

# Software Engineering — Introduction

Arkadiusz Chrobot

Department of Computer Science, Kielce University of Technology

Kielce, October 7, 2020

1 / 28

Notes

---

---

---

---

---

---

---

---

## Outline

Bibliography

Motto

Introduction

- Definition
- Genesis
- Current Situation
- Causes

Computer Science vs. Software Engineering

- Features of Software
- Programming Paradigms
- Software Development Paradigms

Software Development

- Model of the Software Development Process
- Waterfall Model
- Incremental Development
- Formal Methods
- Integration and Configuration
- Agile Methods

Miscellaneous Topics

- Software Engineering Methods
- Software Development Costs
- Computer-Aided Software Engineering
- Challenges

Summary

2 / 28

Notes

---

---

---

---

---

---


---

---

## Bibliography

 Ian Sommerville,  
*Software Engineering*,  
Pearson Education Limited, USA, 2016, on-line:  
<http://iansommerville.com>.

 Pierre Bourque, Richard E. Fairley,  
*SEWBOK v3.0 Guide to the Software Engineering Body of Knowledge*  
IEEE Computer Society, USA, 2014.

 Sungdeok Cha, Richard N. Taylor, Kyochul Kang,  
*Handbook of Software Engineering*,  
Springer Nature Switzerland AG, Cham, 2019.

 Gerald O'Regan  
*Concise Guide to Software Engineering*  
Springer International Publishing AG, Cham, Switzerland, 2017.

...and many other will be presented in slides and on the website!

3 / 28

Notes

---

---

---

---

---

---

---

---

## Motto

"If you look at software today, through the lens of the history of engineering, it's certainly engineering of a sort — but it's the kind of engineering that people without the concept of the arch did. Most software today is very much like an Egyptian pyramid with millions of bricks piled on top of each other, with no structural integrity, but just done by brute force and thousand of slaves."  
Alan Kay

4 / 28

Notes

---

---

---

---

---

---

---

---

## Introduction

Notes

---

---

---

---

---

---

---

---

### Software Engineering

“Software engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software, and the study of such approaches.”

IEEE 610.2 definition

### Software

Software — computer programs and associated documentation.

5 / 28

## Software Crisis

Notes

---

---

---

---

---

---

---

---

In the 60ies of last century the third generation of computer systems had been introduced to the market. That opened the possibility of developing complex software, but it soon had become apparent that it is very difficult. Nobody had known how to successfully build complex, efficient and reliable software. The majority of software projects ended with a disaster. The situation had become known as the *software crisis*. To overcome the problem in 1968 the NATO organized a conference in Garmisch, Germany. There the concept of software engineering was defined.

6 / 28

## Results

Notes

---

---

---

---

---

---

---

---

The principles of developing software successfully are still unknown. The overall situation in the software industry hasn't improved significantly:

- ▶ NIST estimates that in USA the annual financial loss caused by unsuccessful software projects is about \$60 billions,
- ▶ according to the Standish Group Chaos Report “...30% of all software projects are canceled, nearly half come in over budget, 60% are considered failures by the organizations that initiated them, and 9 out of 10 come in late.”,
- ▶ Peter G. Neumann — “Illustrative Risks to the Public in the Use of Computer Systems and Related Technology”

7 / 28

## Causes

Notes

---

---

---

---

---

---

---


---

Some of the causes:

- ▶ software projects are usually innovative,
- ▶ the product of software engineering is not physical,
- ▶ requirements in software projects change a lot,
- ▶ bad project management,
- ▶ „the exponents war”.

Sources:

 Terence Parr  
*Why Writing Software Is Not Like Engineering*  
<http://www.cs.usfca.edu/~parrt/doc/software-not-engineering.html>

 Chuck Allison  
*Code Quality*  
[This source is no longer available]

8 / 28

# Computer Science vs. Software Engineering

Notes

---

---

---

---

---

---

---

---

## Computer Science

How to create effective software? (algorithms, data structures, computational complexity, programming languages, programming paradigms).

