Introduction to Software Engineering

(CS350)

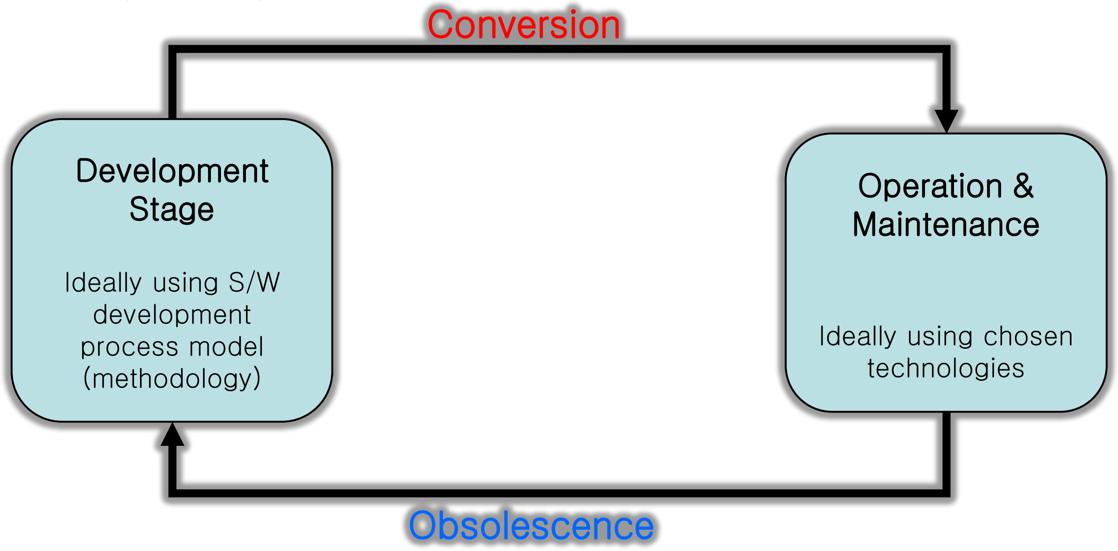
Lecture 02

Jongmoon Baik

Process Models

What is a S/W Life Cycle?

- "The series of stages in form and functional activity through which an organism passes between successive recurrence of a specified primary stage" – Webster (1892)
- "Period of time that begins when a software product is conceived and ends when the product is retired from use" – Don Reifer (1997)


Conversion
Development Stage
Ideally using S/W development process model (methodology)
Operation & Maintenance
Ideally using chosen technologies
Obsolescence

So...What is a Process?

- A process is a sequence of steps performed for a given purpose.

Websters:

"a series of actions or operations conducing to an end."

The concept of software process is rarely presented in undergraduate education.

- Software process – a set of activities, methods, best practices, deliverables, and automated tools that stakeholders use to develop and continuously improve software.
  - Using a consistent process for software development:
    - Create efficiencies that allow management to shift resources between projects
    - Produces consistent documentation that reduces lifetime costs to maintain the systems
    - Promotes quality

What is a S/W Process Model?

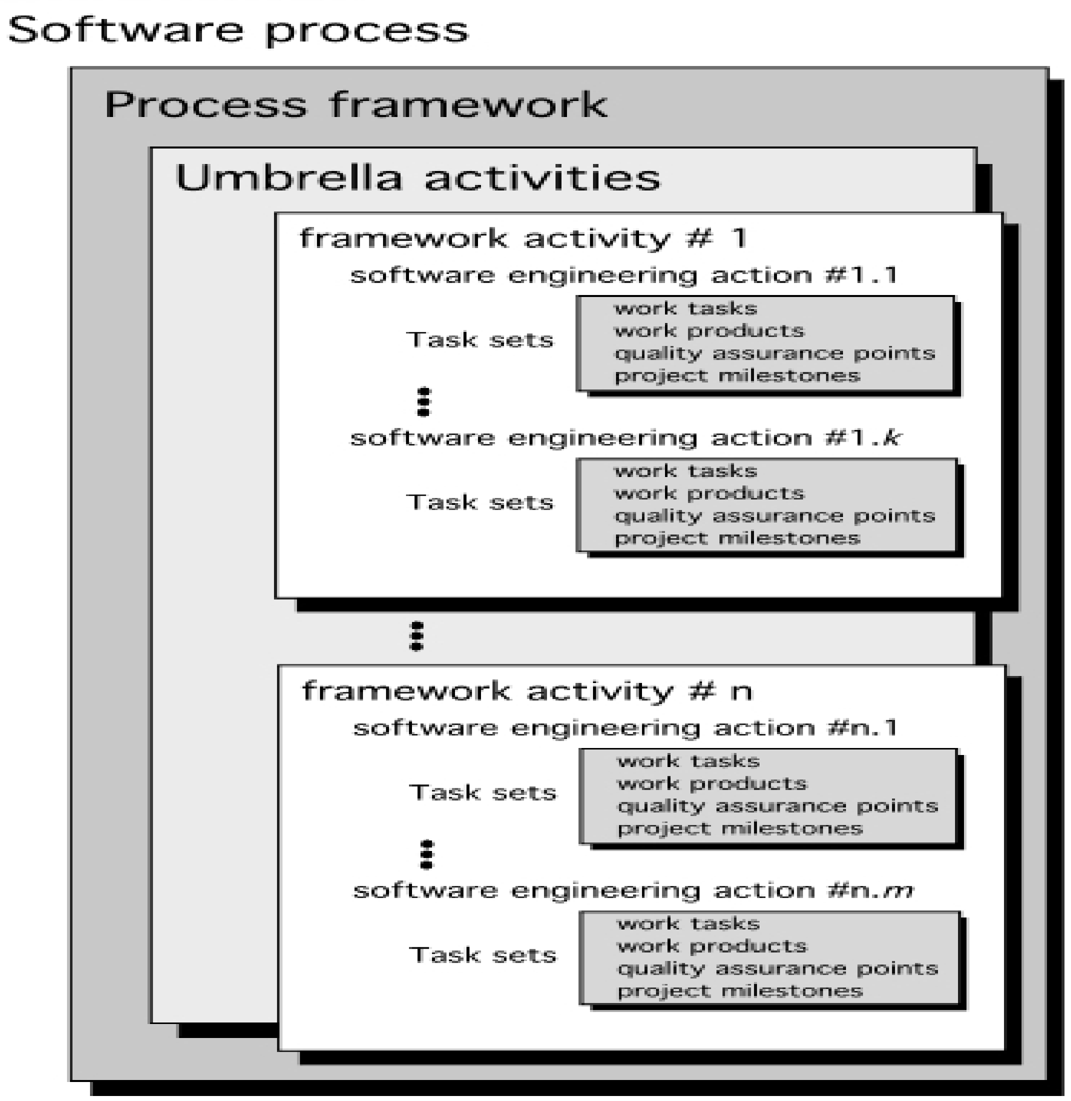
- An abstract representation of a process
  - A description of a process from some particular perspective
  - Attempt to generalize the s/w development process into steps with associated activities and/or artifacts

