# Let's Teach Architecting High Quality Software

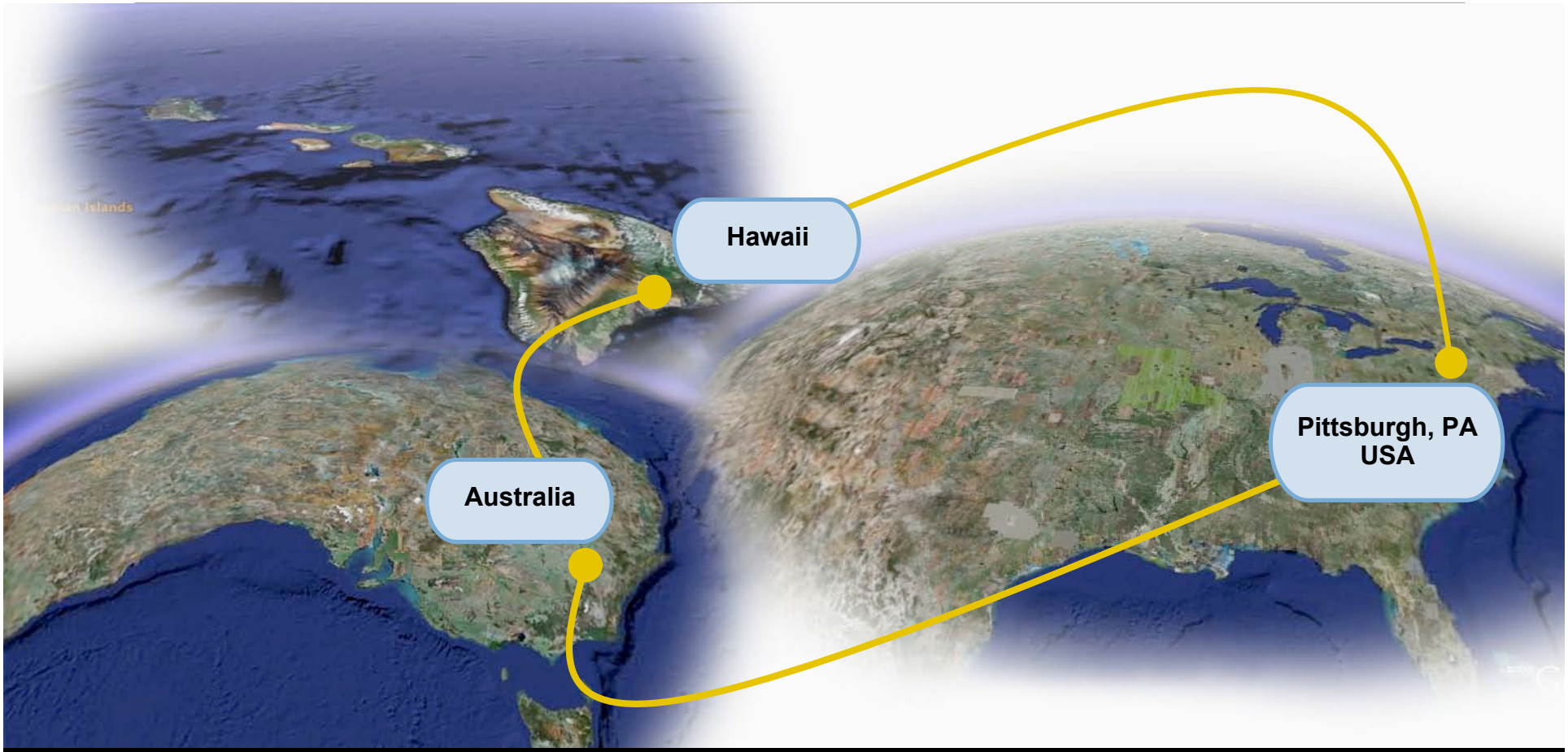
Linda Northrop

CSEET 2006



Software Engineering Institute | Carnegie Mellon

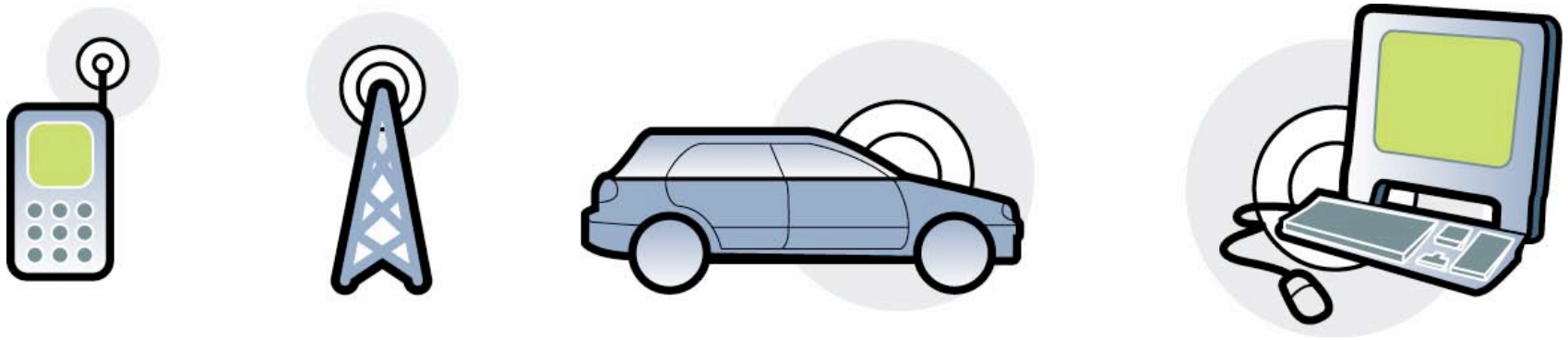
# MY GEOGRAPHY LESSON



# Let's Teach Architecting High Quality Software

# Software

# SOFTWARE PERVADES OUR WORLD



Software is an integral part of our everyday lives.

Software has become the bottom line for many organizations who never envisioned themselves in the software business.

# WHERE BEHAVIOR COUNTS MOST

*much is required of software.*

# Quality

# QUALITY

Quality software is software that is fit for its intended purpose.

# High Quality

# HIGH QUALITY

High quality software meets business goals and user needs.

It has the right features and the right attributes.

