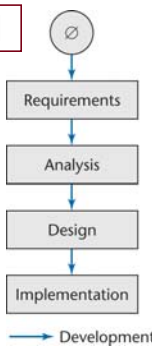
Chapter 2

Software Life Cycle Models

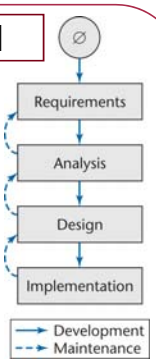
Idealized Model

- Linear
- Start from scratch
- "Greenfield"
- Got it right the first time

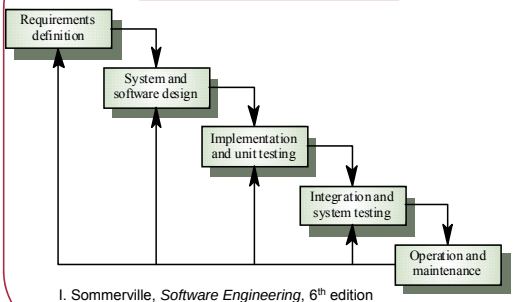


Waterfall Model

- Linear with feedback

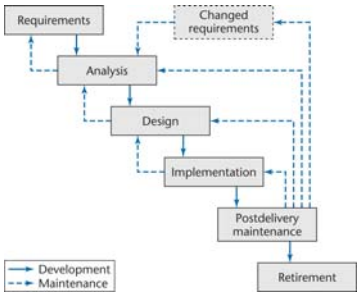


Waterfall Model



I. Sommerville, *Software Engineering*, 6th edition

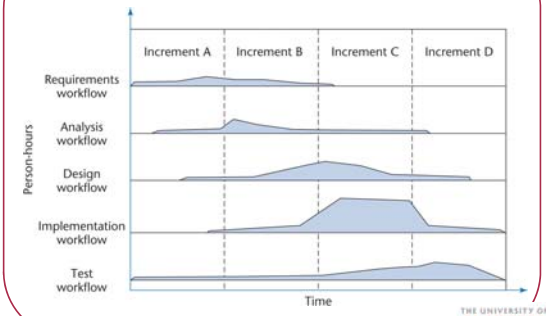
Waterfall Model



Waterfall

- Pros ?
- Cons ?

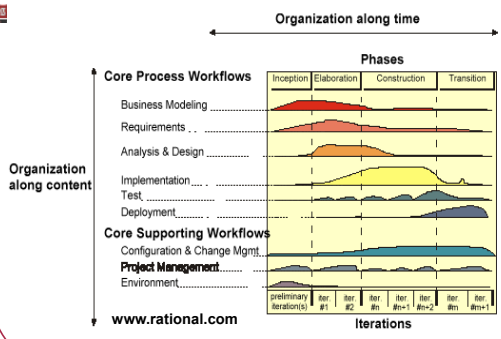
Iterative and Incremental



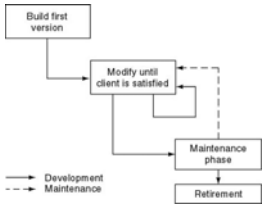
Iterative and Incremental

- All "flows" (activities) are done in each increment but to varying degrees.
- Multiple opportunities to test, receive feedback, and adjust.
- Specific expectations (deliverables) for each increment and each workflow
- REDUCE RISK

(Rational) Unified Process



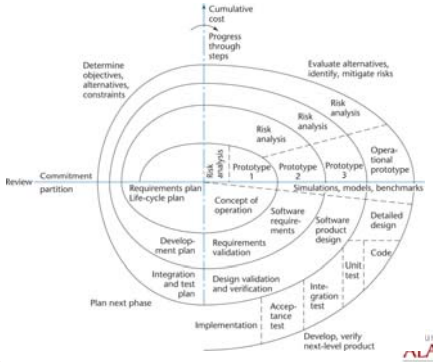
Evolutionary development Build-and-Fix



- No design
- No specifications
  - Maintenance nightmare
- The easiest way to develop software
- The most expensive way

