The Software Lifecycle

15-413: Introduction to Software Engineering

Jonathan Aldrich

Software Development Activities

- *Student Comments*
  - *Define the problem – requirements*
  - *Estimate size of task, how long it will take to complete*
  - *Provide initial support/teach people to support the project*
  - *Teach people how to use the products*

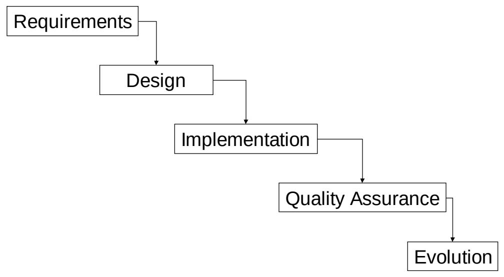
29 August 2005

Software Development Activities

- Gathering Requirements
- Team Management
- Software Design
- Coding
- Testing
- Documentation
- Software Maintenance

29 August 2005

Waterfall Model of S/W Dev.



29 August 2005

Requirements

- Determining what clients need from software
  - Problem space, not solution space
  - May include quality attributes
    - Performance, security, maintainability…
- Challenges
  - Clients don't know what they want
  - Clients can't express what they want
  - Bound to change
    - Better communication
    - Better client
    - Changes to environment

29 August 2005

Design

- Engineering solution that addresses requirements
- Designs include
  - Architecture
  - Code interfaces
  - User interfaces
  - Components
  - Data structures
  - Algorithms

29 August 2005

Implementation

- Realizing a design in code
- More than just coding
  - Documentation
  - Assertions/Invariants
  - Coding standards
  - Pair programming
  - Tools
  - Configuration management

Quality Assurance

- Ensuring the implementation meets quality standards
- Testing
  - Unit
  - Functional
  - Regression
- Analysis
- Design and code reviews

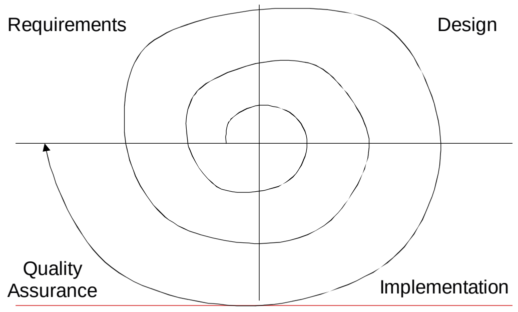
Evolution

- Changing the software to fix defects meet new requirements
- Most development today is really evolution
- Differs from initial development
  - Significant investment in existing code
  - Have to work within additional constraints
  - Many SE techniques focus on making evolution easier

Problems with Waterfall

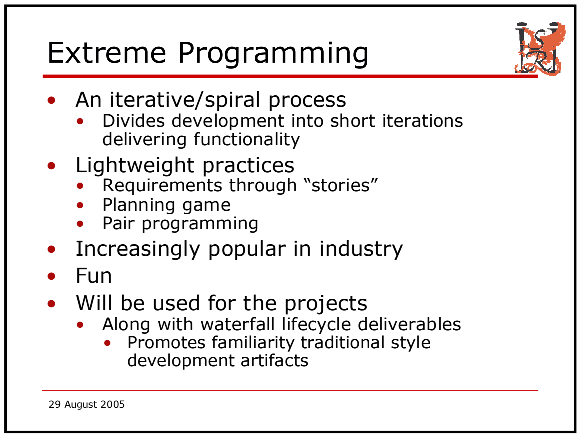
- Change is ubiquitous
  - Occurs even *during* software development
- Waterfall assumes one stage completes before others begin
  - Unrealistic in most environments
    - Requirements constantly changing
    - Lessons learned in later stages affect earlier ones
- Useful applied where communication costs high
  - Stable requirements
  - Very large software systems
  - Distributed teams

Spiral Model of S/W Dev.



Benefits of Spiral Development

- Delivers initial value early
  - Mitigates risk of failure
  - Focus on high-priority functionality
- Frequent requirements refinement
  - Uses feedback from one iteration to refine requirements for the next
  - Mitigates impact of change
- Note: the Spiral model is driven by uncertainty and change
  - A theme of the whole course



Extreme Programming

- An iterative/spiral process
  - Divides development into short iterations delivering functionality
- Lightweight practices
  - Requirements through “stories”
  - Planning game
  - Pair programming
- Increasingly popular in industry
- Fun
- Will be used for the projects
  - Along with waterfall lifecycle deliverables
    - Promotes familiarity traditional style development artifacts

29 August 2005