[CS3704] Intermediate Software Design and Engineering

Dr. Chris Brown

Virginia Tech

8/28/2023

Software Process I

What is a software process?
SE Process Framework
Examples of SE Frameworks

Announcements

- HW1 due next Friday (9/8) at 11:59pm
- Ut Prosim activity for research study
  - See details on Slack
  - More coming soon
- Reminder: Pay attention to instructions for Workshop activities!
  - Repository should have been named “Basics”
  - Name should be in README file

Syllabus Review Questions so far…

- Will we be able to choose our groups for the presentations? Sign-up will be first come, first served
- Late submissions without a valid excuse will receive a -25% deduction. Is this per day or a flat 25%? Per day, have 4 days to submit (0%)
- How will the exam be formatted? Will it be multiple choice? True or false? Essay? TBD
- Is there already a determined date for our final exam? Dec 11 3:25-5:25, Dec 4 and Dec 6 in class

Syllabus Review Questions so far…

- Will we be able to select at least one of our partners for the project? Yes
- How much coding, if any, will there be? Not too much
- Add deadlines for assignments to schedule Added

Learning Outcomes

By the end of the course, students should be able to:

- **Understand software engineering processes, methods, and tools used in the software development life cycle (SDLC)**
- **Use techniques and processes to create and analyze requirements for an application**
- **Use techniques and processes to design a software system**
- **Identify processes, methods, and tools related to phases of the SDLC**
- **Explain the differences between software engineering processes**
- **Discuss research questions and current topics related to software engineering**
- **Create and communicate about the requirements and design of a software application**

What is SE?

A discipline that encompasses:

- the *process* of software development
- *methods* for software analysis, design, construction, testing, and maintenance
- *tools* that support the process and the methods

[Pressman]

What is a SE process?

– a framework for the tasks that are required to build high-quality software to provide stability, control and organization to an otherwise chaotic activity

[Pressman]

– a software development process defines ***who*** does ***what***, ***when***, in order to build a piece of software.

[Wilson]

Warm-Up

Discuss: What is your experience with software engineering processes in a development team (i.e. class project or internship) setting?

Quiz!

- Not graded on correctness
- ~5 minutes to complete
- https://bit.ly/cs3704_828quiz

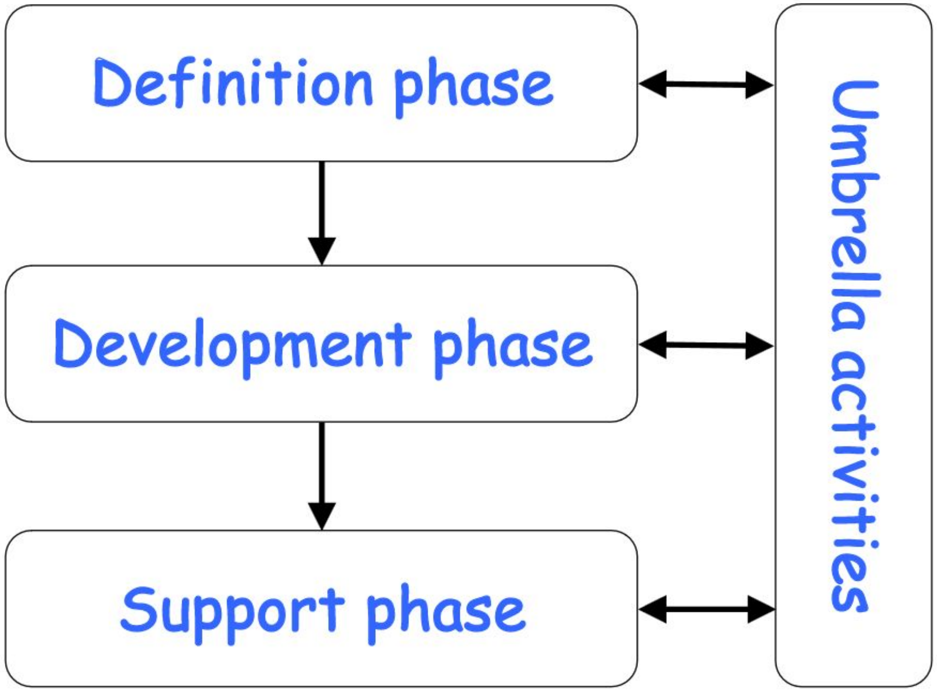
What does SE Process mean…

- For a single programmer?
  - Planning (time, resources, assignments…)
  - Design and development
  - Tracking and measuring progress
- For a development team?
  - Organizational planning (time, resources, etc.)
  - Hiring, training, tool acquisition, etc.
  - Process assessment and improvement
- For software development in general?
  - Help organize “best practices” for SE

Elements of SE Processes

Term	Examples
People	Software developers, managers, customers, etc.
Tasks	i.e. analyze requirements
Work products	i.e. requirements specification
Planning	Estimate needed resources, time, defects
Conducting	Track progress and work results
Assessing	Define measurement metrics for quality, progress, etc.

