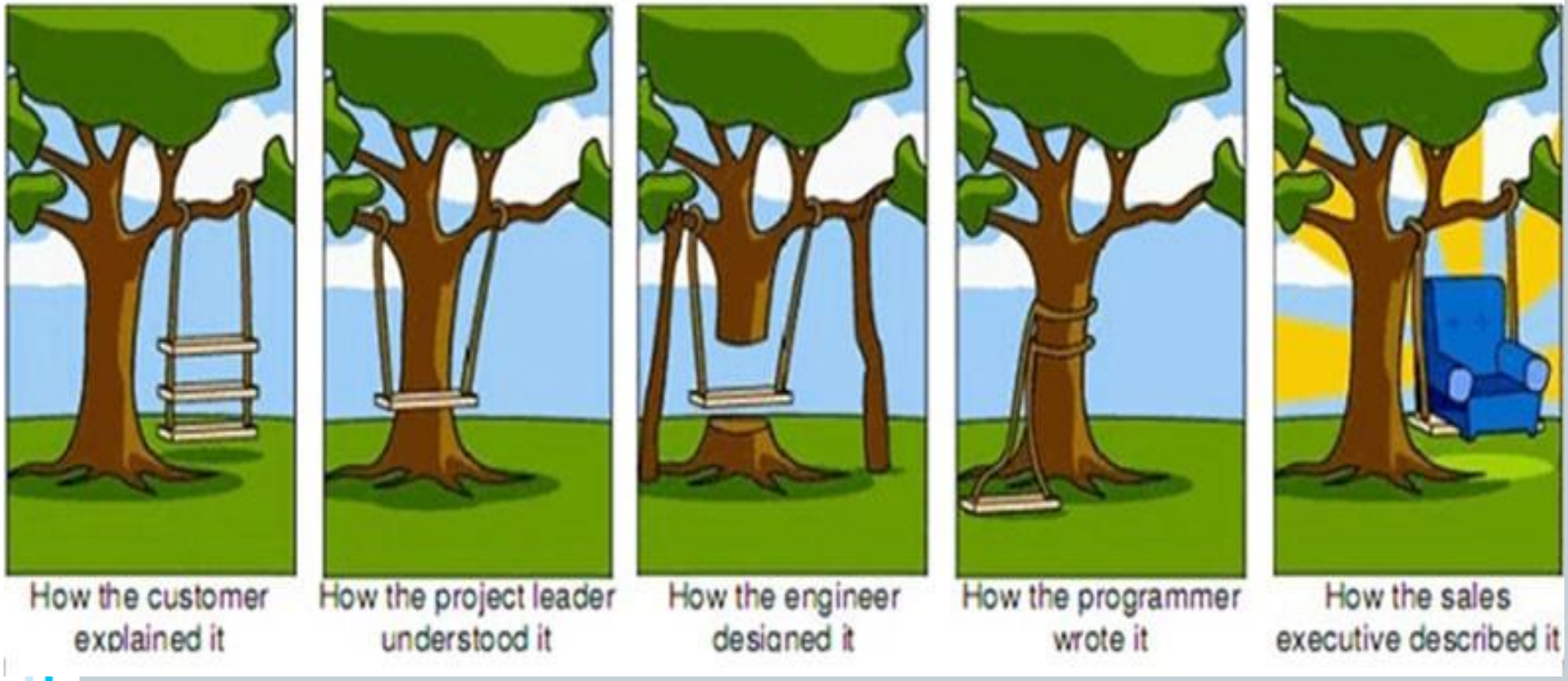