- Proliferation of S/W Process Models
  - Provides flexibility for organizations to deal with the wide variety of software project situations, cultures, and environments
  - Weakens the defenses against some common sources of project failure

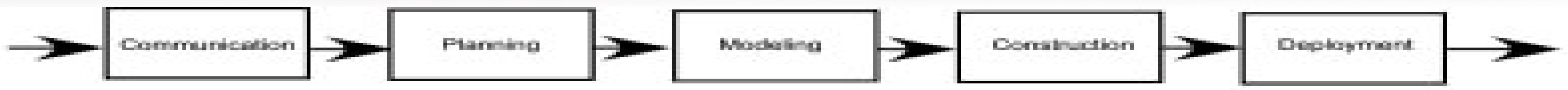
Process ≠ Lifecycle

- Software process is not the same as life cycle models.
  - process refers to the specific steps used in a specific organization to build systems
  - indicates the specific activities that must be undertaken and artifacts that must be produced
  - *process definitions* include more detail than provided lifecycle models
- Software processes are sometimes defined in the context of a lifecycle model.

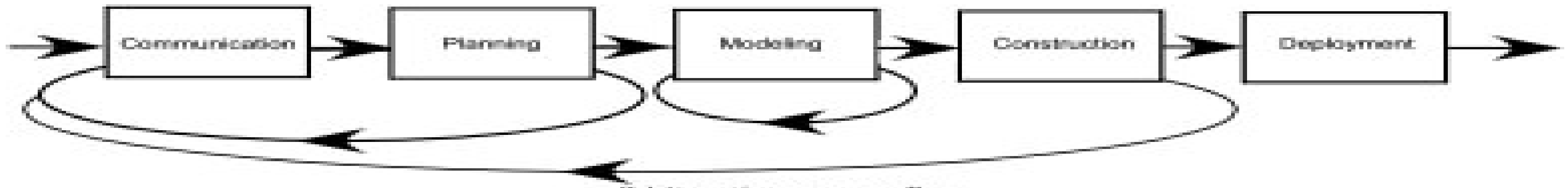
A Generic Process Model



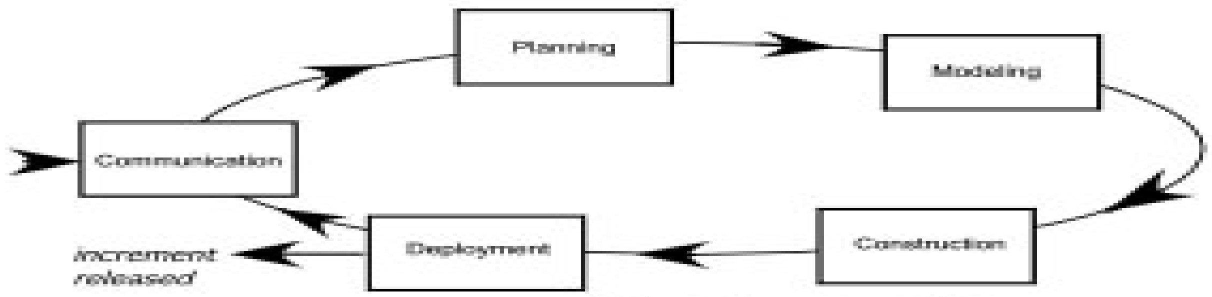
Process Flow



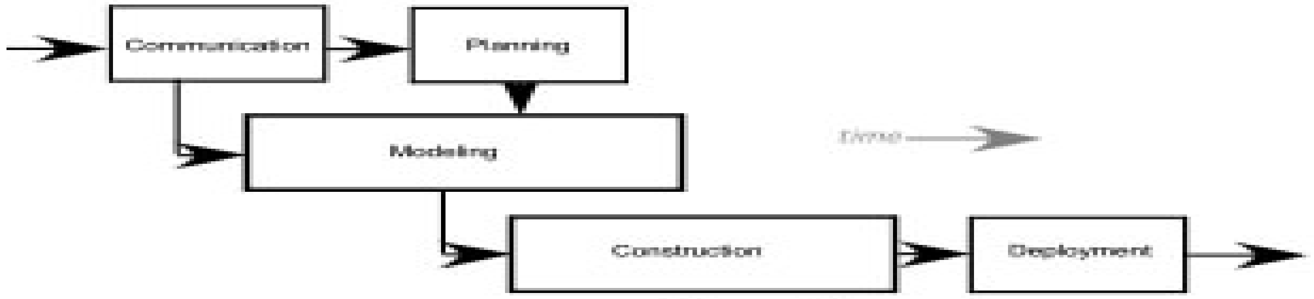
(a) linear process flow



(b) iterative process flow



(c) evolutionary process flow



(d) parallel process flow

Identifying a Task Set

- A task set defines the actual work to be done to accomplish the objectives of a software engineering action.
  - A list of the task to be accomplished
  - A list of the work products to be produced
  - A list of the quality assurance filters to be applied

Process Patterns

- A *process pattern*
  - describes a process-related problem that is encountered during software engineering work,
  - identifies the environment in which the problem has been encountered, and
  - suggests one or more proven solutions to the problem.

- Stated in more general terms, a process pattern provides you with a *template* [Amb98] — a consistent method for describing problem solutions within the context of the software process.

