#)
- [Apple's GOTO bug - G8](#)
- [Ariane 5 Explosion \(1997\) - G1](#)
- [Fiat Chrysler Software Recall \(2016\) - G7](#)
- [Ford Motor Company and Oracle G6](#)
- [Hartford Coliseum Collapse \(1978\) - G5](#)
- [Heartbleed \(2014\) - G8](#)
- [Heathrow terminal 5 opening 2008 - G7](#)
- [HSBC Online Banking Glitch - G6](#)
- [Inmates Released Early - G1](#)
- [Intel Pentium's FDIV bug - G1](#)
- [IPO Facebook G6](#)
- [IPv4 address exhaustion - G2](#)
- [Japanese Hitomi Spacecraft - G4](#)
- [Jeep Cherokee Steering Attacks - G1](#)
- [Knight's 440\\$ Million Glitch - G4](#)
- [Mariner 1 Launch Failure \(1961\) - G7](#)
- [Medical Machine Kills \(1985\) - G2](#)
- [Millennium Bug - G1](#)
- [MIM Patriot failure in Saudi Arabia - G7](#)
- [Mt Gox bitcoin hack - G8](#)
- [Multidata Systems \(2000\) - G4](#)
- [NASA's Mars Climate Orbiter - G3](#)
- [Nissan's Airbag Software Malfunction \(2013\) - G2](#)
- [North American Blackout \(August 14, 2003\) - G3](#)
- [Northeast Blackout of 2003 - G5](#)
- [Pentium Fails Long Division \(1993\) - Grupo 2](#)
- [Phobos 1 Drone Malfunction - G3](#)
- [Swedish Social Security Agency - G2](#)
- [Tesla driver killed in crash with Autopilot active - G5](#)
- [The 2003 Northeast Blackout - G2](#)
- [The Collapse of the Hartford Civic Center - G3](#)
- [The Meltdown Attack - G5](#)
- [The Morris Worm - G8](#)
- [The Patriot Missile Failure - G4](#)
- [Therac 25\(1985 1987\) - G7](#)
- [Toyota´s electronic throttle control system failure - G1](#)
- [Uber - False Positive - G6](#)
- [US Prisoners Released Early - G1](#)
- [USS Yorktown Incident - G8](#)
- [WannaCry Ransomware Cyber Attack - G6](#)
- [Y2K bug \(2000\) - G2](#)
- [Yahoo! Data Theft - G8](#)



Explore Tricentis: qTest | Tosca | Flood | RPA

Support Community Company CIO Corner Blog English ▾

TRICENTIS

Products Solutions Services Resources

Contact Us

WHITE PAPERS

Software Fail Watch: 5th Edition

White Paper

The Software Fail Watch is an analysis of software failures found in a year's worth of English language news articles. The result is an extraordinary reminder of the role software plays in our daily lives, the necessity of software testing in every industry, and the far-reaching impacts of its failure.

The 5th Edition of the Software Fail Watch identified 606 recorded software failures, impacting half of the world's population (3.7 billion people), \$1.7 trillion in assets, and 314 companies. And this is just scratching the surface—there are far more software defects in the world than we will likely ever know about.

LOSSES FROM SOFTWARE FAILURES (USD)

1,715,430,778,504

ONETRILLIONSEVENHUNDREDFIFTEENBILLIONFOURHUNDREDTHIRTYMILLIONSEVENHUNDREDSIXTYEIGHTHOUSANDFIVEHUNDREDFOUR

Download Now

*Salutation:

*Email Address:

*First Name:

*Last Name:

*Company Name:

NOUN

RSET

KEY
REL

0

1

2

3



APOLLO 11

THE COMPLETE DESCENT



BASED ON
THE APOLLO LUNAR SURFACE JOURNAL

DATE
20 JULY 1969



Ethiopian Airlines

FLIGHT 302



Why do software projects fail so often?

- Unrealistic or unarticulated project **goals**
- Inaccurate **estimates** of needed resources
- Badly defined system **requirements**
- Poor **reporting** of the project's **status**
- Unmanaged **risks**
- Poor **communication** among customers, developers, and users
- Use of **immature technology**
- Inability to handle the project's **complexity**
- Sloppy development practices
- Poor project management
- Stakeholder politics
- Commercial pressures

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

Exercise: security vulnerability

```
bool IsPasswordOK(void) {  
    char Password[12]; // Memory storage for pwd  
    gets(Password); // Get input from keyboard  
    if (!strcmp(Password, "goodpass")) return(true); // Password Good  
    else return(false); // Password Invalid  
}  
  
void main(void) {  
    bool PwStatus; // Password Status  
    puts("Enter Password:"); // Print  
    PwStatus=IsPasswordOK(); // Get & Check Password  
    if (PwStatus == false) {  
        puts("Access denied"); // Print  
        exit(-1); // Terminate Program  
    }  
    else puts("Access granted");// Print  
}
```

```
Enter Password:  
1234567890123456j*!  
Access granted
```

What happened?

What is the cause?



3 min

Goals and scope of software engineering

What is Software Engineering?

- "Research, design, develop, and test operating systems-level software, compilers, and network distribution software for medical, industrial, military, communications, aerospace, business, scientific, and general computing applications" – Bureau of Labor Statistics
- "the systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software"—The Bureau of Labor Statistics — IEEE *Systems and software engineering - Vocabulary*
- "The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software" — IEEE *Standard Glossary of Software Engineering Terminology*

•••

- "an engineering discipline that is concerned with all aspects of software production" — Ian Sommerville
- "the establishment and use of sound engineering principles in order to economically obtain software that is reliable and works efficiently on real machines" — Fritz Bauer
- “a systematic approach to design, construct, and maintain software more ‘citus, vilis, and bonus’ — Ademar Aguiar

Guide to the Software Engineering Body of Knowledge

Version 3.0



Editors

Pierre Bourque, École de technologie supérieure (ÉTS)
Richard E. (Dick) Fairley, Software and Systems Engineering Associates (S2EA)



Software engineering knowledge areas

Table I.1. The 15 SWEBOK KAs

Software Requirements
Software Design
Software Construction
Software Testing
Software Maintenance
Software Configuration Management
Software Engineering Management
Software Engineering Process
Software Engineering Models and Methods
Software Quality
Software Engineering Professional Practice
Software Engineering Economics
Computing Foundations
Mathematical Foundations
Engineering Foundations

Guide to the Software Engineering
Body of Knowledge
Version 3.0 SWEBOK®, IEEE
Computer Society

<https://www.computer.org/web/swebok/v3>

Table I.2. Related Disciplines

Computer Engineering
Computer Science
General Management
Mathematics
Project Management
Quality Management
Systems Engineering

Brief history of software engineering

- Important phases, dates and facts; famous historical software episodes.

