Ch.4 Process Models

4.1 Prescriptive Models

- **Prescriptive process models advocate an orderly approach to software engineering**

- **Questions:**

1. If prescriptive process models strive for structure and order, are they inappropriate for a software world that thrives on change?

2. 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?

4.1.1 The Waterfall Model

Communication
- Project initiation
- Requirements gathering

Planning
- Estimating
- Scheduling and tracking

Modeling
- Analysis and design

Construction
- Code and test

Deployment
- Delivery
- Support and feedback

👎 **Real projects rarely follow the sequential flow.**

👎 **Customers usually can't state all requirements explicitly.**

👎 **A working version will not be available until late in the project time-span.**

Classic

Life Cycle

4.1.1 The Waterfall Model

- **The V-model**

Requirement modeling → **Acceptance testing**

Architectural design → **System testing**

Component design → **Integration testing**

Code generation → **Unit testing**

Executable software
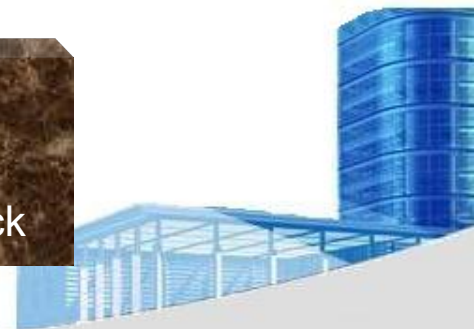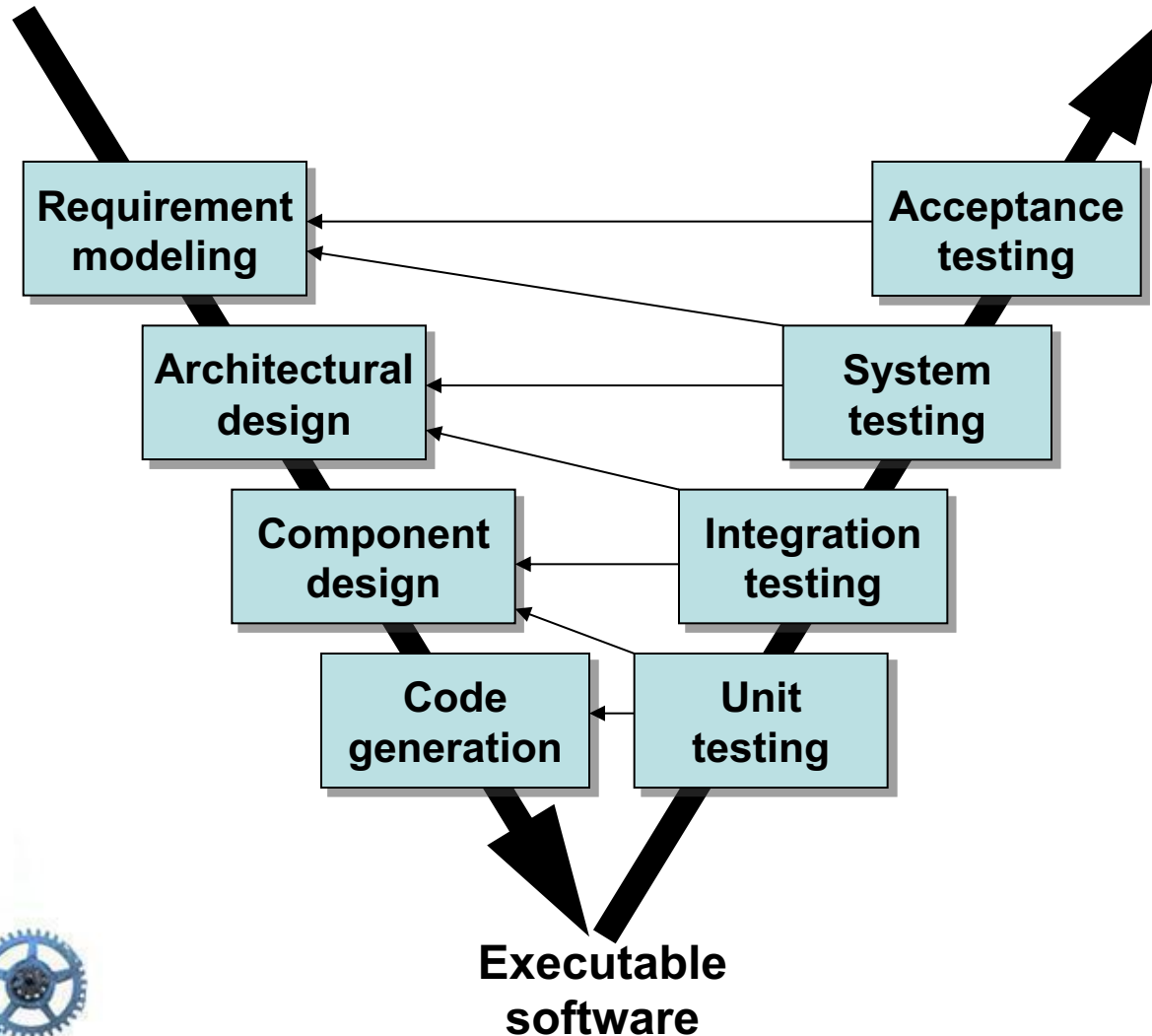
4.1.2 Incremental Process Models

