REKAYASA PERANGKAT LUNAK

Lecturer Team for Even Semester Year 2019-2020:

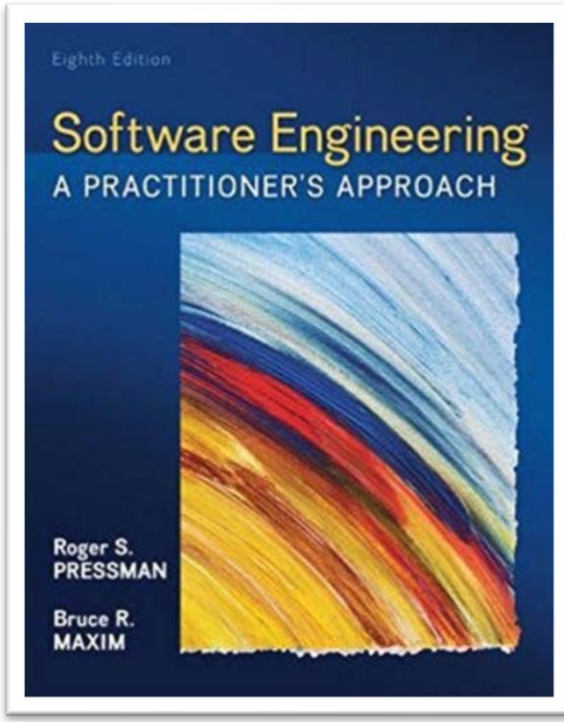
Hetti Hidayati

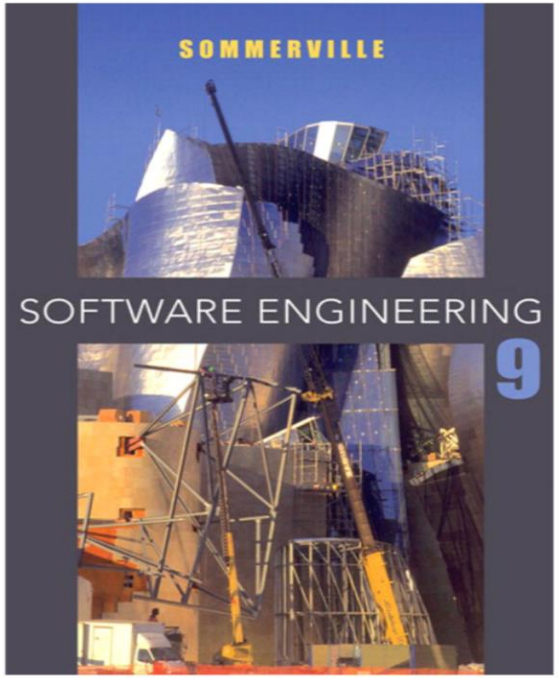
Reza Budiawan

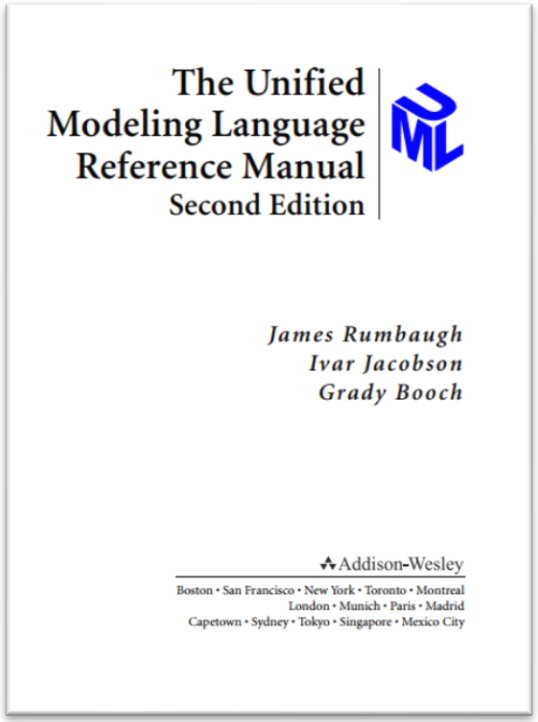
Rules

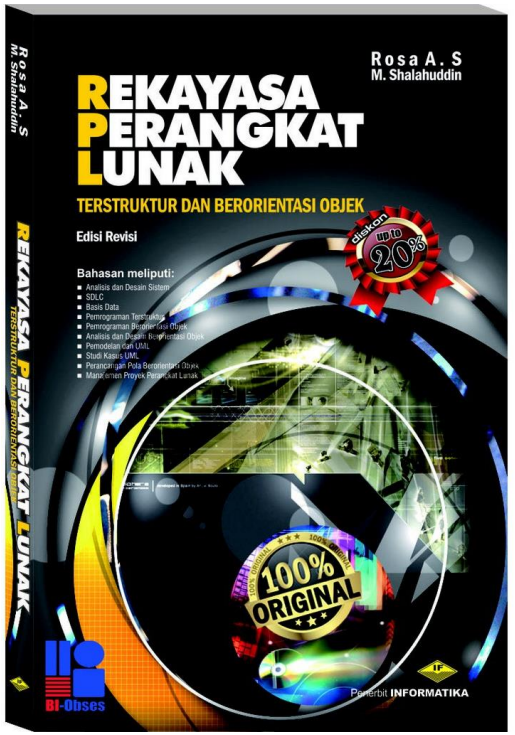
1. Presence's Tapping:
   a) At the beginning
   b) After 1 hour of the class
2. Tardiness tolerance: 15 minutes
3. Uniform rule: based on Tel-U's rule about uniform
4. Total minimal presences: 75%
5. There is a quiz for every meeting except for assessment or presentation schedule's meeting.
6. **All acts of cheating will get an E grade**

RESOURCES


Eighth Edition
Software Engineering
A PRACTITIONER'S APPROACH
Roger S. PRESSMAN
Bruce R. MAXIM


SOMMERVILLE
SOFTWARE ENGINEERING
9


The Unified Modeling Language Reference Manual
Second Edition
James Rumbaugh
Ivar Jacobson
Grady Booch
Addison-Wesley
Boston • San Francisco • New York • Toronto • Montreal
London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City


Rosa A. S
M. Shalahuddin
REKAYASA PERANGKAT LUNAK
TERSTRUKTUR DAN BERORIENTASI OBJEK
Edisi Revisi
Bahasan meliputi:
100% ORIGINAL
Penerbit INFORMATIKA

This Presentation Covers:

1. Software Process Structure
   a) Generic Process
   b) Process Flow
2. Process Model
   a) Prescriptive Process Models: Waterfall, Incremental, Evolutionary (Prototyping & Spiral), concurrent models
   b) Specialized Process Models: Component-based development, The Formal Method Model, Aspect-Oriented Software Development,
   c) The Unified Process
   d) Product & Process

