Software engineering: Architecture-driven Development

Richard Schmidt
Sirrush Corporation
703-919-8531

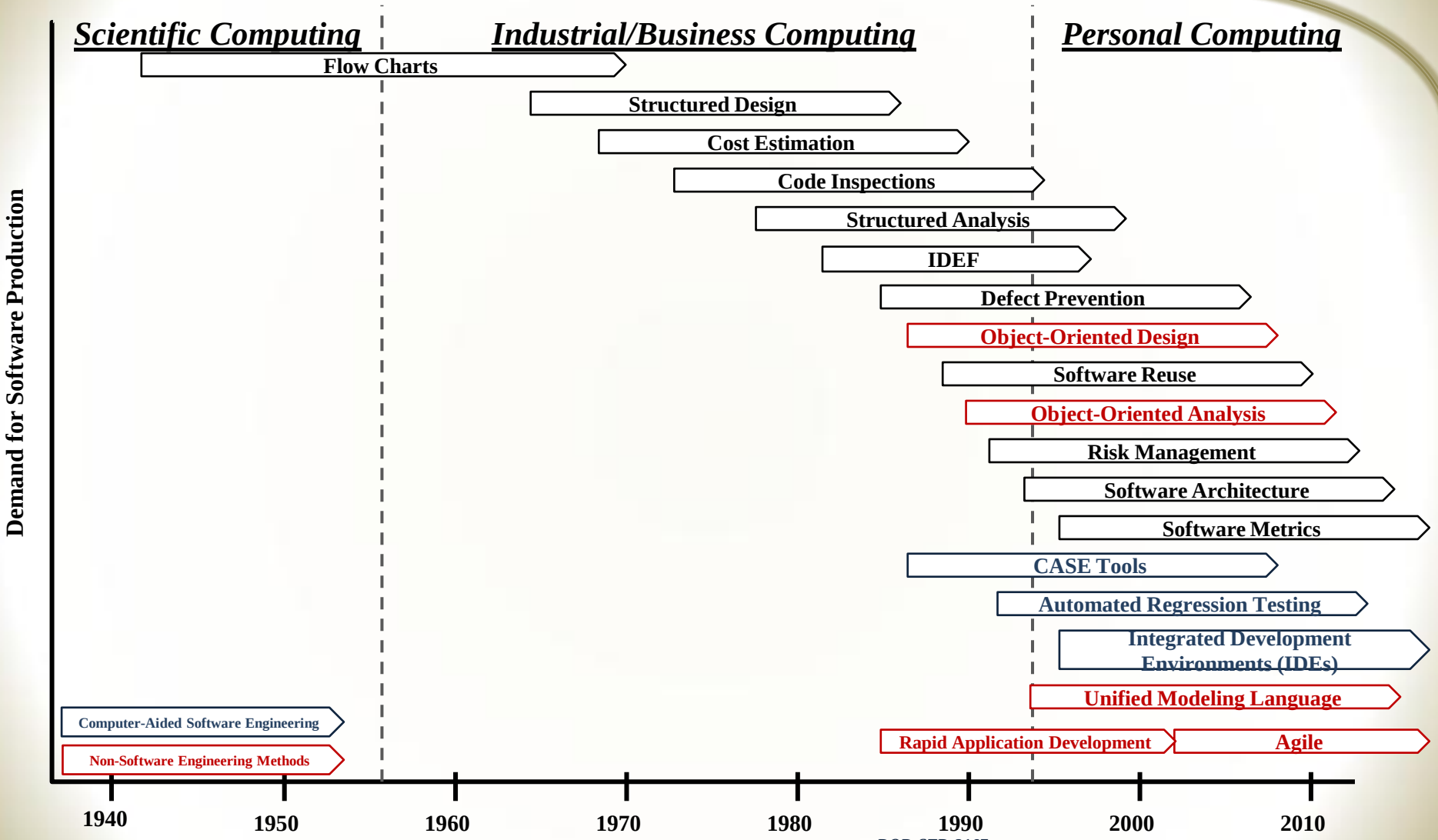
NDIA 15th Annual Systems Engineering Conference
Hyatt Regency Mission Bay
San Diego, California
October 24

Appears in the work Software Engineering, Architecture-Driven Development, published by Morgan Kaufmann, an imprint of Elsevier, Inc.

Overview

- Software Development *CHAOS*
- What is a Software Architecture
- How is a Software Architecture Developed
- Software Engineering Practices
- Software Architecture Design Strategy
- Relationship to Other Software Methodologies

Software Development Trends



Short History of Software Methods, By David F. Rico

Appears in the work Software Engineering, Architecture-Driven Development, published by Morgan Kaufmann, an imprint of Elsevier, Inc.

Chaos Reports

In the United States, we spend more than $250 billion each year on IT application development of approximately 175,000 projects. The average cost of a development project for a large company is $2,322,000; for a medium company, it is $1,331,000; and for a small company, it is $434,000. A great many of these projects will fail. Software development projects are in chaos, and we can no longer imitate the three monkeys -- hear no failures, see no failures, speak no failures.

	1994	1996	1998	2000	2002	2004	2006	2009
Successful	16%	27%	26%	28%	34%	29%	35%	32%
Challenged	53%	33%	46%	49%	51%	53%	46%	44%
Failed	31%	40%	28%	23%	15%	18%	19%	24%

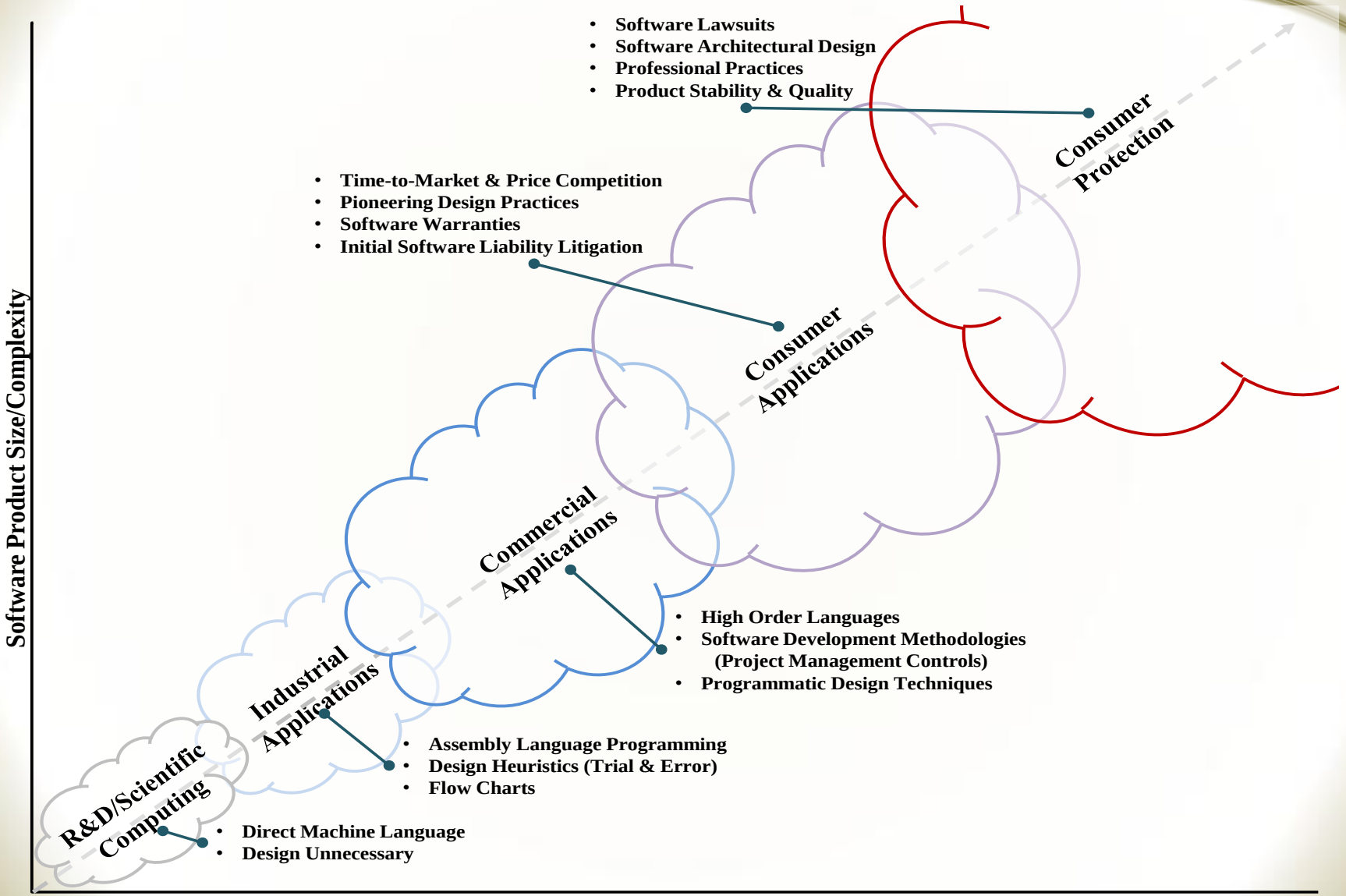
When a bridge falls down, it is investigated and a report is written on the cause of the failure. This is not so in the computer industry where failures are covered up, ignored, and/or rationalized. As a result, we keep making the same mistakes over and over again.