• The Incremental Model

Increment #n

Delivery of nth increment

☞ **Makes a better use of resources.**

Increment #2

Delivery of 2nd increment

More features and functionality

Increment #1

Delivery of 1st increment

Core product

Software Functionality and Features

Project Calendar Time

Communication
Planning
Modeling
Construction
Deployment
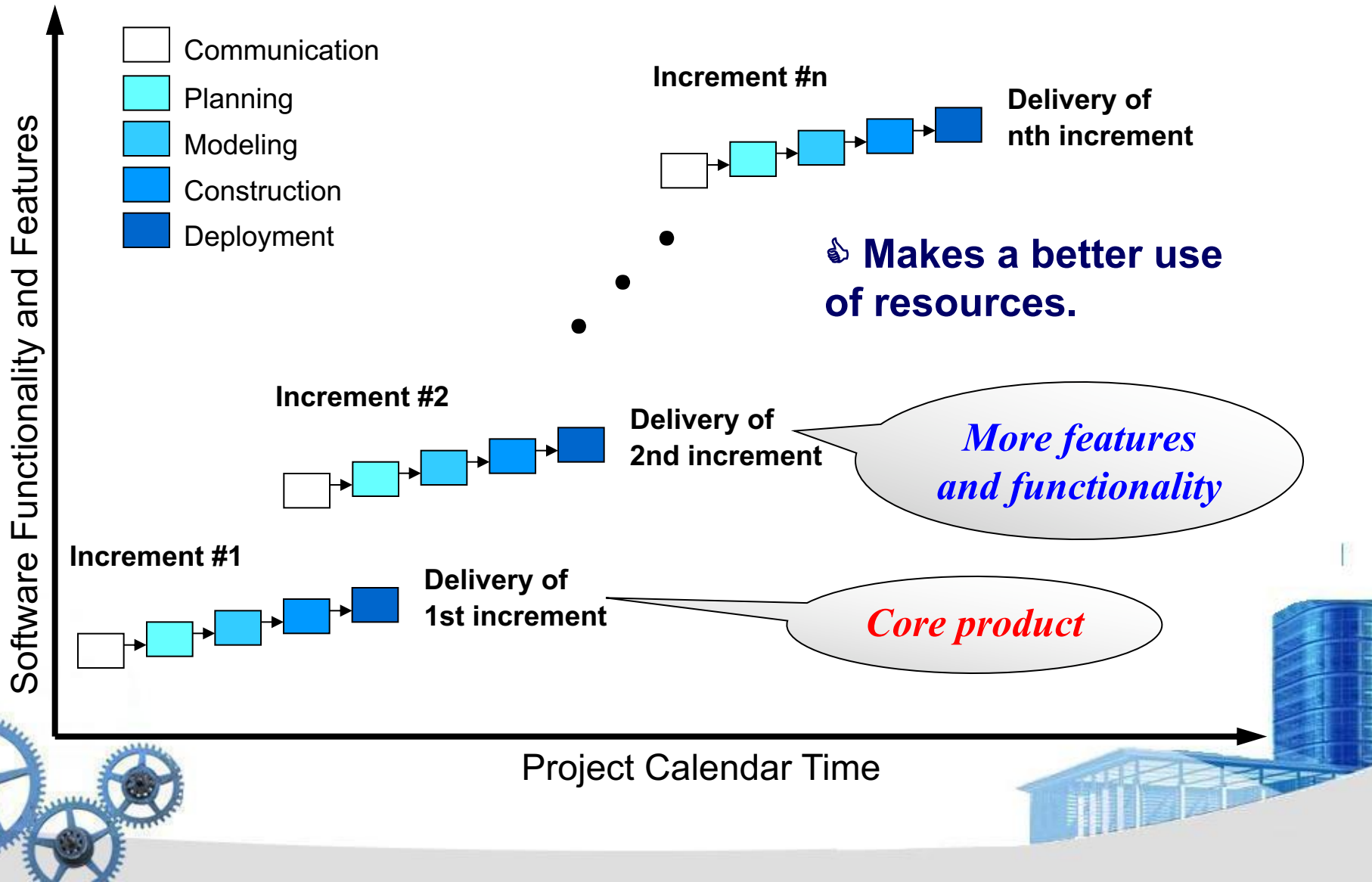
- **The Rapid Application Development (RAD) Model**

