Definition of Software Engineering

Introduction to Software Engineering

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Tao Xie, Professor
Computer Science @ Illinois
Learning Objectives

By the end of this video, you will be able to

• Explain example definitions of software engineering.
• Explain key points emphasized when defining software engineering.
• List example topics in software engineering.
Some Definitions of Software Engineering

• “The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.” (Bauer, 1971)

• “The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software.” (IEEE, 1990)

• “Software engineering is the branch of computer science that creates practical, cost-effective solutions to computation and information processing problems, preferentially by applying scientific knowledge, developing software systems in the service of mankind.” (Shaw, Herbsleb, Ozkaya, & Root, 2005)
Key Points of Software Engineering

• Systematic application of process
  • “sound engineering principles”
  • “the application of a systematic, disciplined, quantifiable approach”
  • “applying scientific knowledge”

• Practical, economic value
  • “obtain economically software”
  • “creates practical, cost-effective solutions”
  • “developing software systems in the service of mankind”

• Quality
  • “software that is reliable and works efficiently”
Example Topics Studied in Software Engineering

- Process
- Tools
- Techniques
- Models (of software development)
- Modeling (of developed systems)
Phases in Software Development Cycle - 1

• Requirements
  • Specify what the software should do
  • Analysis to eliminate/reduce ambiguities, inconsistencies, incompleteness

• Design
  • Split software into modules
  • Specify overall structure and interaction of these modules
  • Checking conformance to requirements
Phases in Software Development Cycle - 2

• Implementation
  • Specify **how** the modules work
  • Unit-testing each module in isolation

• Integration
  • Specify **how** the modules interact
  • System-testing module interactions

• Maintenance
  • Evolve software as requirements change
  • Regression-testing changes
References


The End