The History of Software Engineering

Grady Booch

IBM Fellow & Chief Scientist for Software Engineering

Email: gbooch@us.ibm.com

Twitter: [@grady_booch](https://twitter.com/grady_booch)

Web: computingthehumanexperience.com

V1.0

The First Recorded Software Bug (1945)

- On September 9, 1945, U.S. Navy officer Grace Hopper found a **moth** between the relays on the Harvard Mark II computer she was working on.
- In those days computers filled (large) rooms and the warmth of the internal components attracted moths, flies and other flying creatures. Those creatures then **shortened circuits** and caused the computer to malfunction.

9/9

0800 Anton started
1000 " stopped - Anton ✓ { 1.2700 9.037 847 025
13° UC (034) MP - MC 1.263160000 9.037 846 995 connect
033 PRO 2 2.130476415 4.615925059 (-)
connect 2.130676415
Relays 6-2 in 033 failed special speed test
in relay " 10.00 test .
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Multi Adder Test.
1545 Relay #70 Panel F
(Moth) in relay.
1650 Anton started.
1700 closed down.
First actual case of bug being found.



Grace Hopper

The First (?) Documented Software Development Process (1956)

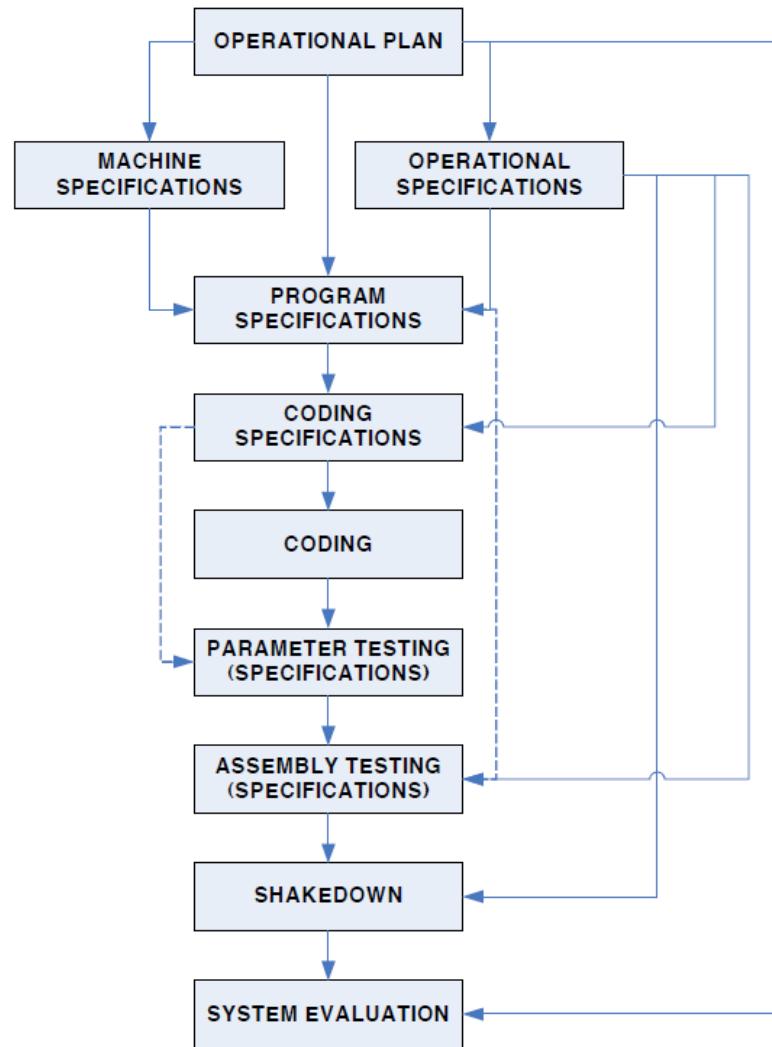


Figure 1. The SAGE Software Development Process (1956)

SAGE: Semi-automatic Ground Environment Air Defense System

Margaret Hamilton: SAGE & Apollo (1960's)

- Margaret Hamilton was Director of the Software Engineering Division of the MIT Instrumentation Laboratory, which developed on-board flight software for the Apollo space program.
- "The Heart and Soul of Apollo: Doing it Right the First Time"
- From 1961 to 1963, she worked on the Semi-Automatic Ground Environment (SAGE) Project at Lincoln Lab,
- On November 22, 2016, she was awarded the Presidential Medal of Freedom by U.S. President Barack Obama



Hamilton in 1969, standing next to listings of the software she and her MIT team produced for the Apollo project

Edgar Dijkstra: Go To Statement Considered Harmful (1968)

For a number of years I have been familiar with the observation that the quality of programmers is a decreasing function of the density of `go to` statements in the programs they produce. More recently I discovered why the use of the `go to` statement has such disastrous effects, and I became convinced that the `go to` statement should be abolished from all "higher level" programming languages (i.e. everything except, perhaps, plain machine code). At that time I did not attach too much importance to this discovery; I now submit my considerations for publication because in very recent discussions in which the subject turned up, I have been urged to do so.