Generic View of SE Process



Definition Phase

- Tasks related to **problem definition**
  - requirements, constraints, environment, etc.
- **Step 1:** System engineering
  - Ascertain roles of hardware, software, people, databases, etc.
- **Step 2:** Analysis of the problem
  - Requirement analysis
    - Understanding what the users need and want
  - Domain analysis
    - Illustrate key concepts in a set of SW systems (reuse)
- **Step 3:** Project planning
  - Resources (e.g., people), cost, schedule

Development Phase

- Tasks related to **problem solution**

– architecture, programming, testing, etc.

- **Step 1:** software design (the blueprint)

– Design models that describe structure, interactions, etc.

- **Step 2:** code generation/implementation
- **Step 3:** software testing

– *Goal:* uncover as many errors as possible

Support Phase

- Tasks related to **software evolution**

– Changes? Definition and development in the context of existing software

- Adaptation to change in the environment

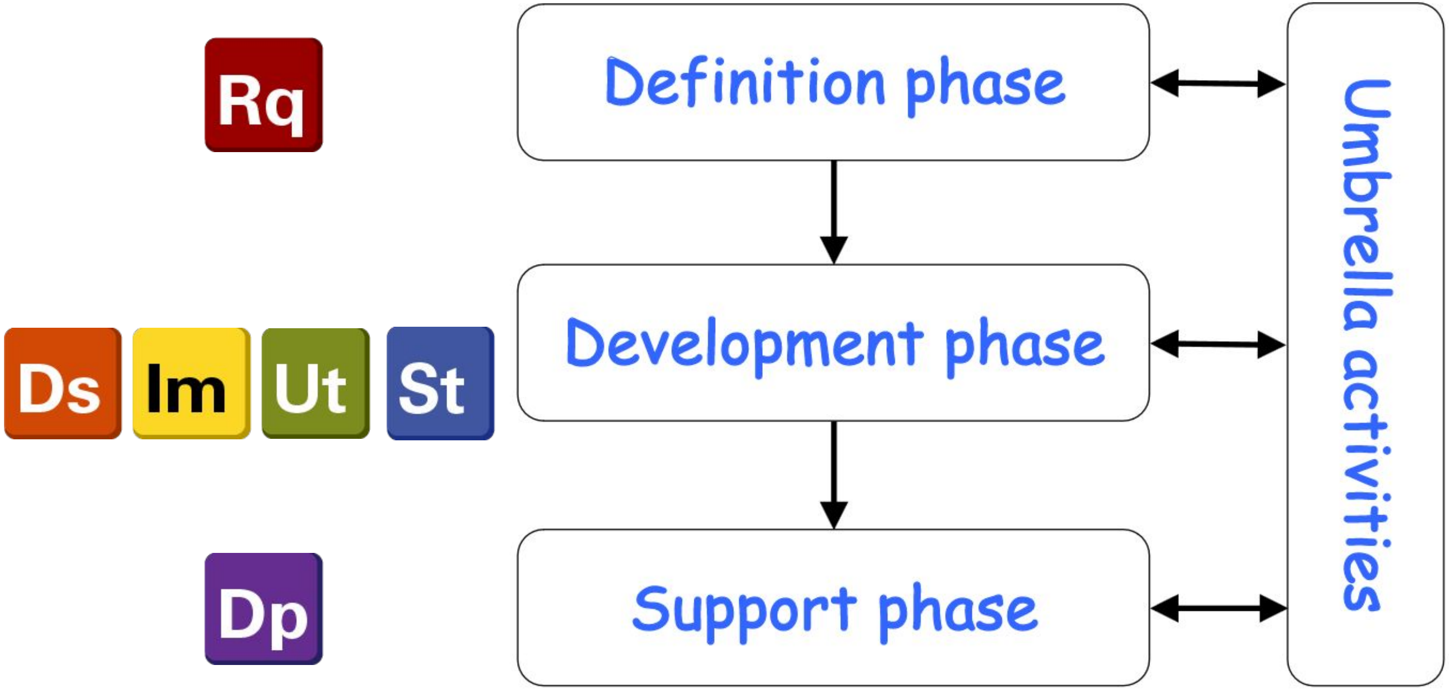
– New hardware, changes in OS, business rules, etc.

- Correction of defects (Y2K problem, $308B)
- Enhancements (new features, etc.)
- Refactoring (to ease future changes)

Some Umbrella Activities

- Project management
  - Tracking and control of people, process, cost, etc.
- Quality assurance (QA)
  - Formal technical reviews of work products
  - Software testing
  - Keeping docs consistent with code base
- Configuration management
  - Controls the changes in work products using systems like SVN, Git

Generic View of SE Process w/ SDLC



Rq

Definition phase

Ds Im Ut St

Development phase

Dp

Support phase

Umbrella activities

SE Process Observations

- SE processes are idealizations
  - The real world is a very complex place
  - Software engineering is a very complex activity
- But, they provide a useful roadmap for SE work to organize development activities

Examples of Process Models

Plan-Driven Models

1. Code-and-Fix*
2. Waterfall
3. V-model

Iterative Models

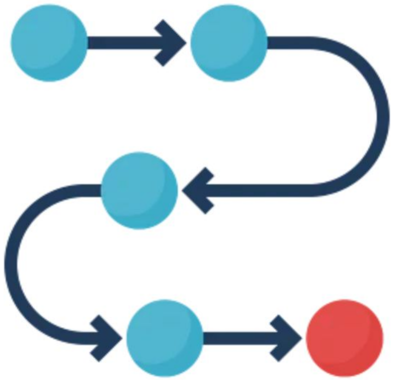
4. Incremental
5. Prototyping
6. Spiral

- *Agile (next class…)*

Plan-Driven Models

Also known as incremental development

- The process is divided into small, workable increments. Each succeeding increment builds on the work completed in the previous increment.