Process Pattern Types

- *Stage patterns*—defines a problem associated with a framework activity for the process.
  - e.g.: EstablishingCommunication
- *Task patterns*—defines a problem associated with a software engineering action or work task and relevant to successful software engineering practice
  - e.g. : RequirementGathering
- *Phase patterns*—define the sequence of framework activities that occur with the process, even when the overall flow of activities is iterative in nature.
  - e.g.: SpiralModel or Prototyping

Process Assessment and Improvement

- **Standard CMMI Assessment Method for Process Improvement (SCAMPI)** — provides a five step process assessment model that incorporates five phases: initiating, diagnosing, establishing, acting and learning.
- **CMM-Based Appraisal for Internal Process Improvement (CBA IPI)**—provides a diagnostic technique for assessing the relative maturity of a software organization; uses the SEI CMM as the basis for the assessment [Dun01]
- **SPICE—The SPICE (ISO/IEC15504)** standard defines a set of requirements for software process assessment. The intent of the standard is to assist organizations in developing an objective evaluation of the efficacy of any defined software process. [ISO08]
- **ISO 9001:2000 for Software—**a generic standard that applies to any organization that wants to improve the overall quality of the products, systems, or services that it provides. Therefore, the standard is directly applicable to software organizations and companies. [Ant06]

Need to look at with respect to:

- **Scope, Time, Resources, Quality**

  (remember these throughout the course)

- Stakeholders
- Environments
  - Business / Market
  - Cultures
- Moral, legal constraints …

So, when looking at projects

Need to ask:

What SDLC would fit my project best?

(Is this better than what type of project would fit each SDLC?)

Prescriptive Models

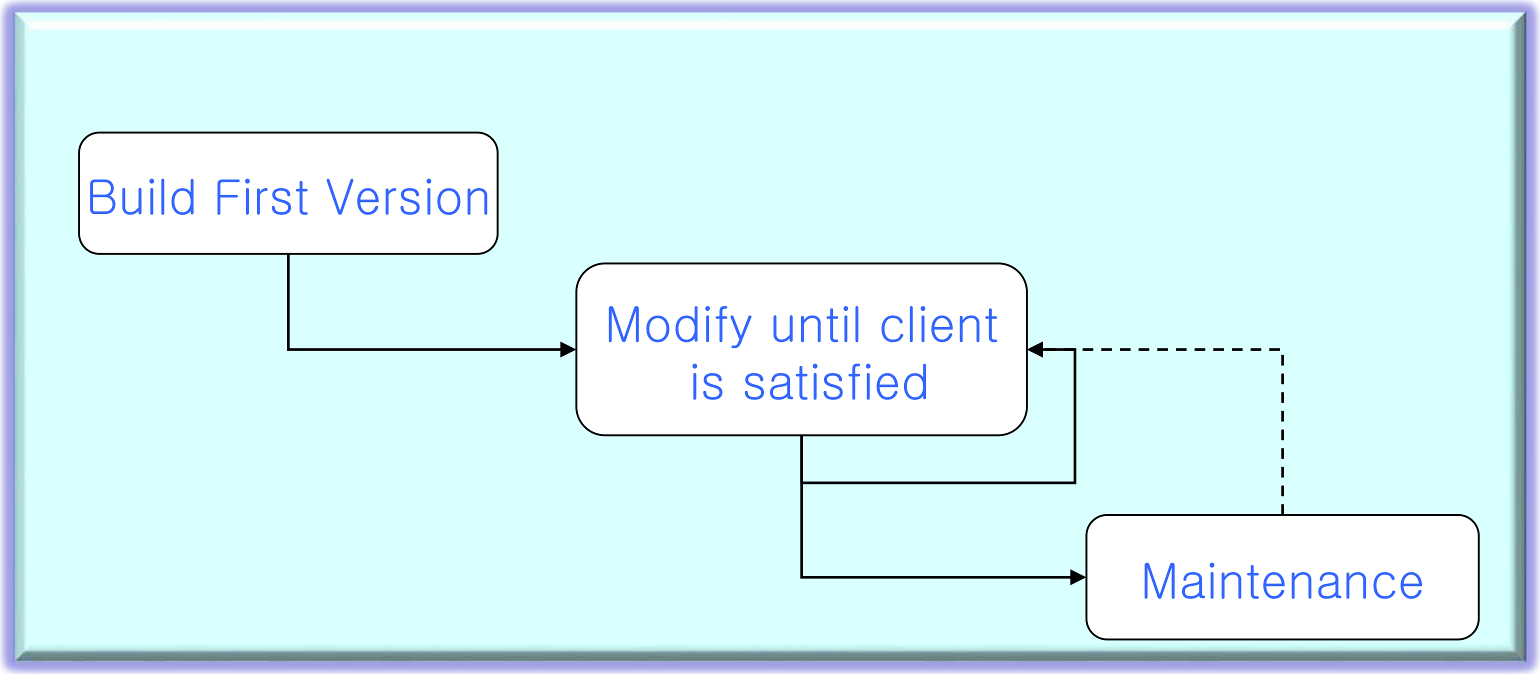
- Prescriptive process models advocate an orderly approach to software engineering

That leads to a few questions …

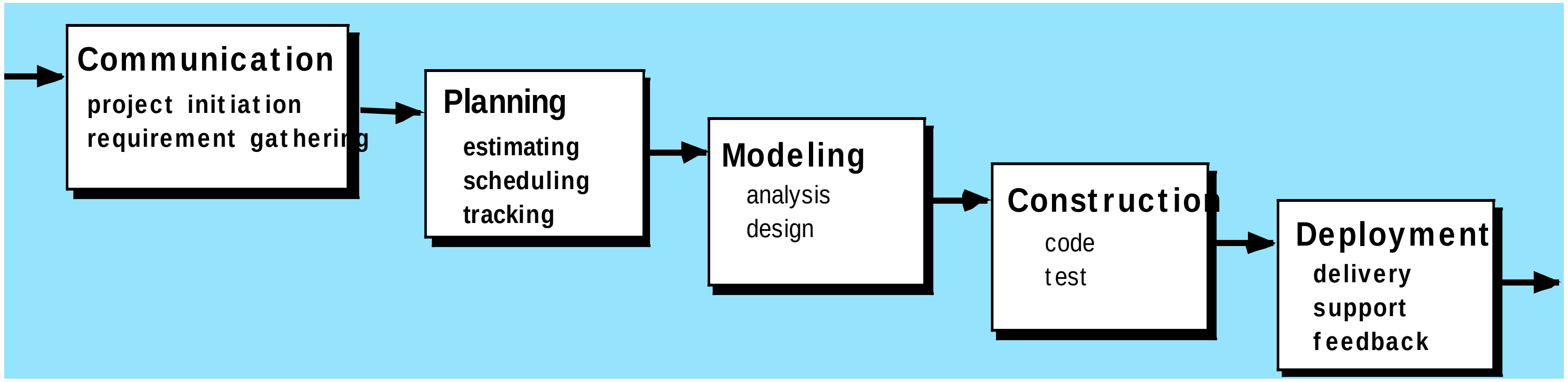
- If prescriptive process models strive for structure and order, are they inappropriate for a software world that thrives on change?
- Yet, if we reject traditional process models (and the order they imply) and replace them with something less structured, do we make it impossible to achieve coordination and coherence in software work?