SOFTWARE PROCESS STRUCTURE

Software Engineering

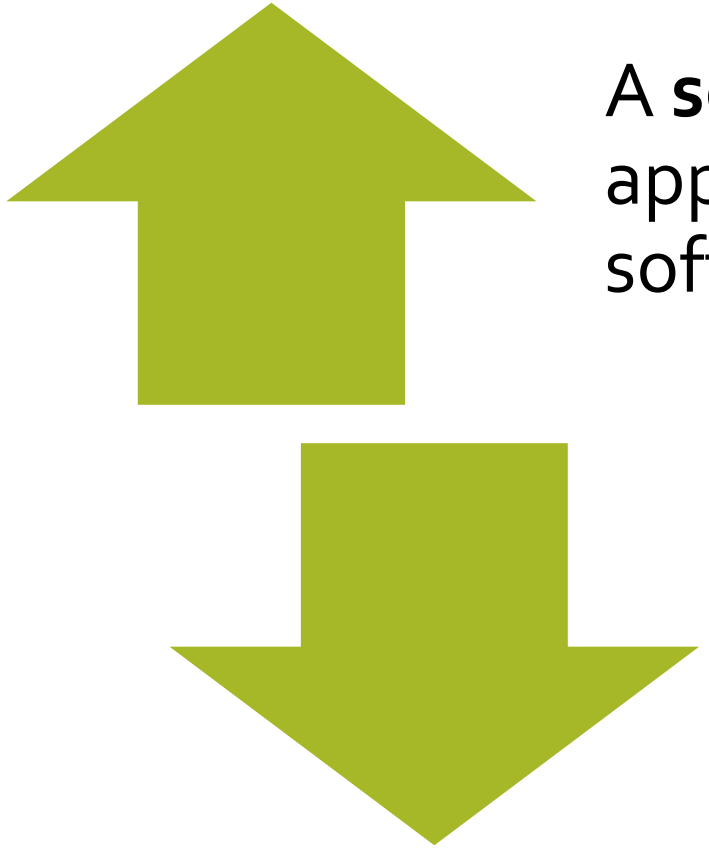
This Presentation Covers:

1. Software Process Structure
   a) Generic Process
   b) Process Flow
2. Process Model
   a) Prescriptive Process Models: Waterfall, Incremental, Evolutionary (Prototyping & Spiral), concurrent models
   b) Specialized Process Models: Component-based development, The Formal Method Model, Aspect-Oriented Software Development,
   c) The Unified Process
   d) Product & Process

Definition

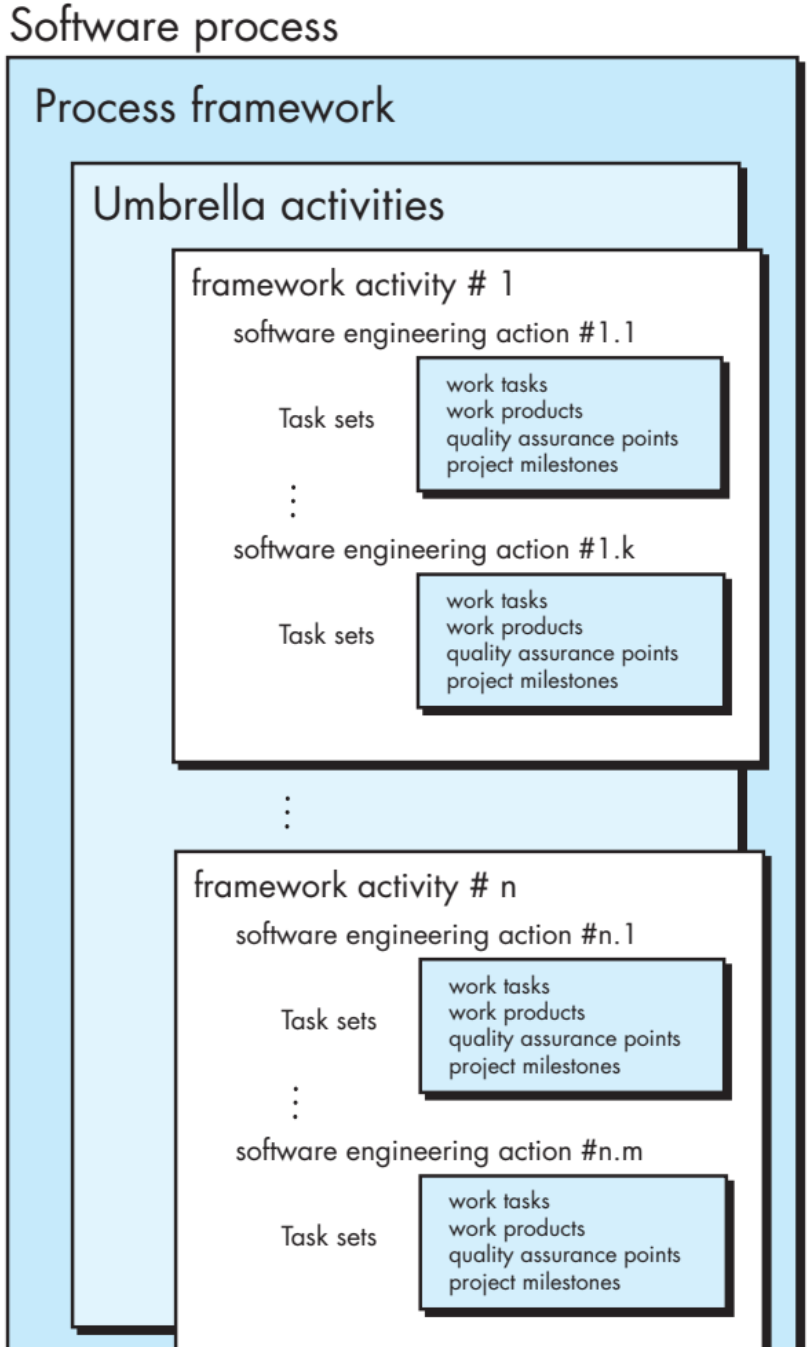
- **What is it?** a road map that helps you create a timely, high-quality result. The road map that you follow is called a "software process."
- **Who does it?** Software engineers and their managers adapt the process to their needs and then follow it.
- **Why is it important?** Because it provides stability, control, and organization to an activity that can, if left uncontrolled, become quite chaotic.
- **What are the steps?** At a detailed level, the process that you adopt depends on the software that you're building.
- **What is the work product?** Programs, documents, and data that are produced
- **How do I ensure that I've done it right?** There are a number of software process assessment. Quality, timeliness, and long-term viability of the product you build are the best indicators of the efficacy of the process that you use.

Process VS Software Engineering



A **software process** defines the approach that is taken as software is engineered

Software engineering also encompasses technologies that populate the process—technical methods and automated tools



SOFTWARE PROCESS

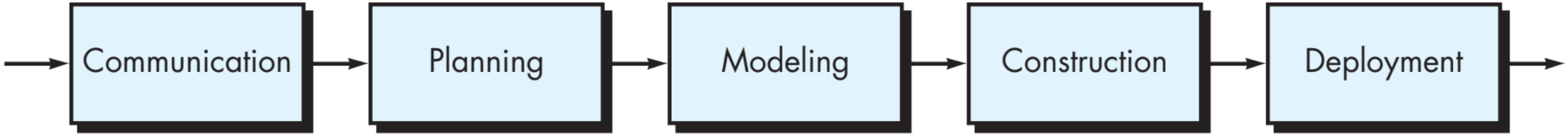
- Each framework activity is populated by a set of **software engineering actions**.
- Each software engineering action is defined by a task set that identifies the work tasks that are to be completed, the work products that will be produced, the quality assurance points that will be required, and the milestones that will be used to indicate progress.

