CS342 Software Engineering

Dr. Ismail Hababeh
German Jordanian University

Lecture 2

Adapted from Software Engineering, by Dr. Paul E. Young & slides by Dr. Mohammad Daoud

Software Development Life-Cycle

Series of software development steps, from concept exploration through final retirement:

1. **Requirements phase** (concept explored, includes rapid prototyping)
2. **Specification/Analysis phase** (contract)
3. **Design phase**
   a) high-level (architectural design => modules)
   b) detailed (design of each module)
4. **Implementation phase** (coding)
5. **Testing phase** (coding/testing)
   a) Unit testing
   b) Integration of sub-systems
6. **Maintenance phase** (any changes after acceptance)
7. **Retirement**

Need of Software Development Life Cycle

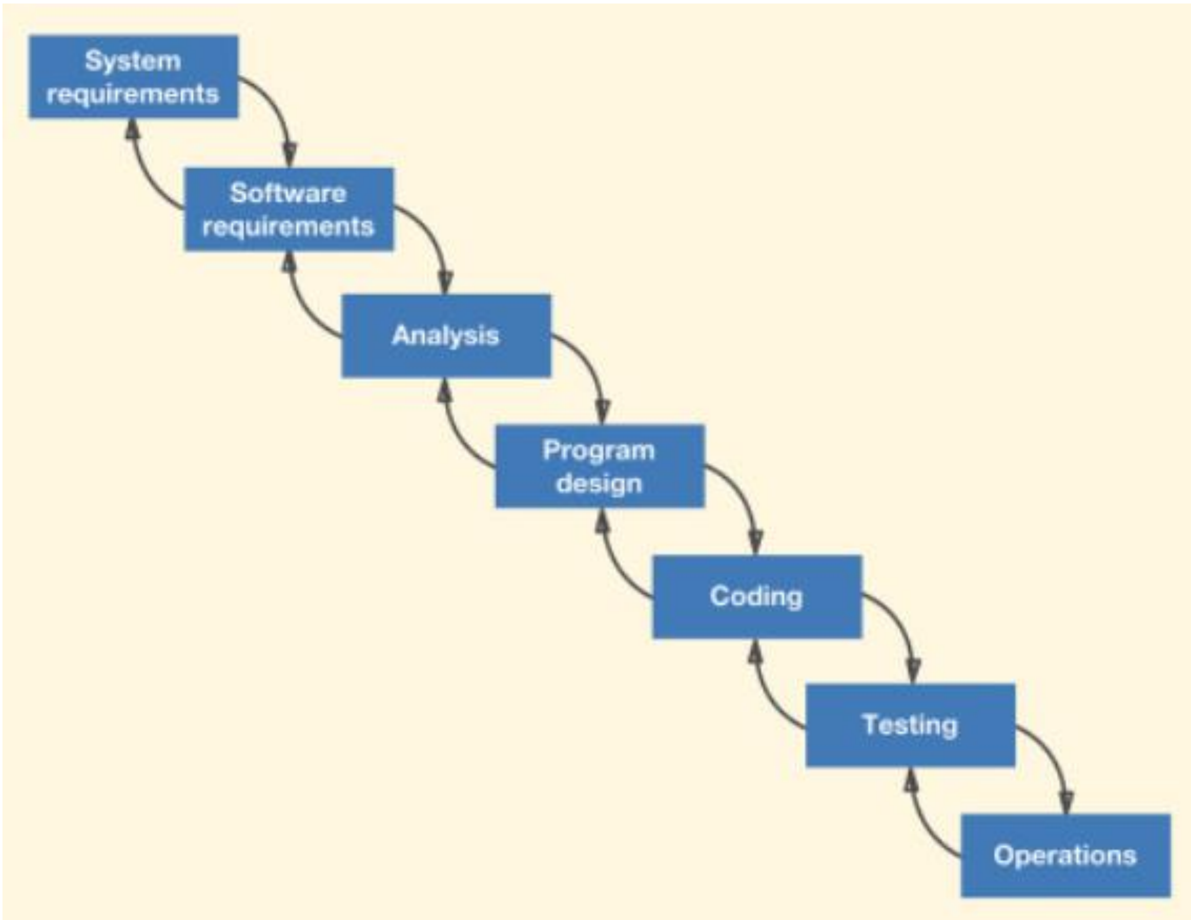
- Allows a systematic and well-organized way to develop a software.
  - –Team members know to do what and when
- Allows development of large software projects.
- Allows smooth interfacing between different development sectors:
  - Helps in identifying inconsistencies, redundancies, and omissions in the development process.
  - Helps in modifying a process model for specific projects.

Software Life-Cycle Model

- Life-cycle model:
  - The steps (*phases*) to follow when building software
  - a description of the sequence of activities carried out in a project, and the relative order of these activities.

Classical Software Life-Cycle Model

Waterfall Classical model: a linear, sequential approach to the software development.



Software Typical Classical Phases

1. Requirements:
   – Explore the concept
   – Extract the client's requirements
2. Analysis (specification):
   – Analyze the client's requirements
   – Draw up the specification document
   – Draw up the software project management plan
   – "What the product is supposed to do"

Software Typical Classical Phases

3. Design:
   – Architectural design, followed by
   – Detailed design
   – "How we do it"
4. Implementation (Coding):
   – Choosing the language
   – Unit testing
   – Integrated testing

Software Typical Classical Phases

5. Testing:
   - Product testing
   - Acceptance testing (done by the client)
6. Post-delivery maintenance:
   - Corrective maintenance
   - Adaptive maintenance
   - Perfective maintenance (client wants to increase products’ functionality)
6. Retirement (product is removed from service)

Classical and Modern Views of Maintenance

- Classical Maintenance is defined in terms of the time at which the activity is performed.
  - Development-then-maintenance model

Classification of Development Fault and Maintenance Fault

- Classical Development Fault:

  A fault is detected and corrected before software installation.