Build-And-Fix Model

- The worst (Ad Hoc) model for a software project
  - Become a mess, chuck it, start over
- No proper specifications and design steps
- Discouraged from using this model



The Waterfall Model



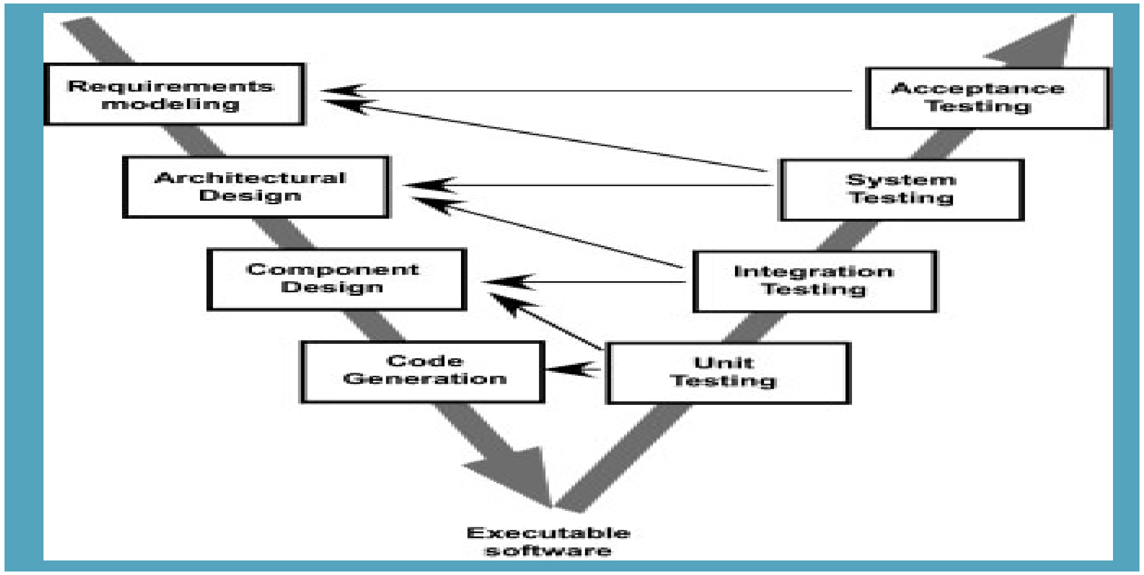
- First proposed in 1970 by W.W. Royce
- Development flows steadily through:
  - requirements analysis, design implementation, testing, integration, and maintenance.
- Royce advocated iterations of waterfalls adapting the results of the precedent waterfall.
- Referred to as a linear sequential model, document-driven model

Waterfall Problems

- Increasing use of resources?
- Oops
  - Go back to a previous step
  - Progressively more costly
- Downside
  - Cost
  - Time
  - Cascading Bugs

- Where appropriate?