CHAOS, The Standish Group Report, 1995

Why Such CHAOS?

- Computer technology's rapid transition into Industrial, Commercial & Consumer systems/products
- Majority of Software R&D
  - Initially Programming Language focused (1950-1985)
  - Programming productivity focused (1985-2010)
- Software development project management emphasis on documentation
  - Inadequate design methodologies
  - Software Professional untrained in "product" design
- Software workforce demand exceeded availability of skilled professionals
- Variety of software application domains
- No sponsored research to establish formal software design practices

Software As a Critical Material



Appears in the work Software Engineering, Architecture-Driven Development, published by Morgan Kaufmann, an imprint of Elsevier, Inc.

Applying Systems Engineering Practices

"The Government's present system for procuring software does not meet the Government's needs and wastes resources. The application of **"systems engineering"** disciplines is needed to remedy the procurement system's defects. . . Software Development is a complex process that requires modern **"systems engineering"** techniques."

Bugs In The Program,
Problems in Federal Government Computer Software Development and Regulation,
Staff Study by the Subcommittee on Investigations and Oversight,
Congress, September 1989

What are **"systems engineering"** disciplines ?

How can **"systems engineering"** be adapted to development of software product?

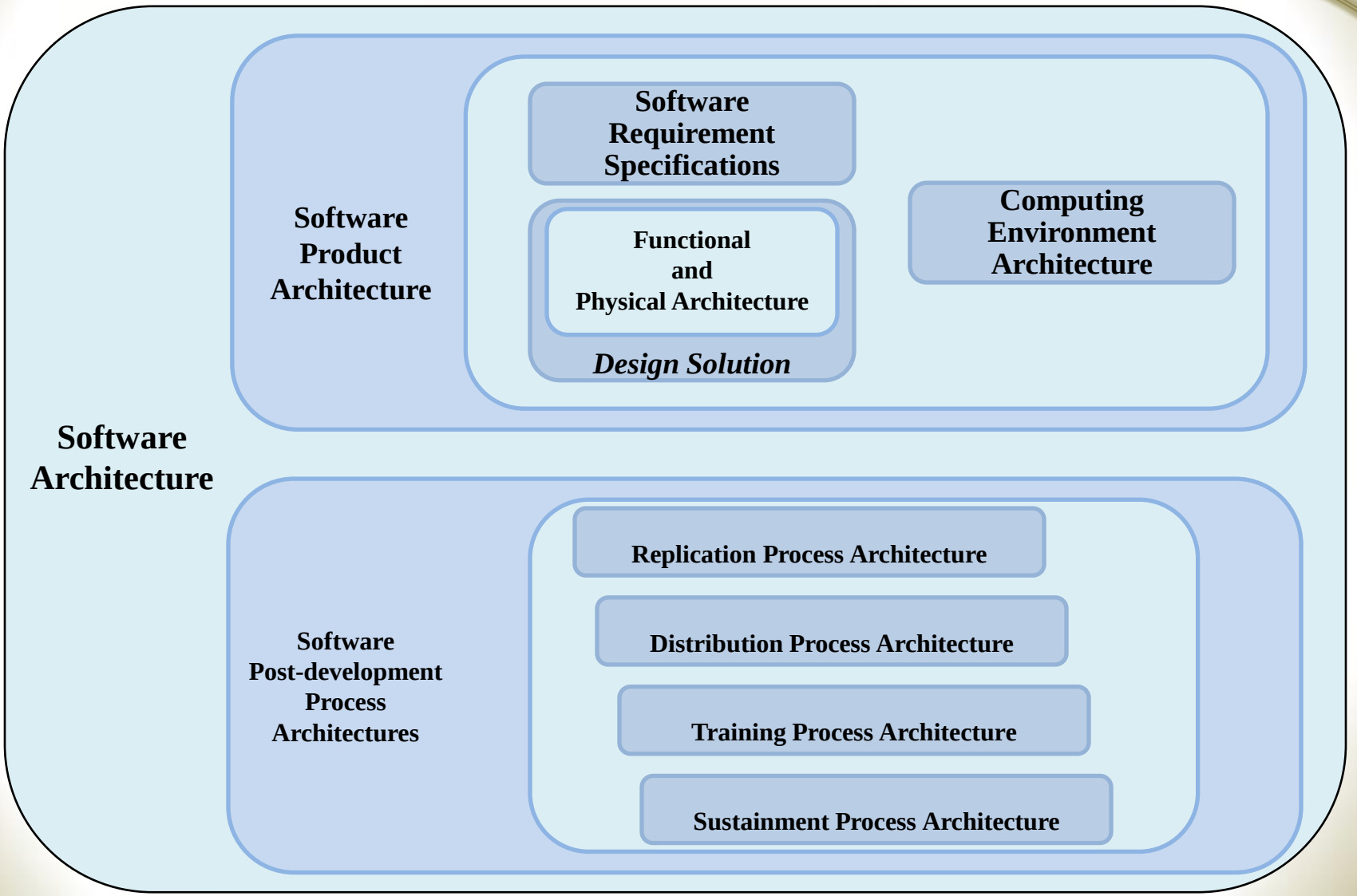
Appears in the work Software Engineering, Architecture-Driven Development, published by Morgan Kaufmann, an imprint of Elsevier, Inc.

What is Architecture-driven Development?