Plan-Driven Models (cont.)

- Traditional software process models
- Project goes through phases sequentially
  - Possible for feedback across phases (i.e. design problems can be fixed during coding)
- Project requirements are set up front and stable
- Typically few or no iterations
  - Project is “frozen” after a certain time, no changes
- *“Do it right the first time”*...

History of Software Engineering

1950s: Engineer software like hardware

- First programming languages
  - **1956:** FORTRAN (Formula Translation);
  - **1958:** LISP (List Processor);
  - **1959:** COBOL (Common Business Oriented Language);
- Punch cards
- **Software Process: Code-and-Fix**

Code-and-Fix

Based on a given a project specification:

```
1. Write code
2. Improve it
3. GOTO 1
```

When should you use Code-and-Fix?

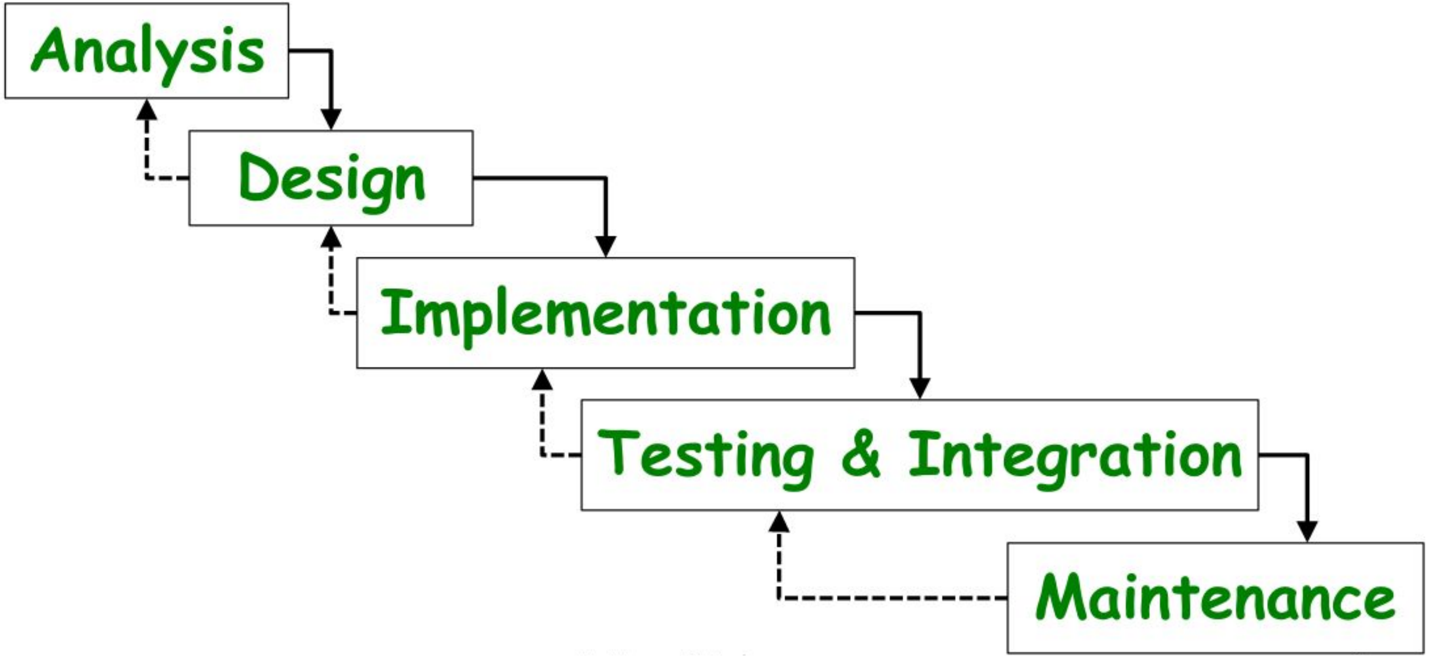
- Maybe small 1-person projects and/or course assignments
- Never 🛑

Problems with Code-and-Fix

- Poor match with user needs
- Bad overall structure – No blueprint
- Poor reliability - no systematic testing
- Maintainability? What’s that?
- What happens when a programmer quits?