# UNIVERSAL BUSINESS GOALS

INCREASED MARKET SHARE

QUICK (OR RIGHT) TIME TO MARKET

EFFECTIVE USE OF LIMITED RESOURCES

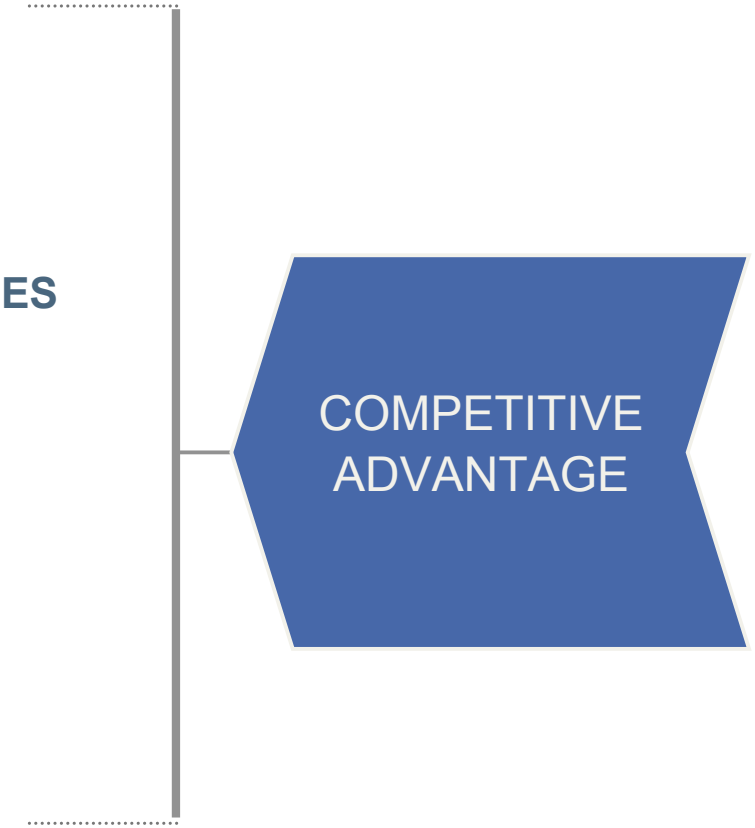
PRODUCT ALIGNMENT

LOW-COST PRODUCTION

LOW-COST MAINTENANCE

MARKET AGILITY

MIND SHARE



# THE ULTIMATE UNIVERSAL GOAL

# USER NEEDS

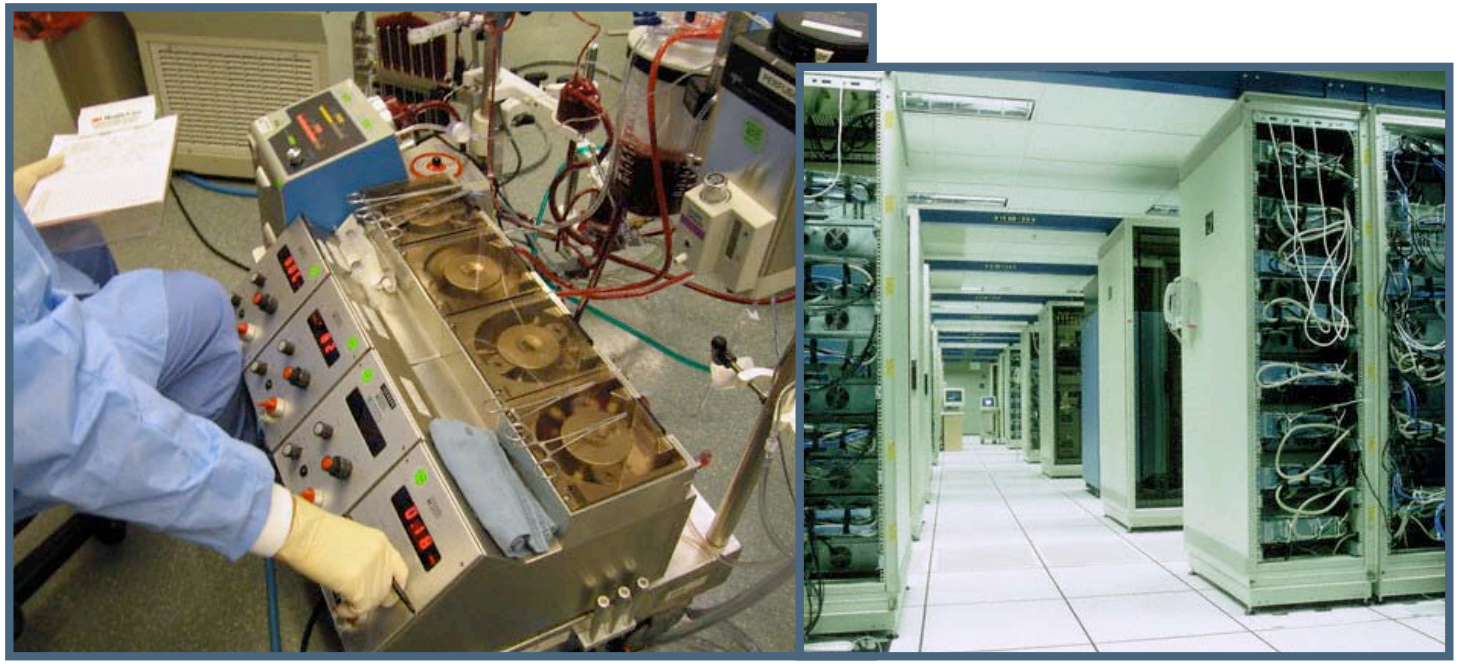
- Required capability
- Low learning threshold
- Ease of use
- Predictable behavior
- Dependable service
- Timely response
- Timely throughput
- Protection from unintended intruders and viruses
- ……

***Software product goals should address user needs.***

# THE RIGHT FEATURES

## SOFTWARE NEEDS TO HAVE THE RIGHT FUNCTIONALITY:

The software does what I want it to do and not (a lot) more.



# THE RIGHT QUALITY ATTRIBUTES

Quality attributes include

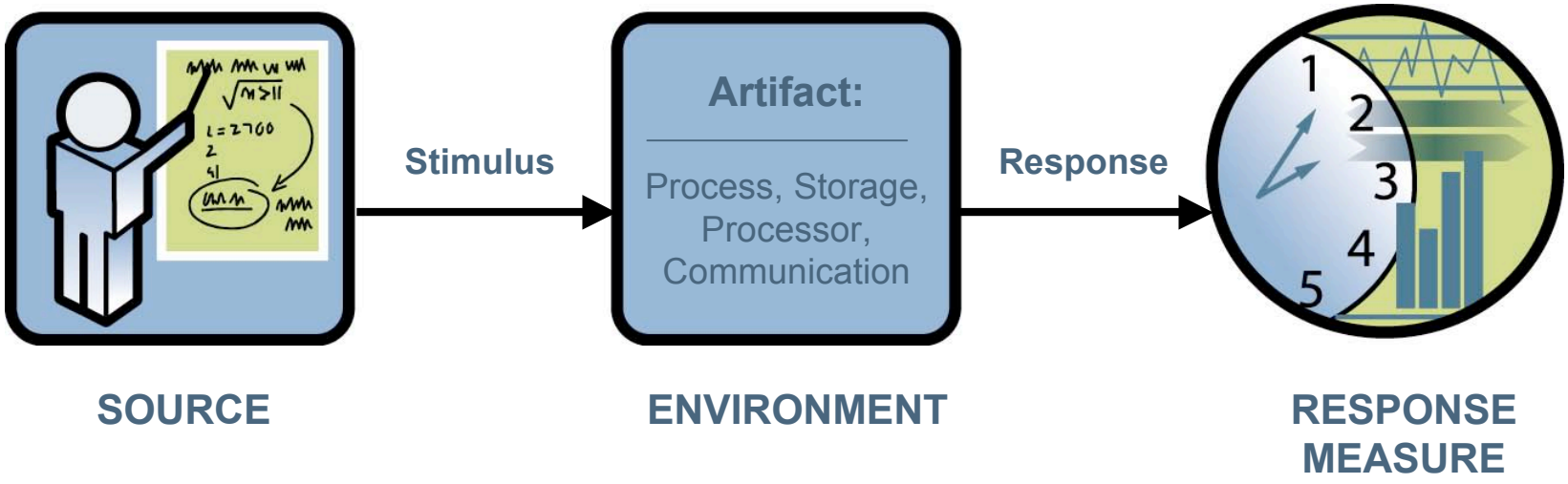
- Performance
- Availability
- Usability
- Modifiability
- Security
- Etc.

Quality attribute requirements stem from business and product goals.

Key quality attributes need to be characterized in a system-specific way.

Scenarios are a powerful way to characterize quality attributes and represent stakeholder views.

# PARTS OF A QUALITY ATTRIBUTE SCENARIO



# SOFTWARE SYSTEM DEVELOPMENT

Functional Software Requirements

If function were all that mattered, any monolithic software would do, ..***but other things matter***…

***The important quality attributes and their characterizations are key.***

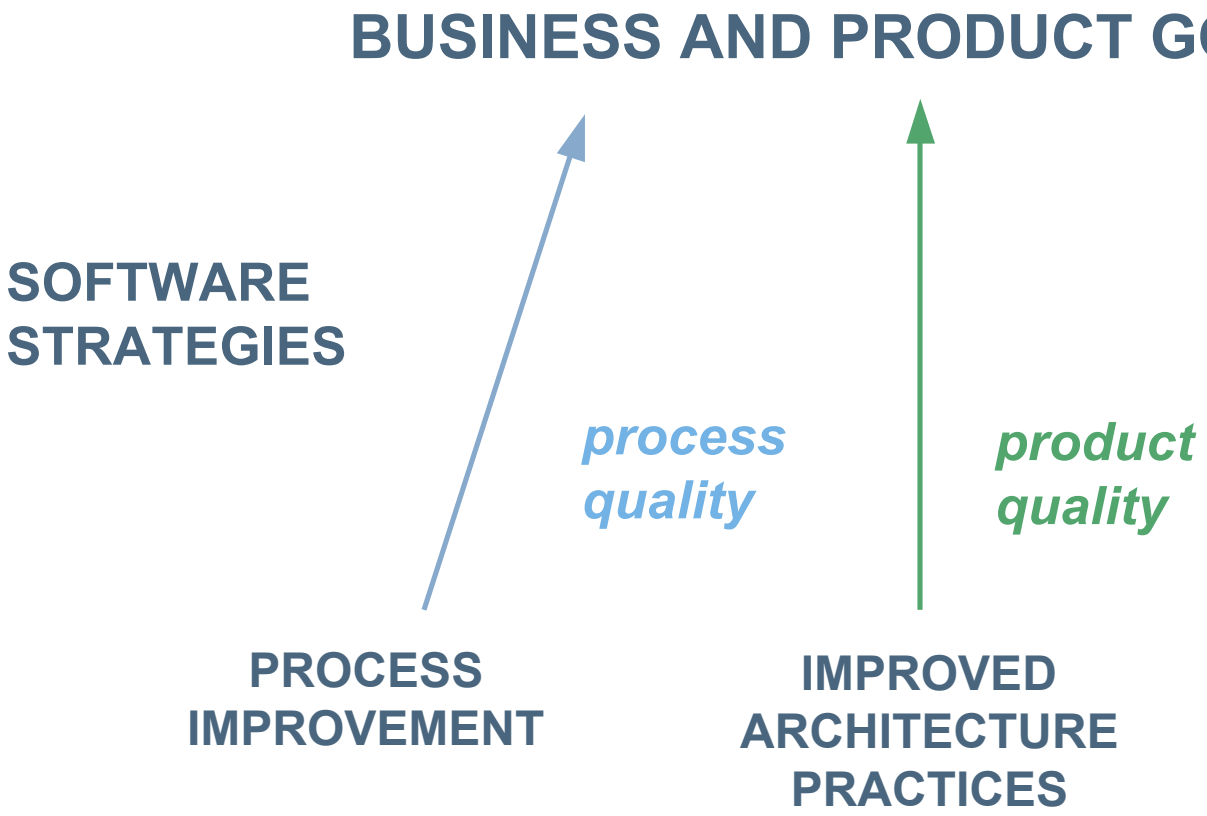
- Modifiability
- Interoperability
- Availability
- Security
- Predictability
- Portability

*analysis, design, development*

Quality Attribute Drivers → Software Architecture → Software

***has these qualities***

# SOFTWARE STRATEGIES ARE NEEDED



Let's Teach Architecting High Quality Software

# Architecture

# WHAT WE NEED IN SOFTWARE

Well-designed software architecture that
- lays out the basic elements of construction
- is known to satisfy important quality goals

Well-defined parts – components that
- have specified roles and interfaces
- have known properties
- behave predictably in a given assembly

Well-defined production plan that prescribes
- the order and method of assembly
- individual and team goals

# WHAT WE NEED IN SOFTWARE

**Well-designed software architecture that**
- **lays out the basic elements of construction**
- **is known to satisfy important quality goals**

Well-defined parts – components that
- have specified roles and interfaces
- have known properties
- behave predictably in a given assembly

Well-defined production plan that prescribes
- the order and method of assembly
- individual and team goals

# FOCUS: SOFTWARE ARCHITECTURE

From our experience

The quality and longevity of a software-intensive system is largely determined by its architecture.

Many large system and software failures point to

- inadequate software architecture education and practices
- the lack of any real software architecture evaluation early in the life cycle

Risk mitigation early in the life cycle is key.

- Mid-course correction is possible before great investment.
- Risks don't become problems that have to be addressed during integration and test.

