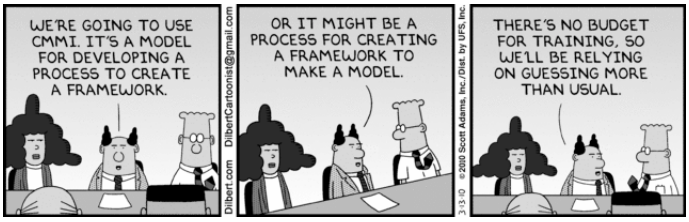
CIS 422/522

Software Life cycles and Process Models

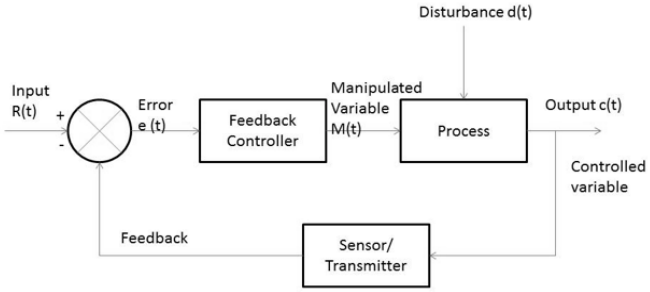


View of SE in this Course

- The purpose of software engineering is to *gain and maintain* intellectual and managerial control over the *products* and *processes* of software development.
- Intellectual control implies
  - We understand the developmental goals
  - Can distinguish good choices from bad
  - We can effectively build to meet our goals
    - Behavioral requirements (functionality)
    - Software Qualities (reliability, security, maintainability, etc.)
- Managerial control implies
  - We make accurate recourse estimates
  - We deliver on schedule and within budget

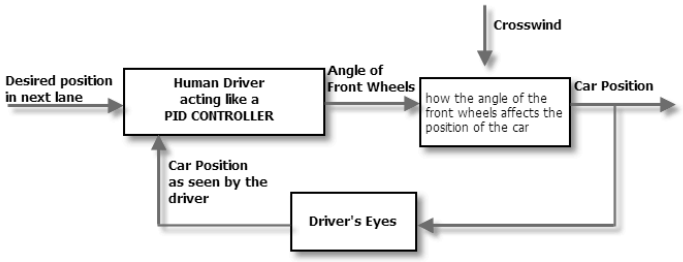
Control Realities

- Reality Check:
  - Cannot fully predict consequences of our choices
  - Control is never absolute
- Implication: maintaining control is an active process (view as a feedback-control loop)



3

Active Control



- Control in a software development means
  - Understand where we want to be (ideal)
  - Evaluate current delta
  - Make adjustments

Control and Risk

- Risk: a *risk* is defined as a condition that can lead to a loss of control
  - Incorrect, misunderstood, or missing requirements
  - Poor design choices
  - Differing assumptions by developers
  - Inadequate testing, validation, etc.
- Can lead to delivering wrong product, late, over cost..
- Assessing and mitigating risk is a critical SE activity
- Assertion: well defined processes help organize work and control risks

Need to Organize the Work

- Nature of a software project
  - Software development produces a set of interlocking, interdependent work products
    - E.g. Requirements -> Design -> Code -> Test
  - Implies dependencies between tasks
  - Implies dependencies between people
- Must organize the work such that:
  - Every task gets done
  - Tasks get done in the right order
  - Tasks are done by the right people
  - The required qualities are built in
  - Steps are done on schedule to meet delivery

Addressed by Software Processes

- Developed as a conceptual tool for organizing complex software developments
- Answers the “who”, “what”, “when”, etc. questions
  - What product should we work on next?
  - What kind of person should do the work?
  - What information is needed to do the work?
  - When is the work finished?
- Intended use (idealized)
  1. *Model* of development (what does or should occur)
  2. *Guide* to developers in what to produce and when to produce it

Definitions

- *Software Life Cycle*: evolution of a software development effort from concept to retirement
- *Software Process Model*: Abstract representation of a software life cycle as a set of
  - *Activities*: tasks to be performed (how)
  - *Artifacts*: work products produced (what)
  - *Roles*: skills needed (who)
  - and the relationships between them
- *Software Process*: institutionalized version of a software model defining specific roles, activities, and artifacts