Waterfall Model

The “classic” process model since the 1970s



Analysis

Design

Implementation

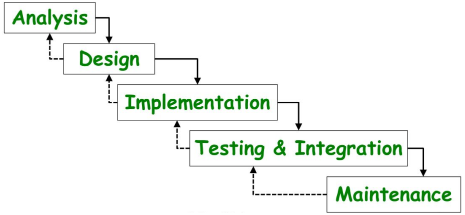
Testing & Integration

Maintenance

Waterfall Model Assumptions

- All requirements are known at the start and stable
- Risks (unknowns) can be turned into known through schedule-based invention and innovation
- The design can be done abstractly and speculatively
  - i.e., it is possible to correctly guess in advance how to make it work
- Everything will fit together when we start the integration

Waterfall Pros and Cons



Pros:

– Widely used and systematic

– Good for projects with well-defined requirements

– Development cost minimized by up front planning

– Fits needs for managers, accountants, lawyers, etc. (i.e. not developers)

Cons:

– Development is not sequential

– Assumes all requirements are known at the start

– Cannot predict risk and design

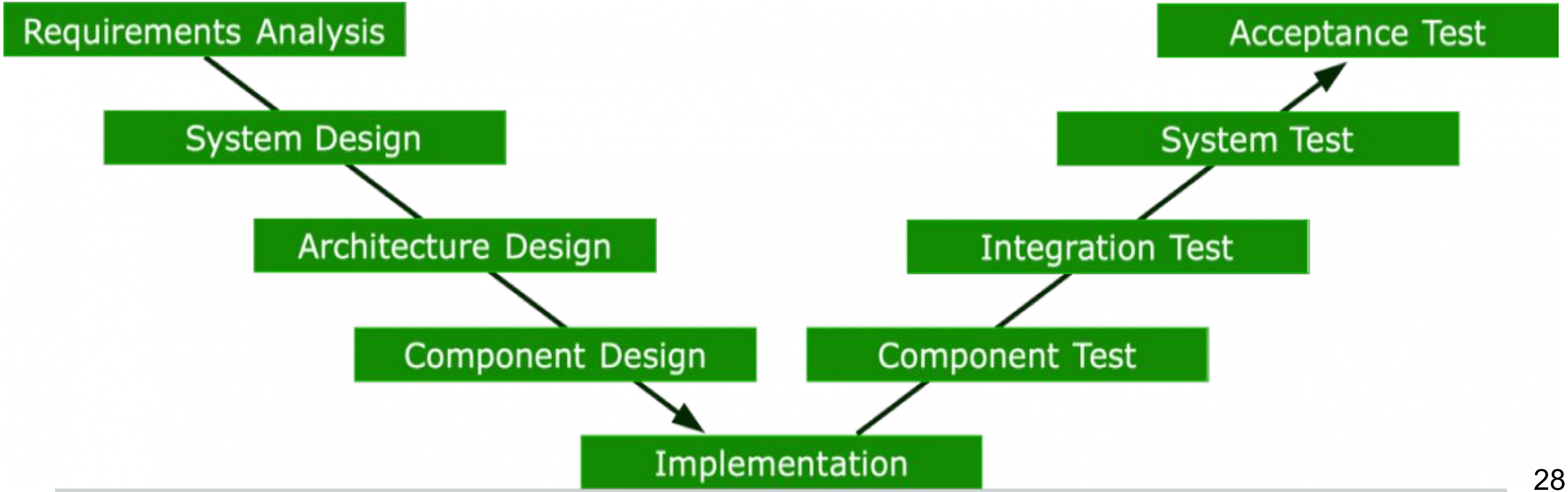
– Working programs are not available early

– Assumes everything will fit together during integration

– Expensive and time-consuming

V-Model

- Extension of the waterfall model
- Greater emphasis on testing and design!



When to use plan-driven models?

- Working for big clients that enforce formal approaches (i.e. government)
- Working on fixed-scope, fixed-price contracts without many rapid changes
- For safety- and mission-critical systems
- Work in an experienced team

Success Story: Space Shuttle

As the 120-ton space shuttle sits surrounded by almost 4 million pounds of rocket fuel, exhaling noxious fumes, visibly impatient to defy gravity, its on-board computers take command.

Charles Fishman, 1996

Success Story (cont.)

“This software is bug-free”

- Some impressive statistics

– The last 3 versions of the program--420,000 lines of code had just 1 error each

– The last 11 versions of the software had a total of 17 errors

– Commercial programs of equivalent complexity would have 5,000 errors

How did they get it right?

- 1/3 of the process before coding
- NASA and Lockheed Martin groups agree in the most minute detail about everything
- Specs are almost pseudo-code
- Nothing in the specs is changed without agreement and understanding

Ex.) Task to upgrade software to add GPS navigation

– 1.5% changes in program/6366 LOC
– 2500 page specs for the change

How much did it cost?