Team #n

Modeling
business modeling
data modeling
process modeling

Construction
component reuse
automatic code
generation
testing

Team #1

Modeling
business modeling
data modeling
process modeling

Construction
component reuse
automatic code
generation
testing

Communication

Planning

Deployment
integration
delivery
feedback

☞ **If tuning interfaces is needed, RAD may not work.**

☞ **RAD is not appropriate when technical risks are high.**

☞ **Require sufficient human resources.**

☞ **Require commitment to the rapid-fire activities from both developers and customers.**

☞ **If a system cannot be properly modularized, RAD may not work.**

60 – 90 days

4.1.3 Evolutionary Process Models

- **Prototyping**

Communication

Quick Plan

Modeling Quick design

Construction of prototype

Deployment Delivery & Feedback

- ➢ Good first step when customer has a legitimate need, but is clueless about the details
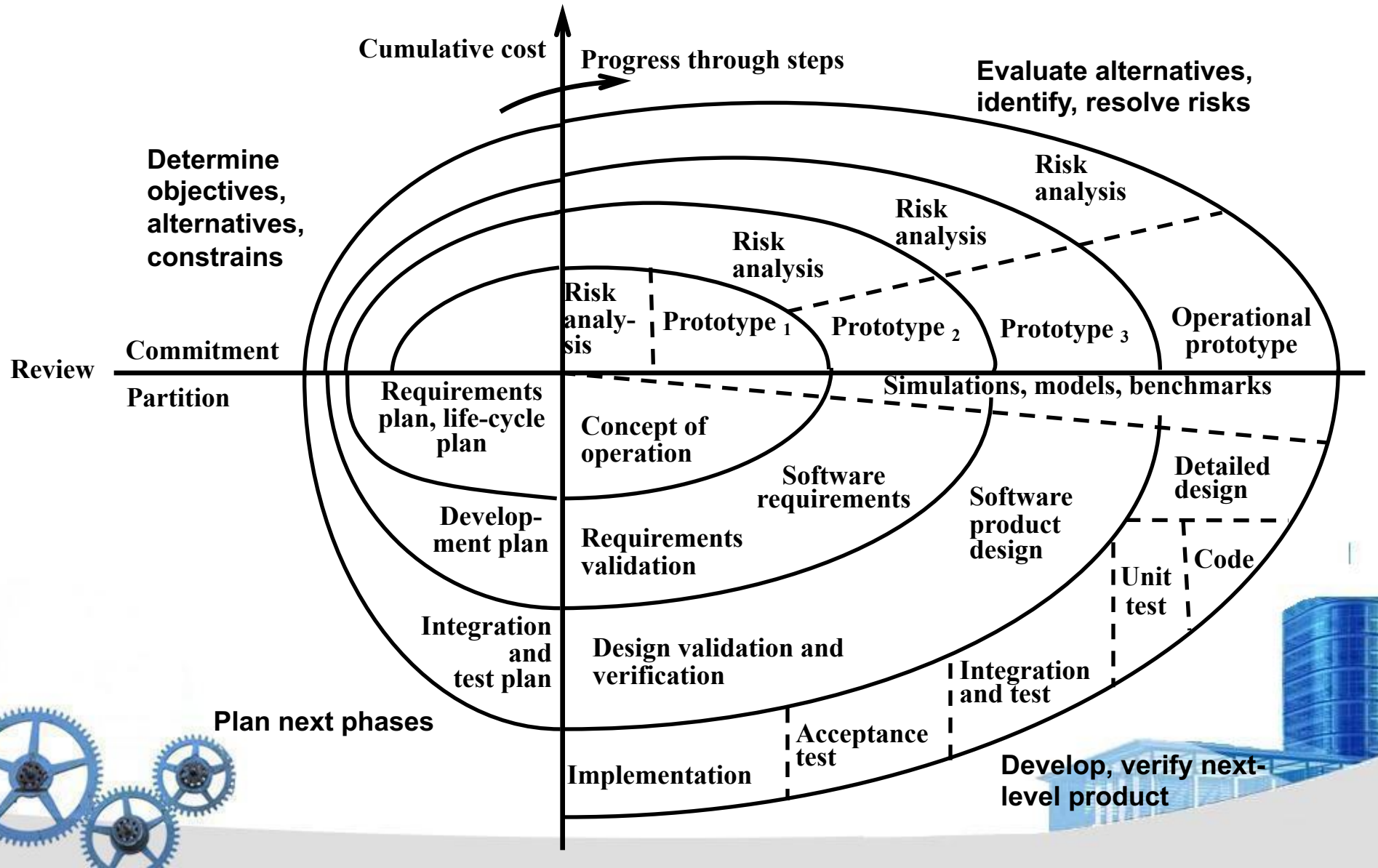- ➢ The prototype must be thrown away

• The Spiral Model

Cumulative cost

Progress through steps

Evaluate alternatives, identify, resolve risks

Determine objectives, alternatives, constrains

Review

Commitment

Partition

Risk analysis

Risk analysis

Risk analysis

Risk analysis

Risk analysis

Prototype 1

Prototype 2

Prototype 3

Operational prototype

Requirements plan, life-cycle plan

Concept of operation

Simulations, models, benchmarks

Software requirements

Software product design

Detailed design

Development plan

Requirements validation

Code

Unit test

Integration and test plan

Design validation and verification

Integration and test

Plan next phases

Acceptance test

Implementation

Develop, verify next-level product

- **The Concurrent Development Model**

➢ Defines a series of events that will trigger transitions from state to state for each of the activities, actions or tasks.

➢ Especially good for client/server applications.

➢ Defines a network of activities instead of linear sequence of events.