Generic Process

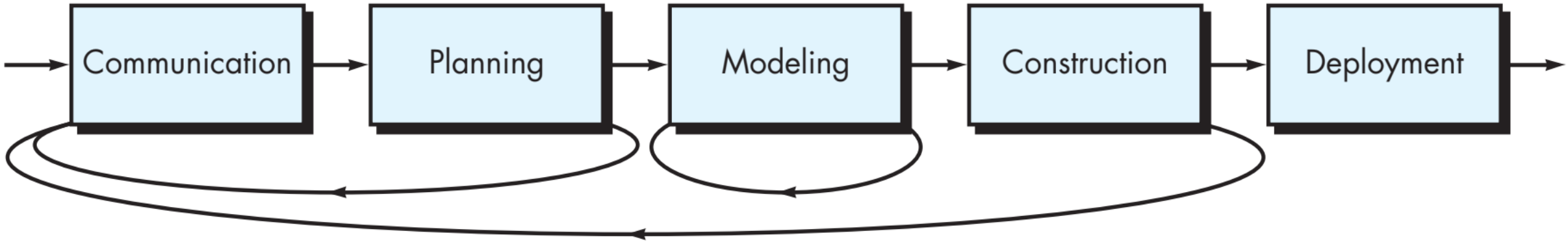
- A generic **process framework** for software engineering defines five framework activities—communication, planning, modeling, construction, and deployment.
- Besides, it supports by a set of **umbrella activities**—project tracking and control, risk management, quality assurance, configuration management, technical reviews.
- One important aspect of the software process has not yet been discussed: ***process flow***.

Process Flow

- Describes **how the framework activities** and the actions and tasks that **occur** within each framework activity are organized with respect to sequence and time.
- Categories:
  - A **linear** process flow
  - An **iterative** process flow
  - An **evolutionary** process flow
  - A **parallel** process flow

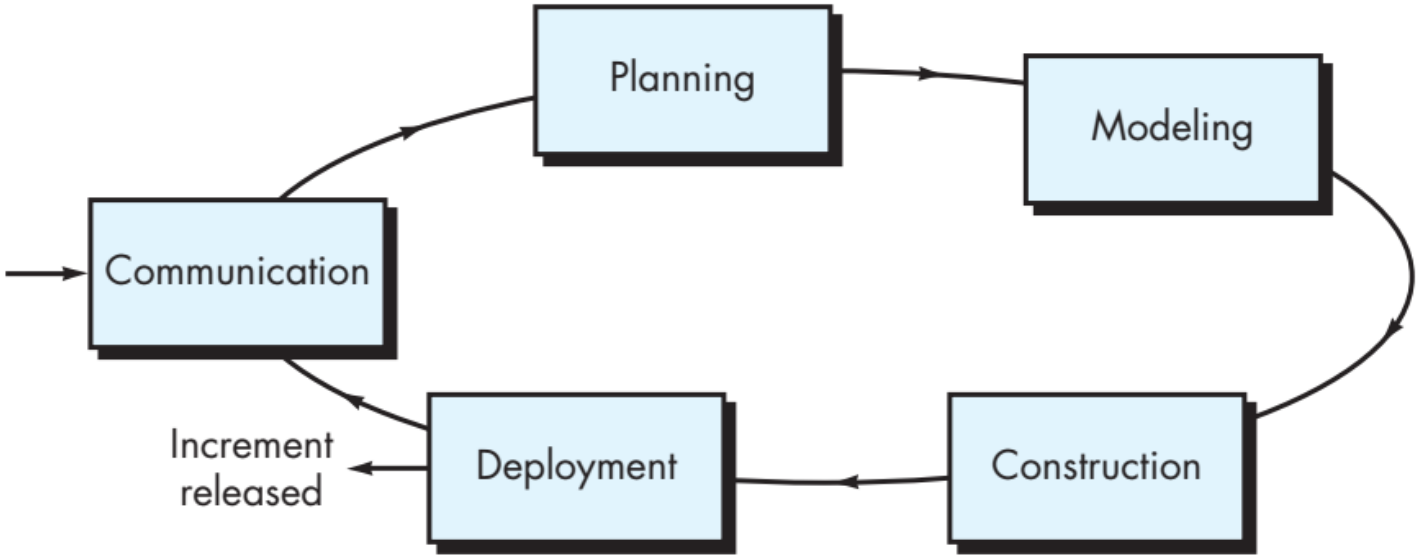


(a) Linear process flow

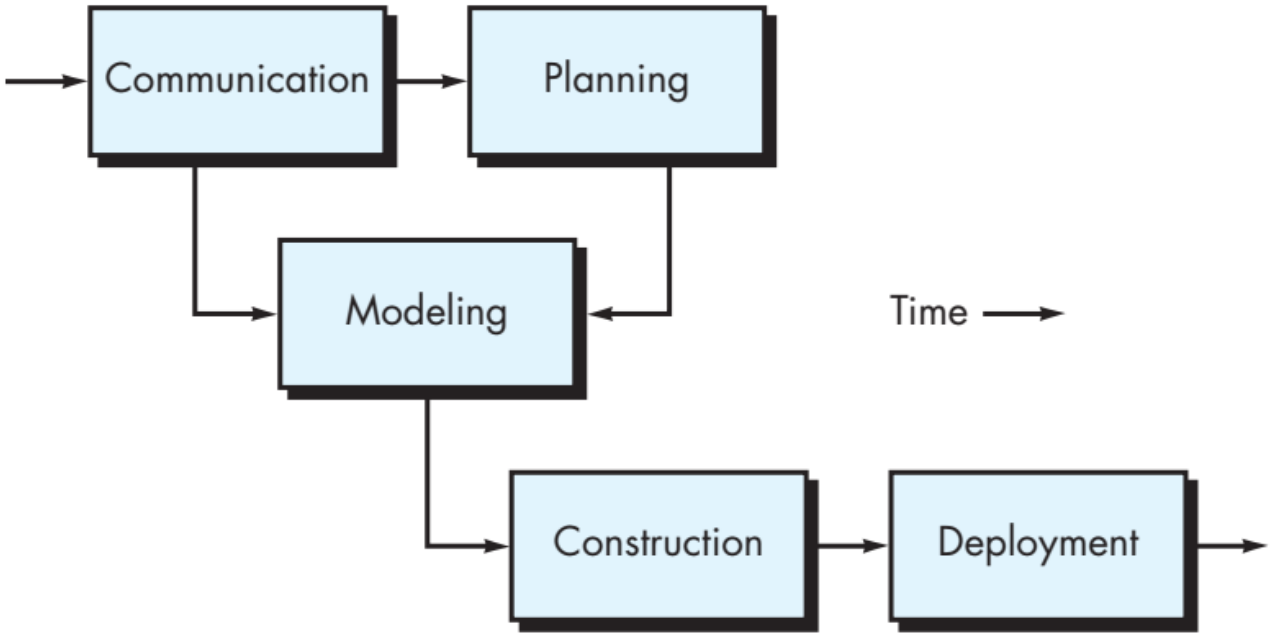


(b) Iterative process flow

PROCESS FLOW



(c) Evolutionary process flow



(d) Parallel process flow

PROCESS FLOW

Process Pattern

- 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.
- A process pattern provides you with a template—a consistent method for describing problem solutions within the context of the software process.