# WITHOUT SOFTWARE ARCHITECTURE FOCUS

Poorly designed software architectures result in

- greatly inflated integration and test costs
- inability to sustain systems in a timely and affordable way
- lack of system robustness
- in the worst case, product or project cancellation
- in all cases, failure to best support the user

# WHY IS SOFTWARE ARCHITECTURE IMPORTANT?

Represents ***earliest*** design decisions

- hardest to change
- most critical to get right
- communication vehicle among stakeholders

***First*** design artifact addressing

- performance
- modifiability
- reliability
- security

Key to systematic ***reuse***

- transferable, reusable abstraction

The **right architecture** paves the way for system **success.**
The **wrong architecture** usually spells some form of **disaster.**

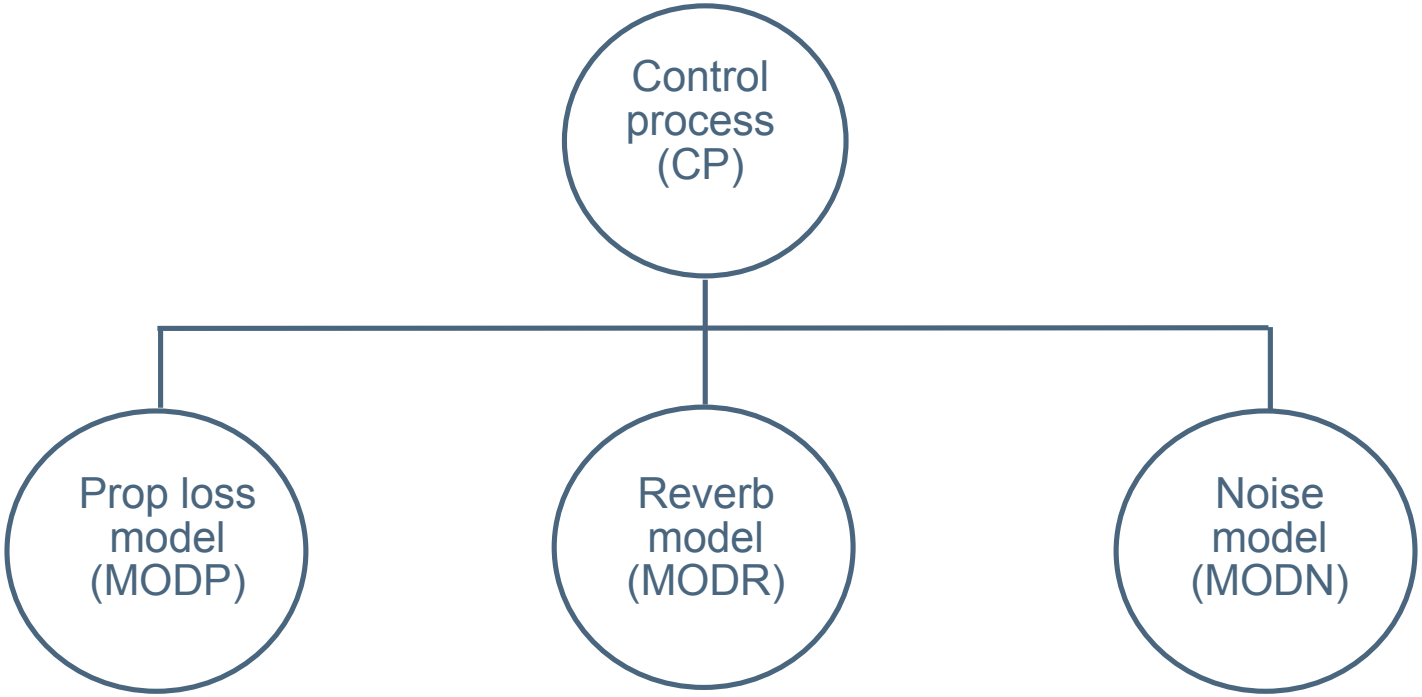
# WHAT IS A SOFTWARE ARCHITECTURE?

Informally, software architecture is the blueprint describing system composition.

A software architecture is often depicted using an ad hoc box-and-line drawing of the system that is intended to solve the problems articulated by the specification.

- Boxes show elements or “parts” of the system.
- Lines show relationships among the parts.

# A SOFTWARE ARCHITECTURE (?)



# DEFINITION: SOFTWARE ARCHITECTURE

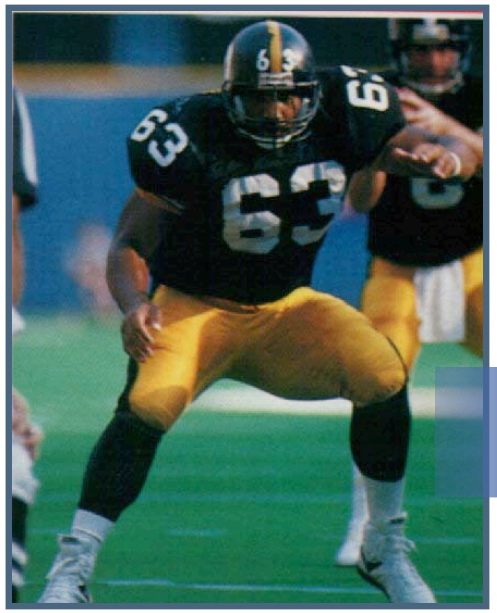
"The **software architecture** of a program or computing system is the **structure or structures** of the system, which comprise the **software elements**, the **externally visible properties** of those elements, and the **relationships among** them."[1]

[1] Bass, L.; Clements; P. & Kazman, R. *Software Architecture in Practice, Second Edition.* Boston, MA: Addison-Wesley, 2003.

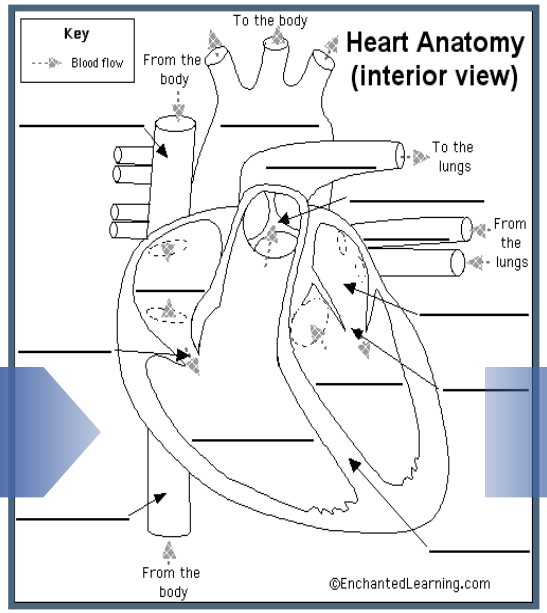
# IMPLICATIONS OF OUR DEFINITION

- Software architecture is an abstraction of a system.
- Software architecture defines the properties of elements.
- Systems can and do have many structures.
- Every software-intensive system *has* an architecture.
- Just having an architecture is different from having an architecture that is known to everyone.
- If you don't develop an architecture, you will get one anyway – **and you might not like what you get!**

# STRUCTURES AND VIEWS – 1



A human body comprises multiple *structures.*

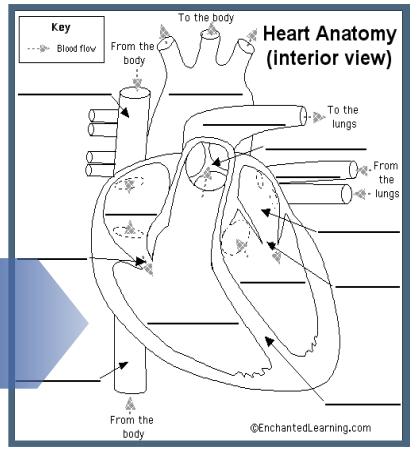



a *static view* of one human *structure*

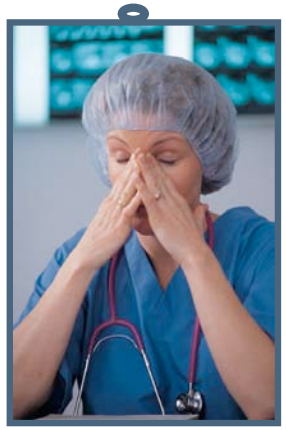
a *dynamic view* of that *structure*

*One body has many structures, and those structures have many views.*
*So it is with software…*

# STRUCTURES AND VIEWS – 2




These views are needed by the cardiologist…

…but will they work for the orthopedist?

*Different stakeholders are interested in different structures.*
*Views must represent the structures in which the stakeholders are interested.*
*So it is with software…*

# STRUCTURES AND VIEWS – 3

## You should know

- **structure** – an actual set of architectural elements as they exist in software or hardware
- **view** – a representation of a coherent set of architectural elements, as written by and read by system stakeholders.
  - A view consists of a representation of a set of elements and the relations among them.

## You should provide

- views that help evaluators and stakeholders understand the software architecture.

# THIS IS WHAT HAPPENS


*without careful architectural design.*

***And so it is with software.***

# OTHER ARCHITECTURES - 1

**Enterprise architectures** are a means for describing business structures and processes that connect business structures.[1]

- focus on business processes, dataflow, systems (including software packages), and their interconnection
- do not address the details of software design
- DoDAF, FEAF, and TEAF are generally regarded as enterprise architectures.

[1] Zachman, John A., *"A Framework for Information Systems Architecture." IBM Systems Journal*, 26, 3 (1987): 276-292.

# OTHER ARCHITECTURES - 2

A **system architecture** is a means for describing the elements and interactions of a complete system including its hardware elements and its software elements.

System architecture is concerned with the elements of the system and their contribution toward the goal of the system, but not with their substructure.

***System Architecture:*** "*The fundamental and unifying system structure defined in terms of system elements, interfaces, processes, constraints, and behaviors.*"[1]

*Systems Engineering* is a design and management discipline useful in designing and building large, complex, and interdisciplinary systems.[2]

[1] Rechtin, E. *Systems Architecting: Creating and Building Complex Systems*. Englewood Cliffs, NJ : Prentice-Hall, 1991.

[2] International Council On Systems Engineering (INCOSE), Systems Architecture Working Group, 1996.

