The *M*ethod-*Framework* for *E*ngineering *S*ystem *A*rchitectures (MFESA)

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Donald Firesmith

System and Software Technology Conference (SSTC)
Salt Lake City, Utah
20-23 April 2009

© 2009 Carnegie Mellon University

Tutorial Objectives

Introduce attendees to the Method Framework for Engineering System Architectures (MFESA):

- MFESA *Ontology* of reusable concepts and terminology
- MFESA *Metamodel* of reusable method components
- MFESA *Repository* of reusable method components:
  - MFESA Architectural Work Units and Work Products
  - MFESA Architectural Workers
- MFESA *Metamethod* for generating appropriate project-specific system architecture engineering methods

Thereby improve the attendees' system architecture engineering methods and associated processes (process improvement)

© 2009 Carnegie Mellon University

MFESA Project

Started January 2007

Collaborators:

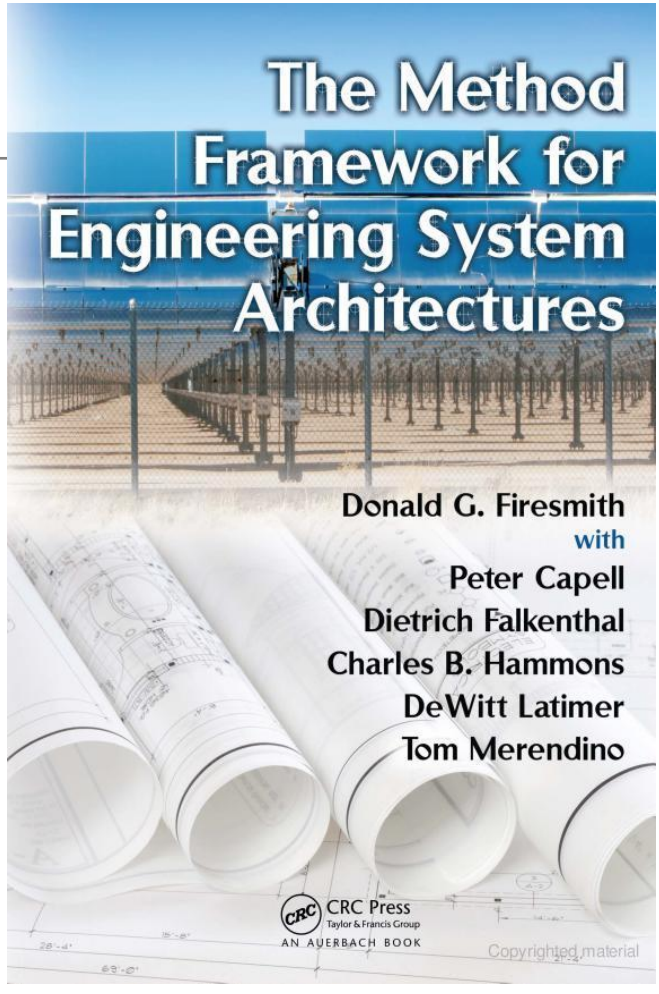
- SEI Acquisition Support Program (ASP) – Don Firesmith (Team Lead), Peter Capell, Bud Hammons, and Tom Merendino
- MITRE – Dietrich Falkenthal (Bedford MA)
- USAF – DeWitt Latimer (USC)

Current work products:

- Reference Book (CRC Press – Auerbach Publishing, November 2008)
- Tutorials and Training Materials
- Articles

Eventual work products (we hope!):

- Informational website with method components and associated tools



© 2009 Carnegie Mellon University

Intended Tutorial Attendees

System and Subsystem Architects

Process Engineers

Requirements Engineers

Technical and Administrative Managers

Acquirers

Developers

Testers

Trainers and Educators

Standards Developers

Academic Researchers

Any other Stakeholders

© 2009 Carnegie Mellon University

Topics

Motivation

MFESA Overview

MFESA Ontology of Concepts and Terminology

MFESA Metamodel of Reusable Method Components

MFESA Repository of Reusable Method Components

- Architectural Work Units and Work Products
- Architectural Workers

MFESA Metamethod

Conclusion

© 2009 Carnegie Mellon University

System Architecture – Traditional Definition

System Architecture

> the organization of a system including its major components, the relationships between them, how they collaborate to meet system requirements, and principles guiding their design and evolution

Note that this definition is primarily oriented about the system's structure.

Yet systems have many static and dynamic logical and physical structures.

© 2009 Carnegie Mellon University

System Architecture – MFESA Definition

System Architecture

all of the ***most important, pervasive, top-level, strategic decisions, inventions, engineering tradeoffs, assumptions***, and their associated ***rationales*** concerning ***how*** the system will meet its derived and allocated requirements

Includes:

- All major logical and physical and static and dynamic structures
- Other architectural decisions, inventions, tradeoffs, assumptions, and rationales:
  - Approach to meet quality requirements
  - Approach to meet data and interface requirements
  - Architectural styles, patterns, mechanisms
  - Approach to reuse (build/buy decisions)
- *Strategic* and *pervasive* design-level decisions
- *Strategic* and *pervasive* implementation-level decisions

© 2009 Carnegie Mellon University

Architecture vs. Design

Architecture	Design
Pervasive **(Multiple Components)**	***Local*** **(Single Components)**
Strategic **Decisions and Inventions**	***Tactical*** **Decisions and Inventions**
Higher-Levels **of System**	***Lower-Levels*** **of System**
Huge Impact **on Quality, Cost, & Schedule**	***Small Impact*** **on Quality, Cost, & Schedule**
Drives **Design and Integration Testing**	***Drives*** **Implementation and Unit Testing**
Driven by **Requirements and Higher-Level Architecture**	***Driven by*** **Requirements, Architecture, and Higher-Level Design**
Mirrors **Top-Level Development Team Organization (Conway's Law)**	***No*** **Impact on Top-Level Development Team Organization**

© 2009 Carnegie Mellon University

System Architecture Engineering

System Architecture Engineering

the ***subdiscipline of systems engineering*** consisting of all ***architectural work units*** performed by ***architectural workers*** (architects, architecture teams, and their tools) to develop and maintain ***architectural work products*** (including system or subsystem architectures and their representations)

© 2009 Carnegie Mellon University

System Architecture is Critical

Supports achievement of critical architecturally significant requirements

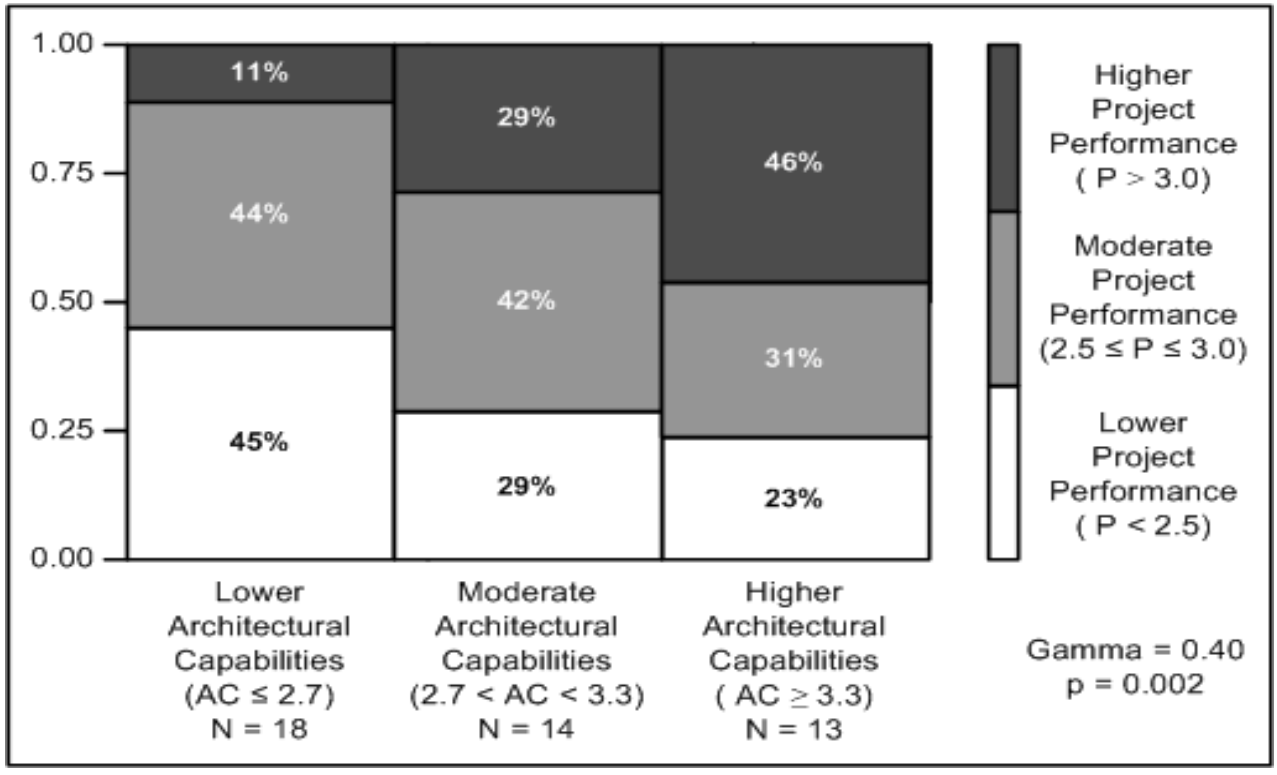
Greatly affects cost and schedule

Enables engineering of system quality characteristics and attributes

Drives all downstream activities

© 2009 Carnegie Mellon University

System Architecture Engineering is critical to Project Success



Joe Elm, Dennis R. Goldenson, Khaled El Emam, Nicole Donatelli, and Angelica Neisa, *A Survey of Systems Engineering Effectiveness – Initial Results*, CMU/SEI-2007-SR-014, Software Engineering Institute, November 2007, p. 222.

© 2009 Carnegie Mellon University

Limitations of Current Methods and Standards

Do not adequately address:

- The increasing size and complexity of many current systems
- All types of architectural components (e.g., software)
- All types of interfaces (interoperability and intraoperability)
- All potentially important system structures, views, models, and other architectural representations
- All life cycle phases (production, evolution, and maintenance of architectural integrity)
- System quality characteristics, attributes, and requirements
- Reuse and Component-Based Development (CBD)
- Specialty engineering areas (such as safety and security)

© 2009 Carnegie Mellon University

More Limitations of Current Methods and Standards$_2$

Current methods:

- Overemphasize two structures:
  - Static logical functional decomposition view
  - Static physical aggregation decomposition view
- Are weak on structure, view, and model consistency.
- Confuse requirements engineering with architecture engineering.
- Tend to assume that *One Size Fits All*.
- Produce only a single architectural vision.
- Excessively emphasize architectural models over other architectural representations.

© 2009 Carnegie Mellon University

Architecture Engineering Challenges$_1$

How good is ‘Good enough’?

We lack sufficient adequately trained and experienced architects.

- Many young architects must perform tasks for which many are under qualified.

Architects may use multiple inconsistent architecture engineering methods.

Architecture engineering methods are often incomplete or incompletely documented.

Architects can rely too much on architectural engineering tools.

© 2009 Carnegie Mellon University

Architecture Engineering Challenges$_2$

Different stakeholders have different and possibly conflicting needs for different architectural representations at different levels of abstractions:

- **Requirements Engineers** – Ensure architecturally significant (e.g., quality) requirements are properly engineered
- **Architects** – Capture and convey their architecture to themselves, other architects, and other stakeholders
- **Designer** and **Implementers** – Constrain designs and implementations
- **Specialty Engineers** – ensure architecture supports specialty engineering requirements and incorporates related patterns/mechanisms.
- **Testers** – Integration tests and whitebox system and component testing
- **Manufacturers** – Producibility of the system given its architecture
- **Acquirers** and **Funders** – Understand what is being acquired and paid for
- **Managers** – Manage development and Conway's Law
- **Certifiers**, **Accreditors**, and **Regulators** – Ensure system will be able to be safely and securely operated

© 2009 Carnegie Mellon University

Why Method Engineering? – *Systems* Vary Greatly

Size (small through ultra-large-scale)

Complexity

Autonomy of subsystems (useful, self-contained, not controlled by others)

Criticality (business, safety, and security of system and individual subsystems)

Domains (such as aviation, telecommunications, weapons)

Driven by requirements (top-down) or subsystem availability (bottom-up)

Emergent behavior and characteristics (necessary, beneficial, foreseeable)

Geographical distribution of subsystems

© 2009 Carnegie Mellon University

Why Method Engineering? – *Systems* Vary Greatly$_2$

Homogeneity/heterogeneity of subsystems

Intelligence

Operational dependence on other systems

Reconfigurability (adding, replacing, or removing subsystems)

Relative amounts of hardware, software, people, facilities, manual procedures, …

Requirements (existence, volatility, quality characteristics and attributes, constraints)

Self-regulation (proactive vs. reactive, homeostasis)

Synergism/independence of subsystems

Technologies used (including diversity, maturity, and volatility)

© 2009 Carnegie Mellon University

Why Method Engineering? – *Organizations* Vary Greatly

Number of organizations

Size of organizations

Types of organizations:

- Owner, Acquirer, Developer, Operator, User, Maintainer
- Prime contractor, subcontractors, vendors, system integrator

Degree of centralized/distributed governance:

- Authority, policy, funding, scheduling
- Directed, Acknowledged, Collaborative, or Virtual

Management and engineering culture

Geographical distribution

Staff expertise and experience

© 2009 Carnegie Mellon University

Why Method Engineering? – Endeavors Vary Greatly

Type (project, program of projects, enterprise)

Contracting:

- Formality
- Type (e.g., fixed-price or cost plus fixed fee)

Lifecycle scope (development, sustainment)

System scope (subsystem, system, “system of systems”)

Schedule (adequacy, criticality, coordination)

Funding (adequacy, distribution)

© 2009 Carnegie Mellon University

Why Method Engineering? – Stakeholders Vary Greatly

Type of stakeholders:

- Acquirer, developer, maintainer, member of the public, operator, regulator, safety/security accreditor/certifier, subject matter expert, user, …

Number of stakeholders

Authority (requirements, funding, policy, …)

Accessibility of the stakeholders to the architecture teams

Volatility of stakeholder turnover (especially acquirers)

Motivation and needs

© 2009 Carnegie Mellon University

Why Method Engineering? – Bottom Line

No *single* system architecture engineering method is sufficiently general and tailorable to meet the needs of all endeavors.

Method engineering enables the creation of appropriate, system/organization/endeavor/stakeholder-specific architecture engineering methods.

© 2009 Carnegie Mellon University

Topics

Motivation

MFESA Overview

MFESA Ontology of Concepts and Terminology

MFESA Metamodel of Reusable Method Components

MFESA Repository of Reusable Method Components

- **Architectural Work Units and Work Products**
- **Architectural Workers**

MFESA Metamethod

Conclusion

© 2009 Carnegie Mellon University

Definition

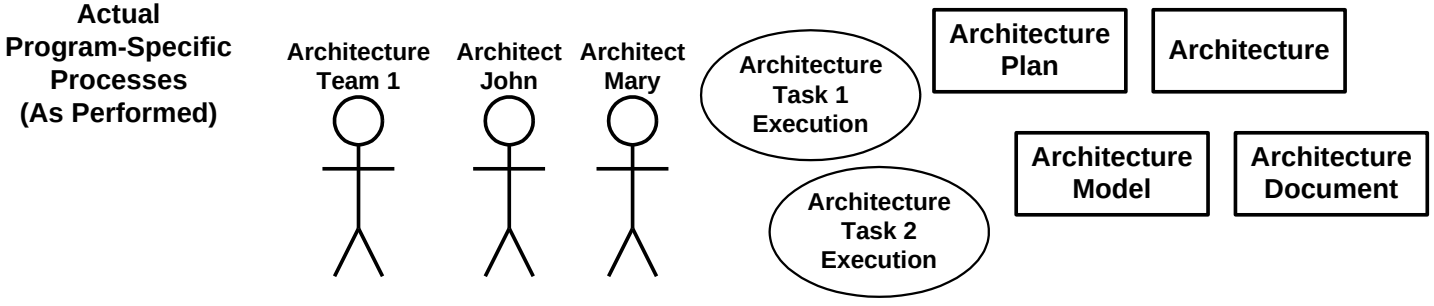
Method-Framework for Engineering System Architectures (MFESA)

> a method framework for engineering appropriate situation-specific system architecture engineering (SAE) methods

MFESA is not a single system architecture engineering method.

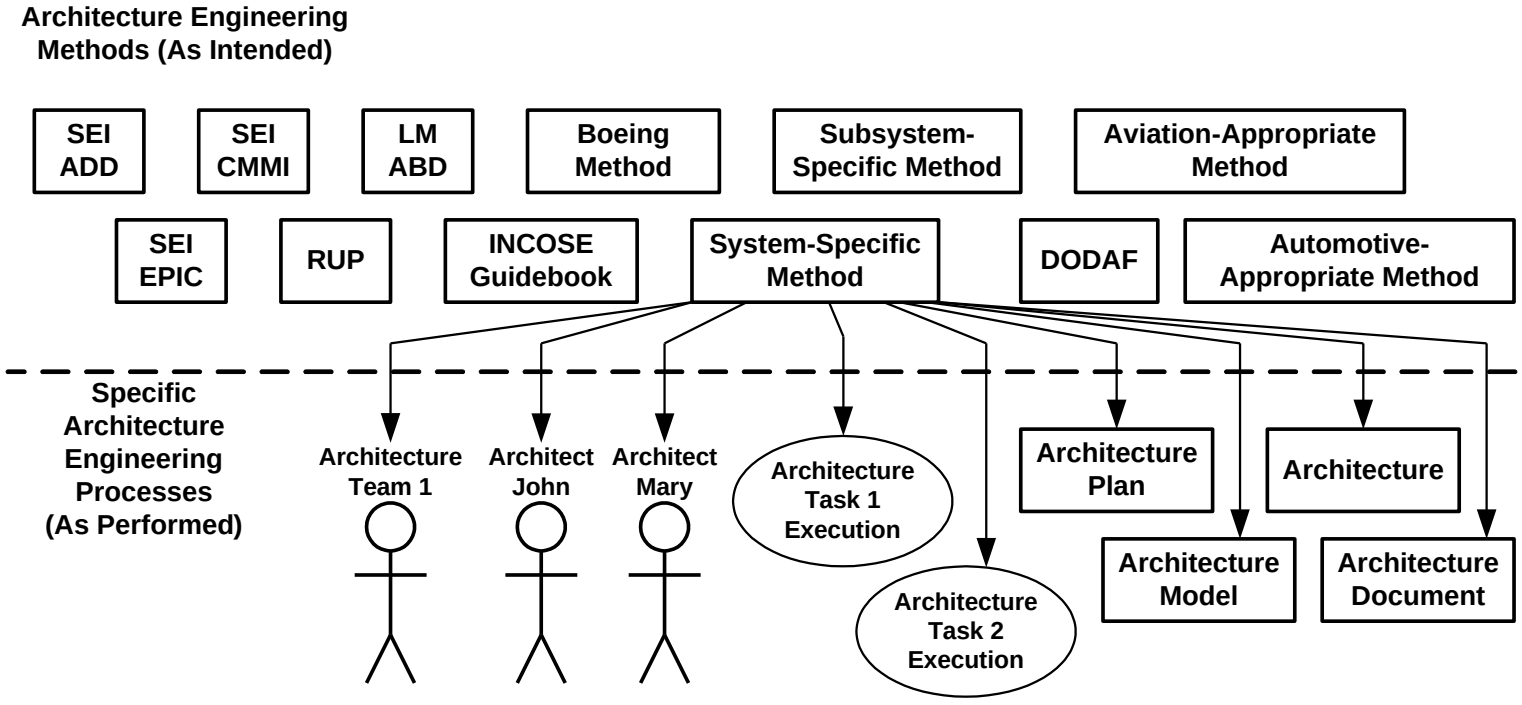
© 2009 Carnegie Mellon University

As-Performed Processes



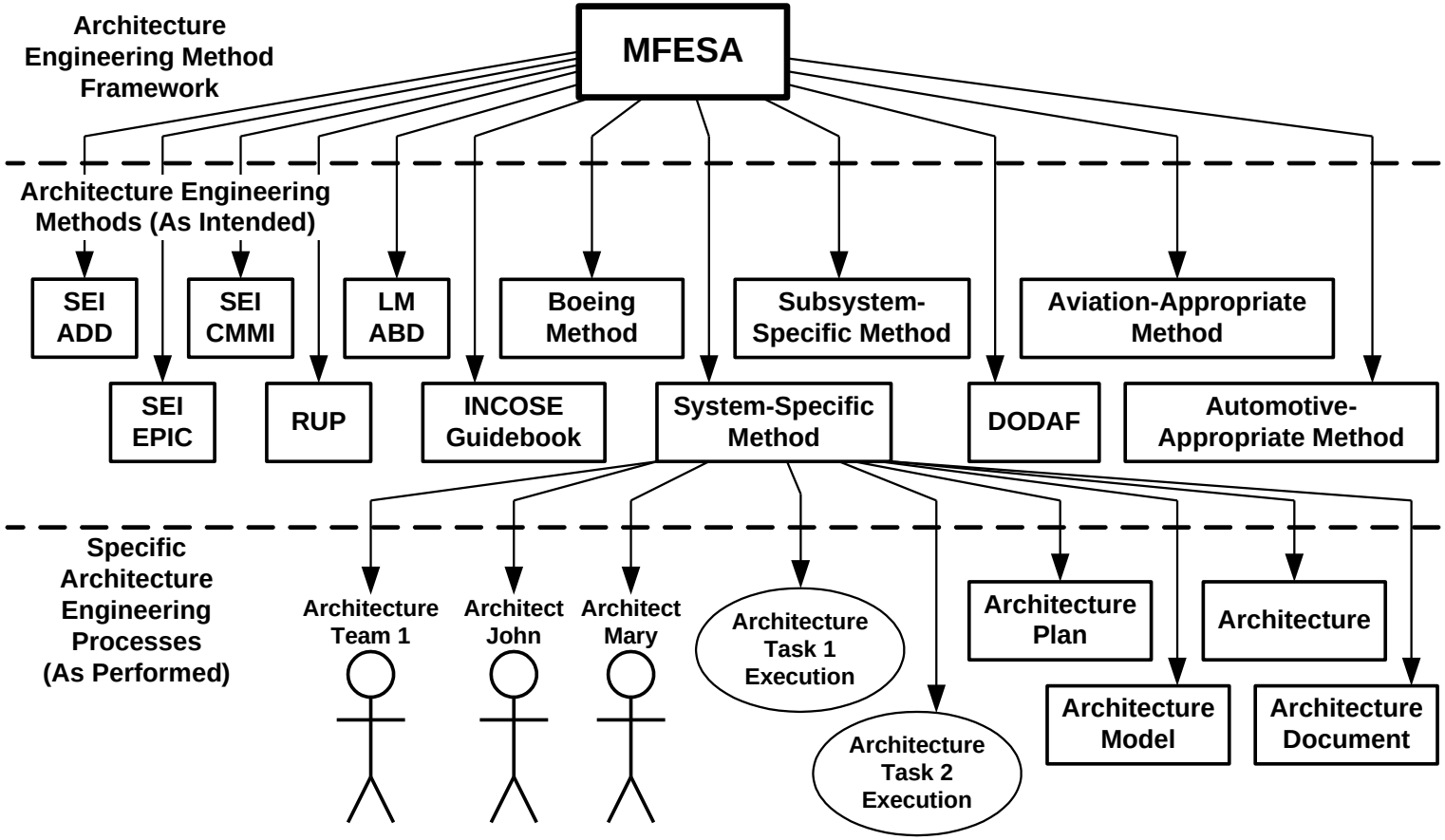
© 2009 Carnegie Mellon University

As-Intended Methods



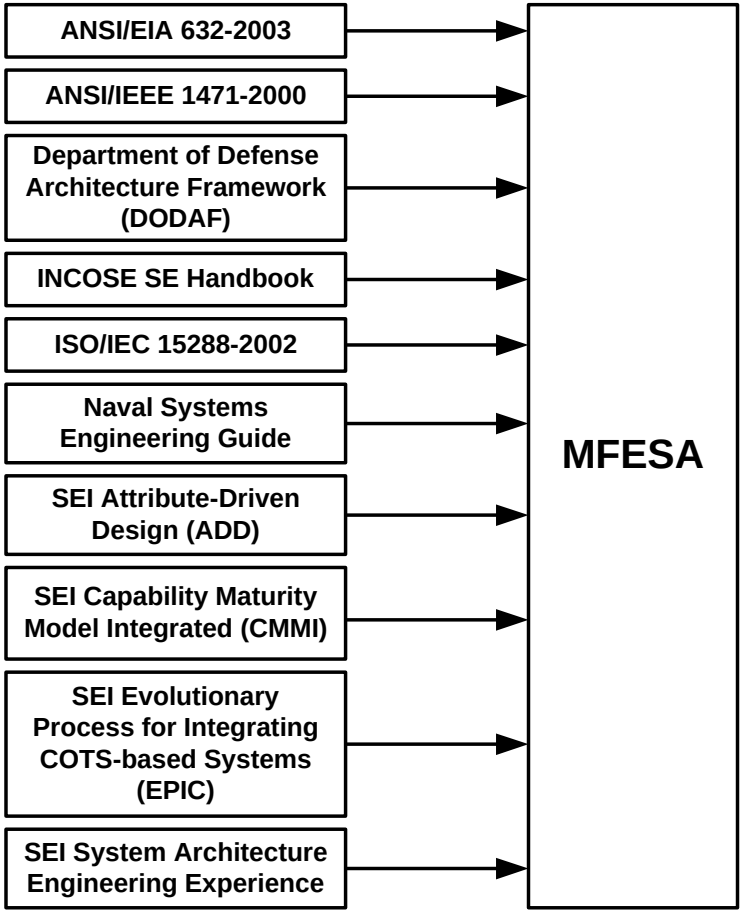
© 2009 Carnegie Mellon University

Method Frameworks



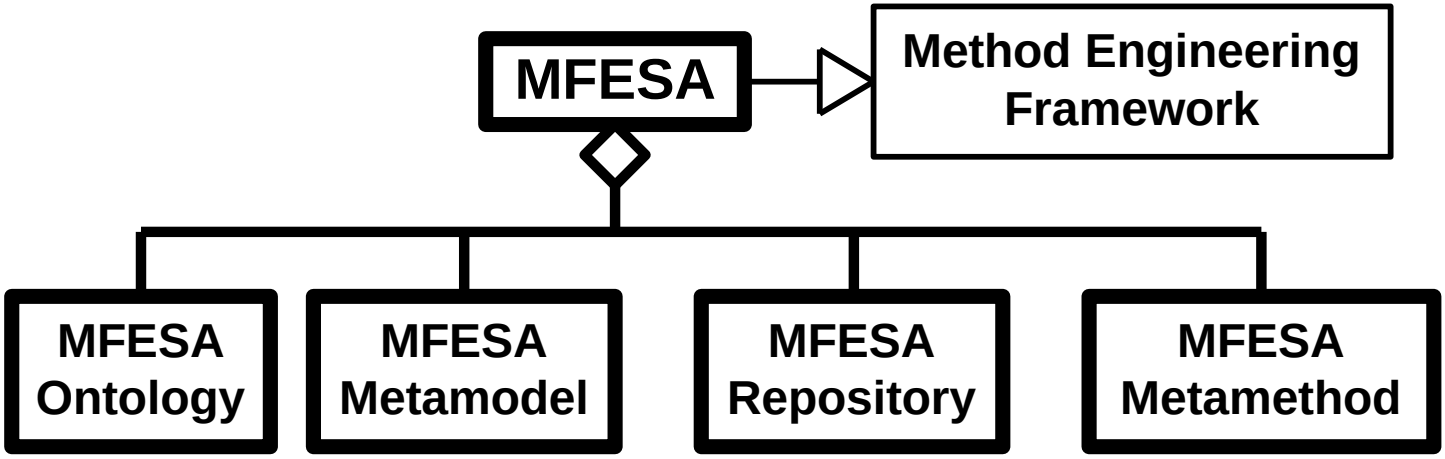
© 2009 Carnegie Mellon University

Primary Inputs to MFESA



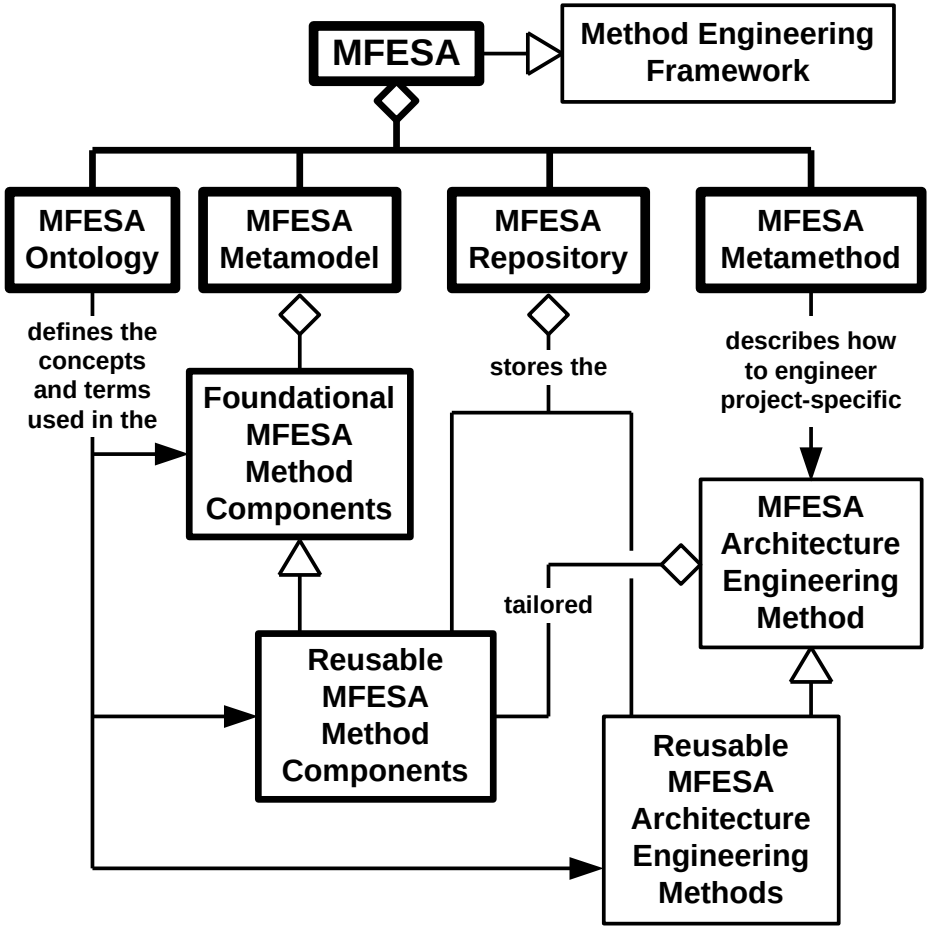
© 2009 Carnegie Mellon University

MFESA Components (Top View)