## Software Engineering

How effectively create software? (project management, methods of coping with project complexity, software architecture, documentation, production costs, testing, reliability, maintenance).

9 / 28

# Features of Software

Notes

---

---

---

---

---

---

---

---

The features of “good” software:

- ▶ maintainability,
- ▶ reliability,
- ▶ effectiveness,
- ▶ usability.

10 / 28

# Programming Paradigms

Notes

---

---

---

---

---

---

---

---

Some of the main programming paradigms:

1. imperative
  - 1.1 procedural/structural (Pascal, C, Perl),
  - 1.2 object-oriented (C++, Java),
2. declarative
  - 2.1 functional programming (Erlang, LISP, JavaScript),
  - 2.2 logic (Prolog).

11 / 28

# Software Development Paradigms

Notes

---

---

---

---

---

---

---

---

There are several models of developing software. Some of them are:

- ▶ waterfall,
- ▶ incremental development,
- ▶ formal methods,
- ▶ integration and configuration,
- ▶ agile methods.

12 / 28

## Software Development

Notes

---

---

---

---

---

---

---

---

The software development is a process of creating a product which is a computer program (or computer programs). The course of this process depends on the type of the created software, but in each case four main phases can be found:

1. specification — easier for *generic software* and much harder in case of *custom software*,
2. development,
3. validation,
4. evolution.

13 / 28

## Model of the Software Development Process

Notes

---

---

---

---

---

---

---

---

A model of the process (or a paradigm) is a simplified description of the process from a particular perspective. The models can be classified as follows:

- ▶ workflow model,
- ▶ data flow model ,
- ▶ role activity model.

14 / 28

## Waterfall Model

Notes

---

---

---

---

---

---

---

---

The Waterfall Paradigm is the oldest Software Development Life Cycle model applied in Software Engineering. It was adopted from other engineering disciplines. In this model there are five main activities that are performed only once in the whole process: *requirements analysis and definition, system and software design, implementation and unit testing, integration and system testing* and *operation and maintenance*. This model organizes the work on software, but any change in requirements generates a lot of costs — the process has to start from the beginning. It is appropriate for software projects that are a part of a larger engineering projects, like for example space probe programs.

15 / 28

## Incremental Development

Notes

---

---

---

---

---

---

---

---

In the Incremental Development the product (software) is build gradually. First an initial version based on early specification is build and delivered to users. After getting feedback form the users the development team improves the product and the sends it back to them for further evaluation. The process is repeated until a required system is build. The activities that are common for all development models are interleaved in the Incremental Development rather than separate. This approach to software building can be combined with prototyping. The prototype can be crated as an experimental implementation of new functionalities and then incorporated to the main product or abandoned. In the latter case the prototype is build only to specify or discover requirements. The Incremental Development is suitable for projects in which requirements frequently change (in fact it is true for most of the software projects). The resulting product better responds to customer needs, but it internal structure may not be optimal. The progress of development may be hard to follow.

16 / 28

## Formal Methods

Formal Methods incorporate mathematical techniques into the software development process. Requirements for the product are written in a formal, strict specification language. The software is *derived* from its specification and its correctness is verified with the use of mathematical proofs. The resulting product is reliable and of high quality. Unfortunately, not any type of software can be developed like this. Moreover, the development is not always cost-effective. Nowadays Formal Methods are usually used for safety-critical products.

**Further readings:**

 [Gerald O'Regan](#)  
*Concise Guide to Formal Methods*  
Springer International Publishing AG, Cham, Switzerland, 2017

17 / 28

Notes

---

---

---

---

---

---

---

---

---

---

## Formal Methods

*“Ten years ago, researchers into formal methods (and I was the most mistaken among them) predicted that the programming world would embrace with gratitude every assistance promised by formalisation to solve the problems of reliability that arise when programs get large and more safety-critical. Programs have now got very large and very critical — well beyond the scale which can be comfortably tackled by formal methods. There have been many problems and failures, but these have nearly always been attributable to inadequate analysis of requirements or inadequate management control. It has turned out that the world just does not suffer significantly from the kind of problem that our research was originally intended to solve.”*

**Tony Hoare** in 1995

18 / 28

Notes

---

---

---

---

---

---

---

---

---

---

## Integration and Configuration

The main idea behind this approach to software development is to reuse components that have been created in previous projects, bought or are available as open-source software. The development can be fast and cheap, but usually some compromises regarding the software functionality have to be made. It means that the resulting product may not exactly be what the customer needs.

