Software Engineering in Brief

CSCI 21, Spring 2018

Overview

1. Software engineering
2. Programming paradigms
3. Software development methodologies
4. Tools
5. Daily/weekly procedures

Software engineering (definition)

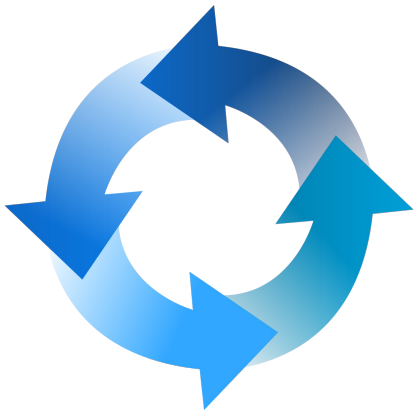
...the application of a systematic,disciplined,quantifiable approach to the development,operation and maintenance of software; that is, the application of engineering to software.

… in order to obtain economically software that is reliable and work efficiently on real machines.

Software engineering (process)

1. **Requirements**
2. **Systems Analysis**
3. **System Design**
4. **Code Design**
5. **Implementation**

6. Testing and Debugging
7. Deployment
8. Maintenance



Software engineering (process explained)

1. **Requirements**

   User expectations for a new or modified product.

2. Systems Analysis
3. System Design
4. Code Design
5. Implementation
6. Testing and Debugging
7. Deployment
8. Maintenance

Software engineering (process explained)

1. Requirements
2. **Systems Analysis**

   Studying a process in order to identify its goals and purposes and create systems and procedures that will achieve them in an efficient way.

3. System Design
4. Code Design
5. Implementation
6. Testing and Debugging
7. Deployment
8. Maintenance

Software engineering (process explained)

1. Requirements
2. Systems Analysis
3. **System Design**

   Defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements.

4. Code Design
5. Implementation
6. Testing and Debugging
7. Deployment
8. Maintenance

Software engineering (process explained)

1. Requirements
2. Systems Analysis
3. System Design
4. **Code Design**

   *Let's talk about **programming paradigms**.*

5. Implementation
6. Testing and Debugging
7. Deployment
8. Maintenance

Programming paradigms (definition)

...a style, technique, and culture of computer programming.

...a way to classify programming languages based on their features.

Programming paradigms (examples)

- Procedural paradigm
  - Describe exactly each step that must be followed to solve the problem
  - Often written to be *domain specific*
  - Ex: COBOL (business) and FORTRAN (science and engineering)
- Functional paradigm
  - Programs designed around discrete mathematical functions
  - Immutable data and heavily recursive
  - Ex: Haskell and Scala
- **Object-oriented paradigm (!)**
  - Programs organized using objects that contain data and methods to work on that data
  - Ex: Java and **C++** (!)

Software engineering (process explained)

1. Requirements
2. Systems Analysis
3. System Design
4. Code Design
5. **Implementation**

   Let's talk about software development methodologies.

6. Testing and Debugging
7. Deployment
8. Maintenance

Software development methodologies (definition)

...a framework that is used to structure, plan, and control the process of developing a software product.

Software development methodologies (examples)

- Waterfall
  - a project is broken up into distinct stages that must be completed in sequence.
- Prototyping
  - a prototype is built, tested, and reworked as necessary until an acceptable prototype is finally achieved from which the complete system or product can now be developed
- Spiral
  - combines the features of the prototyping model and the waterfall model; favored for large, expensive, and complicated projects.
- **Agile (!)**
  - an iterative approach that focuses on being lean, and producing minimum viable products (MVPs) over set periods of time while improving with each iteration.

Software development methodologies (discuss)

Now, please take a few minutes to read: https://en.wikipedia.org/wiki/Agile_software_development

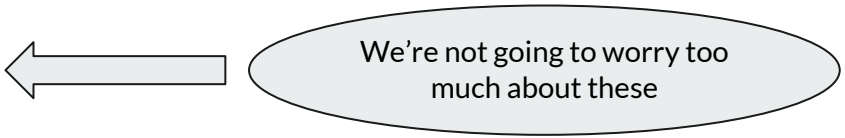
Be on the lookout for the following terms:

- Iterative, incremental and evolutionary
- Efficient and face-to-face communication
- Very short feedback loop and adaptation cycle
- Quality focus

Software engineering (process explained)

1. Requirements
2. Systems Analysis
3. System Design
4. Code Design
5. Implementation

6. Testing and Debugging
7. Deployment
8. Maintenance



Tools

- g++ (compiler)
- make (build automation)
- Catch2 (unit testing)
- gdb (debugging)
- GitHub (version control)

Daily/weekly procedures (challenges)

- Programming Challenges
  - Submitted before 12pm/noon each Friday
  - Must be emailed with:
    - All source code
    - Unit testing output file
    - Certification of **peer code review** completion

Daily/weekly procedures (projects)

- Programming Projects
  - Submitted before midnight on their due date
  - Must be emailed with:
    - All source code
    - Unit testing output file
    - Certification of **peer code review** completion

Daily/weekly procedures (contributions)

- Each one of us will make one **unique contribution** to the class
- Guidelines:
  - Should require reasonable but also significant effort
  - Must be something no one else has already done for the class
  - Must contribute to our learning, well-being, or something else that is positive and constructive
  - Can be something assigned by the instructor or something that you propose

Though no course credit will be awarded for multiple unique contributions, you are invited to make as many unique contributions to the course as you would like.


Ceci n'est pas une peep.