

CIS W338

Lecture SWE02

Modeling the Process and Life Cycle

What Is a Process?

- The text defines process as “a series of steps involving activities, constraints, and resources that produce an intended output of some kind.”
- ISO/IEC 12207 defines process as “a set of interrelated activities, which transform inputs into outputs.”
- Mark Paulk (author of the CMM[®] for Software) gives the definition “what you do to do what you do.”

® Capability Maturity Model and CMM are registered trademarks of Carnegie Mellon University.

Process Characteristics

- Prescribes all of the major activities.
- Uses resources subject to a set of constraints, and produces intermediate and final products.
- May be composed of sub-processes that are linked together or as a hierarchy.
- Each activity has an entrance and exit criteria.
- Activities are organized in a sequence.
- There are a set of guiding principals that explain the goals of each activity.
- Constraints may be applied to an activity, resource, or product.

What Is a Life Cycle?

- In biology the term *life cycle* is “the continuous sequence of changes undergone by an organism from one primary form to the development of the same form again*.”
 - An insect starts as an egg, which becomes a larva, then a pupa, and finally an adult, which lays eggs, starting the cycle over again.
- A software product undergoes a sequence of changes starting with an initial concept and ending with its retirement.

* The Random House Dictionary of the English Language, 1966

Stages in Software Development

- Requirements analysis and definition
- System design
- Program design
- Writing the program
- Unit testing
- Integration testing
- System testing
- System delivery
- Maintenance

Life Cycle Models

- Code and Fix
- Waterfall Model
- 'V' Model
- Prototyping Model
- Operational Specification
- Transformational Model
- Phased Development
- Spiral Model
- Unified Development Model

Code and Fix Model

- Write some code
- Fix the problems in the code

Problems With Code and Fix

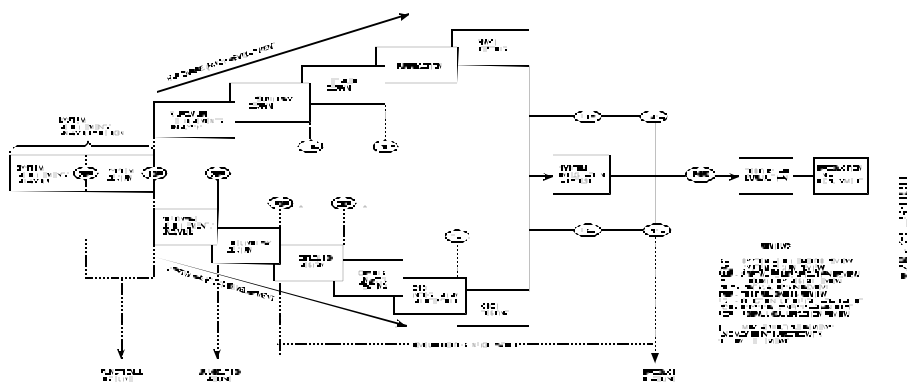
- Code becomes poorly structured
- Poor match for user's needs
- Expensive

Waterfall Model

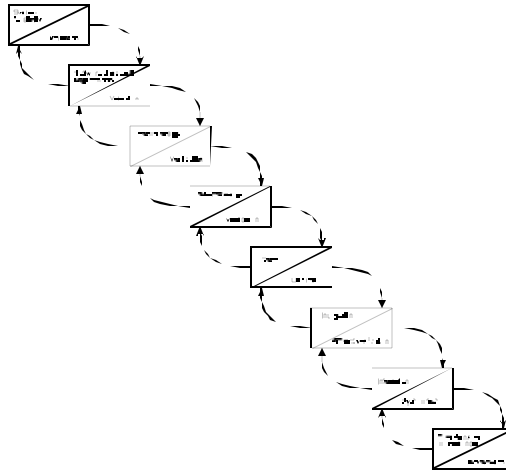
- Development in successive stages (or phases)
- Results of one phase basis for subsequent phase
- Reviews between phases

Waterfall Lifecycle

(As Shown in DOD-STD-2167A)