19 / 28

Notes

---

---

---

---

---

---

---

---

---

---

## Agile Methods

In 2001 several software professionals created and signed the Agile Manifesto:

*“ 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.”*

**Source:**

 <https://agilemanifesto.org/>

The Manifesto is accompanied by Twelve Principles of Agile Software.

20 / 28

Notes

---

---

---

---

---

---

---

---

---

---

## Agile Methods

The Agile Manifesto and Agile Principles led to the creation of many software development *methodologies* that are known as Agile Methods. Also some already existing methods that fulfill the requirements of the Agile Manifesto and Agile Principles were classified as Agile. Among them are: Lean Software Development, PDCA (Plan — Do — Check — Act), Kanban, Extreme Programming (XP), Feature-Driven Development, Dynamic System Development Method, Crystal, Scrum. Those methods can be wholly or partially combined with others, also those "non-agile". The main goal of Agile Methods is to provide the customer with a working product and reduce the amount of any additional work in the project. The Agile Methods seem to be suitable for small projects with small teams and with vague requirements. They do not scale well for bigger projects.

21 / 28

Notes

---

---

---

---

---

---

---

---

## Software Engineering Methods

The Software Engineering not only defines the models of software development process but also provides means (called Software Engineering Methods) that help to follow them. The methods allow software engineers to build models of computer programs that constitute the specification of the software. The UML (Unified Modeling Language), Object-Oriented or Structural Analysis are examples of such means.

22 / 28

Notes

---

---

---

---

---

---

---

---

## Software Development Costs

The overall costs of developing software depend on its type and the kind of the development process applied. Generally, the most expensive phase is the verification (testing), which takes up to 40% of the project budget and in case of safety-critical software it can even reach 50% of total costs. But more expensive than the development is the maintenance of the software (its modification after the deployment).

23 / 28

Notes

---

---

---

---

---

---

---

---

## Computer-Aided Software Engineering

To make the software development more efficient many tools were created, including a software that helps build another software. Those programs are commonly known as CASE — Computer-Aided Software Engineering. Those software tools can be split into two groups: UpCASE (project management software, UML editors, etc.), and LowCASE (IDEs, code repositories, software testing tools, and so on).

24 / 28

Notes

---

---

---

---

---

---

---

---

## Challenges

The most important challenges of the modern software engineering are:

- ▶ legacy software,
- ▶ diversity of systems,
- ▶ delivery times,
- ▶ software quality (security, reliability, etc.).

25 / 28

Notes

---

---

---

---

---

---

---

---

## Summary

The computer software becomes more and more complex. Also more and more aspects of human life depends on it. We (as humanity) do not know exactly how to develop reliable, efficient software. There are many failures in the short history of Software Engineering (recent example: MCAS software in Boeing 737-MAX), but also many successes (for example: the Voyager space probes). **Adhering to the principles of Software Engineering doesn't guarantee the success of the software project, but neglecting them is the recipe for assured failure.**

**For entertainment:**

 [Aaron Cummings, Uptime 15,364 days - The Computers of Voyager, https://www.youtube.com/watch?v=H62hZJVqs2o](https://www.youtube.com/watch?v=H62hZJVqs2o)

26 / 28

Notes

---

---

---

---

---

---

---

---

## Questions

?

27 / 28

Notes

---

---

---

---

---

---

---

---

## THE END

Thank You for Your attention!

28 / 28

Notes

---

---

---

---

---

---

---

---