Process Pattern Content

- Pattern Name
- Intent
- Initial Context
- Problem
- Solution
- Resulting Context
- Related Pattern
- Known Uses & Examples

Pattern Name. **RequirementsUnclear**

Intent. This pattern describes an approach for building a model (a prototype) that can be assessed iteratively by stakeholders in an effort to identify or solidify software requirements.

Type. Phase pattern.

Initial Context. The following conditions must be met prior to the initiation of this pattern: (1) stakeholders have been identified; (2) a mode of communication between stakeholders and the software team has been established; (3) the overriding software problem to be solved has been identified by stakeholders; (4) an initial understanding of project scope, basic business requirements, and project constraints has been developed.

Problem. Requirements are hazy or nonexistent, yet there is clear recognition that there is a problem to be solved, and the problem must be addressed with a software solution. Stakeholders are unsure of what they want; that is, they cannot describe software requirements in any detail.

Solution. A description of the prototyping process would be presented here and is described later in Section 4.1.3.

Resulting Context. A software prototype that identifies basic requirements (e.g., modes of interaction, computational features, processing functions) is approved by stakeholders. Following this, (1) the prototype may evolve through a series of increments to become the production software or (2) the prototype may be discarded and the production software built using some other process pattern.

Related Patterns. The following patterns are related to this pattern: **CustomerCommunication, IterativeDesign, IterativeDevelopment, CustomerAssessment, RequirementExtraction.**

Known Uses and Examples. Prototyping is recommended when requirements are uncertain.

PROCESS MODEL

This Presentation Covers:

1. Software Process Structure
   a) Generic Process
   b) Process Flow
2. Process Model
   a) Prescriptive Process Models: Waterfall, Incremental, Evolutionary (Prototyping & Spiral), concurrent models
   b) Specialized Process Models: Component-based development, The Formal Method Model, Aspect-Oriented Software Development,
   c) The Unified Process
   d) Product & Process

Definition

- **What is it?** A process model provides a specific roadmap for software engineering work.
- **Who does it?** Software engineers and their managers adapt a process model to their needs and then follow it.
- **Why is it important?** Because process provides stability, control, and organization to an activity that can, if left uncontrolled, become quite chaotic.
- **What are the steps?** The process model provides you with the "steps" you'll need to perform disciplined software engineering work.
- **What is the work product?** A customized description of the activities and tasks defined by the process.

PROCESS MODEL

Prescriptive Process Model

This Presentation Covers:

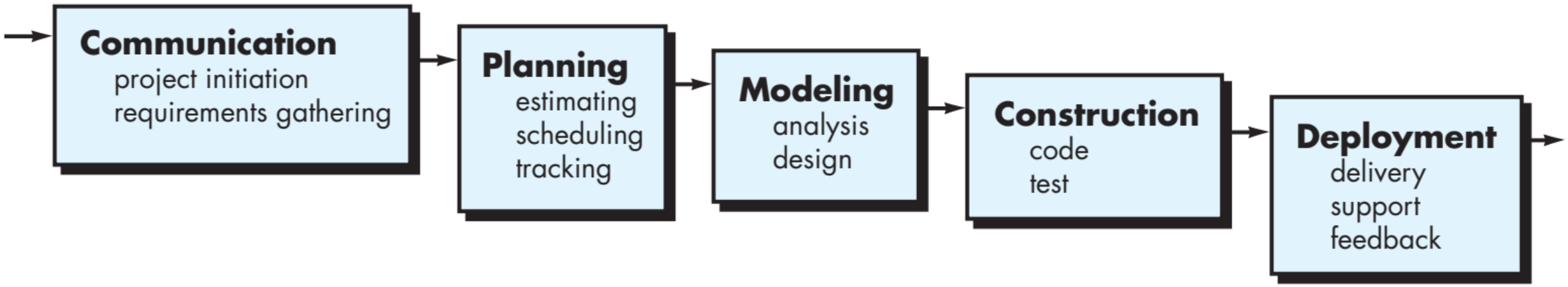
1. Software Process Structure
   a) Generic Process
   b) Process Flow
2. Process Model
   a) Prescriptive Process Models: Waterfall, Incremental, Evolutionary (Prototyping & Spiral), concurrent models
   b) Specialized Process Models: Component-based development, The Formal Method Model, Aspect-Oriented Software Development,
   c) The Unified Process
   d) Product & Process

PRESCRIPTIVE PROCESS MODELS

- A prescriptive process model strives for structure and order in software development.
- Activities and tasks occur sequentially with defined guidelines for progress.
- It is called "prescriptive" because they **prescribe a set of process elements**—framework activities, software engineering actions, tasks, work products, quality assurance and change control mechanisms for each project.

The Waterfall Model

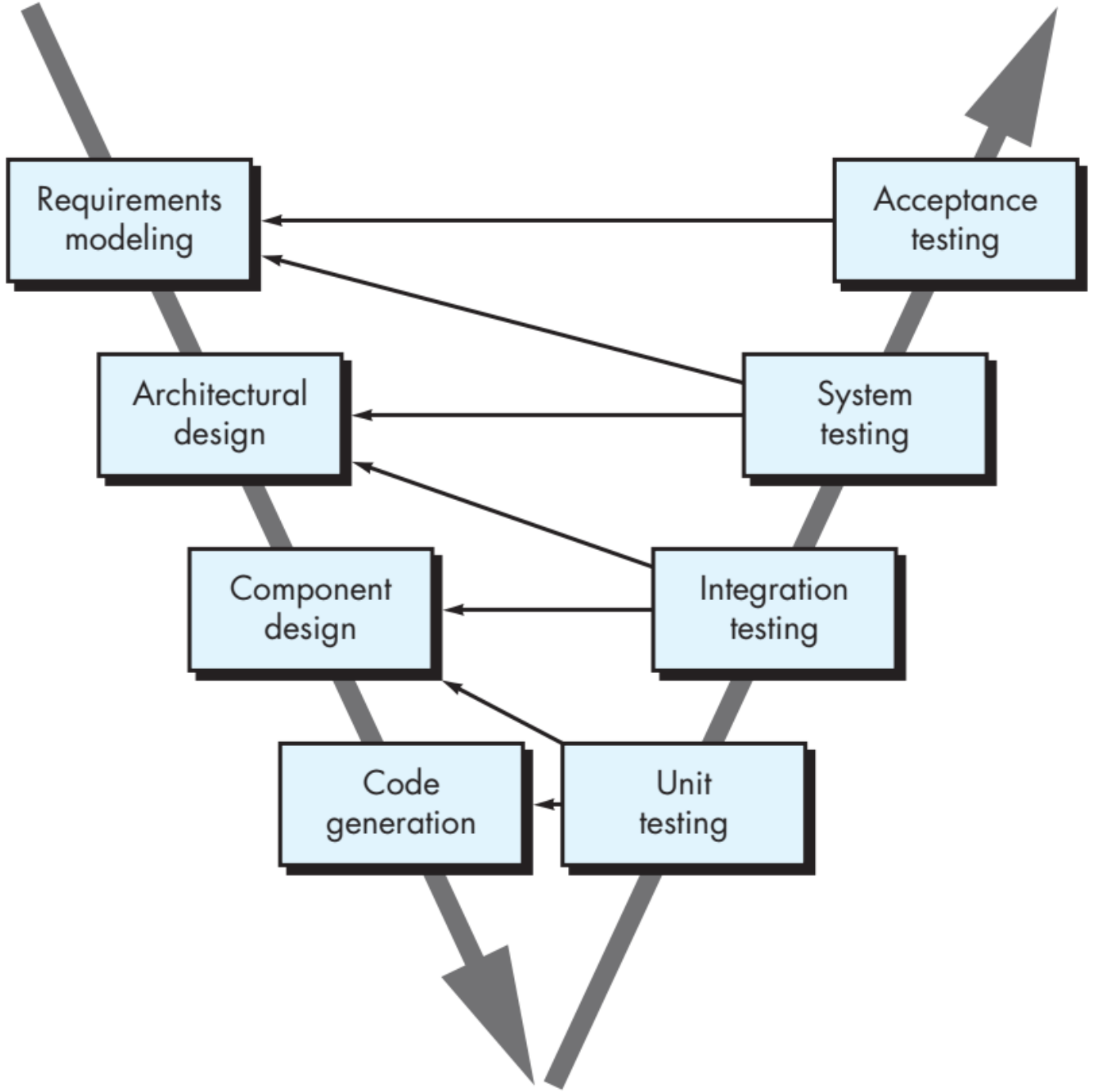
- The waterfall model, sometimes called the classic life cycle, suggests a **systematic, sequential approach** to software development that begins with customer specification of requirements and progresses through planning, modeling, construction, and deployment, culminating in ongoing support of the completed software.
- It is applied when the requirements for a problem are well understood—work flows from communication through deployment in a reasonably linear.



