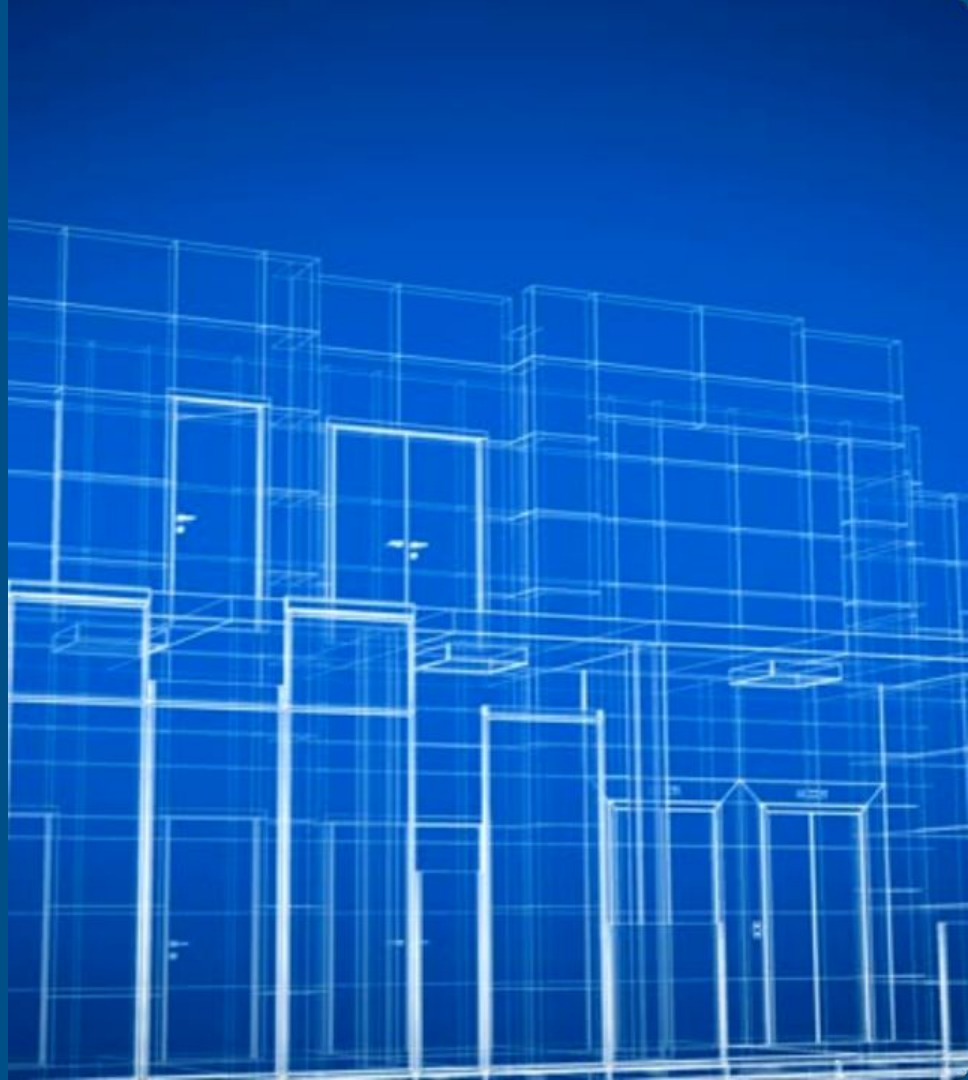
Process Models

Chapter 2: Software Engineering

The Foundation

"Imagine building a skyscraper without a blueprint or a construction schedule. Chaos, right?"

Building complex software without a structured approach leads to crisis. Today, we move from the **what** (software) to the **how**—the structured roadmaps we call Process Models.



Lecture Objectives

- **Deconstruct** a generic process model into framework activities, task sets, and patterns.
- **Understand** the purpose of process assessment and improvement (CMMI).
- **Compare** prescriptive process models: Waterfall, Incremental, and Evolutionary.
- **Identify** which model is suited for different project scenarios.

The Generic Process Model

The 5 Core Framework Activities that form the skeleton of any process:

Communication

Collaborate with stakeholders to gather requirements.

Planning

Estimate, schedule, and mitigate risks.