Flexibility
Extensibility
Speed of development

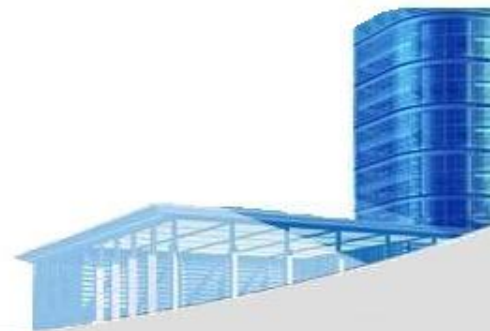**High Quality**
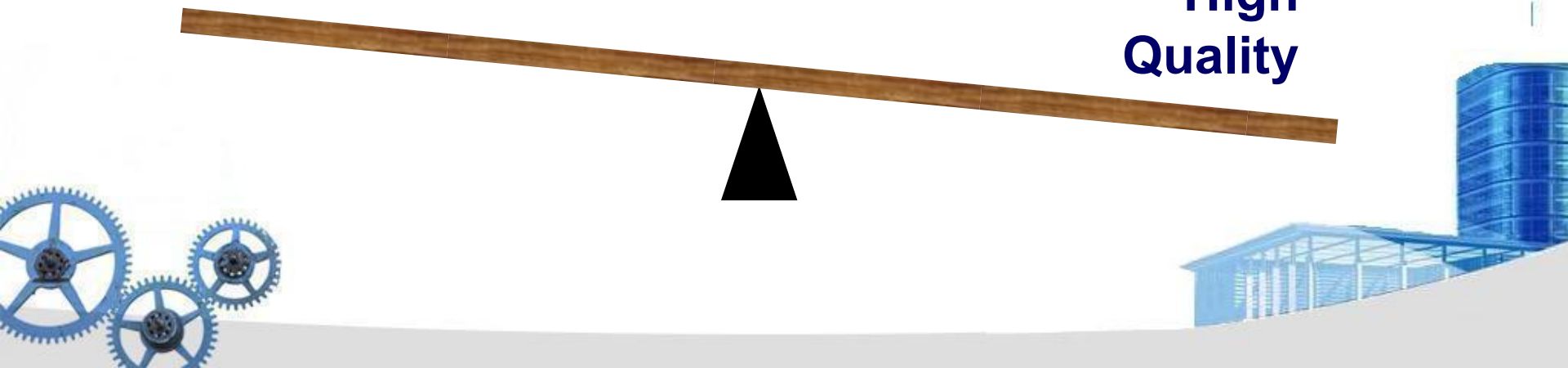
4.2 Specialized Process Models

- **Component based development** — the process to apply when reuse is a development objective
- **Formal methods** — emphasizes the mathematical specification of requirements
- **Aspect-Oriented Software Development** — provides a process and methodological approach for defining, specifying, designing, and constructing aspects

- A "**use-case** driven, **architecture**-centric, **iterative** and **incremental**" software process closely aligned with the **U**nified **M**odeling **L**anguage (**UML**)

