# Collaborative Development of Systems Architecting Design Rules

**Georgia**Institute
of **Tech**nology

Tom McDermott

Dir. of Research and Dep. Dir., GTRI

tom.mcdermott@gtri.gatech.edu

(404) 407-8240

Tommer R. Ender, PhD

Senior Research Engineer

tommer.ender@gtri.gatech.edu

(404) 407-8639

Nicholas Bollweg

Research Engineer I

nicholas.bollweg@gtri.gatech.edu

(404) 407-7207

**Georgia Tech** | **Research Institute**

Problem. Solved.

# Motivation

- **Systems architect is challenged as both a facilitator and rule-setter**
  - Takes the role of evaluator and documentation lead
  - Provides insight into the design
  - Extrapolates key architectural constraints

- **Architecture documentation is reactive to the design process**
  - Formal architecture specification created after the fact
  - Little iteration with requirements/needs
  - Design process proceeds

# Motivation

- **Architect's experience counts more than use of formal tools**
  - MBSE tools enable integration of process and documentation for **high-level** design and requirements
  - Transition from architectural definition to actual design rules requires derivation of **lower-level** requirements

- **Design rules are needed that specify the architecture**
  - Want a narrative that utilizes heuristics derived from stakeholders and developers
  - Is it …**valuable**?...**effective**?...**useful**?

# Agile Application

◆ **Goal is to improve agility of the architecture definition process**

– People-based vice process based methods

– Built to change vice built to last (flexible)

◆ **Architecture provides a stable framework for incremental development**

– Architecture iterates around quality attributes

– Early planning involves rapidly evolving capabilities and convergence of stable architecture rules

– Requires convergence to a well-defined reference architecture developed in process with system capabilities

# Systems Engineering Methods

◆ Facilitate collaborative agreement and development of reference architecture

◆ Lifecycle focus to mature the architectural rule set
  – Architect leads **spiral development** iteration of architecture views
  – Enabled through tailorable SE tools

◆ Goal: mature the architecture documentation aligned with the design process

◆ In development: applied research is a work in progress
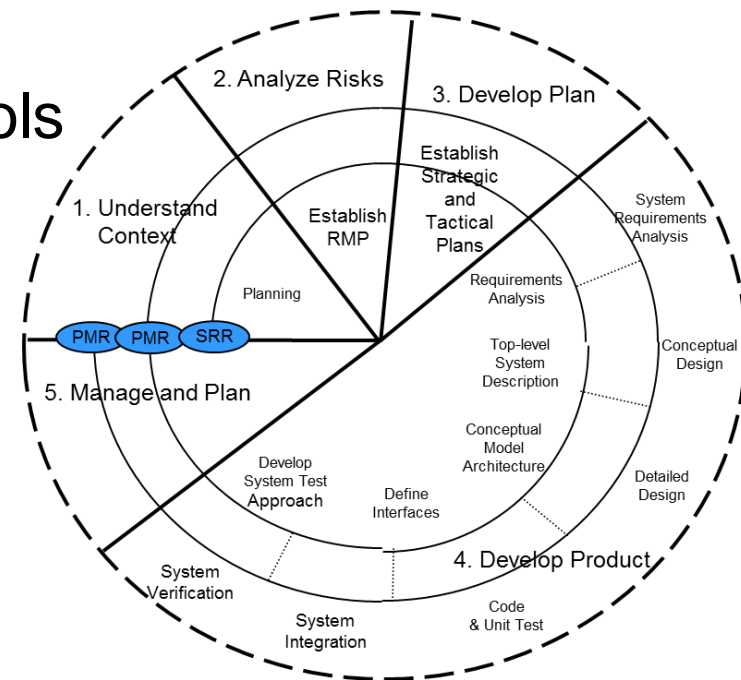
# Systems "Architecting" vs. "Engineering"

◆ Systems architecting differs from systems engineering in that it relies more on **heuristic** reasoning and less on use of analytics

◆ There are **qualitatively** different problem solving techniques required by high and low complexity levels

  – The lower levels would certainly benefit from purely analytical techniques, but those same techniques may be overwhelming at higher levels which may benefit more from heuristics derived from experience, or even abstraction

  – It is important to concentrate on only what is essential to solve the problem

> *The system should be modeled at as a high a level as possible, then the level of abstraction should be reduced progressively as needed*

# Insight

- The ability to structure a complex situation in a way that greatly increases understanding of it

