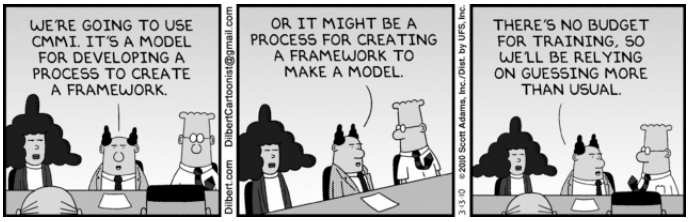
CIS 422/522

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" means that we are able make rational choices based on an understanding of the downstream effects of those choices (e.g., on system properties).
  - Managerial control similarly means we are able to make rational choices about development *resources* (budget, schedule, personnel).
- Memorize this!

Both are necessary for success!

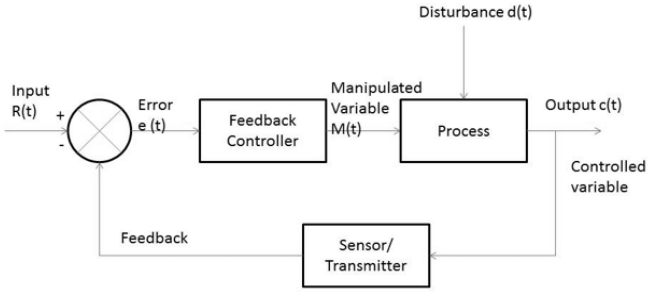
- Intellectual control implies
  - We understand what we are trying to achieve
  - Can distinguish good choices from bad
  - We can reliably and predictably build to our goals
    - Functional behavior
    - Software Qualities (reliability, security, usability, etc.)
- Managerial control implies
  - We make accurate estimations
  - We deliver on schedule and within budget

Project 1 focus

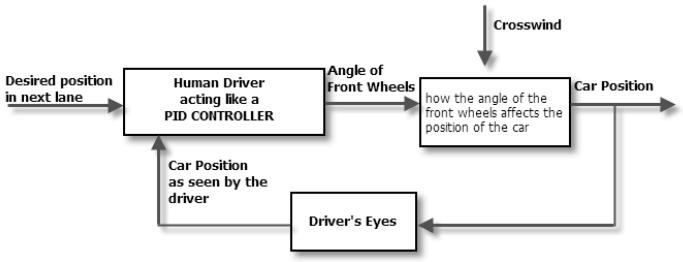
- Assertion: managerial control is not really possible without intellectual control

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)



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 (e.g. "during the design process the software architect creates the architectural speciation document")
- *Software Process*: institutionalized version of a software model defining specific roles, activities, and artifacts

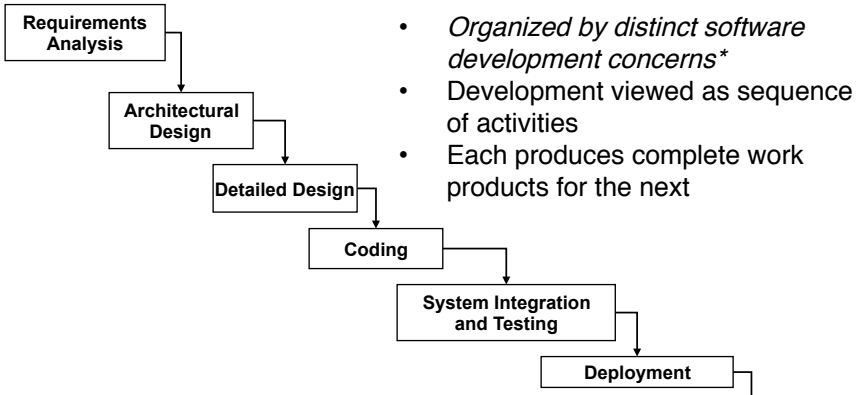
Examples of Use

- 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)

