Software Process
(*process models*)

Categories of Process Models

- Prescriptive model (traditional model)
  - Sequential model
  - Incremental model
  - Evolutionary model
- Specialized process model
  - Component-based model
  - Formal method model
  - Aspect-Oriented model
- Unified process model
  - Software development with UML

Specialized Process Models

Specialized Process Models

- Designed for specific software engineering approach.
- May incorporate characteristics from one/more traditional (prescriptive) models.
- Examples:
  - Component-based Development Model
  - Formal Methods Model
  - Aspect-Oriented Software Development Model

In some cases, these specialized process models might better be characterized as a collection of

techniques or a “methodology” for accomplishing a specific software development goal. However,

they do imply a process.

Component Based Development Model

- Applied when reuse is a development objective
  - Underlying technology is component-based design.
- Comprises applications from prepackaged software components
  - COTS (Commercial off-the-self) software components
- Sharing the characteristics of other models
  - Iterative (evolutionary: spiral)
- Modeling and construction activities begin with the identification of candidate components

the component-based development (CBD) model incorporates many of the iterative characteristics of the spiral model. The main difference is that in CBD the emphasis is on composing solutions from prepackaged software components or classes. This CBD emphasizes software reusability. Further discussion of CBD can be found in Chapter 10.

Formal Method Model

- Emphasize the mathematical specification of requirements
  - Underlying technology is formal specification
- Offers the promise of *defect-free* software
  - Examples: aircraft avionics and medical devices
- Specify, develop and verify a computer-based system by applying a *rigorous mathematical notation*
- Ambiguity, incompleteness, and inconsistency can be *discovered and corrected more easily* through the application of mathematic analysis
- Problems
  - Expensive to develop and apply (needs extensive training)
  - Difficult to communicate with team members and customers

It is difficult to use the models as a communication mechanism for technically unsophisticated customers

Aspect-oriented Software Development

- Concerns
  - High-level properties of a system (E.g. security and fault tolerance) that span the entire software architecture
- Crosscutting concerns
  - Concerns that cut cross multiple system functions, features, and information
- Aspects
  - Mechanisms for localizing the expression of a crosscutting concern
- AOSD/AOP process model
  - provides a methodological approach for defining, specifying, designing, and constructing *aspects*
  - Likely adopt characteristics of both evolutionary and concurrent process models

A standalone AOSD process model has not matured yet, so only extrapolation here.

Unified Process

Unified Process

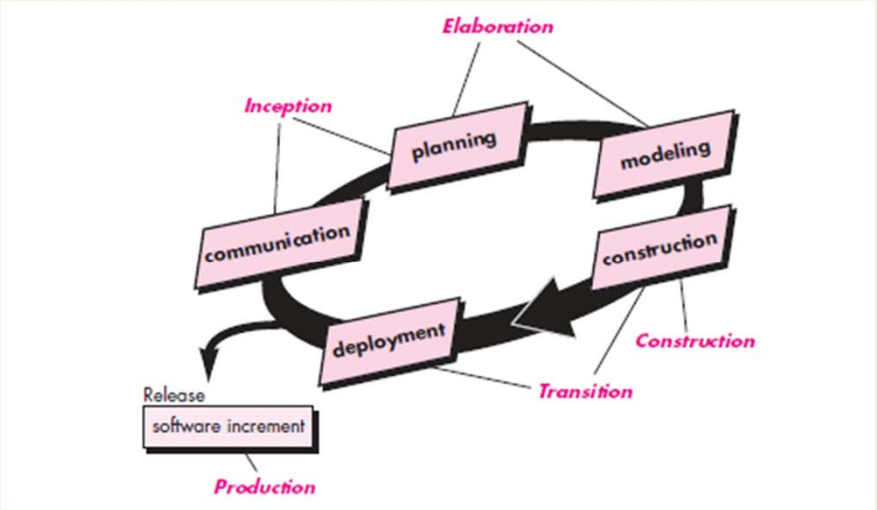
- A “use-case driven, architecture-centric, iterative and incremental” software process
  - Draw on the best features and characteristics of traditional software process models
  - Implements many of the best principles of agile software development
- Key features:
  - Recognizes the importance of *customer communication*
  - Emphasizes the important role of *software architecture*
  - Helps the architect focus on the right goals, such as *understandability, reliance to future changes, and reuse*
  - iterative and incremental, providing the *evolutionary feel* that is essential in modern software development