The V-Model

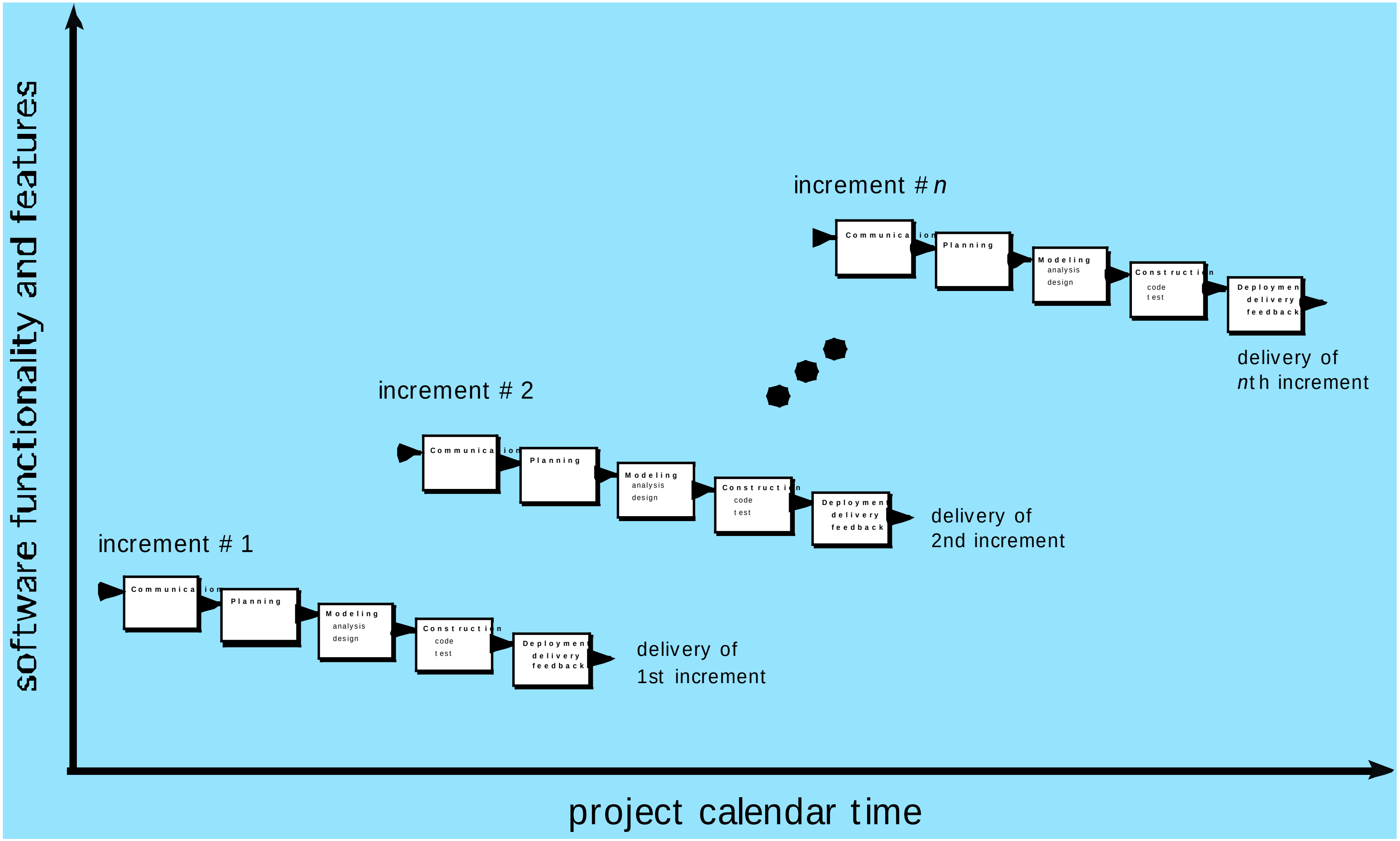


- Often used in system engineering environments to represent the system development lifecycle.
  - summarizes the main steps taken to build *systems not specifically software*
  - describes appropriate deliverables corresponding with each step in the model.

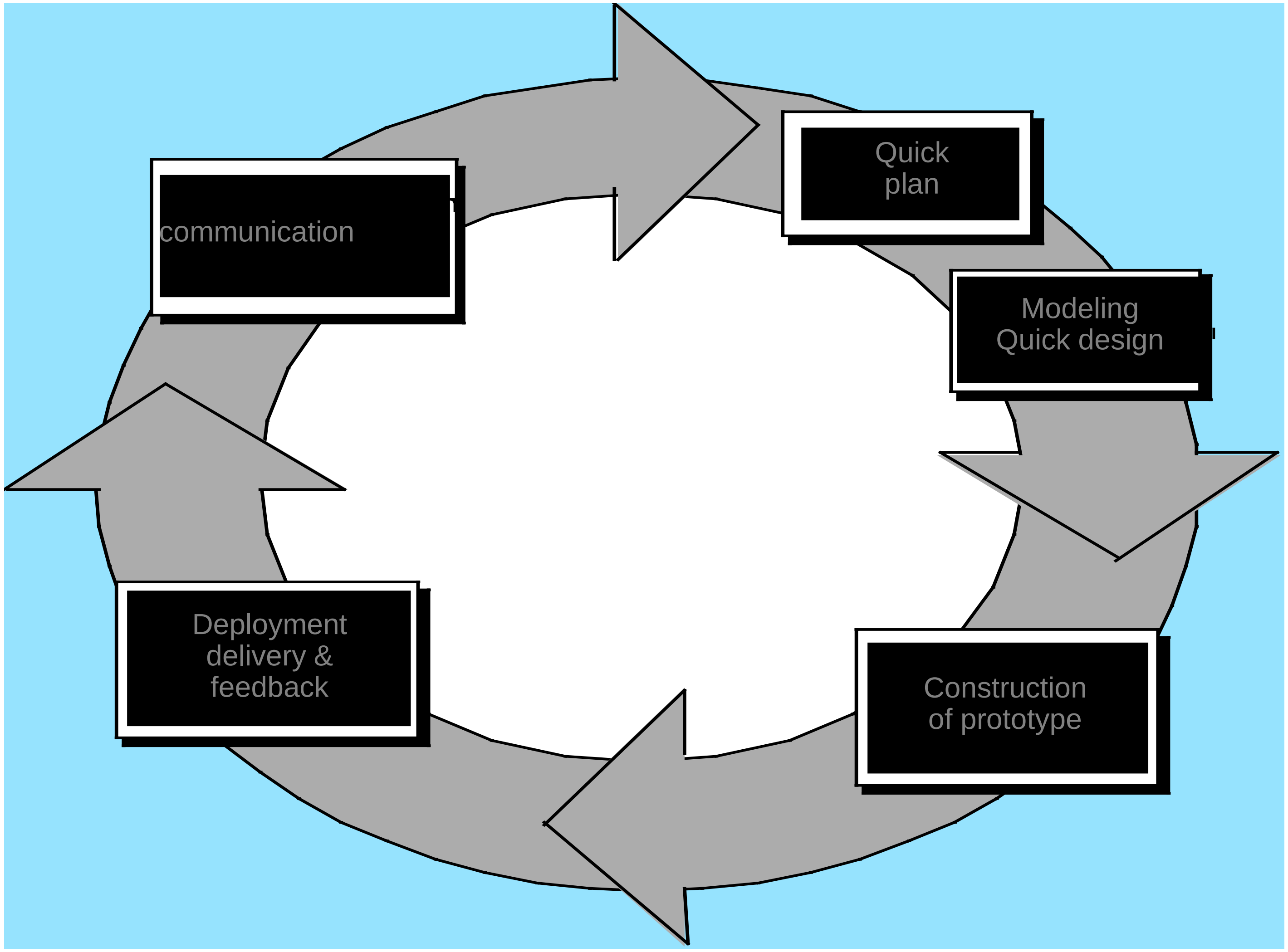
Incremental Model

- The incremental model prescribes *developing* and *delivering* the product in *planned increments* (or builds). = "Staged Delivery" model
  - The product is *designed* to be *delivered* in *increments*.
  - Each increments provides (in theory) more functionality than the previous increment.
- User requirements are prioritized and the highest priority requirements are included in early increments.
- Once the development of an increment is started, the requirements are frozen though requirements for later increments can continue to evolve.
- How is this different from Evolutionary?