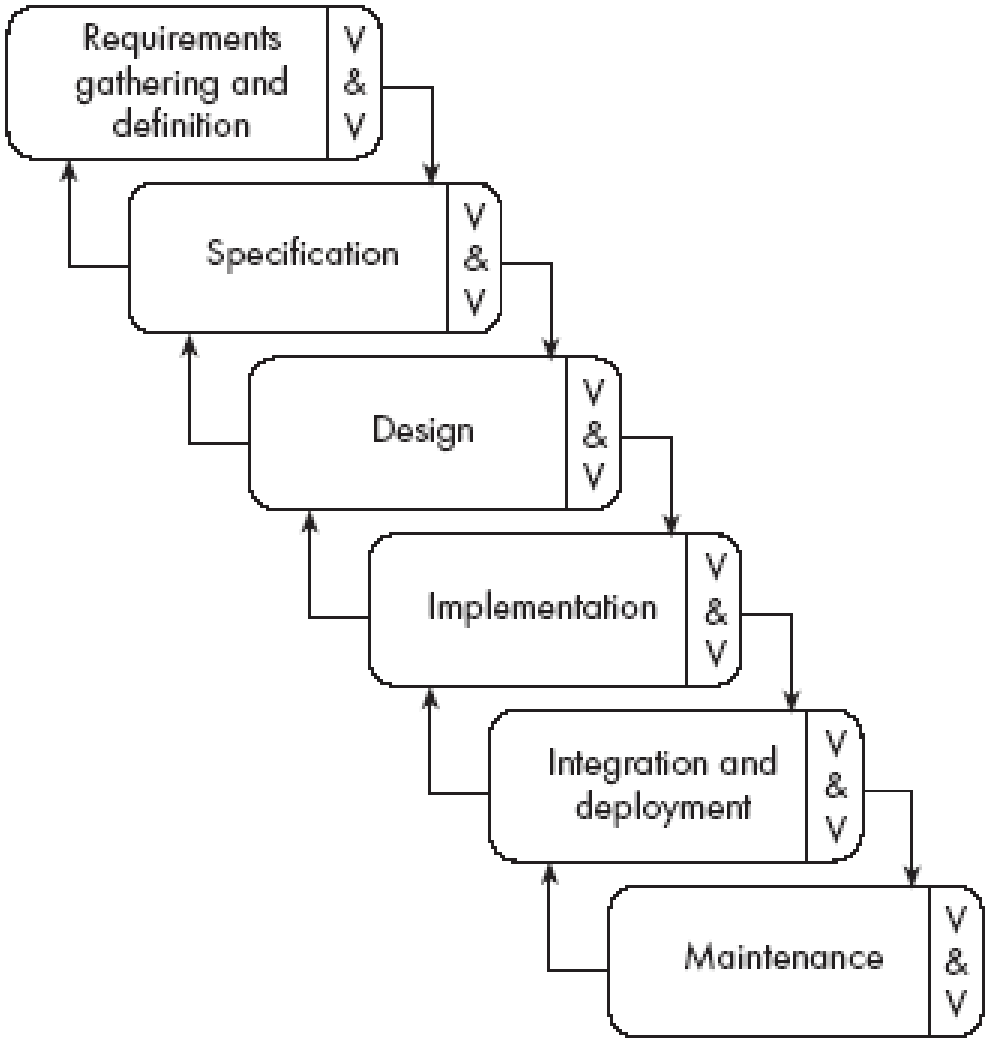
CS3004D Software Engineering

