

TOPICS

**The Nature and History of Software
Development**

Problems with Software Development

**Software Engineering Paradigms and
Technology**

SOFTWARE ENGINEERING PARADIGMS

- ❑ **Life Cycle**
- ❑ **Prototyping Model**
- ❑ **Spiral Model**
- ❑ **Fourth Generation Techniques**
- ❑ **Combining Paradigms**
- ❑ **Generic Paradigm**

Software Engineering

**System
Engineering**

Analysis

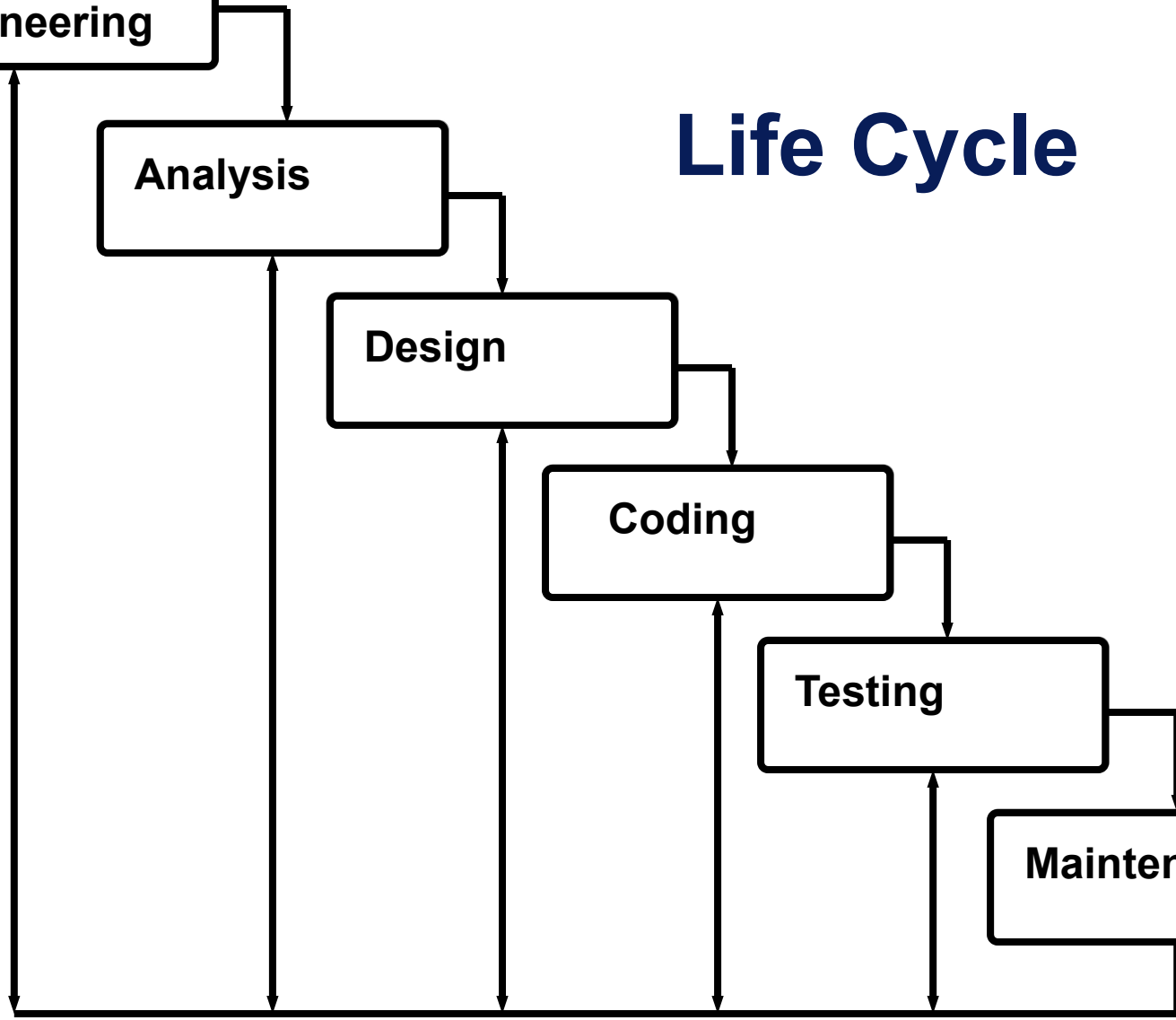
Design