# WHERE DOES SOFTWARE ARCHITECTURE FIT?

Enterprise architecture and system architecture provide an environment in which software lives.

- Both provide requirements and constraints to which software architecture must adhere.
- Elements of both are likely to contain software architecture.
- **Neither are a substitute for or obviate a software architecture.**

In large, complex, software-intensive systems **both software and system architectures are critical** for ensuring that the system is fit for the intended purpose.

# Let's Teach Architecting High Quality Software

## Architecting

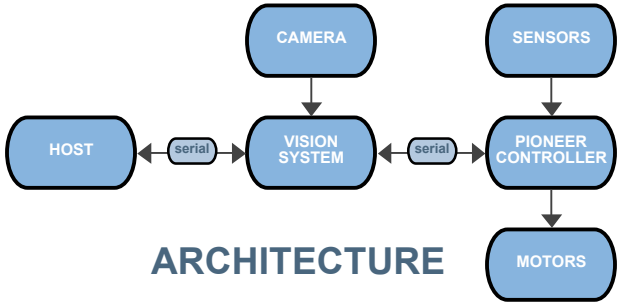
# REQUIREMENTS BEGET DESIGN

REQUIREMENTS IN VARIOUS FORMS

AVAILABLE KNOWLEDGE

SYSTEM

DESIGNER




ARCHITECTURE

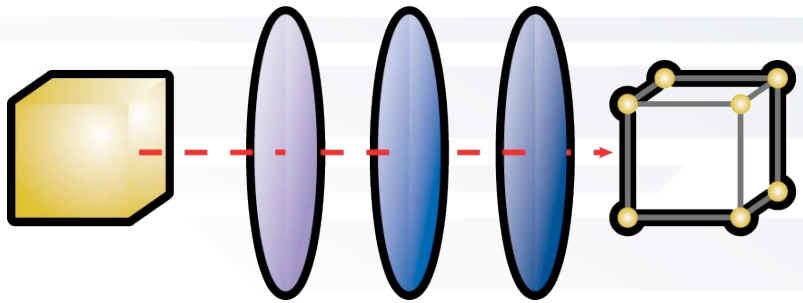
# FACTORS INFLUENCING ARCHITECTURES

Where do architectures come from?

System requirements, constraints, business and product goals certainly, but that's not all.

Architectures are influenced by

- stakeholders of a system
- technical and organizational factors
- architect's background



# INFLUENCE OF SYSTEM STAKEHOLDERS – 1

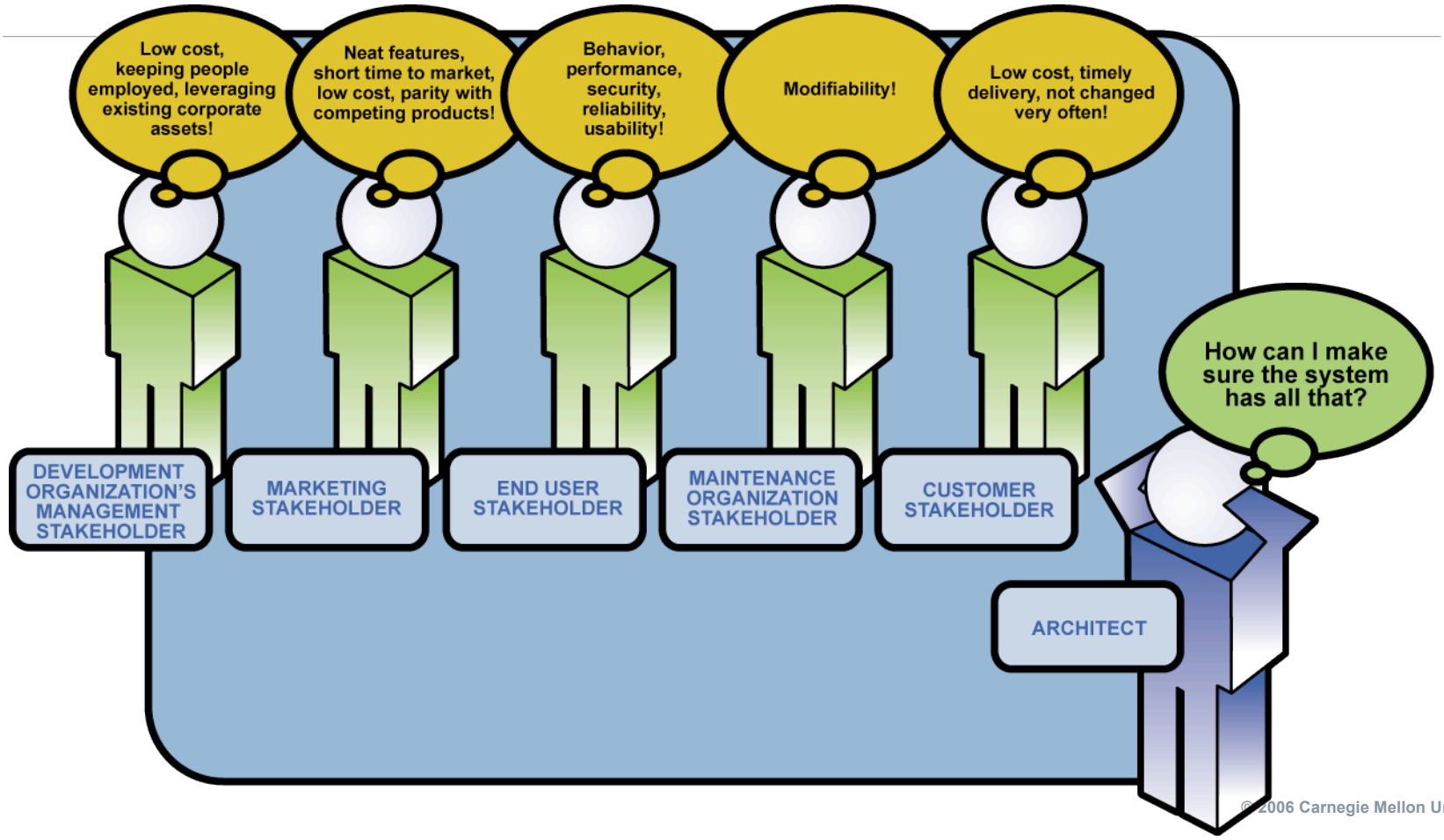
Stakeholders have an interest in the construction of a software system.

Stakeholders might include

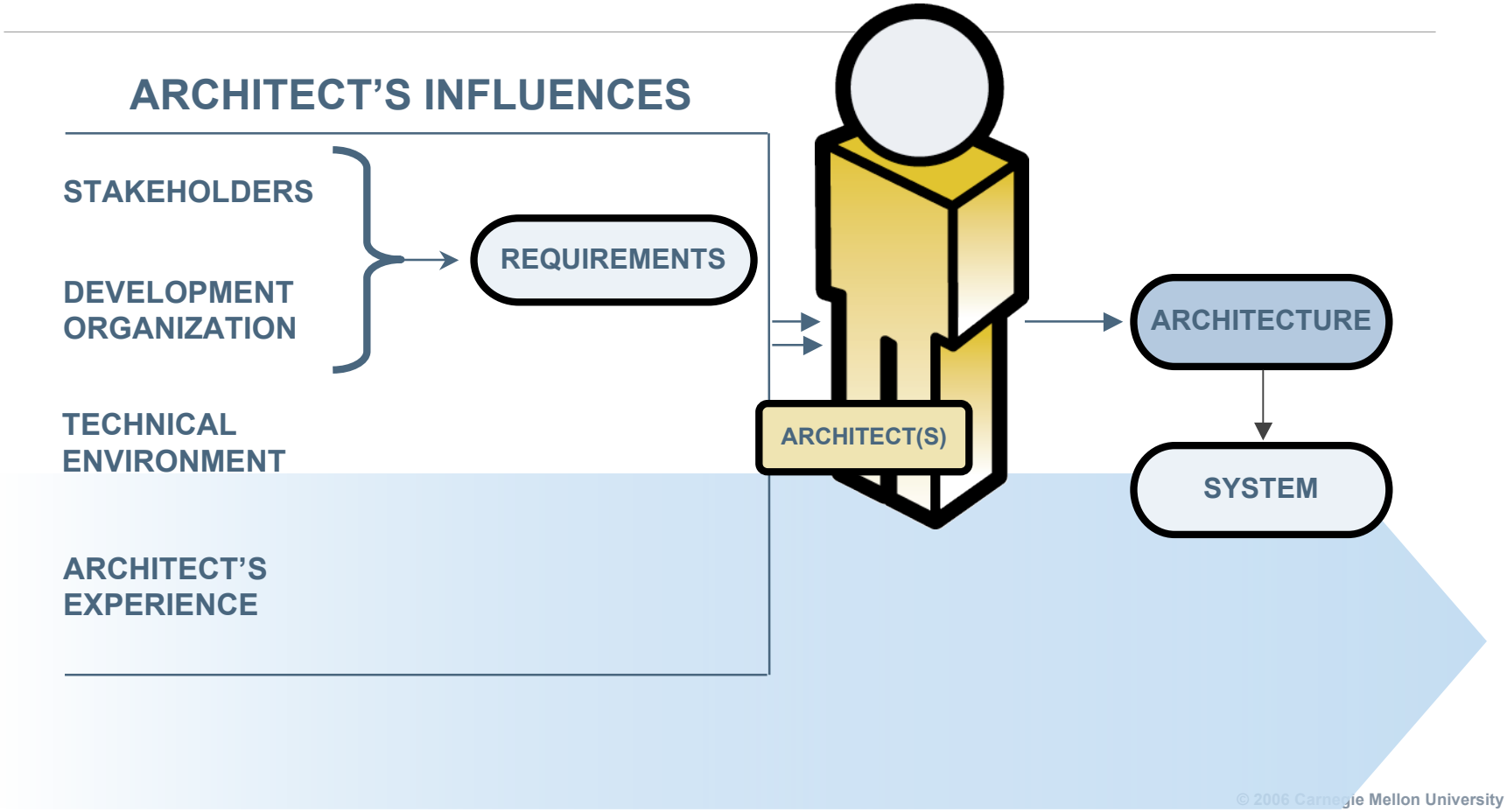
- customers
- users
- developers
- project managers
- marketers
- maintainers

Stakeholders have different concerns that they wish to guarantee and/or optimize.