Software Crisis



The waterfall model



Software Process Models

- Software process models are general approaches for organizing a project into activities.
- Also termed as Software Life Cycle Model
  - Help the project manager and his or her team to decide:
    - What work should be done;
    - In what sequence to perform the work.
  - The models should be seen as *aids to thinking*, not rigid prescriptions of the way to do things.
  - Each project ends up with its own unique plan.

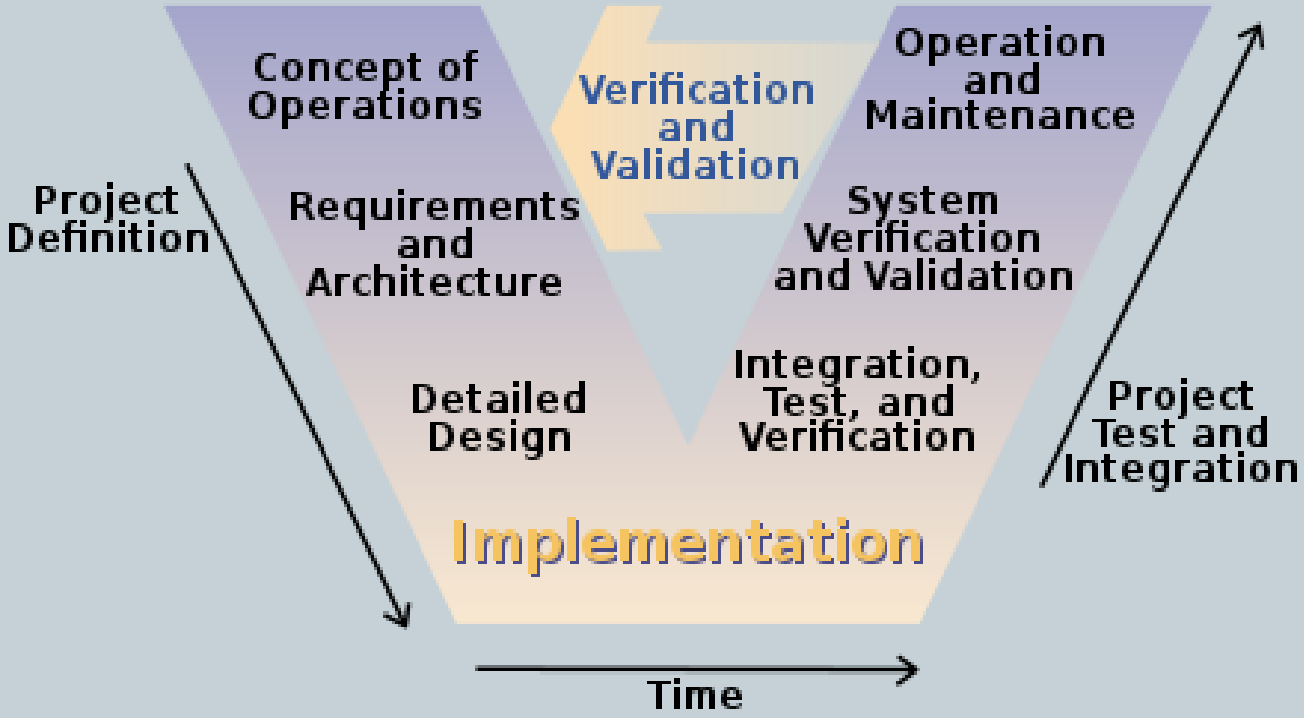
The waterfall model

- The classic way of looking at S.E. that accounts for the importance of requirements, design and quality assurance.
  - The model suggests that software engineers should work in a series of stages.
  - Before completing each stage, they should perform quality assurance (verification and validation).
  - The waterfall model also recognizes, to a limited extent, that you sometimes have to step back to earlier stages.

Limitations of the waterfall model

- The model implies that you should attempt to complete a given stage before moving on to the next stage
  - Does not account for the fact that requirements constantly change.
  - It also means that customers can not use anything until the entire system is complete.
- The model makes no allowances for prototyping.
- Have to wait till integration, and may give false impression on the progress of project
- It implies that you can get the requirements right by simply writing them down and reviewing them.
- The model implies that once the product is finished, everything else is maintenance.

