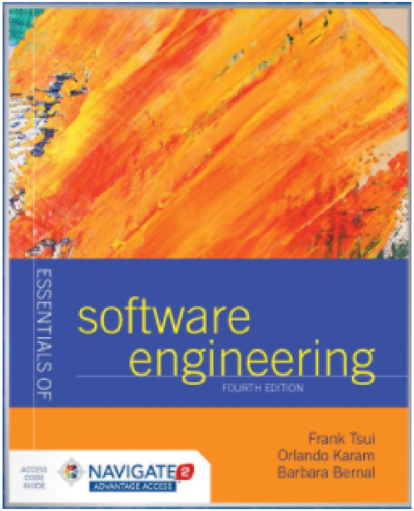
Software Process Models

CSC 354, Software Engineering I

Chapter 4: Software Process Models



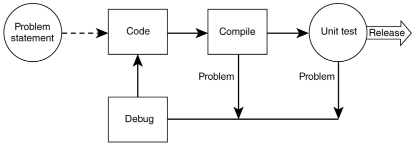
What is a Process Model?

- It is a description of:
    - what *tasks* need to be performed in
    - what *sequence* under
    - what *conditions* by
    - *whom* to achieve the "desired results"

Why Have a Process Model?

- It provides *guidance* for a *systematic*
    - coordination and controlling
    - of the tasks
    - and the personnel who perform the tasks

A “Simple and Familiar” Process



- Most programmers perform and follow this simple process (but some unfortunately skip unit testing and debugging)
- Some programmers proceed without thoroughly considering and understanding the “problem statement” (that is, the requirement)

Extending the “Simple” Process

- As projects got larger and more complex:
  - Needed to clarify and stabilize the *requirements*
  - Needed to *test* more functionalities
  - Needed to *design* more carefully
  - Needed to use more existing software and tools (for example, databases, version control, etc.)
  - Need more *people* to be involved

More People and More Tasks

- Needed to *define*:
    - the set of *tasks* that need to be performed
    - the *sequence* of flow of the tasks
    - the *input* and *output* from these tasks
    - the *preconditions* and *postconditions* for each task
    - The *people* and *skills* needed to perform the tasks

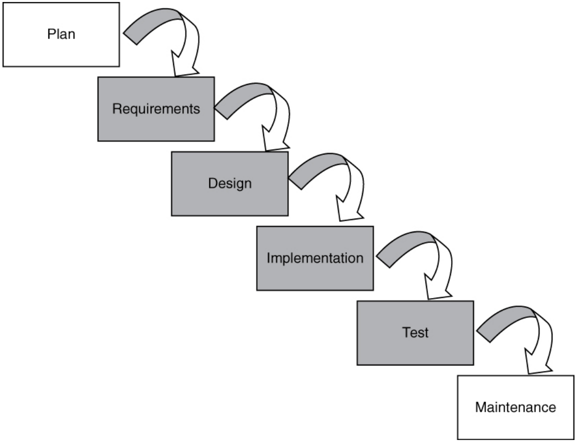
Traditional Software Development Processes

- The earlier “simple” process was employed by many years *without* formally embracing other important development activities such as:
    - requirements analysis
    - design
    - formal testing
    - packaging

Traditional Software Development Processes

- The recognition of the need for formal processes was initially driven by failures in developing large complex software
- **Waterfall**: earliest process to cope with no process
- **Incremental**: decomposing the large systems
- **Spiral**: risk management
- **Rational Unified Process**: multiple development and management issues

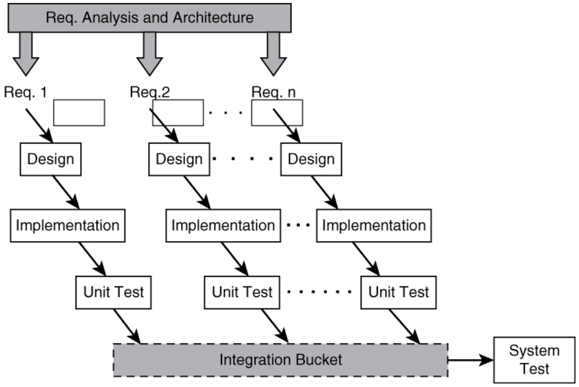
Waterfall Model



Waterfall Model

1. Requirements must be specified
2. Four main tasks must be completed in *sequence*: requirements, design code, and test
3. The output of one stage feeds into the next stage in the sequence, and thus is easily *tracked* by management

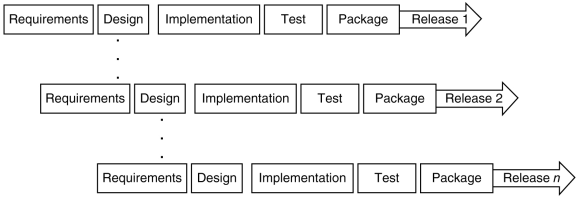
Incremental Model (Continuous Integration)



Incremental Model (Continuous Integration)

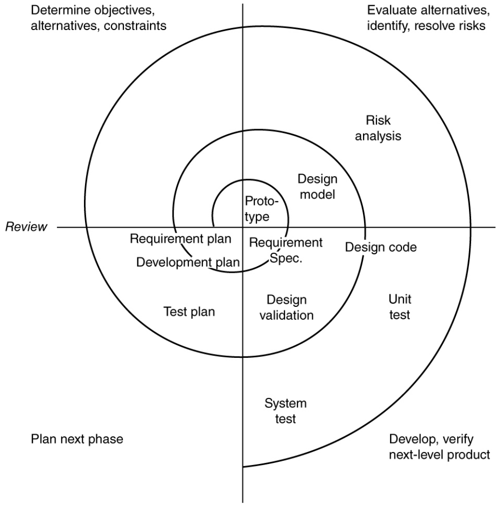
- Each “major requirement” is developed separately through the sequence: requirements, design, code, and test
- As the developed pieces are completed, they are continuously merged and integrated into a common bucket for the *integrated system test*

Incremental Model (Multiple Releases)



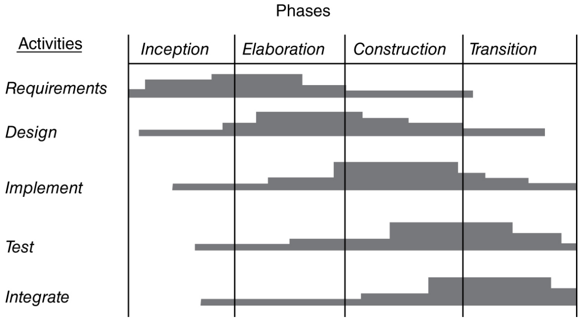
- Each small set of requirements is developed, packaged, and released in multiple releases

Spiral Model



- Software development activities are cycled through the four phases

Rational Unified Process (RUP)



- Every software activity is "addressed" in the four phases of: inception, elaboration, construction, and transition

Assessment of Software Organizations

- Software development and software support may be done with little process or with sophisticated process, well-defined, well-organized, and well-executed processes
- How mature is your software engineering organization and do you need to improve?
- ISO (ISO 9000 series) and Software Engineering Institute (SEI) are two leading organizations that help in process assessment