Historically, this process model has a very close connection with UML (a unified modeling language):

Contains a robust notation for the modeling and development of object-oriented systems

By 1997, UML became a de facto industry standard for object-oriented software development

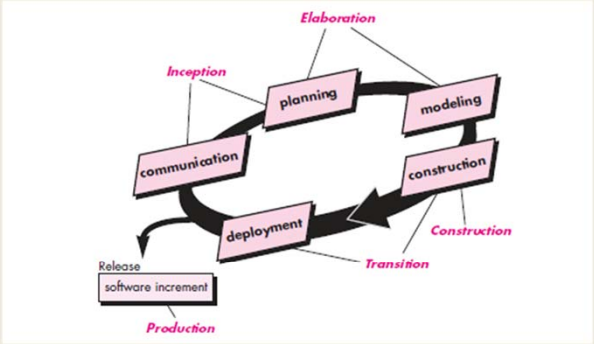
The Unified Process



- Every iteration - identify use cases, create a design, implement the design
- Every iteration is a complete development process

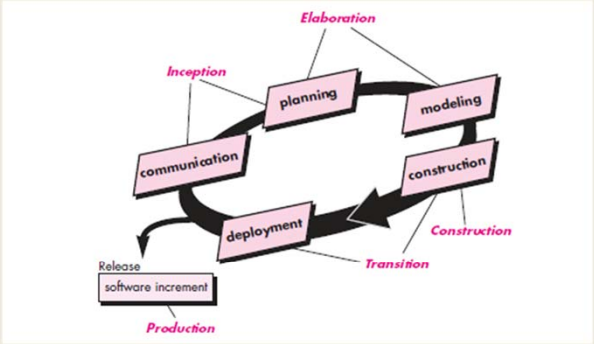
UP *phases* are similar in intent to the generic framework activities defined in this book.

Unified Process: Phases



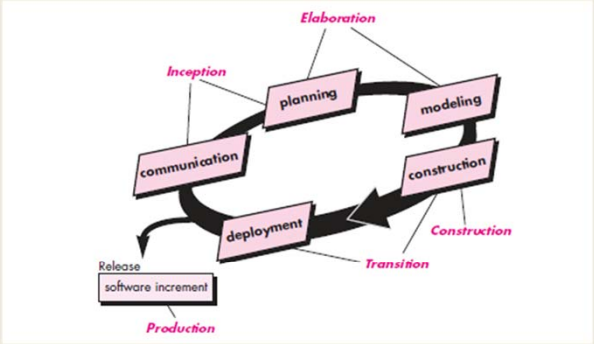
- Inception Phase
  - Encompasses both customer communication and planning activities
    - Business requirements are described through a set of preliminary use case
  - Describe which features and functions each major class of users desires

Unified Process: Phases



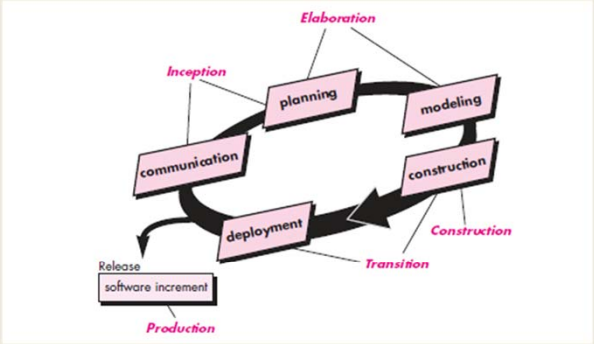
- Elaboration phase
  - Encompasses the planning and modeling activities
  - Refines and expands the preliminary use cases
  - Expands the architectural representation to include five different views of the software(the use case model, the analysis model, …)