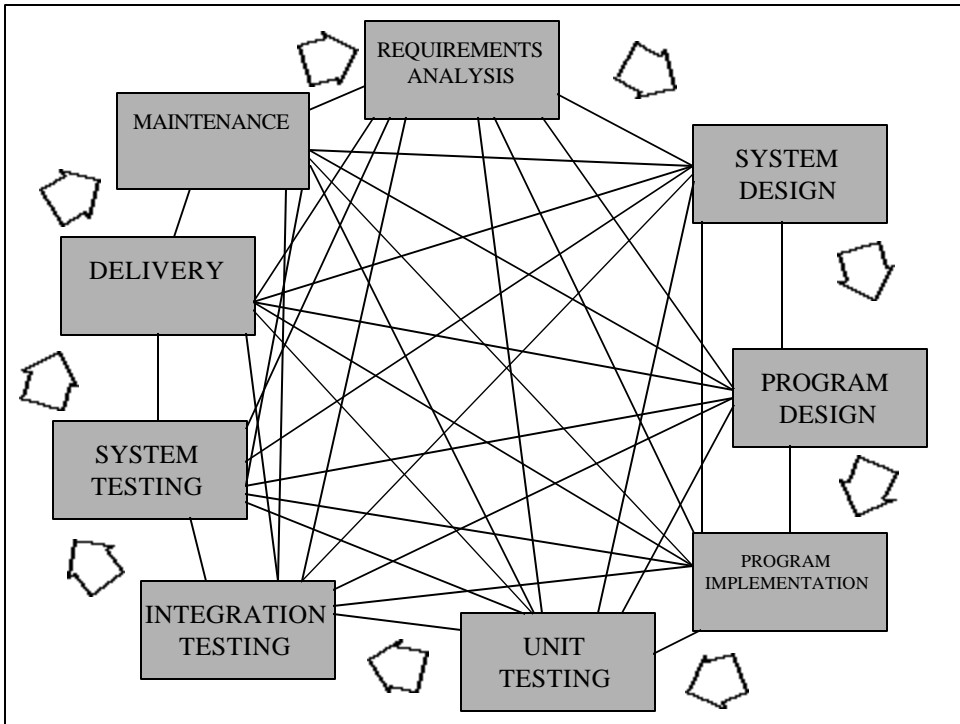


Bohem's Waterfall Model



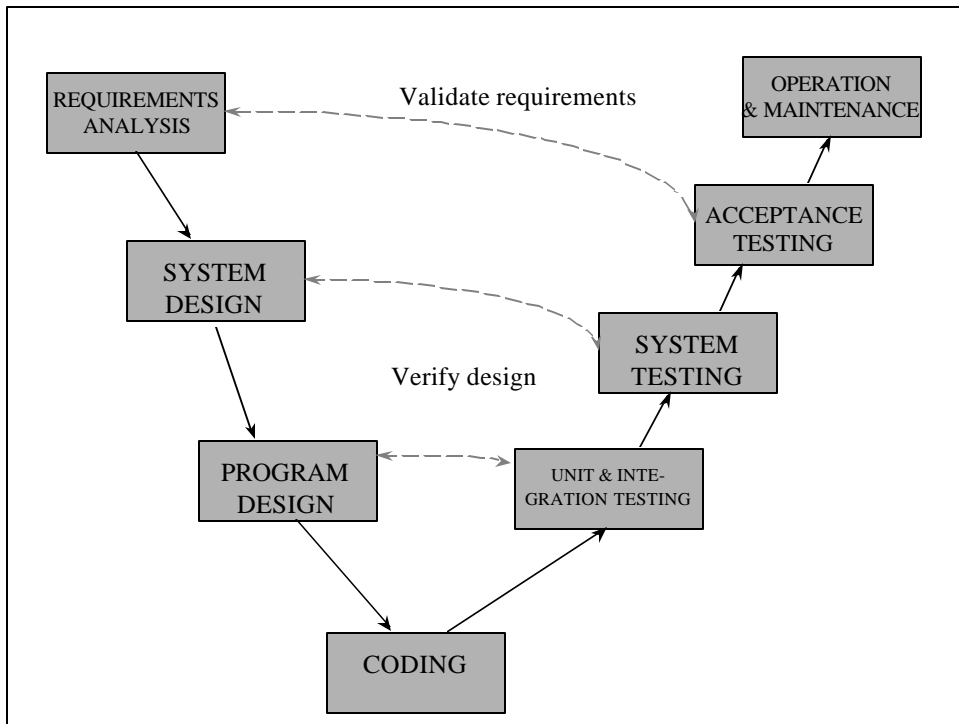
Issues With Waterfall Model

- Too much emphasis on fully elaborated documents as completion criteria.
- Reviews became elaborate “dog and pony” shows.