- Classical Maintenance Fault:

  A fault is detected and corrected after software installation.

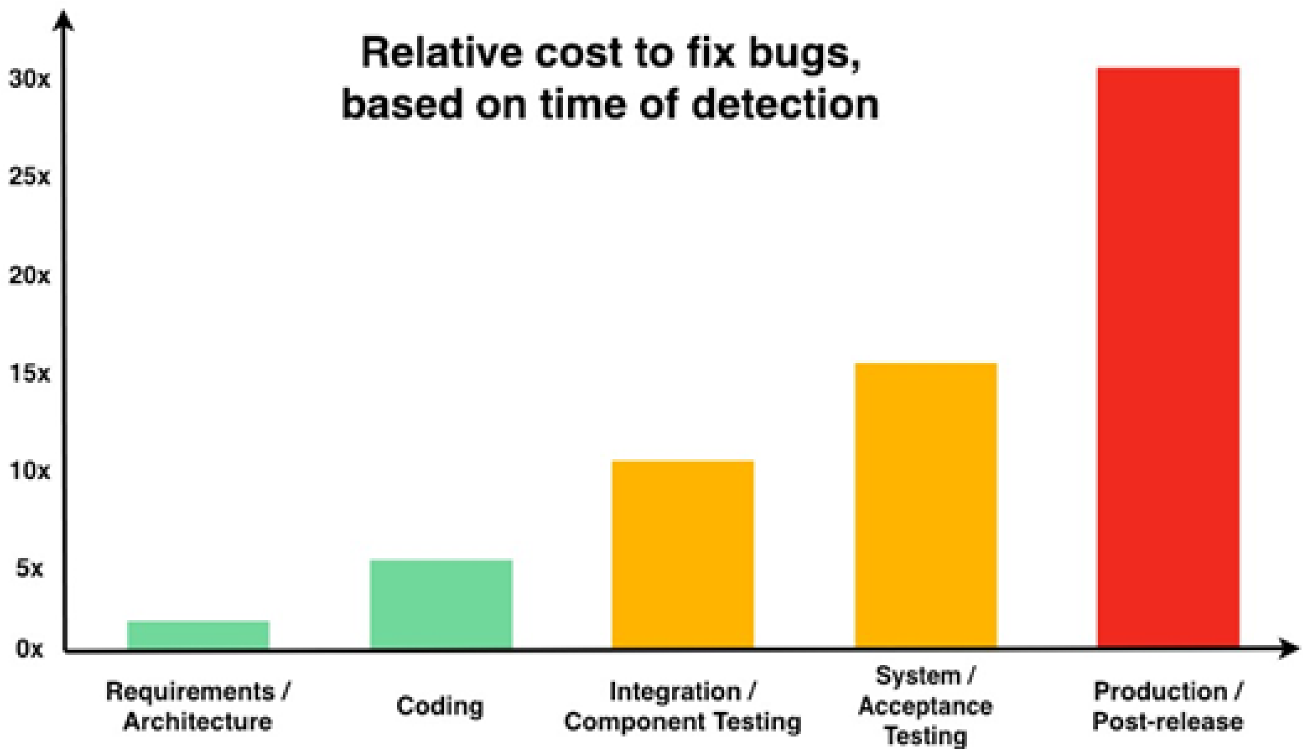
Modern Maintenance Definition

- The International Standards Organization (ISO) and International Electro-technical Commission (IEC) defined *maintenance operationally.*

  "*The process that occurs when a software is modified because of a problem or a need for improvement or adaptation. Regardless of whether this takes place before or after installation of the software product*".

Fault Detection and Correction

- The earlier we detect and correct a fault, the less it costs.



Fault Detection and Correction

- To correct a fault early in the software life cycle (before coding) usually needs documents to be changed.
- To correct a fault late in the software life cycle (after installation) needs:
  - Change the code and the documentation
  - Test the change itself
  - Perform regression testing (any type of software testing that seeks to detect new errors)
  - Re-install the product.

Fault Detection and Correction

- In general, 60% - 70% of the faults in large-scale products found in early phases: requirements, analysis, and design.
- It is important to develop requirements, analysis, and design techniques to:
  - Find faults as early as possible
  - Reduce the overall number of faults and, hence, the overall cost.

Team Programming Aspects

Organizations can easily afford hardware that can run large products. However, these products are too large to be done by one developer.

- An aspect of software engineering is to manage and organize teams.
- The software built by teams may:
  - Interfacing problems between modules
  - Communication problems among team members

Software Engineering Paradigms

- Structured paradigm:
  - Structured methods are either process (action) oriented or data oriented but not both.
- Object-Oriented paradigm:
  - Object-Oriented methods are for both process (action) and data.

The Structured Paradigm

- Successful with small products.
- Fails with large products (> 50,000 lines of code)
- Problems of structured paradigm:
  - Large number of code lines may increase number of errors.
  - Increases cost of maintenance, because structured methods are either process oriented or data oriented; but not both.

Software Structured Methods

- Process-oriented

includes real time computing, automats, communication and protocols, etc.

- Data-oriented

includes databases, transactional computing, data retrieval systems, multimedia systems, web data, … etc.

The Object-Oriented Paradigm

- An object is a software component that incorporates both data and actions performed on that data.
- Data and actions are of equal importance
- Example: Bank Account
    - Data: account balance
    - Actions: deposit, withdraw, ….etc.