Modeling

Analysis & Design: Creating system representations.

Construction

Code generation and testing.

Deployment

Delivery, feedback, and support.

FIGURE 2.1

A software process framework

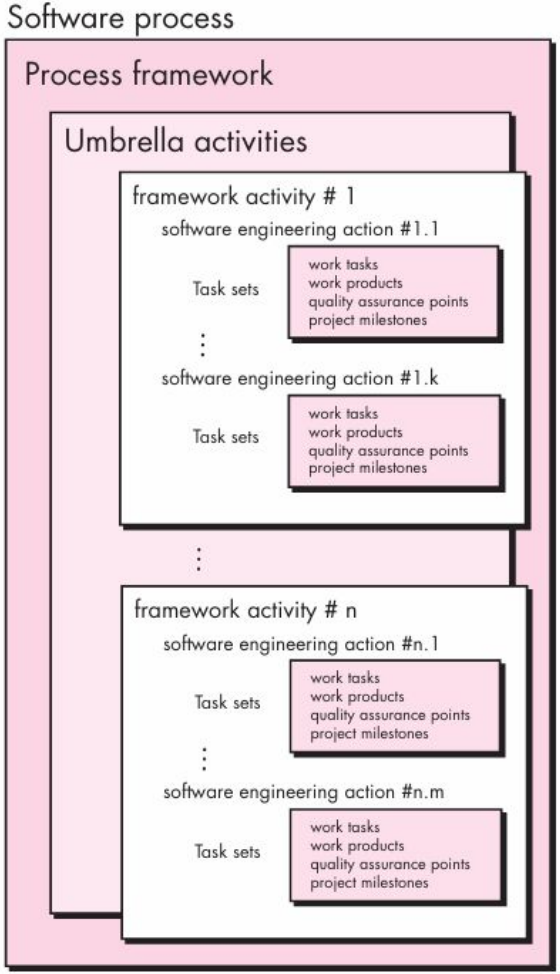
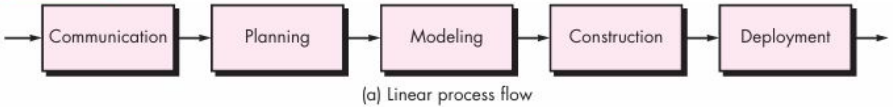
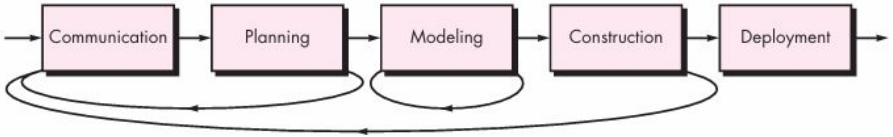


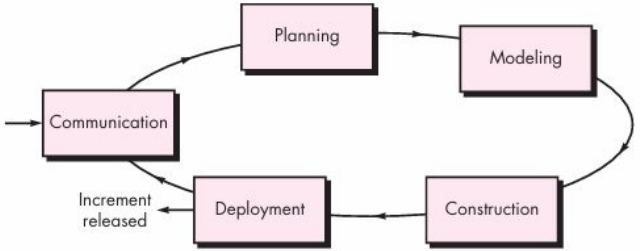
FIGURE 2.2 Process flow



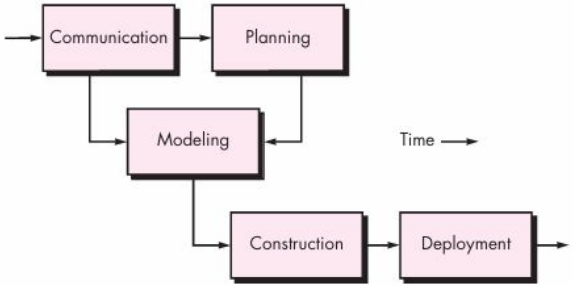
(a) Linear process flow



(b) Iterative process flow



(c) Evolutionary process flow



(d) Parallel process flow

Adapting the Process: Task Sets

What is a Task Set?

A Task Set defines the actual work (actions, products, quality checks) needed for a framework activity.