In Communications of the ACM, 1968



Edgar Dijkstra, ACM Turing Award winner, 1972

First Software Engineering Conference (1968)

- Since the late 1950s and early 1960s, programmers debated what engineering might mean for software.
- Two conferences on software engineering sponsored by the NATO Science Committee in 1968 and 1969 as a response to the software crisis marked the official start of the profession of software engineering and gave the field its initial boost
- See also: http://en.wikipedia.org/wiki/History_of_software_engineering



SOFTWARE ENGINEERING

**Report on a conference sponsored by the
NATO SCIENCE COMMITTEE
Garmisch, Germany, 7th to 11th October 1968**

Chairman: Professor Dr. F. L. Bauer

Co-chairmen: Professor L. Bollett, Dr. H. J. Helms

Editors: Peter Naur and Brian Randell

Fred Brooks: The Mythical Man-Month (1975)



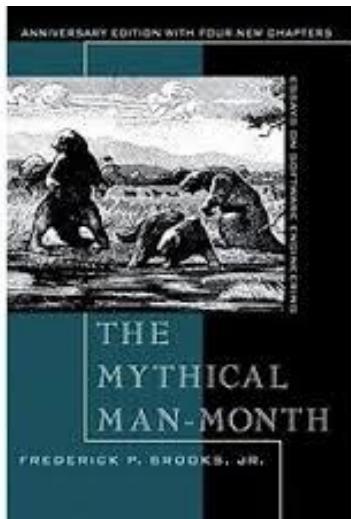
16 The Mythical Man-Month

for there is a probability distribution for the delay that will be encountered, and "no delay" has a finite probability. A large programming effort, however, consists of many tasks, some chained end-to-end. The probability that each will go well becomes vanishingly small.

The' Man-Month

The second fallacious thought mode is expressed in the very unit of effort used in estimating and scheduling: the man-month. Cost does indeed vary as the product of the number of men and the number of months. Progress does not. *Hence the man-month as a unit for measuring the size of a job is a dangerous and deceptive myth.* It implies that men and months are interchangeable.

Men and months are interchangeable commodities only when a task can be partitioned among many workers *with no communication among them* (Fig. 2.1). This is true of reaping wheat or picking cotton; it is not even approximately true of systems programming.



Therac-25 Accidents (1985-1987)

- At least 5 patients died because of massive overdoses of radiation caused by a software error (**race condition**)
- References:
 - An investigation of the Therac-25 Accidents, N.G. Leveson, C.S. Turner, Computer, IEEE, 1993
 - <https://en.wikipedia.org/wiki/Therac-25>



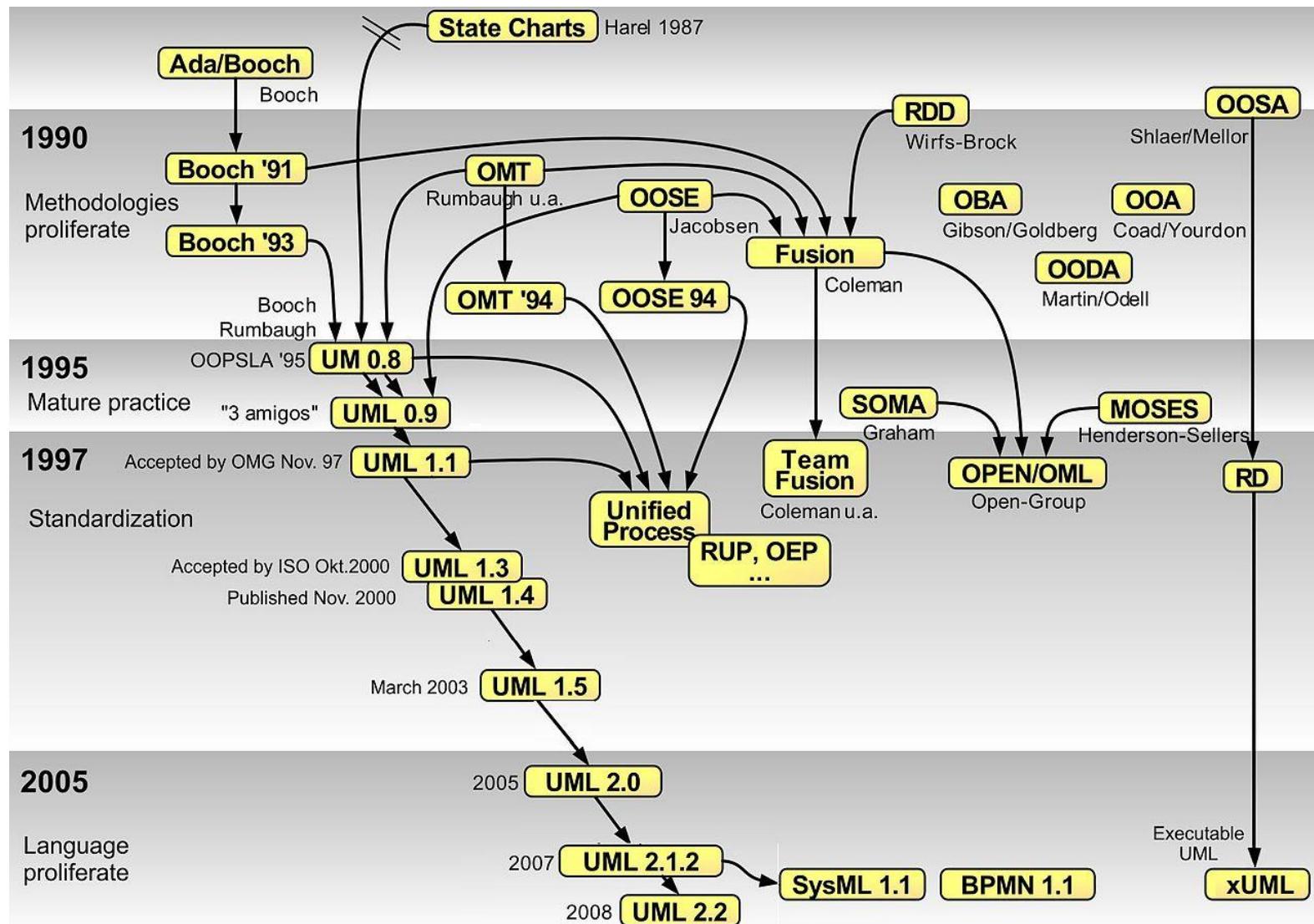
Fred Brooks: No Silver Bullet (1986)

- “No Silver Bullet – Essence and Accident in Software Engineering”, Fred Brooks, 1986
- “there is no single development, in either technology or management technique, which by itself promises even one order of magnitude [tenfold] improvement within a decade in productivity, in reliability, in simplicity.”
- "we cannot expect ever to see two-fold gains every two years" in software development, as there is in hardware development (Moore's law).