- 260 people
- >40,000 pages of specifications
- 20 years
- $35 million Annual budget
- $700 million overall budget
- 700 million/420k = $1600/line of code

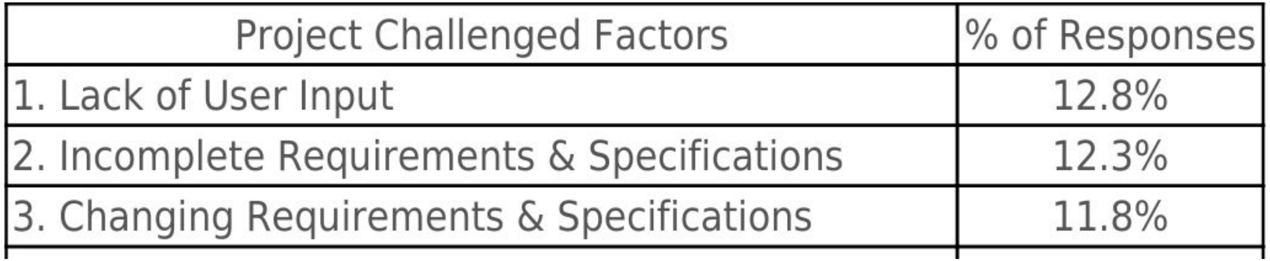
The CHAOS Report 1995

“In the United States, we spend more than $250 billion each year on IT application development…A great many of these projects will fail. Software development projects are in chaos, and we can no longer imitate the three monkeys -- hear no failures, see no failures, speak no failures. The Standish Group research shows a staggering 31.1% of projects will be canceled before they ever get completed. Further results indicate 52.7% of projects will cost 189% of their original estimates. The cost of these failures and overruns are just the tip of the proverbial iceberg.”

[Standish Group]

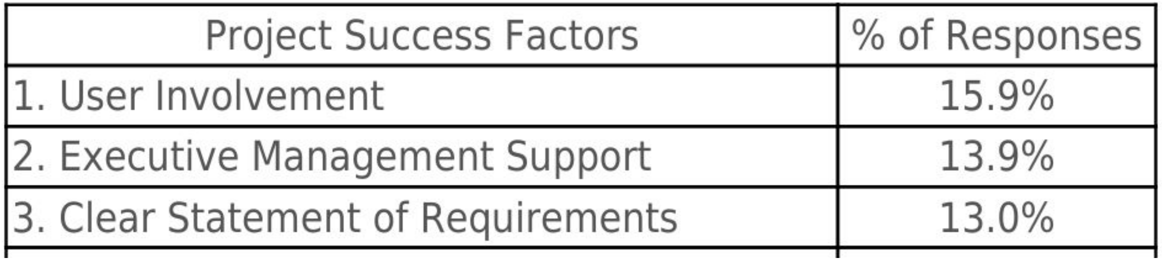
The CHAOS Report 1995 (cont.)

Top 3 reasons for project failure:



Project Challenged Factors	% of Responses
1. Lack of User Input	12.8%
2. Incomplete Requirements & Specifications	12.3%
3. Changing Requirements & Specifications	11.8%

Top 3 reasons for project success:

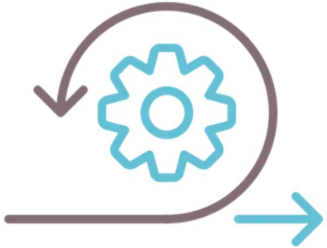


Project Success Factors	% of Responses
1. User Involvement	15.9%
2. Executive Management Support	13.9%
3. Clear Statement of Requirements	13.0%

[Standish group 1995]

Iterative Models

- This involves the development of a system that follows repeated cycles. Changes are made based on results from the most recent iteration, enabling the project to evolve over time.



Iterative Models (cont.)

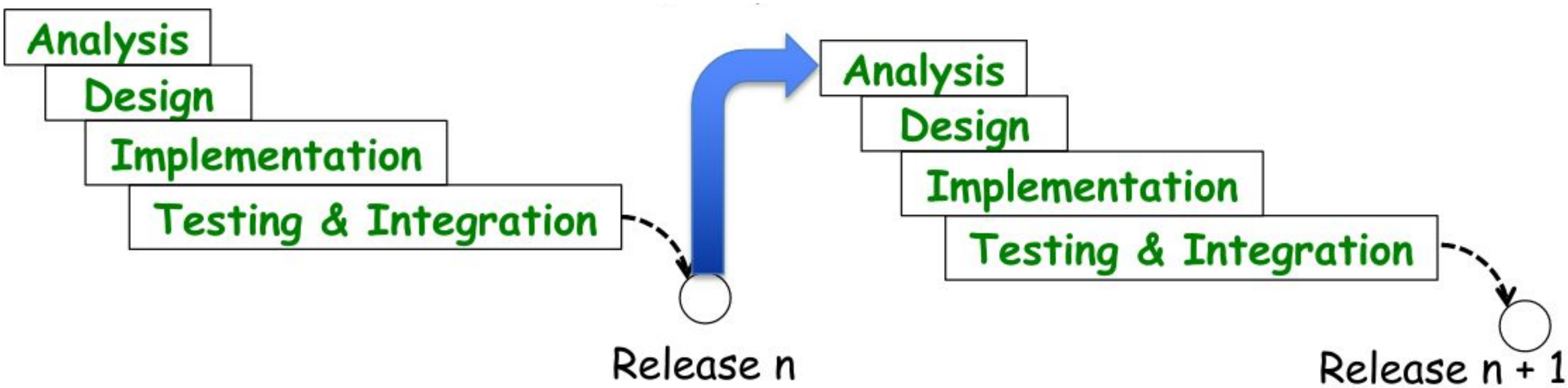
- Modern software process models
- Software is developed through repeated cycles (*iterations*).
  - Easier to modify software design, functionality, etc.
  - Faster operational product (weeks vs months)
- Usually more user involvement