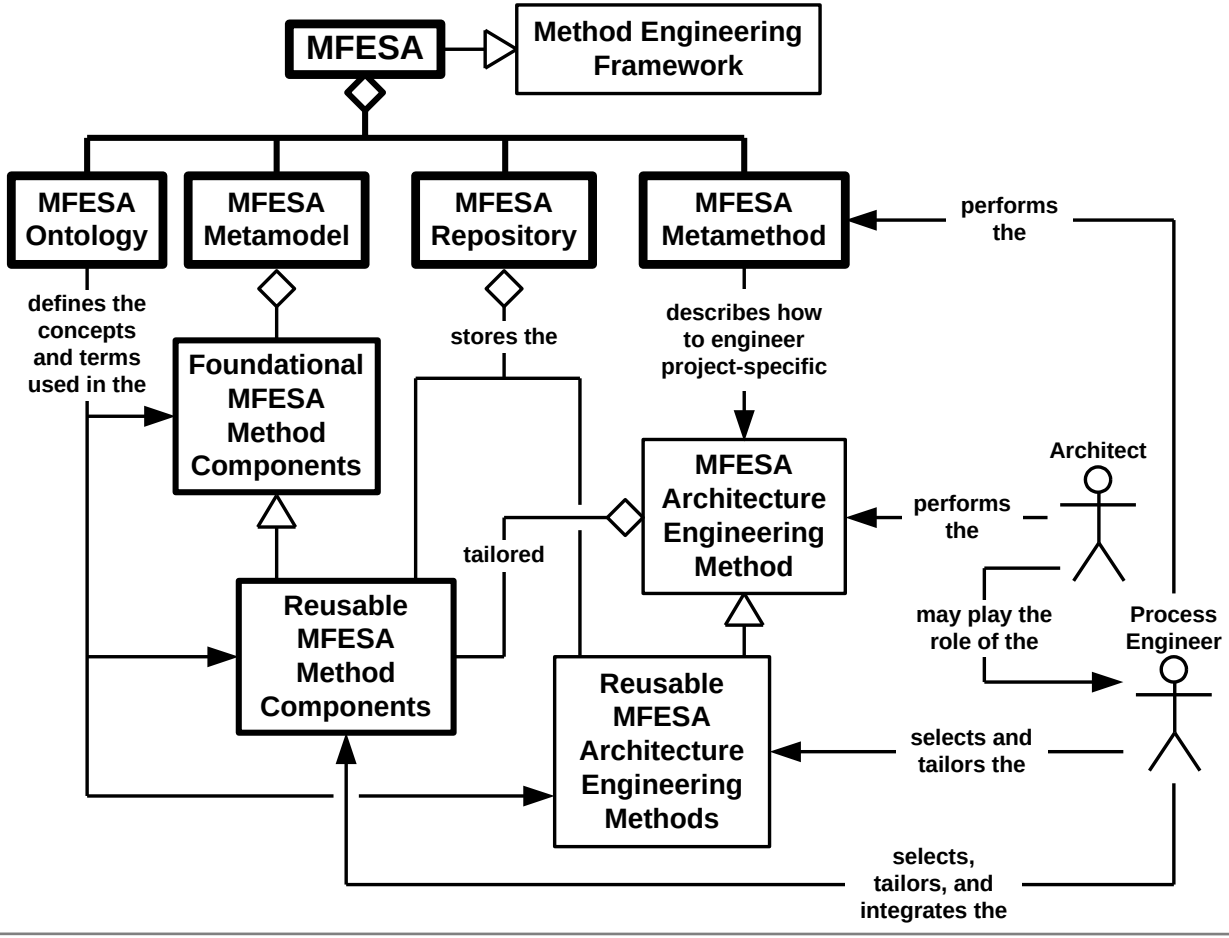
© 2009 Carnegie Mellon University

MFESA Components (Detailed View)



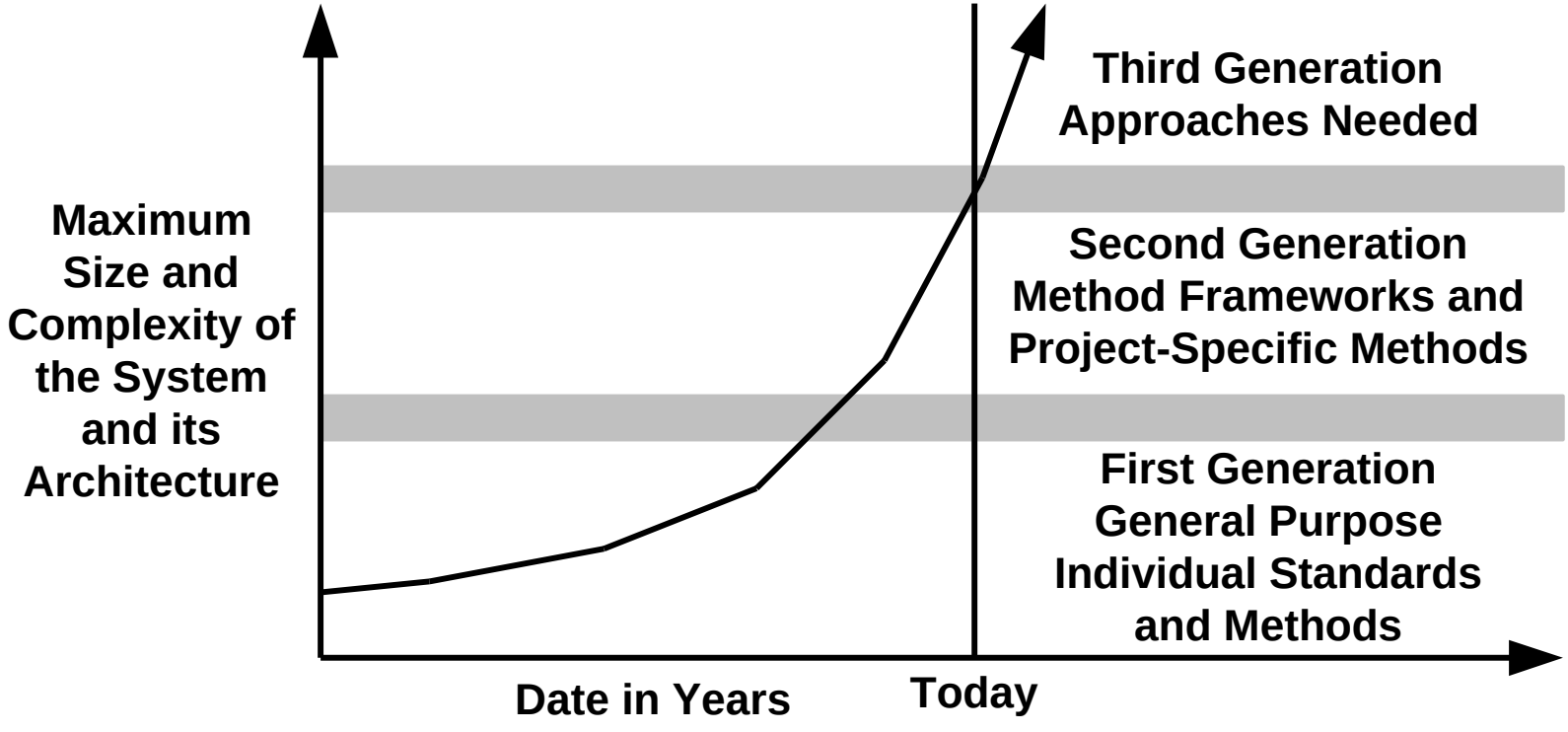
© 2009 Carnegie Mellon University

MFESA Components (Usage)



© 2009 Carnegie Mellon University

MFESA Addresses Size and Complexity



© 2009 Carnegie Mellon University

Topics

Motivation

MFESA Overview

MFESA Ontology of Concepts and Terminology

MFESA Metamodel of Reusable Method Components

MFESA Repository of Reusable Method Components

- **Architectural Work Units and Work Products**
- **Architectural Workers**

MFESA Metamethod

Conclusion

© 2009 Carnegie Mellon University

MFESA Ontology

More than merely an architectural glossary

Information model of system architecture engineering

Defines foundational architectural concepts and terminology

Defines relationships between concepts

© 2009 Carnegie Mellon University

MFESA Ontology of Concepts and Terminology

System

System Architecture

Architectural Structures

Architectural Styles, Patterns, and Mechanisms

Architectural Drivers and Concerns

Quality Model, Quality Requirements,

Architectural Representations

Architectural Models, Structures, Views, and Focus Areas

Architectural Quality Cases

Architectural Visions

© 2009 Carnegie Mellon University

System - Definition

System

a cohesive integrated set of system components (i.e., an aggregation structure) that collaborate to provide the behavior and characteristics needed to meet valid stakeholder needs and desires

Important Ideas:

- Modeled as hierarchical aggregate structure
- Integrated system components
- Components collaborate
- Emergent behavior and properties

© 2009 Carnegie Mellon University

System Component Types

Subsystems

Consumable materials (e.g., ammunition, fuel, lubricants, reagents, and solvents)

Data

Documentation (both separate physical and built-in electronic documentation)

Equipment (e.g., maintenance, support, and training equipment)

Facilities (e.g., maintenance, manufacturing, operations, support, training, and disposal facilities including their component property, buildings, and their furnishings)

Hardware

Manual procedures

Networks (for the flow of data, power, and material)

Organizations

Personnel

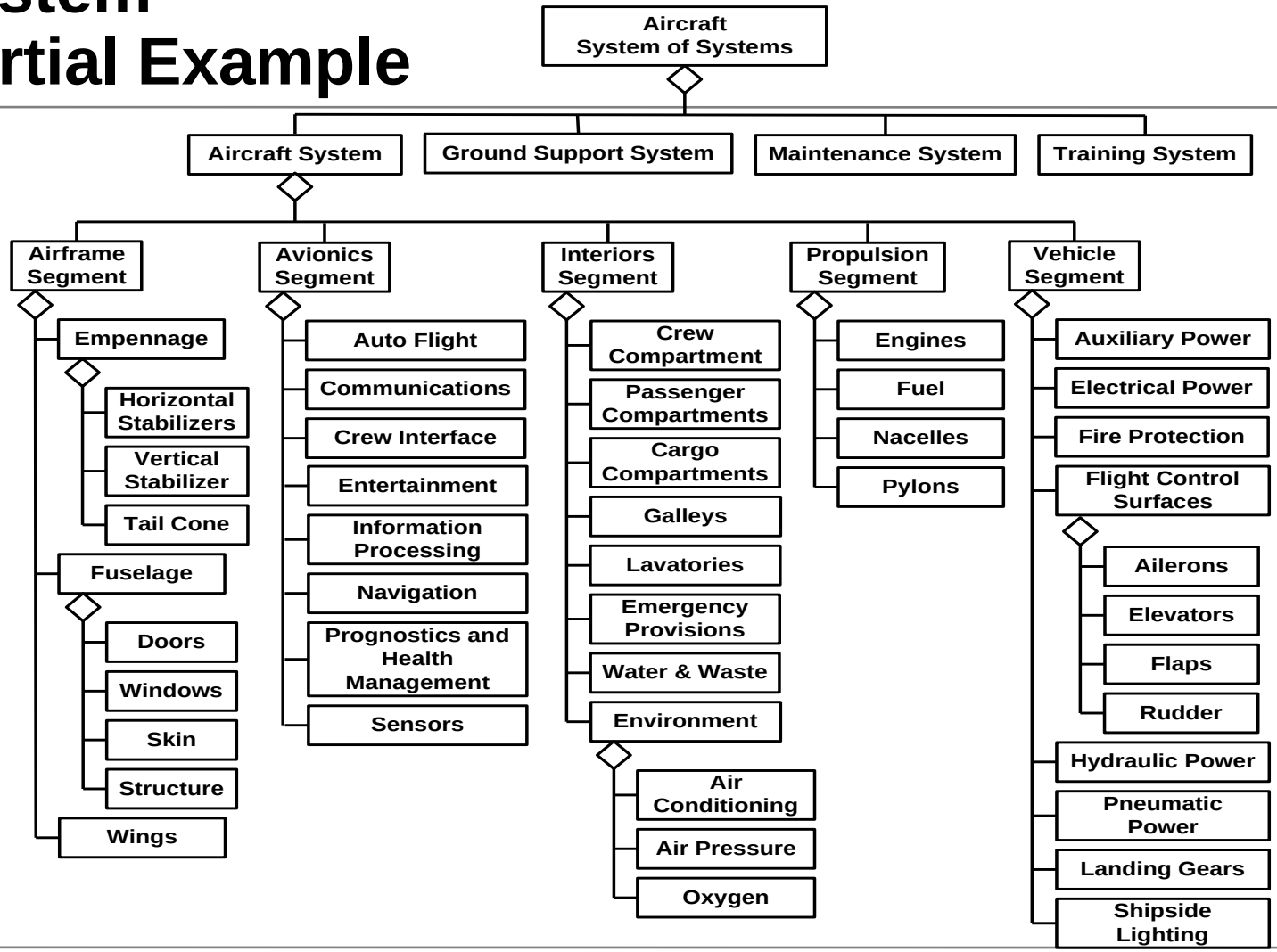
Physical interfaces

Software

Tools

© 2009 Carnegie Mellon University

System – Partial Example



© 2009 Carnegie Mellon University

Some System Characteristics

Multiple Components

Multiple Interactions between Components

Multiple Structures (Logical and Physical, Static and Dynamic)

Multiple:

- Views and Viewpoints
- Models
- Focus Areas

© 2009 Carnegie Mellon University

What about Systems of Systems?

System of Systems (SOS)

> a system composed of systems

Almost all systems are composed of systems (i.e., subsystems)

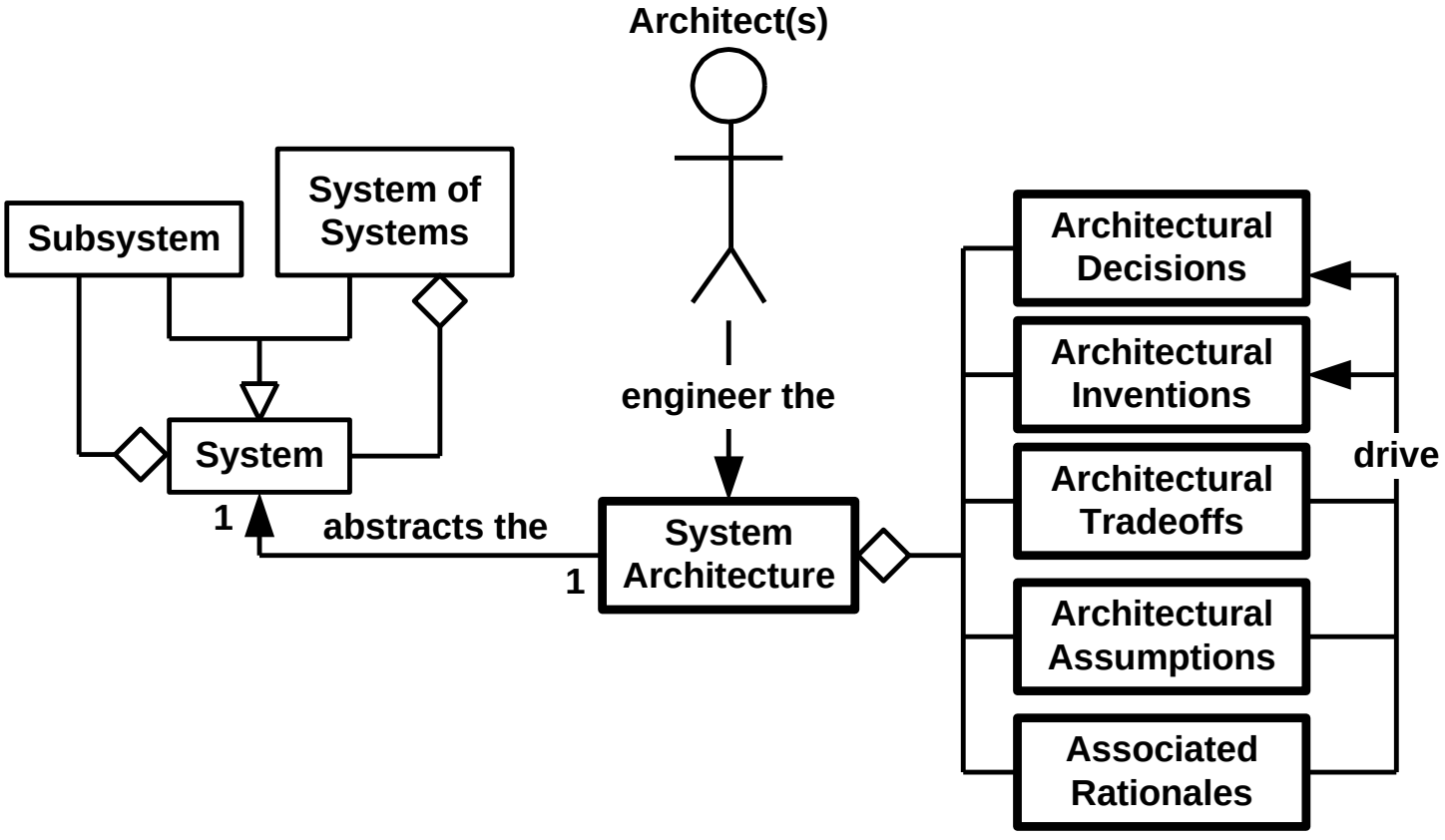
When most people say *systems of systems*, what they really mean is something like this:

> an ultra-large and complex, highly flexible, dynamically evolving, technologically ambitious, and geographically-distributed **system of** pre-existing, heterogeneous, autonomous, self-contained, and independently governed (e.g., acquired, developed, operated, scheduled, and funded} **systems**, whereby the system of systems exhibits significant amounts of unexpected emergent behavior and characteristics

Engineering the architecture of such systems of systems calls for a different architecture engineering method than simpler systems.

© 2009 Carnegie Mellon University

System and System Architecture - Ontology



© 2009 Carnegie Mellon University

Architectural Structure, Element, and Component – Definitions

Architectural Structure

a cohesive set of architectural elements connected by associated relationships that captures a set of related architectural decisions, inventions, tradeoffs, assumptions, and rationales

Architectural Element

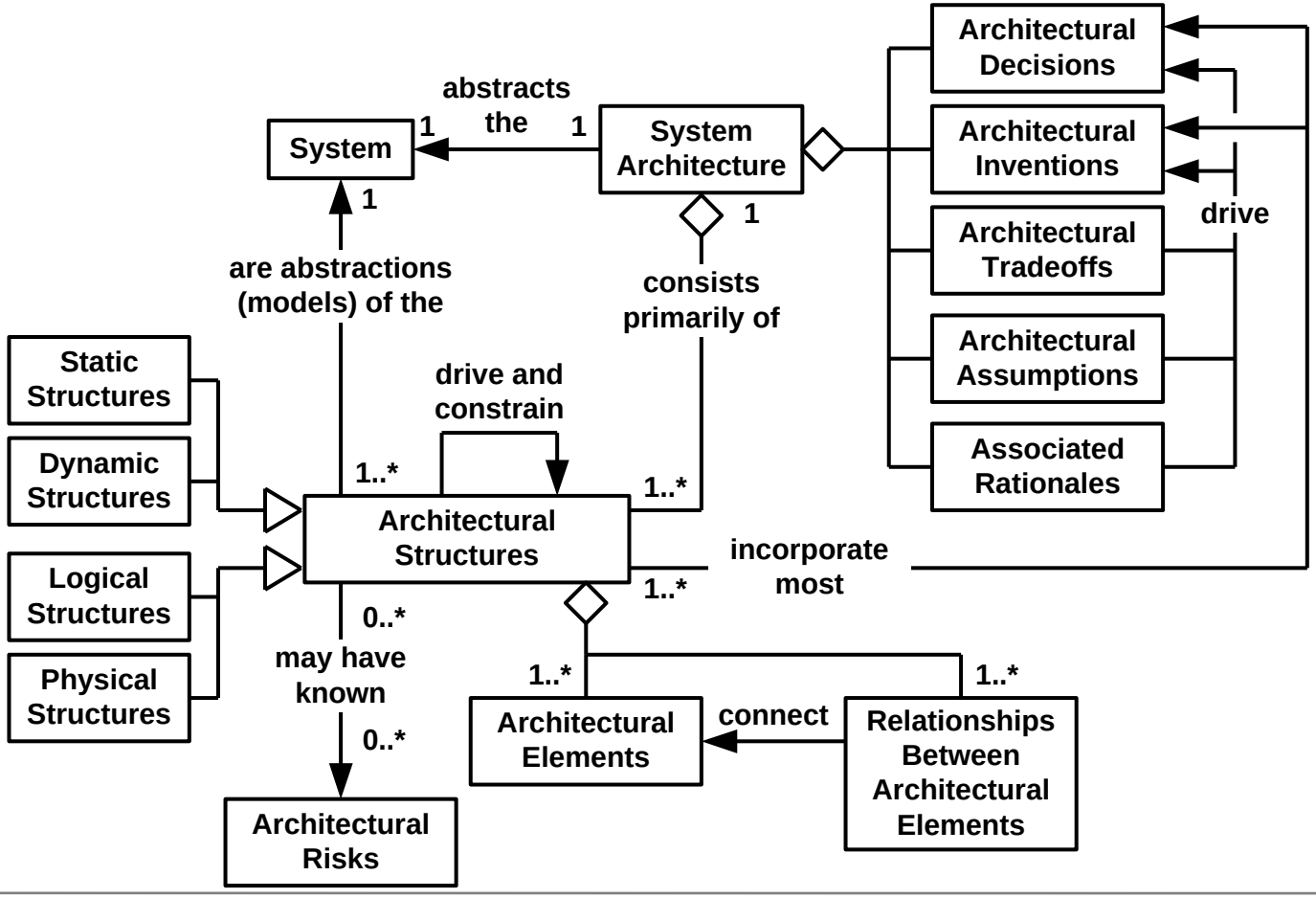
a part of an architectural structure

Architectural Component

a physical architectural element of a static physical aggregation structure

© 2009 Carnegie Mellon University

Architectural Structure - Ontology



© 2009 Carnegie Mellon University

Architectural Styles, Patterns, and Mechanisms - Definitions

Architectural Pattern

a well-documented reusable solution to a commonly occurring architectural problem within the context of a given set of existing architectural concerns, decisions, inventions, engineering trade-offs, and assumptions

Architectural Style

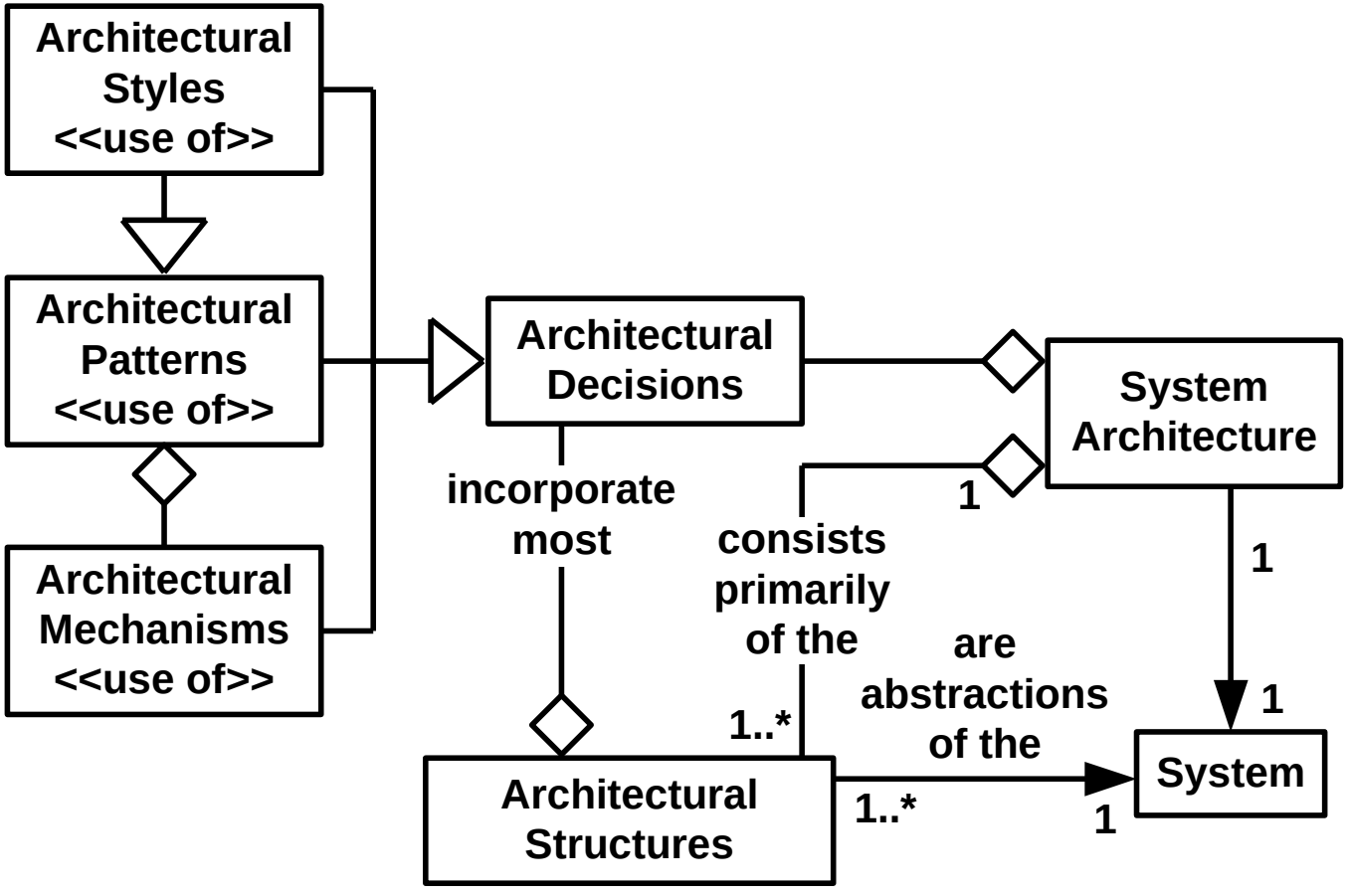
a top-level architectural pattern that provides an overall context in which lower-level architectural patterns exist

Architectural Mechanism

a major architectural decision or invention, often an element of an architectural pattern