- Establishing a Requirements Baseline
  - Balance the needs and expectations of all stakeholders
  - Provides a basis for *DESIGNING* the software product
  - Establishes the basis for software acceptance testing
- Establishment of a comprehensive software product design
  - Functional basis for ensuring product performance
  - Structural basis for software implementation
- Software Post-development Processes Specifications
- Full traceability throughout the software architecture
  - Software Specifications, Functional Specifications, Physical Specifications
- Basis for continual planning and resource allocation
- Architectural Design Decisions
  - Risk-based decision-making
  - Focus on project success criteria

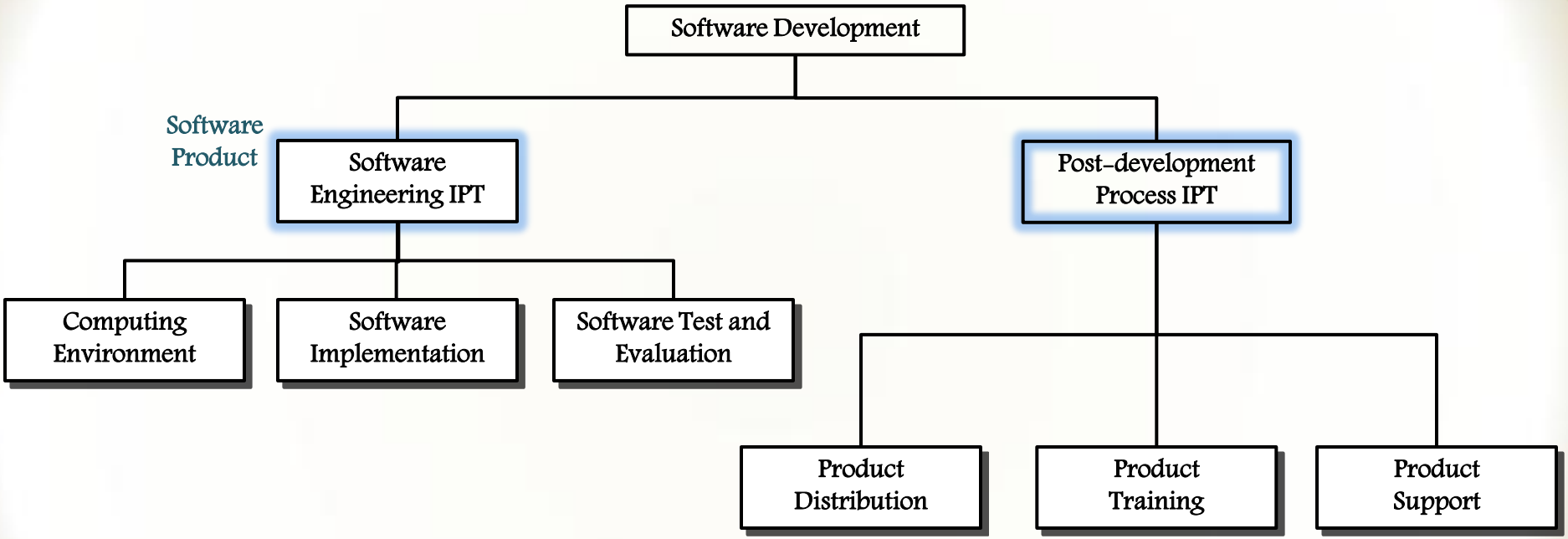
Appears in the work Software Engineering, Architecture-Driven Development, published by Morgan Kaufmann, an imprint of Elsevier, Inc.

Software Architecture



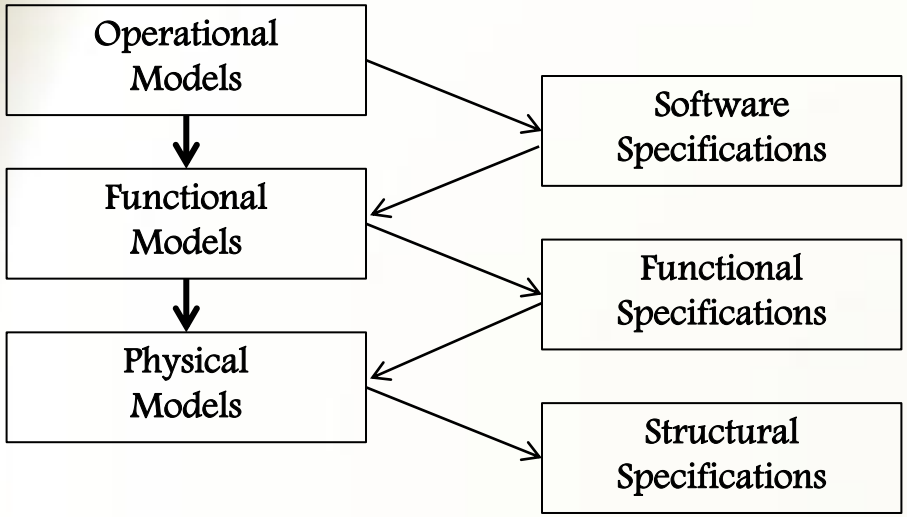
Appears in the work Software Engineering, Architecture-Driven Development, published by Morgan Kaufmann, an imprint of Elsevier, Inc.

Organizing for Software Engineering



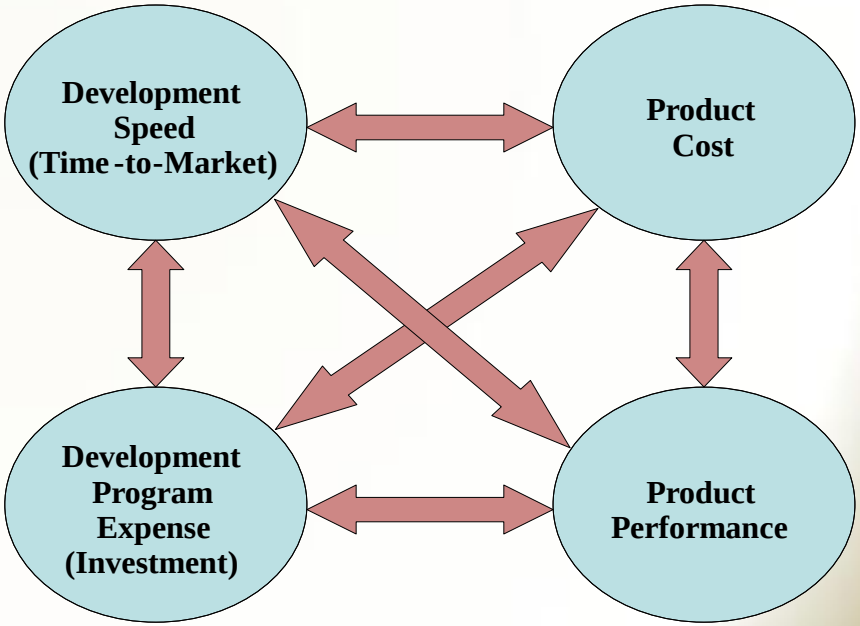
Value of Software Architecture

Provides specifications for every software module, routine or class



Objective architectural decision-making

- Project objectives
- Resource constraints
- Technical challenges
- Risk aware



Basis for Technical and Project Planning

- Software Breakdown Structure
- Work Packages & Dependencies
- Resource Allocations
- Integrated Technical Planning
- Integrated project Planning

Appears in the work Software Engineering, Architecture-Driven Development, published by Morgan Kaufmann, an imprint of Elsevier, Inc.