Iterations

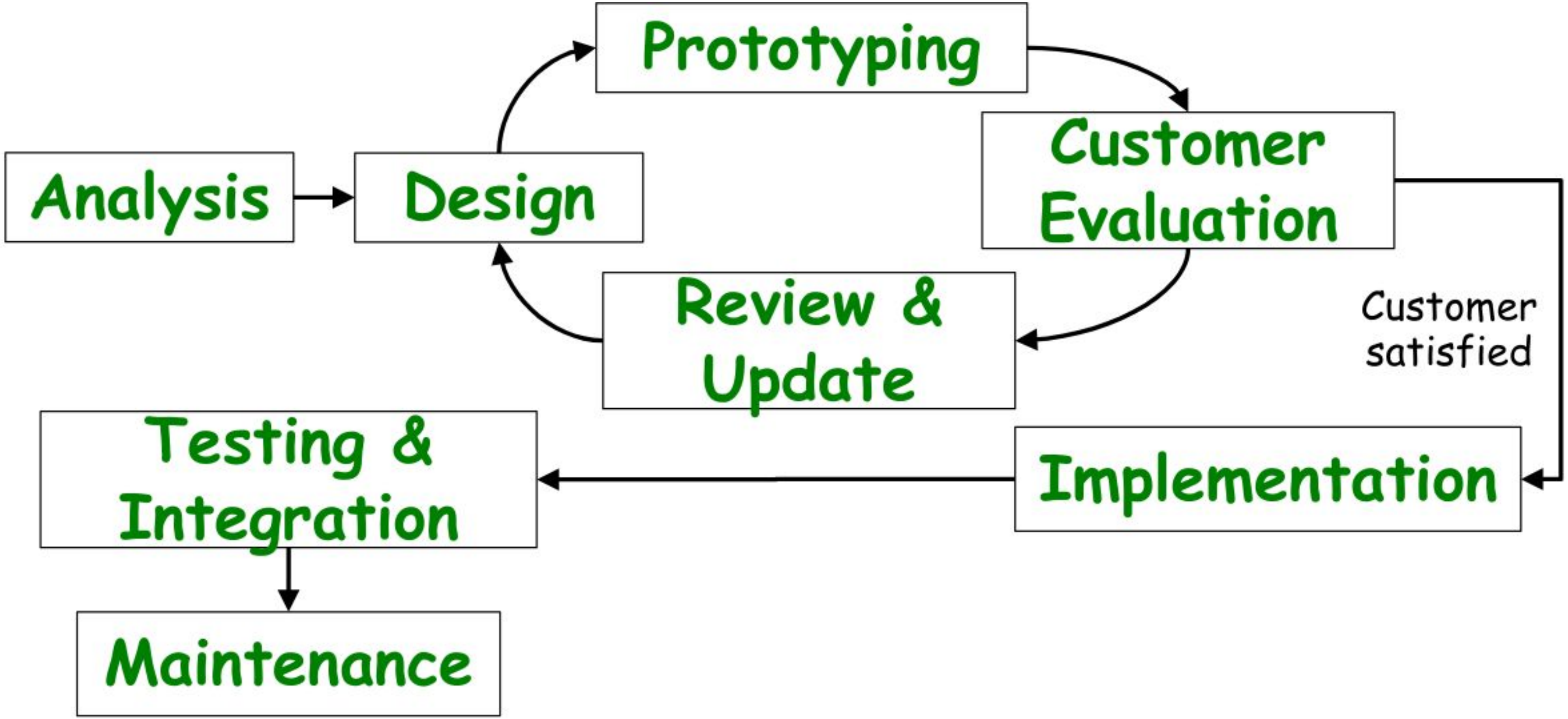
- Iterations should be short (2-6 weeks)
  - Small steps, rapid feedback and adaptation
  - Massive teams with lots of communication
- Iterations should be time-boxed (fixed length)
  - Integrate, test and deliver the system by a scheduled date
  - If not possible: move tasks to the next iteration
  - Improves programmer productivity with deadlines
  - Encourages prioritization and decisiveness

Incremental Model

- An iterative sequence of waterfall models



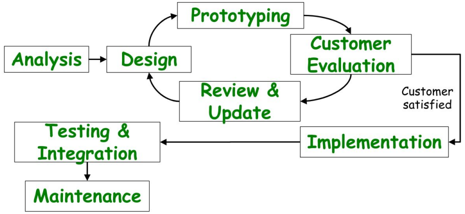
Prototyping Model



Prototyping Model

- Iteratively build a prototype when customers have ambiguous requirements.
- Can model an entire system with real data or a few screens with sample data.
- Note: Prototype is thrown away!
- Used at a variety of organizations
  - Boeing builds digital prototypes of aircrafts allowing for the detection of design conflicts
  - Disney uses storyboards to work through the process of producing feature-length films
- Online systems and web interfaces