Fred Brooks,
ACM Turing Award winner,
1999

The Last Method War (1990-1994)



Ariane 5 explosion (1996)

- 10 years of development and \$7 billion lost in an explosion due to a software error (**overflow**)
- A data conversion from 64-bit floating point value to 16-bit signed integer value to be stored in a variable representing horizontal bias caused a processor trap because the floating point value was too large to be represented by a 16-bit signed integer.
- References:
 - <http://www.irisa.fr/pampa/EPEE/Ariane5.html>
 - <https://www.youtube.com/watch?v=A1gGGDG580E>
 - https://en.wikipedia.org/wiki/Ariane_5



Agile Manifesto (2001)

The screenshot shows a web browser window displaying the Agile Manifesto website at <http://agilemanifesto.org>. The page features a background image of several people in a workshop or office setting. At the top, the title "Manifesto for Agile Software Development" is displayed. Below it, a central text block states: "We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:" followed by the four core values of Agile. A note below explains the relative importance of these values. At the bottom, the names of the original signatories are listed in three columns.

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Kent Beck Mike Beedle Arie van Bennekum Alistair Cockburn Ward Cunningham Martin Fowler	James Grenning Jim Highsmith Andrew Hunt Ron Jeffries Jon Kern Brian Marick	Robert C. Martin Steve Mellor Ken Schwaber Jeff Sutherland Dave Thomas
--	--	--

<http://agilemanifesto.org/>

Overview of some popular software processes and process models

- Waterfall, Iterative, Spiral, RUP, Scrum, XP

Waterfall

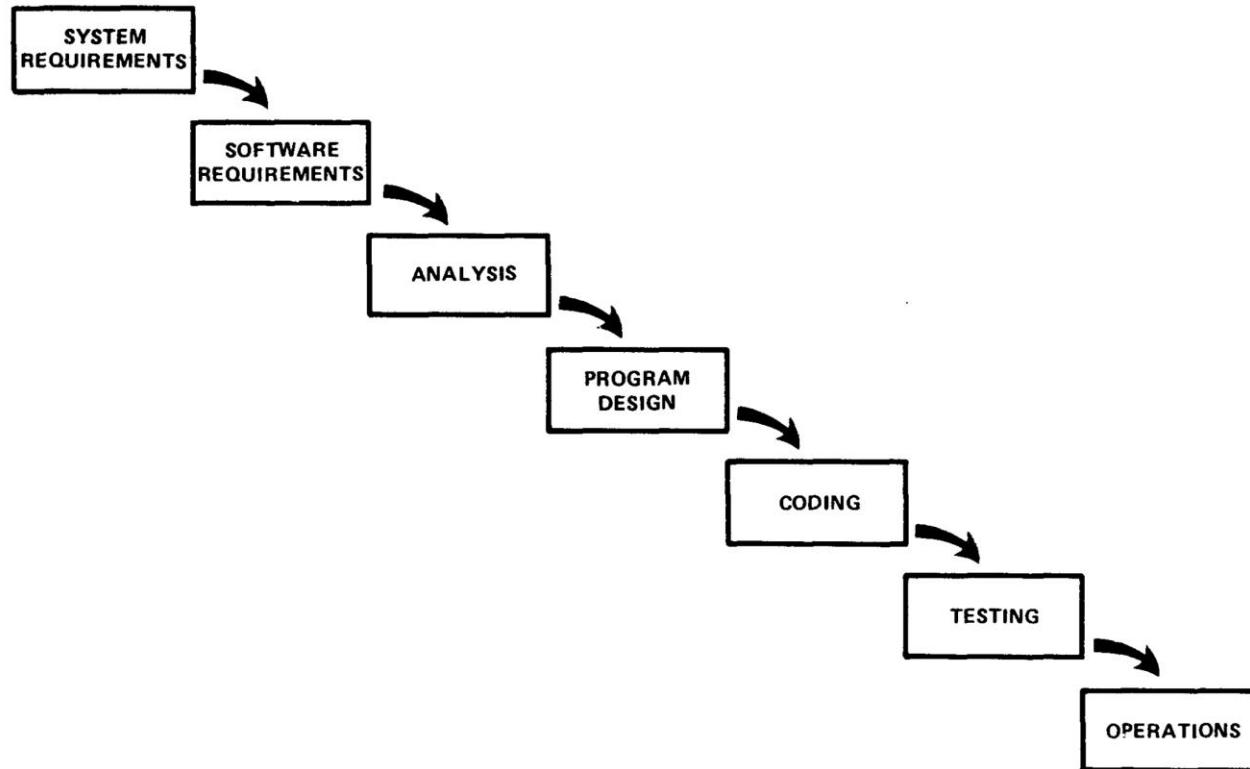


Figure 2. Implementation steps to develop a large computer program for delivery to a customer.