- Meaningful review and feedback not possible.
- No insight as to how each activity transforms one artifact to another. Thus, no guidance on how to handle changes that occur during development.



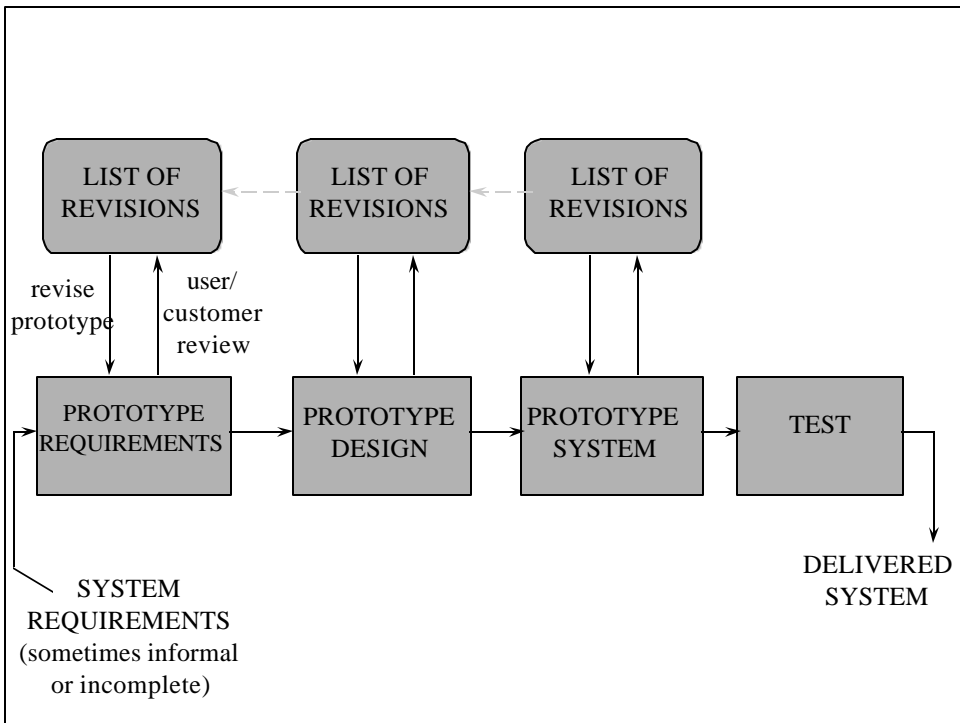
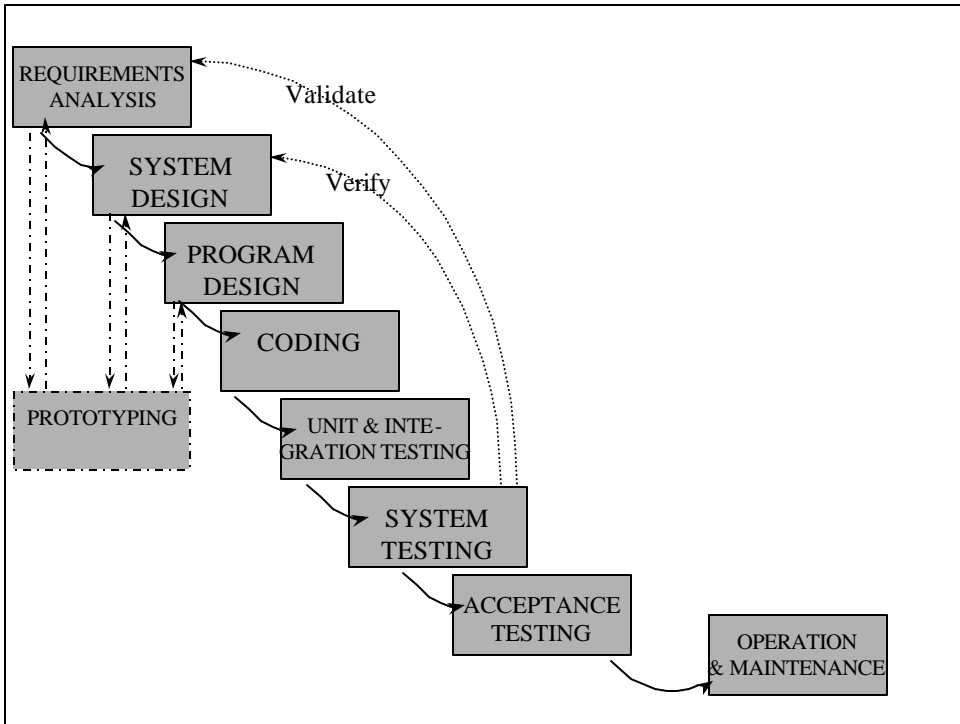
‘V’ Model