# INFLUENCE OF SYSTEM STAKEHOLDERS – 2

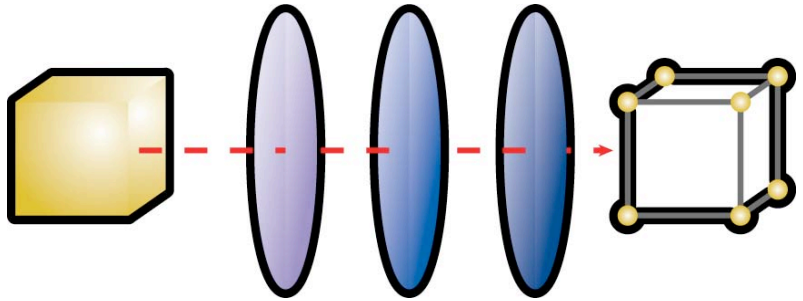


# SUMMARY: INFLUENCES ON THE ARCHITECTURE



# FACTORS INFLUENCED BY ARCHITECTURES

- Structure of the development organization
- Goals of the development organization
- Customer requirements
- Architect's experience
- Technical environment
- The architecture itself

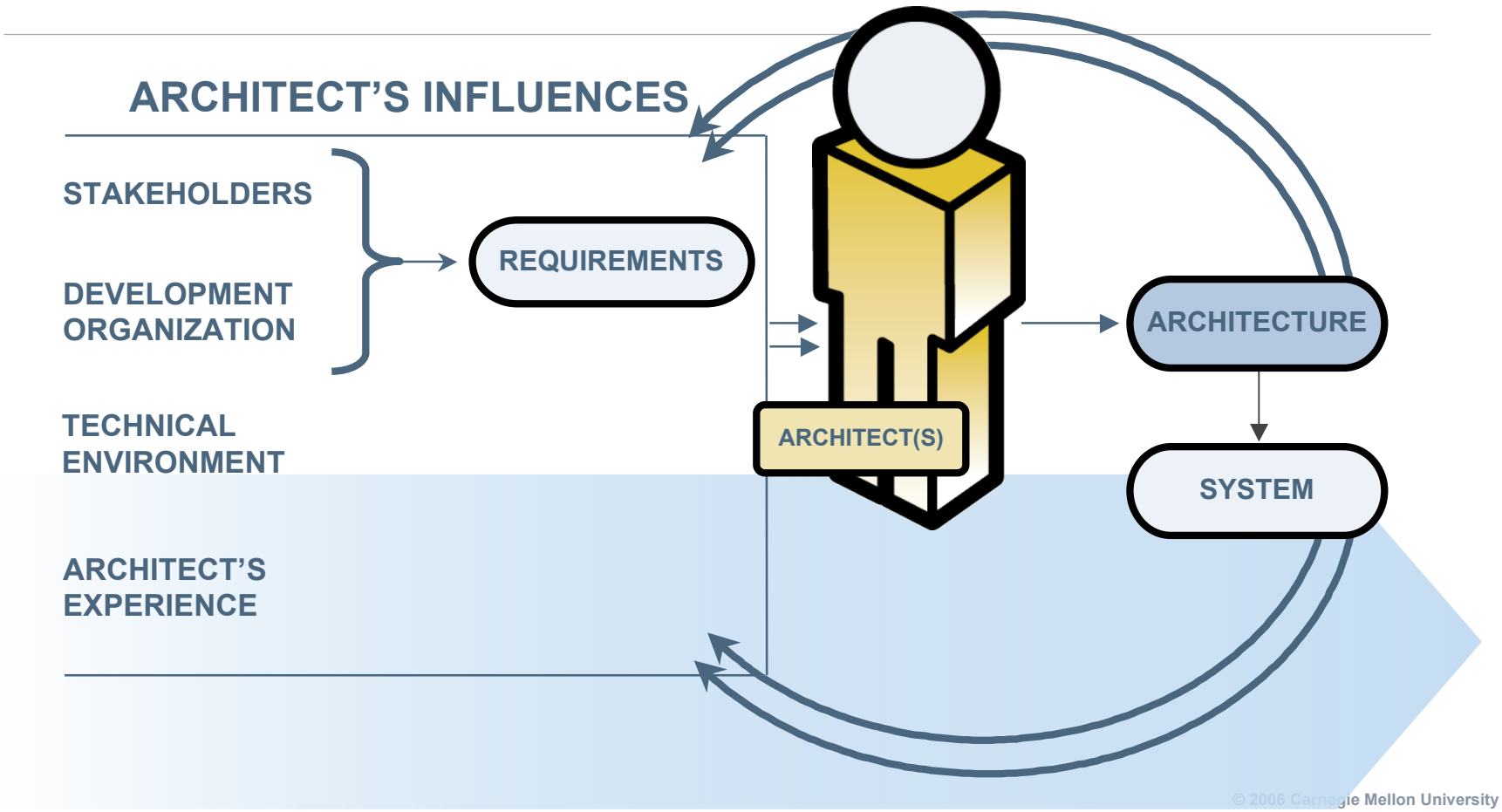


# A CYCLE OF INFLUENCES

Relationships among business goals, product requirements, architects' experience, architectures and fielded systems form a cycle with feedback loops.

- Influences to and from architectures form a cycle.
- An organization can manage this cycle to its advantage.

# ARCHITECTURE BUSINESS CYCLE (ABC)



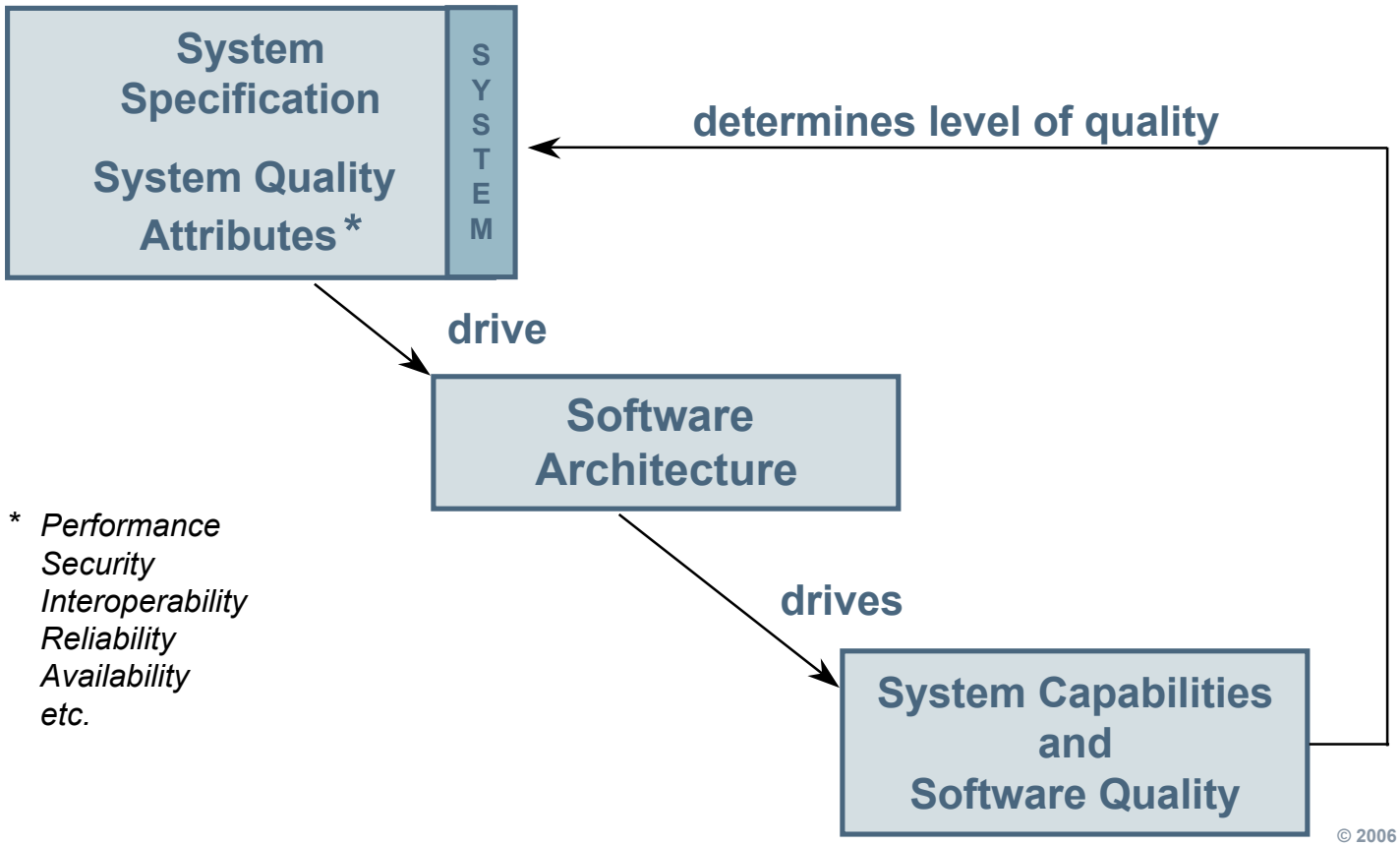
# SOFTWARE ARCHITECTURE AXIOMS

1. Software architecture is the bridge between business and product goals and a software-intensive system.

2. Quality attribute requirements drive software architecture design.

3. Software architecture drives software development through the life cycle.

# SYSTEM QUALITIES AND SOFTWARE ARCHITECTURE




* *Performance*
*Security*
*Interoperability*
*Reliability*
*Availability*
*etc.*

# SOFTWARE ARCHITECTURE COROLLARIES

1. Software architecture is the bridge between business and product goals and a software-intensive system.
2. **Quality attribute requirements drive the design of the software architecture.**
   - Quality attribute requirements stem from business and mission goals.
   - Key quality attributes need to be characterized in a system-specific way.
   - Scenarios are a powerful way to characterize quality attributes and represent stakeholder views.
3. Software architecture drives software development throughout the life cycle.