Stephen R. Schach, *Object-Oriented and Classical Software Engineering* 8th Edition

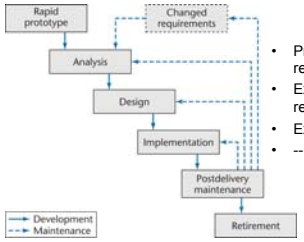
Spiral Model



Evolutionary development

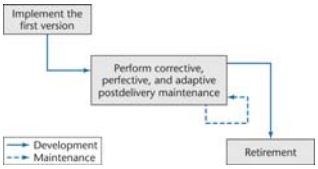
- Problems
  - Lack of process visibility
  - Systems are often poorly structured
  - Special skills (e.g. in languages for rapid prototyping) may be required
- Applicability
  - For small or medium-size interactive systems
  - For parts of large systems (e.g. the user interface)
  - For short-lifetime systems

Rapid Prototyping



- Prototype to define requirements
- Explore poorly understood requirements
- Explore new technology
- --RAPID--

Open Source

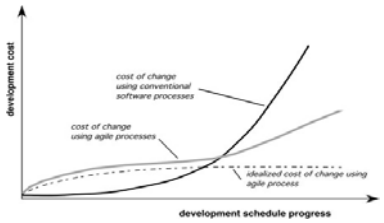


- Core Group with “Support Group”
- Quick releases
- Lacks documentation & design

Agile Processes

- Is driven by customer descriptions of what is required (scenarios)
- Recognizes that plans are short-lived
- Develops software iteratively with a heavy emphasis on construction activities
- Delivers multiple ‘software increments’
- Adapts as changes occur

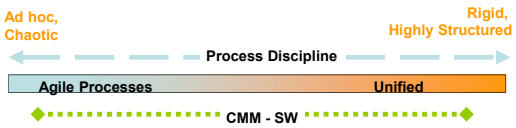
Agility and the Cost of Change



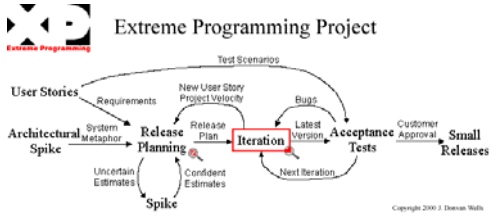
Planning

- No plan of battle ever survives contact with the enemy.
  -Heinz Guderian
- In preparing for battle I have always found that plans are useless, but planning is indispensable.
  -Dwight D. Eisenhower
- Plans are nothing; planning is everything.
  -Dwight D. Eisenhower

Agile

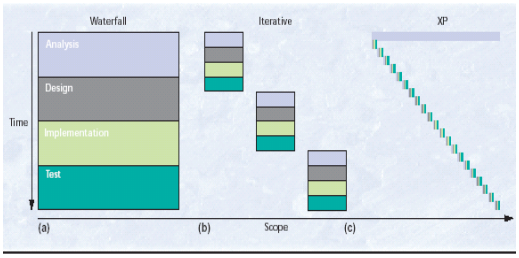


Extreme Programming (XP)



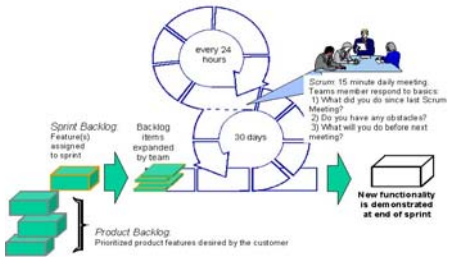
http://www.extremeprogramming.org

Extreme Programming (XP)