They are adaptive based on:

- Project Type (Critical vs. Casual)
- Project Size (3 months vs. 3 years)
- Team Experience (Novice vs. Expert)

Analogy: Preparing for a desert trek vs. a business trip requires different "task sets."

INFO

Task Set

A task set defines the actual work to be done to accomplish the objectives of a software engineering action. For example, *elicitation* (more commonly called "requirements gathering") is an important software engineering action that occurs during the communication activity. The goal of requirements gathering is to understand what various stakeholders want from the software that is to be built.

For a small, relatively simple project, the task set for requirements gathering might look like this:

1. Make a list of stakeholders for the project.
2. Invite all stakeholders to an informal meeting.
3. Ask each stakeholder to make a list of features and functions required.
4. Discuss requirements and build a final list.
5. Prioritize requirements.
6. Note areas of uncertainty.

For a larger, more complex software project, a different task set would be required. It might encompass the following work tasks:

1. Make a list of stakeholders for the project.
2. Interview each stakeholder separately to determine overall wants and needs.
3. Build a preliminary list of functions and features based on stakeholder input.
4. Schedule a series of facilitated application specification meetings.
5. Conduct meetings.
6. Produce informal user scenarios as part of each meeting.
7. Refine user scenarios based on stakeholder feedback.
8. Build a revised list of stakeholder requirements.
9. Use quality function deployment techniques to prioritize requirements.
10. Package requirements so that they can be delivered incrementally.
11. Note constraints and restrictions that will be placed on the system.
12. Discuss methods for validating the system.

Both of these task sets achieve "requirements gathering," but they are quite different in their depth and formality. The software team chooses the task set that will allow it to achieve the goal of each action and still maintain quality and agility.

Process Patterns

The Concept

A Process Pattern describes a proven, successful approach to a recurring process-related problem. It allows teams to build their process from a catalog of successful best practices.

Pattern Structure

Pattern Name: e.g., "Technical Review"

Intent: What problem does it solve?

Type: Stage, Task, or Phase pattern.

Solution: Specific steps to perform.

Resulting Context: Outcomes and artifacts.

INFO

An Example Process Pattern

The following abbreviated process pattern describes an approach that may be applicable when stakeholders have a general idea of what must be done but are unsure of specific software requirements.

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 2.3.3.

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 Assessment: CMMI

Moving from chaotic to optimizing using the Capability Maturity Model Integration.

1

Initial

Chaotic, success depends on heroes.

2

Managed

Basic project management is repeatable.

3

Defined

Process standardized across organization.

4

Measured

Quantitatively managed and controlled.

5

Optimizing

Continuous process 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. The SCAMPI method uses the SEI CMMI as the basis for assessment [SEI00].

4 The SEI's CMMI [CMM07] describes the characteristics of a software process and the criteria for a successful process in voluminous detail.

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 (ISO/IEC15504)—a standard that 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].

Prescriptive Process Models

The classic "Planned" approaches that prescribe a specific order of activities.

The Waterfall Model

Characteristics

- Linear, sequential flow.
- Each phase must be 100% complete before the next begins.

Use When:

Requirements are fixed, technology is understood, and the project is short.

The Fatal Flaw:

Inflexible. Working software is delivered very late, making it high risk for undefined requirements.