- Guided by lessons learned from experience and observations

- Where systems architecting becomes more an art than a science
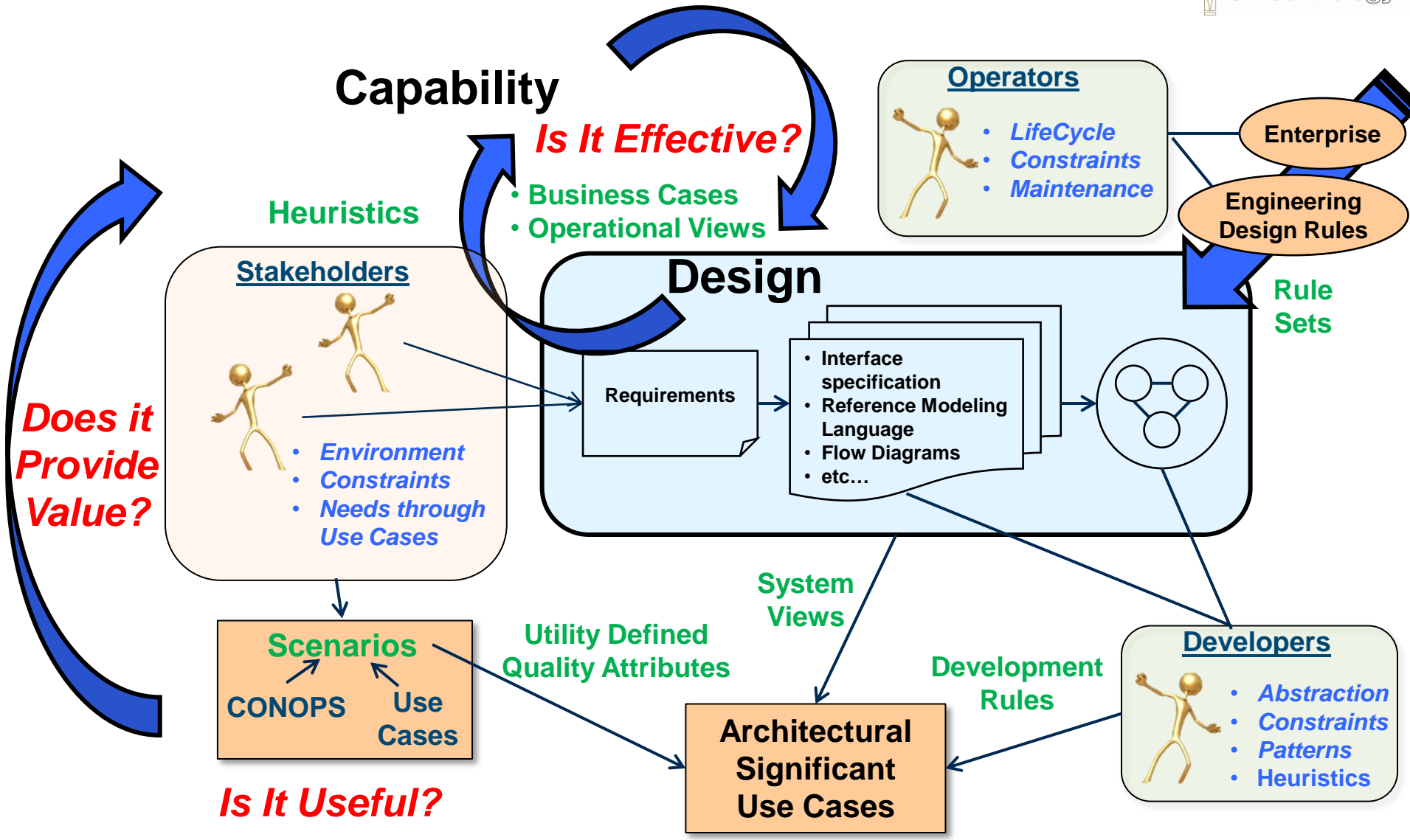
*Success comes from wisdom…*

