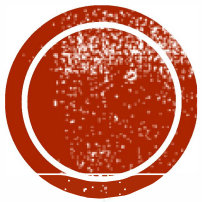
SOFTWARE ENGINEERING

Lecturer Dr. Osman AKBULUT



SOFTWARE DEVELOPMENT METHODOLOGIES: HEAVYWEIGHT

OUTLINE

- Problems with Software Production
- Basic Life – Cycle
- Heavyweight Methodologies
  - Build&Fix
  - Waterfall
  - Boehm's Spiral
  - Prototyping
  - Incremental

PROBLEMS WITH SOFTWARE PRODUCTION

- Complexity
- Conformity
- Changeability
- Invisibility

Fred Brooks, 1986,

"No Silver Bullet"

*conformity: riayet

*changeability: istikrarsızlık

SOFTWARE'S BASIC LIFE–CYCLE

- Requirements Phase
- Specification Phase
- Design Phase
- Implementation Phase
- Integration Phase
- Maintenance Phase
- Retirement Phase

REQUIREMENTS PHASE (REQS.)

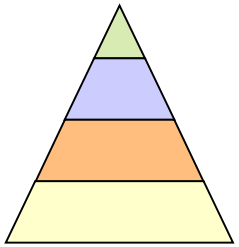
- Defining constraints
  - Functions
  - Due dates
  - Costs
  - Reliability
  - Size
- Types
  - Functional
  - Non-Functional

SPECIFICATION PHASE (SPECS.)

- Documentation of requirements
  - Inputs&Outputs
  - Formal
  - Understandable for user&developer
  - Usually functional reqs. (what to do)
  - Base for testing&maintenance
- The contract between customer & developer ?

DESIGN PHASE

- Defining Internal structure (how to do)
- Has some levels (or types of docs)
  - Architectural design
  - Detailed design
  - ...
- Important;
  - To backtrack the aims of decisions
  - To easily maintain



IMPLEMENTATION PHASE

- Simply coding
- Unit tests
  - For **verification**

INTEGRATION PHASE

- Combining modules
- System tests
  - For **validation**
- Quality tests

MAINTENANCE PHASE

- Corrective
- Enhancement
  - Perfective
  - Adaptive
- Usually maintainers are not the same people with developers.
- The only input is (in general) the source code of the software?!?

RETIREMENT PHASE

- When the cost of maintenance is not effective.
  - Changes are so drastic, that the software should be redesigned.
  - So many changes may have been made.
  - The update frequency of docs is not enough.
  - The hardware (or OS) will be changed.

METHODOLOGIES

- Types:
  - Heavyweight (Classical)
  - Lightweight (Agile)

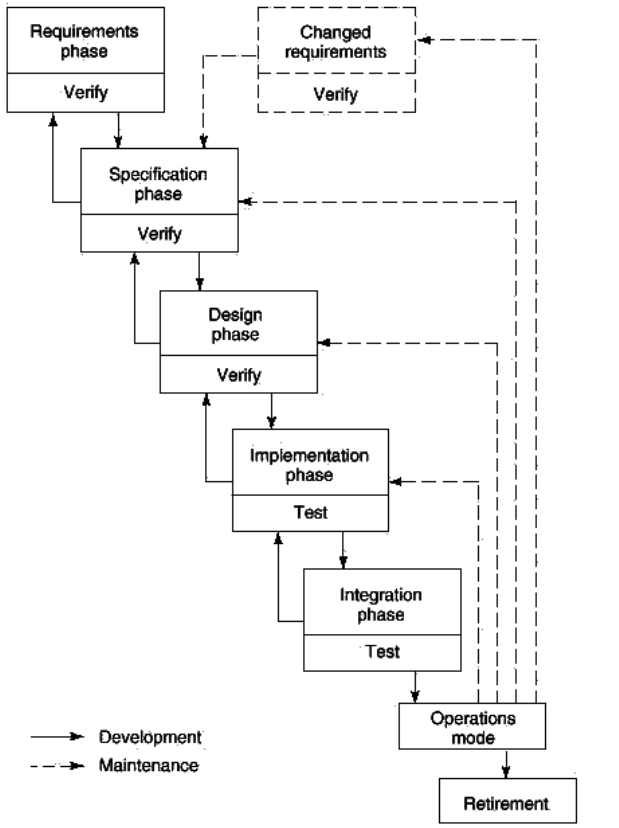
HEAVYWEIGHT METHODOLOGIES

- Classical Methodologies
- Founded before 1990s
- Documentation based
- Disciplined
- Well controlled

WATERFALL (FLOW)

Figure 3.2
Waterfall model.