FIGURE 2.3 The waterfall model

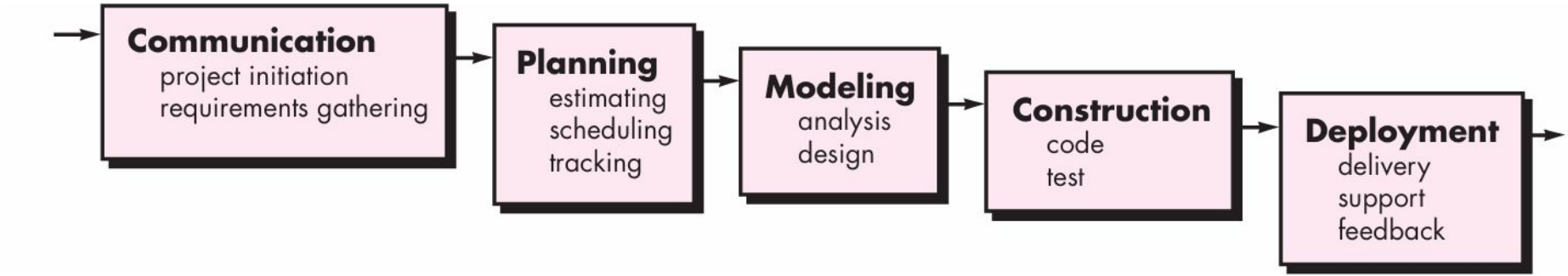
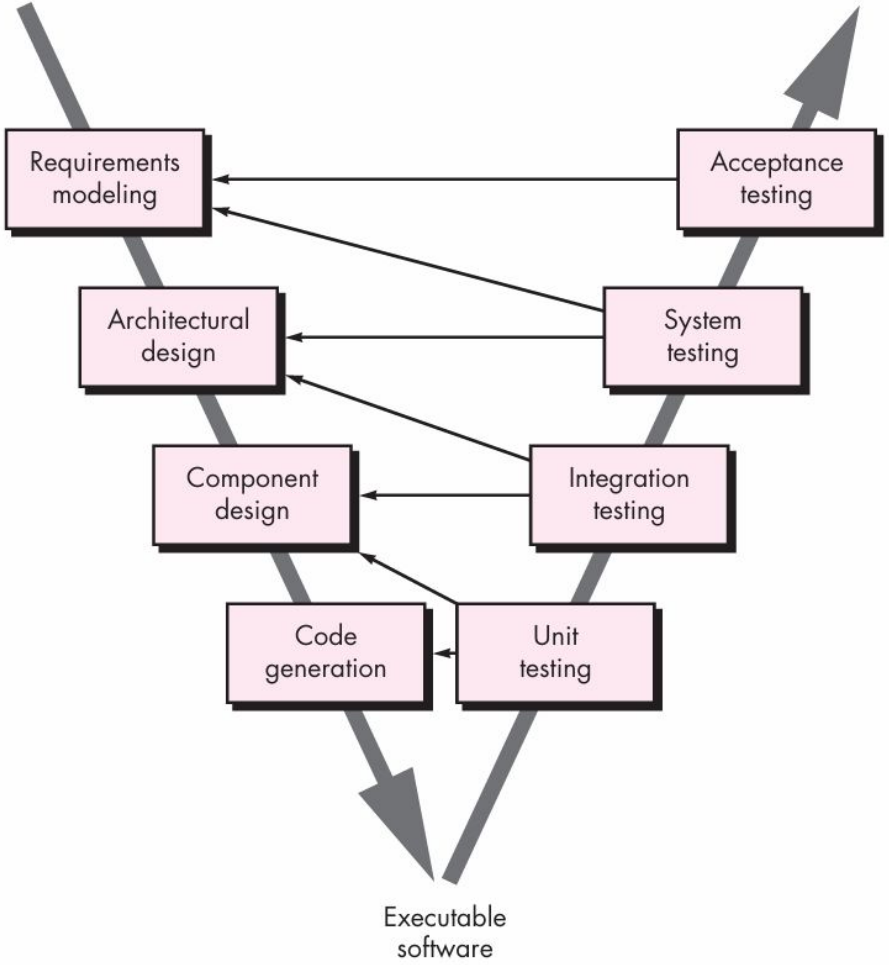


FIGURE 2.4

The V-model



early construction activities. As a software team moves down the left side of the V, basic problem requirements are refined into progressively more detailed and technical representations of the problem and its solution. Once code has been generated, the team moves up the right side of the V, essentially performing a series of tests (quality assurance actions) that validate each of the models created as the team moved down the left side.[7] In reality, there is no fundamental difference between the

Incremental Process Models

Philosophy

"Don't deliver the whole system at once. Deliver it in pieces."

Core requirements are addressed in the first increment. Each subsequent increment adds functionality like a mini-waterfall.