- Variation on the waterfall model that shows how the testing activities are related to analysis and design.



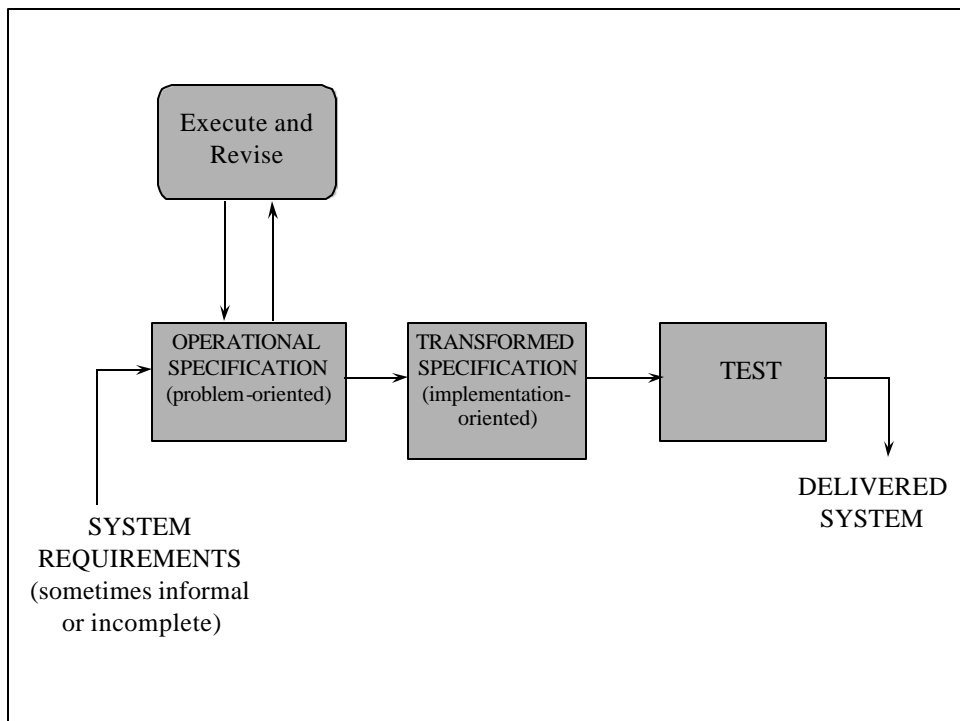
Prototyping Model

- A prototype is a partially developed product that enables customers and developers to examine some aspect of the proposed system and decide if it is suitable or appropriate for the finished product.