TM-9



WCB/McGraw-Hill

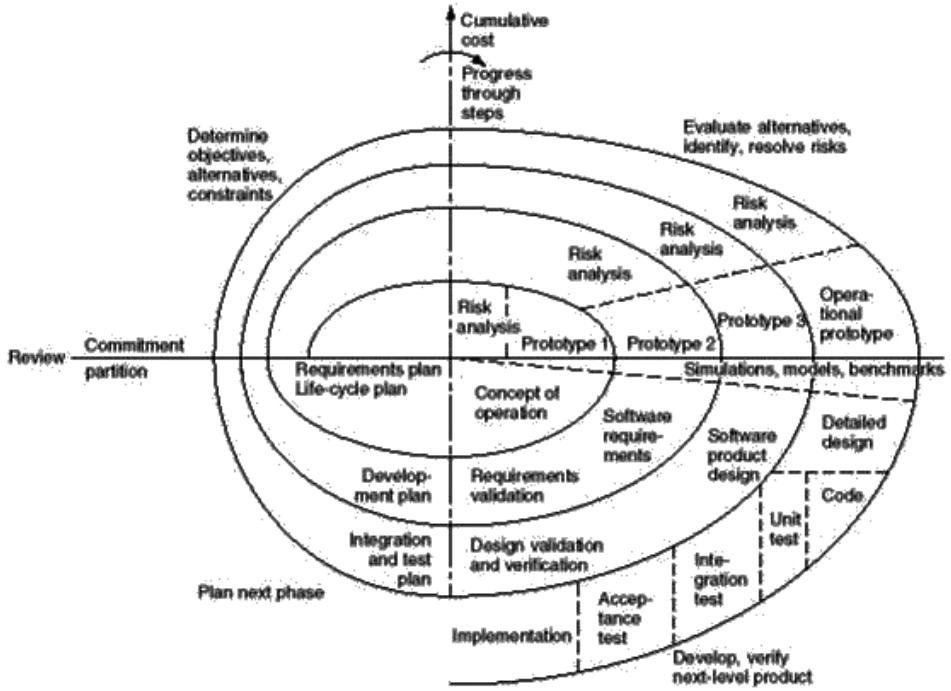
WATERFALL

- Royce, 1970
- Embedded testing
- Advantages
  - Enforced disciplined approach
  - Well documented
  - Checked by SQA at all phases
- Disadvantages
  - Well documented
  - Too Technical !

BOEHM'S SPIRAL (FLOW)

Figure 3.8
Full spiral model [Boehm, 1988]. (© 1988 IEEE)