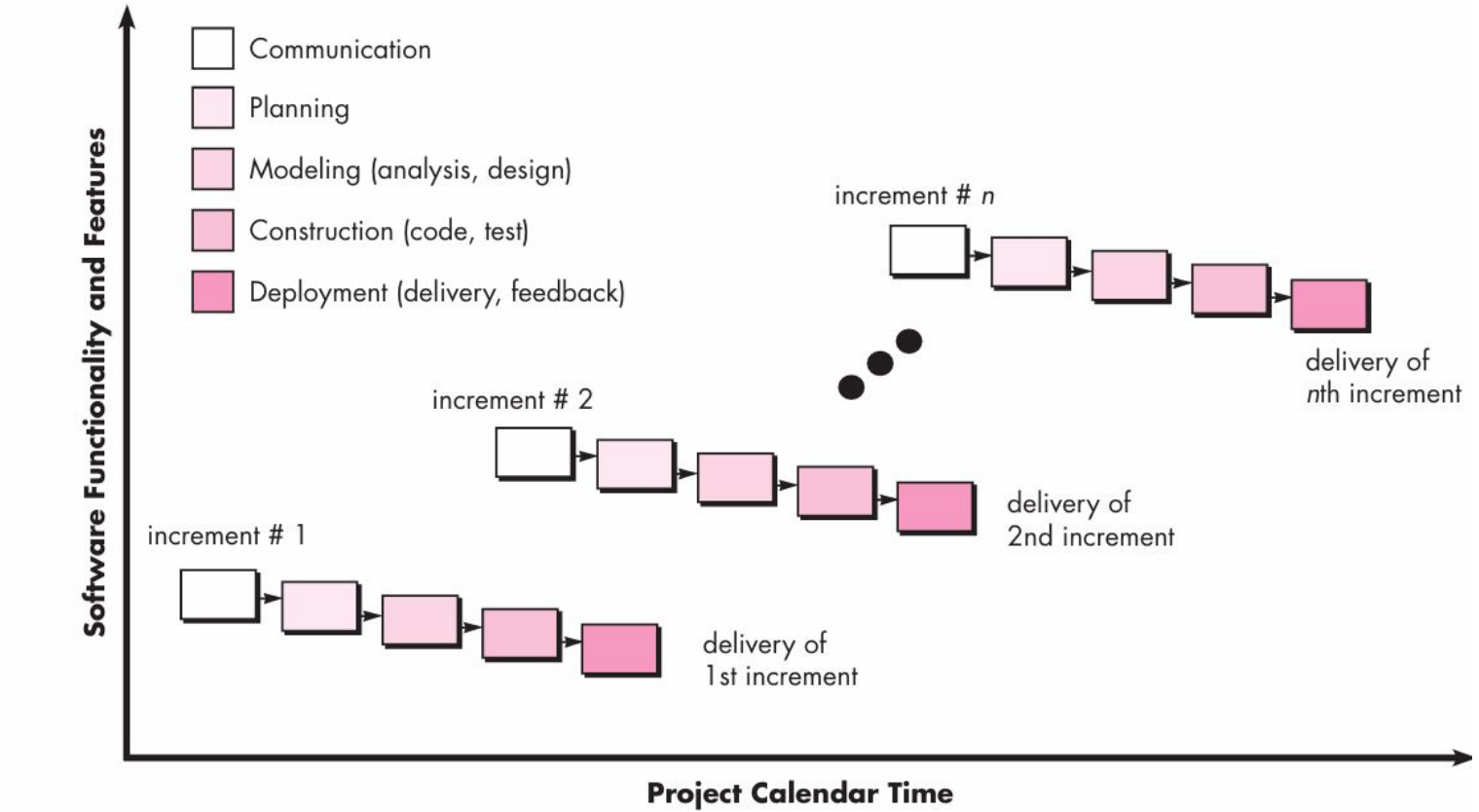
Strategic Value

When to use:

- Need early delivery of a partial system (ROI).
- Core features are stable, but extras are evolving.
- To mitigate staffing risks (team builds up over time).

FIGURE 2.5

The incremental model



Prototyping

Goal: To understand unclear requirements (e.g., UI workflows).

> Cycle: Quick Design → Build Prototype → Customer Evaluation → Refine.

Risk: Management may mistake the "quick & dirty" prototype for the final product.