The Incremental Model



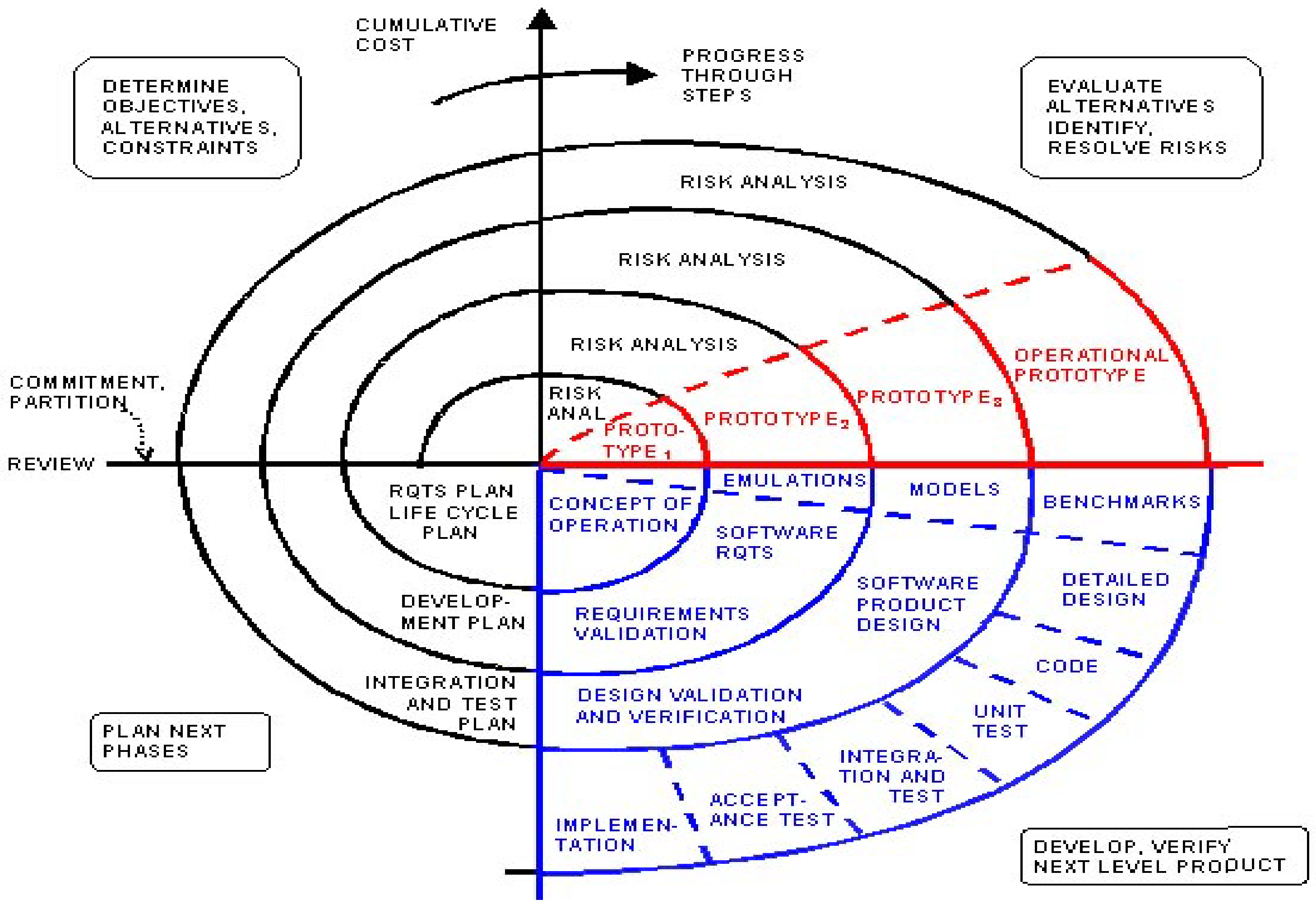
Evolutionary Models: Prototyping



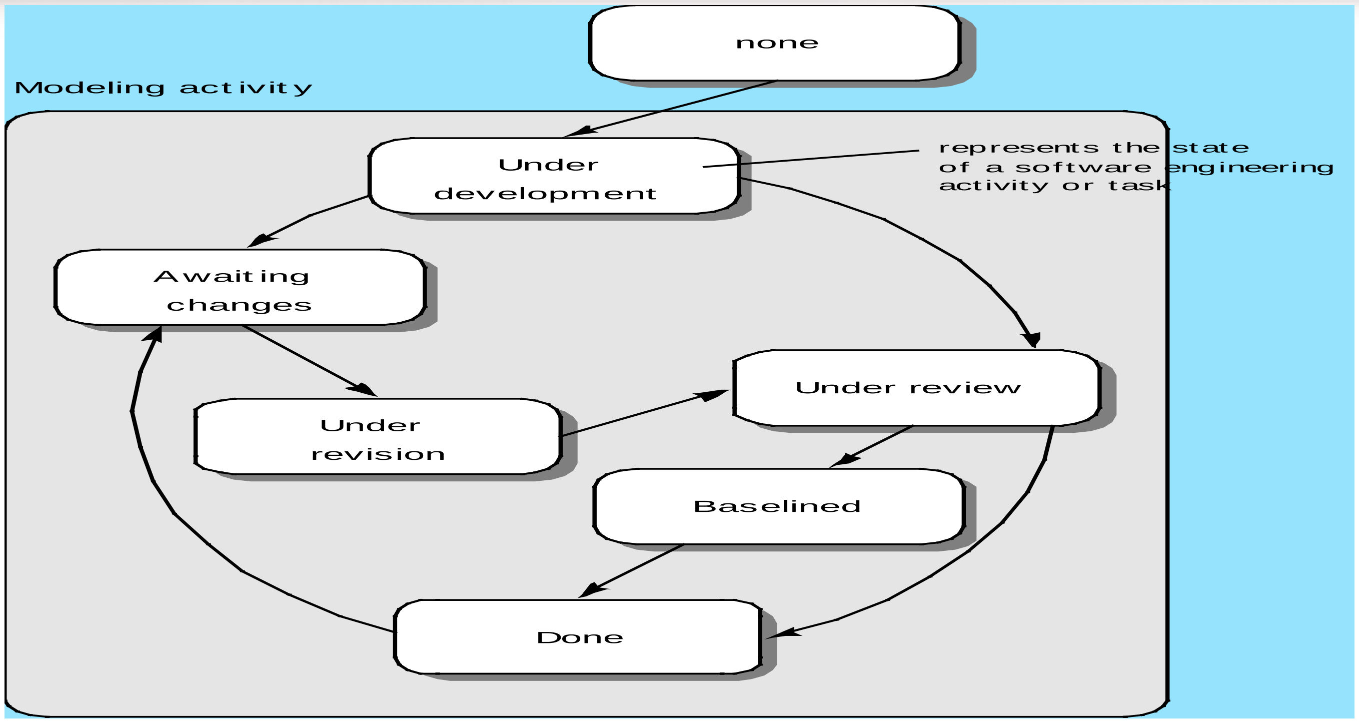
Spiral Model

- First defined by Barry Boehm
- Combines elements of:
  - evolutionary, incremental, and prototyping models
- First model to explain
  - why iteration matters
  - How iteration could be used effectively
- The term "*spiral* " refers to successive iterations outward from a central starting point.