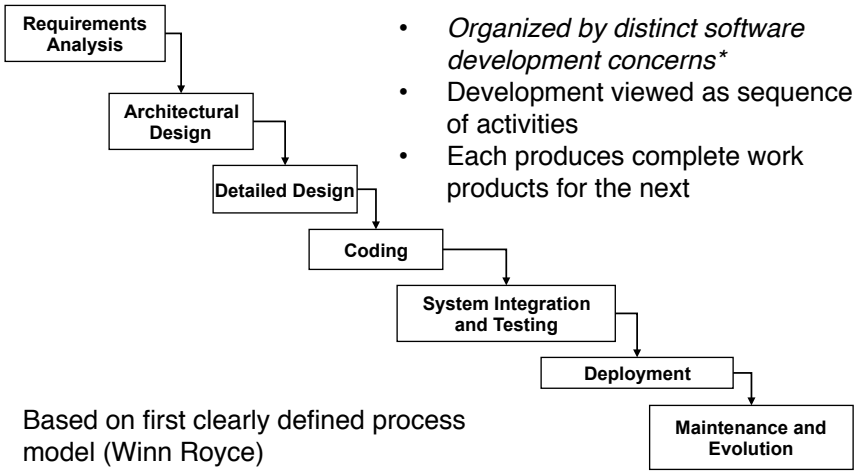
Examples of Use

- Software life-cycle: in choosing whether to build or buy, companies should consider the entire life-cycle cost of software
- Software process model: many companies are currently adapting some form of agile model of development
- Software process: organizations often standardize their software process across developments

Common Process Models

Waterfall
Prototyping
Iterative
Spiral
Agile

A "Waterfall" Model



- *Organized by distinct software development concerns**
- Development viewed as sequence of activities
- Each produces complete work products for the next

Based on first clearly defined process model (Winn Royce)

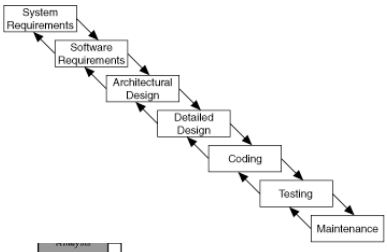
Activities, Artifacts & Roles

- Requirements Analysis
  - Activities: understand and define what the software must do and any properties it must have
  - Artifacts: Software Requirements Specification (SRS)
  - Roles: Requirements Analyst
- Architectural Design
  - Activities: decompose the problem into components that together satisfy the requirements
  - Artifacts: architectural design specification, interface specs.
  - Roles: Software Architect
- Detail Design
  - Activities: internal design of components (e.g., objects) defining algorithms and data structures supporting the interfaces
  - Artifacts: design documentation, code documentation
  - Roles: Coder

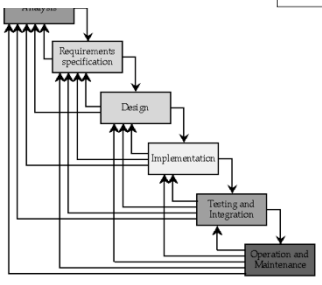
Activities, Artifacts & Roles

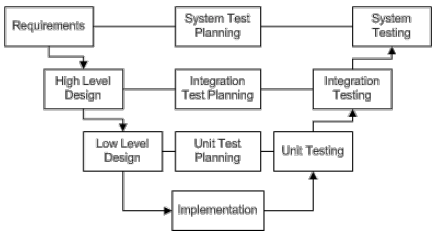
- Implementation
  - Activities: realization of the design in executable form
  - Artifacts: code, makefiles, etc.
  - Roles: Coder
- Integration and Testing
  - Activities: validation and verification of the implementation against requirements and design
  - Artifacts: test plan, test cases
  - Roles: tester, user (customer)
- Maintenance (really multiple distinct activities)
  - Activities: repair errors or update deployed system
  - Artifacts: bug fixes, patches, new versions
  - Roles: Architect, Coder, Tester