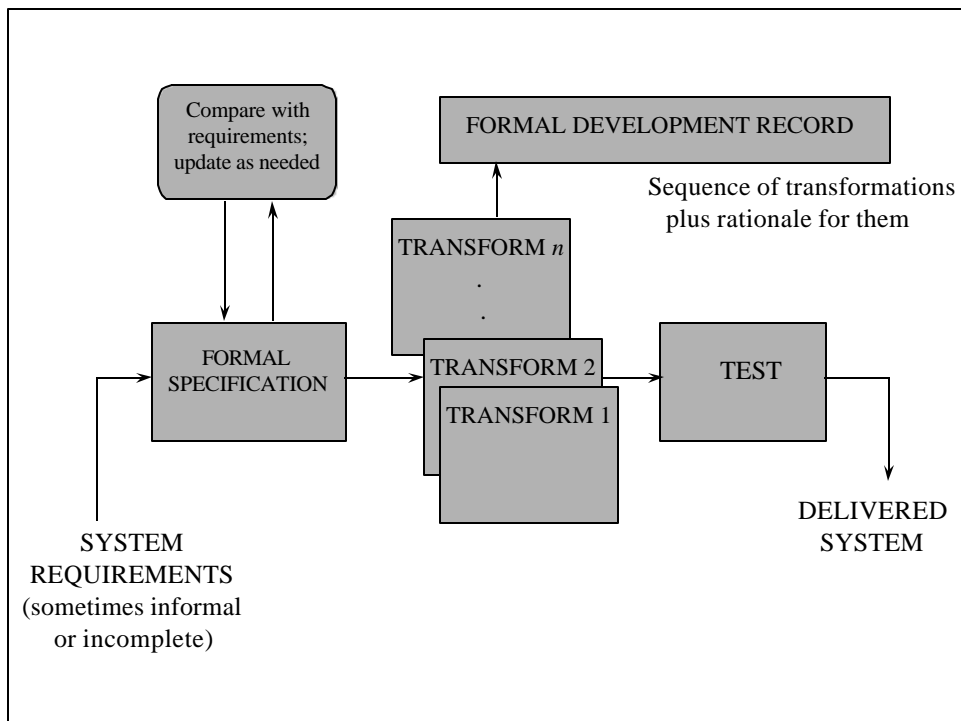
Operational Specification

- For many systems, uncertainty about requirements leads to changes.
- In the operational specification model, the system requirements are evaluated or executed in a way that demonstrates the behavior of the system.