© 2009 Carnegie Mellon University

Architectural Styles, Patterns, and Mechanisms - Ontology



© 2009 Carnegie Mellon University

Architectural Drivers and Concerns - Definitions

Architectural Driver

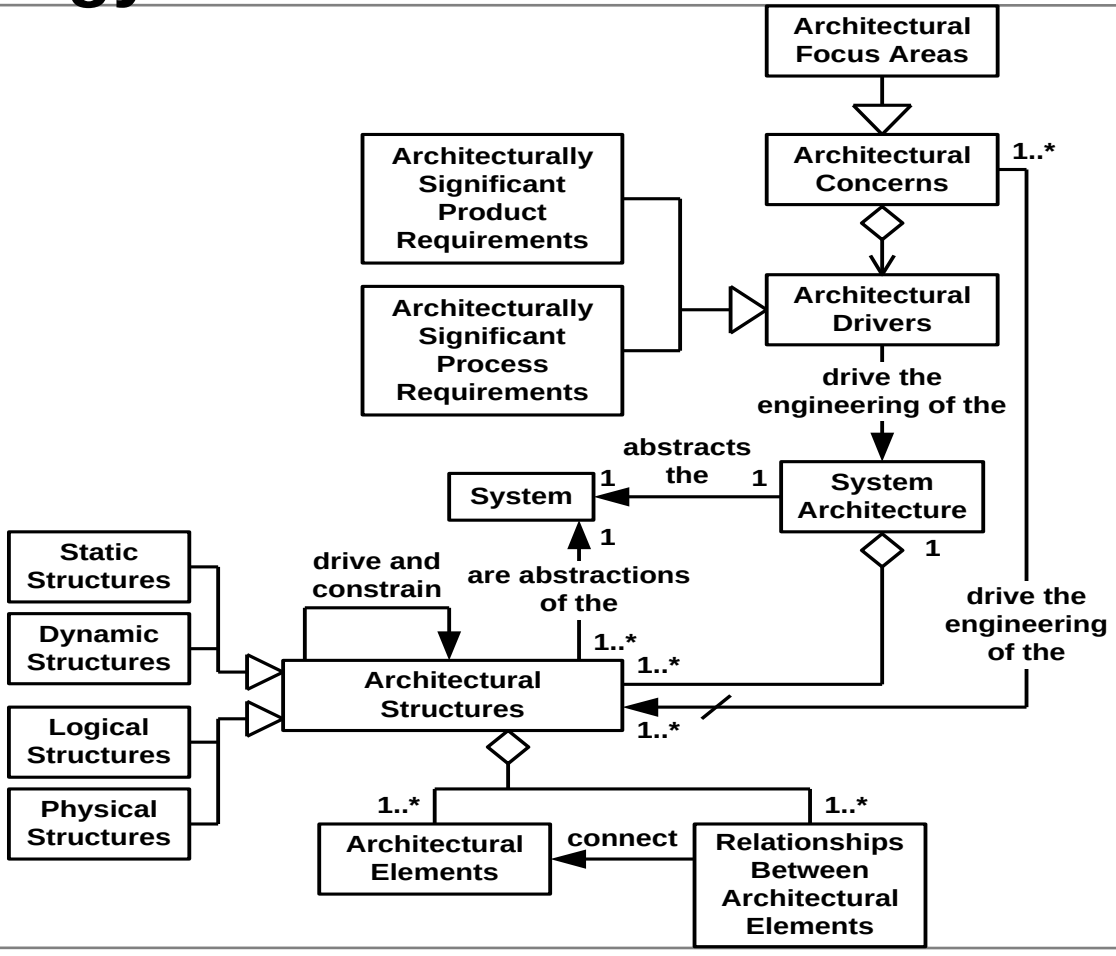
an architecturally significant product or process requirement that drives the engineering of the system architecture

Architectural Concern

a *cohesive* collection of architectural drivers

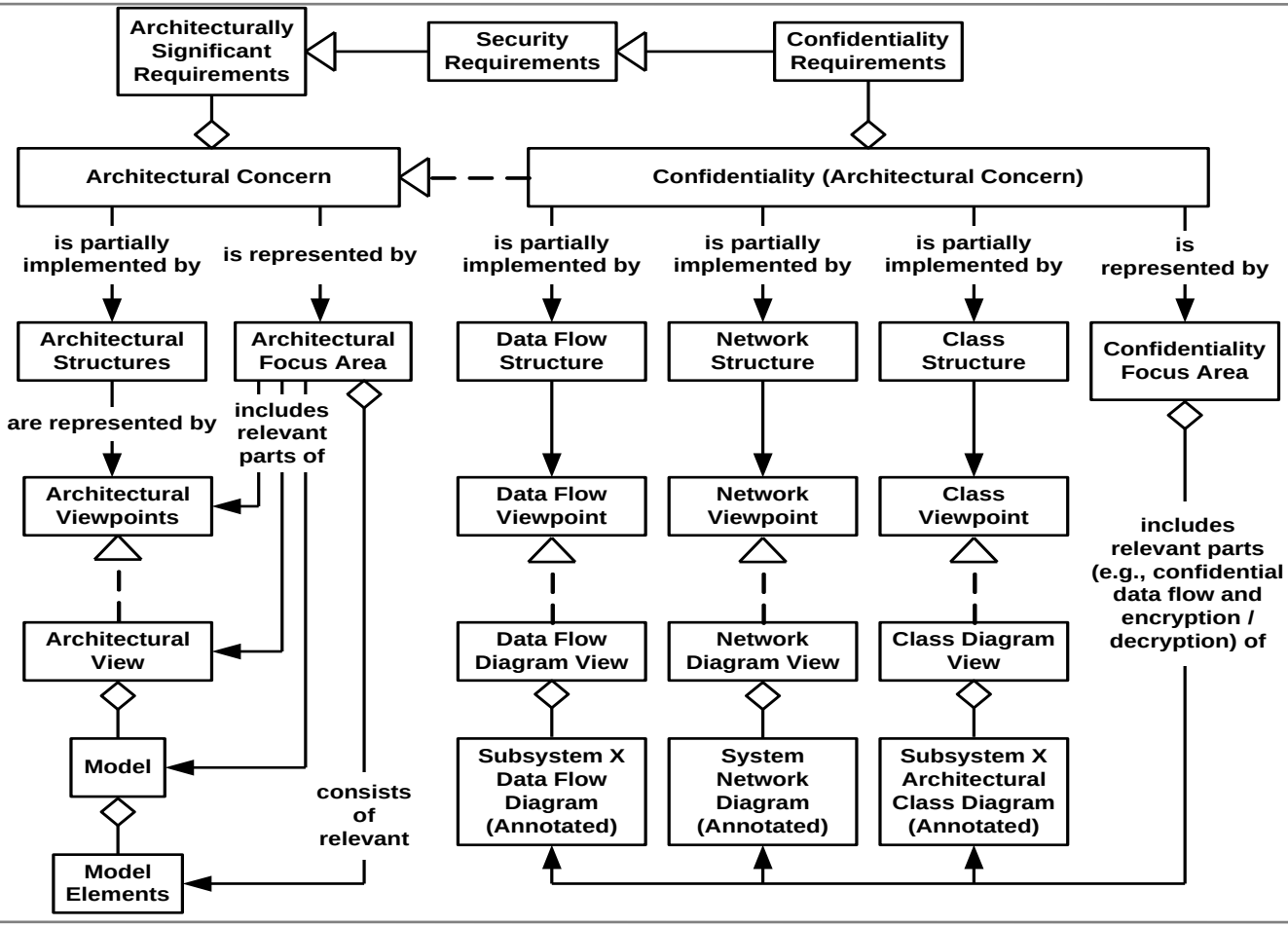
© 2009 Carnegie Mellon University

Architectural Drivers and Concerns - Ontology



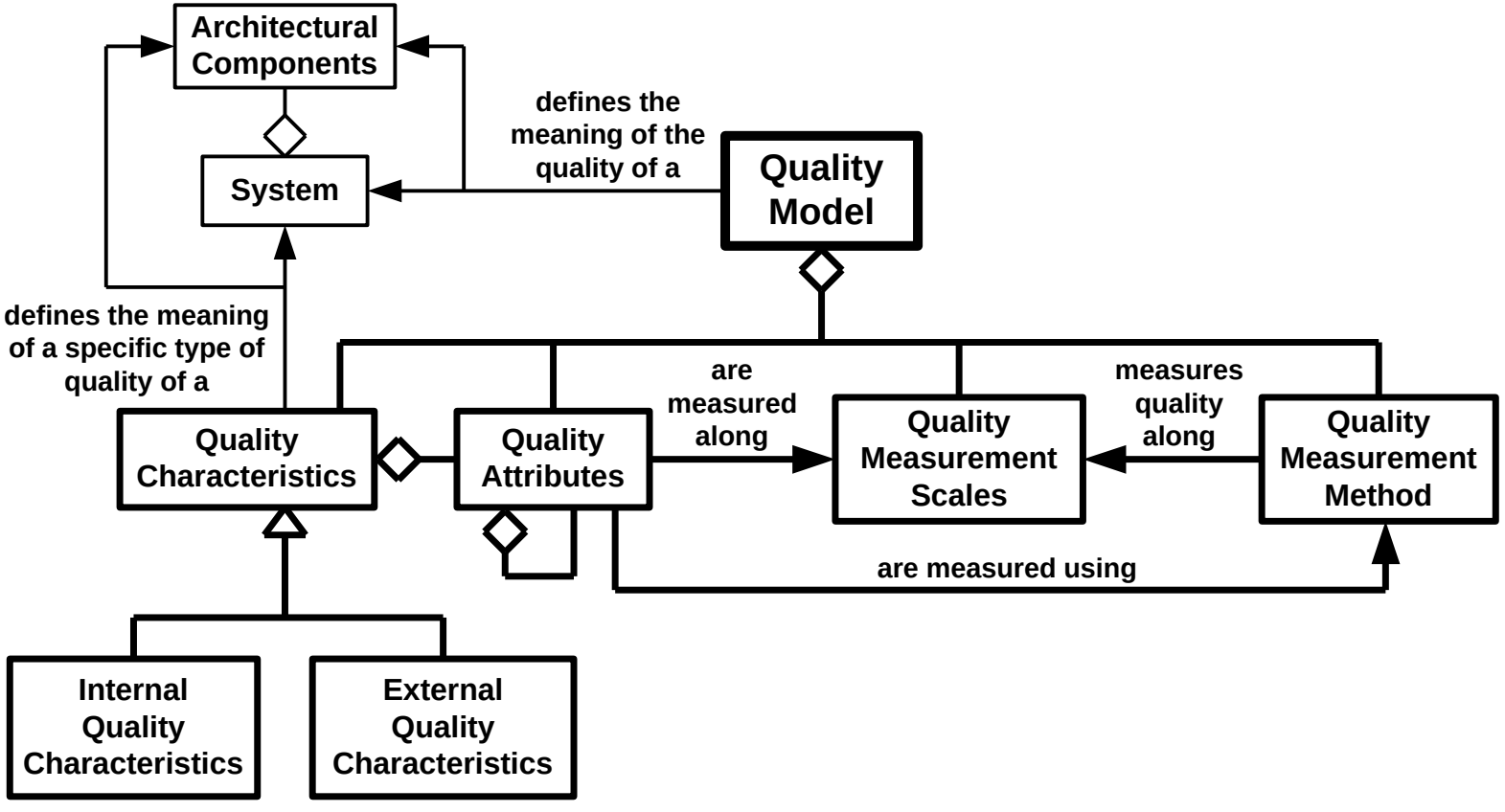
© 2009 Carnegie Mellon University

Architectural Concern – An Example



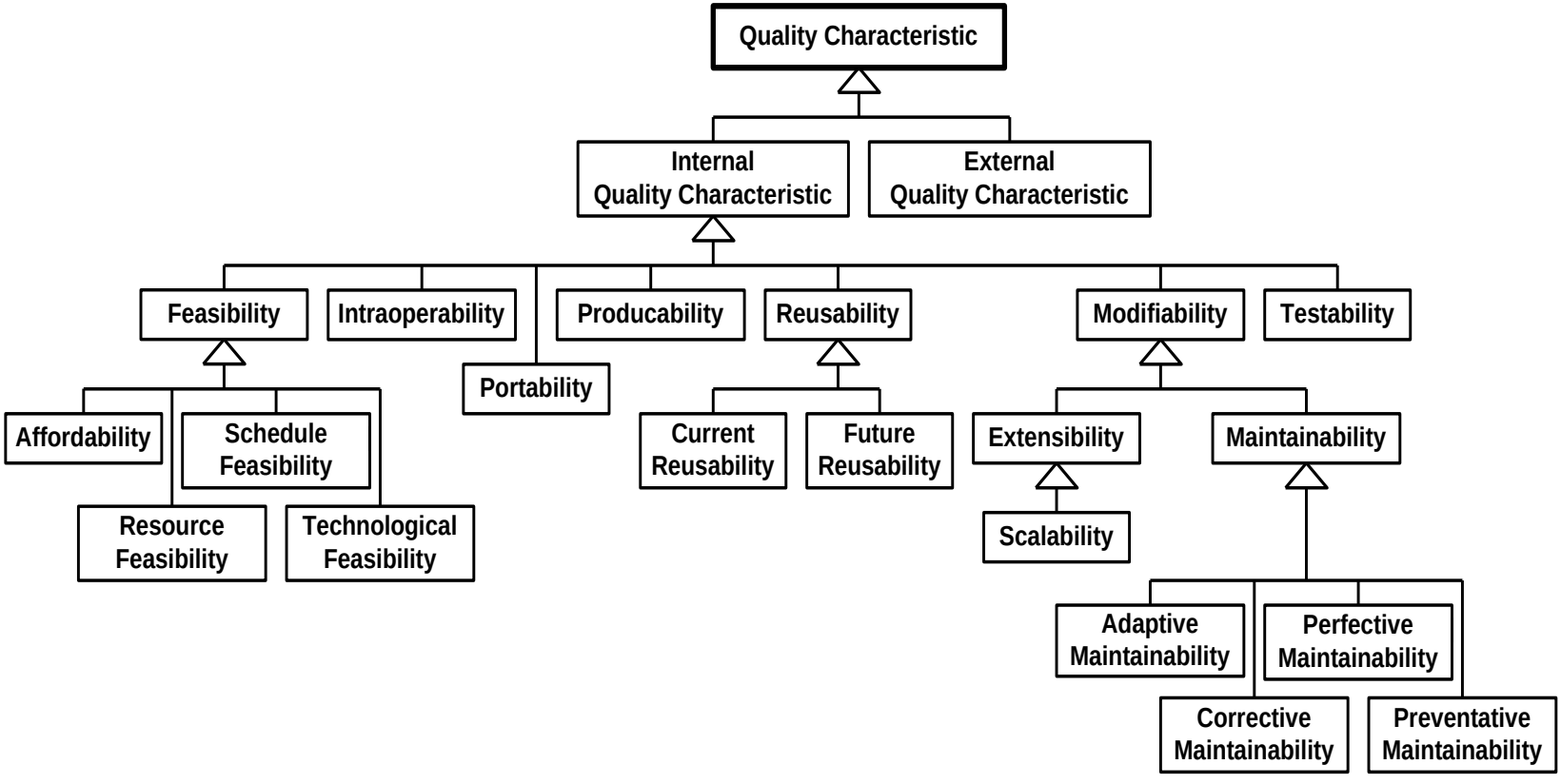
© 2009 Carnegie Mellon University

MFESA Quality Model



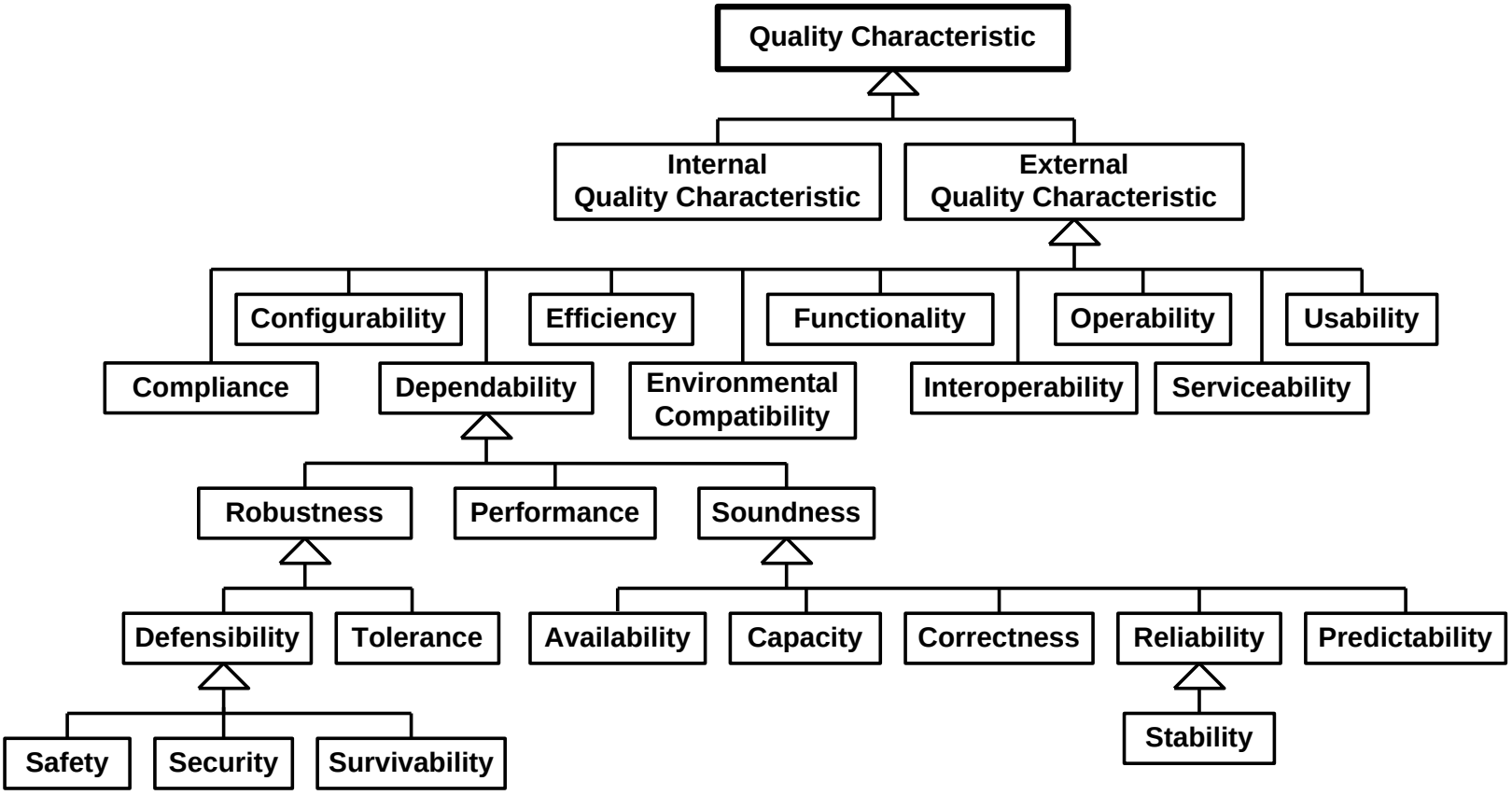
© 2009 Carnegie Mellon University

Internal Quality Characteristics



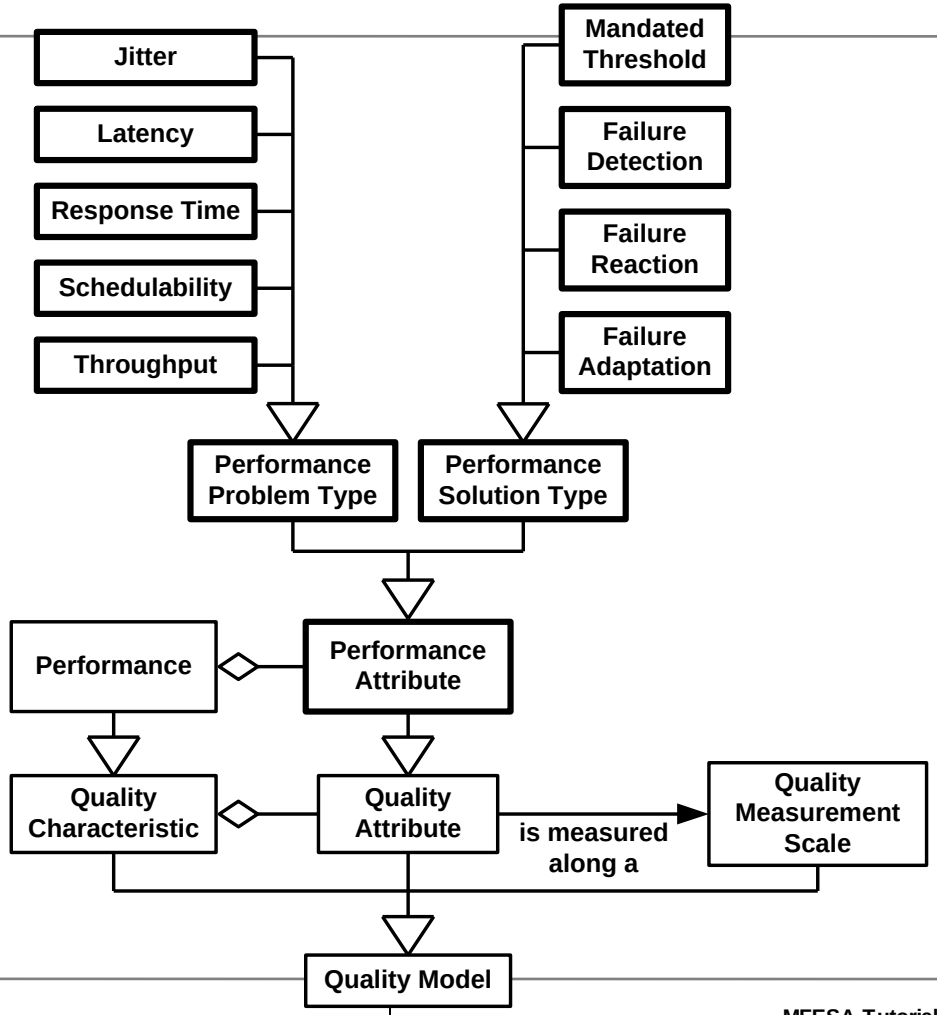
© 2009 Carnegie Mellon University

External Quality Characteristics



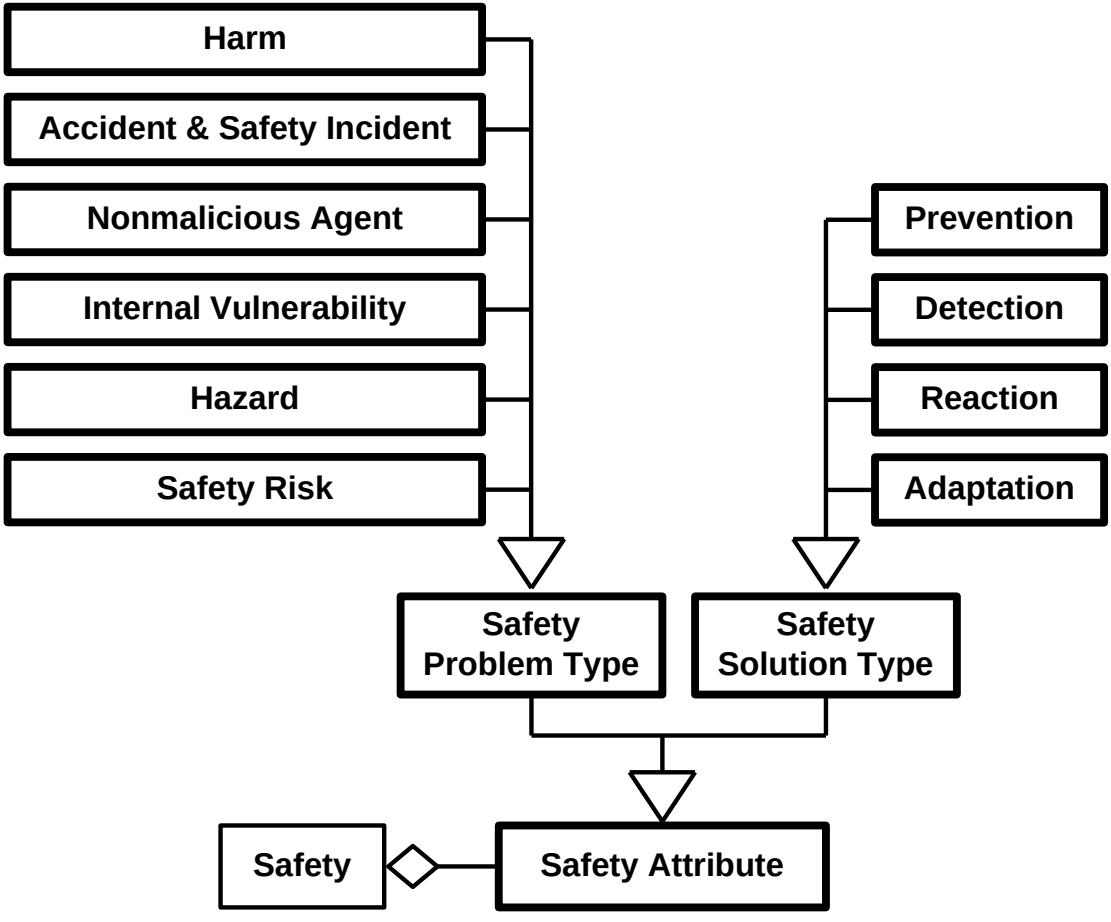
© 2009 Carnegie Mellon University

Example Characteristic and Attributes



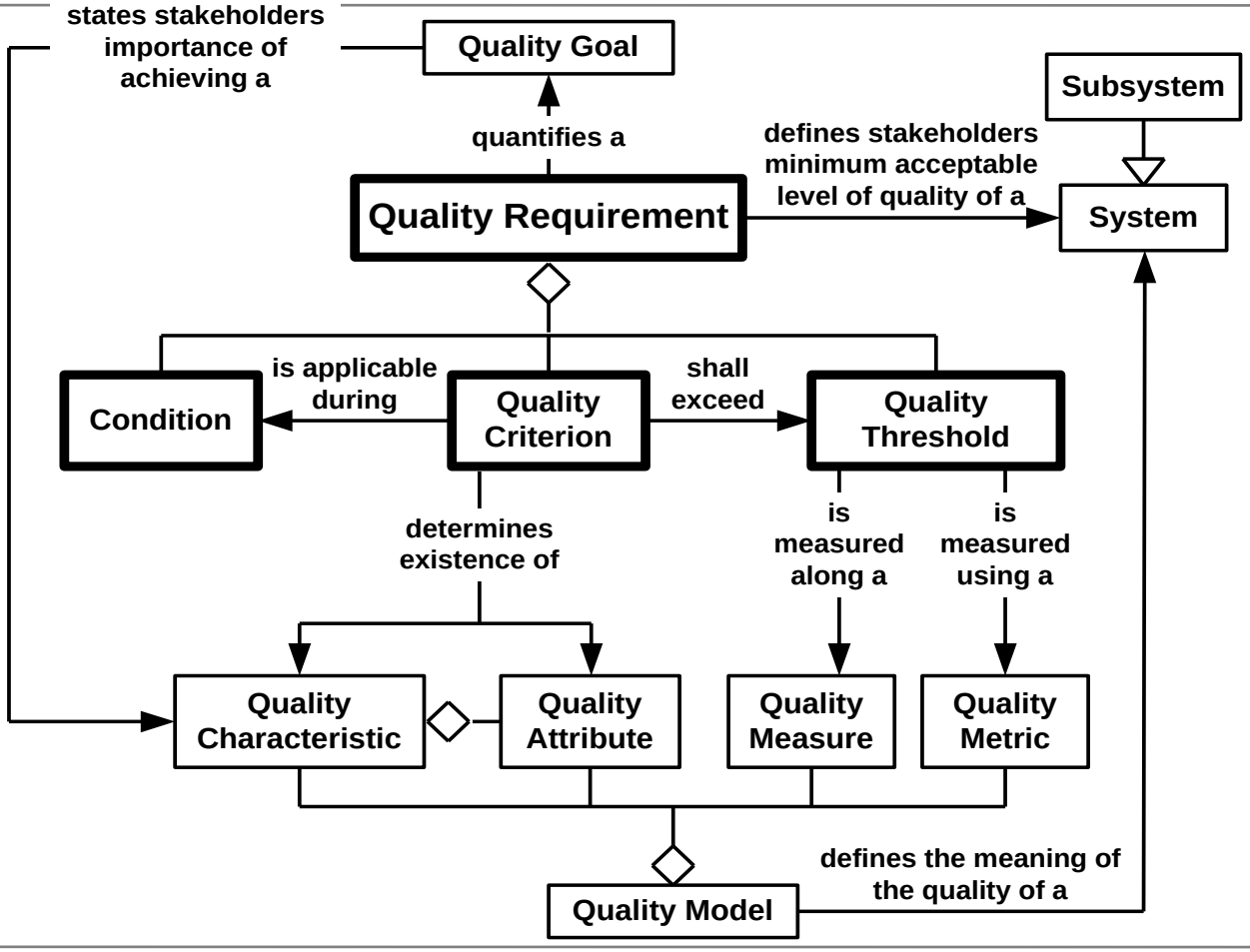
© 2009 Carnegie Mellon University

Example Characteristic and Attributes



© 2009 Carnegie Mellon University

Quality Requirements



© 2009 Carnegie Mellon University

Architectural Representations - Definition

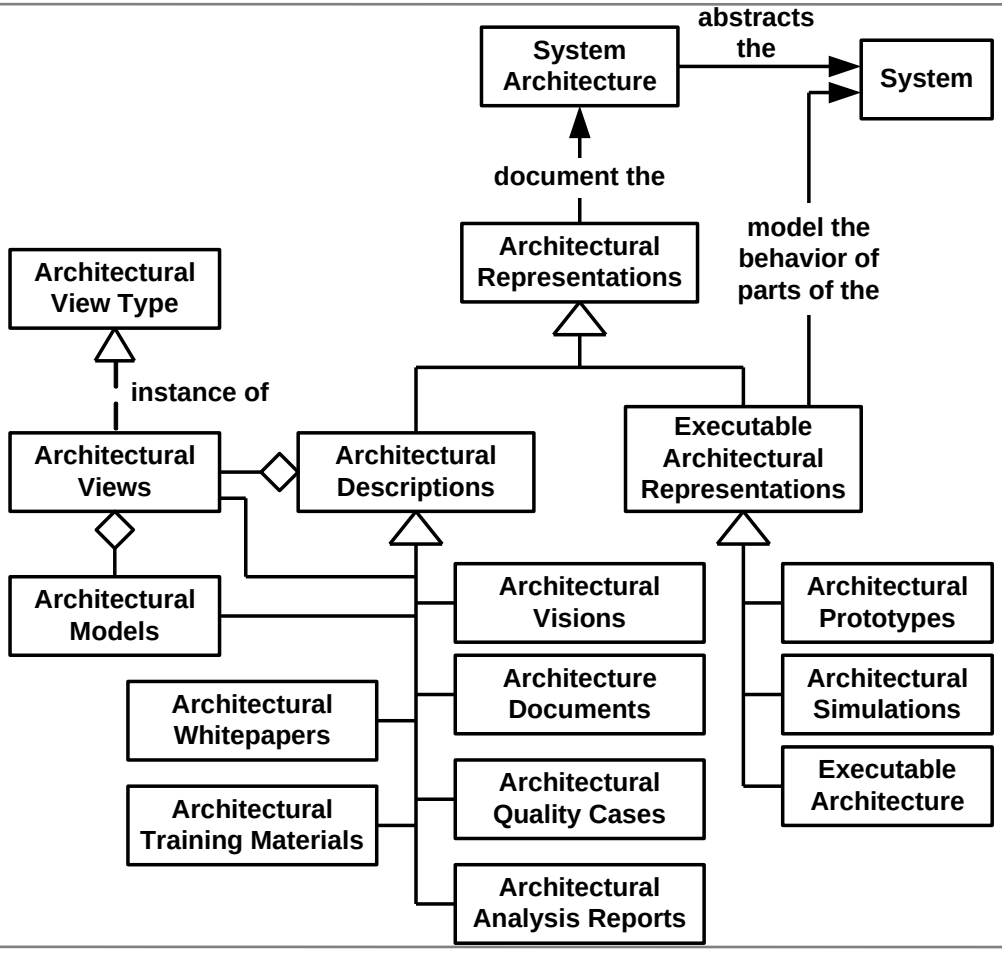
Architectural Representation

> a cohesive collection of information that documents a system architecture

Not the same thing as the architecture

© 2009 Carnegie Mellon University

Architectural Representations - Ontology



© 2009 Carnegie Mellon University

Architectural Models, Views, and Focus Areas - Definitions

Architectural Model

an architectural representation that abstracts a single system structure in terms of the structure's architectural elements and the relationships between them