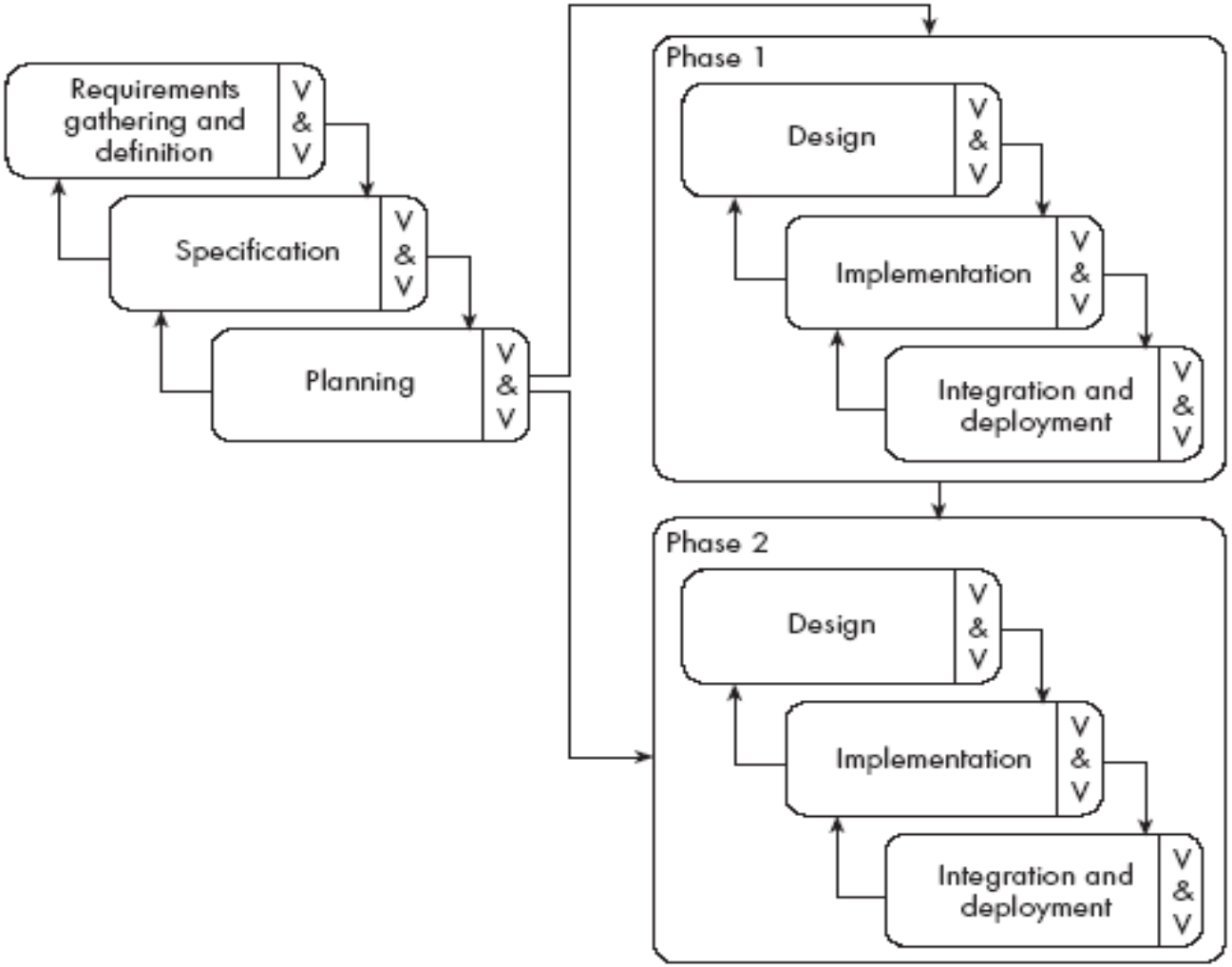
V Model


Concept of Operations
Verification and Validation
Operation and Maintenance
Project Definition
Requirements and Architecture
System Verification and Validation
Detailed Design
Integration, Test, and Verification
Project Test and Integration
Implementation
Time

V Model

- A variation of waterfall model
  - Emphasises Verification and Validation
  - Testing activities are planned in parallel with the development
  - Used in applications where reliability and safety are important