Evolutionary Models: The Spiral



Evolutionary Models: Concurrent

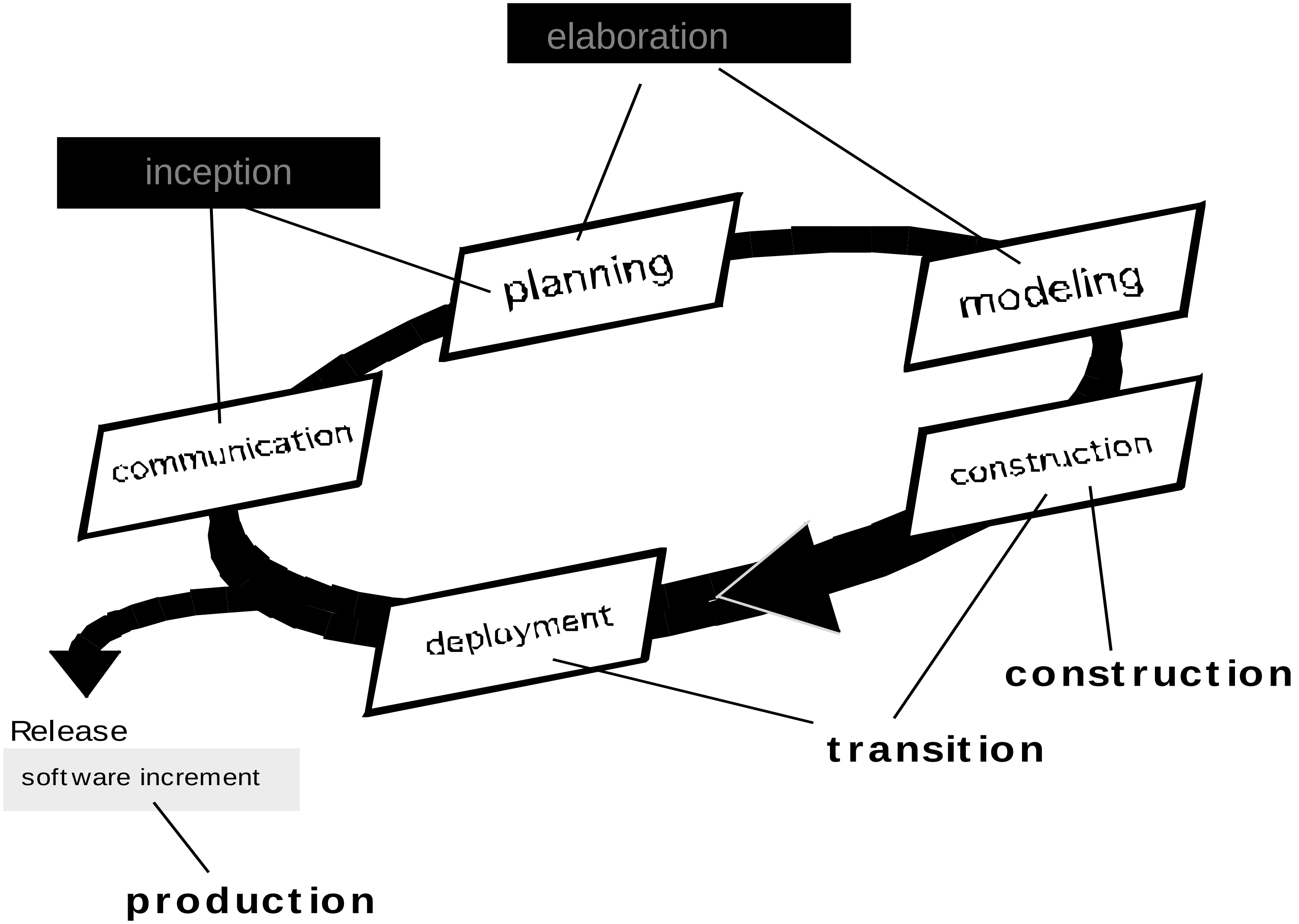


- Complementary applications
  - High Interdependence with Modules
- State Charts
- Triggers for transition
- Examples : Client – Server