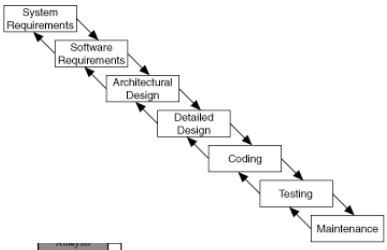
Activities, Artifacts & Roles

- 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

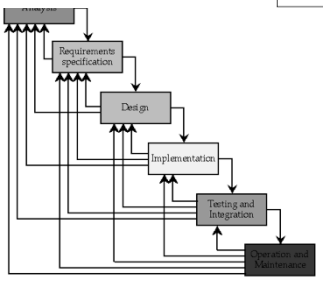
Activities, Artifacts & Roles

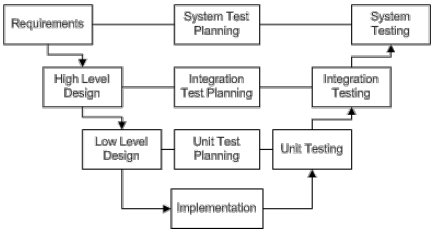
- Implementation
  - Activities: realization of the design in executable form
  - Artifacts: code, makefiles, etc.
  - Roles: Coder
- Integration and Testing
  - Activities: validation and verification of the implementation against requirements and design
  - Artifacts: test plan, test cases
  - Roles: 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

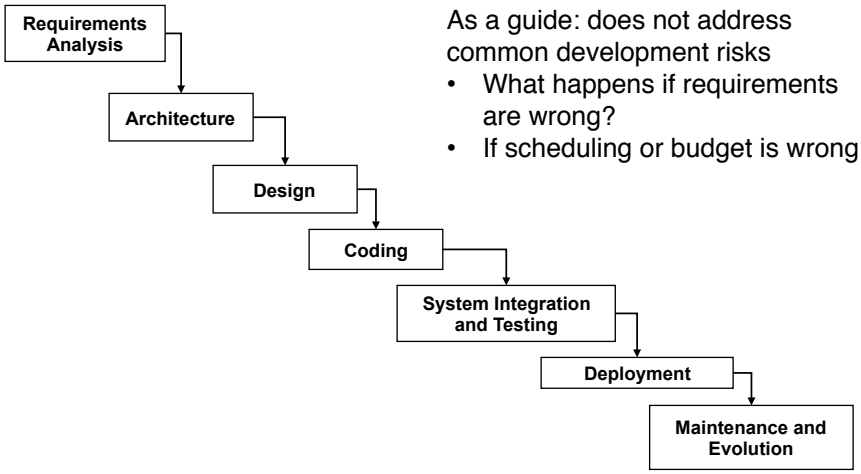




Issues with the Waterfall Model

- Variations created to address perceived shortcomings
- Model implies that you should complete each stage before moving on to the next
  - Implies that you can obtain the correct requirements 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

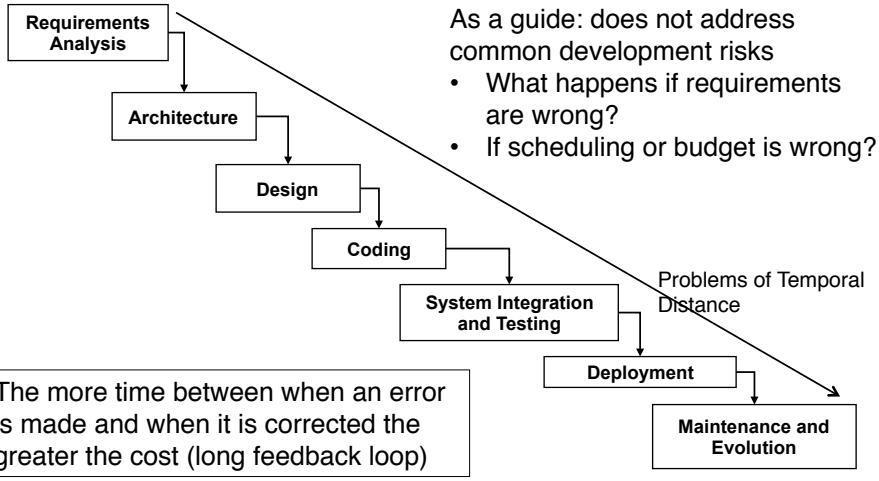
A “Waterfall” Model



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?