*Wisdom comes from experience…*

*Experience comes from mistakes*

**Those mistakes and experience may come from one's predecessors**

***Insight = Heuristics***

# Perspective of the Systems Architect

**Capability**

*Is It Effective?*

**Operators**
- *LifeCycle*
- *Constraints*
- *Maintenance*

Enterprise

Engineering Design Rules

**Rule Sets**

**Heuristics**

- Business Cases
- Operational Views

**Design**

**Stakeholders**

- *Environment*
- *Constraints*
- *Needs through Use Cases*

Requirements

- Interface specification
- Reference Modeling Language
- Flow Diagrams
- etc…

*Does it Provide Value?*

**System Views**

**Developers**
- *Abstraction*
- *Constraints*
- *Patterns*
- Heuristics

**Scenarios**

CONOPS    Use Cases

**Utility Defined Quality Attributes**

**Development Rules**

**Architectural Significant Use Cases**

*Is It Useful?*

# Classical Architecting Methods
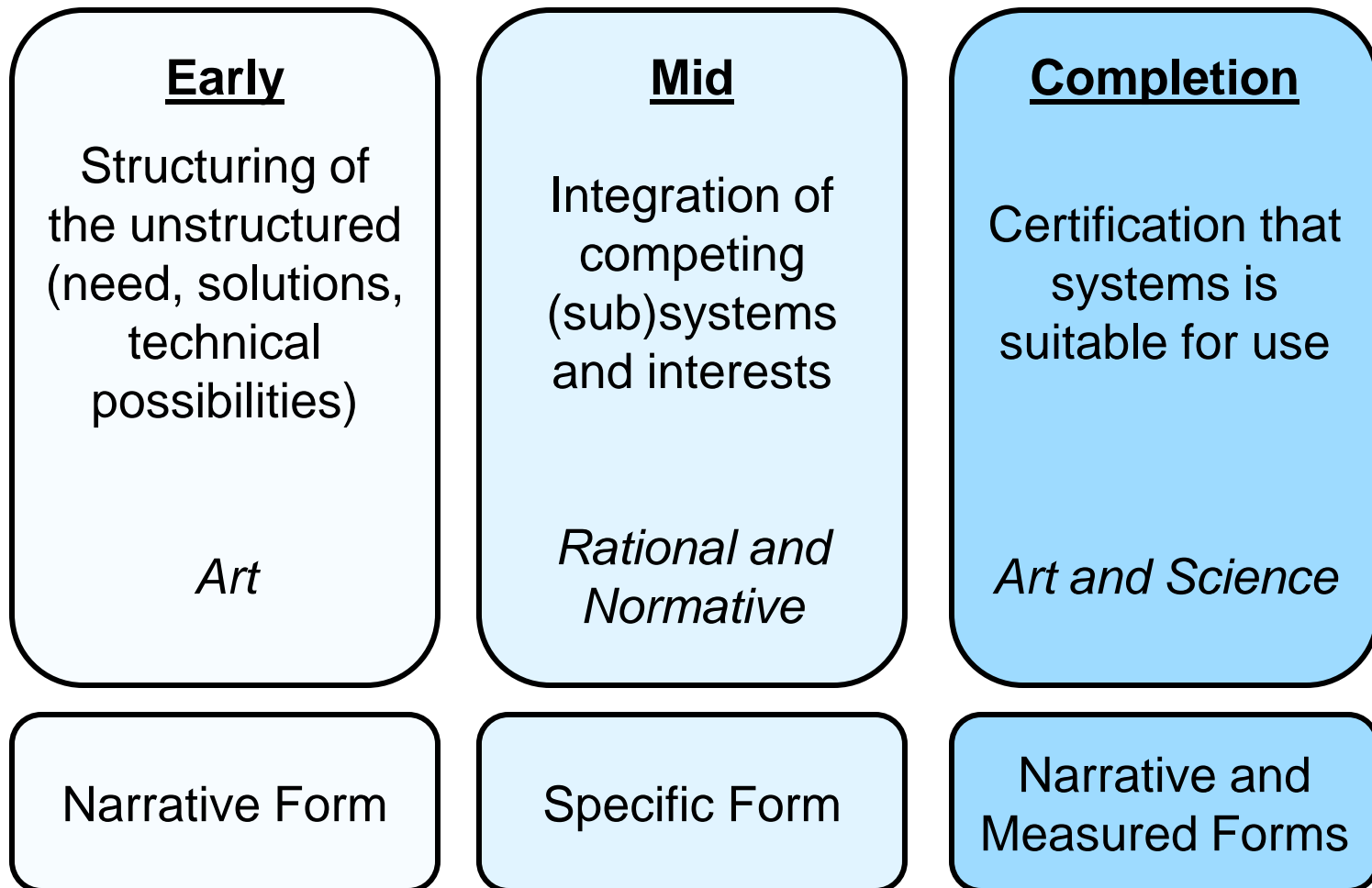
- **Science based**
  - Analytic, deductive, experiment based, easily certified, well understood, widely taught

- **"Art" or practice of architecting**
  - Nonanalytic, inductive, difficult to certify, less understood, seldom formally taught
  - Process of insights, vision, intuitions, judgment calls, subjective "taste"
  - Deals with immeasurables, sanity checks
  - Leads to "**unprecedented** systems"

# Phases of Architecting

Changes as project moves from phase to phase

| **Early** | **Mid** | **Completion** |
|---|---|---|
| Structuring of the unstructured (need, solutions, technical possibilities) | Integration of competing (sub)systems and interests | Certification that systems is suitable for use |
| *Art* | *Rational and Normative* | *Art and Science* |
| Narrative Form | Specific Form | Narrative and Measured Forms |