# SOFTWARE ARCHITECTURE COROLLARIES

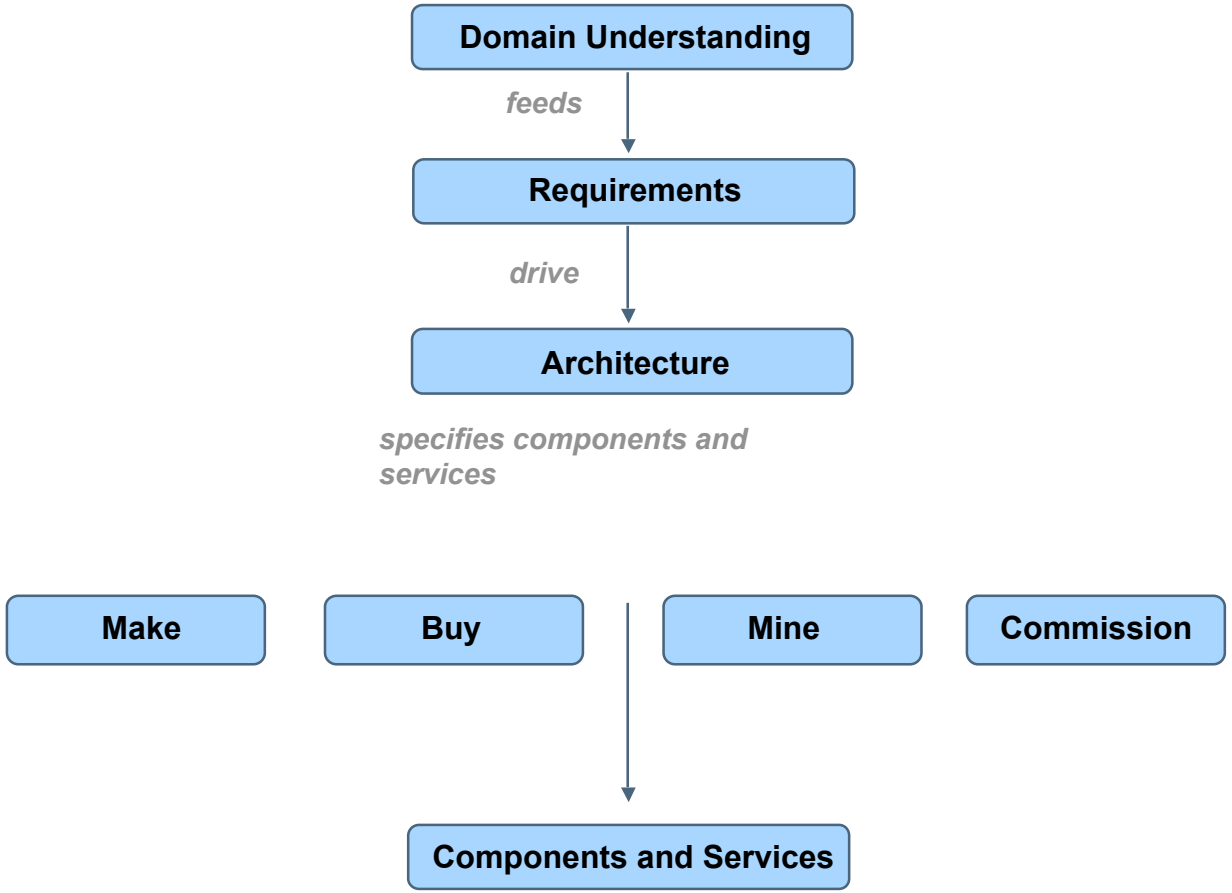
1. Software architecture is the bridge between business and product goals and a software-intensive system.
2. Quality attribute requirements drive the software architecture design.
3. **Software architecture drives software development throughout the life cycle.**
   - Software architecture must be central to development activities.
   - These activities must have an explicit focus on quality attributes.
   - These activities must directly involve stakeholders.
   - The architecture must be ***descriptive*** **and** ***prescriptive*****.**

# ARCHITECTURE-CENTRIC DEVELOPMENT ACTIVITIES

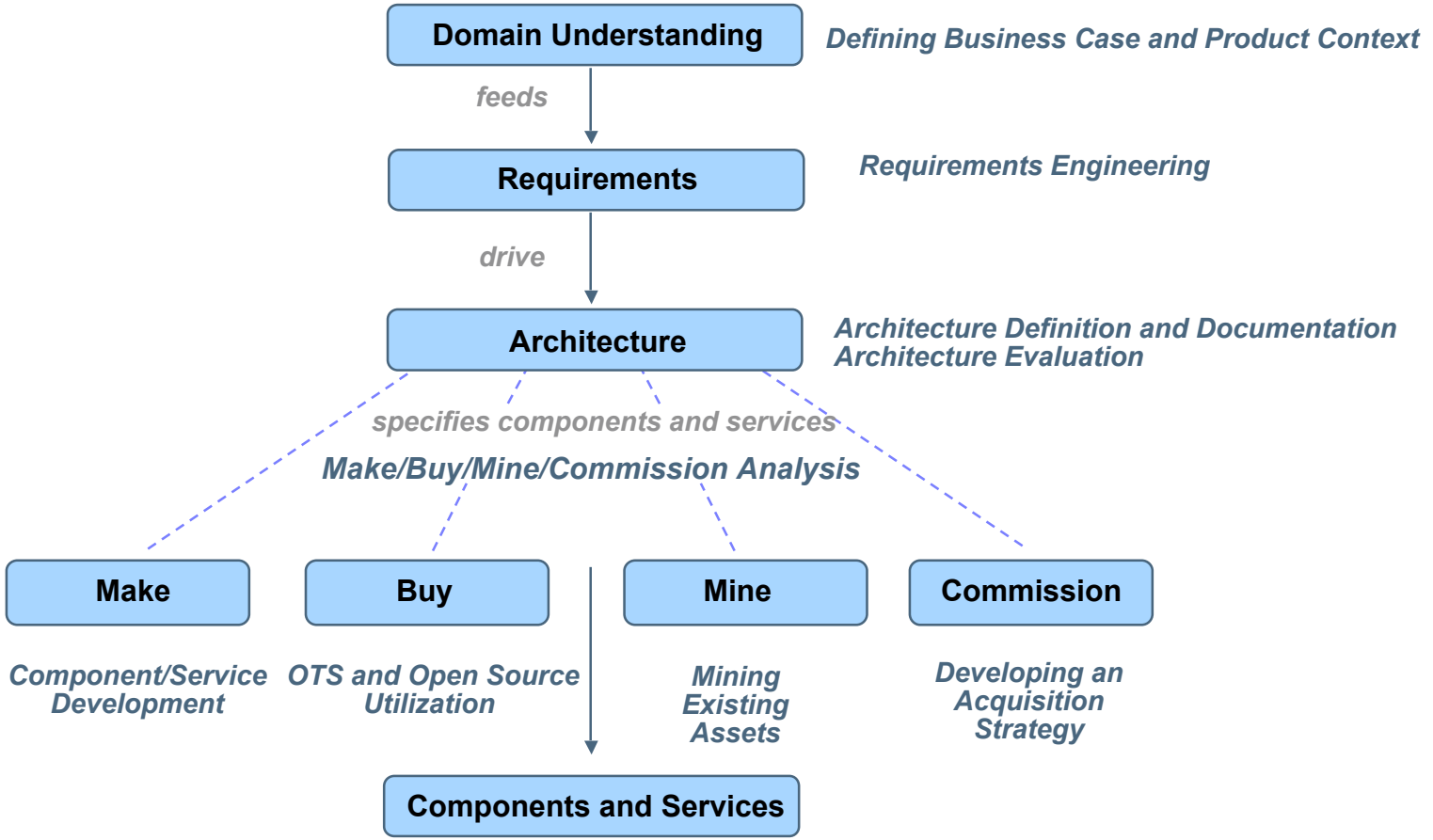
Architecture-centric activities include the following:

- creating the **business case** for the system
- understanding the **requirements**
- **creating and/or selecting** the architecture
- **documenting and communicating** the architecture
- analyzing or evaluating the architecture
- setting up the appropriate **tests and measures** against the architecture
- **implementing** the system based on the architecture
- ensuring that the implementation **conforms** to the architecture