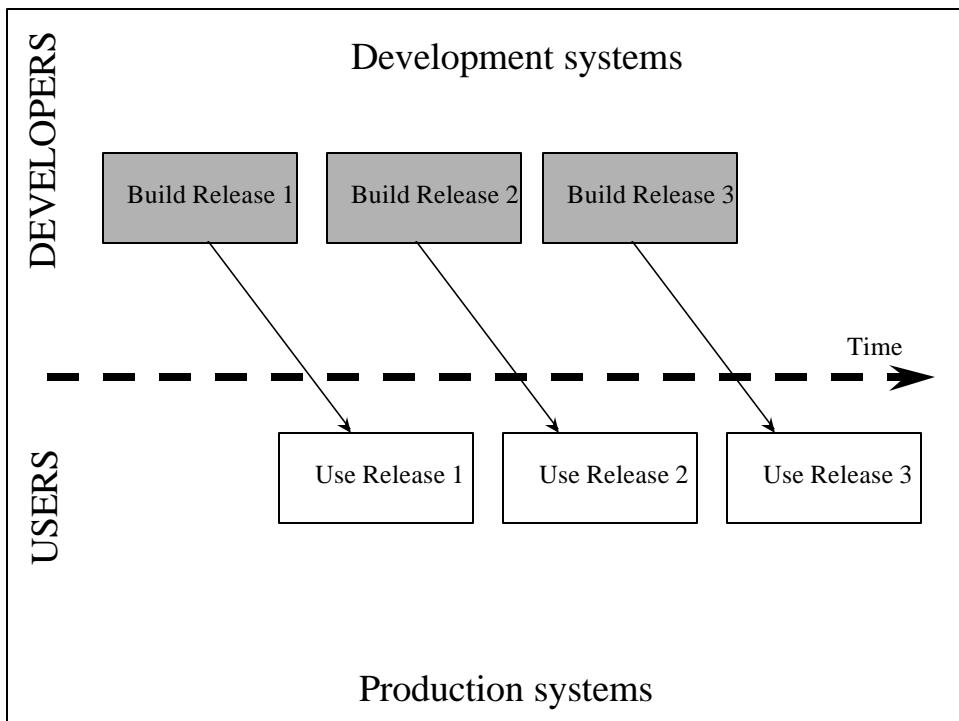
Transformational Model

- The transformational model tries to reduce the opportunity for error by eliminating major developmental steps.
- It applies a series of transformations using automated support.
- Example of transformations
 - Changing data representations
 - Selecting algorithms
 - Optimizing
 - Compiling

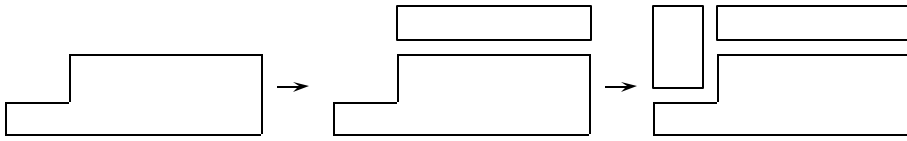


Phased Development

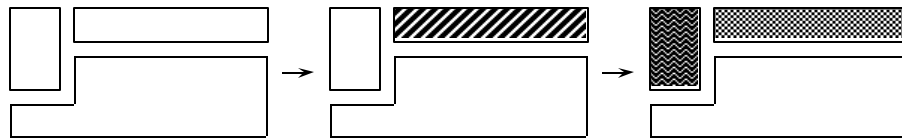
- The system is developed and delivered in pieces, thus giving the user some functionality while the rest of the system is developed.
- Incremental development
 - The system is partitioned into subsystems by functionality.
- Iterative development
 - Deliver a full system and then change the functionality of subsystems in subsequent releases.



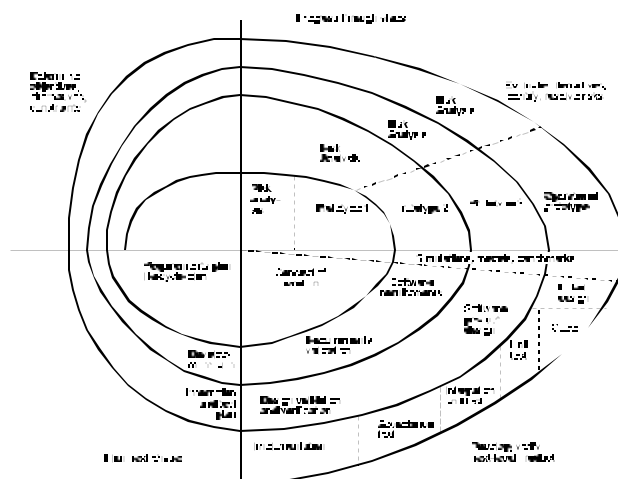
INCREMENTAL DEVELOPMENT



ITERATIVE DEVELOPMENT



Bohem's Spiral Model



Typical Cycle of the Spiral

- Identify
 - objectives
 - alternatives
 - constraints
- Evaluate alternatives and identify risks
- Build increment
- Plan next cycle

Unified Process

- Use Case Driven
- Architecture-Centric
 - components
 - interfaces
- Iterative and Incremental

Use Case Driven

Use Case: A description of a set of sequences of actions, including variants, that a system performs that yields an observable result to a particular actor.

Actor: A coherent set of roles that users of use cases plan when interacting with those use cases.