# Language of the Architect

## Changes as project moves from phase to phase

### Early

- Heuristics
- Stories
- Con-ops
- Scenarios

*Narrative, Visual*

### Mid

- Requirements
- Behavior
- Structure
- Function
- Rules

*Visual, Functional*

### Completion

- Performance
- Analysis
- Evaluation
- Utility

*Participative*

Narrative Form

Specific Form

Narrative and Measured Forms
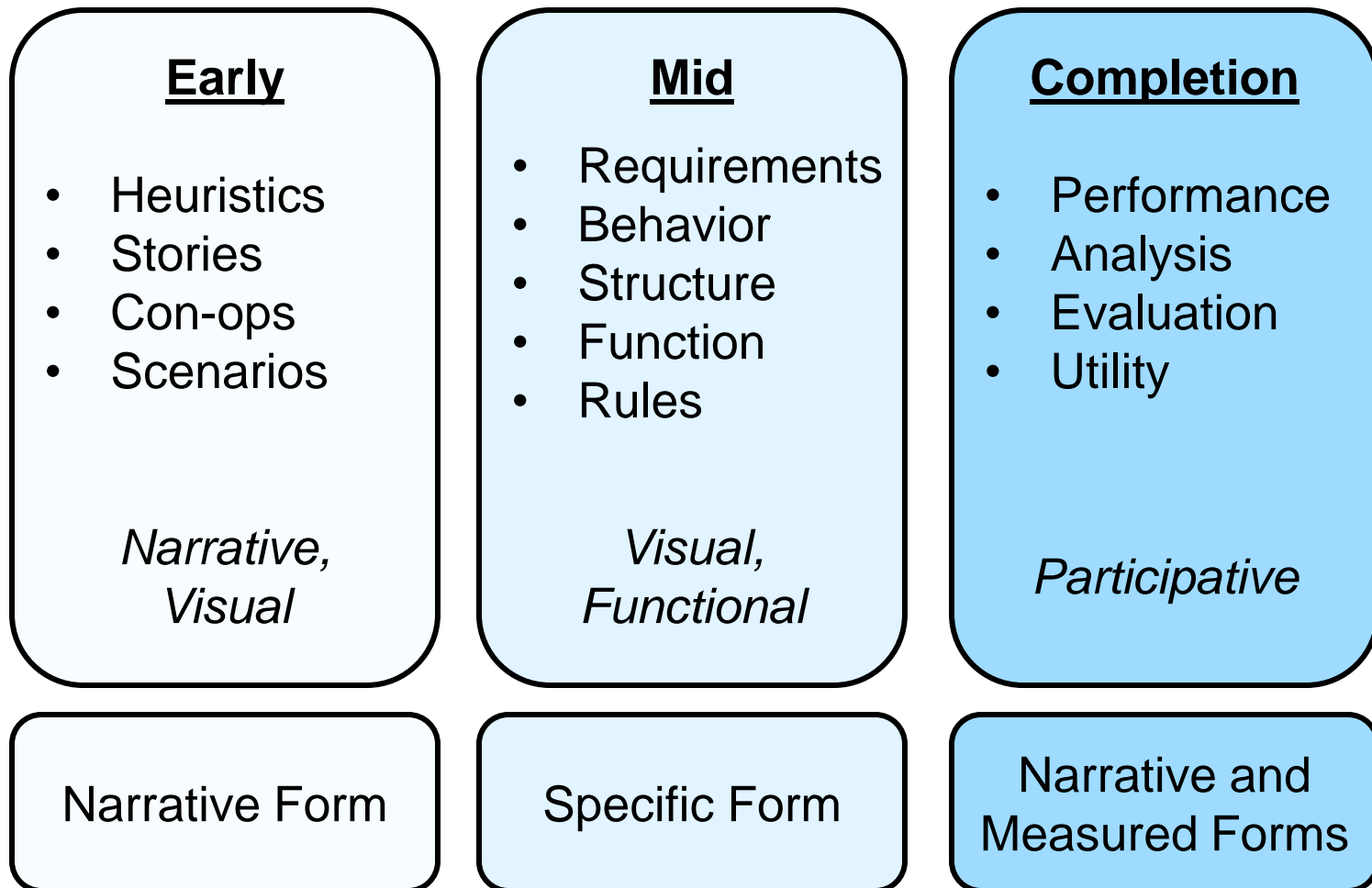
# Collaborative, Iterative, Narrative