The Waterfall Model Variant: **V Model**

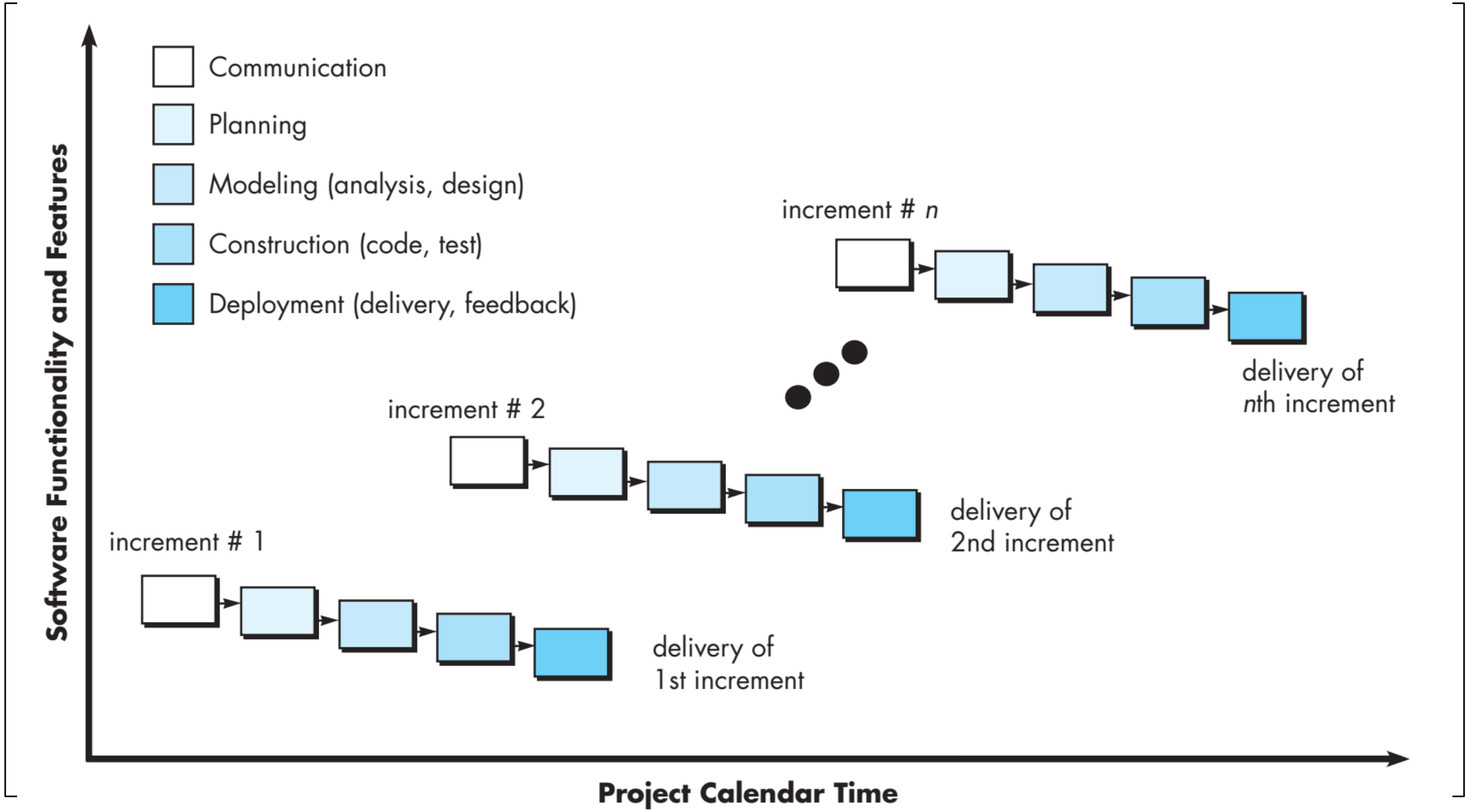
There is **no fundamental difference between the classic life cycle and the V-model.**

The V-model provides a way of visualizing how verification and validation actions are applied to earlier engineering work.

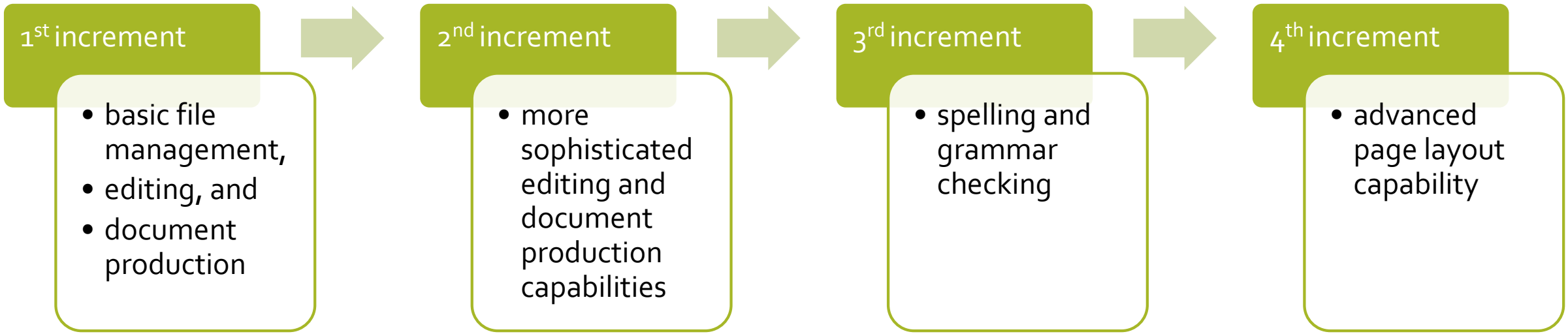


Incremental Process Models

- The incremental model combines the elements' linear and parallel process flows.
- The incremental model applies linear sequences in a staggered fashion as calendar time progresses
- Condition: there may be a compelling need to **provide a limited set of software functionality** to users quickly and **then refine and expand** on that functionality in later software releases.