Unified Process: Phases

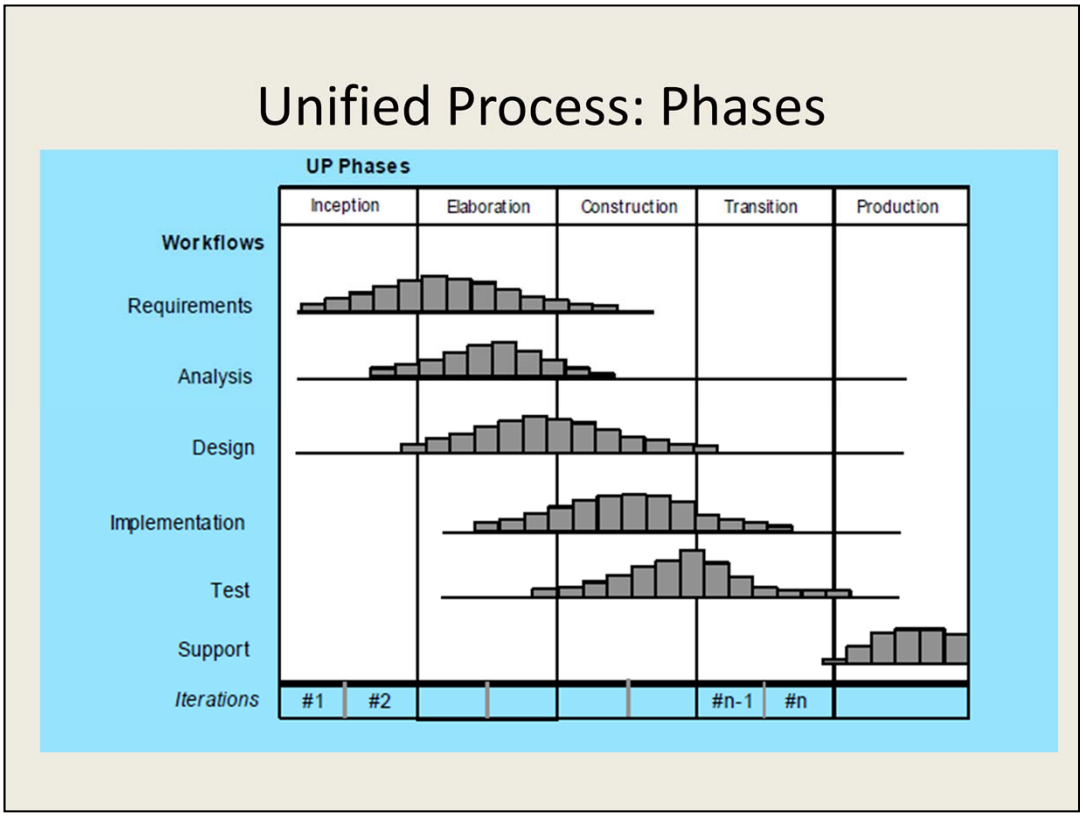


- Construction Phase
  - Implement design / construction activity in generic process
- Transition Phase
  - Encompasses the latter stages of the generic construction activity and the first part of generic deployment activity
  - Software is given to end uses for beta testing
  - User feedback reports both defects and necessary changes

Unified Process: Phases



- Production Phase
  - Coincides with the development activity of generic process
  - The ongoing use of the software is monitored
  - Defect reports and requests for changes are submitted and evaluated