Still Other Process Models

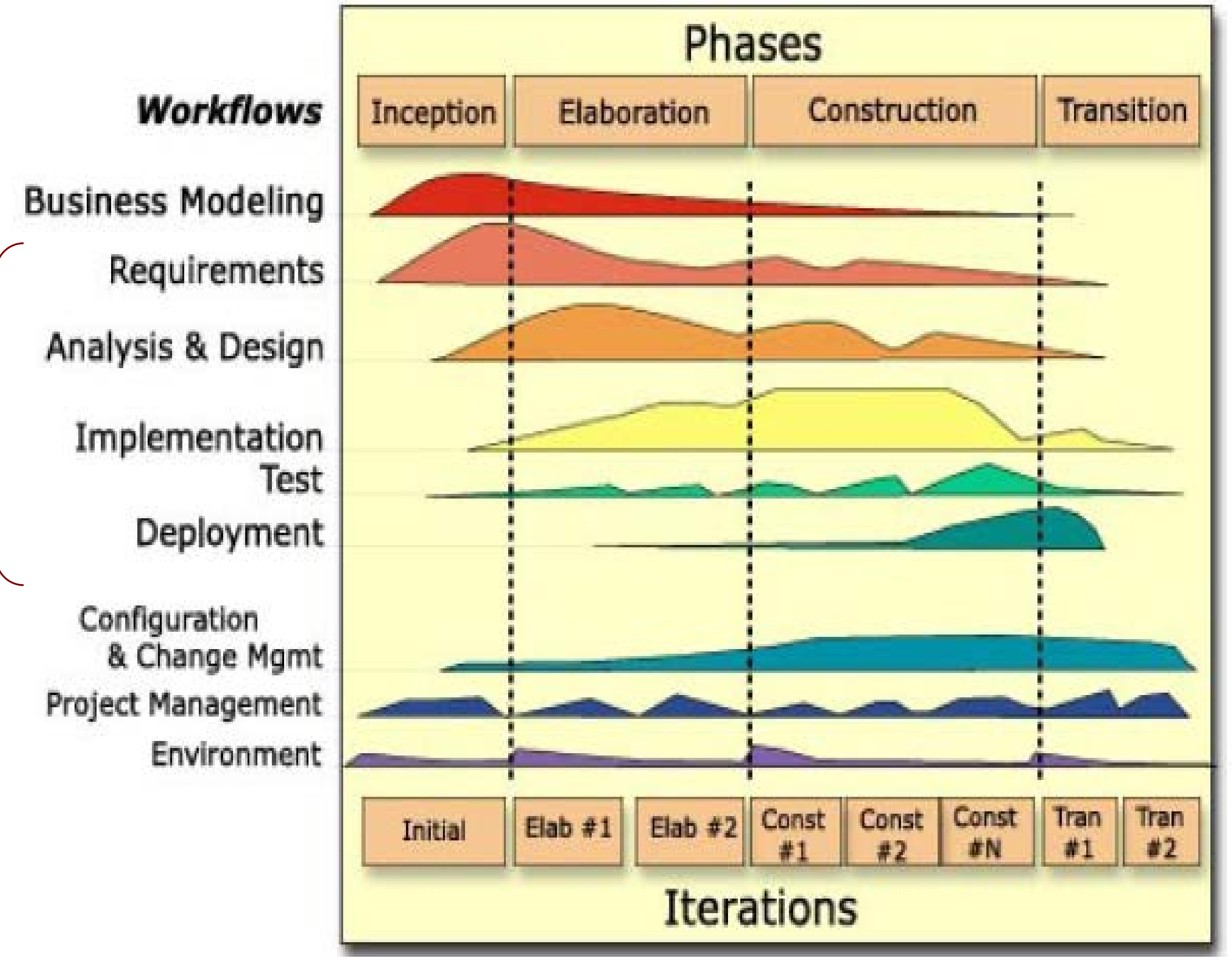
- Component based development—the process to apply when reuse is a development objective
- Formal methods—emphasizes the mathematical specification of requirements
- AOSD—provides a process and methodological approach for defining, specifying, designing, and constructing *aspects*
- Unified Process—a "use-case driven, architecture-centric, iterative and incremental" software process closely aligned with the Unified Modeling Language (UML)

The Unified Process (UP)



The process outline - Phases versus Iterations

Core of almost any process
What Unified Process spans



UP Work Products

Inception phase

Vision document
Initial use-case model
Initial project glossary
Initial business case
Initial risk assessment.
Project plan,
  phases and iterations.
Business model,
  if necessary.
One or more prototypes

Elaboration phase

Use-case model
Supplementary requirements
  including non-functional
Analysis model
Software architecture
  Description.
Executable architectural
  prototype.
Preliminary design model
Revised risk list
Project plan including
  iteration plan
  adapted workflows
  milestones
  technical work products
Preliminary user manual

Construction phase

Design model
Software components
Integrated software
  increment
Test plan and procedure
Test cases
Support documentation
  user manuals
  installation manuals
  description of current
    increment

Transition phase

Delivered software increment
Beta test reports
General user feedback

Personal Software Process (PSP)

- **Planning.** This activity isolates requirements and develops both size and resource estimates. In addition, a defect estimate (the number of defects projected for the work) is made. All metrics are recorded on worksheets or templates. Finally, development tasks are identified and a project schedule is created.
- **High-level design.** External specifications for each component to be constructed are developed and a component design is created. Prototypes are built when uncertainty exists. All issues are recorded and tracked.
- **High-level design review.** Formal verification methods (Chapter 21) are applied to uncover errors in the design. Metrics are maintained for all important tasks and work results.
- **Development.** The component level design is refined and reviewed. Code is generated, reviewed, compiled, and tested. Metrics are maintained for all important tasks and work results.
- **Postmortem.** Using the measures and metrics collected (this is a substantial amount of data that should be analyzed statistically), the effectiveness of the process is determined. Measures and metrics should provide guidance for modifying the process to improve its effectiveness.

Team Software Process (TSP)

- Build self-directed teams that plan and track their work, establish goals, and own their processes and plans. These can be pure software teams or integrated product teams (IPT) of three to about 20 engineers.
- Show managers how to coach and motivate their teams and how to help them sustain peak performance.
- Accelerate software process improvement by making CMM Level 5 behavior normal and expected.
  - The Capability Maturity Model (CMM), a measure of the effectiveness of a software process, is discussed in Chapter 30.
- Provide improvement guidance to high-maturity organizations.
- Facilitate university teaching of industrial-grade team skills.

Team Software Process (TSP) Review

- Builds on PSP (prerequisite)
  - Prevent defects earlier
- Defined framework - Scripted
  - Cyclic, iterative
  - Standard measures for quality and performance
  - Precise data collection- on almost everything
  - Established Roles
  - Discipline and guidance
  - Post Mortem each cycle – find the problems
- CMM

Team Software Process (TSP)

(Every Cycle Produces an Artifact)

A TSP Cycle:

1- Strategy

2- Plan

3- Requirements

4- Design

5- Implementation

6- Test

7- Postmortem

Team Software Process
Document Driven

- The "Good" of TSP
  - Structured
  - Scripted
  - Highly defined
  - New teams, or ones with no structure
  - SEI has data on about 20 projects
- The "other"
  - Possible problems with change/flexibility
  - Requirements?

TSP Center Piece

Data Collection

- Only source of “bugs” is humans
- Developers on the whole are unstructured and tend towards “hacking”
- Need to identify source and mitigate problem

Q & A