Prototyping Pros and Cons



Pros:

– Facilitate communication about requirements

– Easy to change or discard

– Educate future customers

Cons:

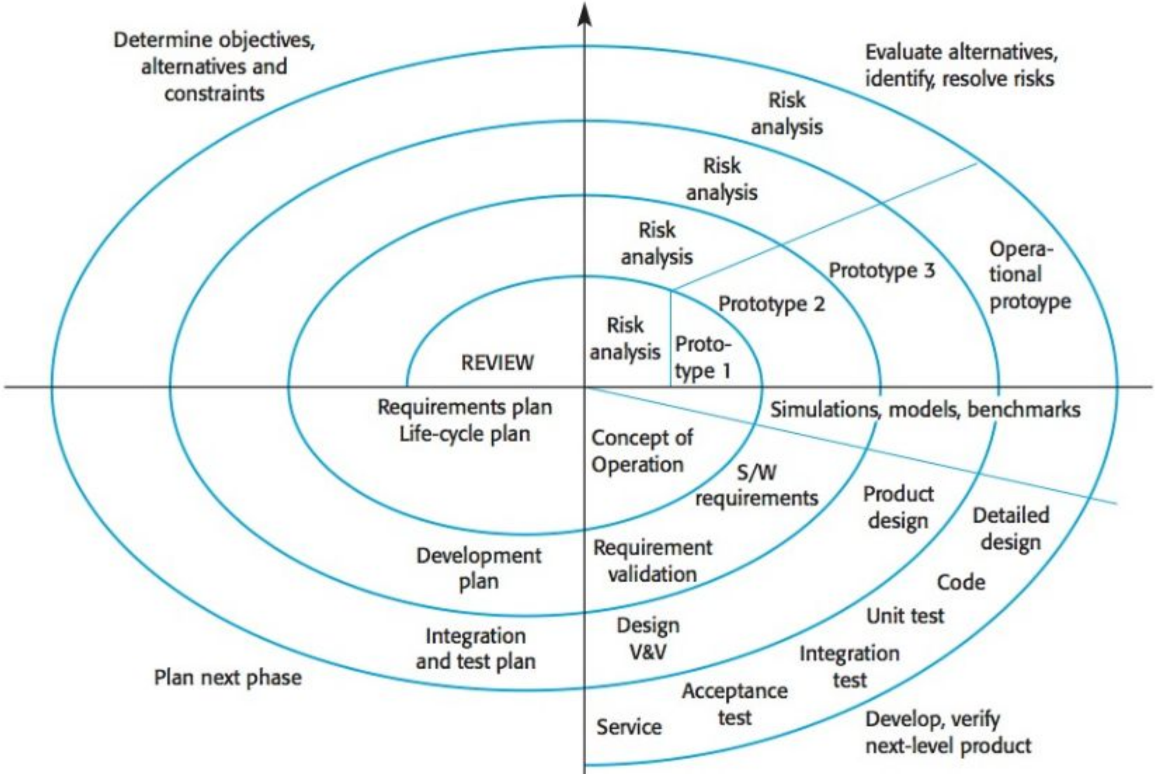
– Iterative nature makes it difficult to plan and schedule

– Excessive investment in the prototype

– Bad decisions based on prototype

• E.g., bad choice of OS or PL

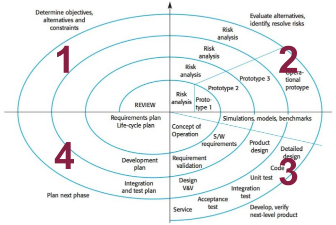
Spiral Model



Spiral Model

- A risk-driven evolutionary model that combines development models (waterfall, prototyping, etc.)
- What is risk?
  - Anything that can go wrong (people, tasks, work products, etc…)