The phased-release model



The phased-release model

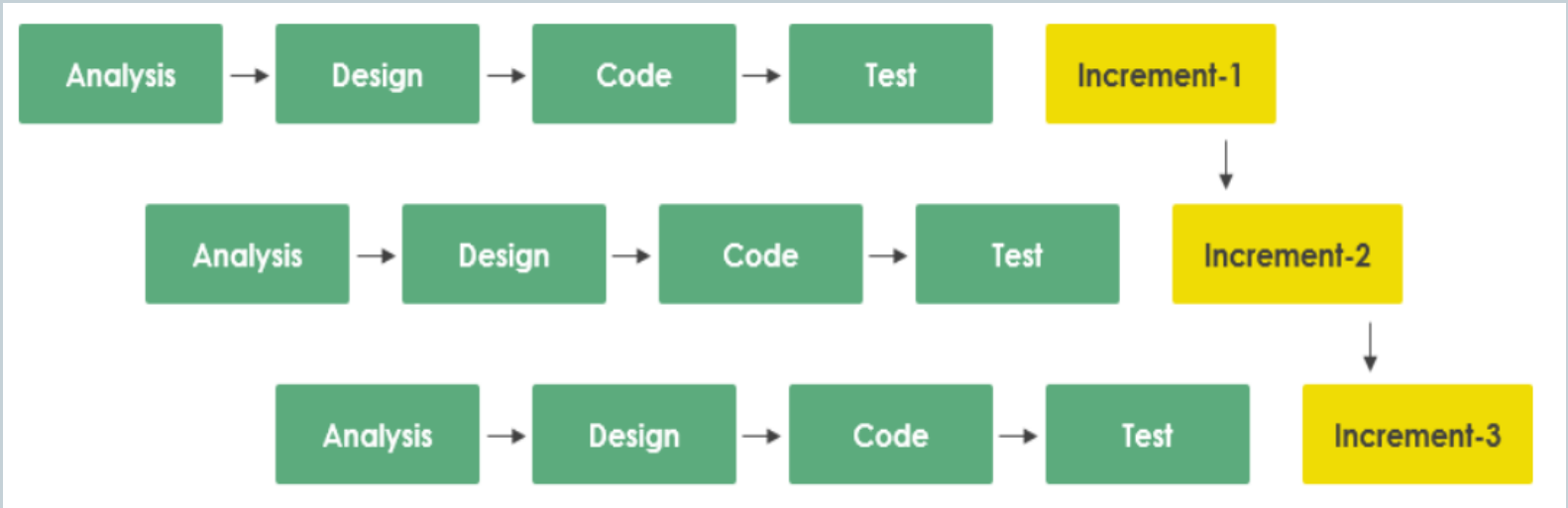
- It introduces the notion of *incremental* development.
  - After requirements gathering and planning, the project should be broken into separate subprojects, or *phases*.
  - Each phase can be released to customers when ready.
  - Parts of the system will be available earlier than when using a strict waterfall approach.
  - However, it continues to suggest that all requirements be finalized at the start of development.

Prototyping

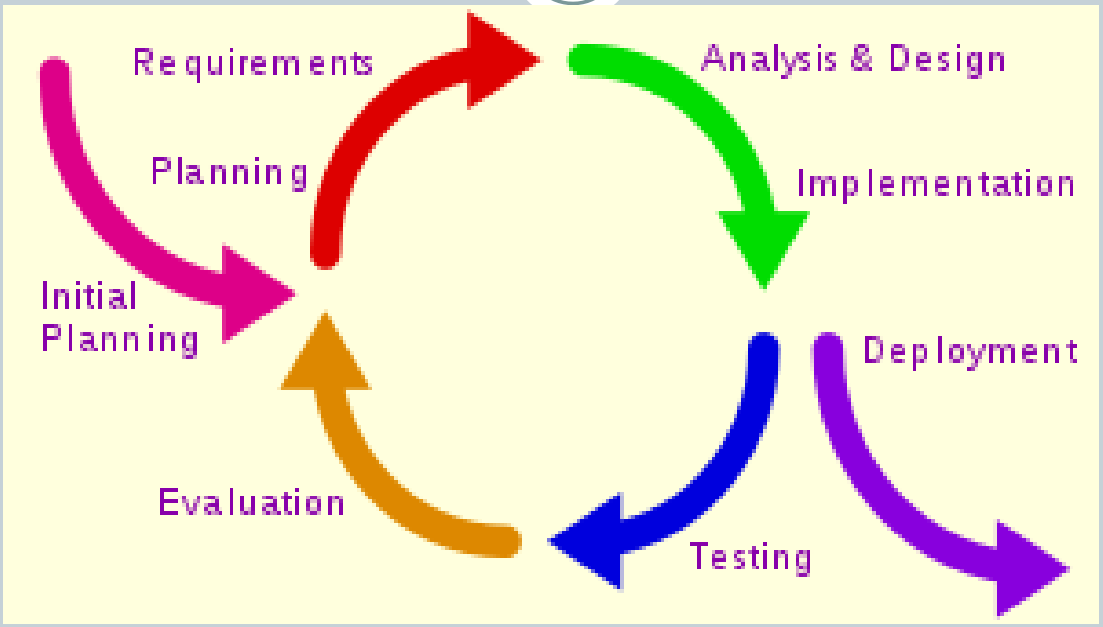
- Before the development starts a prototype needs to be built.
- A prototype is a toy implementation of the system.
- The software designer and implementer can get valuable feedback from the users early in the project