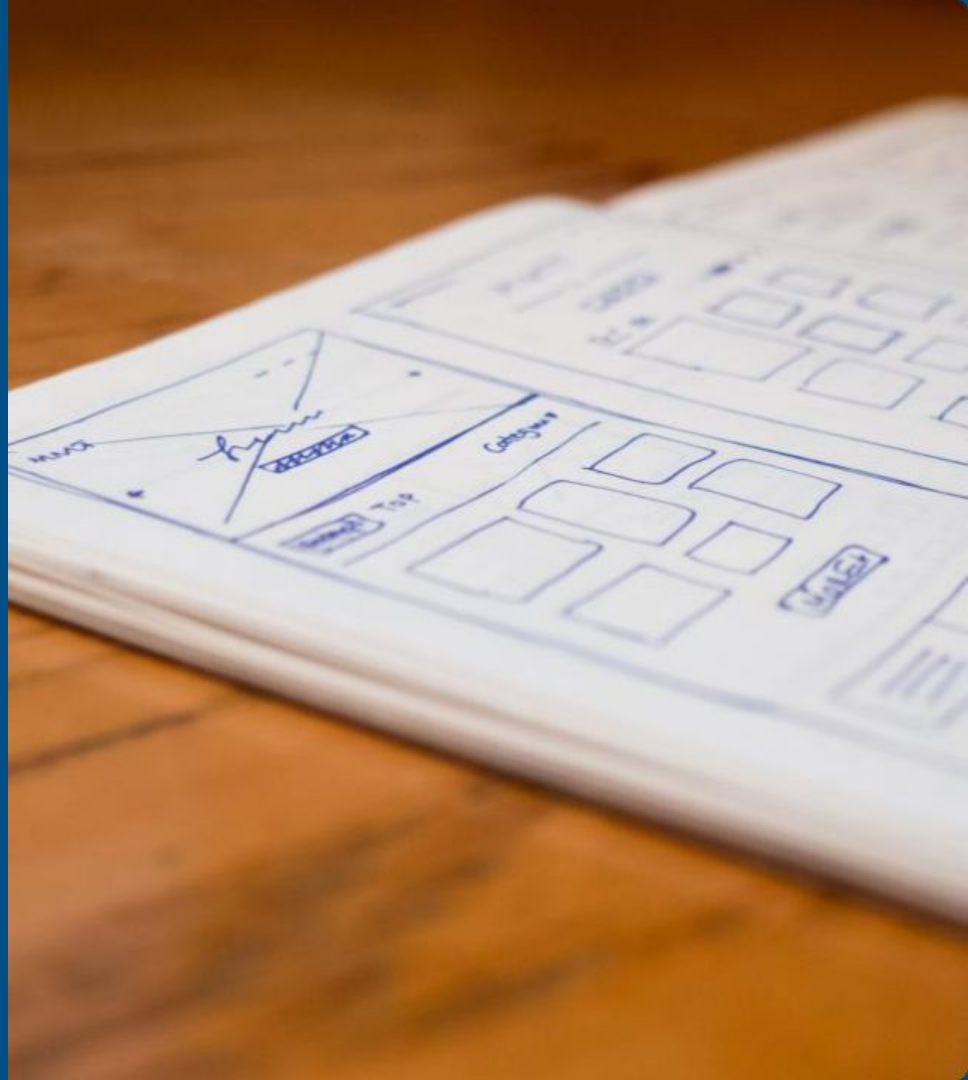
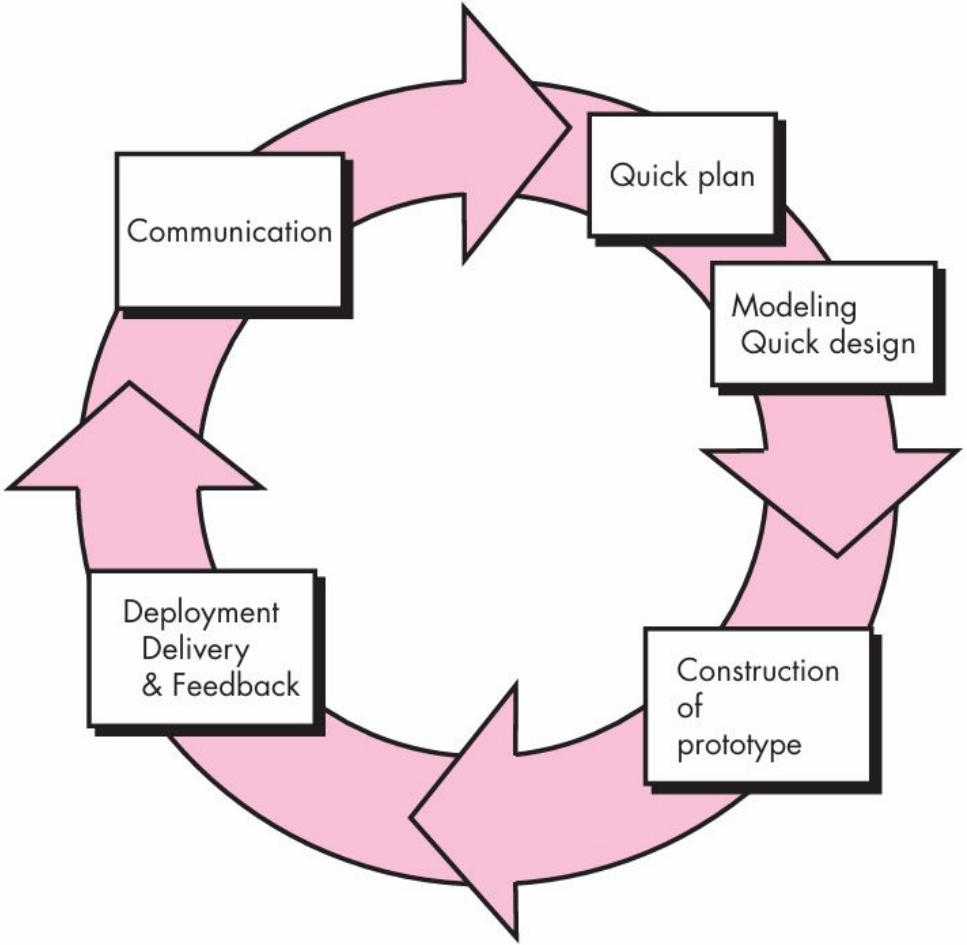