Architecture-centric

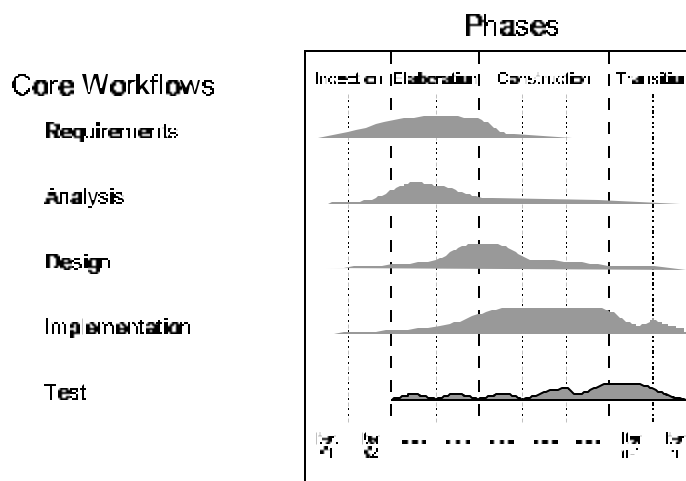
Software Architecture — similar to building

- Looks at different viewpoints
- Embodies the most significant static and dynamic aspects of the system
- Grows out of the user needs
- View of the whole design with important characteristics made visible by leaving details aside.

Iterative and Incremental

- Break into mini-projects
- Controlled iterations
- Cost risk bound to a single iteration
- Schedule risk bound to a single iteration
- Speeds up tempo
- Gets early user feedback

Unified Process Life Cycle



Inception Phase

- The first phase of the software life cycle, where the seed idea for the development is brought up to the point of being sufficiently well founded to warrant entering into the elaboration phase.

Elaboration Phase

- The second phase of the software lifecycle, where the architecture is defined.

Construction Phase

- The third phase of the software life cycle, where the software is brought from an executable architectural baseline to the point where it is ready to be transitioned to the user community.

Transition Phase

- The fourth phase in the software life cycle, where the software is turned into the hands of the user community.

Iteration

- A distinct set of activities conducted according to a devoted (iteration) plan and evaluation criteria that results in a release, either internal or external.

Release

- A relatively complete and consistent set of artifacts — possibly including a build — delivered to an internal or external user.
 - Artifact: A tangible piece of information
 - Build: An executable version of the system

Iteration Plan

- A fine-grained plan for an iteration. A plan that states the expected costs in terms of time and resources, and the expected output in terms of artifacts.
- A plan that states who should do what within the iteration and in what order.

Iteration Workflow

- A workflow representing an integration of the core workflows:
 - requirements capture
 - analysis
 - design
 - implementation
 - test

Major Milestone

- Milestone where management makes important business decisions.
- Each phase ends with a major milestone.
- Critical go/no-go decision point.

Why Model Processes?

- Whatever you do, you use a process to do it.
- By modeling the process, you can gain understanding so that:
 - The process is repeatable
 - The process is manageable
 - The process can be improved

Terminology Confusion

There are many different views of the Software [Life Cycle] Prozesse[s].

- ISO/IEC 12207
- IEEE 1074
- SEI Capability Maturity Model[®] for Software

® Capability Maturity Model and CMM are registered trademarks of Carnegie Mellon University.

ISO/IEC 12207

Standard for Software Life Cycle Processes

- Primary Life Cycle Processes:
 - Acquisition, Supply, Development, Operation, Maintenance
- Supporting Live Cycle Processes:
 - Documentation, Configuration Management, Quality Assurance, Verification, Validation, Joint Review, Audit
- Organizational Life Cycle Processes:
 - Management, Infrastructure, Improvement, Training

IEEE 1074

Standard for Developing Software Life Cycle Processes

- Activity Groupings:
 - Project Management
 - Pre-Development
 - Development
 - Post-Development
 - Integral

CMM[®] for Software

Defines Key Process areas by Level

- Level 2
 - Requirements Management, Project Planning, Project Tracking and Oversight, Subcontract Management, Quality Assurance, Configuration Management
- Level 3
 - Organization Process Focus, Organization Process Definition, Training Program, Integrated Software Management, Software Product Engineering, Intergroup Coordination, Peer Reviews
- Level 4
 - Quantitative Process Management, Software Quality Management
- Level 5
 - Defect Prevention, Technology Change Management, Process Change Management

® Capability Maturity Model and CMM are registered trademarks of Carnegie Mellon University.