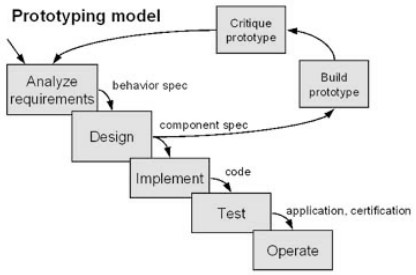
The more time between when an error is made and when it is corrected the greater the cost (long feedback loop)

Common Process Models

Waterfall
Prototyping
Iterative
Spiral
Agile

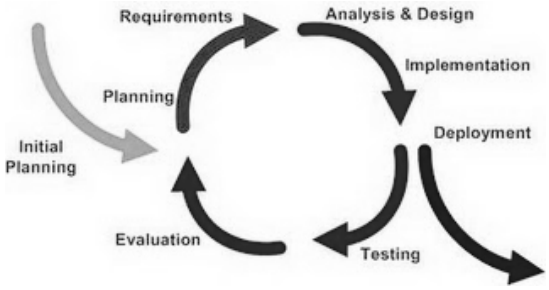
Characteristic Model: Prototyping

- Waterfall variation
- First system versions are prototypes, either:
  - Interface
  - Functional
- Which waterfall risks does this try to address?



Characteristic Processes: The Iterative Model

- Process is viewed as a sequence of iterations
  - Essentially, a *series of waterfalls*
  - Each iteration builds on the previous one (e.g., adds requirements, design components, code features, tests)
  - Each iteration produces complete set of work products deliverable software
  - Customers provide feedback on each release
  - There is no "maintenance" phase – each version includes problem fixes as well as new features



21

Iterative Model

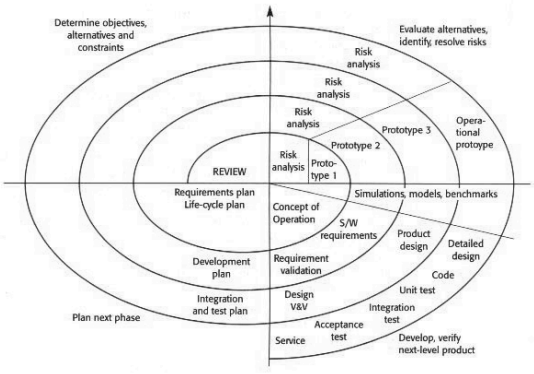
- Also called "incremental development"
- Addresses some common waterfall risks
  - Risk that software cannot be completed – build incremental subsets
  - Risk of building the wrong system – stakeholder have opportunities to see the software each increment
  - Each iteration provides checkpoint for feasibility, schedule, budget and others issues

Advantages of Incremental Development

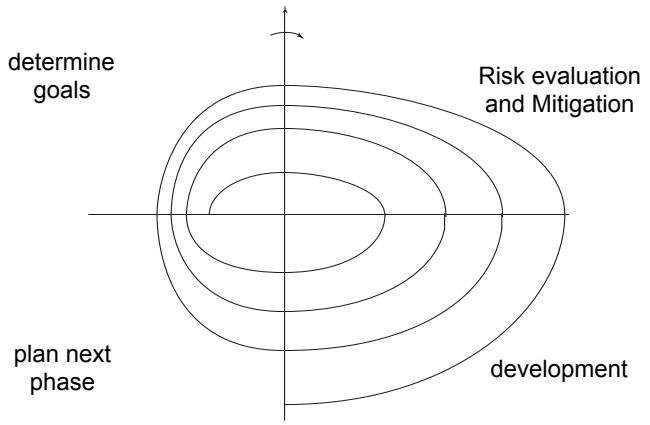
- Customers get usable functionality earlier than with waterfall
- Early feedback improves likelihood of producing a product that satisfies customers
  - Reduces market risk: if customers hate the product, find out before investing too much effort and money
- The quality of the final product is better
  - The core functionality is developed early and tested multiple times
  - Only a relatively small subset of functionality added in each release: easier to get it right and test it thoroughly
  - Detect design problems early and get a chance to redesign