- The System Architect uses interviews to collect concepts, use cases, and stakeholder perspective
- The System Architect facilitates brainstorming techniques to arrive at commonly accepted con-ops and use cases
  - Scenarios are collected and used to reach agreement
  - Architecturally significant scenarios are collected and saved for evauation
- The System Architect uses visual methods and stories to articulate the specific forms
- The System Architect uses evaluative techniques to determine architectural attributes of the design
- Model, model, model,…

# System/Architecture "Views"

**Purpose/Objective**: What the client wants

**Behavioral** (or functional): What the system does

**Managerial**: The process by which the system is constructed and managed
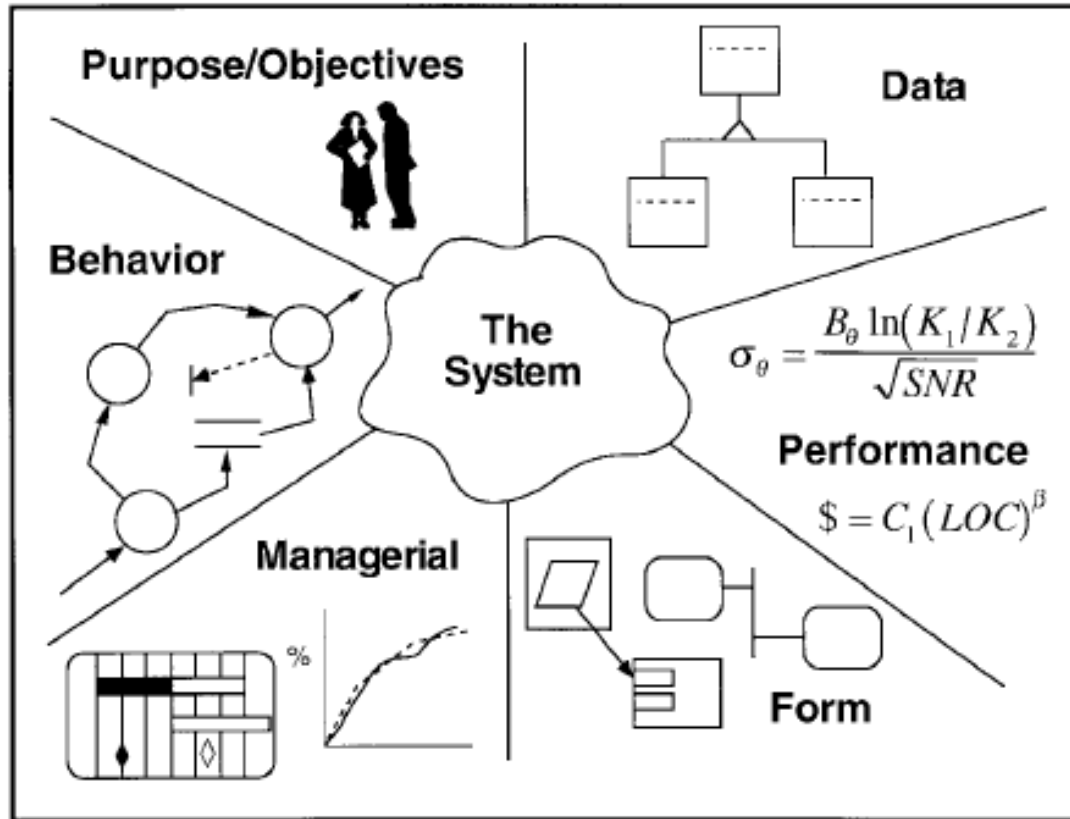


**Data**: The information retained in the system and its interrelationships
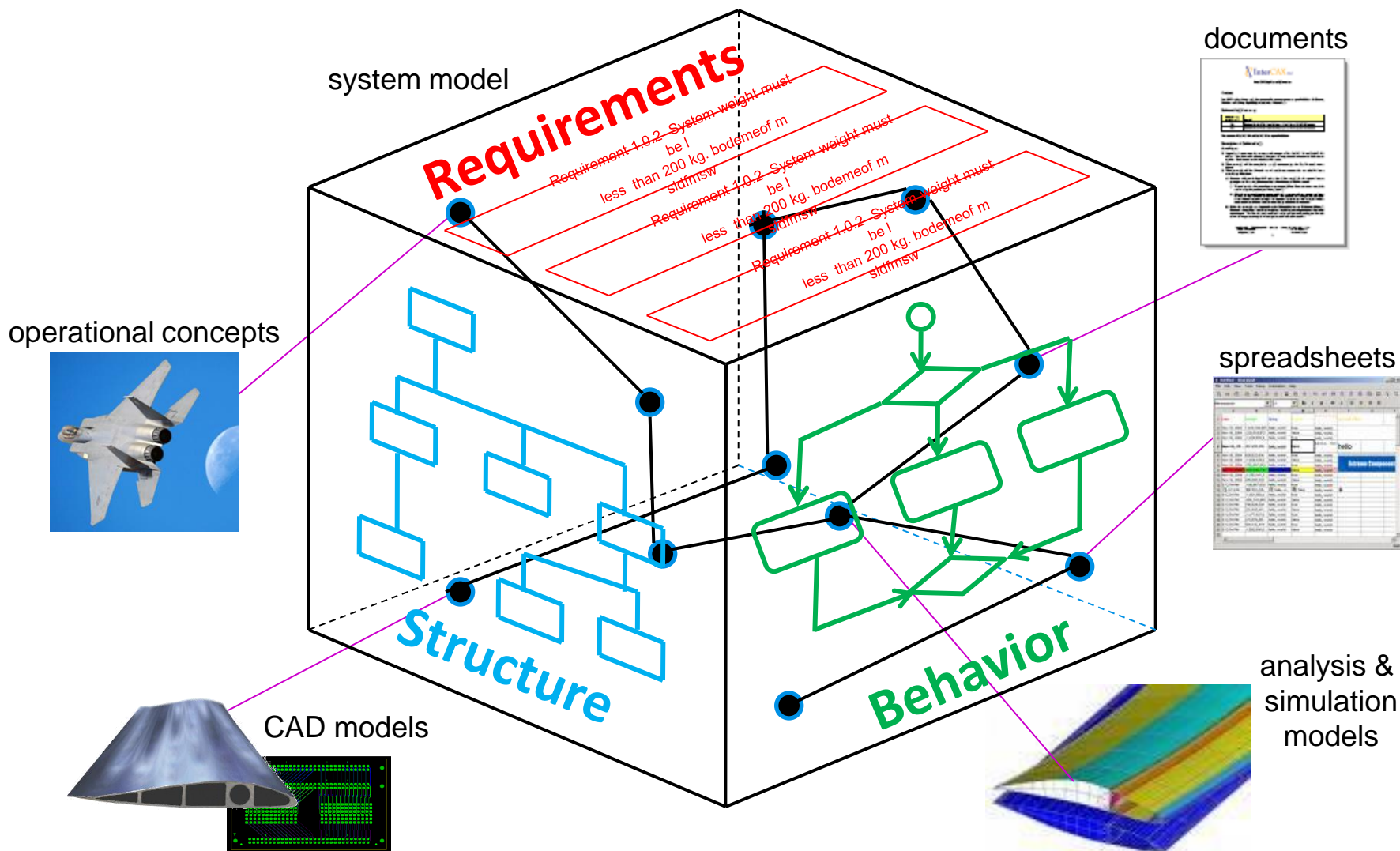
**Performance** (objectives or requirements): How effectively the system does it

**Form**: What the system is

- ◆ Each view represents an aspect of the actual system
- ◆ Each view may contain several <u>models</u> to capture information of the view