- **Phases**
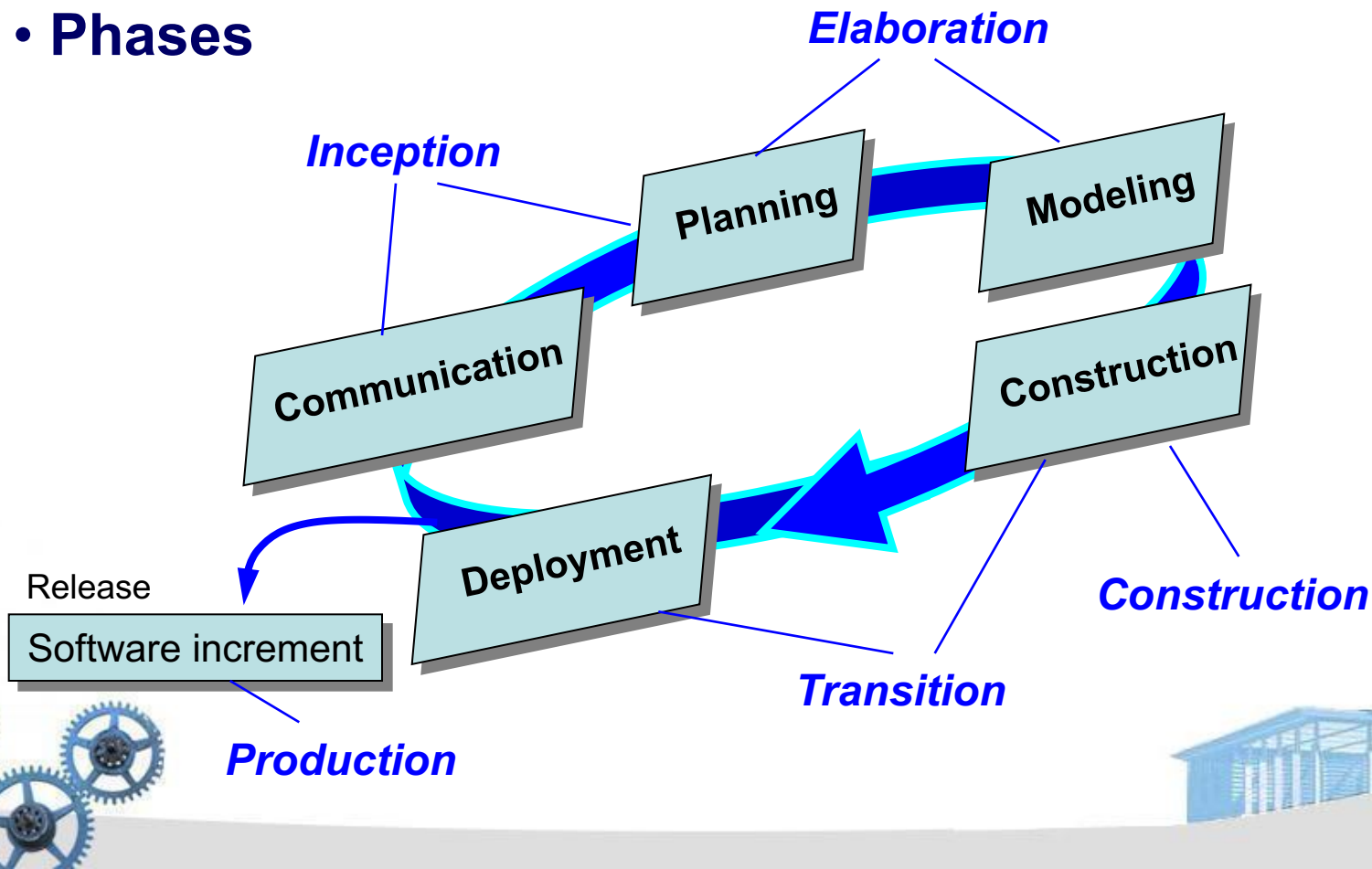
4.3 The Unified Process

• 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
• Prototypes

Elaboration phase

• Use-case model
• Functional and non-
   functional requirements
• Analysis model
• Software architecture
   description
• Executable architectural
   prototype
• Preliminary
   design model
• Revise risk list
• Project plan
   iteration plan, workflow,
   milestones
• Preliminary user manual

Construction phase

• Design model
• Software components
• Integrated software
   increment
• Test plan
• Test cases
• Support documentation
   user
   installation
   increment

Transition phase

• Delivered software
   increment
• Beta test reports
• User feedback

4.4 Personal and Team Process Models

- **Personal Software Process (PSP)**

➢ Recommends five framework activities:

1. Planning

2. High-level design

3. High-level design review

4. Development

5. Postmortem

➢ Stresses the need for each software engineer to identify errors early and as important, to understand the types of errors

4.4 Personal and Team Process Models

- **Team Software Process (TSP)**

➢ Each project is "launched" using a "script" that defines the tasks to be accomplished

➢ Teams are self-directed

➢ Measurement is encouraged

➢ Measures are analyzed with the intent of improving the team process