The reference for this has been lost but is not mine

An Overview of Scrum



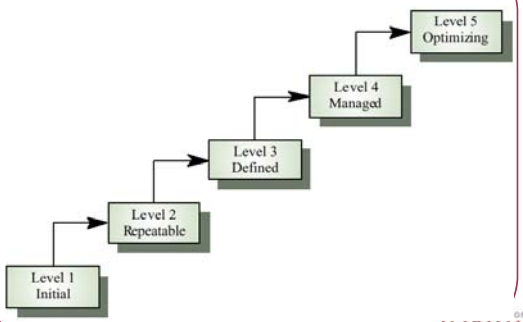
http://www.controlchaos.com/

Still Other Process Models

- **Sync-and-Stabilize - Microsoft Model**
- Requirements analysis — interview potential customers
- Draw up specifications
- Divide project into 3 or 4 builds
- Each build is carried out by small teams working in parallel
- At the end of the day — synchronize (test and debug)
- At the end of the build — stabilize (freeze the build)
- Components always work together
  - Get early insights into the operation of the product

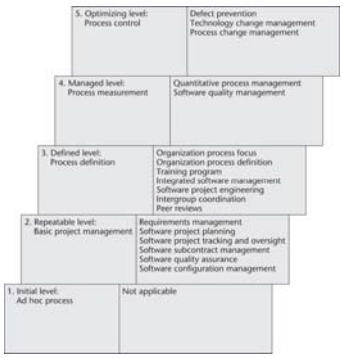
Life-Cycle Model	Strengths	Weaknesses
Evolution-tree model (Section 2.2)	Closely models real-world software production Equivalent to the iterative-and-incremental model	
Iterative-and-incremental life-cycle model (Section 2.5)	Closely models real-world software production Underlies the Unified Process	
Code-and-fix life-cycle model (Section 2.9.1)	Fine for short programs that require no maintenance	Totally unsatisfactory for nontrivial programs
Waterfall life-cycle model (Section 2.9.2)	Disciplined approach Document driven	Delivered product may not meet client's needs
Rapid-prototyping life-cycle model (Section 2.9.3)	Ensures that the delivered product meets the client's needs	Not yet proven beyond all doubt
Open-source life-cycle model (Section 2.9.4)	Has worked extremely well in a small number of instances	Limited applicability
Agile processes (Section 2.9.5)	Work well when the client's requirements are vague	Usually does not work Appear to work on only small-scale projects
Synchronize-and-stabilize life-cycle model (Section 2.9.6)	Future users' needs are met Ensures that components can be successfully integrated	Has not been widely used other than at Microsoft
Spiral life-cycle model (Section 2.9.7)	Risk driven	Can be used for only large-scale, in-house products Developers have to be competent in risk analysis and risk resolution

Software Process Maturity



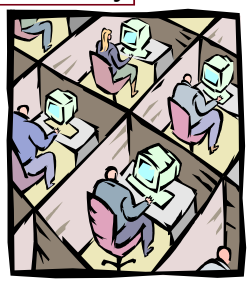
Maturity Model Levels

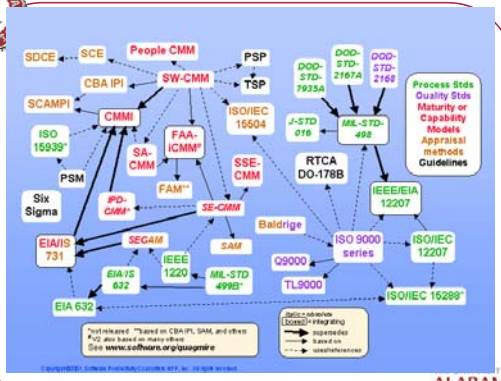
- Initial
  - Essentially uncontrolled
- Repeatable
  - Product management procedures defined and used
- Defined
  - Process management procedures and strategies defined and used
- Managed
  - Quality management strategies defined and used
- Optimizing
  - Process improvement strategies defined and used



Focus on Quality

- Begins with the Individual
  - Awareness
  - Best Practices
  - Commitment
  - Bottom-up and Top-down
- Personal Software Process
- Team Software Process





The Primary Goal of Any Software Process: *High Quality*

What is Quality?

- Quality – How well does a product meet its specification?

- This is problematical for software systems
  - There is a tension between customer quality requirements (efficiency, reliability, etc.) and developer quality requirements (maintainability, reusability, etc.);
  - Some quality requirements are difficult to specify in an unambiguous way;
  - Software specifications are usually incomplete and often inconsistent.