*Source: Maier (2009)*

# Representing System Models
## *With SysML: Unified, Connected, Consistent, Explicit*

documents

system model

**Requirements**

Requirement 1.0.2. System weight must be l less than 200 kg. bodemeof m stdfmsw

Requirement 1.0.2. System weight must be l less than 200 kg. bodemeof m stdfmsw

Requirement 1.0.2. System weight must be l less than 200 kg. bodemeof m stdfmsw

operational concepts

spreadsheets

**Structure**

**Behavior**

CAD models

analysis & simulation models

# Architectural Quality Attributes

➢ *How do I evaluate the quality of the architecture?*

**Separation of Concerns**

**Abstraction**

**Simplicity**

**Information Restriction**

– **Design drivers**
» Requirements, functions
» Hard performance measures

– **Development drivers**
» Development planning
» Coordination of work teams
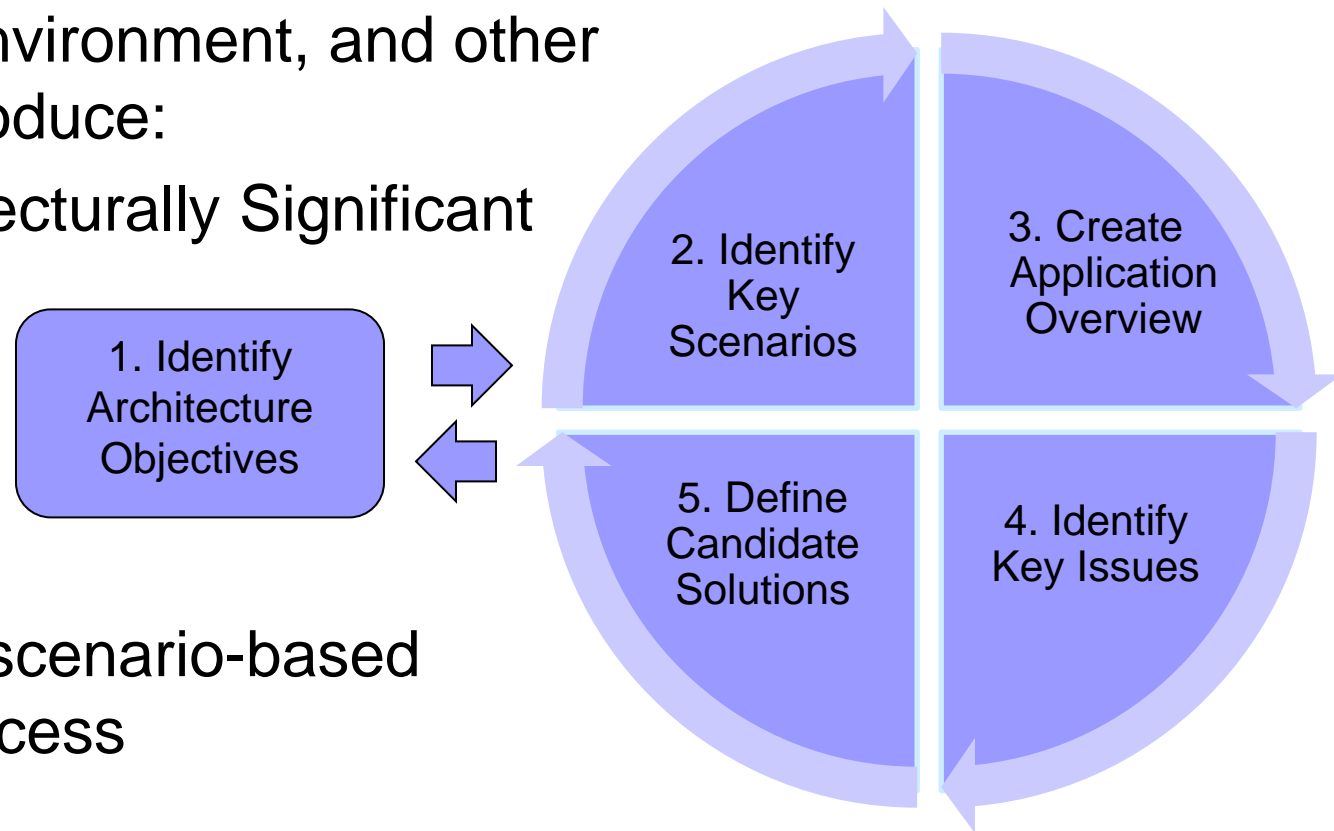
– **Business model drivers**
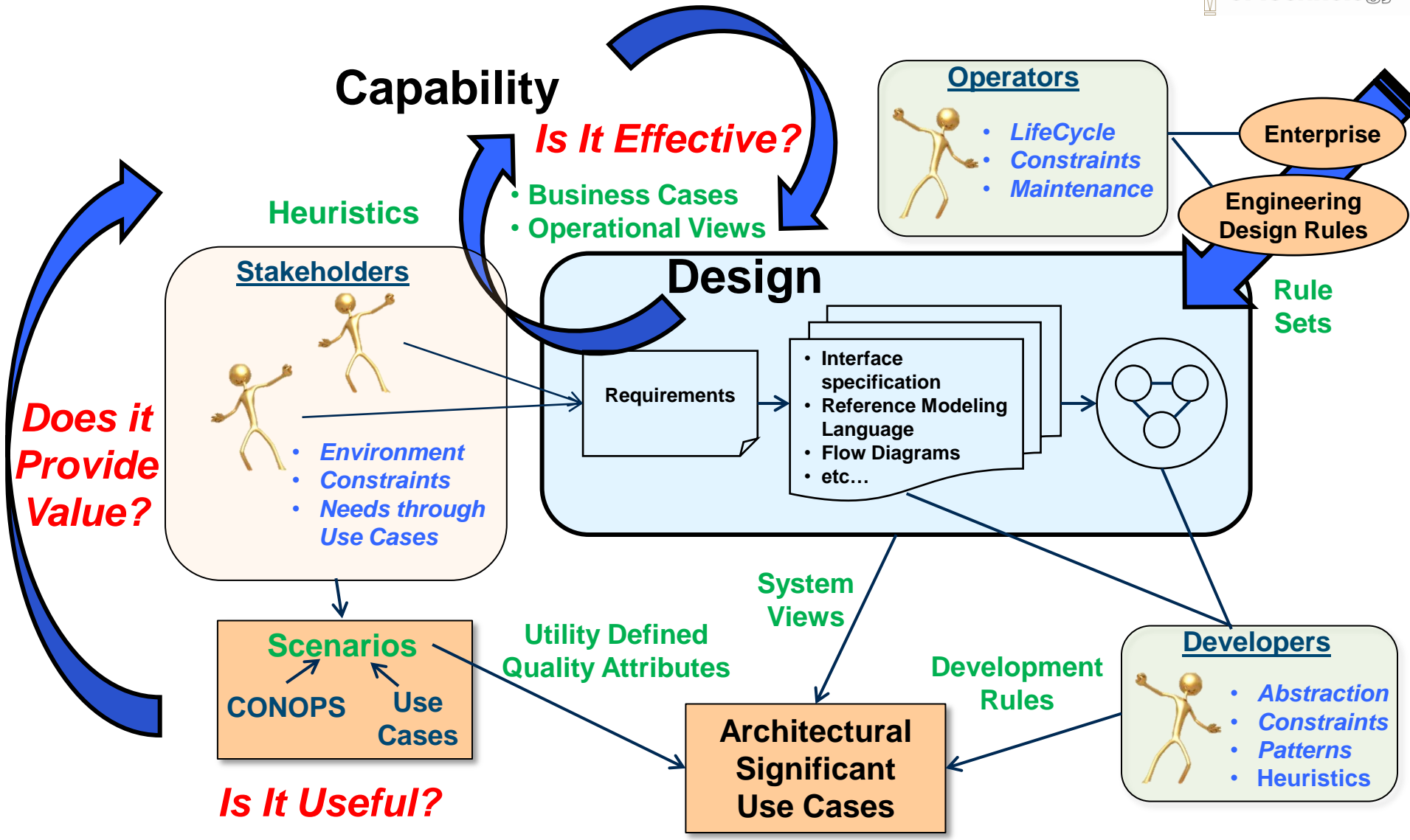» Develop or reuse
» Soft performance measures
» "ilities"