Spiral Phases

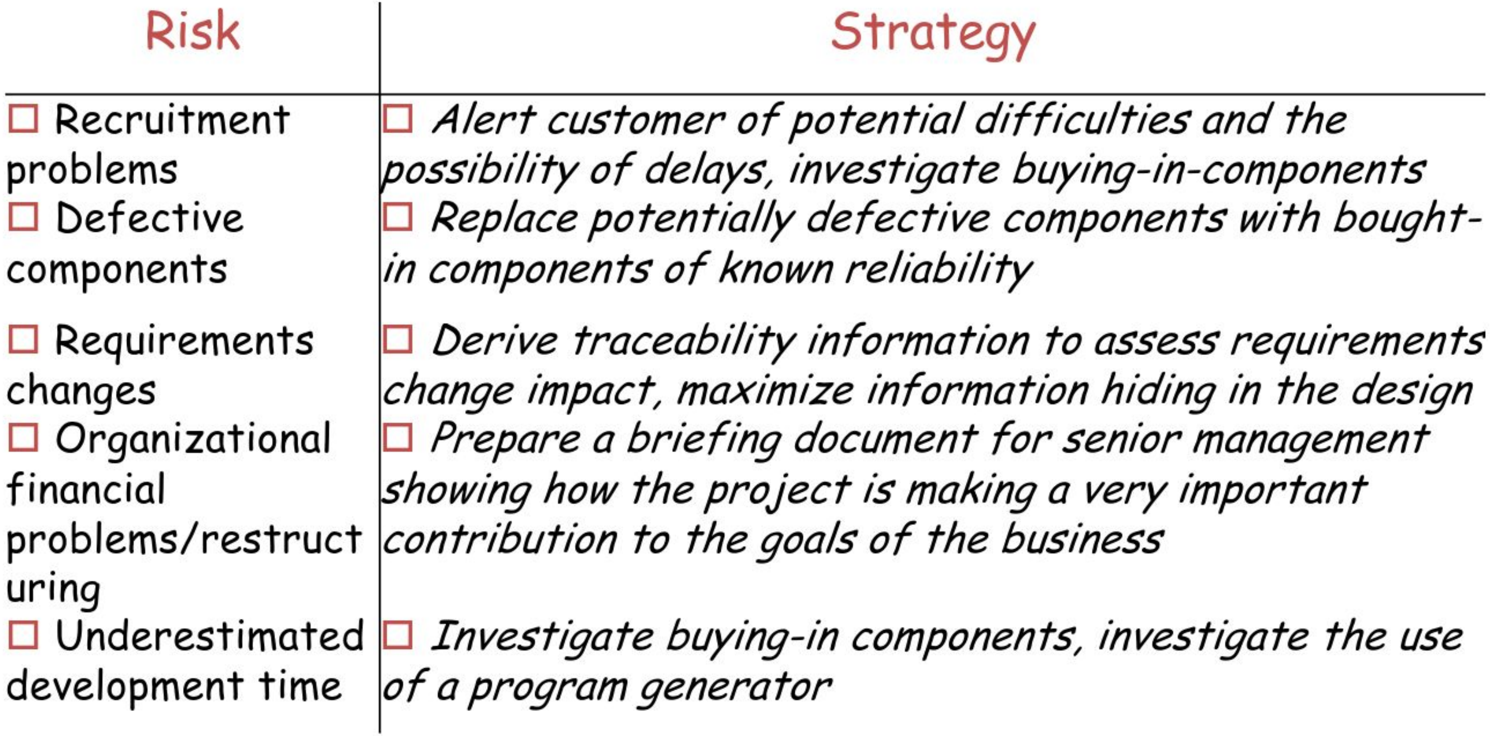


1. Objective setting
   - Define specific objectives, constraints, products, plans
   - Identify risks and alternative strategies
2. Risk assessment and reduction
   - Analyze risks and take steps to reduce risks
3. Development and validation
   - Pick development methods based on risks
4. Planning
   - Review project and decide whether to continue with another loop

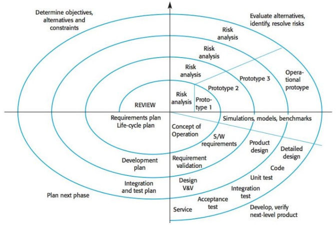
Risk Management

1. Risk identification
2. Risk analysis
    a. the probability of the risk, the effect of the risk
3. Risk planning
    a. various strategies
4. Risk monitoring

Risk Management (Sommerville)



Risk	Strategy
□ Recruitment problems	□ *Alert customer of potential difficulties and the possibility of delays, investigate buying-in-components*
□ Defective components	□ *Replace potentially defective components with bought-in components of known reliability*
□ Requirements changes	□ *Derive traceability information to assess requirements change impact, maximize information hiding in the design*
□ Organizational financial problems/restructuring	□ *Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business*
□ Underestimated development time	□ *Investigate buying-in components, investigate the use of a program generator*



Prototyping Pros and Cons

Pros:

– High amount of risk analysis to avoid/reduce risks

– Early release of software, with extra functionalities added later

– Maintain step-wise approach with “go-backs” to earlier stages

Cons:

– Requires risk-assessment expertise for success

– Expensive

Incremental vs Prototyping vs Spiral

- All are iterative models 🔁

1. Incremental: Release-driven
   - Get product out to users quickly!
2. Prototyping: Client-driven
   - Get feedback from users/clients!
3. Spiral: Risk-driven
   - Prevent and minimize risk!
   - What is risk? *Anything that can go wrong!* **Ex.)** bugs, new features, development tasks, budget, changing requirements, absences, deadlines, people,...

When to use them?

- The major requirements of the complete system are defined **[Incremental]**
- When the desired system has a lot of interactions with users **[Prototyping]**
- Large and mission-critical projects with medium to high risk **[Spiral]**
- Need to get a product to the market earlier
- When significant changes are expected

Next Class

- SE Processes II (Agile)

- HW1 due 9/8 at 11:59pm
- If you were late…