Developing The Software Architecture

- Business Needs and Expectations
- Operational Concepts

Requirements Baseline

Software Requirements Analysis

- Operational Model
- Software Requirement Specifications

Functional Analysis and Allocation

Software Analysis
- Trade-off Analysis
- Risk Assessment
- Complexity Assessment

Functional Architecture

- Functional Behavior Model
- Functional Hierarchy
- Functional Specifications

Software Design Synthesis

Physical Architecture

- Structural Unit Specifications
- Structural Component Specifications
- Component Integration Strategy

Appears in the work Software Engineering, Architecture-Driven Development, published by Morgan Kaufmann, an imprint of Elsevier, Inc.

Design Decisions

- It's not the *Decision* that matters – It's the **Rationale**
- **Decision** implies a choice among multiple alternatives
- The important architectural decisions affect software product life-cycle characteristics:
  - Complexity
  - Supportability
  - Extensibility
  - Usability
  - Product Life-cycle Costs
- Architectural Decisions must align technical scope of work with availability of project resources

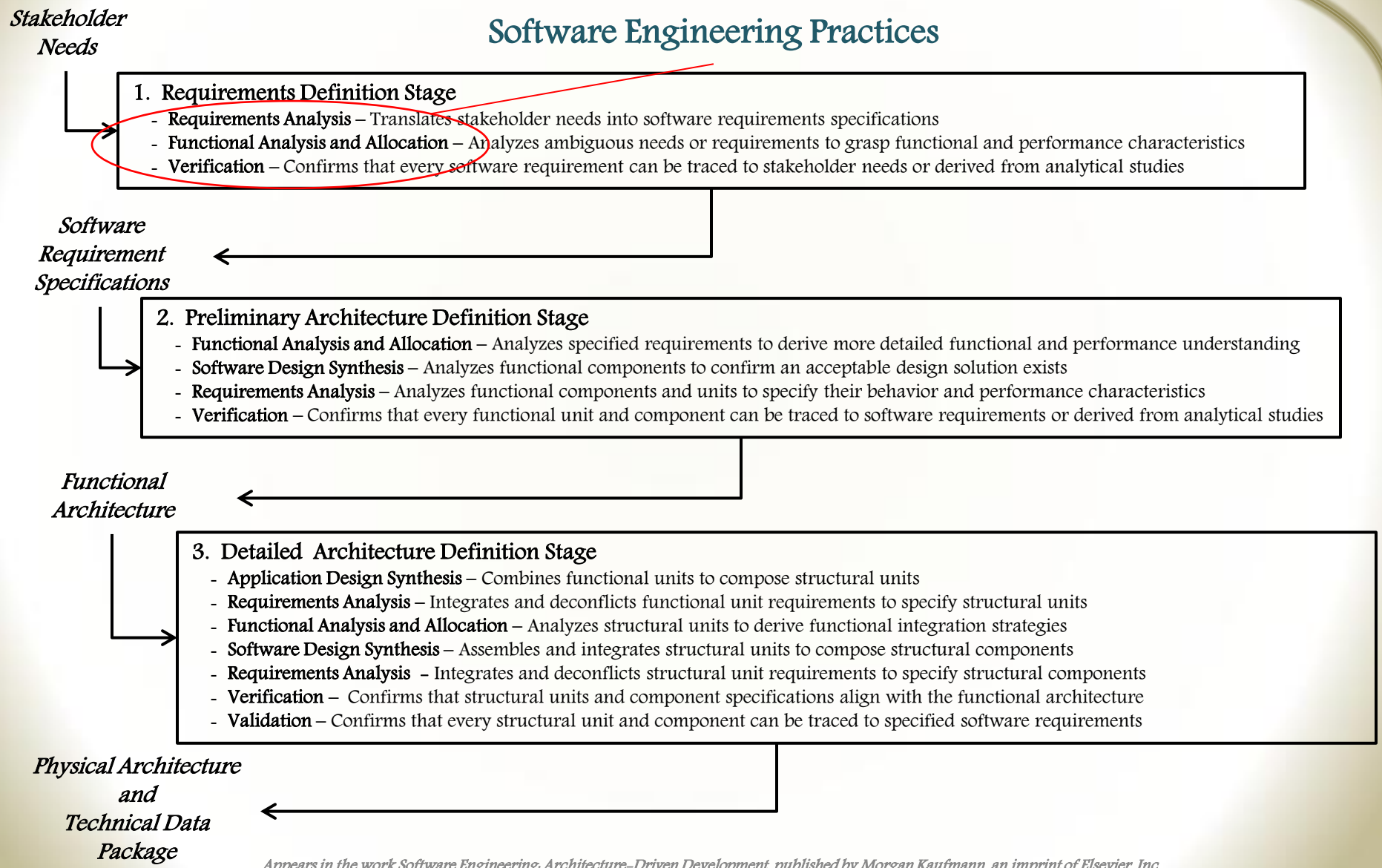
Technical Imperatives

Project Objectives

Technical and Project Risks

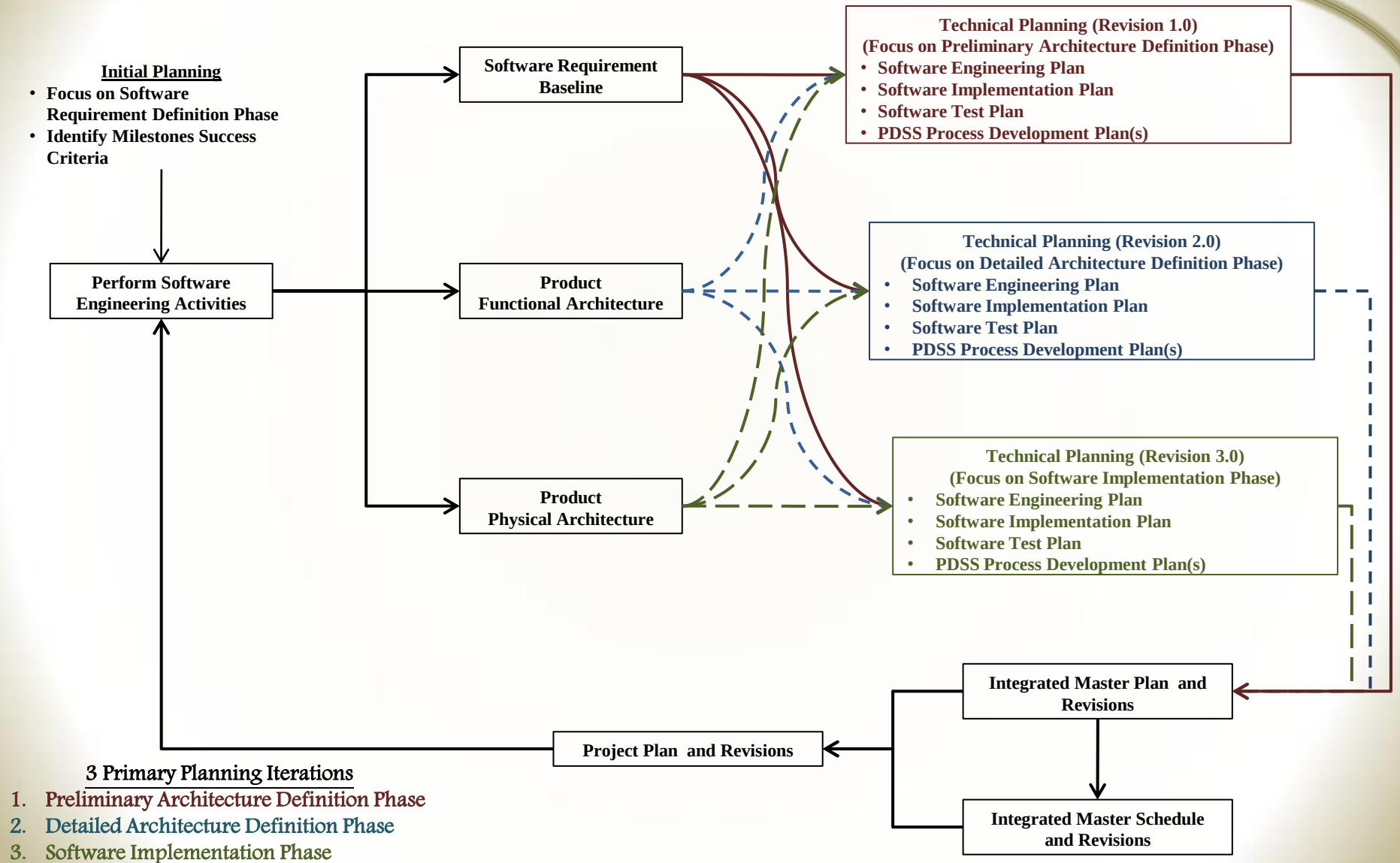
- Software development is a technical effort
- All technical challenges impact project feasibility
- All risk to a software development project is technical in nature
  - Insufficient resources should imply a less robust software product
  - Complexity must be simplified
  - Over-stated requirement must be challenged
- Software prototyping should be used to assess technical solution feasibility
  - Never put a prototype on the *CRITICAL PATH*