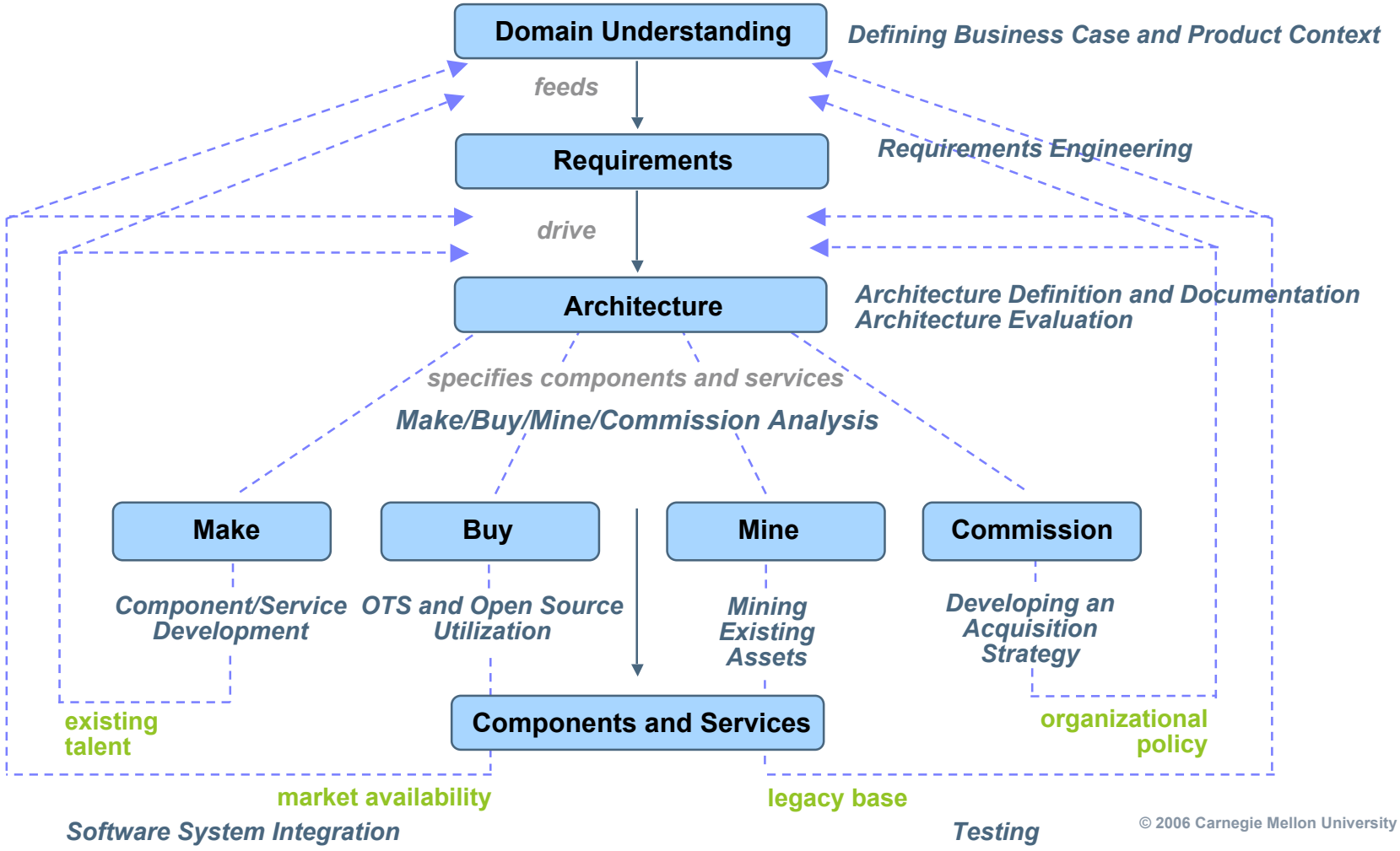
# ARCHITECTURE IN THE LIFE CYCLE



# ARCHITECTURE IN THE LIFE CYCLE



# ARCHITECTURE IN THE LIFE CYCLE



# STAKEHOLDER INVOLVEMENT

The organizational goals and the system properties required by the business are rarely understood, let alone fully articulated.

Customer quality attribute requirements are seldom documented, which results in

- goals not being achieved
- inevitable conflict between different stakeholders

Architects must identify and actively engage stakeholders in order to

- understand real constraints of the system
- manage the stakeholders' expectations
- negotiate the system's priorities
- make tradeoffs

# WHAT MAKES A GOOD ARCHITECTURE?

There is no such thing as an inherently good or bad architecture. Architectures are more or less fit for some stated purpose.

The “goodness” of an architecture can be determined with respect to business and product goals

- Assume that two systems are functionally identical. One system can only be “better” if its architecture promotes qualities that are required to meet business goals and/or product goals.

# IMPEDIMENTS TO ACHIEVING ARCHITECTURAL SUCCESS

## Lack of

- adequate architectural talent and/or experience
- time spent on architectural design and analysis
- architecture-centric acquisition practices

## Failure to

- identify the key quality attributes, characterize them, and design for them
- properly document and communicate the architecture
- evaluate the architecture in a qualitative way
- understand that standards are not a substitute for a software architecture
- ensure that the architecture directs the implementation
- evolve the architecture and maintain documentation that is current
- understand that a software architecture does not come free with COTS or services

# CHALLENGES FOR THE ARCHITECT

- What are the driving quality attributes for your system?
- What precisely do these quality attributes such as modifiability, security, performance, and reliability mean?
- How do you architect to ensure the system will have its desired qualities?
- How do you document a software architecture?
- How do you know if software architecture for a system is suitable without having to build the system first?
- Can you recover an architecture from an existing system?

# SOME SEI TECHNIQUES AND METHODS

- creating the **business case** for the system
- understanding the **requirements**
  - *Quality Attribute Workshop (QAW)*
- **creating and/or selecting** the architecture
  - *Attribute-Driven Design (ADD) and ArchE*
- **documenting and communicating** the architecture
  - *Views and Beyond Approach*
- **analyzing or evaluating** the architecture
  - *Architecture Tradeoff Analysis Method (ATAM)*
  - *Cost Benefit Analysis Method (CBAM)*
- **implementing** the system based on the architecture
- ensuring that the implementation **conforms** to the architecture
  - *ARMIN (and DiscoTect)*

# SEI METHODS AND QUALITY ATTRIBUTES

**QAW**

**ADD**

**ArchE**

**Views and Beyond**

**ATAM**

**CBAM**

**ARMIN**

**(DiscoTect)**

- are explicitly focused on quality attributes
- directly link to business and mission goals
- explicitly involve system stakeholders
- are grounded in state-of-the-art quality attribute models and reasoning frameworks
- are documented for practitioner consumption
- are applicable to real-world challenges and systems

# TRENDS IN SOFTWARE ARCHITECTURE - 1

Organizations big and small are recognizing the importance of software architecture. For example,

- Microsoft
  - Regional Architecture Forums
  - Architect's Council
  - Architect Certification
- Raytheon
  - Architecture Center of Excellence
  - mandatory architecture classes and methods
- IBM
  - Grady Booch writing the online Architect's Handbook
- Automotive domain
  - Siemens, Bosch, and Delphi all have architecture initiatives
- US Army
  - Army Software Architecture Initiative

# TRENDS IN SOFTWARE ARCHITECTURE - 2

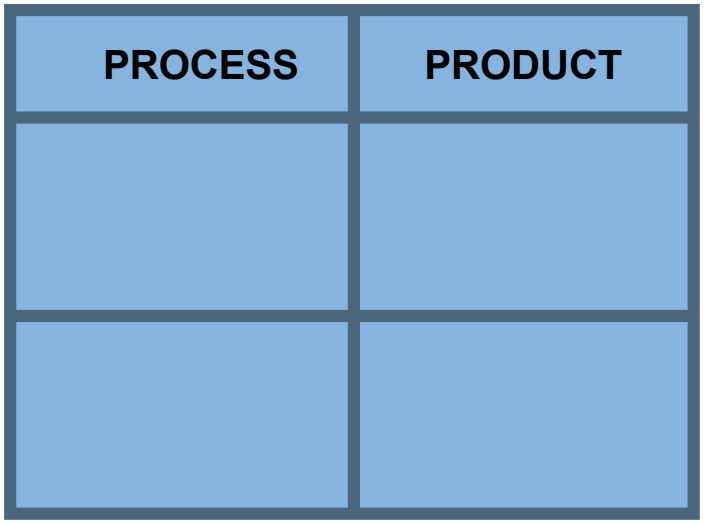
Books, courses, certificate programs, conferences, workshops on software architecture abound.

New technologies (MDA, SOA, aspects) change the incidentals but the fundamentals of software architecture and quality attributes are enduring.

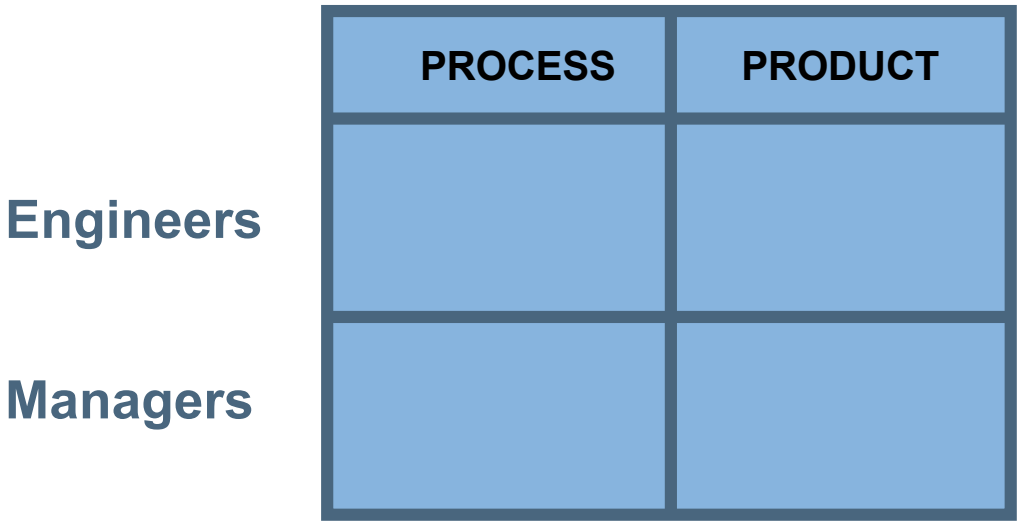
# Let's Teach Architecting High Quality Software