# Iteration for Architecture and Design

- Use cases and usage scenarios, functional requirements, non-functional requirements, technological requirements, the target deployment environment, and other constraints produce:

- A list of Architecturally Significant Use Cases



1. Identify Architecture Objectives

2. Identify Key Scenarios

3. Create Application Overview

4. Identify Key Issues

5. Define Candidate Solutions

- These feed a scenario-based evaluation process

Source: Microsoft Application Architecture Guide, 2nd Edition (Chapters 1-4)

# Perspective of the Systems Architect

**Capability**

*Is It Effective?*

**Operators**
- *LifeCycle*
- *Constraints*
- *Maintenance*

**Enterprise**

**Engineering Design Rules**

**Rule Sets**

• Business Cases
• Operational Views

**Heuristics**

**Stakeholders**
- *Environment*
- *Constraints*
- *Needs through Use Cases*

**Design**

**Requirements**

- Interface specification
- Reference Modeling Language
- Flow Diagrams
- etc…

*Does it Provide Value?*

**System Views**

**Developers**
- *Abstraction*
- *Constraints*
- *Patterns*
- Heuristics

**Scenarios**

CONOPS   Use Cases

*Is It Useful?*

**Utility Defined Quality Attributes**

**Development Rules**

**Architectural Significant Use Cases**

# The Role of the System Architect

- The System Architect is **more a leadership and management role** than a technical role

- Architects **need experience**, and a blend of management and leadership disciplines

- Communication and vision require leadership capacity
  - The architect holds the architectural vision, often their own
  - The architect makes high-level design decisions around interfaces, functional partitioning, and interactions
  - The architect must communicate these effectively, often visually

- The architect's **primary tasks are rule-setting**
  - The architect must direct technical standards, including design standards, tools, or platforms,
  - These should be based on business goals rather than to place arbitrary restrictions on the choices of developers.

# Leadership Competencies

- **Experience and judgment**
  - The architect must balance the customer's view of the system with their organization's business view of the system

- **Communications**
  - The architecture is presented in visuals to all stakeholders
  - The architecture is derived to written guidelines and design rules for the team

- **Leadership and Systems Thinking**
  - The architecture is the high level vision of the system
  - The architecture is defined more by heuristics than requirements
  - The architecture definition contains a number of soft requirements that have to be evaluated in collaborative groups

- **Management**
  - The architect ensures the design team follows design standards

# Conclusions

- Architecture development should not be reactive to the design process

- Role of the architect is to facilitate the design process

- Design rules may be derived using flexible, iterative architecture development

- Tailorable systems engineering tools will enable this

- Research required to better capture narrative forms (heuristics, conops, use cases) into specific forms (requirements, structure, function)