Iterative

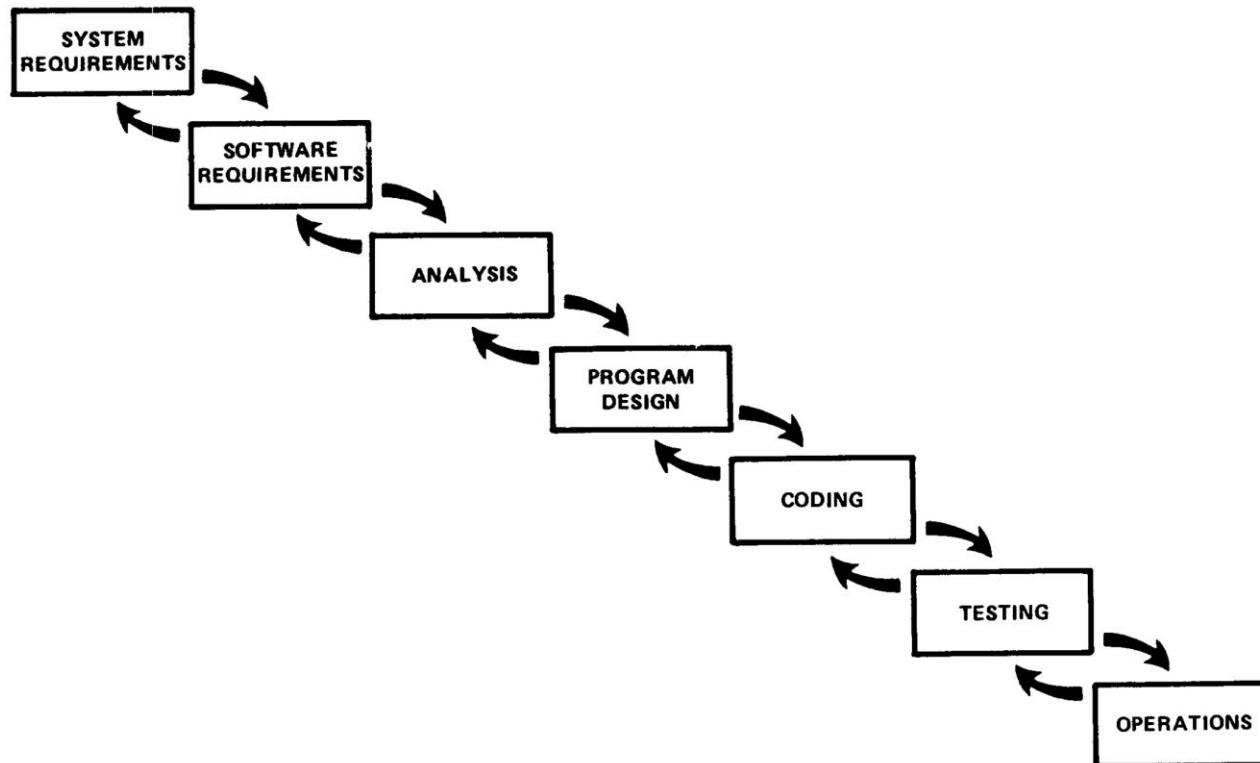


Figure 3. Hopefully, the iterative interaction between the various phases is confined to successive steps.