Deriving the Software Architecture



Appears in the work Software Engineering, Architecture-Driven Development, published by Morgan Kaufmann, an imprint of Elsevier, Inc.

Planning the Software Engineering Effort

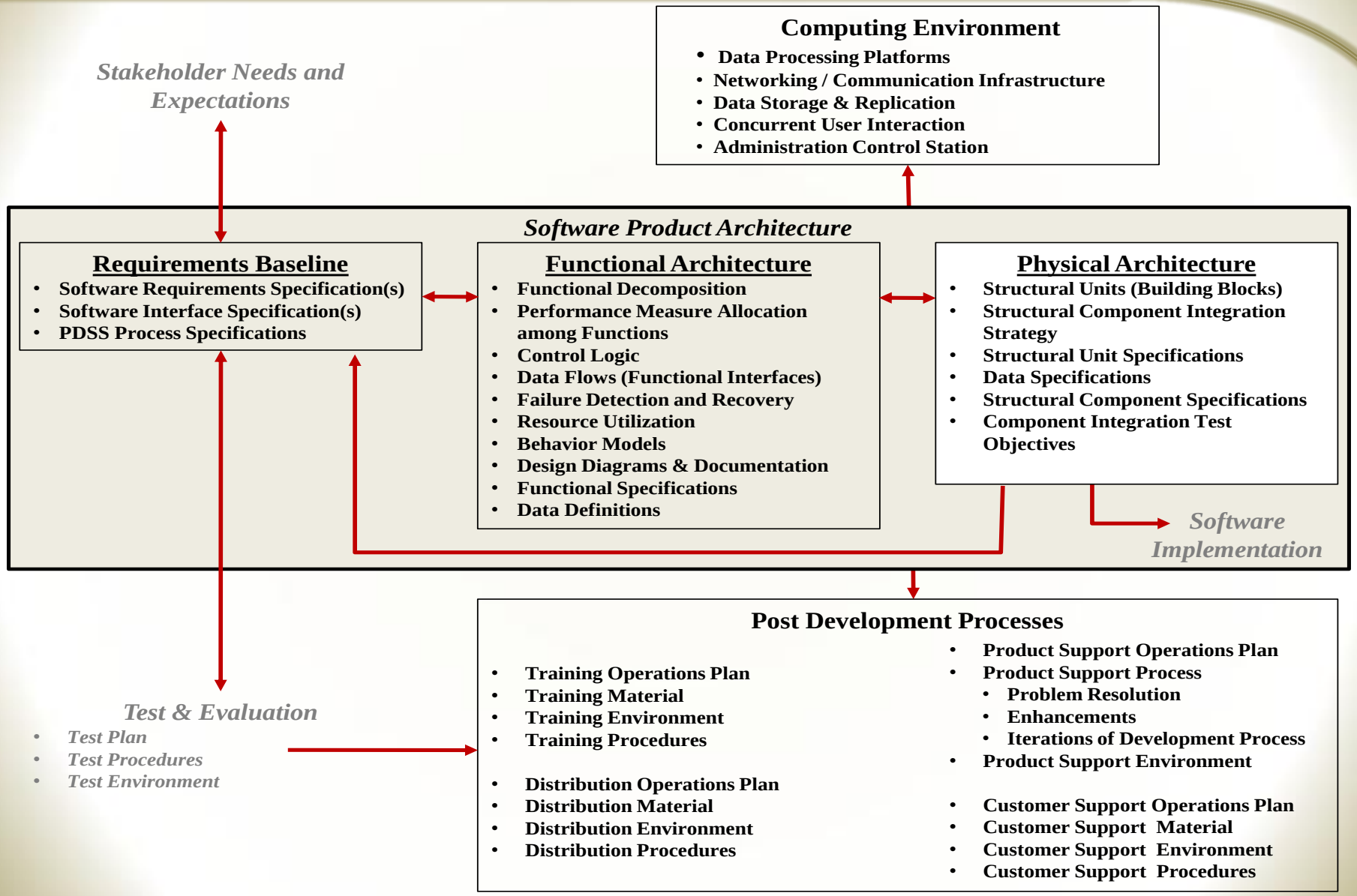


3 Primary Planning Iterations

1. Preliminary Architecture Definition Phase
2. Detailed Architecture Definition Phase
3. Software Implementation Phase

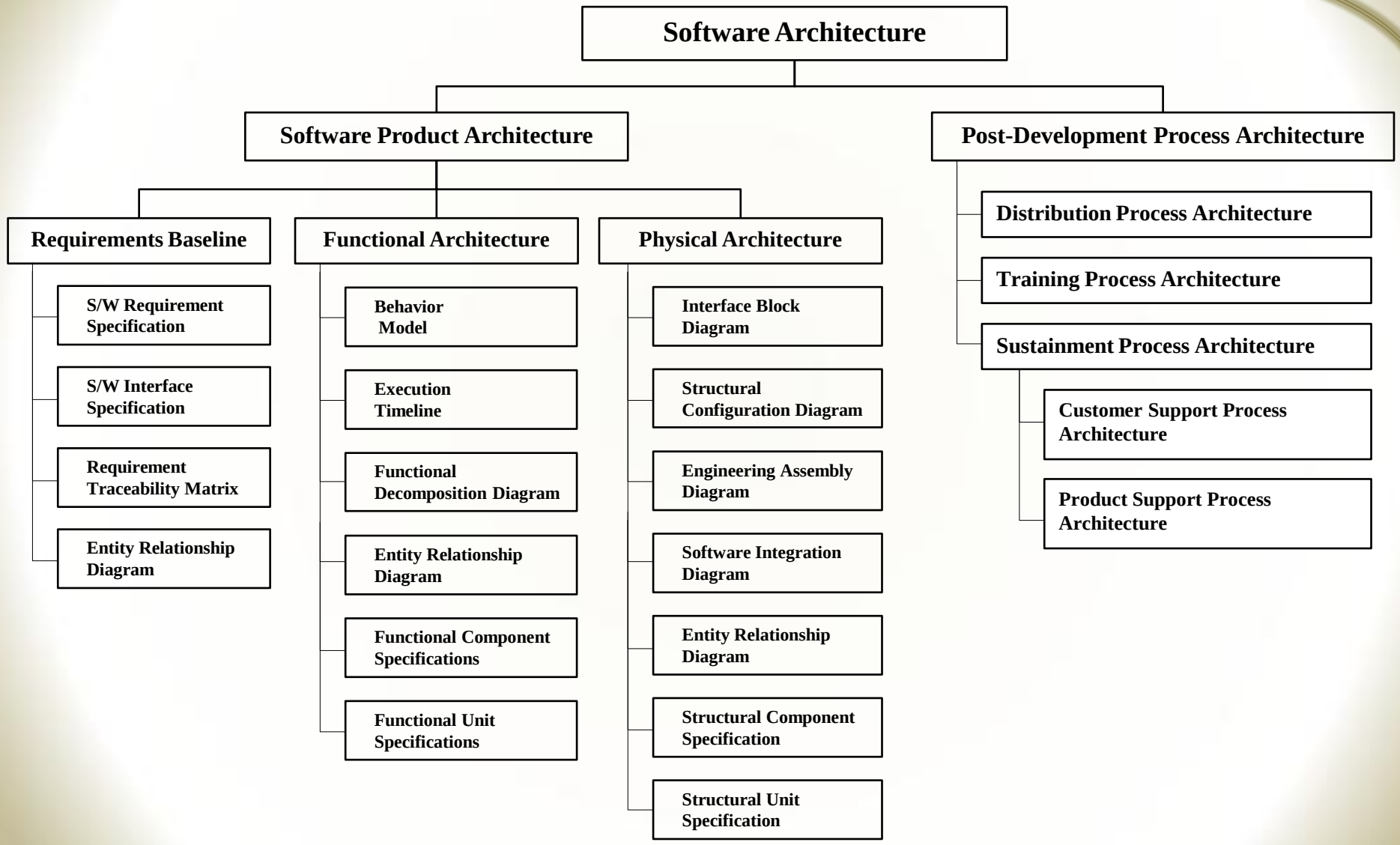
Appears in the work Software Engineering, Architecture-Driven Development, published by Morgan Kaufmann, an imprint of Elsevier, Inc.

Elements of Software Architecture



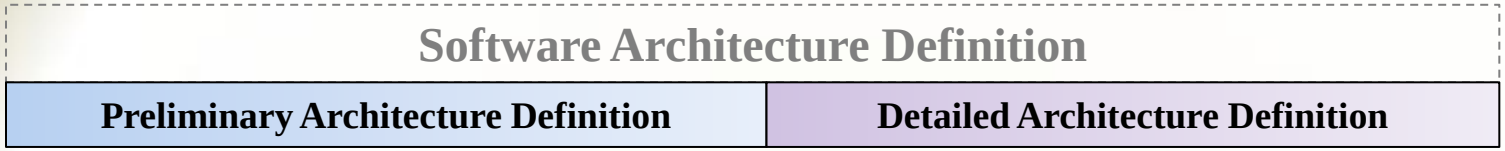
Appears in the work Software Engineering, Architecture-Driven Development, published by Morgan Kaufmann, an imprint of Elsevier, Inc.

Artifacts of Software Architecture



Appears in the work Software Engineering, Architecture-Driven Development, published by Morgan Kaufmann, an imprint of Elsevier, Inc.

Software Architecture Definition



PDR
Preliminary Design Review

CDR
Critical Design Review

Products of Preliminary Architecture Definition:

- Behavior Model (Data Processing Transactions)
- Functional Hierarchy
- Conceptual Design Structure
- Functional Component Specifications
- Functional Unit Specifications
- Database Transaction Specifications
- Interface Transaction Specifications (Protocols & Messaging)
- User Interface Functional Hierarchy

Products of Critical Architecture Definition:

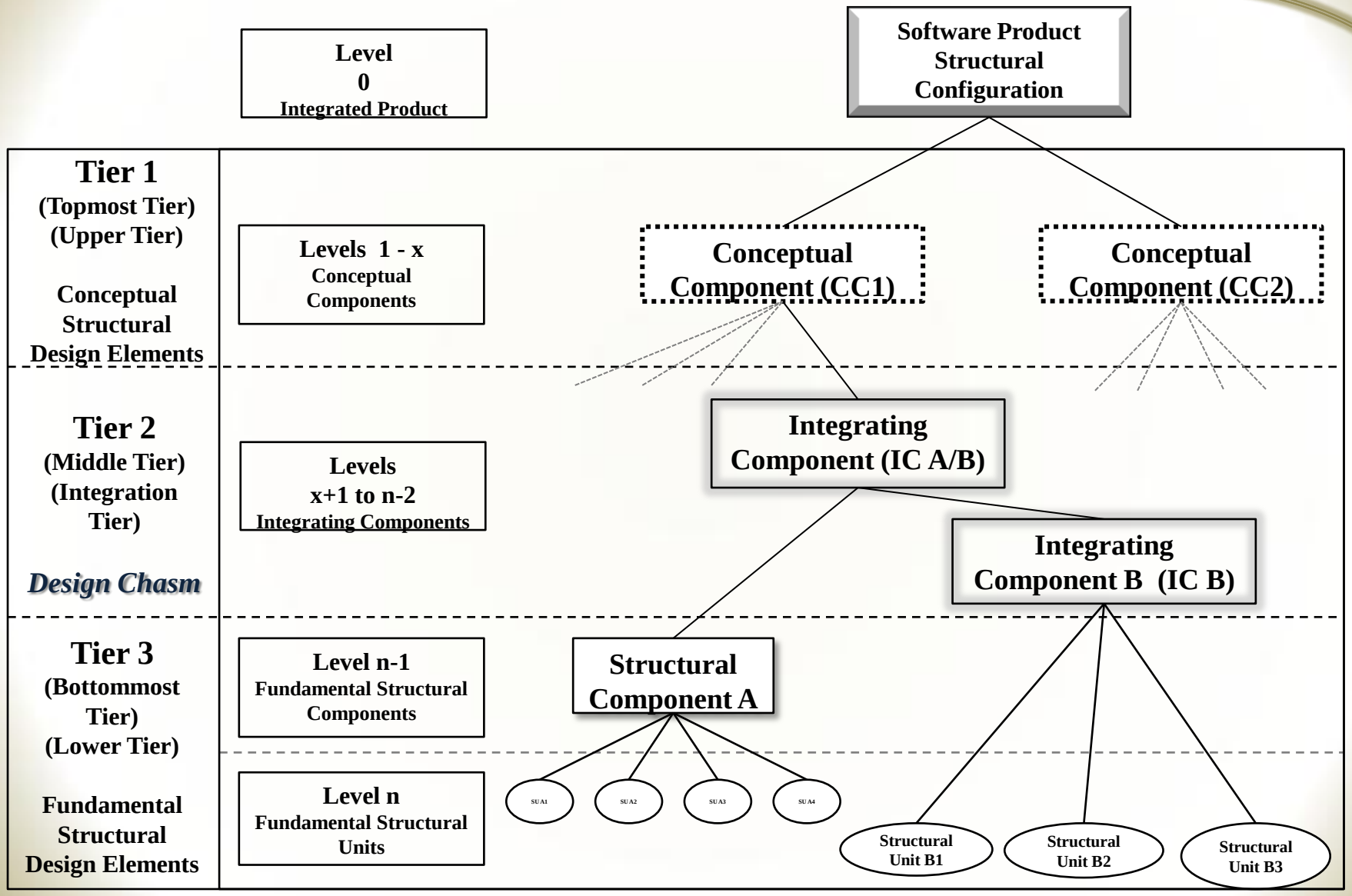
- Structural Unit Specifications
- Software Integration Strategy
- Structural Component Specifications (Integrated Behaviors)
- Interface Design Documents
- Structural Block Diagrams
- Structural Interface
- User Interface Structural Hierarchy

- **Preliminary Functional Architecture**
- **Initial Structural Design Concept**
- **Preliminary Test Procedures**
- **Updated Requirement Traceability Matrix**

- **Revised Functional Architecture**
- **Completed Physical Architecture**
- **Initial GUI Structural Design**
- **Detailed Test Procedures**
- **Updated Requirement Traceability Matrix**

Appears in the work Software Engineering, Architecture-Driven Development, published by Morgan Kaufmann, an imprint of Elsevier, Inc.