Incremental

- The product is designed, implemented and tested incrementally
- A little more is added each time until the product is finished

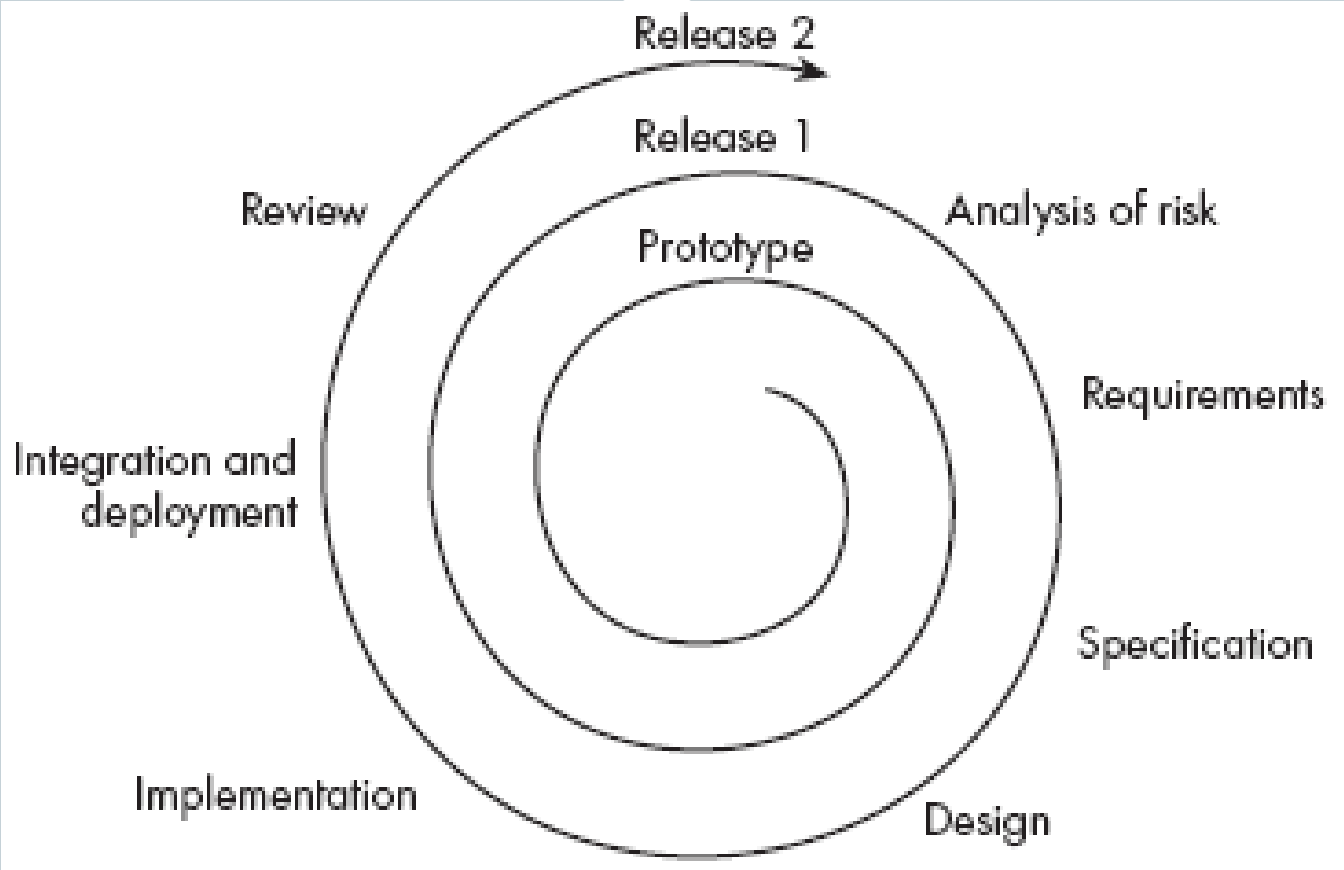


Iterative Approach



- Begins by specifying and implementing just part of the software, which can then be reviewed and prioritized in order to identify further requirements

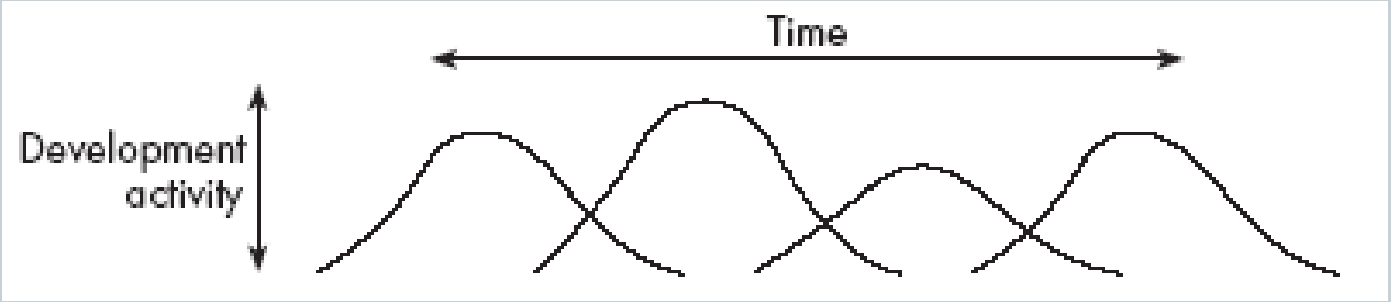
The spiral model



The spiral model

- It explicitly embraces prototyping and an *iterative* approach to software development.
  - Start by developing a small prototype.
  - Followed by a mini-waterfall process, primarily to gather requirements.
  - Then, the first prototype is reviewed.
  - In subsequent loops, the project team performs further requirements, design, implementation and review.
  - The first thing to do before embarking on each new loop is risk analysis.
  - Maintenance is simply a type of on-going development.