Coding

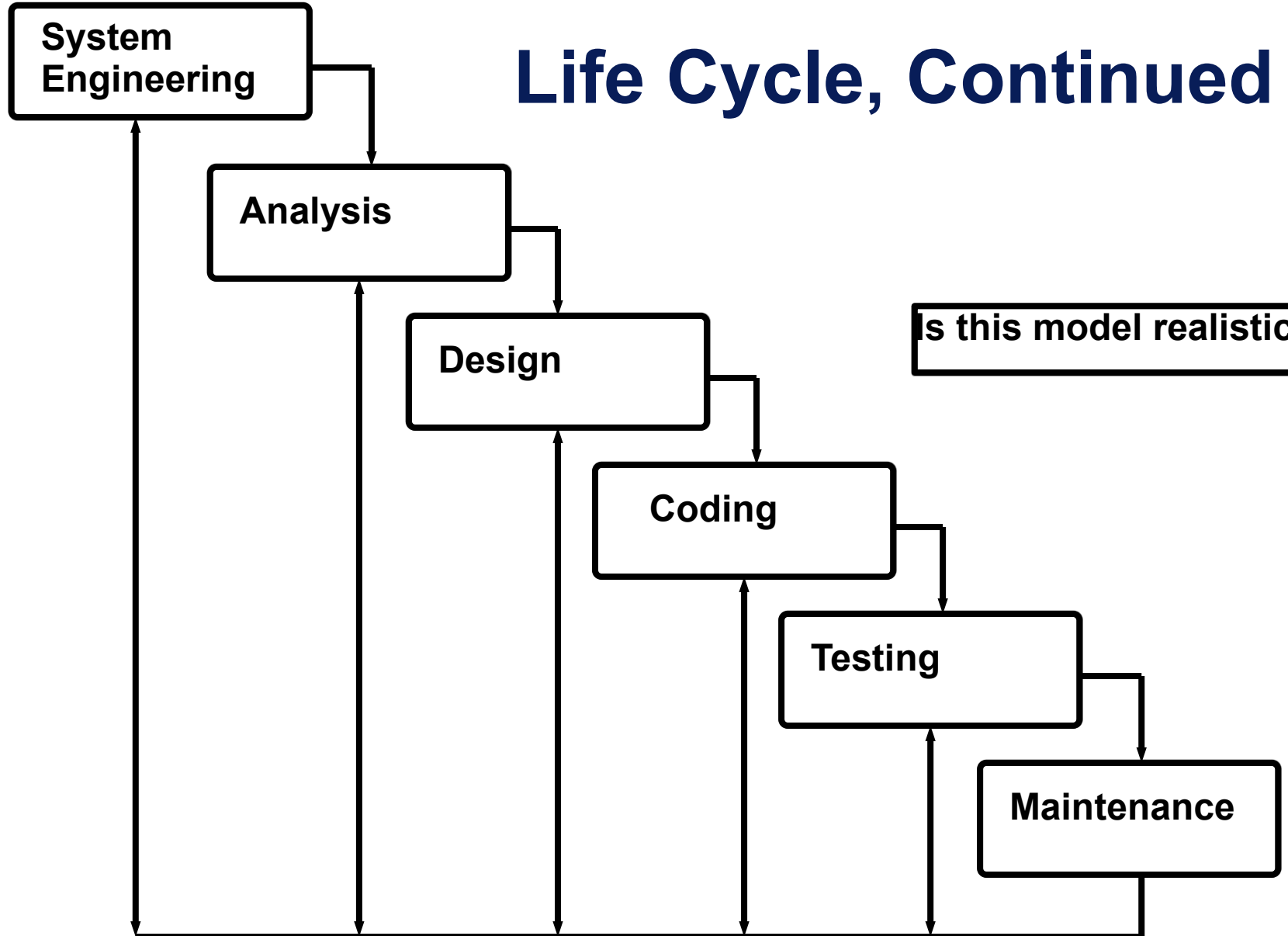
Testing

Maintenance

Life Cycle



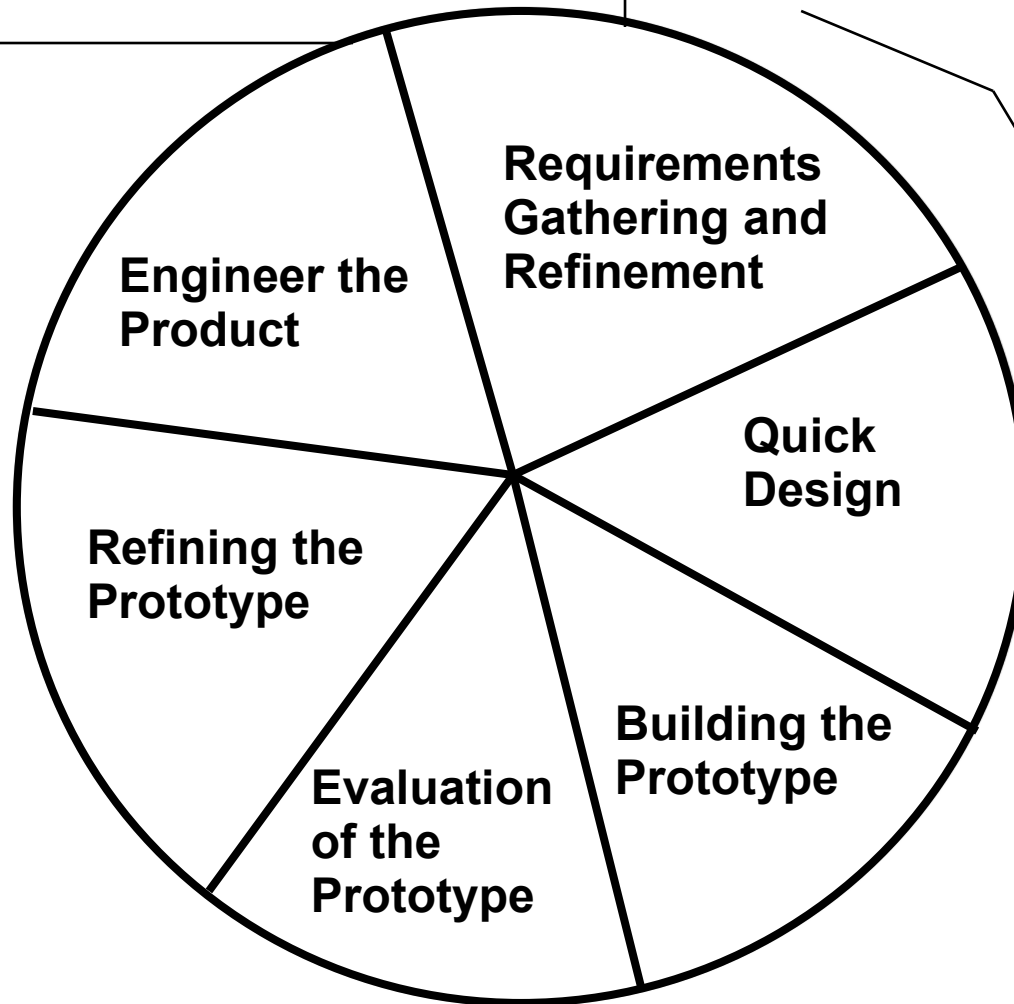
Life Cycle, Continued



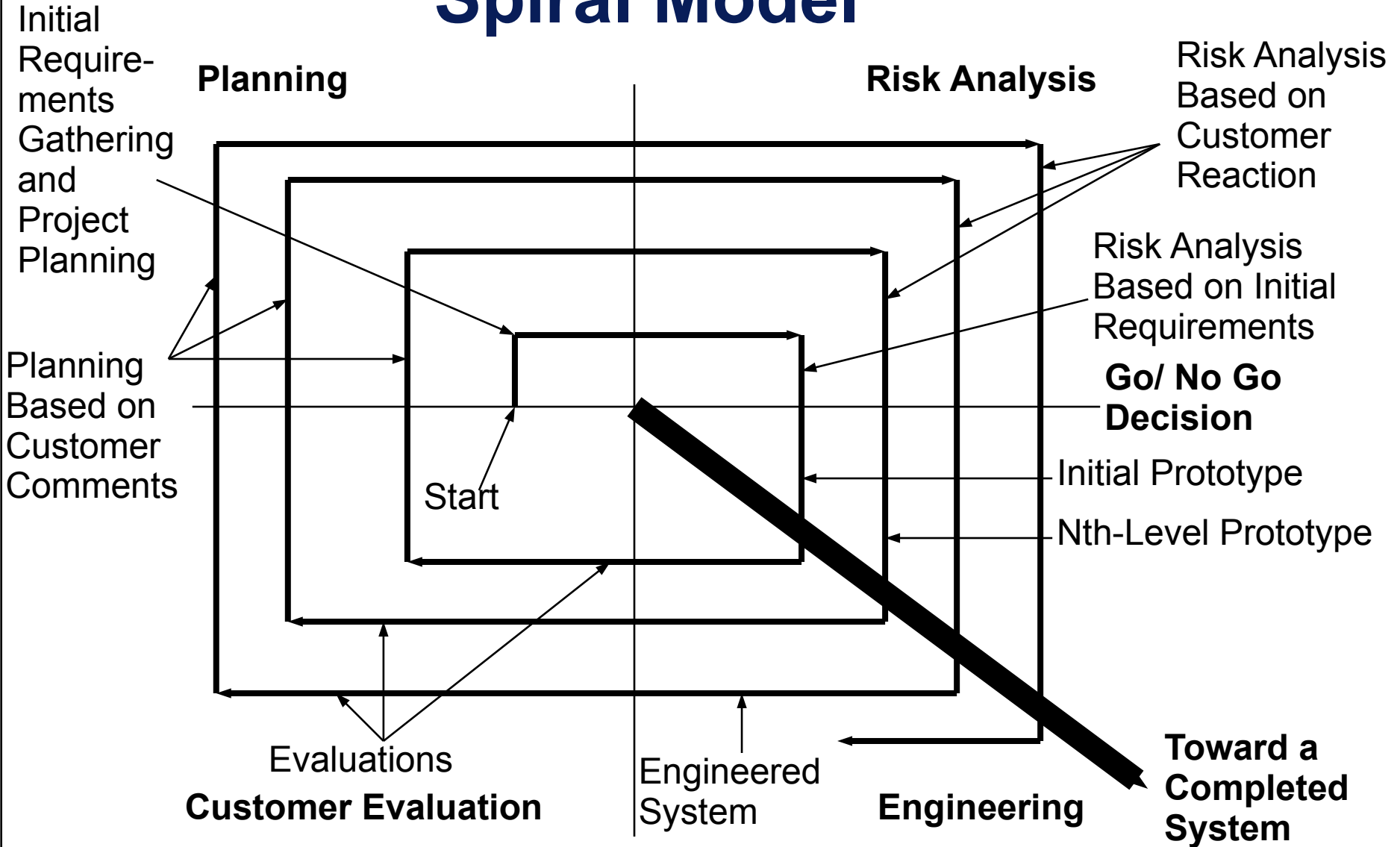
Is this model realistic?