Software Design Chasm



Appears in the work Software Engineering, Architecture-Driven Development, published by Morgan Kaufmann, an imprint of Elsevier, Inc.

Software Product Performance

Software Architecture Definition		Software Implementation	
Preliminary Architecture Definition	**Detailed Architecture Definition**	**Unit Implementation**	**Component Integration & Testing**
Establish resource utilization functional specifications: • Allocate resources among functions • Identify resource supervision behaviors	**Establish resource utilization structural specifications:** • Behavioral thread profiles • Structural component specification • Structural unit specification (if desired) • Identify engineering assembly resource utilization stub specifications	**Implement resource utilization requirements:** • Design units with efficient object creation & destruction mechanisms • Implement connection & object pools	**Assess resource utilization :** • Design units with efficient object creation & destruction mechanisms • Implement connection & object pools

Establish the computing resource utilization strategy

- Software design and coding guidelines
- Identify task prioritization strategy
- Identify multi-tasking scheduling strategy
- Identify garbage collection strategy
- Identify resource queuing strategy

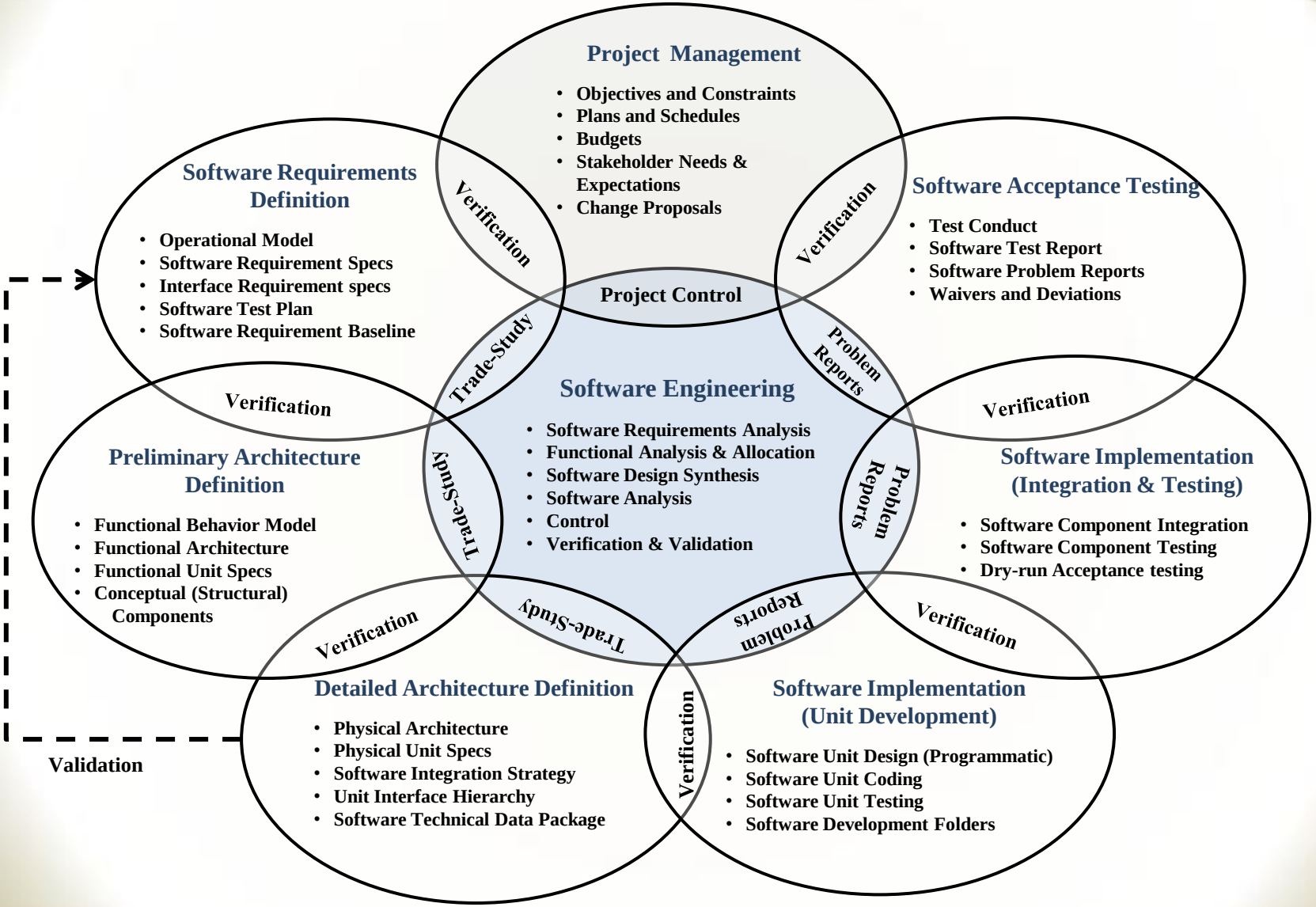
Measure computing resource utilization

- Software unit resource consumption and conservation (average & worst-case)
- Integrated component resource consumption and conservation
- Integrated product consumption and conservation