Architectural View

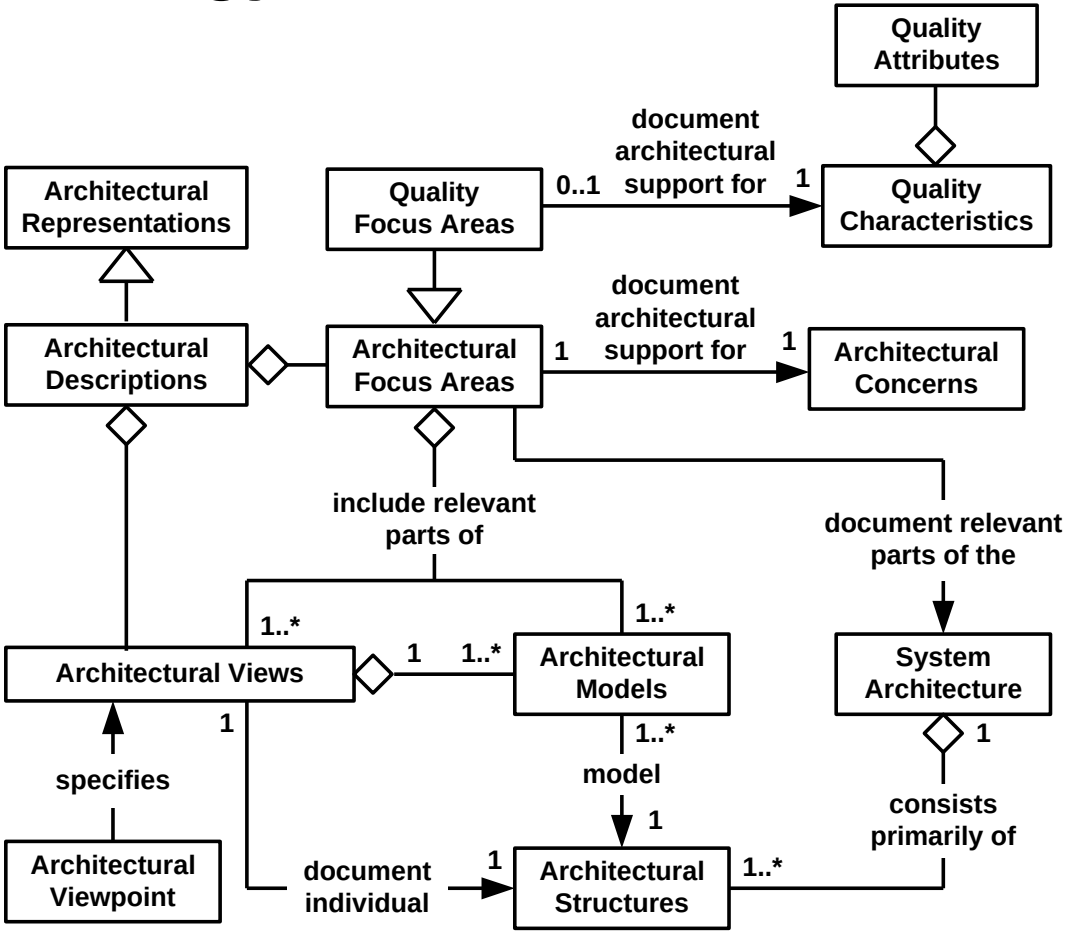
an architectural representation describing a single architectural structure of a system consisting of one or more related models of that structure

Architectural Focus Area

an architectural representation consisting of the cohesive set of all architectural decisions, decisions, and tradeoffs related to a specific architectural concern, regardless of the architectural view, model, or structure where they are documented or found

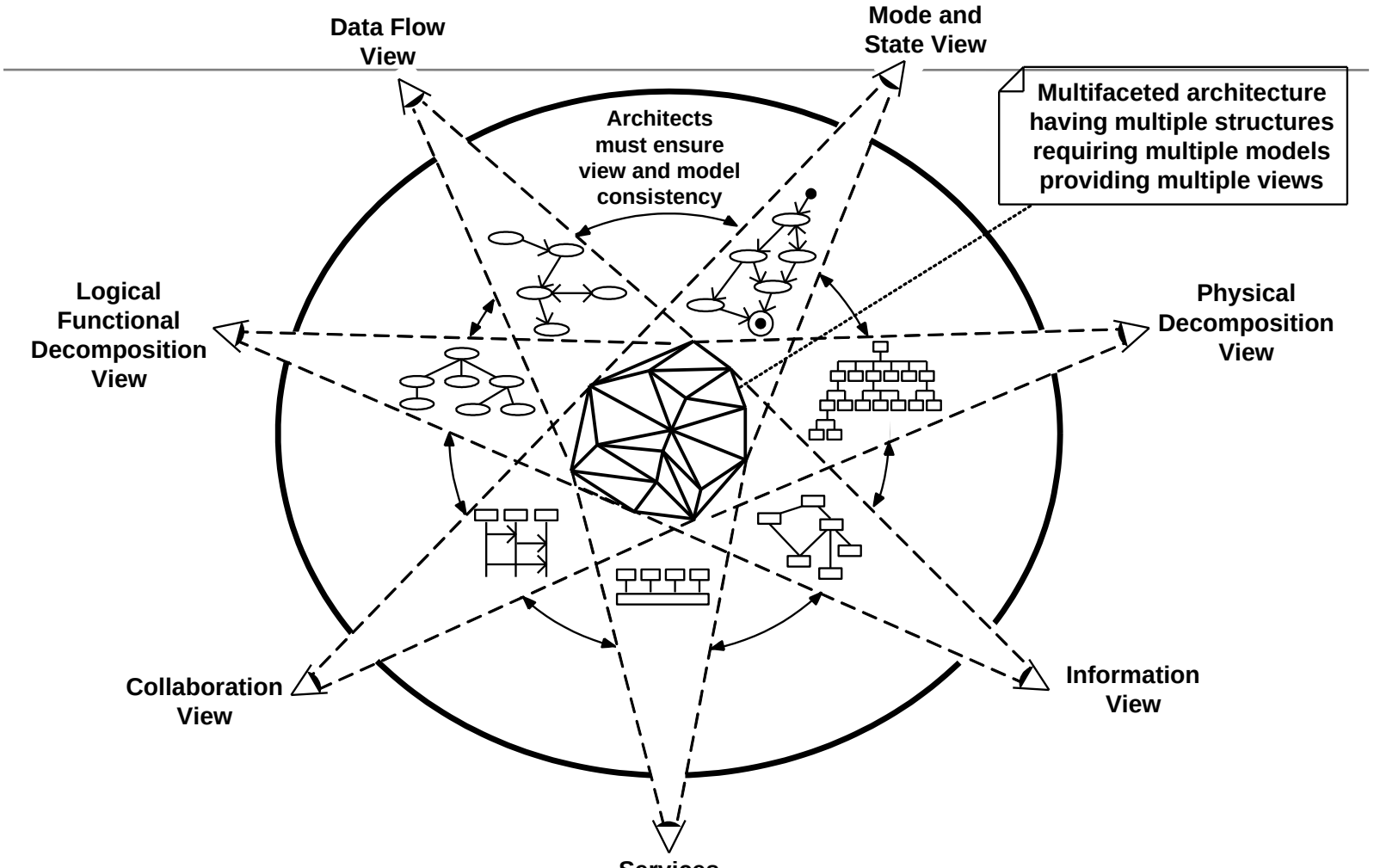
© 2009 Carnegie Mellon University

Architectural Models, Views, and Focus Areas - Ontology



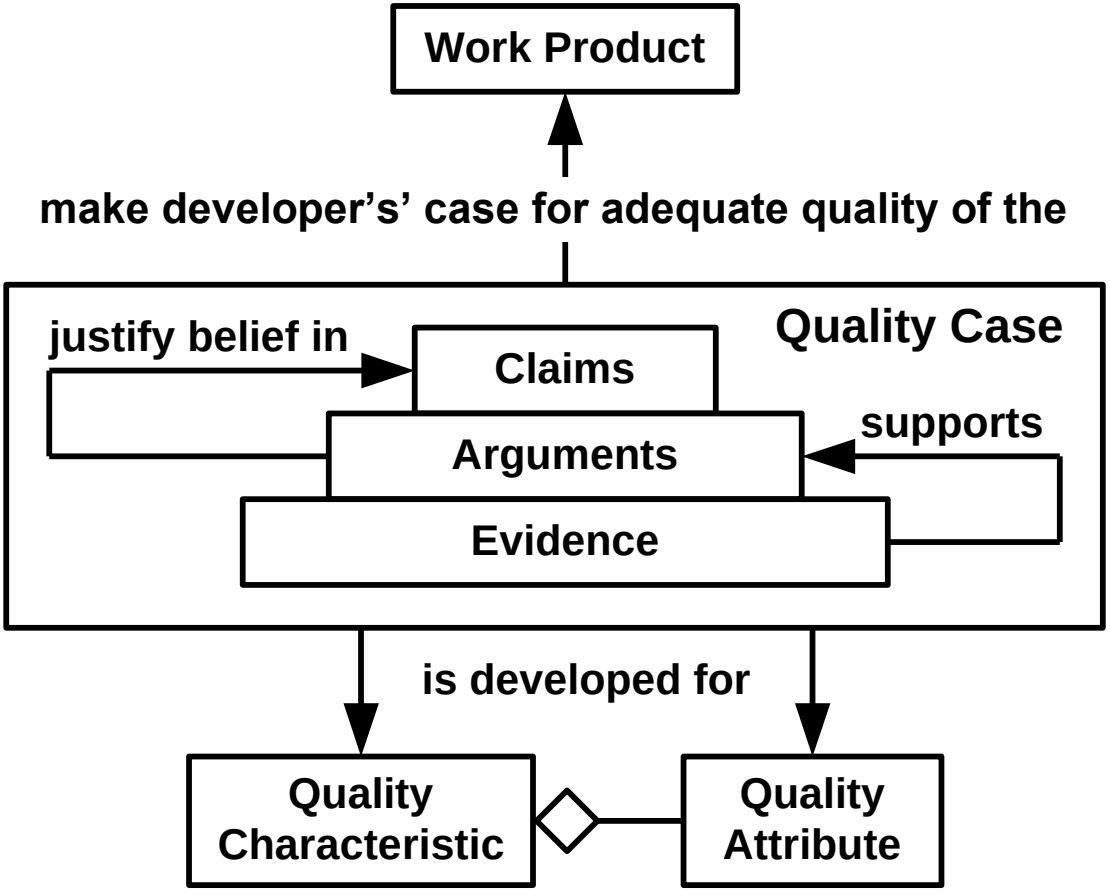
© 2009 Carnegie Mellon University

Architectural Views



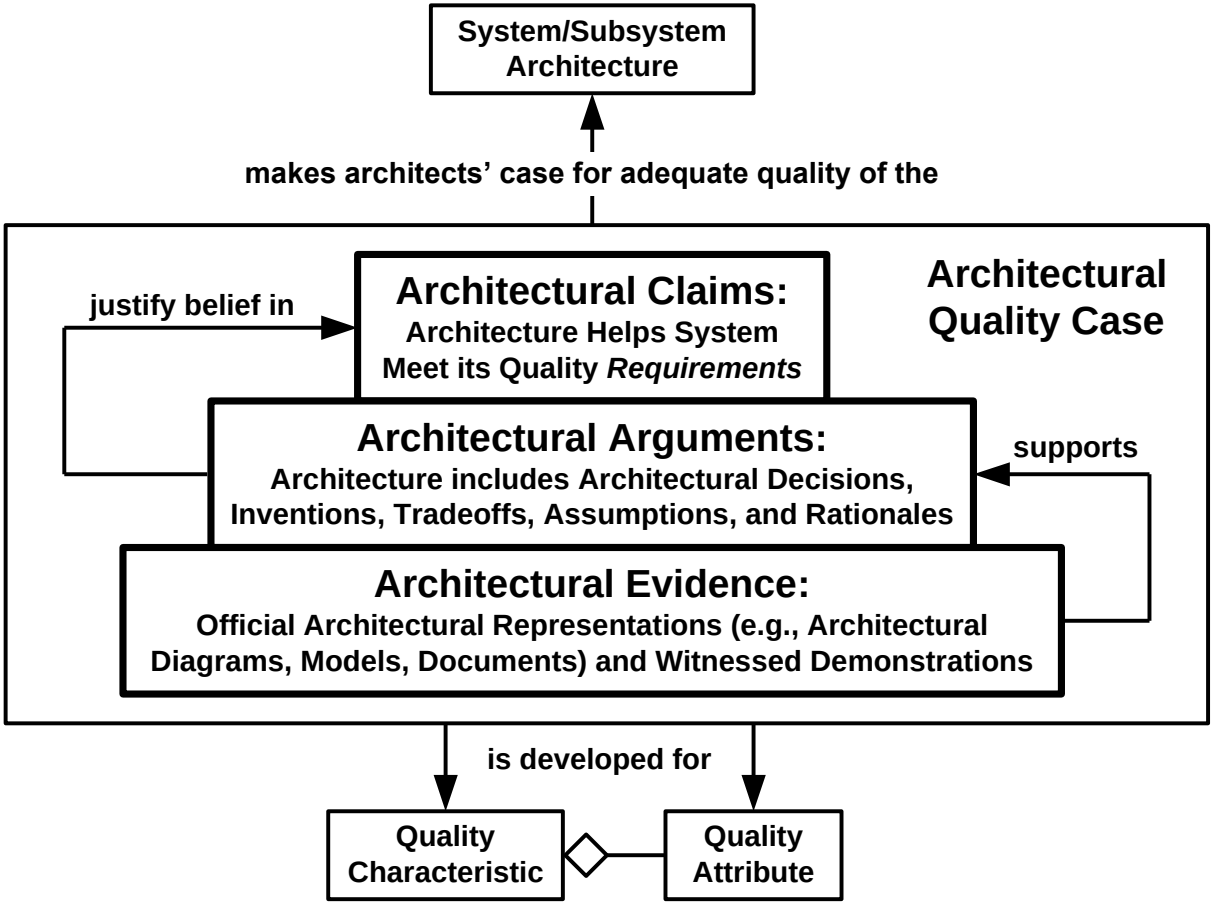
© 2009 Carnegie Mellon University

Quality Cases



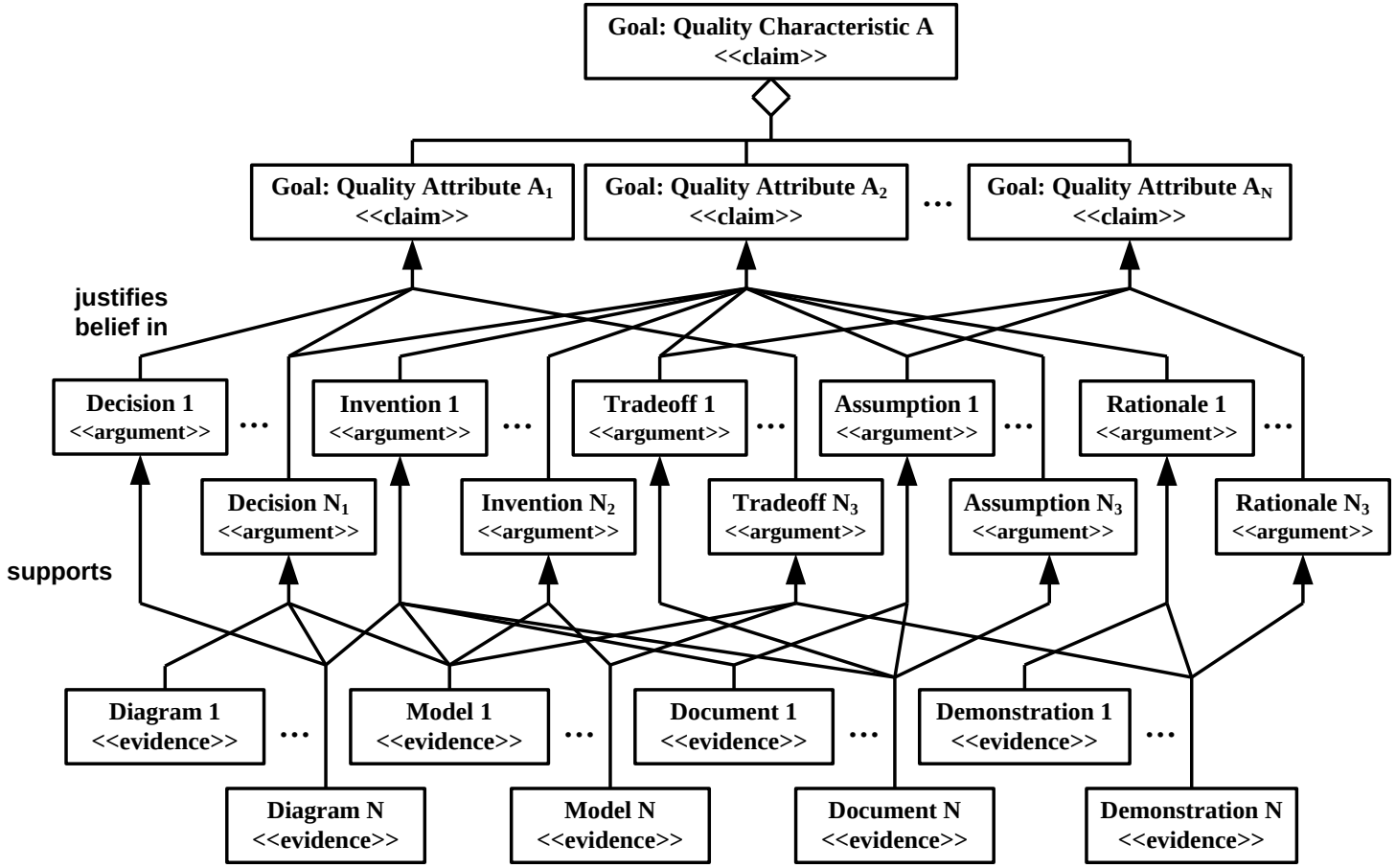
© 2009 Carnegie Mellon University

Architectural Quality Cases



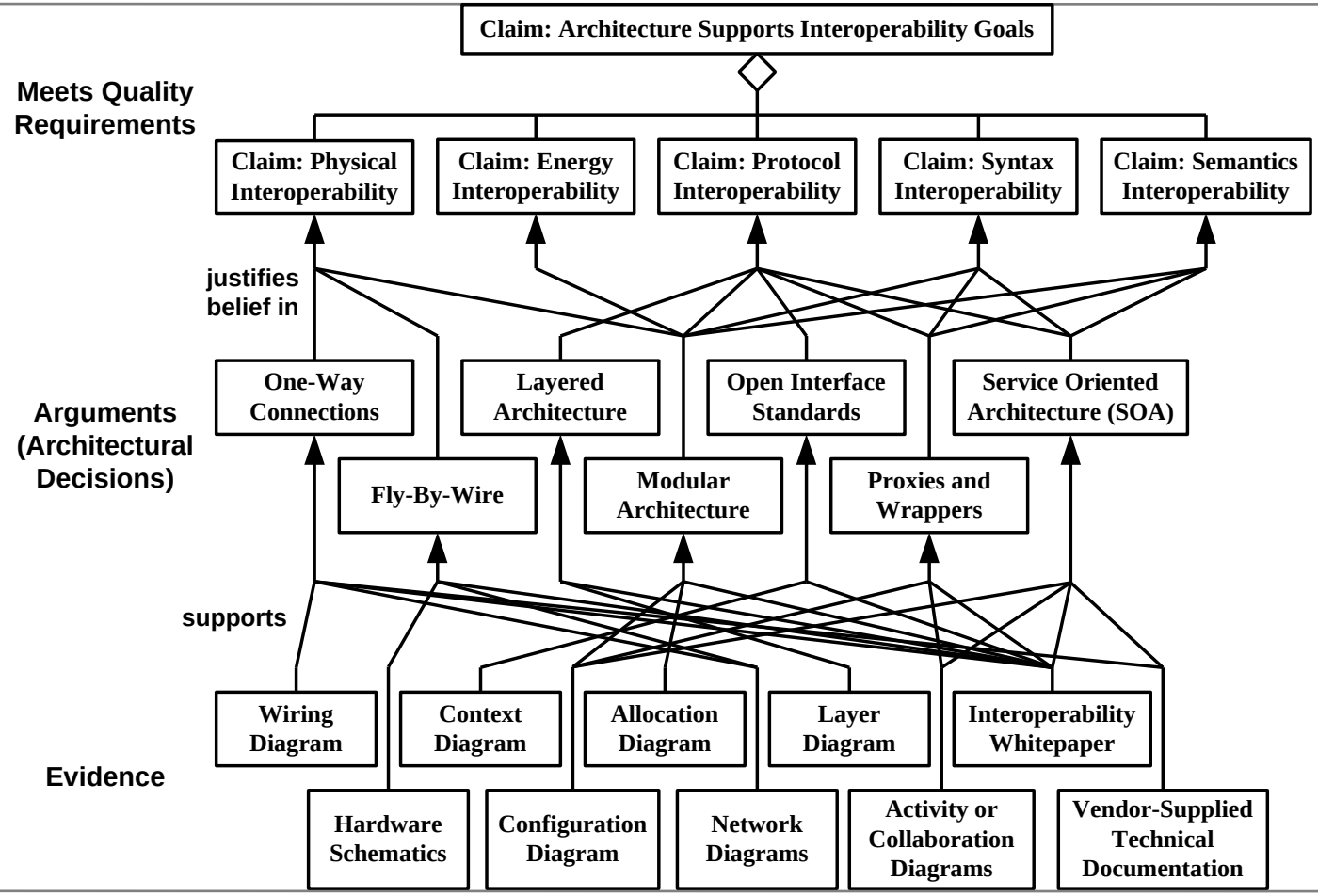
© 2009 Carnegie Mellon University

Architectural Quality Case Diagram



© 2009 Carnegie Mellon University

Example Architectural Quality Case Diagram



© 2009 Carnegie Mellon University

Architecture Visions and Vision Components - Definitions

Architectural Vision

one of the more important actual or potential architectural decisions, inventions, or tradeoffs addressing one or more architectural concerns

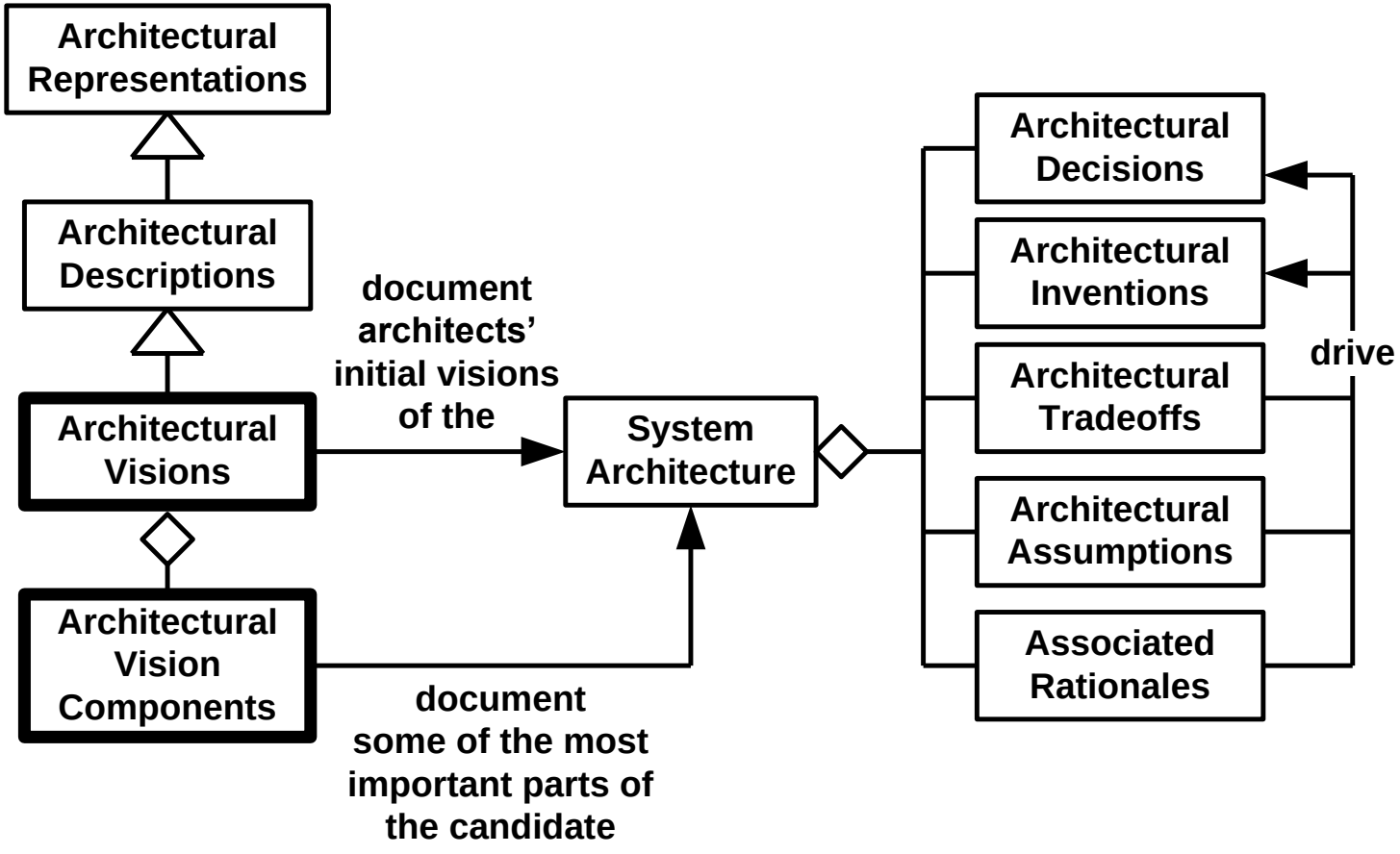
Architectural Vision Component

one of the more important actual or potential architectural decisions, inventions, or tradeoffs addressing one or more architectural concerns

Note that multiple candidate architectural visions are often created before one is selected and completed to produce the actual architecture

© 2009 Carnegie Mellon University

Architecture Visions and Vision Components - Ontology



© 2009 Carnegie Mellon University

Topics

Motivation

MFESA Overview

MFESA Ontology of Concepts and Terminology

MFESA Metamodel of Reusable Method Components

MFESA Repository of Reusable Method Components

- **Architectural Work Units and Work Products**
- **Architectural Workers**

MFESA Metamethod

Conclusion

© 2009 Carnegie Mellon University

MFESA Metamodel

A Metamodel is a Model of a Model.

MFESA Metamodel defines three Foundational Types of Reusable Method Components.

Based on OPEN Process Framework Metamodel.

Simplification of ISO/IEC 24744

Not based on OMG Metamodel.