Communication
Planning
Modeling (analysis, design)
Construction (code, test)
Deployment (delivery, feedback)
increment # n
delivery of
nth increment
increment # 2
delivery of
2nd increment
increment # 1
delivery of
1st increment
Software Functionality and Features
Project Calendar Time

Incremental Process Models Example



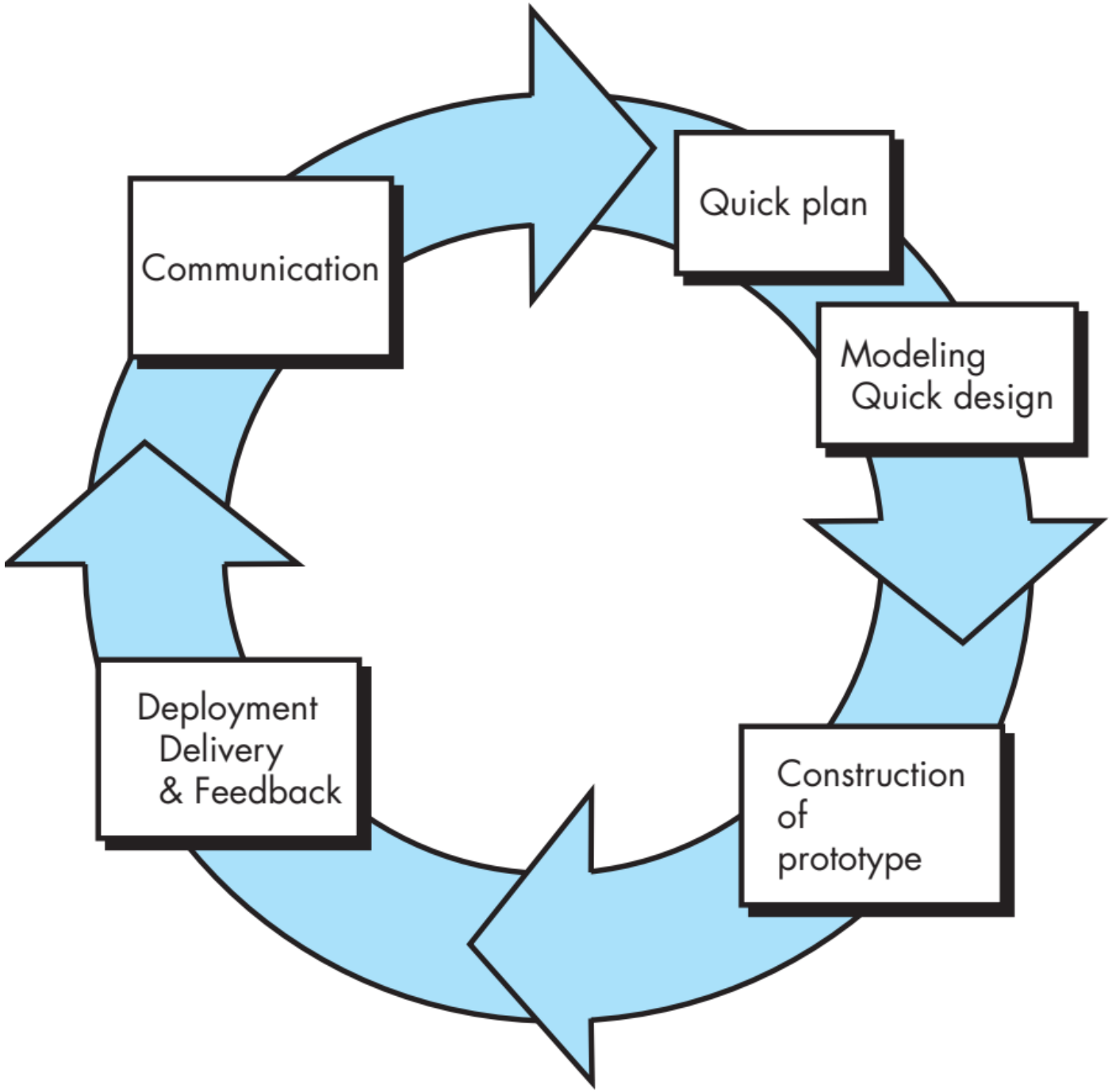
Word Processing App

Evolutionary Process Models

- Evolutionary models are **iterative**. They are characterized in a manner that enables you to develop increasingly more complete versions of the software.
- It applied for condition: tight market deadlines make completion of a **comprehensive software product impossible, but a limited version must be introduced to meet competitive or business pressure**; a set of core product or **system requirements is well understood, but the details of product or system extensions have yet to be defined**.
- two common evolutionary process models:
  - **Prototyping**
  - **The Spiral Model**

Evolutionary Process Models: Prototyping

- A customer defines a set of general objectives for software, but **does not identify detailed requirements** for functions and features.
- The developer may be **unsure of the efficiency** of an algorithm, the adaptability of an operating system, or the GUI form.
- Regardless of the manner in which it is applied, the prototyping paradigm assists you and other stakeholders to better understand what is to be built **when requirements are fuzzy**.