Spiral

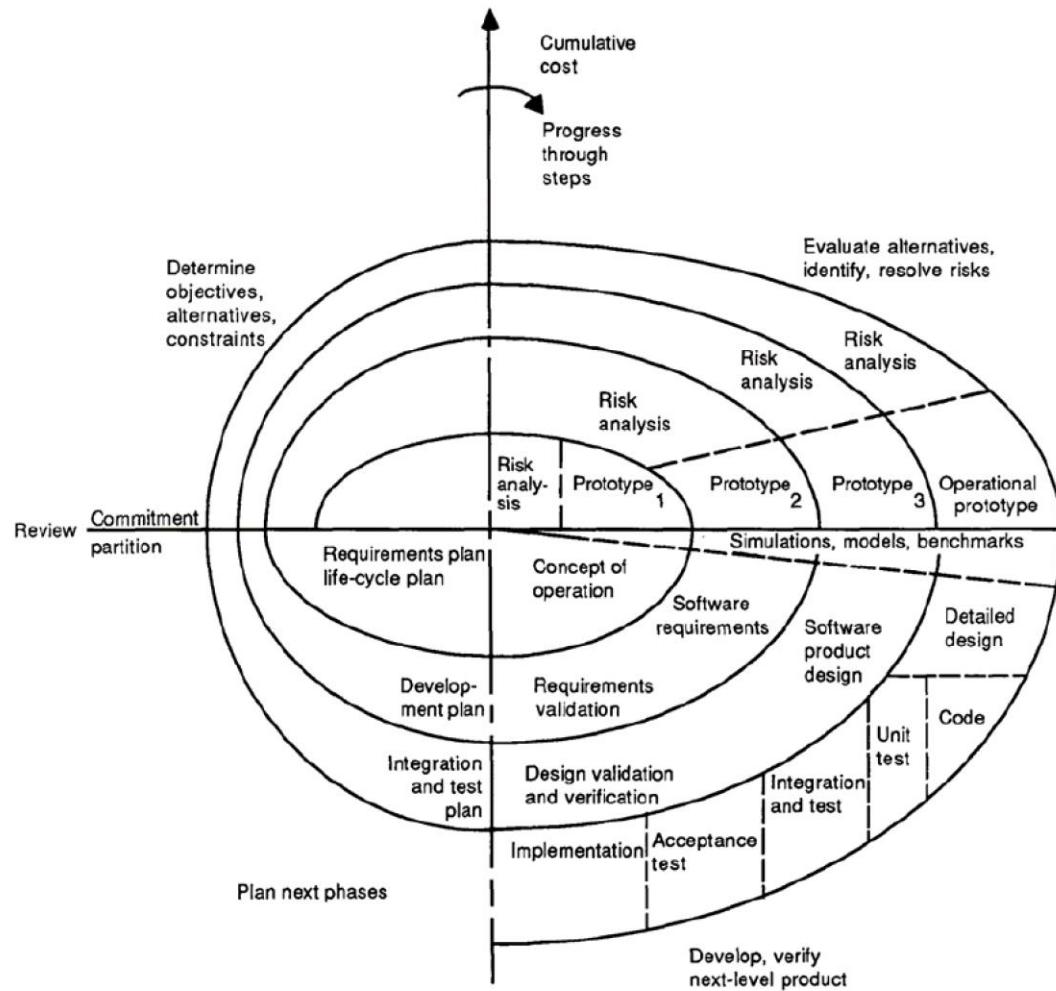
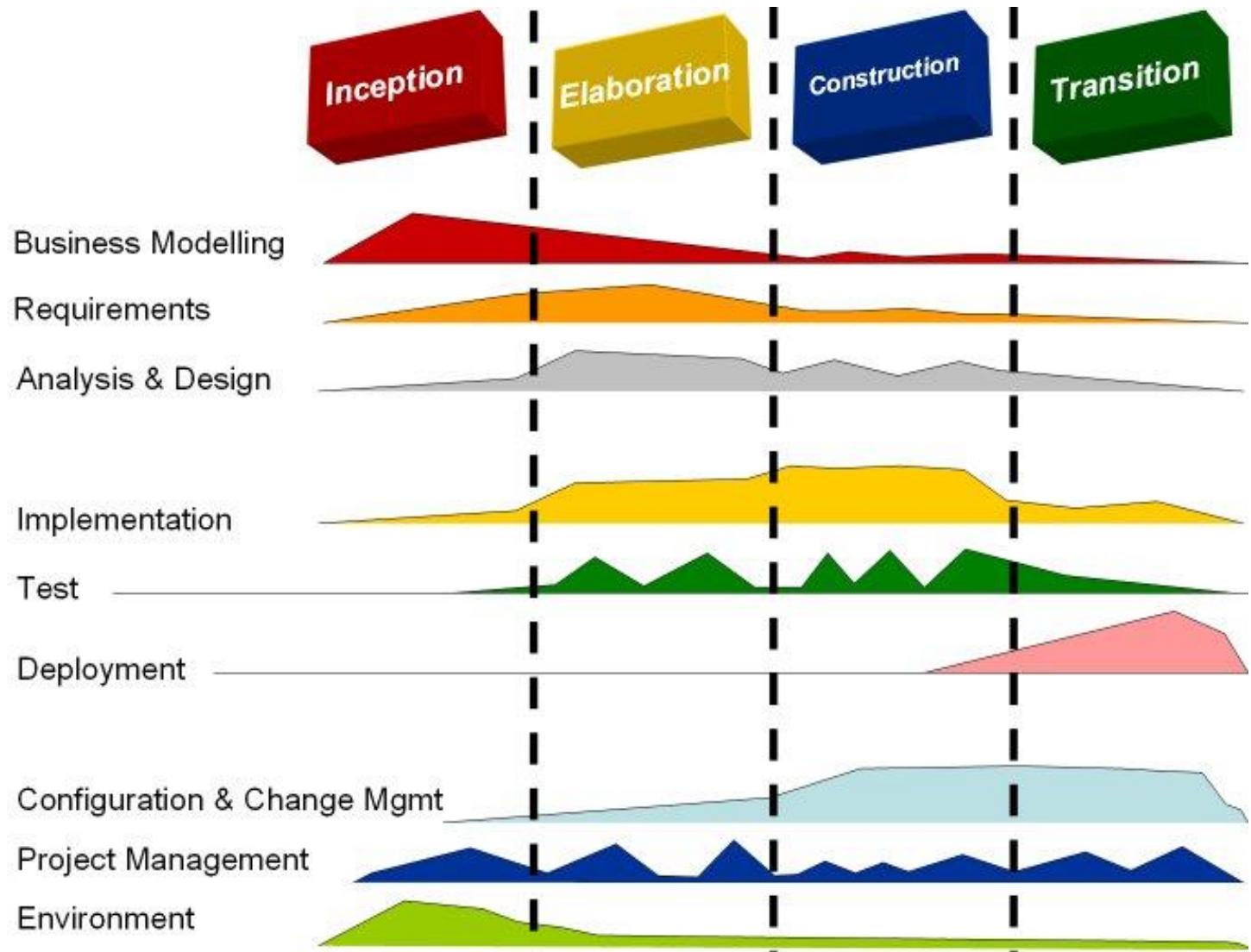
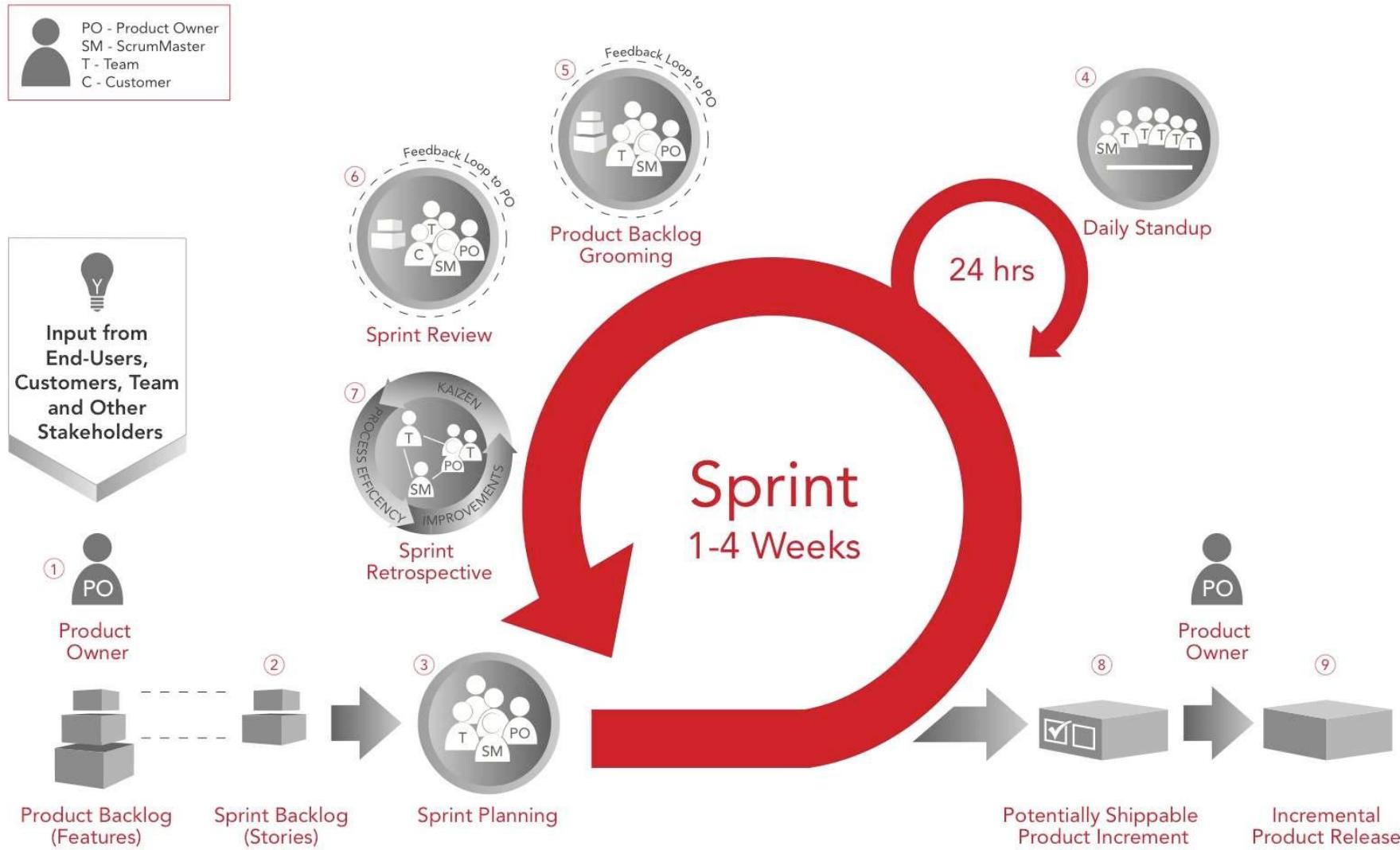