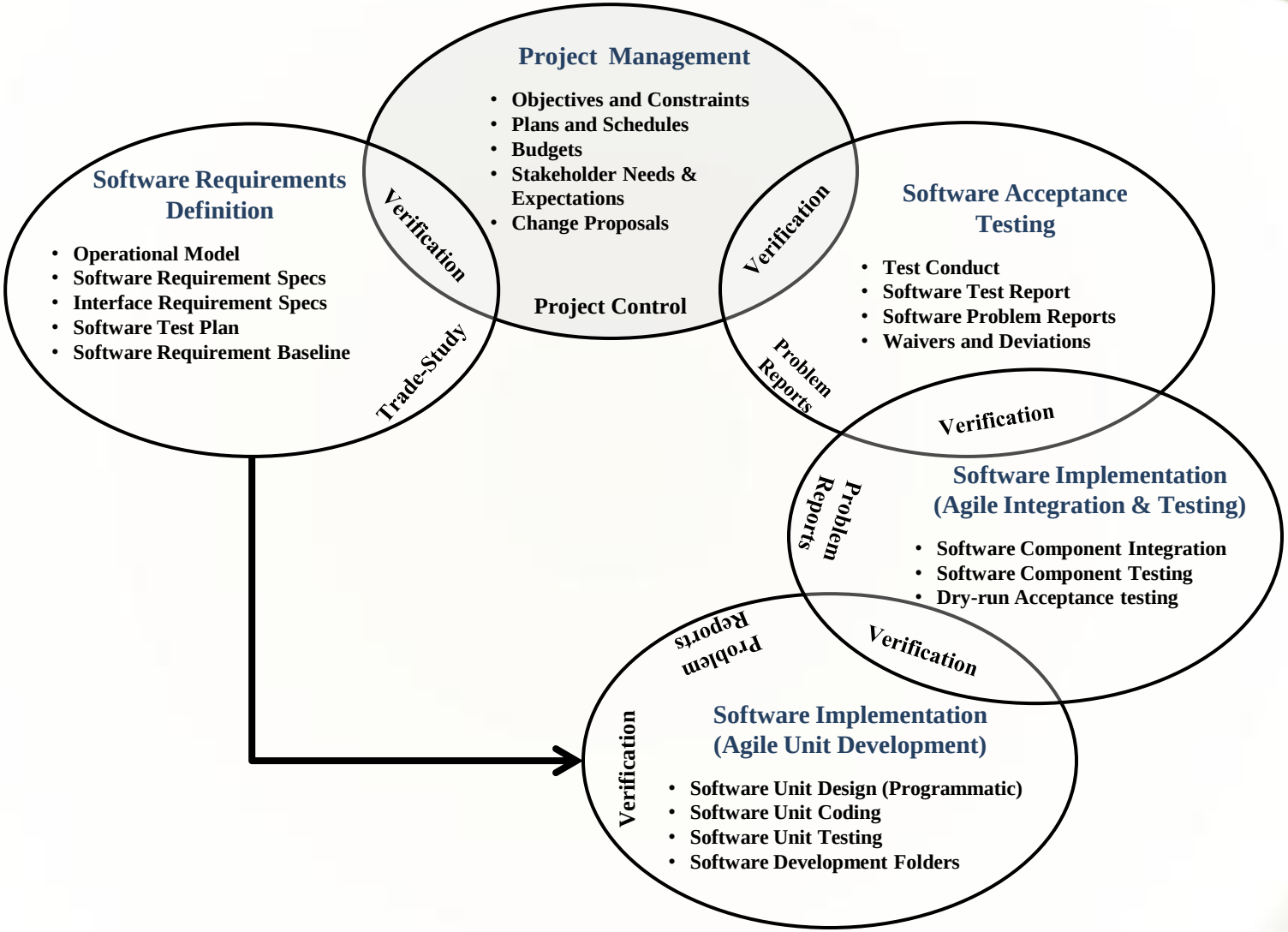
Software performance is predicated on the performance of the Computing Environment

Appears in the work Software Engineering, Architecture-Driven Development, published by Morgan Kaufmann, an imprint of Elsevier, Inc.

Software Engineering Practices

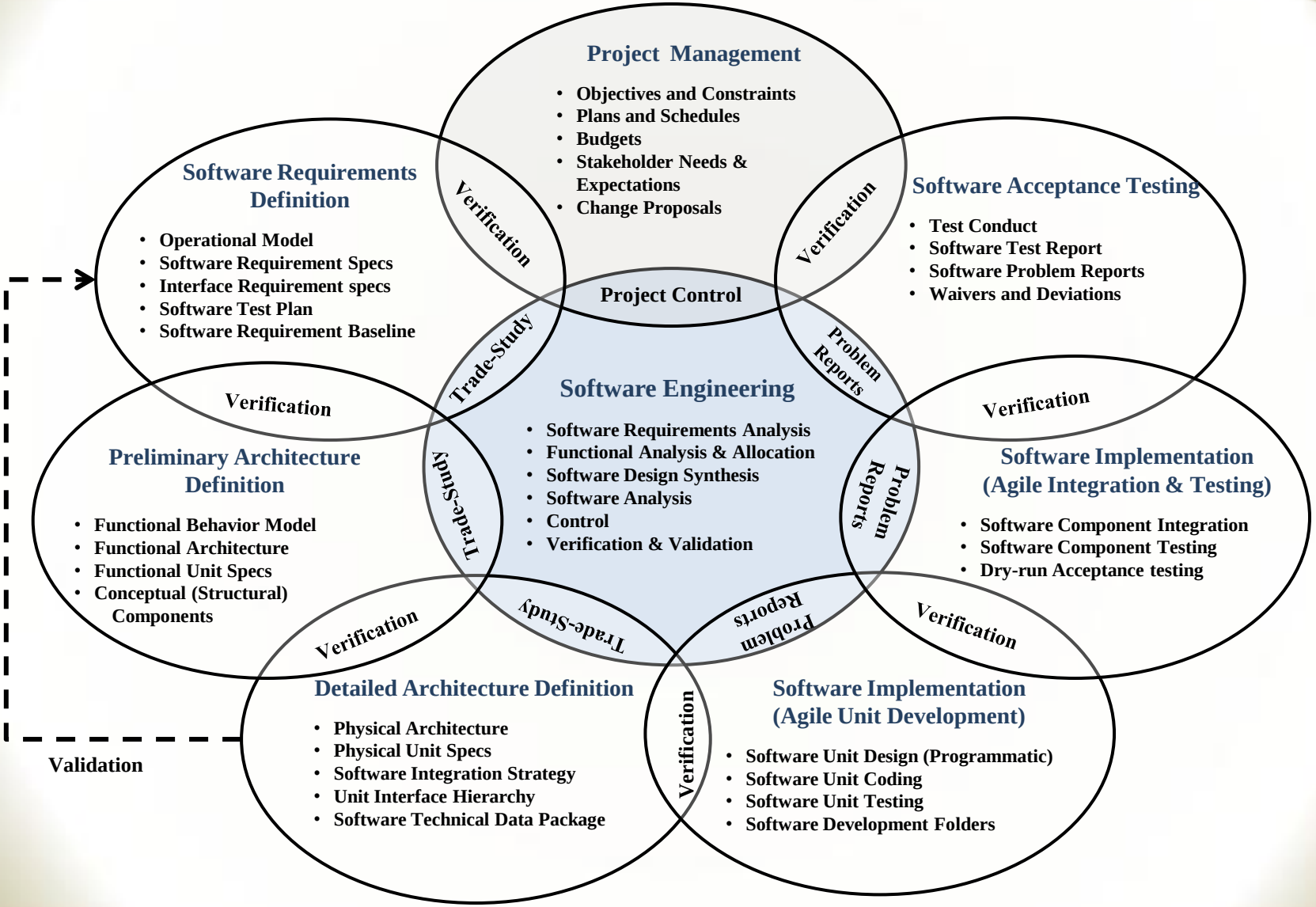


Appears in the work Software Engineering, Architecture-Driven Development, published by Morgan Kaufmann, an imprint of Elsevier, Inc.



Appears in the work Software Engineering, Architecture-Driven Development, published by Morgan Kaufmann, an imprint of Elsevier, Inc.

Architecture-driven Software Development



Appears in the work Software Engineering, Architecture-Driven Development, published by Morgan Kaufmann, an imprint of Elsevier, Inc.

Summary

- Software Industry is in CHAOS
- Computer technology's rapid growth and employment have prevented software engineering practices from arising
- Software Product Complexity must be corralled
- Application of Systems Engineering Practices is a viable solution
- Architecture-driven approach improves Software Development Probability of Project Success
- Software Functional Decomposition must be complete to enable a bottom-up structural design solution
- Software Methodologies rely on Programmatic Design & Coding (Prototyping) – Hence CHAOS!
- Software Engineering is the little brother of Systems Engineering
  - Software as a "material" offers unique challenges!

Appears in the work Software Engineering, Architecture-Driven Development, published by Morgan Kaufmann, an imprint of Elsevier, Inc.