© 2009 Carnegie Mellon University

System Architecture Engineering – Methods and Processes

System Architecture Engineering Method

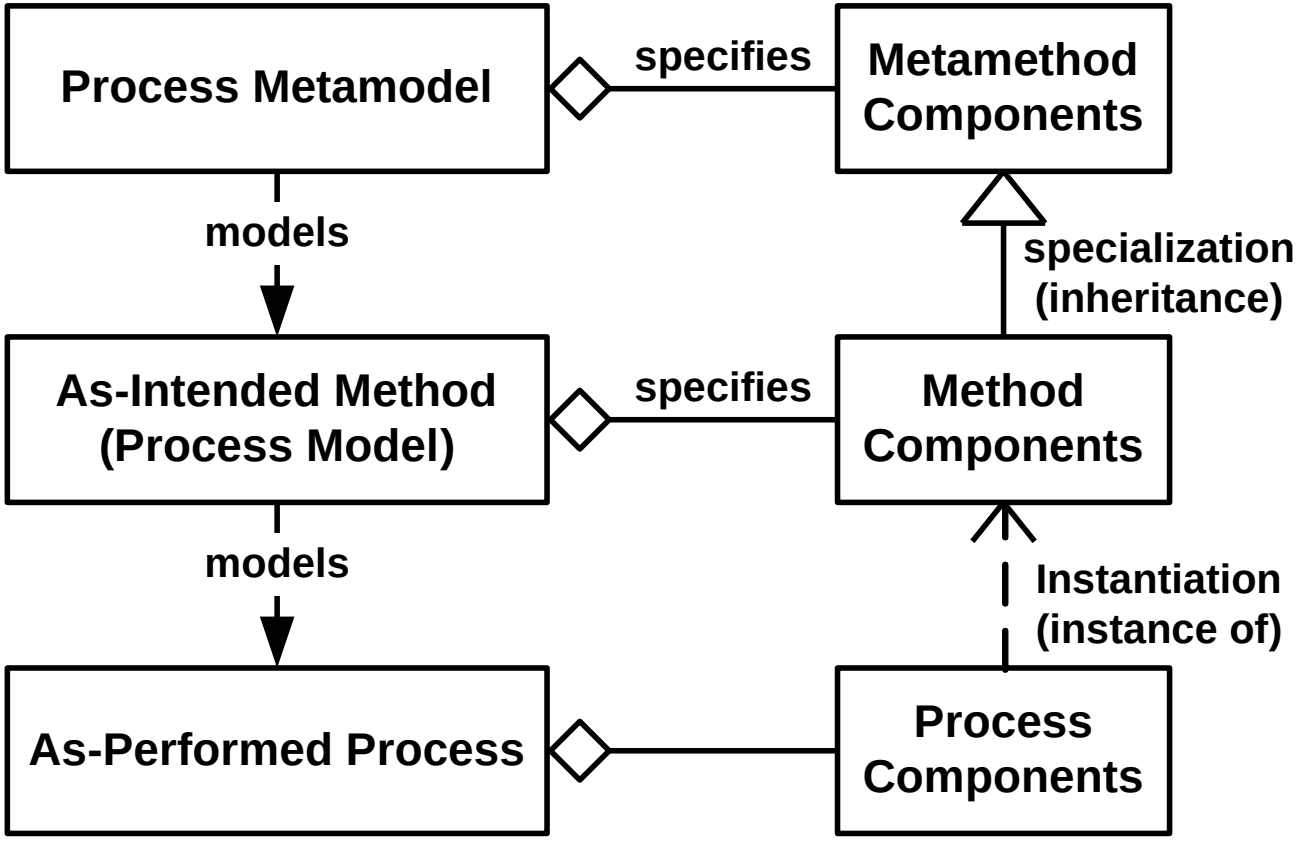
a systematic, documented, intended way that system architecture engineering *should* be performed

System Architecture Engineering Process

an *actual* way that system architecture engineering is performed in practice on an endeavor

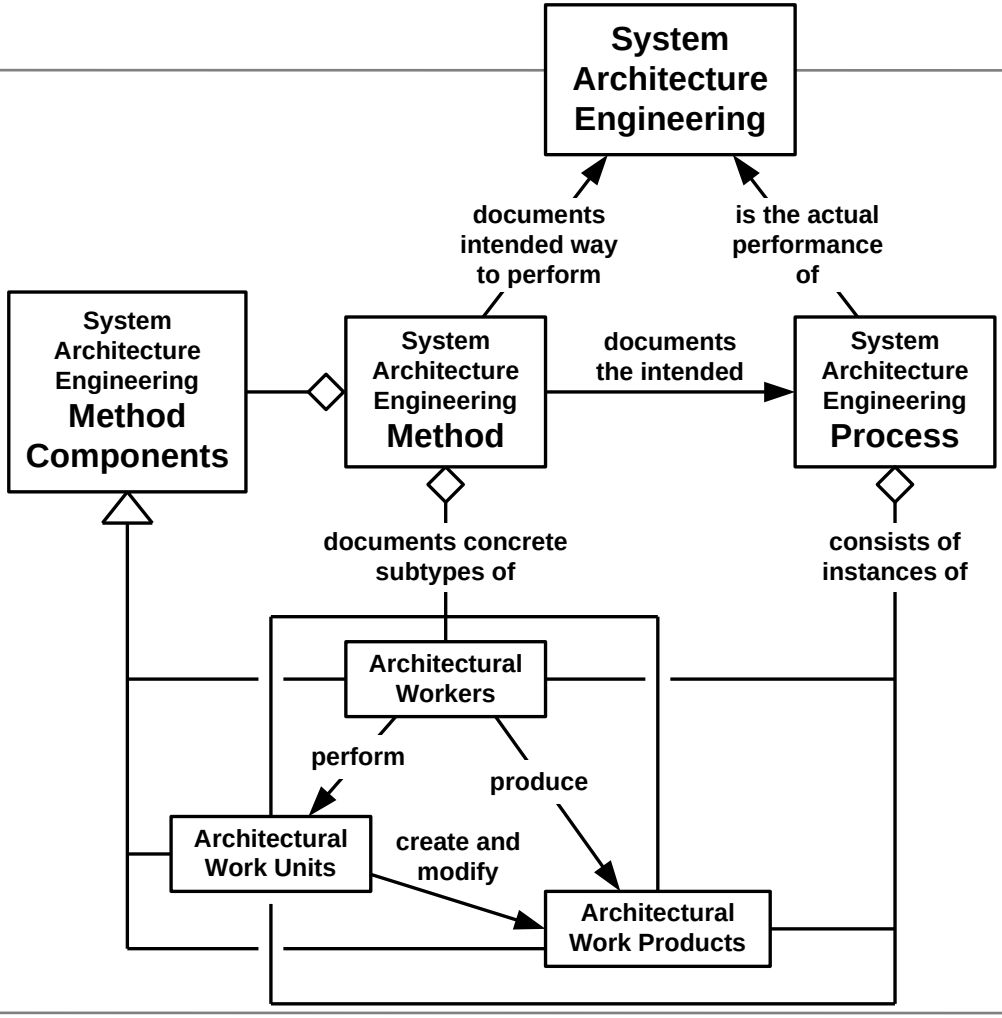
© 2009 Carnegie Mellon University

Method Engineering Models



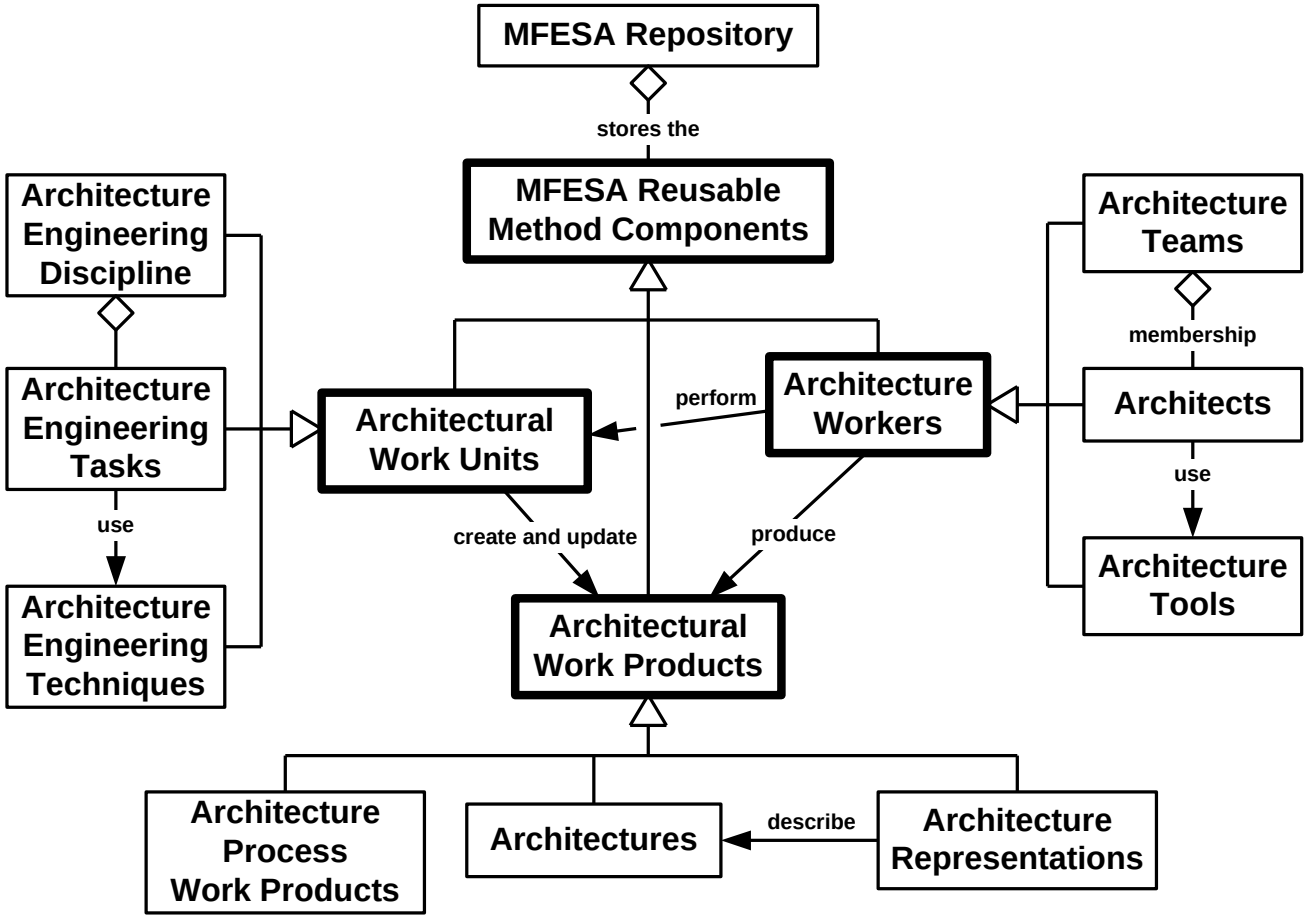
© 2009 Carnegie Mellon University

Method vs. Process



© 2009 Carnegie Mellon University

MFESA Metamodel of Reusable Method Components



© 2009 Carnegie Mellon University

Topics

Motivation

MFESA Overview

MFESA Ontology of Concepts and Terminology

MFESA Metamodel of Reusable Method Components

MFESA Repository of Reusable Method Components

- **Architectural Work Units and Work Products**
- **Architectural Workers**

MFESA Metamethod

Conclusion

© 2009 Carnegie Mellon University

MFESA Repository

Stores reusable system architecture engineering method components:

- Architecture Work Units
- Architecture Work Products
- Architecture Workers

Should provide easy access to method components:

- Identification and selection of relevant method components
- Tailoring of selected method components
- Configuration management of method components

© 2009 Carnegie Mellon University

Topics

Motivation

MFESA Overview

MFESA Ontology of Concepts and Terminology

MFESA Metamodel of Reusable Method Components

MFESA Repository of Reusable Method Components

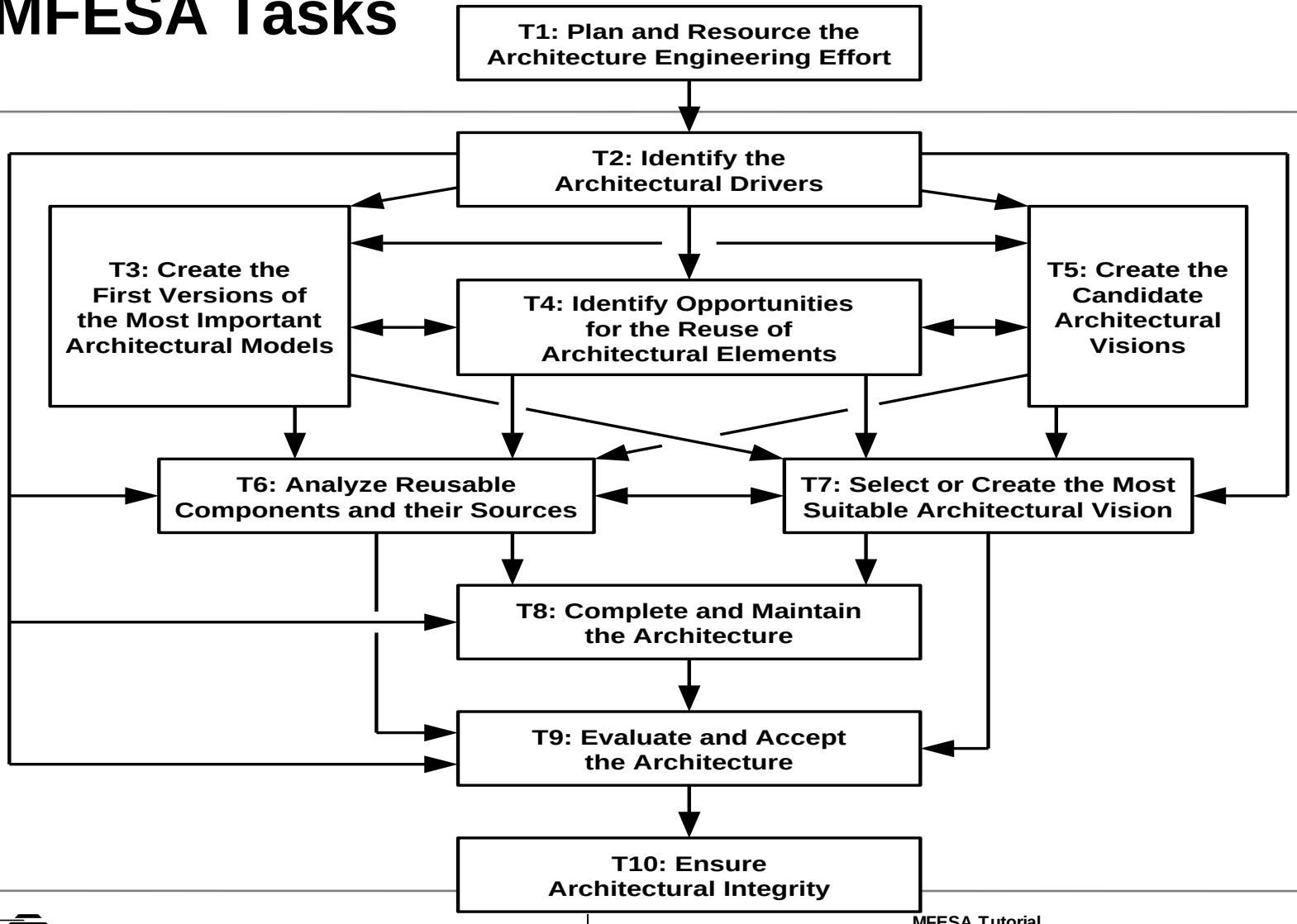
- **Architectural Work Units and Work Products**
- **Architectural Workers**

MFESA Metamethod

Conclusion

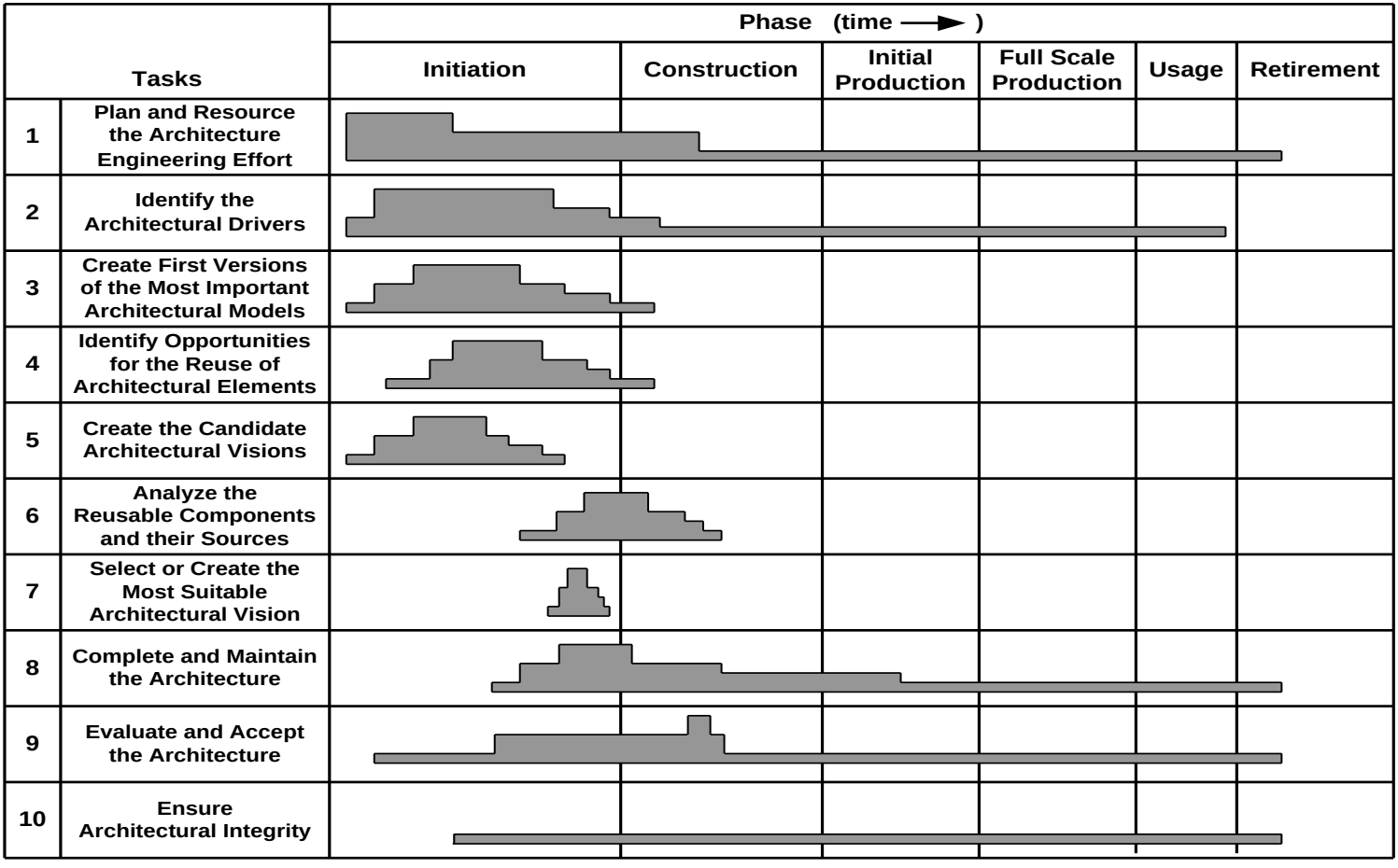
© 2009 Carnegie Mellon University

MFESA Tasks



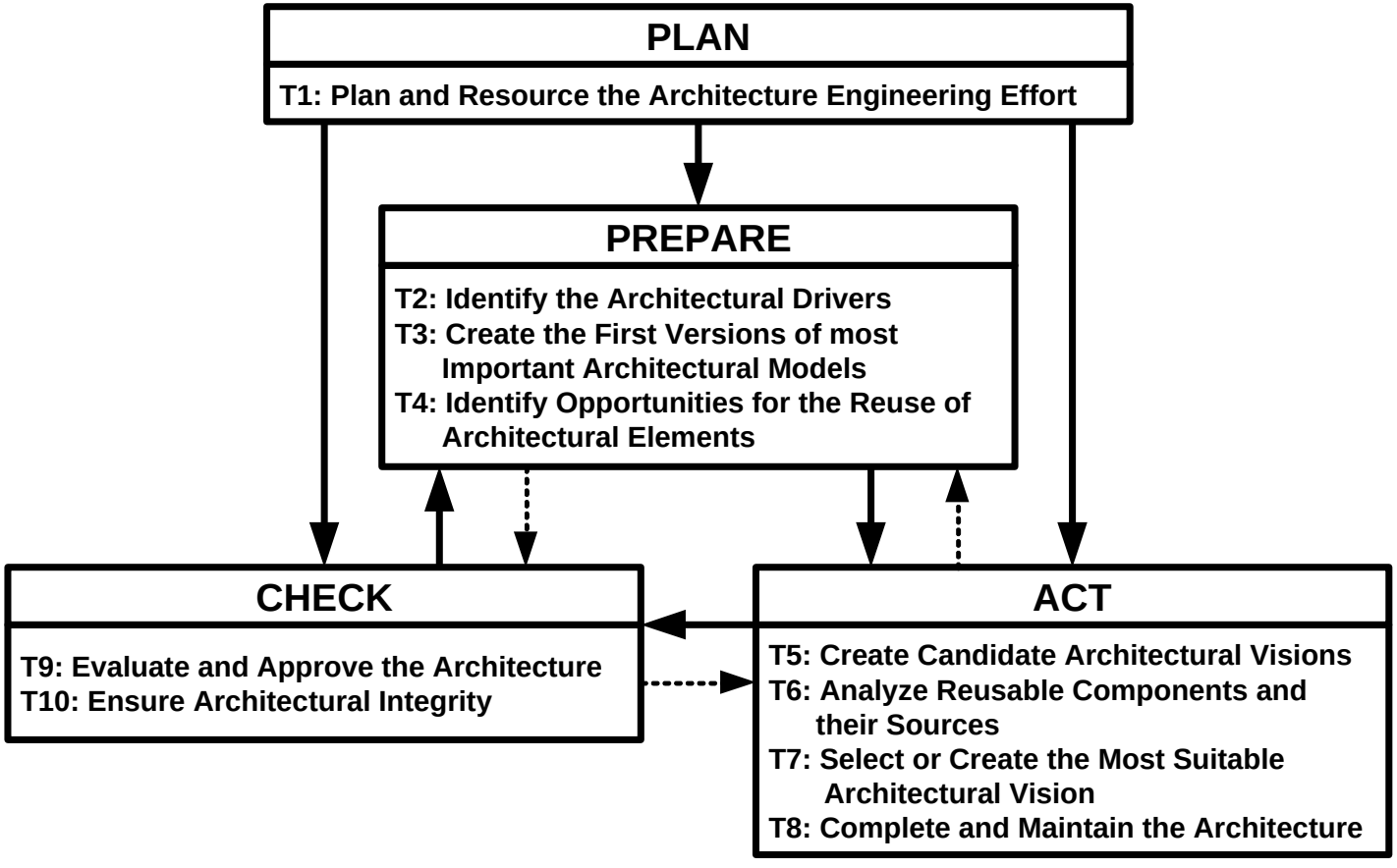
© 2009 Carnegie Mellon University

Effort by MFESA Task



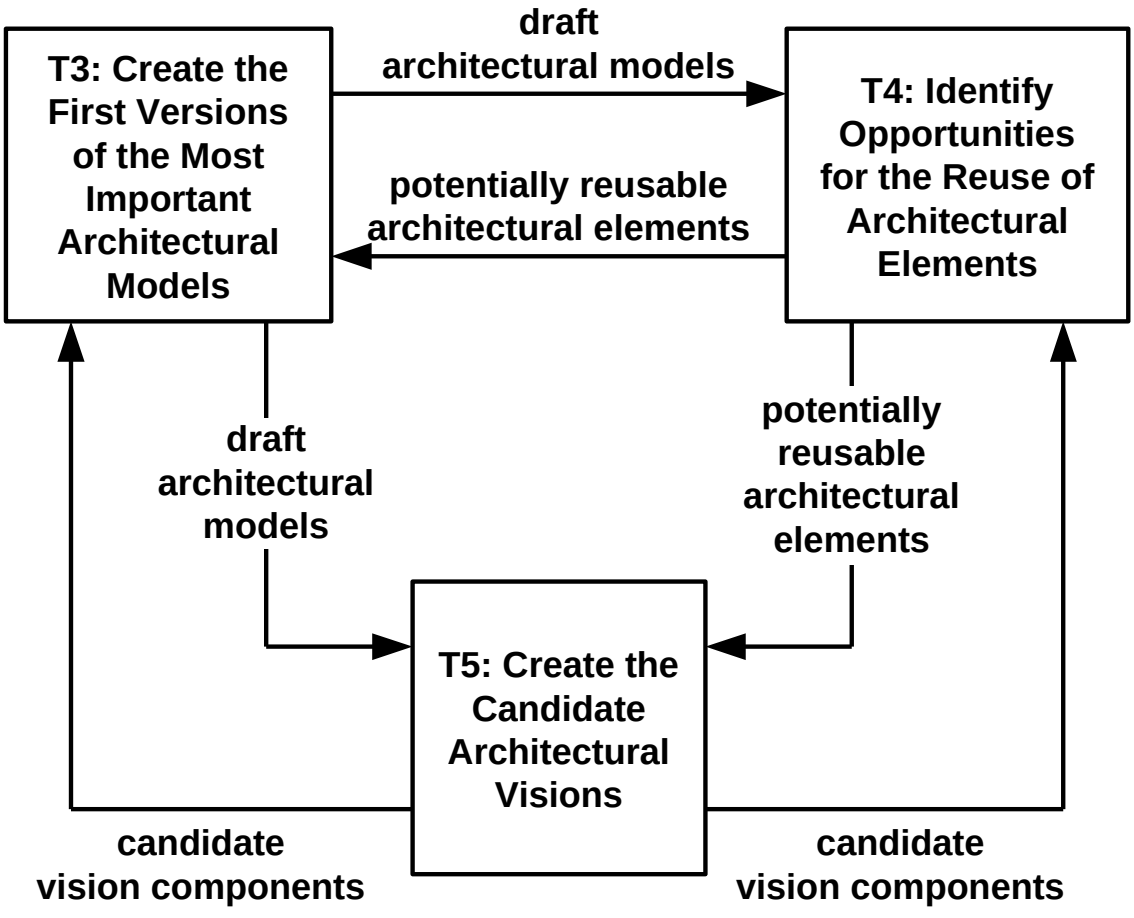
© 2009 Carnegie Mellon University

Plan, Prepare, Act, and Check



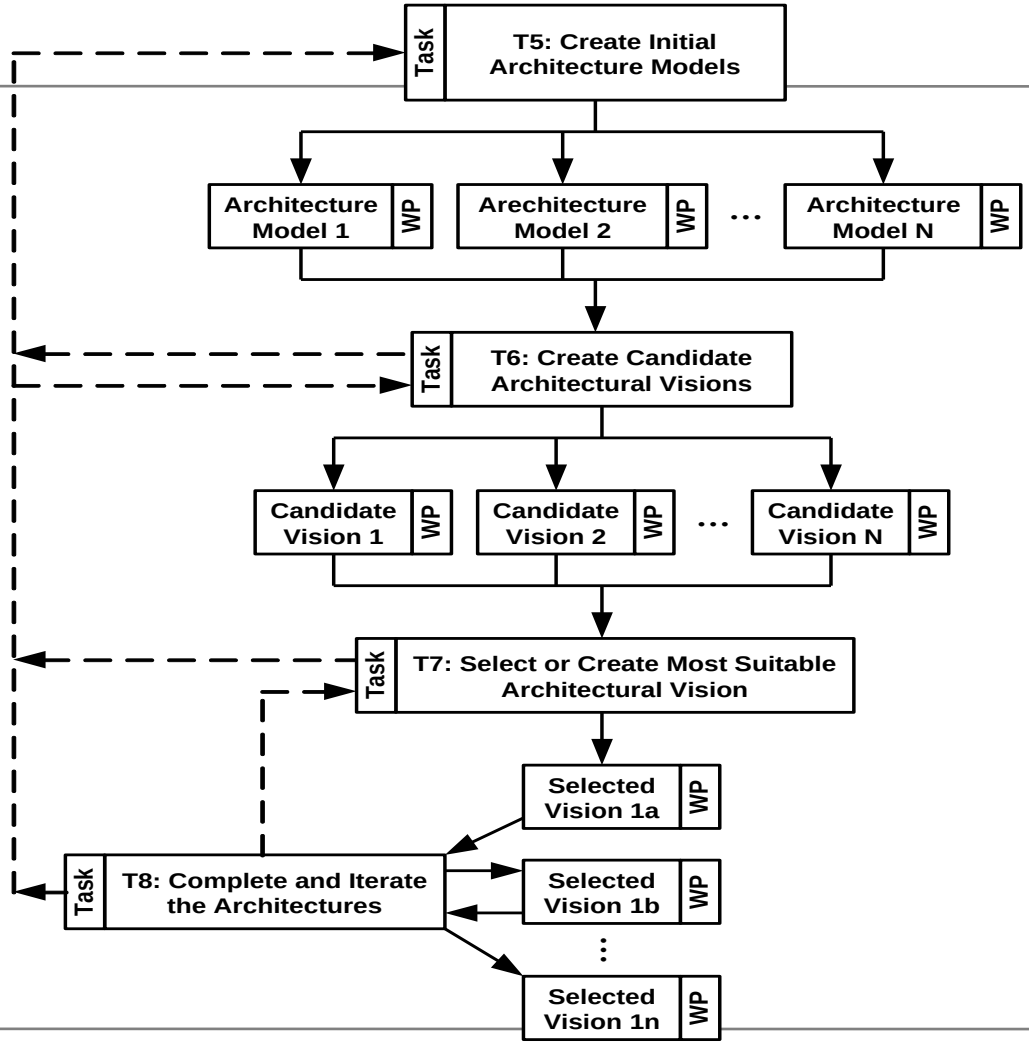
© 2009 Carnegie Mellon University

Concurrent MFESA Tasks



© 2009 Carnegie Mellon University

Architectural Visions - Flow



© 2009 Carnegie Mellon University

MFESA Task 1) Plan and Resource the Architecture Engineering Effort

Task 1) Plan and Resource Architecture Engineering Effort

Task 2) Identify the Architectural Drivers

Task 3) Create Initial Architectural Models

Task 4) Identify Opportunities for Reuse of Architectural Elements

Task 5) Create Candidate Architectural Visions

Task 6) Analyze Reusable Components and their Sources

Task 7) Select or Create Most Suitable Architectural Vision

Task 8) Complete and Maintain the Architecture

Task 9) Evaluate and Accept the Architecture

Task 10) Ensure Architectural Integrity

© 2009 Carnegie Mellon University

MFESA Task 1) Plan and Resource the Architecture Engineering Effort

Goal:

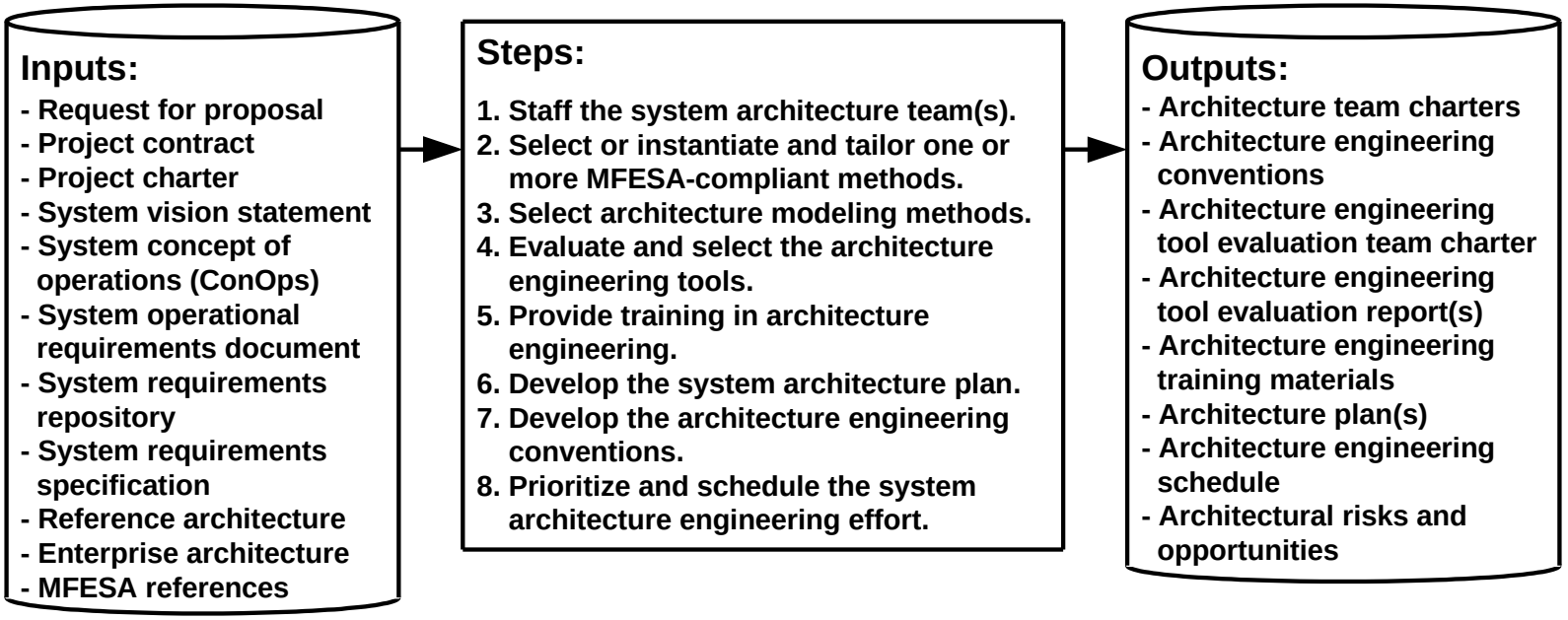
- Prepare the system engineering team to engineer the system architecture and its representations.

Objectives:

- Staff and train system architecture teams to engineer the system architecture.
- Develop and document the system architecture engineering method.
- Develop plans, standards, and procedures for engineering the system architecture.
- Prioritize and schedule the system architecture engineering effort.

© 2009 Carnegie Mellon University

MFESA Task 1) Plan and Resource the Architecture Engineering Effort



© 2009 Carnegie Mellon University

MFESA Task 1) Plan and Resource the Architecture Engineering Effort

Guidelines

- Properly staff the top-level architecture team(s).
- Properly plan the architecture engineering effort.
- Produce and maintain a proper and sufficient schedule.
- Reuse or create appropriate MFESA method(s).
- Select appropriate architecture modeling method(s).
- Select appropriate architecture engineering tools.
- Provide appropriate training.

© 2009 Carnegie Mellon University

MFESA Task 1) Plan and Resource the Architecture Engineering Effort

Pitfalls

- Architects produce incomplete architecture plans and conventions.
- Management provides inadequate resources.
- Management provides inadequate staff and stakeholder training.
- Architects lack authority.
- Architects instantiate the entire MFESA repository without tailoring.
- Tool vendors drive architecture engineering and modeling methods.
- Planning and resourcing are unsynchronized.
- Planning and resourcing are only done once up front.

© 2009 Carnegie Mellon University

MFESA Task 2)
Identify the Architectural Drivers

Task 1) Plan and Resource Architecture Engineering Effort

Task 2) Identify the Architectural Drivers

Task 3) Create Initial Architectural Models

Task 4) Identify Opportunities for Reuse of Architectural Elements

Task 5) Create Candidate Architectural Visions

Task 6) Analyze Reusable Components and their Sources

Task 7) Select or Create Most Suitable Architectural Vision

Task 8) Complete and Maintain the Architecture

Task 9) Evaluate and Accept the Architecture

Task 10) Ensure Architectural Integrity

© 2009 Carnegie Mellon University

MFESA Task 2)
Identify the Architectural Drivers

Goal:

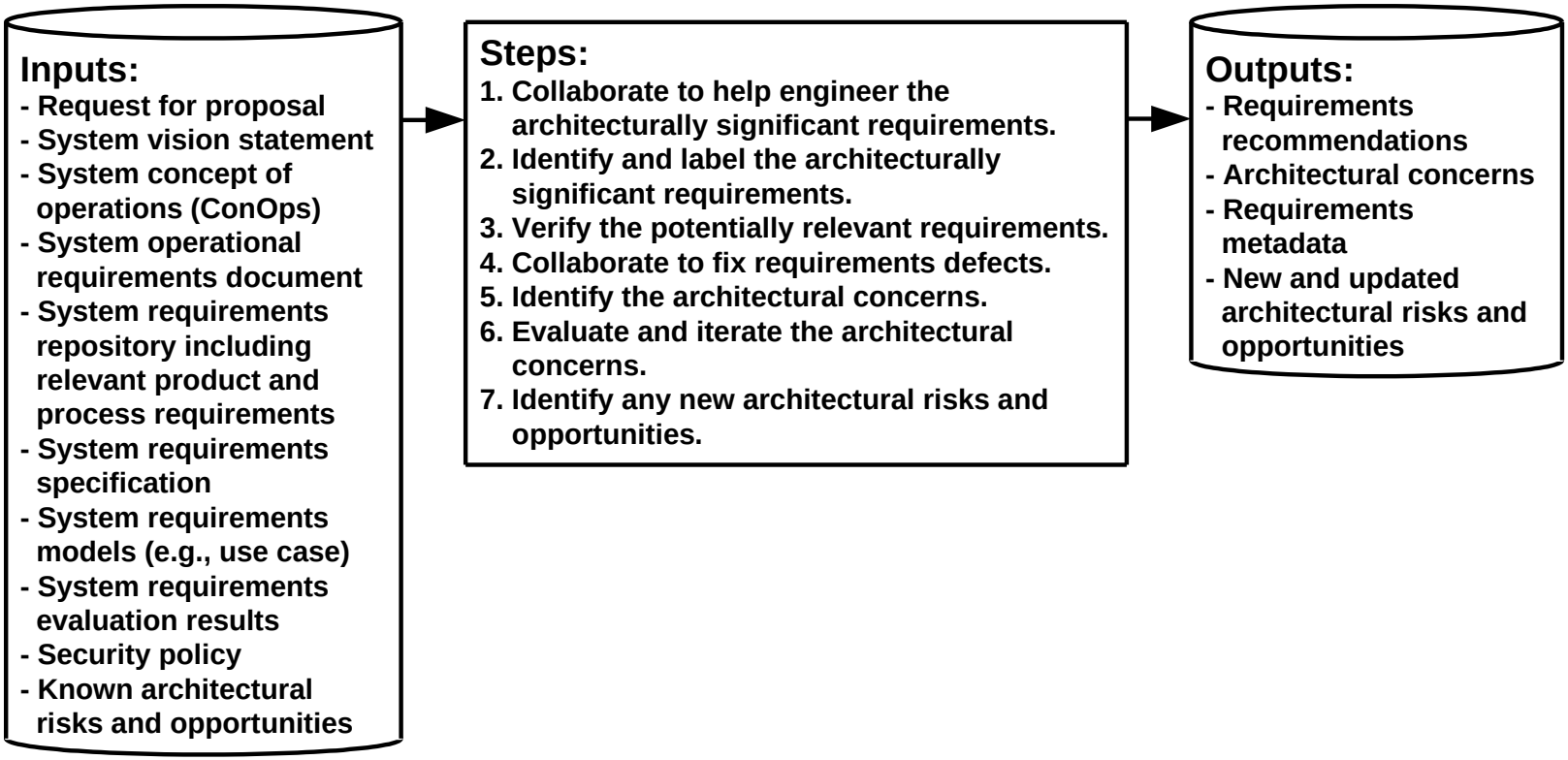
- Identify the architecturally significant product and process requirements that drive the development of the system architecture.

Objectives:

- Understand and verify the product and process requirements that have been allocated to the system or subsystem being architected.
- Categorize sets of related architecturally significant requirements into cohesive architectural concerns.
- Provide a set of architectural concerns to drive the:
  - Identification of potential opportunities for architectural reuse.
  - Analysis of potentially reusable components and their sources.
  - Creation of an initial set of draft architectural models.
  - Creation of a set of competing candidate architectural visions.
  - Selection of a single architectural vision judged most suitable.
  - Completion and maintenance of the resulting system architecture.
  - Evaluation and acceptance of the system architecture.

© 2009 Carnegie Mellon University

MFESA Task 2)
Identify the Architectural Drivers



© 2009 Carnegie Mellon University

MFESA Task 2)
Identify the Architectural Drivers

Guidelines

- Collaborate closely with the requirements team.
- Notify the requirements team(s) of relevant requirements defects.
- Consider the impact of the architecture on the requirements.
- Respect team boundaries and responsibilities.
- If necessary, clarify relevant requirements with the stakeholders.
- Concentrate on the architecturally significant requirements.
- Quality attributes can be architectural concerns too.
- Formally manage architectural risks.

© 2009 Carnegie Mellon University

MFESA Task 2)
Identify the Architectural Drivers

Pitfalls

- All requirements are architecturally significant.
- Well-engineered architecturally significant requirements are lacking.
- Architects rely excessively on functional requirements.
- The architects ignore the architecturally significant functional and process requirements.
- Specialty engineering requirements are misplaced and ignored.
- Unnecessary constraints are imposed on the architecture.
- Architects engineer architecturally significant requirements.
- Requirements lack relevant metadata.
- Architects fail to clarify architectural drivers.

© 2009 Carnegie Mellon University

MFESA Task 3)
Create Initial Architectural Models

Task 1) Plan and Resource Architecture Engineering Effort

Task 2) Identify the Architectural Drivers

Task 3) Create Initial Architectural Models

Task 4) Identify Opportunities for Reuse of Architectural Elements

Task 5) Create Candidate Architectural Visions

Task 6) Analyze Reusable Components and their Sources

Task 7) Select or Create Most Suitable Architectural Vision

Task 8) Complete and Maintain the Architecture

Task 9) Evaluate and Accept the Architecture

Task 10) Ensure Architectural Integrity

© 2009 Carnegie Mellon University

MFESA Task 3)
Create Initial Architectural Models

Goal:

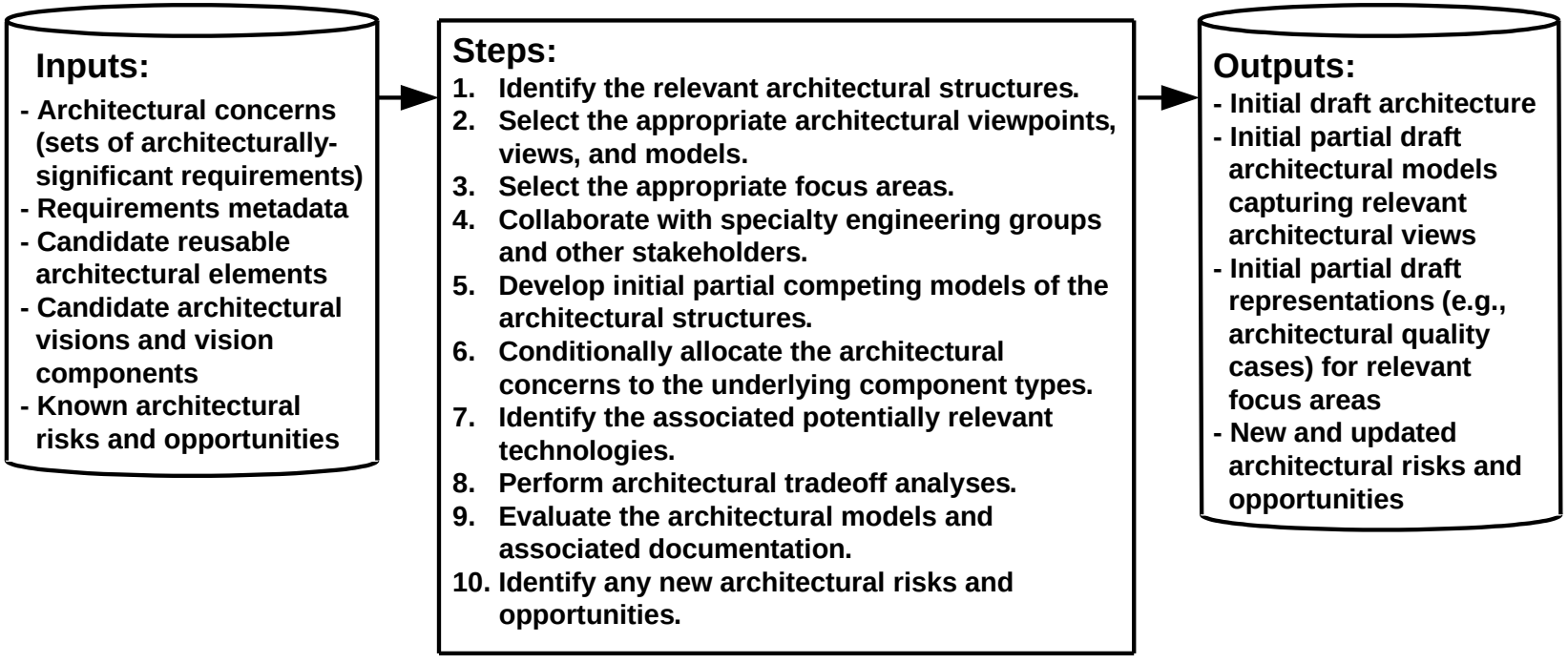
- Create an initial set of partial draft architectural models of the system architecture.

Objectives:

- Capture the most important candidate elements of the eventual system architecture (i.e., architectural decisions, inventions, trade-offs, assumptions, and rationales).
- Provide the most important views and focus areas of the system architecture.
- Ensure that these candidate architectural elements sufficiently support the relevant architectural concerns.
- Provide a foundation of architectural models from which to create a set of competing candidate architectural visions.

© 2009 Carnegie Mellon University

MFESA Task 3)
Create Initial Architectural Models



© 2009 Carnegie Mellon University

MFESA Task 3)
Create Initial Architectural Models

Guidelines

- Perform architectural trade-off analysis.
- Reuse architectural principles, heuristics, styles, patterns, vision components, and metaphors.
- Use iterative, incremental, and parallel development.
- Begin developing logical models before physical models and static models before dynamic models.
- Do not overemphasize the physical decomposition hierarchy.
- Use explicitly documented system partitioning criteria.
- Model concurrency.
- Consider the impact of hardware decisions on usability and software.
- Consider human limitations when allocating system functionality to manual procedures.
- Do not start from scratch.
- Formally manage architectural risks.

© 2009 Carnegie Mellon University

MFESA Task 3)
Create Initial Architectural Models

Pitfalls

- The architects succumb to analysis paralysis.
- The architects engineer too few architectural models.
- The architects engineer inappropriate models and views.
- The architects construct views but no focus areas.
- Some stakeholders believe that the models are the architecture.
- Inconsistencies exist between models, views, and focus areas.
- The architects use inappropriate architectural patterns.
- System decomposition is performed by the acquisition organization.

© 2009 Carnegie Mellon University

MFESA Task 4) Identify Opportunities for Reuse of Architectural Elements

Task 1) Plan and Resource Architecture Engineering Effort

Task 2) Identify the Architectural Drivers

Task 3) Create Initial Architectural Models

Task 4) Identify Opportunities for Reuse of Architectural Elements

Task 5) Create Candidate Architectural Visions

Task 6) Analyze Reusable Components and their Sources

Task 7) Select or Create Most Suitable Architectural Vision

Task 8) Complete and Maintain the Architecture

Task 9) Evaluate and Accept the Architecture

Task 10) Ensure Architectural Integrity

© 2009 Carnegie Mellon University

MFESA Task 4) Identify Opportunities for Reuse of Architectural Elements

Goal:

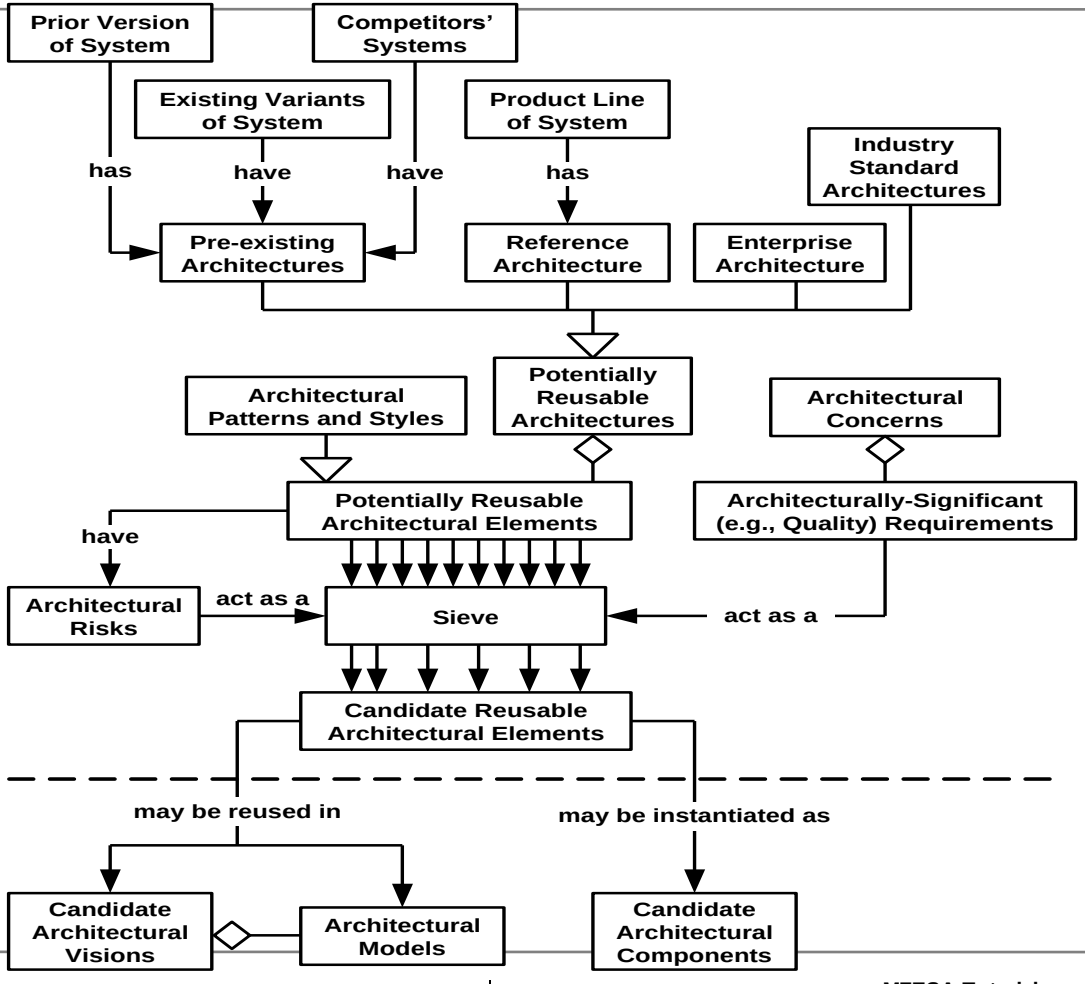
- Identify any opportunities to reuse existing architectural work products as part of the architecture of the target system or subsystem being developed. Any opportunities so identified become a collection of reusable architectural element candidates.

Objectives:

- Identify the architectural risks and opportunities for improving the architectures associated with the relevant legacy or existing system(s) should they be selected for reuse and incorporation within the target environment.
- Identify any additional architectural concerns due to the constraints associated with having legacy or existing architectures.
- Understand the relevant legacy or existing architectures sufficiently well to identify potentially reusable architectural elements.
- Provide a set of reusable architectural element candidates to influence (and possibly include in) a set of initial draft architectural models.

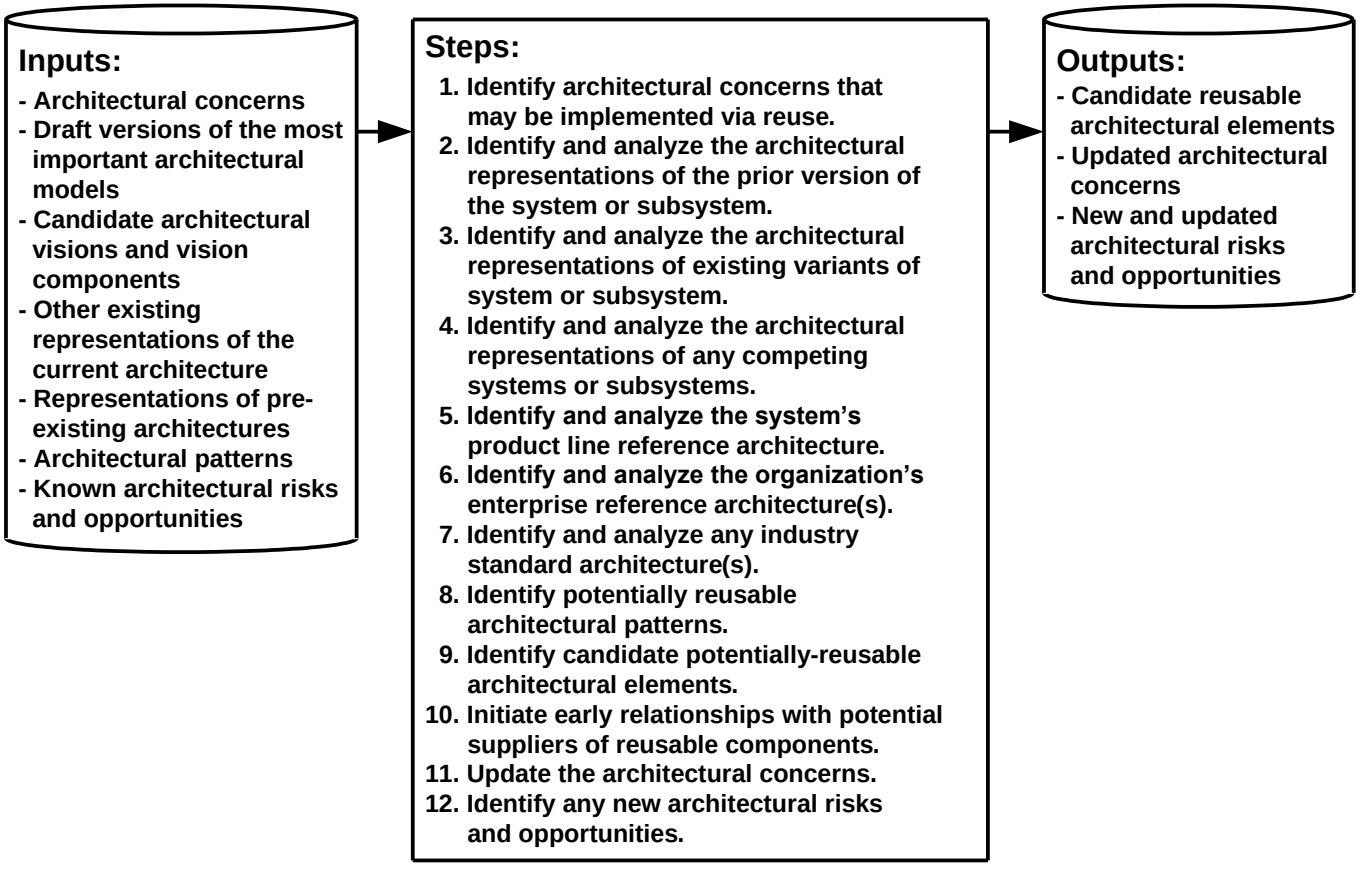
© 2009 Carnegie Mellon University

MFESA Task 4) Identify Opportunities for Reuse of Architectural Elements



© 2009 Carnegie Mellon University

MFESA Task 4) Identify Opportunities for Reuse of Architectural Elements



© 2009 Carnegie Mellon University

MFESA Task 4) Identify Opportunities for Reuse of Architectural Elements

Guidelines

- Do not start from scratch.
- Do not be excessively constrained by the past.
- Conform to the enterprise architecture.
- Conform to the product line reference architecture.
- Consider system architecture patterns.
- Identify opportunities for reuse in the architectural models.
- Formally manage architectural risks.

© 2009 Carnegie Mellon University

MFESA Task 4) Identify Opportunities for Reuse of Architectural Elements

Pitfalls

- The architects start from scratch.
- The architects ignore past lessons learned.
- The architects over-rely on previous architectures.
- The architects select specific OTS components too early.
- The architects assume reuse of immature architectural components.
- The architects assume the reuse of immature technologies.
- Inadequate information exists to determine reusability.