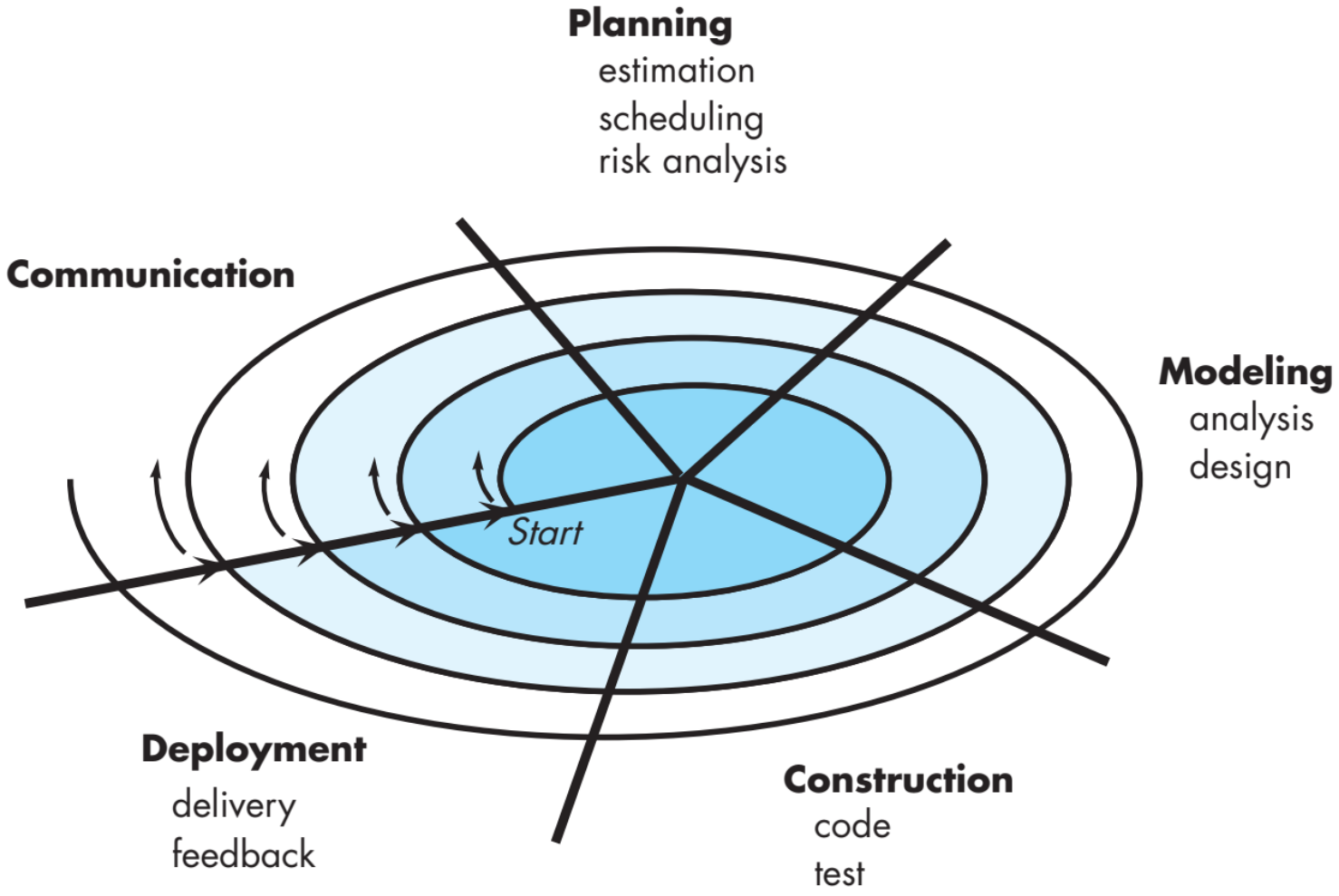
EVOLUTIONARY PROCESS MODELS: PROTOTYPING

Problem of Prototyping

- Stakeholders cry foul and demand that “a few fixes” be applied to make the prototype a working product.
- As a software engineer, you often make implementation compromises in order to get a prototype working quickly.
- An inefficient algorithm may be implemented simply to demonstrate capability. After a time, you may become comfortable with these choices and forget all the reasons why they were inappropriate.

Evolutionary Process Models: Spiral

- The spiral model is an evolutionary software process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model.
- A spiral model is divided into a set of framework activities defined by the software engineering team.
- Unlike other process models that end when software is delivered, the spiral model can be adapted to apply throughout the life of the computer software.
- The spiral model is a realistic approach to the development of large-scale systems and software.



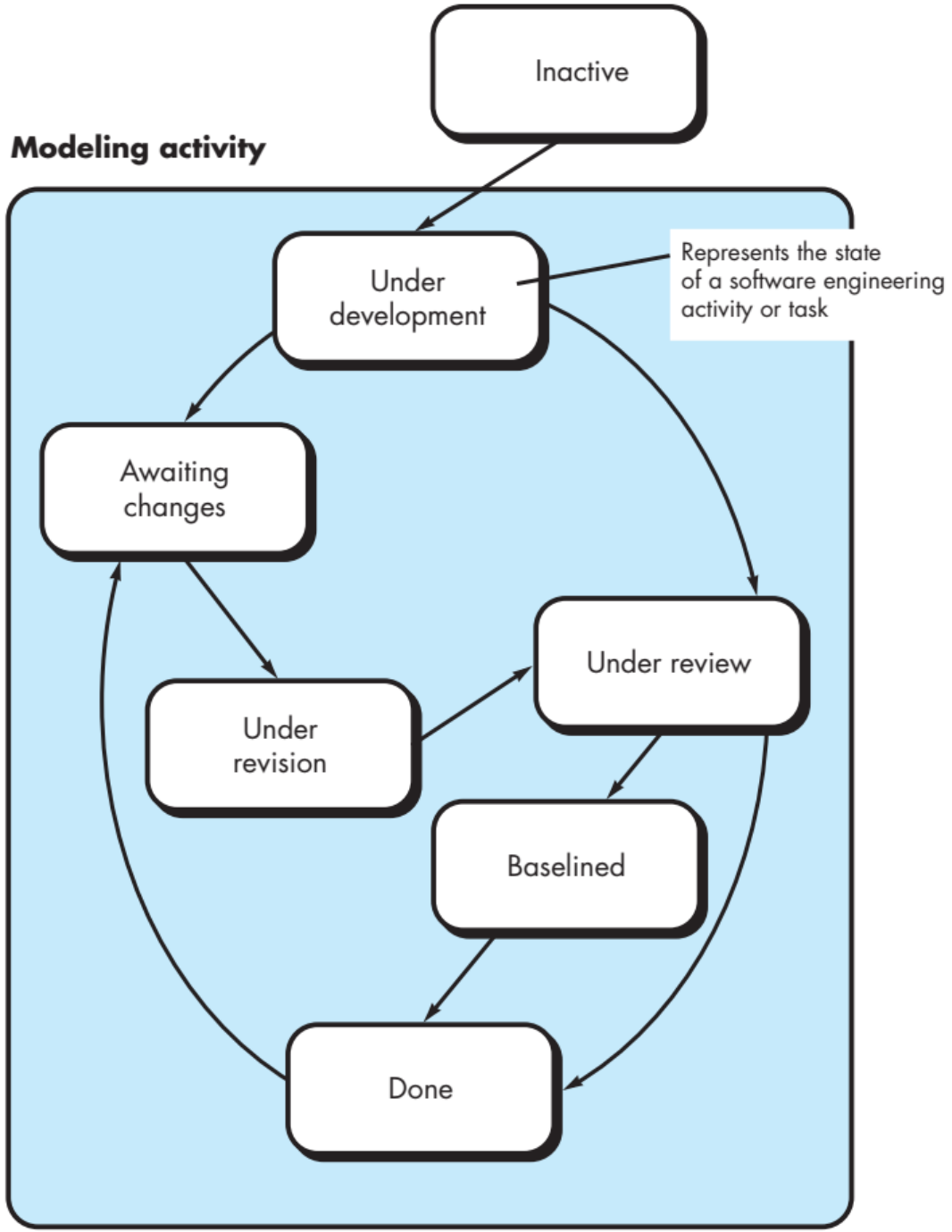
EVOLUTIONARY PROCESS MODELS: SPIRAL

Problem of Spiral

- It may be difficult to convince customers (particularly in contract situations) that the evolutionary approach is controllable.
- It demands considerable risk assessment expertise and relies on this expertise for success.
- If a major risk is not uncovered and managed, problems will undoubtedly occur.