FIGURE 2.6

The prototyping paradigm



The Spiral Model

Risk-Driven Evolution

The most sophisticated prescriptive model. It is an iterative waterfall, but each iteration is preceded by strict **Risk Analysis**.

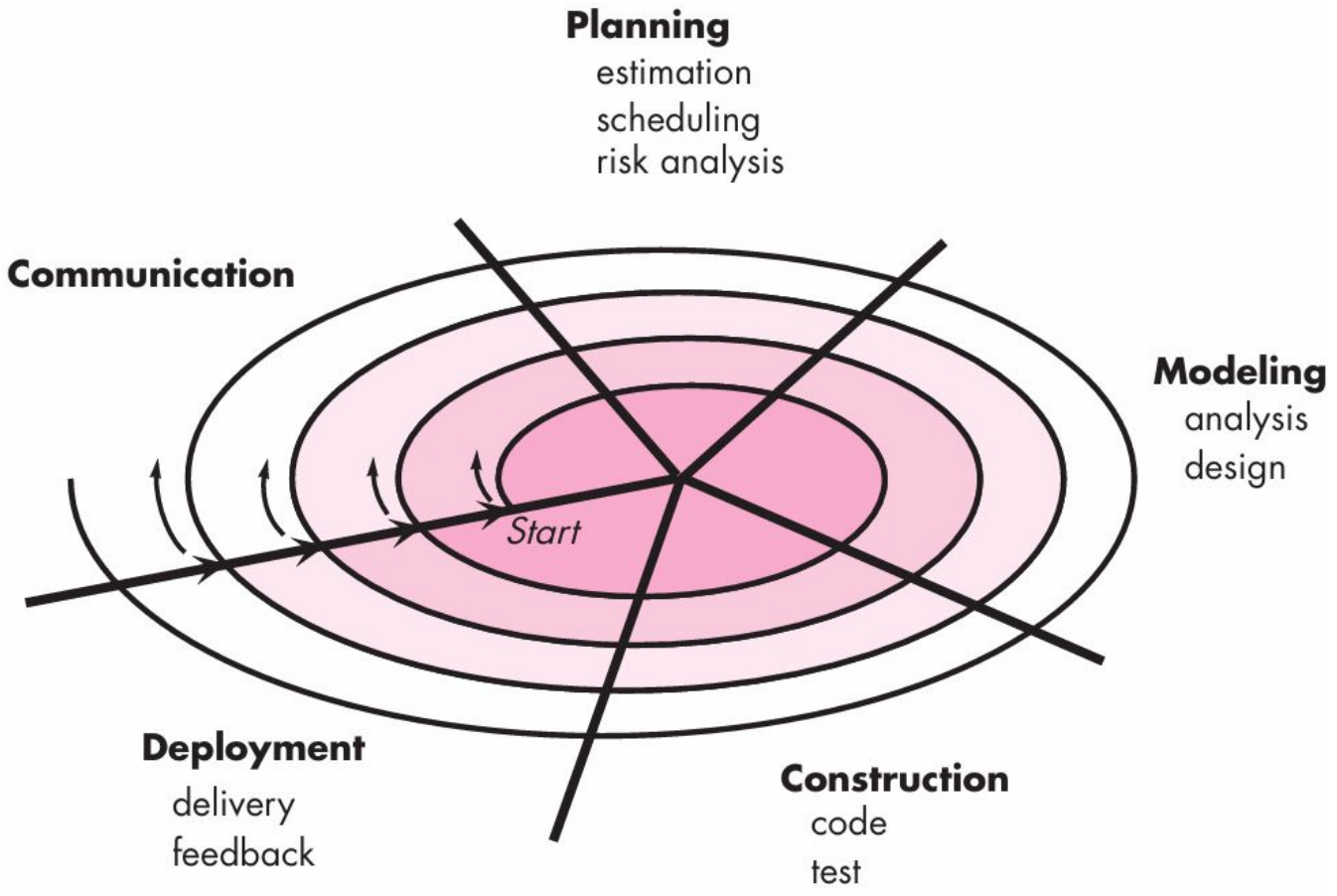
The 4 Quadrants:

1. Determine objectives & constraints.
2. **Identify and resolve risks.**
3. Develop and test.
4. Plan the next iteration.

Best for: Large, expensive, high-risk systems.

FIGURE 2.7

A typical spiral model



Model Comparison

Model	Flow	Customer Involvement	Handles Change?	Best For...
Waterfall	Linear	Start & End (Poor)	Very Poor	Stable, well-understood requirements.
Incremental	Staged	At delivery points	Moderate	Early ROI needed, Core known.
Evolutionary	Iterative	Continuous	Very Good	Unclear requirements, high innovation/risk.

Key Takeaways

- **Structure is Key:** Generic models provide universal blocks (Activities, Task Sets, Patterns).
- **Measure to Improve:** Frameworks like CMMI are essential for organizational maturity.
- **Context Matters:** There is no "best" model.
  - Waterfall for Stability.
  - Incremental for Chunks.
  - Spiral for Risk.

The Next Step

These models are plan-heavy. What if requirements change constantly?

Coming Up: The Agile Revolution

Image Sources

https://media.istockphoto.com/id/1171765357/video/abstract-beautiful-building-process-of-skyscraper-blueprint-grid-seamless-looped-3d-animation.jpg?s=640x640&k=20&c=v-nfH1uLP2fkxm3fiZ-GGJ_Zkt6c9CLGmtwlqk8gWCA=

Source: www.istockphoto.com

https://andrewskurka.com/wp-content/uploads/escalante-navigating-map-compass.jpg

Source: andrewskurka.com

https://alphauniverseglobal.media.zestyio.com/Alpha-Universe-Photo-By-Mahesh-Thapa-starvingphotographer--Figure-6.jpg

Source: alphauniverse.com

https://freerangestock.com/sample/173787/close-up-of-a-website-wireframe-sketch.jpg

Source: freerangestock.com

https://static.vecteezy.com/system/resources/thumbnails/074/702/478/small_2x/abstract-spiral-galaxy-with-glowing-blue-particles-in-dark-space-free-video.jpg

Source: www.vecteezy.com

https://static.vecteezy.com/system/resources/previews/008/100/975/non_2x/modern-abstract-high-speed-technology-movement-dynamic-motion-light-trails-with-motion-blur-effect-on-dark-background-futuristic-technology-pattern-for-banner-or-poster-design-vector.jpg

Source: www.vecteezy.com