© 2009 Carnegie Mellon University

MFESA Task 5)
Create Candidate Architectural Visions

Task 1) Plan and Resource Architecture Engineering Effort

Task 2) Identify the Architectural Drivers

Task 3) Create Initial Architectural Models

Task 4) Identify Opportunities for Reuse of Architectural Elements

Task 5) Create Candidate Architectural Visions

Task 6) Analyze Reusable Components and their Sources

Task 7) Select or Create Most Suitable Architectural Vision

Task 8) Complete and Maintain the Architecture

Task 9) Evaluate and Accept the Architecture

Task 10) Ensure Architectural Integrity

© 2009 Carnegie Mellon University

MFESA Task 5)
Create Candidate Architectural Visions

Goal:

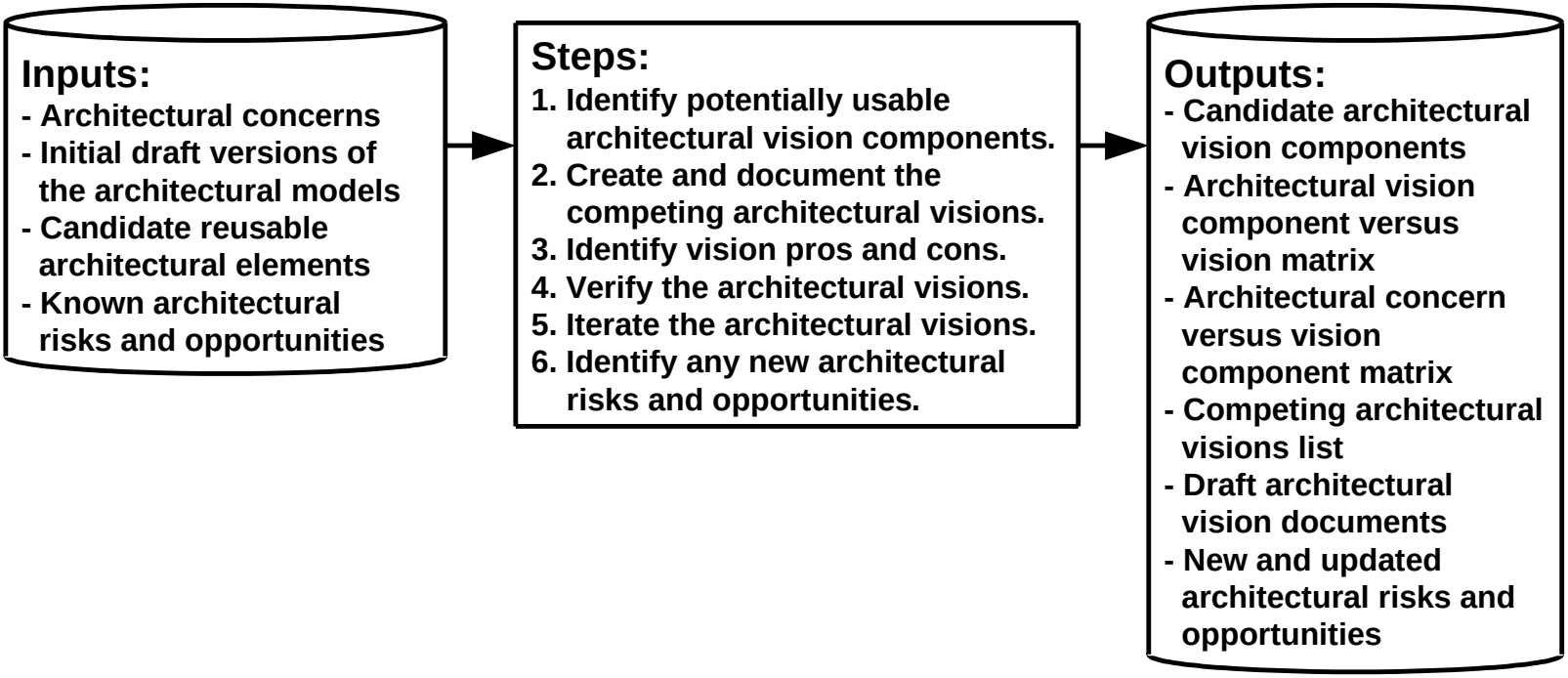
- Create multiple candidate architectural visions of the system architecture.

Objectives:

- Verify that the candidate subsystem architectural visions sufficiently support the relevant architecture concerns.
- Provide a sufficiently large and appropriate set of competing candidate architectural visions from which a single vision may be selected as most suitable.

© 2009 Carnegie Mellon University

MFESA Task 5)
Create Candidate Architectural Visions



© 2009 Carnegie Mellon University

MFESA Task 5)
Create Candidate Architectural Visions

Architectural Vision Component vs. Architectural Vision Matrix		Candidate Architectural Visions			
		Architectural Vision 1	Architectural Vision 2	Architectural Vision 3	Architectural Vision 4
Candidate Architectural Vision Components	Component 1	X	X	X	X
	Component 2	X		X	X
	Component 3	X	X		
	Component 4	X	X	X	
	Component 5	X		X	X
	Component 6	X	X	X	
	Component 7	X		X	X
	Component 8	X	X	X	X
	Component 9		X		X
	Component 10		X	X	X
	Component 11			X	
	Component 12			X	X
	Component 13			X	X

© 2009 Carnegie Mellon University

MFESA Task 5)
Create Candidate Architectural Visions

Example Architectural Concern vs. Vision Component Matrix

Architectural Concern vs. Vision Component Matrix		Architectural Vision Components												
		User Identifier and Pass Phrase	COTS Security Smart Card	COTS Biometrics Reader	Locked Rooms with Keyed Access	Guards with Security Cameras	Dedicated Encryption Decryption HW Server	COTS Encryption Decryption Software	Encrypted Messages	Encrypted Database Records	COTS Intrusion Detection Software	COTS Antivirus Software	Digital Signature	Single Sign-On
Architectural Concerns	Access Control	++	++	++	++	++	0	0	0	0	0	0	+	++
	Confidentiality	+	+	+	+	+	++	++	++	++	0	0	0	0
	Integrity (Message)	+	+	+	+	+	0	0	+	+	+	++	+	0
	Integrity (Software)	+	+	+	+	+	0	0	0	0	+	++	+	0
	Integrity (Data)	+	+	+	+	+	0	0	0	+	+	++	+	0
	Nonrepudiation	0	0	0	0	0	0	0	0	+	+	+	++	0
	Availability	-	-	0	0	0	0	0	0	-	+	+	0	0
	Cost	++	-	- -	0	0	- -	- -	- -	0	-	+	-	-
	Performance	-	-	0	0	0	+	-	-	- -	-	-	-	+
	Usability	-	-	0	0	0	0	0	0	0	0	0	-	++

© 2009 Carnegie Mellon University

MFESA Task 5)
Create Candidate Architectural Visions

Guidelines

- Complete candidate architectural visions to appropriate level of detail.
- Prepare architectural components for OTS incorporation.
- Identify an appropriate number of candidate architectural visions.
- Formally manage architectural risks.

© 2009 Carnegie Mellon University

MFESA Task 5)
Create Candidate Architectural Visions

Pitfalls

- The architects engineer only one architectural vision.
- Management provides insufficient resources.
- Management confuses the architectural vision with the completed architecture.
- Management does not permit architects to make mistakes.
- The architects compare the architectural visions prematurely.
- The architects do not compare the pros and cons of the candidate visions.

© 2009 Carnegie Mellon University

MFESA Task 6) Analyze Reusable Components and their Sources

Task 1) Plan and Resource Architecture Engineering Effort

Task 2) Identify the Architectural Drivers

Task 3) Create Initial Architectural Models

Task 4) Identify Opportunities for Reuse of Architectural Elements

Task 5) Create Candidate Architectural Visions

Task 6) Analyze Reusable Components and their Sources

Task 7) Select or Create Most Suitable Architectural Vision

Task 8) Complete and Maintain the Architecture

Task 9) Evaluate and Accept the Architecture

Task 10) Ensure Architectural Integrity

© 2009 Carnegie Mellon University

MFESA Task 6) Analyze Reusable Components and their Sources

Goal:

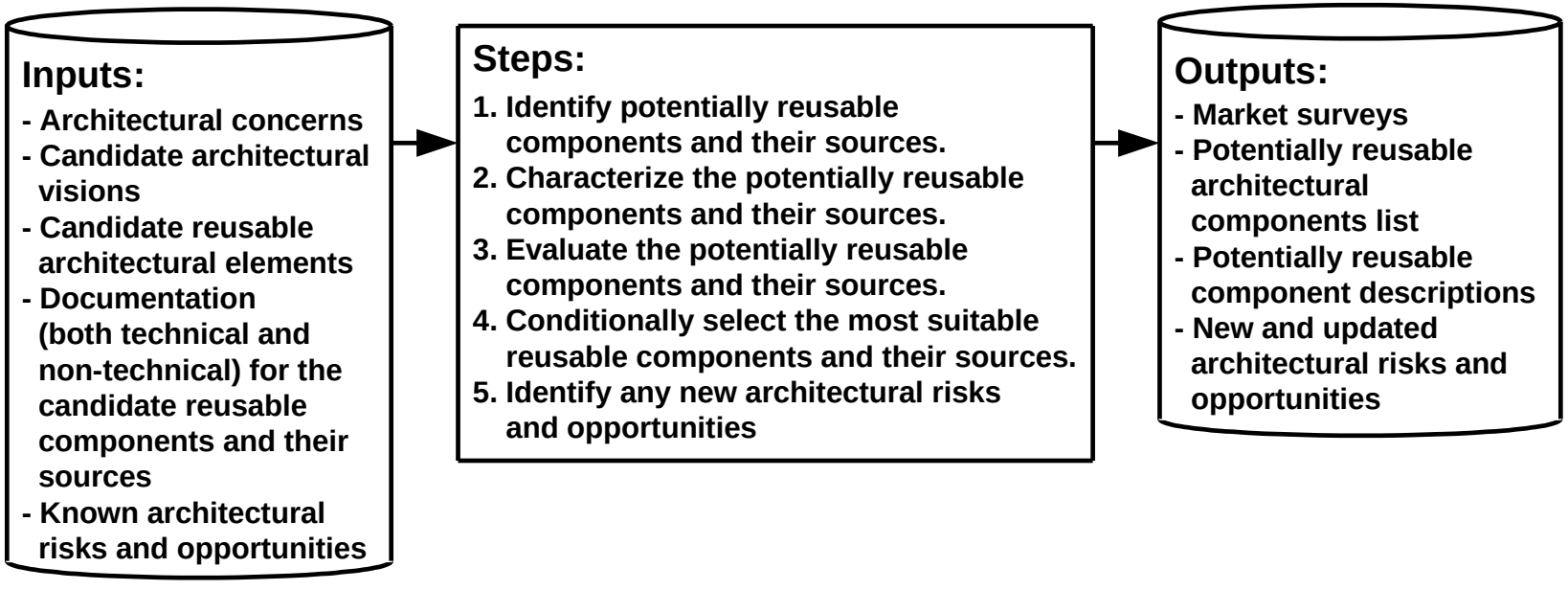
- Determine if any existing components are potentially reusable as part of the architecture of the current system or subsystem.

Objectives:

- Identify any existing components that are potentially reusable as part of the architecture of the current system or subsystem.
- Evaluate these components for suitability.
- Evaluate the sources of these components for suitability.
- Provide a set of potentially reusable components to influence (and possibly include in) a set of initial draft architectural models.

© 2009 Carnegie Mellon University

MFESA Task 6) Analyze Reusable Components and their Sources



© 2009 Carnegie Mellon University

MFESA Task 6) Analyze Reusable Components and their Sources

Guidelines

- Use appropriate decision techniques.
- Perform tasks 6 and 7 concurrently.
- Formally manage architectural risks.

© 2009 Carnegie Mellon University

MFESA Task 6) Analyze Reusable Components and their Sources

Pitfalls

- Authoritative stakeholders assume reuse will improve cost and schedule.
- Insufficient information exists for evaluation and reuse.
- Stakeholders have an unrealistic expectation of “exact fit.”
- Developers have little or no control over future changes.
- The source organization (e.g., vendor) fails to adequately maintain a reusable architectural component.
- Legal rights are unacceptable.
- Incompatibilities exist with underlying technologies.

© 2009 Carnegie Mellon University

MFESA Task 7) Select or Create the Most Suitable Architectural Vision

Task 1) Plan and Resource Architecture Engineering Effort

Task 2) Identify the Architectural Drivers

Task 3) Create Initial Architectural Models

Task 4) Identify Opportunities for Reuse of Architectural Elements

Task 5) Create Candidate Architectural Visions

Task 6) Analyze Reusable Components and their Sources

Task 7) Select or Create Most Suitable Architectural Vision

Task 8) Complete and Maintain the Architecture

Task 9) Evaluate and Accept the Architecture

Task 10) Ensure Architectural Integrity

© 2009 Carnegie Mellon University

MFESA Task 7) Select or Create the Most Suitable Architectural Vision

Goal:

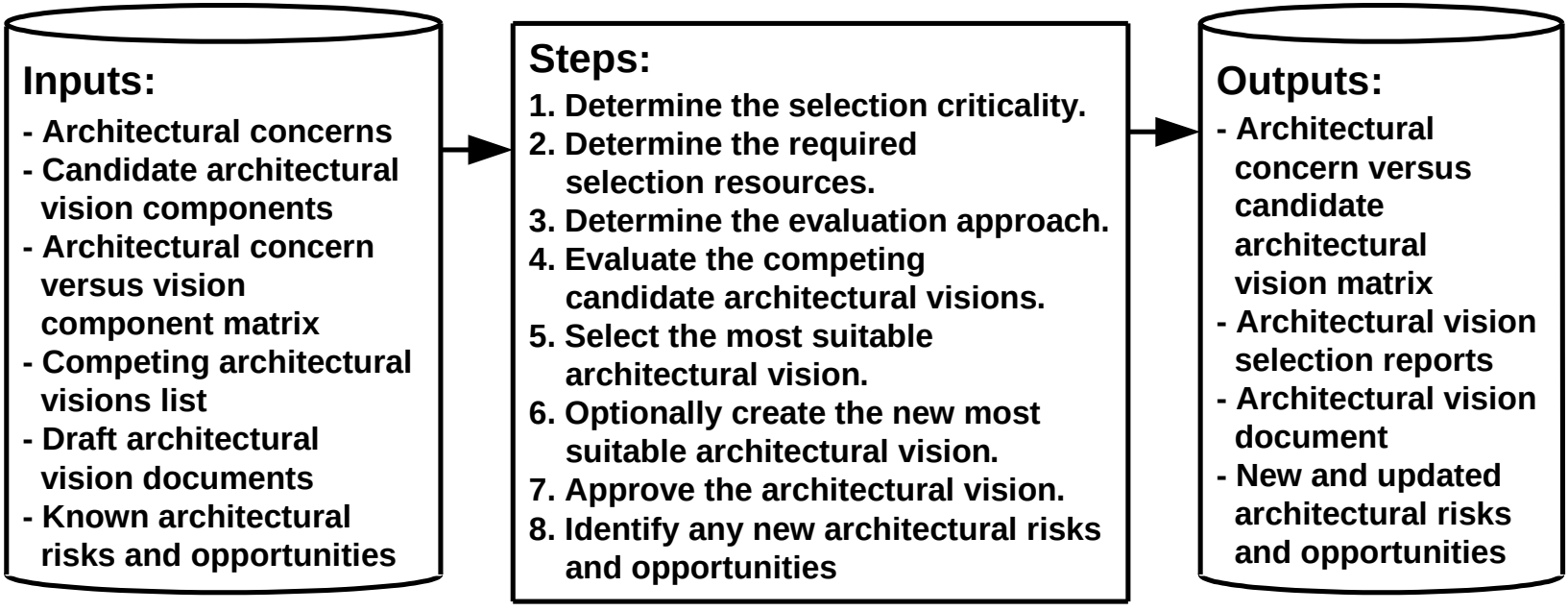
- Obtain a single architectural vision for the system or subsystem architecture from the competing candidate visions.

Objectives:

- Ensure that the selected architectural vision has been properly judged to be most suitable for the system or subsystem architecture.
- Provide a proper foundation on which to complete the engineering of the system or subsystem architecture.

© 2009 Carnegie Mellon University

MFESA Task 7) Select or Create the Most Suitable Architectural Vision



© 2009 Carnegie Mellon University

MFESA Task 7) Select or Create the Most Suitable Architectural Vision

Architectural Concern vs. Architectural Visions Matrix		Candidate Competing Architectural Visions			
		Architectural Vision 1	Architectural Vision 2	Architectural Vision 3	Architectural Vision 4
Architectural Concerns	Availability	+	+	++	+
	Development Cost	0	++	--	0
	Development Schedule	+	++	--	-
	Interoperability	+	-	+	+
	Performance	+	--	+	++
	Portability	0	+	0	+
	Reliability	+	-	++	-
	Safety	-	-	++	0
	Security	-	++	-	0
	Usability	-	0	-	+

© 2009 Carnegie Mellon University

MFESA Task 7) Select or Create the Most Suitable Architectural Vision

Guidelines

- Ensure a commensurate approach.
- Ensure a consistent evaluation approach.
- Ensure complete evaluation criteria.
- Avoid unwarranted assumptions.
- Use common sense when using decision methods to select the most suitable candidate architectural vision.
- Take reuse into account.
- Test reusable architectural component suitability.
- Maintain the architectural vision.
- Formally manage architectural risks.

© 2009 Carnegie Mellon University

MFESA Task 7) Select or Create the Most Suitable Architectural Vision

Pitfalls

- Architects use an inappropriate decision method.
- Management provides inadequate decision resources.
- Selecting the most suitable architectural vision is treated as just a technical decision.
- Stakeholders do not understand risks.
- The decision makers are weak.

© 2009 Carnegie Mellon University

MFESA Task 8)
Complete and Maintain the Architecture

Task 1) Plan and Resource Architecture Engineering Effort

Task 2) Identify the Architectural Drivers

Task 3) Create Initial Architectural Models

Task 4) Identify Opportunities for Reuse of Architectural Elements

Task 5) Create Candidate Architectural Visions

Task 6) Analyze Reusable Components and their Sources

Task 7) Select or Create Most Suitable Architectural Vision

Task 8) Complete and Maintain the Architecture

Task 9) Evaluate and Accept the Architecture

Task 10) Ensure Architectural Integrity

© 2009 Carnegie Mellon University

MFESA Task 8)
Complete and Maintain the Architecture

Goals:

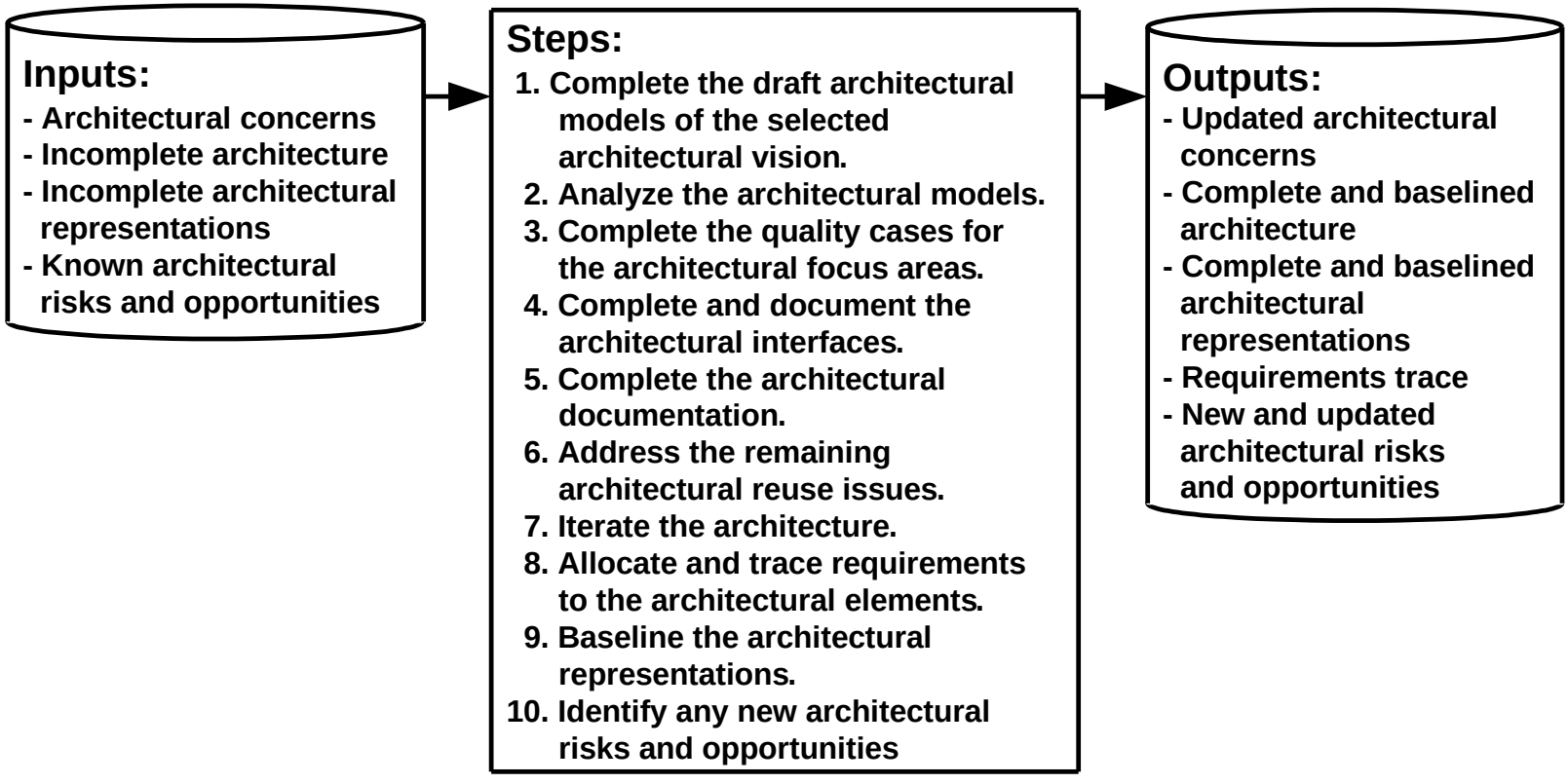
- Complete system or subsystem architecture based on the selected or created architectural vision.
- Maintain the system or subsystem architecture as the architecturally significant requirements change.

Objectives:

- Complete the interface aspects of the architectural.
- Complete the reuse aspects of the architecture.
- Complete the architectural representations (e.g., architectural models, quality cases, white-papers, and documents).
- Provide a system or subsystem architecture that can be evaluated and accepted by its authoritative stakeholders.

© 2009 Carnegie Mellon University

MFESA Task 8)
Complete and Maintain the Architecture



© 2009 Carnegie Mellon University

MFESA Task 8)
Complete and Maintain the Architecture

Guidelines

- Address all relevant types of interfaces.
- Maintain the architectural representations to maintain architectural integrity.
- Formally manage architectural risks.

© 2009 Carnegie Mellon University

MFESA Task 8)
Complete and Maintain the Architecture

Pitfalls

- Architecture engineering is done.
- Management provides inadequate resources.
- The architectural representations lack configuration control.
- The architecture is not maintained.
- A “beautiful” architecture is frozen solid.
- There is inadequate tool support for architecture maintenance.

© 2009 Carnegie Mellon University

MFESA Task 9)
Evaluate and Accept the Architecture

Task 1) Plan and Resource Architecture Engineering Effort

Task 2) Identify the Architectural Drivers

Task 3) Create Initial Architectural Models

Task 4) Identify Opportunities for Reuse of Architectural Elements

Task 5) Create Candidate Architectural Visions

Task 6) Analyze Reusable Components and their Sources

Task 7) Select or Create Most Suitable Architectural Vision

Task 8) Complete and Maintain the Architecture

Task 9) Evaluate and Accept the Architecture

Task 10) Ensure Architectural Integrity

© 2009 Carnegie Mellon University

MFESA Task 9)
Evaluate and Accept the Architecture

Goals:

- Monitor and determine the quality of the system or subsystem architecture and associated representations.
- Monitor and determine the quality of the process used to engineer the system or subsystem architecture.
- Provide information that can be used to determine the passage or failure of architectural milestones.
- Enable architectural defects, weaknesses, and risks to be fixed and managed before they negatively impact system quality and the success of the system development/enhancement project.
- Accept the system or subsystem architecture based on the results of the evaluations.

© 2009 Carnegie Mellon University

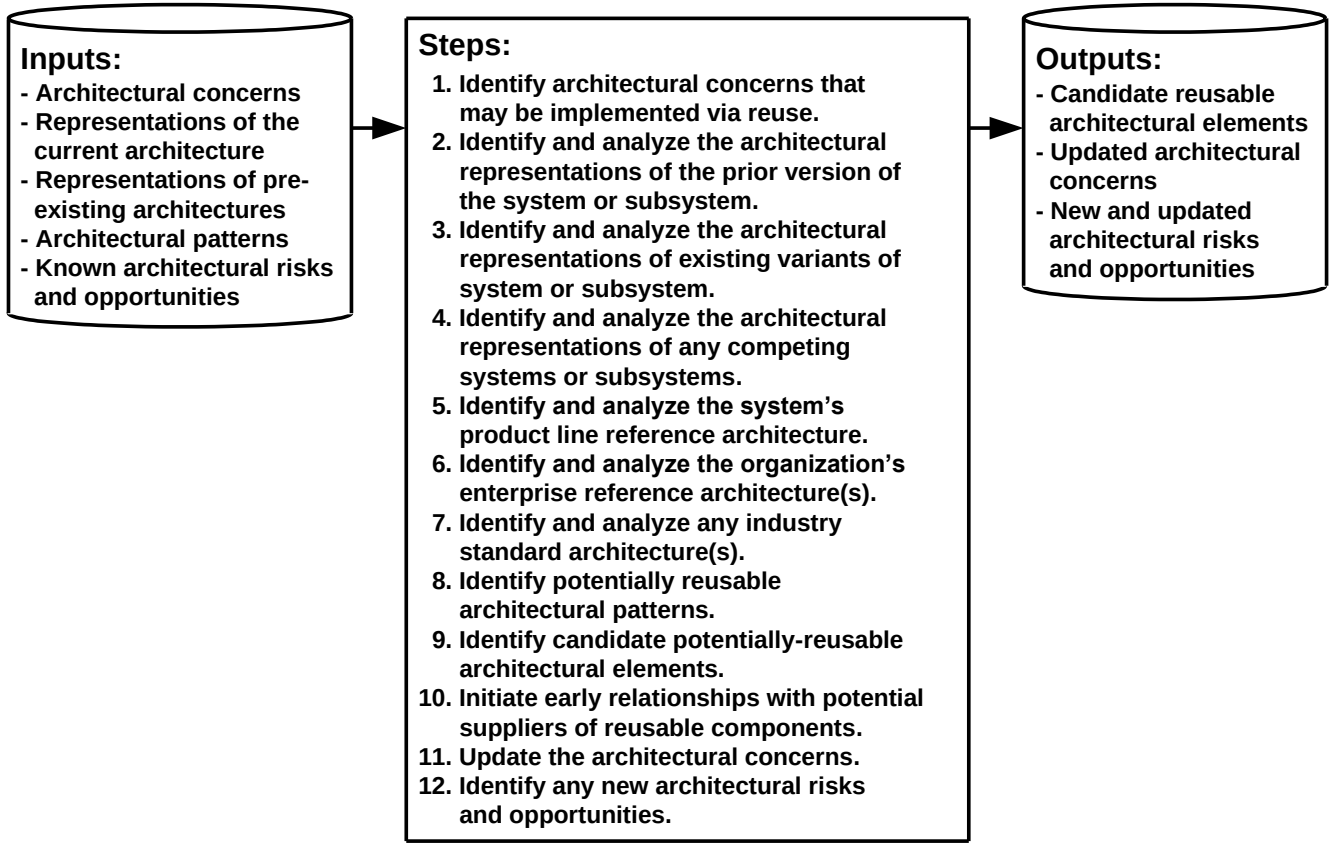
MFESA Task 9)
Evaluate and Accept the Architecture

Objectives:

- **Internally verify** the system or subsystem architecture so that architectural
  - Defects are identified and corrected
  - Risks are identified and managed
- **Independently assess** the system or subsystem architecture to determine compliance with architecturally significant product requirements
- **Validate** that the system or subsystem architecture meets the needs of its critical stakeholders
- **Formally review** the system or subsystem architecture by stakeholder representatives at one or more major project reviews
- **Independently evaluate the 'as performed' architecture engineering process** to determine compliance with the documented architecture engineering method (for example, as documented in the architecture plan, standards, procedures, and guidance)

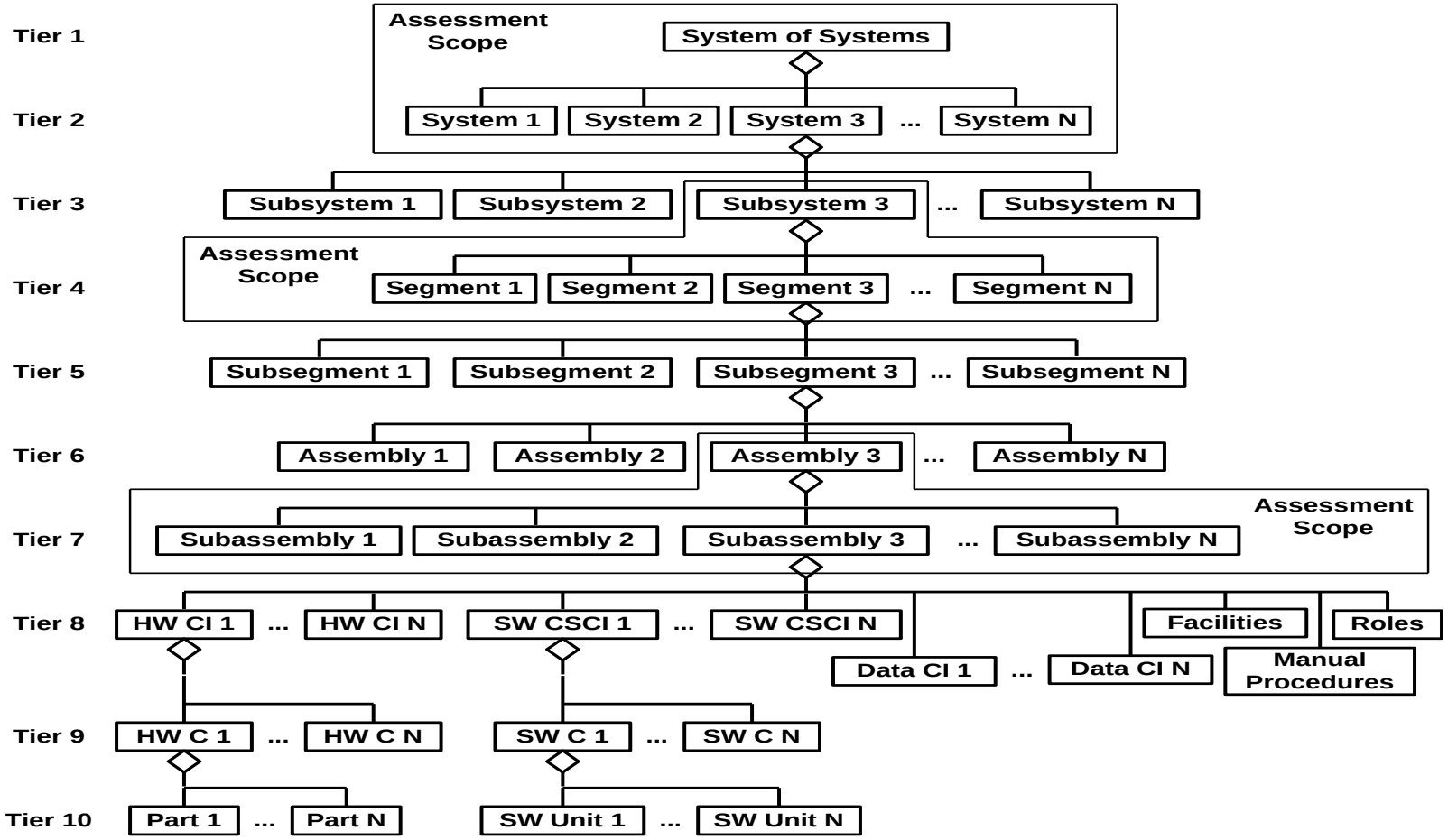
© 2009 Carnegie Mellon University

MFESA Task 9)
Evaluate and Accept the Architecture



© 2009 Carnegie Mellon University

MFESA Task 9)
Evaluate and Accept the Architecture



© 2009 Carnegie Mellon University

MFESA Task 9)
Evaluate and Accept the Architecture

Guidelines

- Use evaluations to support architectural milestones.
- Continuously evaluate the architecture and its representations.
- Internally evaluate models.
- Perform architecture analysis substeps.
- Collaborate with the stakeholders.
- Tailor software evaluation methods.
- Perform independent architecture assessments.
- Formally review the architecture.
- Verify architectural consistency.
- Perform cross-component consistency checking.
- Perform *both* static and dynamic checking.
- Set the evaluation scope based on risk and available resources.
- Formally manage architectural risks.