Figure 2. Spiral model of the software process.

Rational Unified Process

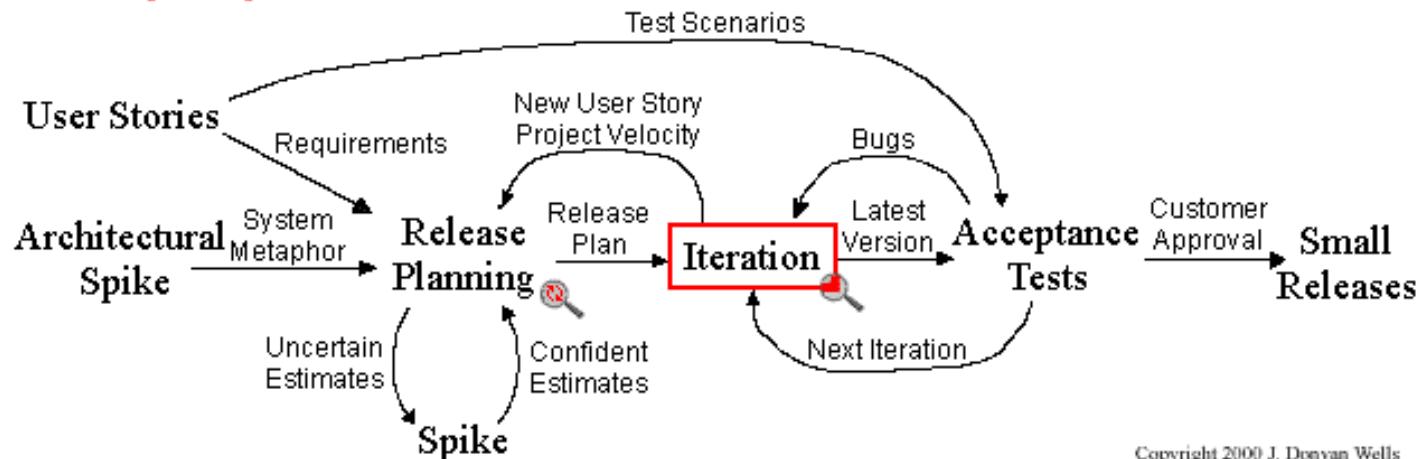


How Scrum Works



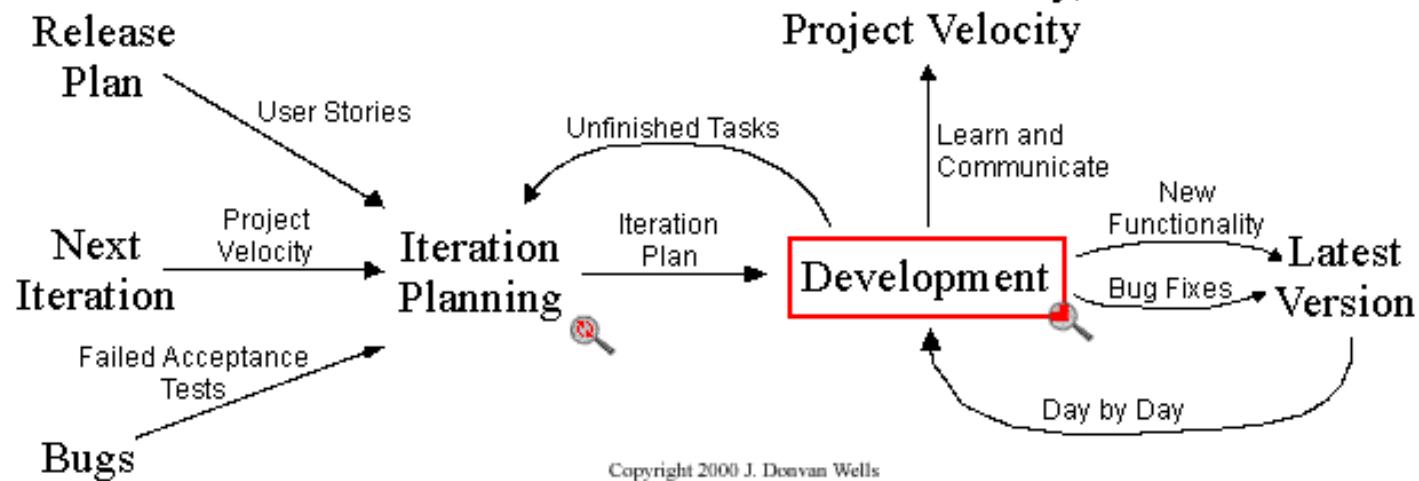


Extreme Programming Project



Iteration

Zoom Out



Quiz (kahoot)



20 min

Course structure and organization

Software Engineering

Code: L.EIC017 Acronym: ES

Keywords	
Classification	Keyword
OFICIAL	Informatics Engineering and Computing

Instance: 2021/2022 - 2S

Active? Yes

Responsible unit: [Department of Informatics Engineering](#)