Characteristic Processes: The Spiral Model

- Process viewed as repeating cycles of increasing scale
- Identify risks and determine (next set of) requirements
- Each cycle builds next version by extension, increasing scale each time



Spiral Model

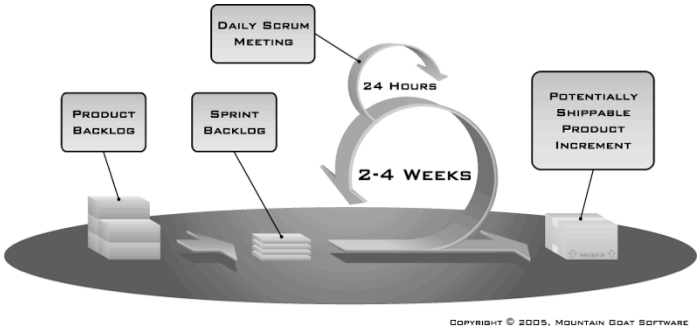


Spiral Model Characteristics

- Response lack of explicit risk analysis and risk mitigation in “waterfall” process
- Includes risk analysis and mitigation activities at each phase (e.g., prototyping)
- Explicit Go/No-Go decision points in process
- Heavy-weight process: substantial overhead not contributing directly to end products

Characteristic Processes: Agile (e.g. scrum)

- Process viewed as nested sequence of builds (sprints)
  - Each build adds very small feature set (one or two)
  - Nightly build/test, frequent customer validation
  - Focus on delivering code, little or no time spent on documentation



How do we Choose a Development Process?

E.g., for your projects

Objectives

- Goal: proceed as rationally and systematically as possible (I.e., in a controlled manner) from a statement of goals to a design that demonstrably meets those goals within design and management constraints
  - Understand that any process description is an abstraction
  - Always must compensate for deviation from the ideal (e.g., by iteration)
  - Still important to have a well-defined process to follow and measure against

A Software Engineering Perspective

- Question of control vs. cost: processes introduce *overhead*
- Choose process to provide an appropriate level of control for the given product and context
  - Sufficient control to mitigate risks, achieve results
  - No more than necessary to contain cost and effort
- Provides a basis for choosing or evaluating processes, methods, etc.
  - Does it achieve our objectives at reasonable cost?
  - Does it address the most important developmental risks?
- Need to agree on kind of control you need and how you will accomplish it

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

Assignment

- Forward your emails from xxx@uoregon.edu
- 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)
- Team meeting objectives
  - Look at examples of the deliverables (pointers on Schedule page)
  - Discuss relevant experiences and skills
  - Choose people for roles (primary and backup)
  - Choose a team name, logo and put on Assembla page

Team Assignments

Team1	Team2	Team3	Team4
Gerendasy, Sam	Elliott, Sam	Chen, Adam	Ericson, Sean
Ma, Sherry	Kilmer, Clayton	Etzel, Shohei	Friedrich, Chris
Merrill, Brett	Miller, Ry	Knees, Jeff	Greenlaw, Sarah
Ronlov, Logan	Schaefer, Kaela	Lopez Raya, Erik	Rozenboim, Matt
Zhang, Yehui	Wang, Freddie	Sov, Brandon	Chen, Bill
			Wei, Fangzheng

Team5	Team6	Team7
Cannon, Madeline	Chavarria, Lucas	Case, Ryan
Hou, Guangyun	Leonard, Jesse	Lewis, Jordan
Jensen, David	Li, Bin	Malynur, Anisha
Kent, Steve	Liu, Angie	Pugh, David
Yu, Andy	Poole, Logan	Wu, Emily

Questions?