Common Theme

- 12207 requires a supplier to “define or select a software life cycle model” and map the processes, activities, and tasks of 12207 to that model.
- IEEE 1074 requires organizing and mapping the activities into a Software Life Cycle Process.
- The CMM, at Level 3, requires an Organization to define an Organizational Software Process, and then for each project tailor it to form the Project’s Software Defined Process.

Information System Example

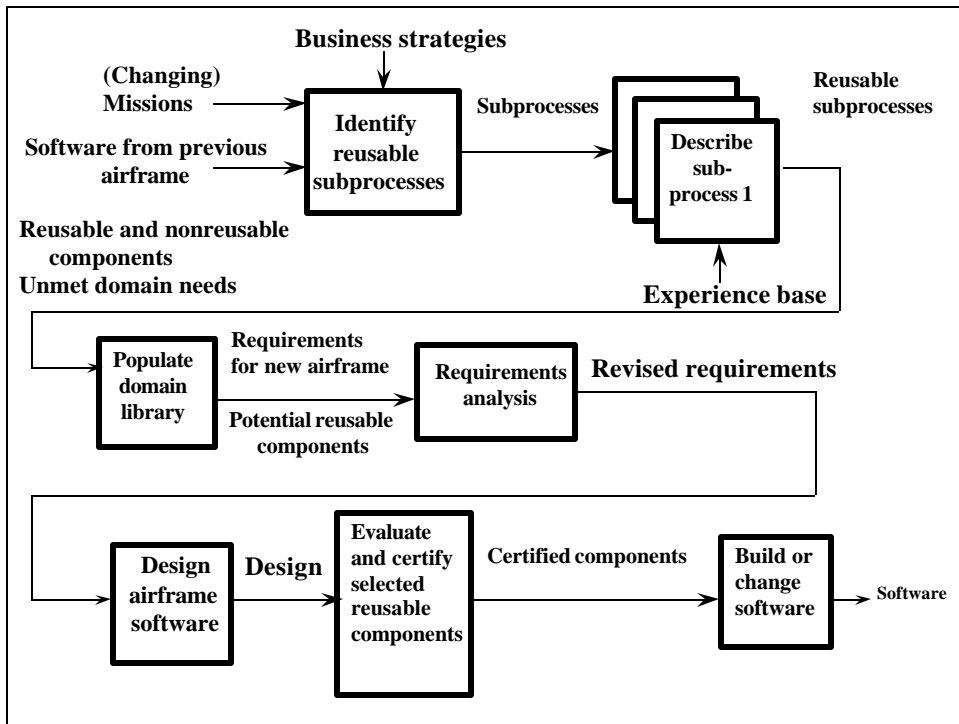
- Waterfall model too rigid
- Prototyping useful for the user interface
- Spiral model provides opportunity to revisit assumptions.
 - Need to manage risk.

Facets of Risk

- Probability
 - Likelihood that a particular problem will occur
- Severity
 - Impact that a particular problem will have
- Example: Developers have little or no experience with object-orientation.
 - Probability:
 - Low – all employees sent to an intensive four-week course
 - Severity:
 - High – development team unable to complete project within schedule.

Real Time Example

- Reuse software from the Ariane-4 in the Ariane-5
- Goal:
 - Reduce risk
 - Increase Productivity
 - Increase Quality



What Really Happened

- The Inertial Reference System from the Ariane-4 was reused, unmodified, in the Ariane-5
- Features of the Ariane-4 not required by the Ariane-5 were retained as “... it was not wise to make changes in software which worked well on Ariane-4.”