Waterfall Model Variations



There have been many variations

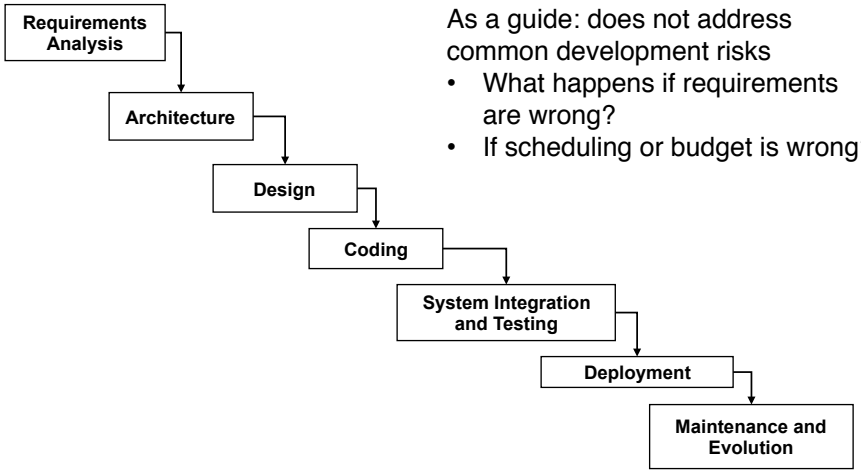




Issues with the Waterfall Model

- Variations created to address perceived shortcomings
- Model implies that you should complete each stage before moving on to the next
  - Implies that you can get the requirements right up front: does not account for inevitable changes
  - Implies testing and validation occur only when development is finished
    - Customers does not see the product until the end
  - Implies that once the product is finished, everything else is maintenance

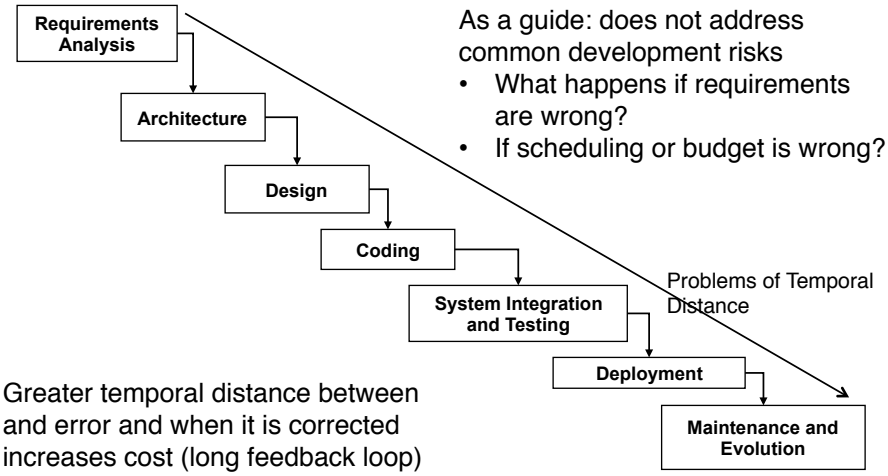
A "Waterfall" Model



As a guide: does not address common development risks

- What happens if requirements are wrong?
- If scheduling or budget is wrong?

A "Waterfall" Model*



As a guide: does not address common development risks

- What happens if requirements are wrong?
- If scheduling or budget is wrong?

Problems of Temporal Distance

Greater temporal distance between and error and when it is corrected increases cost (long feedback loop)

Take-away

- A process definition defines a model for organizing development work
- A process model should define
  - Activities (Tasks)
  - Artifacts (Products)
  - Roles (Skill sets)
- Delay (temporal distance) between when an error occurs and when it is fixed raises costs

Project Preparation

Worksite
Teams

Team Assignments

Team 1
Brewster, Carolyn
Fredericks, Meg
Gao, Andrew
Holstege, Davis
Zhu, Yueqi

Team 2
Heinrich, David
Hill, Jayd
Lan, Logan
Plunkett, Seth
Collier, Ryan

Team 3
Chavarria, Lucas
Guo, Ron
Totten, Craig
Vikupitz, Cole
Xu, James

Team 4
Caluya, Elijah
Clark, Jasmine
Harris, Mike
Liang, Kenneth
Qiu, Debin

Team 5
Brodnax, John
Brogan, Riley
Heng, Brenda
Stidhem, Will
Wang, George

Team 6
Cordes, Sam
Ginsberg, Navarre
Andreason, Trace
Jia, Ben

Team 7
Ebert, Cody
Hebb, Fiona
Schlechter, Natalie
Urban, Jake

Assignment

- First meeting (in class)
  - Exchange contact information
  - Give me a primary point of contact (email)
  - Plan one project meeting out of class (preferably by Friday)
- Project meeting
  - Discuss relevant experiences and skills
  - Look at examples of the deliverables (pointers on Schedule page)
  - Choose people for roles (primary and backup)
  - Choose a team name, logo and put on Assembla page

Questions?