TM-15



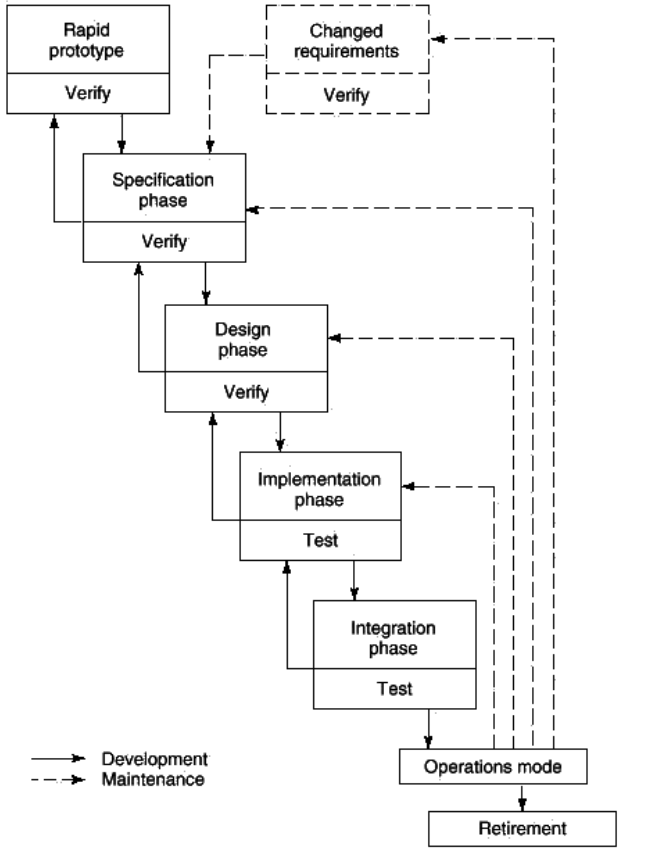
WCB/McGraw-Hill

BOEHM'S SPIRAL

- Boehm, 1988
- Best for internal projects (in-house)
- Best for large scale projects
- Advantages
  - Risk driven
  - Supports reuse
  - Incorporation of Software Quality
- Disadvantages
  - Spending too many for testing and analysis
  - Risk driven

PROTOTYPING (FLOW)

Figure 3.3 TM-10
Rapid prototyping model.



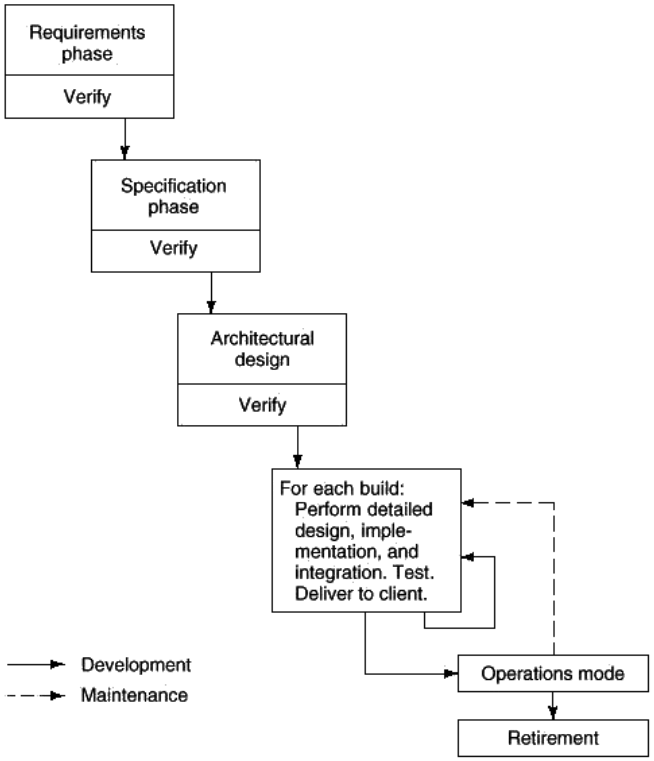
PROTOTYPING

- Changed Requirement Analysis Phase
- Advantages
  - Less Feedback Loops
  - Better Specification
- Risk
  - Require rapid Prototyping

INCREMENTAL (FLOW)

Figure 3.4
Incremental model.

TM-11



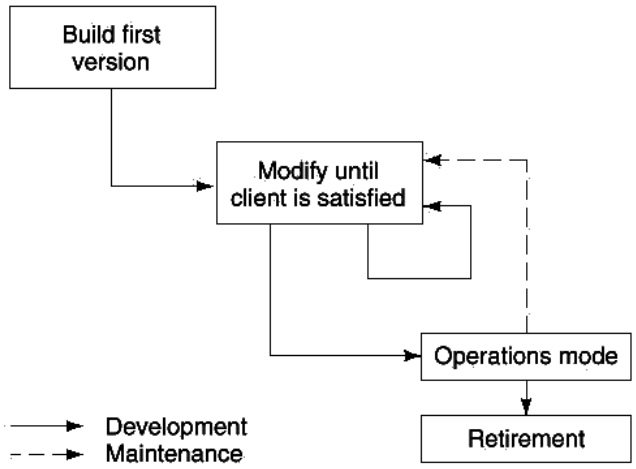
INCREMENTAL

- Advantages
  - Product is available within weeks
  - Reduces traumatic effect of new sw
  - Requires no large capital outlay
- Disadvantages
  - Not to destroy existing structure with new builds
  - Can be degenerated to Build-Fix

BUILD&FIX (FLOW)

Figure 3.1
Build-and-fix model.

TM-8



BUILD&FIX

- Construction with NO specification and design.
- May work well on short programming exercises.

QUESTIONS?

CE 310–Software Engineering

Lecturer Dr. Osman AKBULUT