Prototyping Model

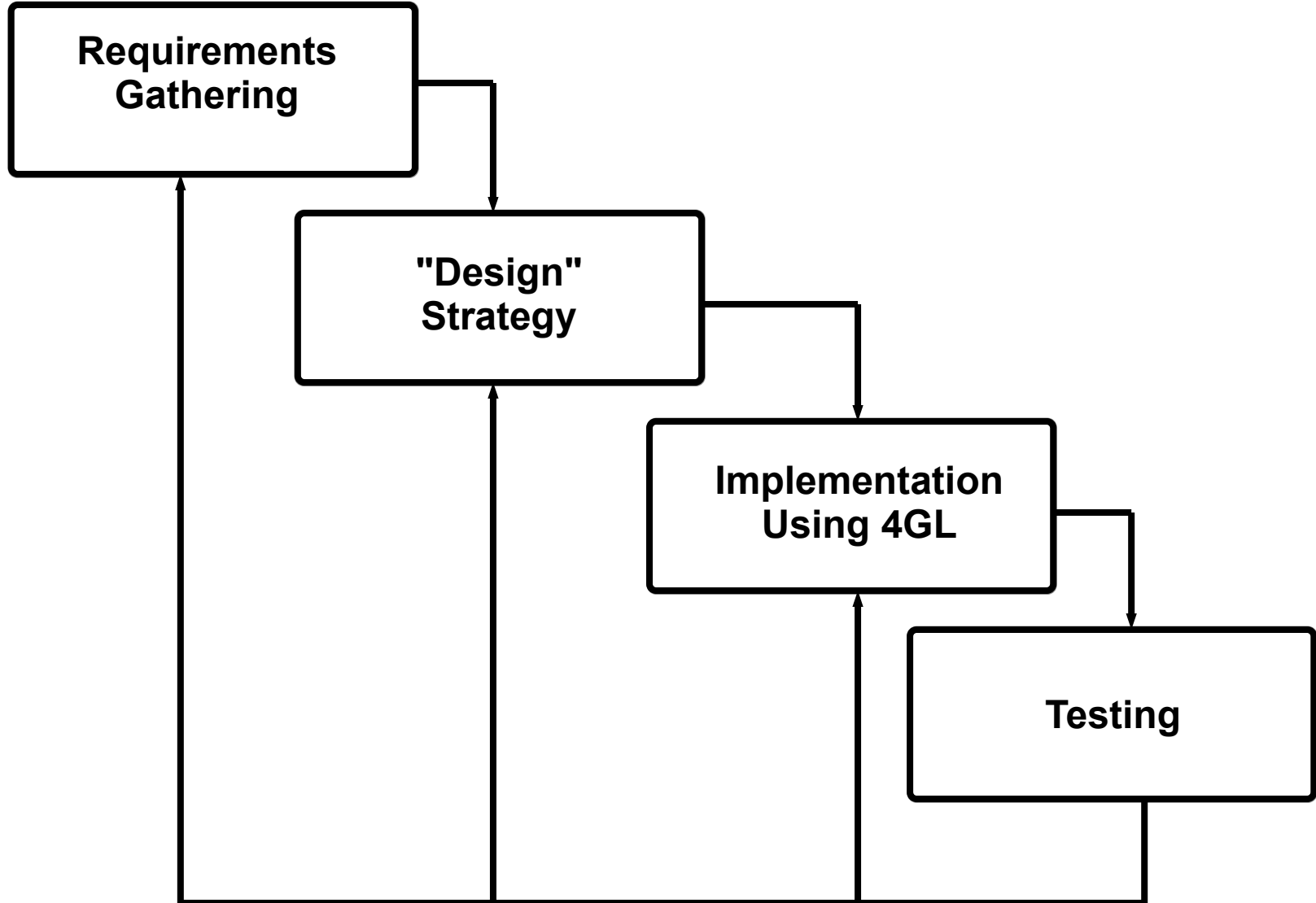
Start
Stop



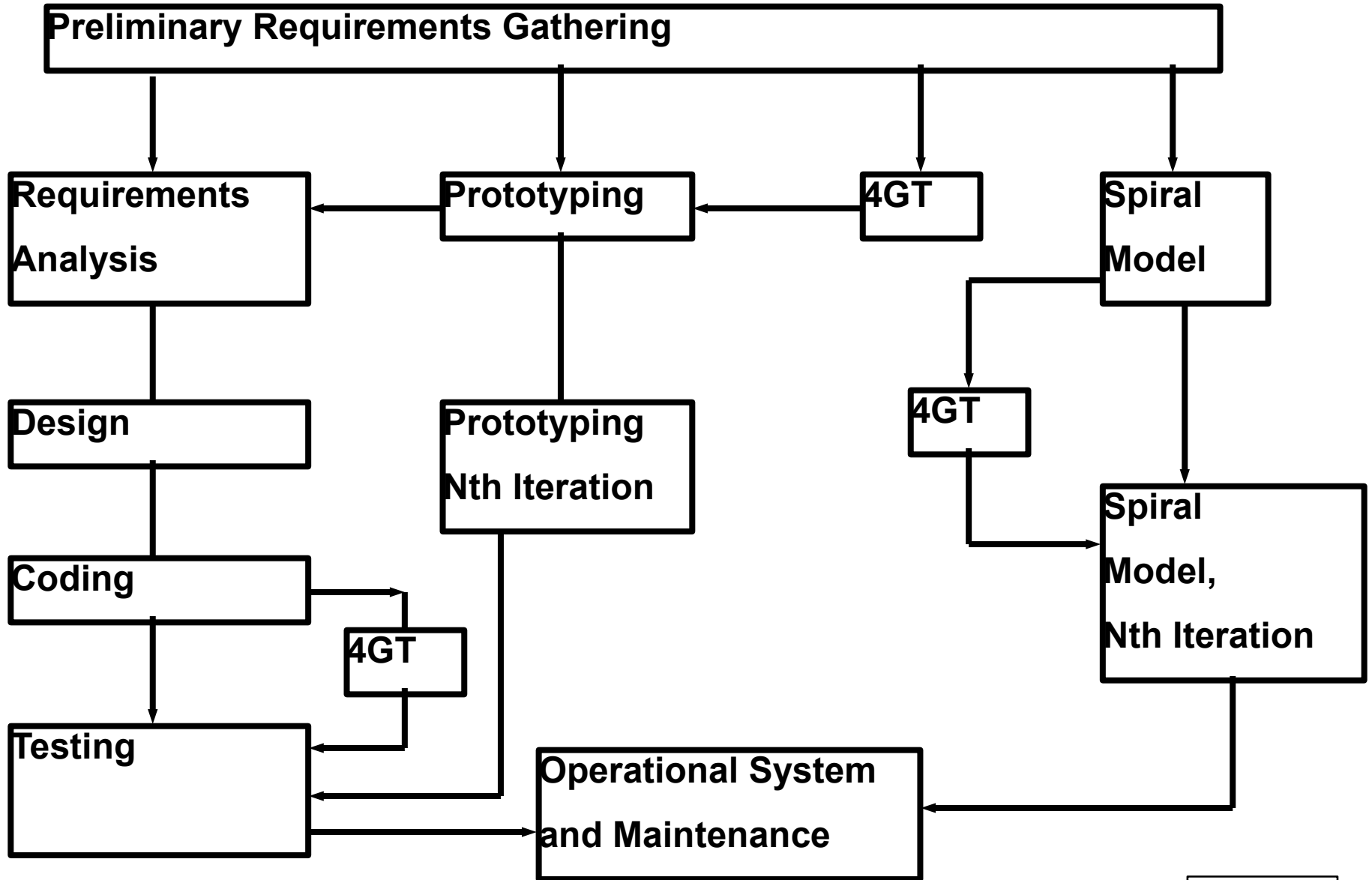
Spiral Model



Fourth Generation Techniques



Combining Paradigms



Generic Paradigm

1. DEFINITION PHASE

- System Analysis
- Software Project Planning
- Requirements Analysis

2. DEVELOPMENT PHASE

- Software Design
- Coding
- Software Testing

3. MAINTENANCE PHASE

- Correction
- Adaptation
- Enhancement

SOFTWARE ENGINEERING TECHNOLOGY

 **What is Software Engineering?**

 **Software Engineering Capability and Its
Measurement**

 **Ada Technology**

What Is Software Engineering?

Methods

 **Analysis**

 **Design**

 **Coding**

 **Testing**

 **Maintenance**

Procedures

 **Project Management**

 **Software Quality Assurance**

 **Software Configuration Management**

 **Measurement**

 **Tracking**

 **Innovative Technology Insertion**

Computer-Aided Software Engineering (CASE)

 **Tools which support the *Methods* and *Procedures***

Software Engineering Capability and Its Measurement

- The maturity of an organization's software engineering capability can be measured in terms of the degree to which the outcome of the process by which software is developed can be predicted.
- Predict the amount of time required to develop a software artifact
- Predict the resources (number of people, amount of disk space, etc.) required to develop a software artifact
- Predict the cost of developing a software artifact
- The *process* and the *technology* go hand in hand.
- One method of measurement is the *Capability Maturity Model for Software* developed by the Software Engineering Institute.