Concurrent Model

- The *concurrent development model*, sometimes called *concurrent engineering*, allows a software team to represent iterative and concurrent elements.
- Concurrent modeling defines a series of events that will trigger transitions from state to state for each of the software engineering activities, actions, or tasks.
- Concurrent modeling is applicable to all types of software development and provides an accurate picture of the current state of a project



CONCURRENT MODEL

One element of the concurrent process model

PROCESS MODEL

Specialized Process Model

This Presentation Covers:

1. Software Process Structure
   a) Generic Process
   b) Process Flow
2. Process Model
   a) Prescriptive Process Models: Waterfall, Incremental, Evolutionary (Prototyping & Spiral), concurrent models
   b) Specialized Process Models: Component-based development, The Formal Method Model, Aspect-Oriented Software Development,
   c) The Unified Process
   d) Product & Process

Specialized Process Model

- Specialized process models take on many of the characteristics of one or more of the traditional models presented in the preceding sections.
- 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

- The component-based development model incorporates many of the characteristics of the spiral model.
- Modeling and construction activities begin with the identification of candidate components.
- These components can be designed as either conventional software modules or object-oriented classes or packages11 of classes.

Steps of Component-Based Development

a) Available component-based products are researched and evaluated for the application domain in question

b) Component integration issues are considered

c) A software architecture is designed to accommodate the components.

d) Components are integrated into the architecture.

e) Comprehensive testing is conducted to ensure proper functionality

The Formal Methods Model

- The formal methods model encompasses a set of activities that leads to formal mathematical specification of computer software.
- Formal methods enable you to specify, develop, and verify a computer-based system by applying a rigorous, mathematical notation.

Aspect-Oriented Software Development

- Aspect-oriented software development (AOSD), often referred to as aspect-oriented programming (AOP) or aspect-oriented component engineering (AOCE), is a relatively new software engineering paradigm that provides **a process and methodological approach for defining, specifying, designing**, and **constructing aspects**.

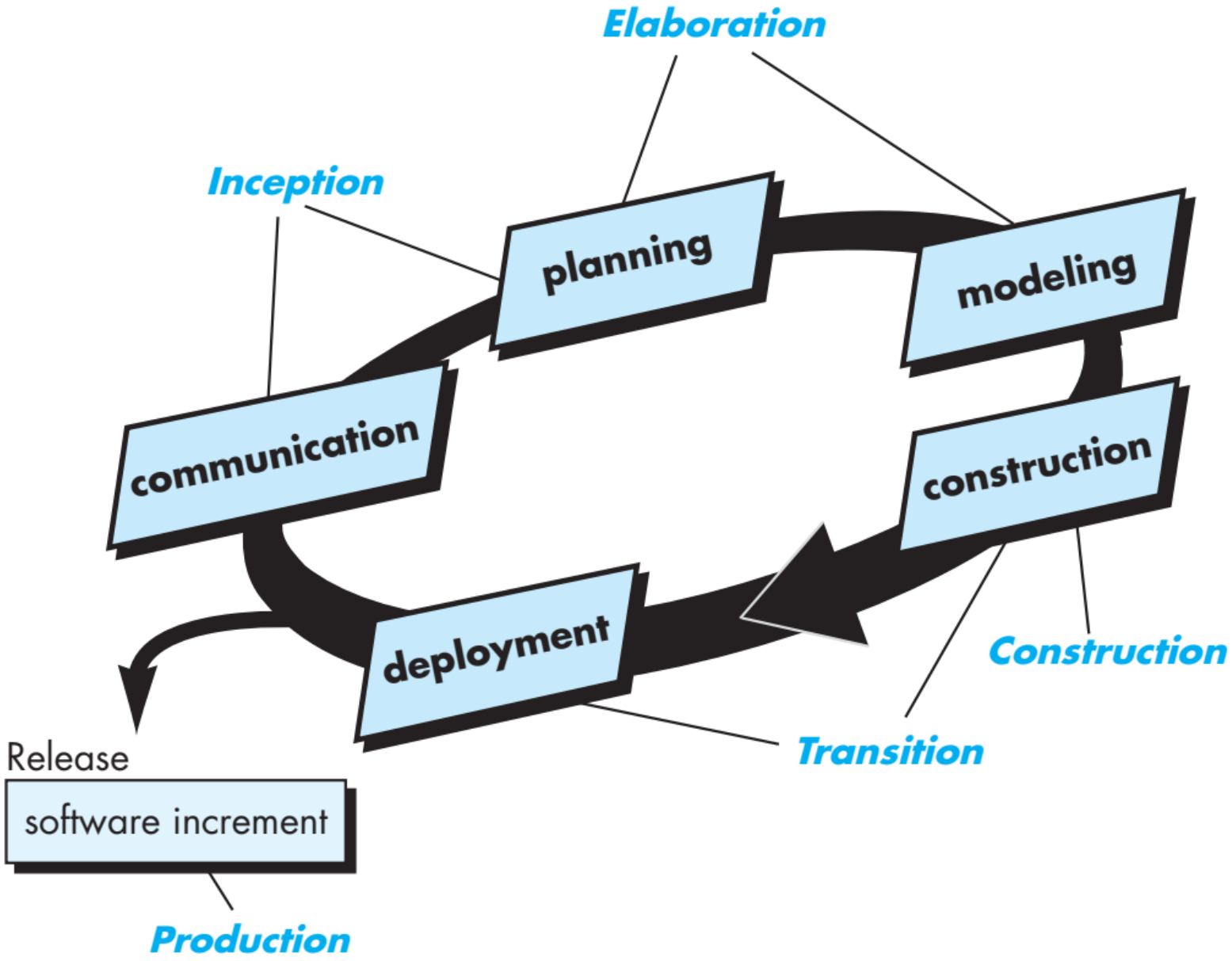
UNIFIED PROCESS

This Presentation Covers:

1. Software Process Structure
   a) Generic Process
   b) Process Flow
2. Process Model
   a) Prescriptive Process Models: Waterfall, Incremental, Evolutionary (Prototyping & Spiral), concurrent models
   b) Specialized Process Models: Component-based development, The Formal Method Model, Aspect-Oriented Software Development,
   c) The Unified Process
   d) Product & Process

The Unified Process

- In some ways the Unified Process is an attempt to draw on the best features and characteristics of traditional software process models, but characterize them in a way that implements many of the best principles of agile software development
- The Unified Process recognizes the importance of customer communication and streamlined methods for describing the customer's view of a system (the use case)
- It emphasizes the important role of software architecture and "helps the architect focus on the right goals, such as understandability, reliance to future changes, and reuse".



THE UNIFIED PROCESS

PRODUCT & PROCESS

This Presentation Covers:

1. Software Process Structure
   a) Generic Process
   b) Process Flow
2. Process Model
   a) Prescriptive Process Models: Waterfall, Incremental, Evolutionary (Prototyping & Spiral), concurrent models
   b) Specialized Process Models: Component-based development, The Formal Method Model, Aspect-Oriented Software Development,
   c) The Unified Process
   d) Product & Process

Process & Product

- If the process is weak, the end product will undoubtedly suffer. But an obsessive overreliance on process is also dangerous.
- People derive as much (or more) satisfaction from the creative process as they do from the end product.
- As creative software professional, you should also derive as much satisfaction from the process as the end product.

QUIZ TIME!!

Quiz 2

VIDEO TIME!!

Video Time

- Hobi VS Passion VS Profesi
- Cara supaya tidak membenci pelajaran

Hobi VS Passion VS Profesi

Tips tidak membenci pelajaran