Software Engineering

Architecture, Design and Patterns

Wladmir Cardoso Brandão
www.wladmirbrandao.com

Department of Computer Science (DCC)
Institute of Exact Sciences and Informatics (ICEI)
Pontifical Catholic University of Minas Gerais (PUC Minas)

The author thanks the support to:
Students who willingly help me in this task

Chapter 01

Key Concepts to Keep in Mind

We build things for people

- People → Clients
- Things → Software
- We → Engineers
- Build → Process

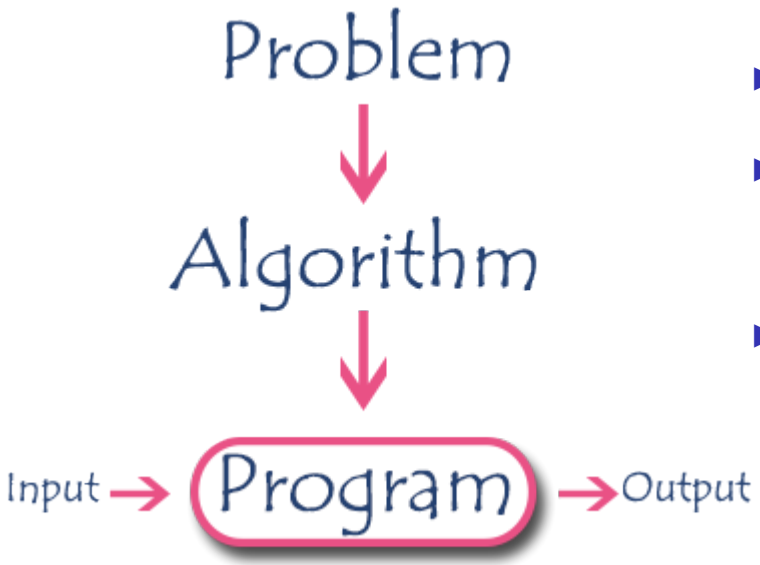
We *"systematically use scientific and technological knowledge, methods, and experience to design, implement, test, deploy and document software"*

IEEE

The Client is the King

- They've needs and demands
- Needs → expressed by **requirements**
- Demands → reflected in **software quality**
  - Functional → how well the software complies with functional requirements? Is it developed **correctly**?
  - Non-functional → how the software meets non-functional requirements that support the delivery of the functional requirements, such as scalability, robustness or maintainability? Is it **works as needed**?

Software solve Problems



- Software → algorithms + related data
- Provides unambiguous specification of how to solve a class of problems
- Algorithms are represented in human-readable source codes written by using programming languages

- Source code may be converted into an executable image by a compiler or executed immediately by an interpreter
- Related data usually includes machine codes, interpreted codes, libraries, documentation, and digital media

The word algorithm derives from the 9th Century Persian mathematician Muhammad ibn Mūsā al′Khwārizmī, latinized "Algoritmi"

Building software is a hard work

"There is no single development, in either technology or management technique, which by itself promises even one order-of-magnitude improvement within a decade in productivity, in reliability, in simplicity"

Fred Brooks, 1986

- Brooks divides the problems facing SE into:
  - Essence → intrinsic difficulties in the nature of software
  - Accident → difficulties related to the production
- For Brooks, most techniques attack the accidents

Factor of 10 improvement?

1. The accidents of SE should account for 90% of the overall effort
2. Tools should reduce accidental problems to zero

- Brooks doesn't believe that the former is true
- The latter is almost impossible. New tools and techniques solve some problems while introducing others

Accidental effort is not 90% of the job, its much smaller than that, and reducing it to zero will not give us an order-of-magnitude improvement

Engineering principles applied to software production

- Software requirement → elicitation, analysis, specification, and validation of requirements
- **Software design** → defining the architecture, components, interfaces, and characteristics
- Software construction → coding, verification, unit and integration testing, and debugging
- Software testing → technical investigation to attest software quality
- Software maintenance → activities required to provide cost-effective support to software

References I

- Sommerville, Ian
  Software Engineering.
  10ed, 2016.
- Larman, Graig
  Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development.
  3ed, 2004.
- Pressman, Roger; Maxim, Bruce
  Software Engineering: A Practitioner's Approach.
  8ed, 2014.
- Booch, Grady; Rumbaugh, James; Jacobson, Ivar
  The Unified Modeling Language User Guide.
  2ed, 2005.

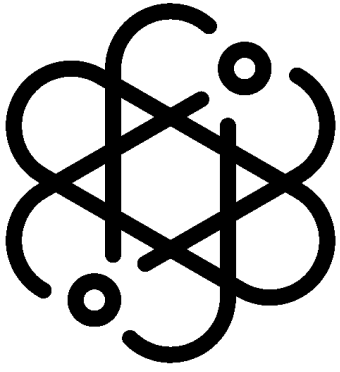
References II


Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John
Design Patterns: Elements of Reusable Object-Oriented Software.
1ed, 1994.


Buschmann, Frank; Rohnert, Hans; Stal, Michael; Sommerlad, Peter; Meunier, Regine
Pattern-Oriented Software Architecture, A System of Patterns.
1ed, 1996.


Frederick P. Brooks
The Mythical Man-Month.
Proceedings of the IFIP Tenth World Computing Conference, 1986.

THANK YOU



IRIS
RESEARCH

Wladmir Cardoso Brandão
www.wladmirbrandao.com
wladmir@pucminas.br

"Science is more than a body of knowledge. It is a way of thinking."
Carl Sagan