Increasing Process Maturity



Initial - Ad hoc;
unpredictable

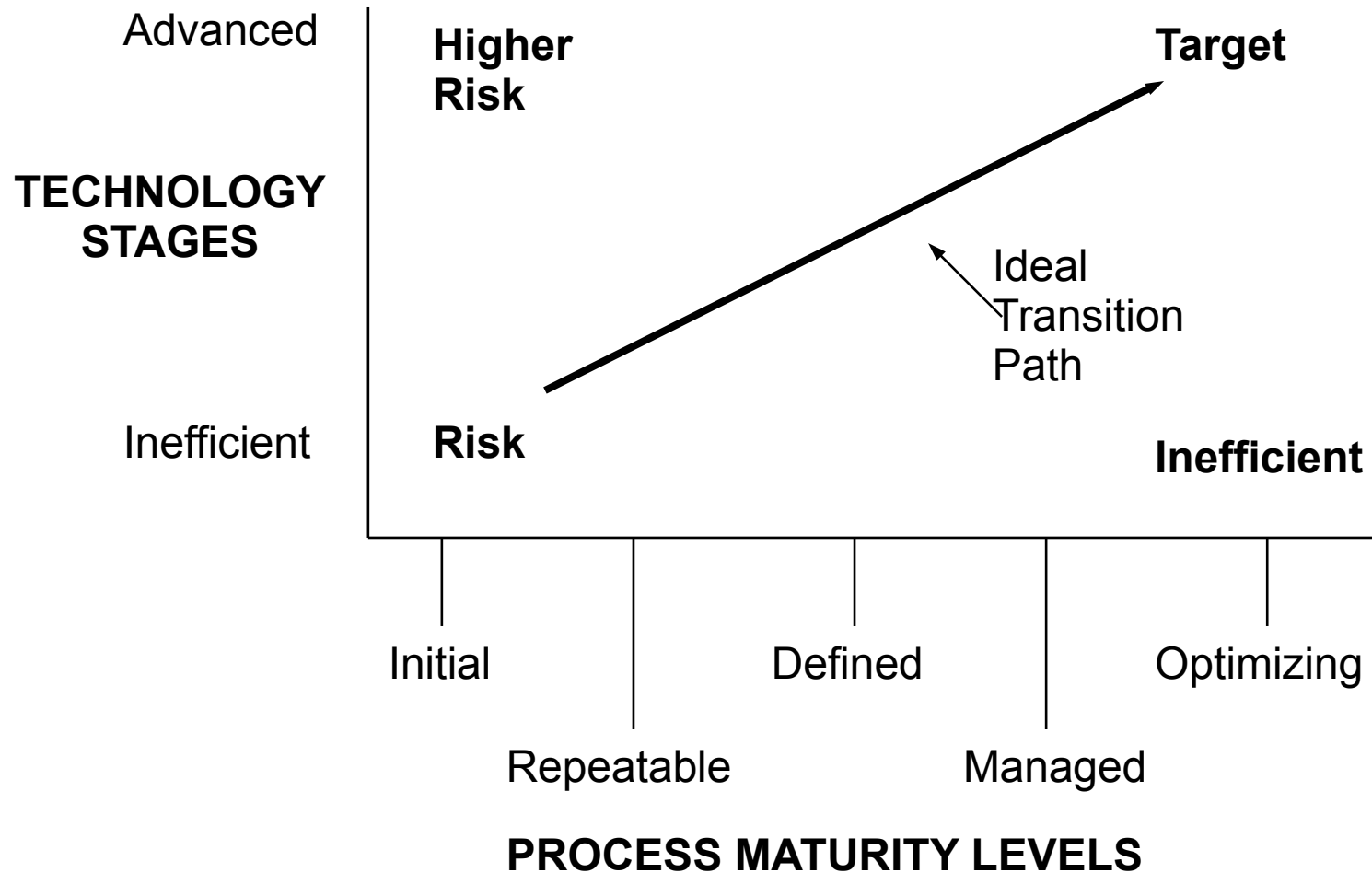
Repeatable - Costs,
Schedules managed

Defined - Process
institutionalized

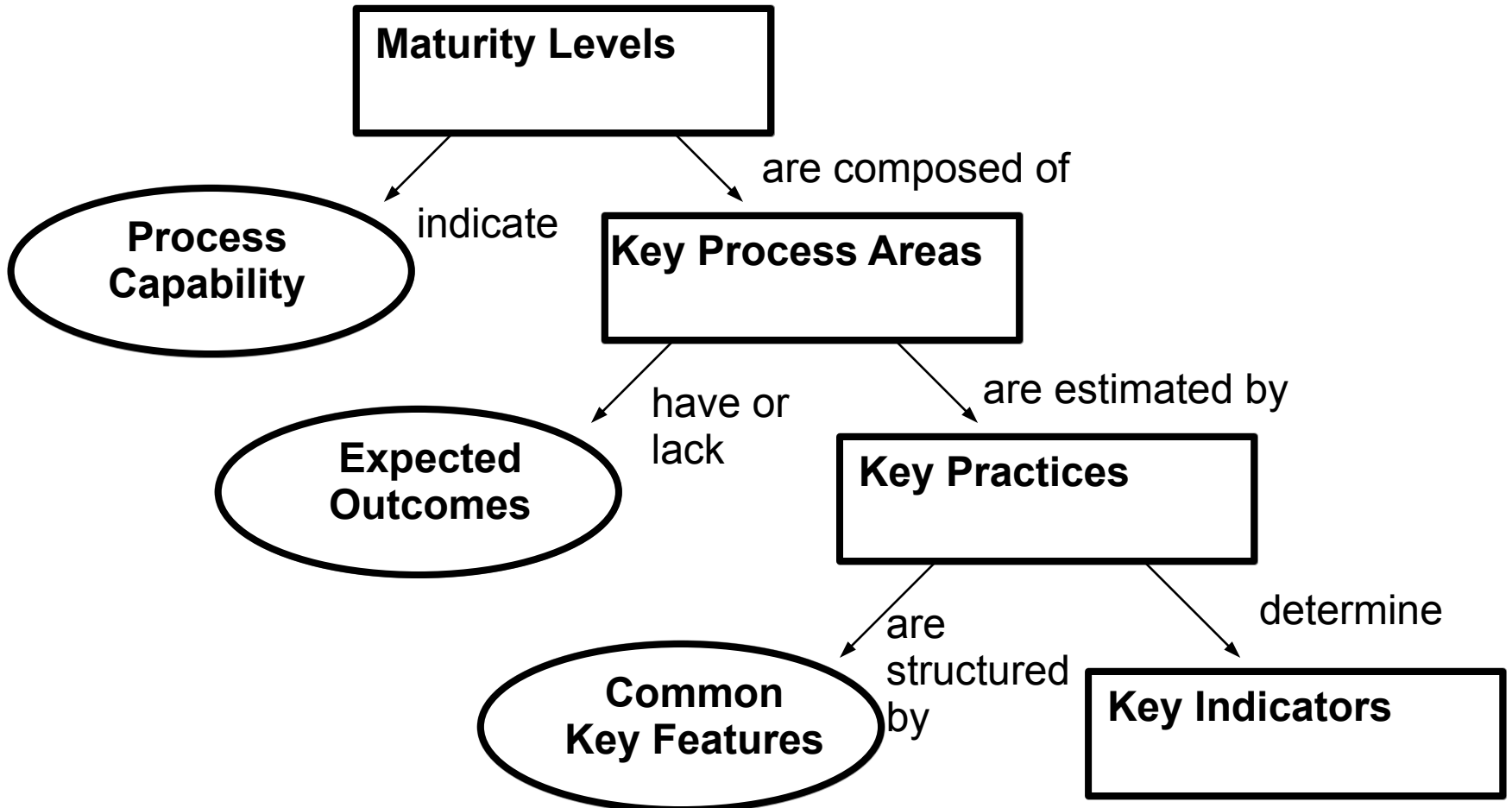
Managed - Process
measured/controlled

Optimizing - Process
refined constantly

Process Maturity and Technology



Maturity Keys



Key Process Areas by Level

Level 2 (Repeatable)

- Requirements Management**
- Software Project Planning**
- Software Project Tracking and Oversight**
- Software Subcontract Management**
- Software Quality Assurance**
- Software Configuration Management**

Key Process Areas by Level Level 2 (Repeatable), Continued

 Requirements Management

 Software Project Planning

 **Software Project Tracking and Oversight**

 **Software Subcontract Management**

 Software Quality Assurance

 Software Configuration Management

Key Process Areas by Level Level 2 (Repeatable), Continued

- Requirements Management
- Software Project Planning
- Software Project Tracking and Oversight
- Software Subcontract Management
- **Software Quality Assurance**
- **Software Configuration Management**

Key Process Areas by Level Level 3 (Defined)

- Organization Process Focus**
- Organization Process Definition**
- Training Program**
- Integrated Software Management**
- Software Product Engineering**
- Intergroup Coordination**
- Peer Reviews**

Key Process Areas by Level Level 3 (Defined), Continued