© 2009 Carnegie Mellon University

MFESA Task 9)
Evaluate and Accept the Architecture

Pitfalls

- Disagreement exists over the need to perform evaluations.
- Consensus does not exist on the evaluation's scope.
- It is difficult to schedule the evaluations.
- Management provides insufficient evaluation resources.
- There are too few evaluations.
- There are too many evaluations.
- How good is good enough?
- Evaluations are not sufficiently independent.
- The evaluators are inadequate.
- Evaluations only verify the easy concerns.
- The quality cases are poor.
- Stakeholders disagree on the evaluation results.
- The evaluations lack proper acceptance criteria.
- The evaluation results are ignored during acceptance.
- The acceptance package is incomplete.

© 2009 Carnegie Mellon University

MFESA Task 10)
Ensure Architectural Integrity

Task 1) Plan and Resource Architecture Engineering Effort

Task 2) Identify the Architectural Drivers

Task 3) Create Initial Architectural Models

Task 4) Identify Opportunities for Reuse of Architectural Elements

Task 5) Create Candidate Architectural Visions

Task 6) Analyze Reusable Components and their Sources

Task 7) Select or Create Most Suitable Architectural Vision

Task 8) Complete and Maintain the Architecture

Task 9) Evaluate and Accept the Architecture

Task 10) Ensure Architectural Integrity

© 2009 Carnegie Mellon University

MFESA Task 10)
Ensure Architectural Integrity

Goal:

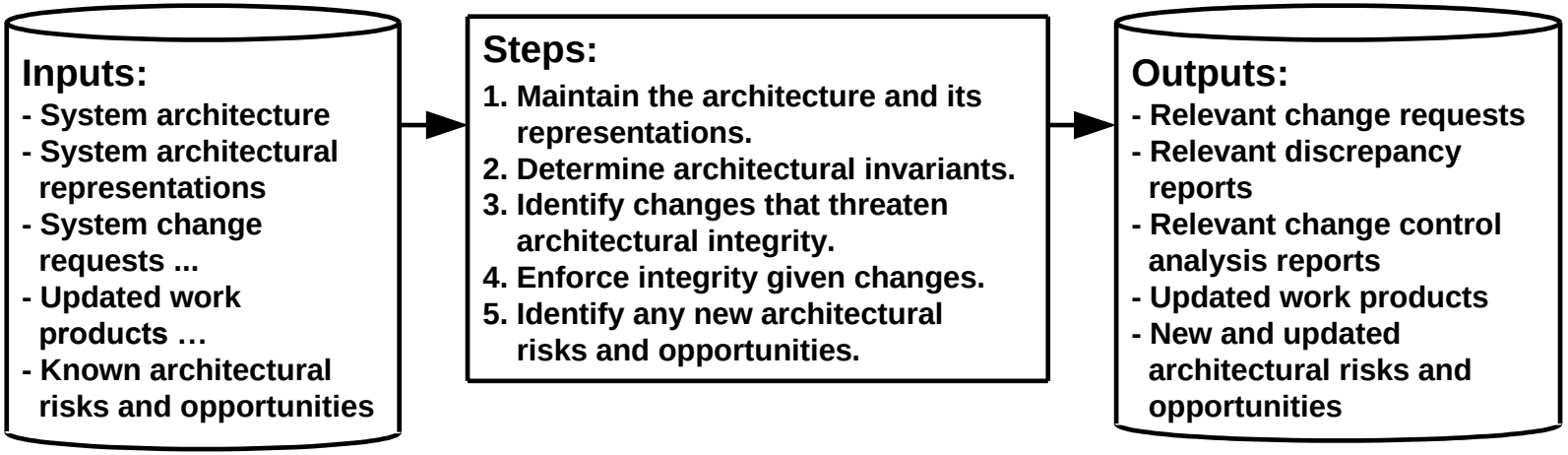
- Ensure the continued integrity and quality of the system architecture as the system evolves.

Objectives:

- Eliminate inconsistencies within the system architecture and its representations.
- Eliminate inconsistencies between the system architecture and its representations and:
  - Architecturally Significant Requirements
  - Enterprise Architecture(s)
  - Reference Architecture(s)
  - The Design of architectural components
  - The Implementation of architectural components
- The system architecture and its representations do not degrade over time.

© 2009 Carnegie Mellon University

MFESA Task 10)
Ensure Architectural Integrity



© 2009 Carnegie Mellon University

MFESA Task 10)
Ensure Architectural Integrity

Guidelines

- Maintain the architectural representations to maintain architectural integrity.
- Consider entire scope of ensure architectural integrity task.
- Consider the sources of architectural change.
- Protect the architectural invariants.
- Determine the scope of architectural integrity.
- Train the architects and designers.
- Formally manage architectural risks.

© 2009 Carnegie Mellon University

MFESA Task 10)
Ensure Architectural Integrity

Pitfalls

- The architectural representations become shelfware.
- Architecture engineering is done.
- The architecture is not under configuration management.

© 2009 Carnegie Mellon University

Topics

Motivation

MFESA Overview

MFESA Ontology of Concepts and Terminology

MFESA Metamodel of Reusable Method Components

MFESA Repository of Reusable Method Components

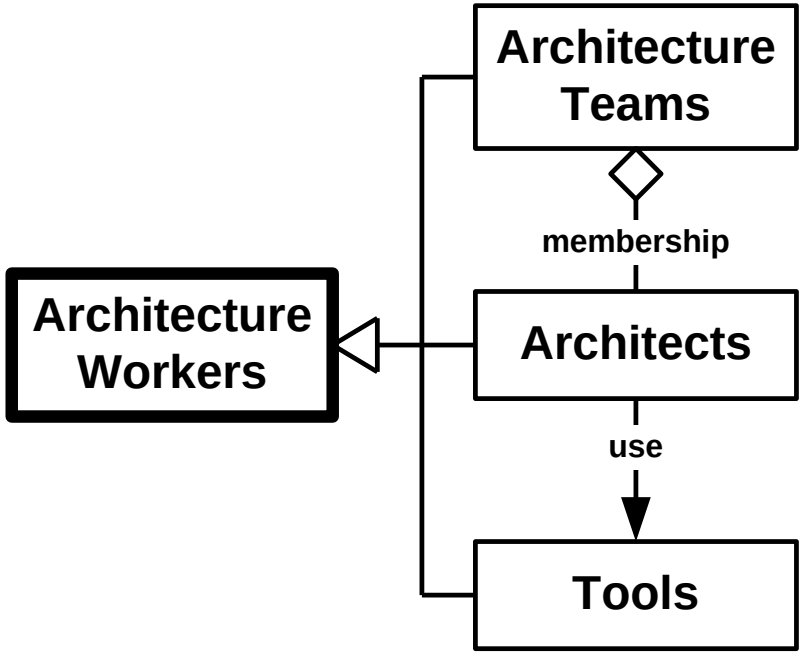
- **Architectural Work Units and Work Products**
- **Architectural Workers**

MFESA Metamethod

Conclusion

© 2009 Carnegie Mellon University

MFESA Repository – Architecture Workers



© 2009 Carnegie Mellon University

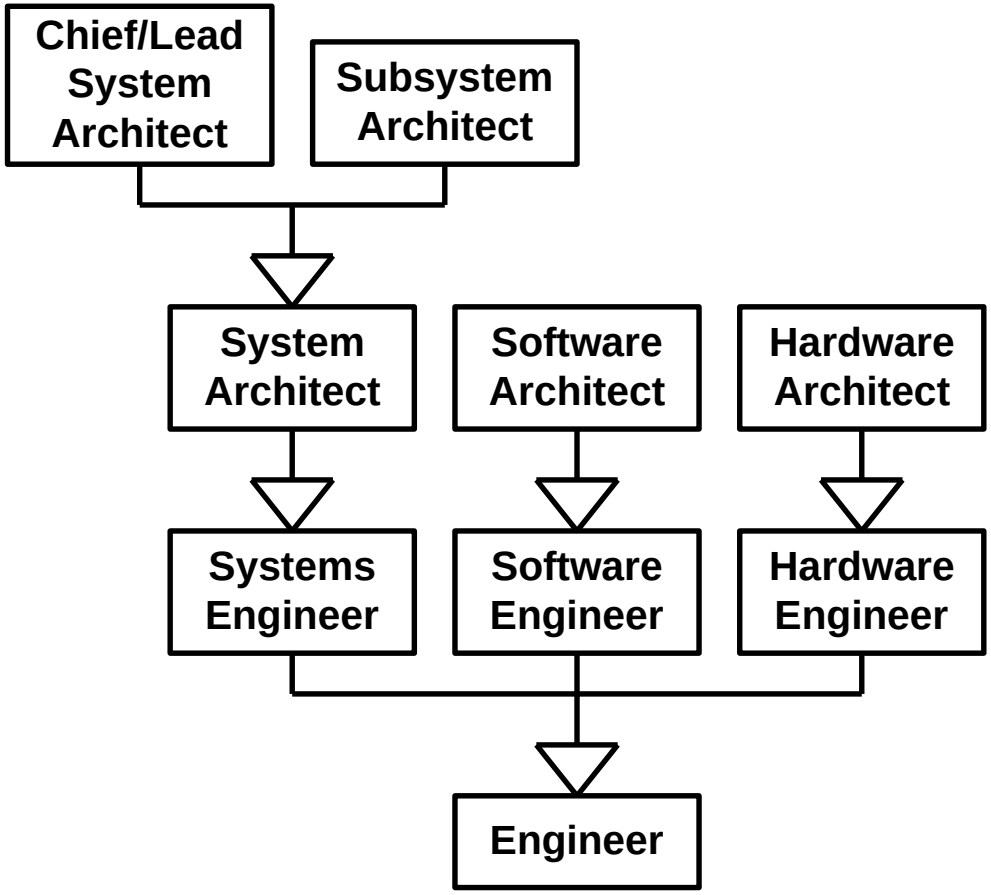
Architects - Definition

System Architect

the highly specialized role played by a systems engineer when performing system architecture engineering tasks to produce system architecture engineering work products

© 2009 Carnegie Mellon University

Types of Architects - Ontology



© 2009 Carnegie Mellon University

Architects – Primary Responsibilities

Determine and Assess Impact of the Architectural Drivers and Concerns

Develop Architecture and Architectural Representations

Analyze Architecture using Architectural Representations

Evaluate Architecture and Architectural Representations

Maintain Architecture and Architectural Representations

Ensure Architectural Integrity

© 2009 Carnegie Mellon University

Architects – Organizational Responsibilities

Lead architectural activities

Manage performance of architecture engineering tasks

Be an architecture advocate

Be a stakeholder advocate

Instantiate and tailor architecture engineering method

Select and acquire architecture engineering tools

Train architecture stakeholders

Evaluate architecture method and process

Interface and collaborate with architecture stakeholders

© 2009 Carnegie Mellon University

Architects – Authority

Determine architecture engineering method

Determine architectural work products to produce including models, documents, and architectural prototypes

Select and acquire architecture engineering tools

Determine architecture

Obtain and evalate Off-The-Shelf architectural components

© 2009 Carnegie Mellon University

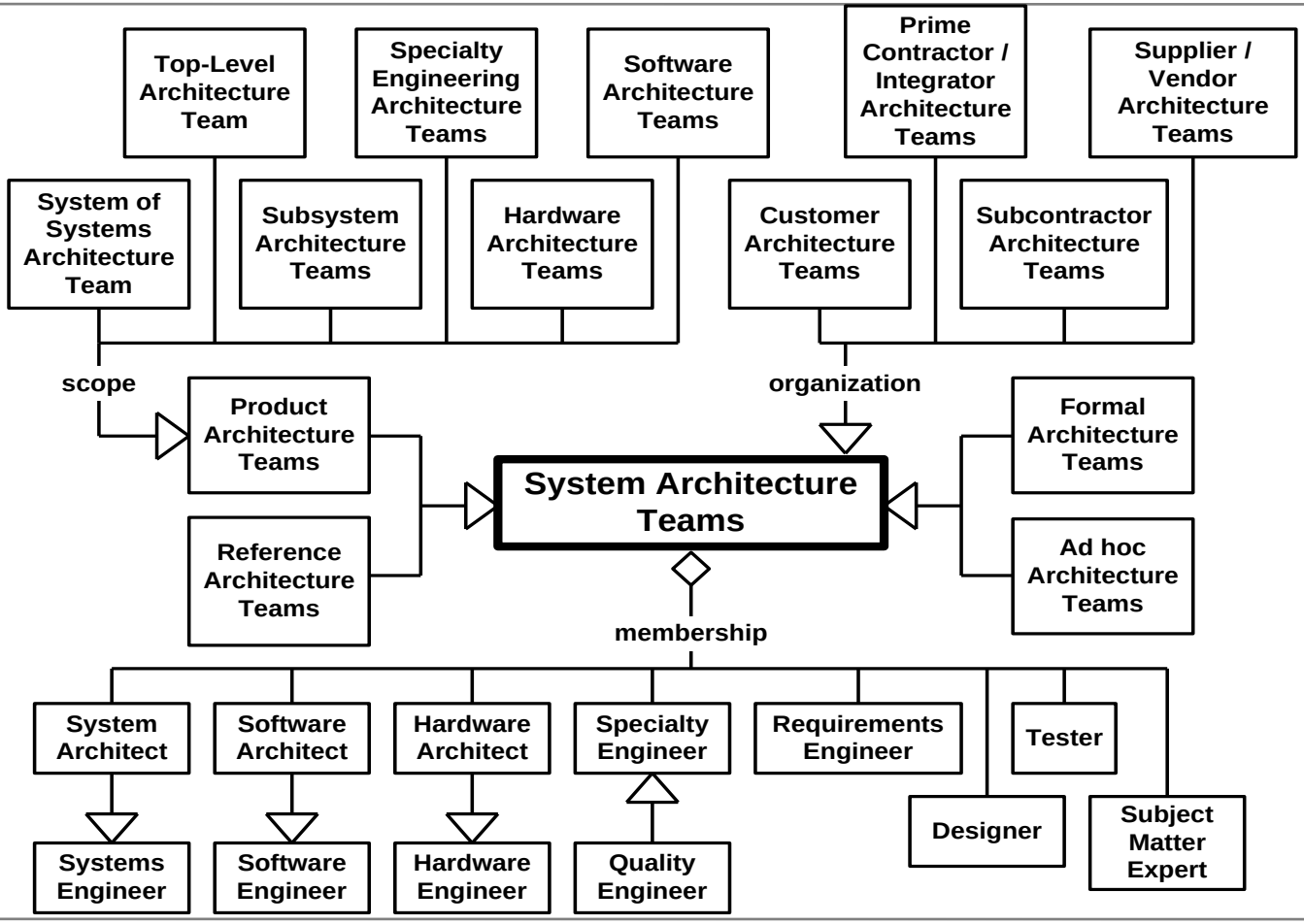
System Architecture Team - Definition

System Architecture Team

a team responsible for developing and maintaining all or part of a system's architecture

© 2009 Carnegie Mellon University

Types of Architecture Teams - Ontology



© 2009 Carnegie Mellon University

System Architecture Tools - Definition

System Architecture Tool

anything that assists with the production, coordination and maintenance of architectural work products

Many types:

- Whiteboard
- Image Capturing Device
- Word Processor
- Spreadsheet
- General-Purpose Drawing Tool
- Graphical Modeling Tool
- CAD/CAM (Computer Aided Design/Computer Aided Manufacturing)
- Simulation Tool
- Configuration Management Tool
- Requirements Engineering Tool
- Information Architecting Tool
- Business Process Modeling Tool
- Mass/Size/Geometry Modeling Tool
- Software Architecture Tool

© 2009 Carnegie Mellon University

Topics

Motivation

MFESA Overview

MFESA Ontology of Concepts and Terminology

MFESA Metamodel of Reusable Method Components

MFESA Repository of Reusable Method Components

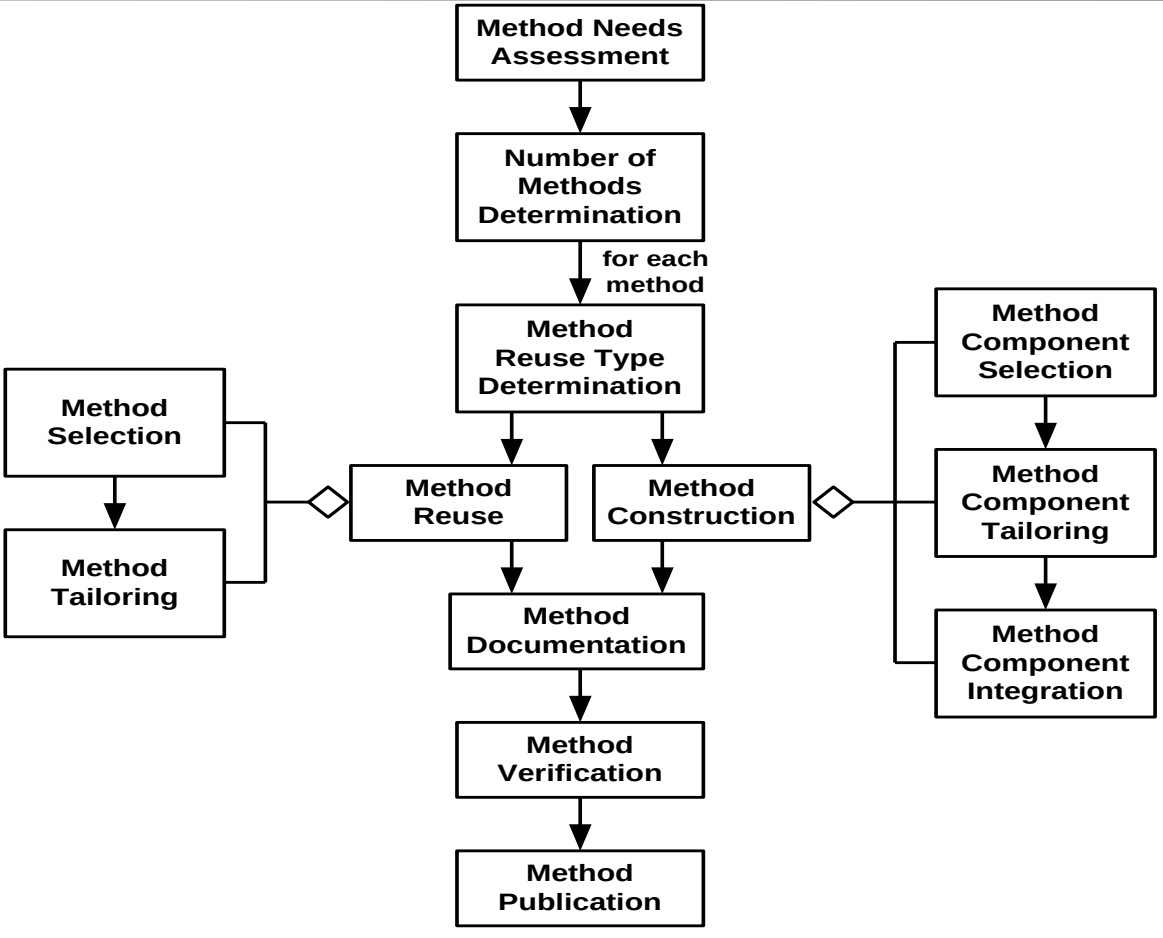
- **Architectural Work Units and Work Products**
- **Architectural Workers**

MFESA Metamethod

Conclusion

© 2009 Carnegie Mellon University

MFESA Metamethod - Tasks



© 2009 Carnegie Mellon University

Topics

Motivation

MFESA Overview

MFESA Ontology of Concepts and Terminology

MFESA Metamodel of Reusable Method Components

MFESA Repository of Reusable Method Components

- **Architectural Work Units and Work Products**
- **Architectural Workers**

MFESA Metamethod

Conclusion

© 2009 Carnegie Mellon University

Key Points to Remember

System architecture and system architecture engineering are critical to success.

MFESA is *not* a system architecture engineering method.

Architectural quality cases make the architects' case that their architecture sufficiently supports the architecturally significant requirements.

It is critical to capture the rationale for architectural decisions, inventions, and trade-offs.

Architects should keep their work at the right level of abstraction.

Reuse has a major impact on system architecture engineering.

Architecture engineering is never done.

© 2009 Carnegie Mellon University

Benefits of using MFESA

The benefits of:

- **Flexibility**: the resulting Architecture Engineering Method meets the unique needs of the stakeholders.
- **Standardization**: built from standard method components implementing best industry practices and based on common terminology and metamodel

Improved system architecture engineering (as-planned) methods and (as-performed) processes.

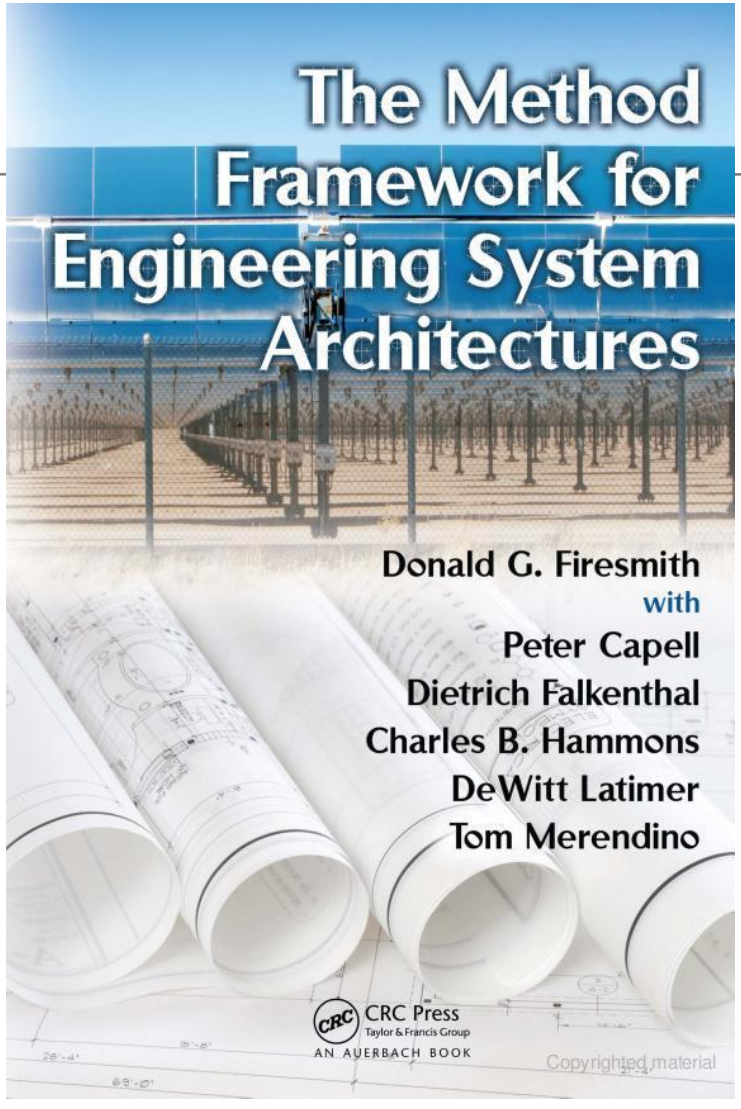
Improved architectures and architecture representations

© 2009 Carnegie Mellon University

Reference Book

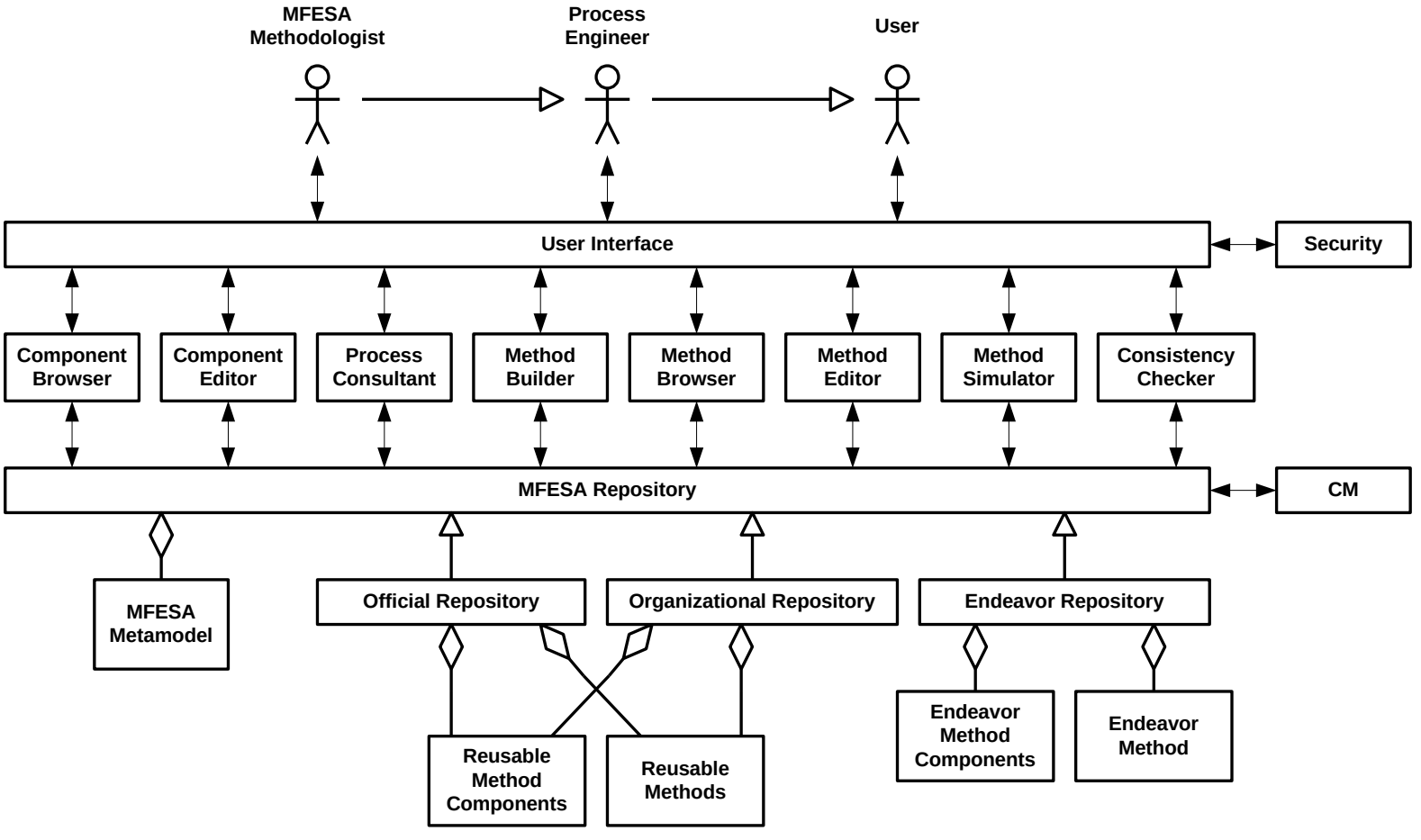
ISBN 1420085751

20 November 2008



© 2009 Carnegie Mellon University

Future Informational Website



© 2009 Carnegie Mellon University

Questions?

For more information, contact:

Donald Firesmith

Acquisition Support Program (ASP)

Software Engineering Institute (SEI)

dgf@sei.cmu.edu

© 2009 Carnegie Mellon University