Course/CS Responsible: [Bachelor in Informatics and Computing Engineering](#)

Cycles of Study/Courses

Acronym	No. of Students	Study Plan	Curricular Years	Credits UCN	Credits ECTS	Contact hours	Total Time
L.EIC	505	Syllabus	2	-	6	52	162

Teaching Staff - Responsibilities

Teacher	Responsibility
Ademar Manuel Teixeira de Aguiar	
João Carlos Pascoal Faria	

Teaching - Hours

Type	Teacher	Classes	Hour
	Totals	3	6,00
Lectures	Ademar Manuel Teixeira de Aguiar		3,00
	João Carlos Pascoal Faria		3,00
	Totals	20	40,00
	André Monteiro de Oliveira Restivo		4,00
	Daniel Ribeiro de Pinho		4,00
	José Carlos Medeiros de Campos		8,00
	Bruno Miguel Carvalhido Lima		4,00
	André Sousa Lago		4,00
	João Carlos Pascoal Faria		2,00
	Hugo José Sereno Lopes Ferreira		6,00
	Ademar Manuel Teixeira de Aguiar		4,00
	Filipe Alexandre Paix de Figueiredo Correia		4,00

Objectives

This course aims to acquaint students with the engineering and management methods necessary for the cost-effective development and maintenance of high-quality complex software systems.

Program

1. INTRODUCTION: large-scale software development challenges; objectives and scope of software engineering; history of software engineering.
2. SOFTWARE PROCESS: the concept of software process; process activities; process models; examples of software processes (RUP, XP, Scrum, etc.).
3. SOFTWARE PROJECT MANAGEMENT: project planning, monitoring, and control; software estimation; agile and classic project management.
4. SOFTWARE REQUIREMENTS: concept of software requirements; types of requirements; requirements elicitation, analysis, specification, and validation; requirements modeling with UML; user interface prototyping.
5. SOFTWARE DESIGN: architectural design; architectural modeling with UML; software reuse; detailed design.
6. SOFTWARE CONSTRUCTION AND EVOLUTION: development environments; continuous integration; version and change management; agile development with XP; software evolution and maintenance.
7. SOFTWARE VERIFICATION AND VALIDATION: basic concepts; unit, integration, system and acceptance testing; software inspections and reviews; defect tracking; static code analysis.

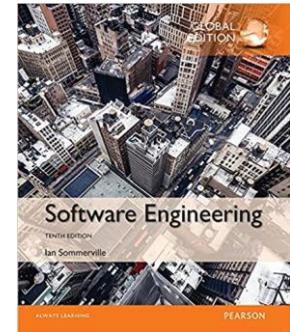
Course strategy

- Introductory (1st cycle) course on software engineering
- Theoretical and practical nature
 - Key concepts overviewed in lectures
 - Key activities, practices and tools used in practical classes, applied to a group project.
 - Detailed in subsequent labs, such as:
 - Database and Web Applications Laboratory (L.EIC)
 - Large Scale Software Development (M.EIC)
 - Project Management Laboratory (M.EIC)
- Breadth-first
 - In-depth elective courses on specific knowledge areas can be found in the 2nd cycle, for those that want to specialize in soft. engineering
 - Requirements Engineering (M.EIC & MESW)
 - Software Systems Architecture (M.EIC & MESW)
 - Advanced Software Construction Techniques (M.EIC)
 - Software Testing, Verification and Validation (M.EIC & MESW)
 - Software Evolution and Program Comprehension (MESW)

Course bibliography

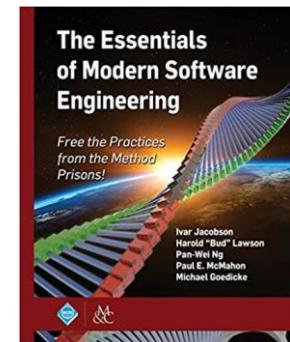
Mandatory

- Software Engineering (10th edition), Ian Sommerville, Pearson, 2015, ISBN: 9781292096131



Complementary

- Software Engineering Essentialized. Ivar Jacobson, Bud Lawson, Paul McMahon, Michael Goedicke. ACM Books. 2019. ISBN: 1947487248



- Russ Miles & Kim Hamilton; Learning UML 2.0, O'Reilly, 2006. ISBN: 0-596-00982-8
- Silva, Alberto Manuel Rodrigues da; UML, metodologias e ferramentas CASE (2^a edição). ISBN: 989-615-009-5

Grades

- Components:

- PQ – classification related to class participation and answering quizzes;
- TP – (individual) classification obtained in the team project;
- EF – classification obtained in the final exam.

- Formula:

$$CF = \text{round}(\min(10\% \text{ PQ} + 55\% \text{ TP} + 35\% \text{ EF}; \text{EF} + 4))$$

- Remarks:

- Final grades greater than or equal to 18 points may require an oral examination, which will cover all aspects of the UC.
- A minimum grade of 40% is required in all 3 components.
- The final grade cannot exceed in more than 4 points the grade obtained in the final exam.
- In the case of students exempt from the attendance rule, the PQ component doesn't apply; the final exam will have a weight of 45%.

Project: uni4all

- Contribute the **uni** project (from the students' association), with a set of **mini-apps** useful for the U.Porto community
- Open source
- Teams of 4-5 students
- Mini-apps to be proposed by the teams themselves
- Apps can use backend services (data storage, SIGARRA integration, etc.) to be supplied by a third-party
- Support by monitors (master students) outside classes



uni - A FEUP no teu bolso

NIAEFEUP Education

E Everyone

✗ This app is not available for your device

Add to wishlist

Project Technologies

- Languages
 - Dart, Java, Python, Javascript, whatever you like
- Models
 - UML
- Apps
 - Mobile, web, ... -> Flutter
- Tools
 - Code editors, UML editors, github, trello, ...



Important dates (tentative)

- 08.Apr.2022 – intermediate assessment of TP (iteration 0)
- 03.Jun.2022 – final delivery of TP (iteration 3)
- 31.Jun.2022 – exam @ moodle



Thank you

ademar.aguiar@fe.up.pt

jpf@fe.up.pt