- ☐ Organization Process Focus
- ☐ Organization Process Definition
- ☐ **Training Program**
- ☐ **Integrated Software Management**
- ☐ Software Product Engineering
- ☐ Intergroup Coordination
- ☐ Peer Reviews

Key Process Areas by Level Level 3 (Defined), Continued

- ☐ Organization Process Focus
- ☐ Organization Process Definition
- ☐ Training Program
- ☐ Integrated Software Management
- ☐ **Software Product Engineering**
- ☐ **Intergroup Coordination**
- ☐ Peer Reviews

Key Process Areas by Level Level 3 (Defined), Continued

- Organization Process Focus
- Organization Process Definition
- Training Program
- Integrated Software Management
- Software Product Engineering
- Intergroup Coordination
- **Peer Reviews**

Key Process Areas by Level Level 4 (Managed)

 **Process Measurement and Analysis**

 **Quality Management**

Key Process Areas by Level Level 5 (Optimizing)

- Defect Prevention**
- Technology Innovation**
- Process Change Management**

Ada Technology

- **Ada** is a computer programming language specifically designed to support software engineering.
- Some of Ada's features include:
 - All of the normal constructs for looping, branching, flow control, and subprogram construction
 - Support for enumeration types, integers, floating point, fixed point, characters, strings, arrays, records, and user-defined data types
 - Support for algorithm templates (called generics) which allow algorithms to be expressed without concern for the kind of data on which the algorithm is applied
 - Support for interrupts and concurrent processing
 - Support for low-level control, such as memory allocation
- **Ada** is a *design* language as well as a *programming* language.
- **Ada** is designed to be read by Ada programmers and non-programmers.

Ada Technology, Continued

Ada
Specification



```
with System;

package Sensor is

  type Device is private;

  -- Abstract concept of a sensor

  procedure Define (S : in out Device;

    Where : in System.Address);

  -- Associate a sensor with memory

  function Read(S : in Device)

    return Integer;

  -- Return sensed value

private

  -- details omitted

end Sensor;
```

Ada Technology, Continued

From the software engineering perspective, Ada helps by acting as something much more than a programming language; Ada can be used as a common language for communicating:

Some aspects of the requirements

Some aspects of the design

All aspects of the code

In particular, by using Ada as a *design language*, code is simply realized as a complete, detailed elaboration of a design.

For large, multi-person teams, Ada can be used as an exact, precise way to communicate requirements and design information -- often in a form which may be syntactically checked by a compiler. Ada is much better than conventional English in this regard.