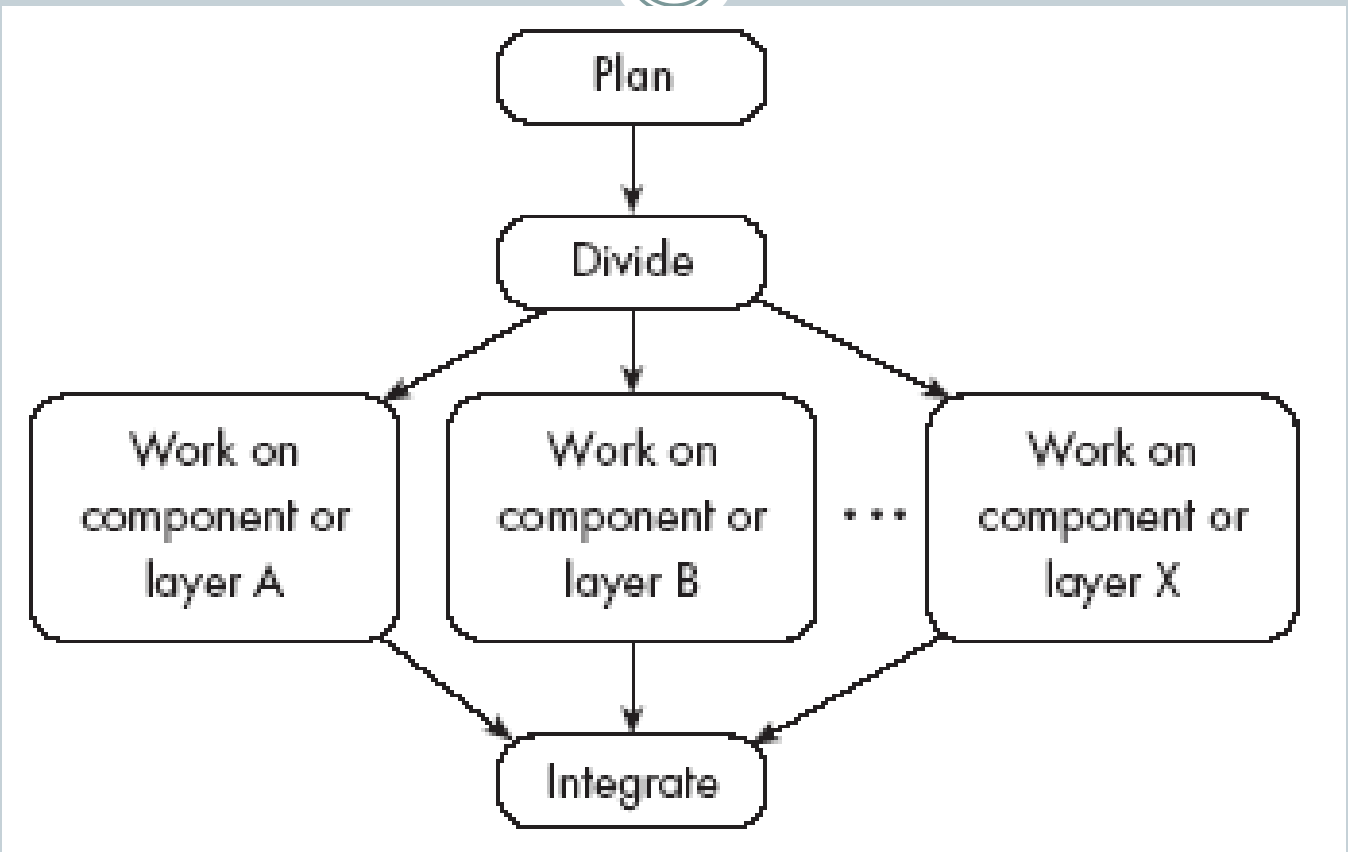
The evolutionary model



The evolutionary model

- It shows software development as a series of hills, each representing a separate loop of the spiral.
  - Shows that loops, or releases, tend to overlap each other.
  - Makes it clear that development work tends to reach a peak, at around the time of the deadline for completion.
  - Shows that each prototype or release can take
    - different amounts of time to deliver;
    - differing amounts of effort.

The concurrent engineering model



The concurrent engineering model

- It explicitly accounts for the divide and conquer principle.
  - Each team works on its own component, typically following a spiral or evolutionary approach.
  - There has to be some initial planning, and periodic integration.

- Open source models
  - software is distributed for free along with the source code, and interested people contribute improvements, without being paid by users
- Agile Model

Choosing a process model

When planning a particular project, you can **combine** the features of the models that apply best to your current project

Reengineering

- Periodically project managers should set aside some time to re-engineer part or all of the system
  - The extent of this work can vary considerably:
    - Cleaning up the code to make it more readable.
    - Completely replacing a layer.
    - *Re-factoring* part of the design.
  - In general, the objective of a re-engineering activity is to increase maintainability.