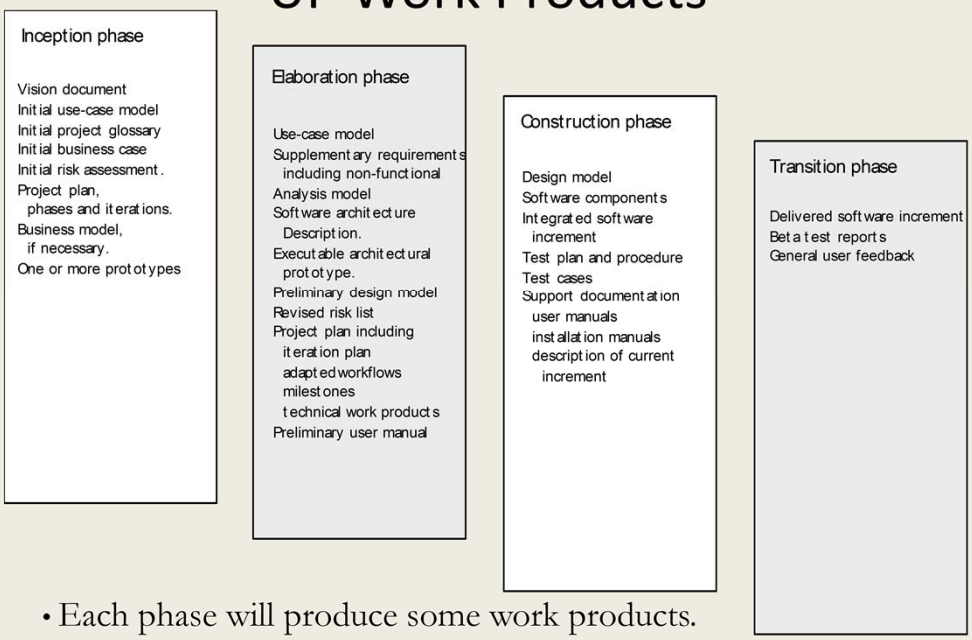
It is likely that at the same time the construction, transition, and production phases are being conducted, work may have already begun on the next software increment. This means that the five UP phases do not occur in a sequence, but rather

with staggered concurrency.

A software engineering workflow is distributed across all UP phases. In the context of UP, a *workflow* is analogous to a task set. That is, a workflow identifies the tasks required to accomplish an important software

engineering action and the work products that are produced as a consequence of successfully completing the tasks.

UP Work Products



- Each phase will produce some work products.
- Each iteration will refine these work products.

Personal/Team Process

In an ideal setting, you would create a process that best fits your needs, and at the same time, meets the broader needs of the team and the organization

Alternatively, the team itself can create its own process, and at the same time meet the narrower needs

of individuals and the broader needs of the organization.

The best processes are those close to the people who will be doing the work.

Personal and team processes are developed to fit individual and team’s need, while still fulfill the project goal.

Personal Software Process (PSP)

- Emphasizes the need to record and analyze the types of errors you make, thus to develop strategies to eliminate them
- Defines five framework activities
  - Planning
    - Estimate resource and decide schedule.
  - High-level design
    - Architecture and component design.
  - High-level design review
    - Peer-review or other formal review shall be performed.
  - Development
    - Generate/review/compile/test code
  - Postmortem
    - Measure/assess process effectiveness, prepare for process improvement

PSP stresses the need to identify errors early and, just as important, to understand

the types of errors that you are likely to make.

When PSP is properly introduced to software engineers, the resulting improvement in software engineering productivity and software quality are significant

Team Software Process (TSP)

- Goal
  - to build self-directed software development teams.
- Defines five framework activities
  - Project launch
  - High-level design
  - Implementation
  - Integration and test
  - Postmortem
- Approaches
  - use a series of scripts, forms, and standards to guide team members in their works.
    - E.g. each project is “launched” using a “script” that defines the tasks to be accomplished

many industry-grade software projects are addressed by a team of practitioners,

Watts Humphrey extended the lessons learned from the introduction of PSP

and proposed a *Team Software Process* (TSP).

Summary

- Generic process model / process framework
  - Framework Activities
  - Umbrella Activities
  - Process Flow
- Process Models
  - Prescriptive (traditional) models
  - Specialized models
  - Unified model
  - PSP/TSP
- Key: relationships, differences, pros/cons.
  - Be able to choose a model, and justify the choice