## Teach

# SOFTWARE ENGINEERING PERSPECTIVES - 1



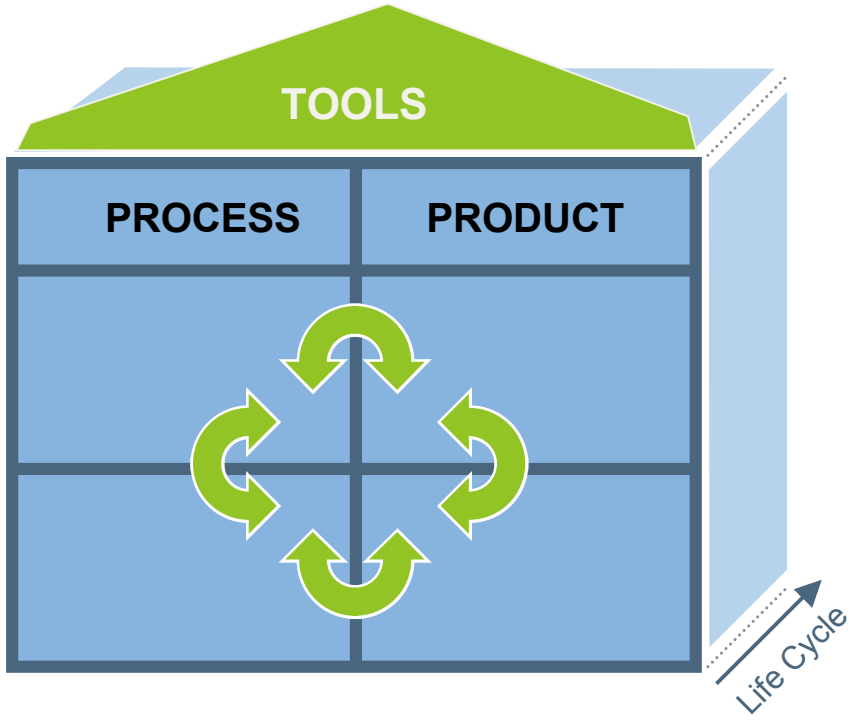
# SOFTWARE ENGINEERING PERSPECTIVES - 2



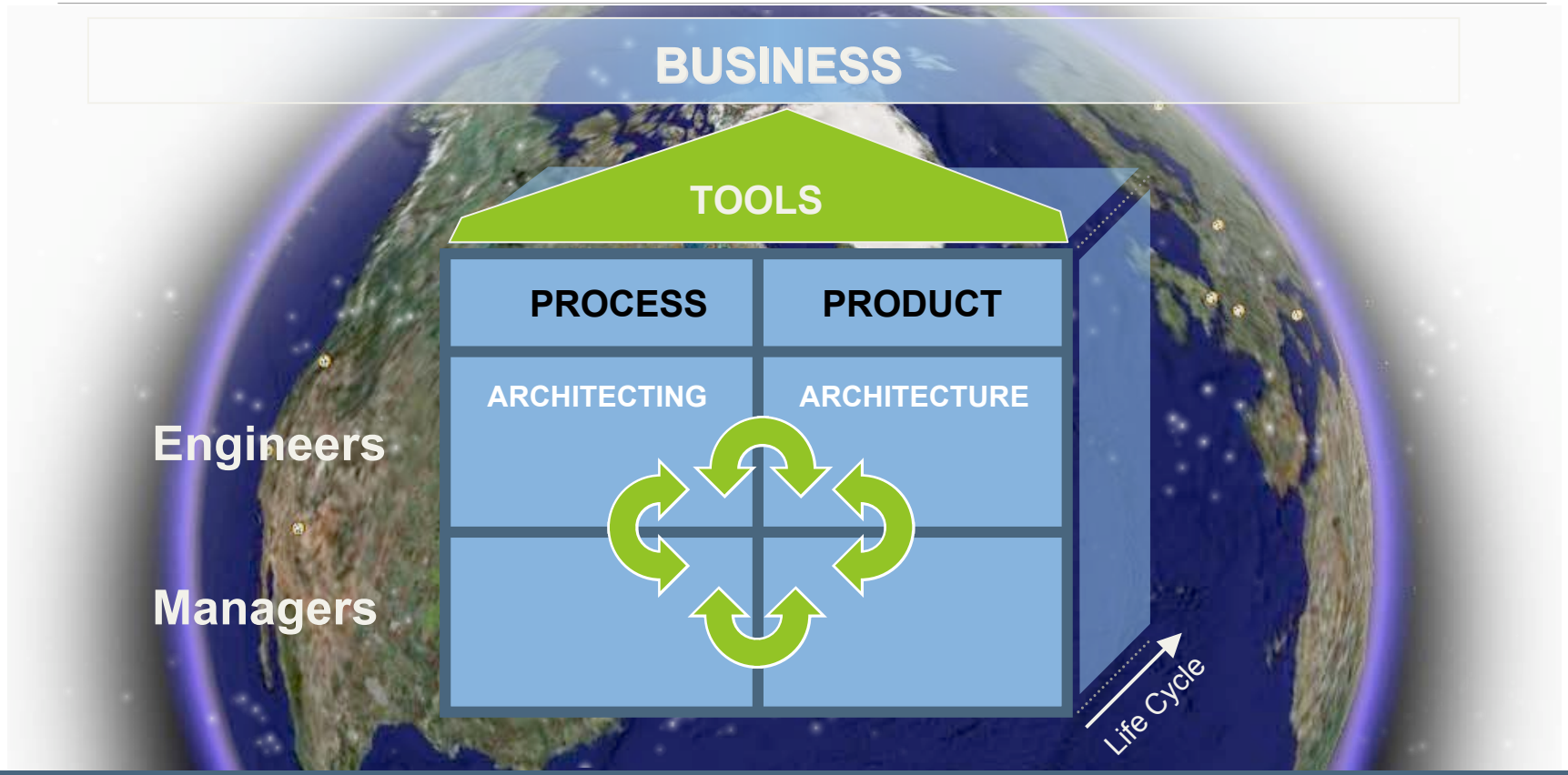
# SOFTWARE ENGINEERING PERSPECTIVES - 3

Engineers

Managers



# SOFTWARE ARCHITECTURE




BOTH **INFLUENCES** AND **IS INFLUENCED** BY ALL PERSPECTIVES

# SOFTWARE ENGINEERING EDUCATION

What do we teach and how does it address this model?

The norm

Product perspective: OOA + OOD + OOP + …

Process perspective: agile methods, XP, TSP + …

These are a good starting perspective but software engineering students need more

- Business context
- Quality attributes
- Software architecture

# ARCHITECTURE PRINCIPLES TO TAKE AWAY

Software architecture is important because it

- provides a communication vehicle among stakeholders
- is the result of the earliest design decisions
- is a transferable, reusable abstraction of a system

The degree to which a system meets its quality attribute requirements is dependent on architectural decisions.

Every software-intensive system has a software architecture.

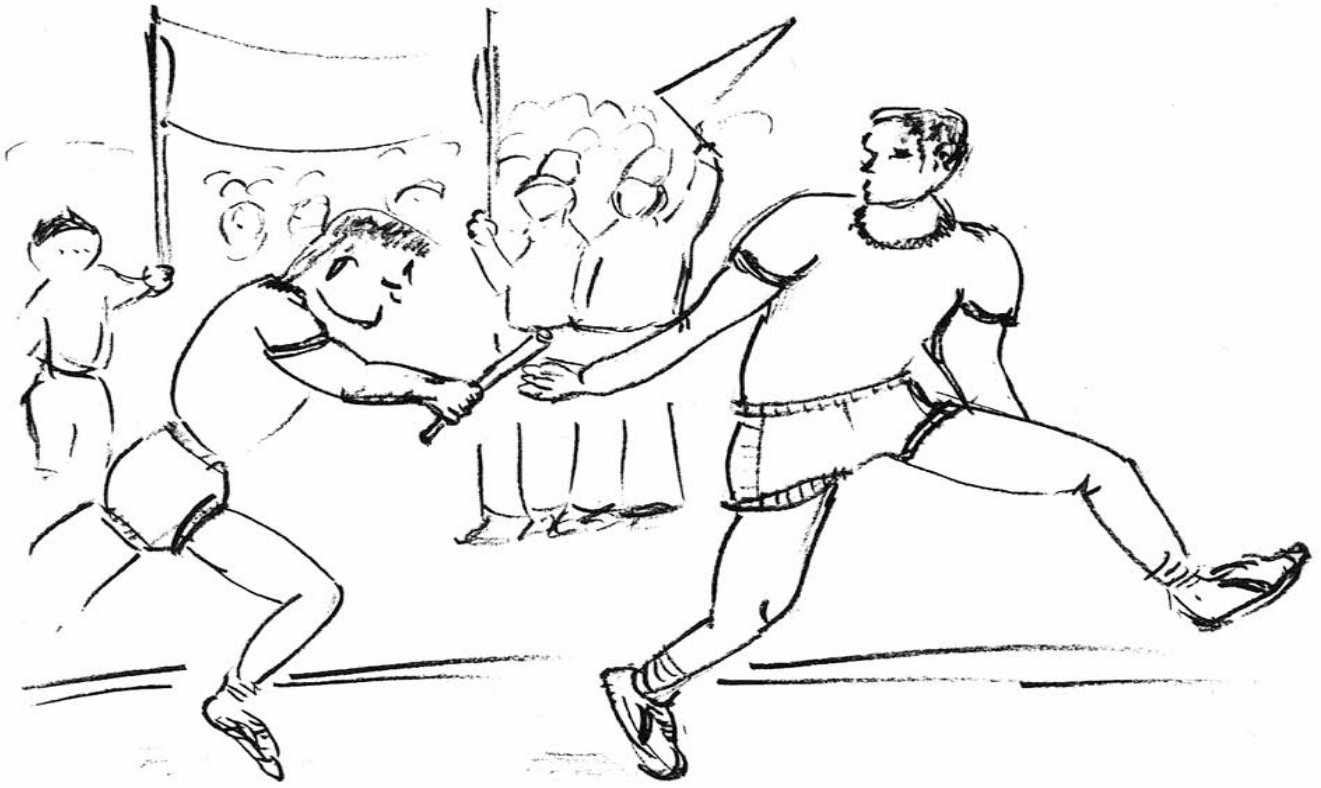
Just having an architecture is different from having an architecture that is known to everyone, much less one that is fit for the system's intended purpose.

An architecture-centric approach is critical to achieving and implementing an appropriate architecture.

High quality software requires architecture practices.

# THE ROLE OF EDUCATORS

# THE WELL EDUCATED GRADUATE



# LET'S TEACH ARCHITECTING HIGH QUALITY SOFTWARE

It's time that all software engineering students know the principles of software architecture and how to use effective software architecture practices.

Every facet of our society depends on software.

To ensure high quality software we need to teach our students to architect high quality software.

# THANKS TO THE SEI SOFTWARE ARCHITECTURE TEAM

Felix Bachmann, Len Bass, Joe Batman, John Bergey, Phil Bianco, Paul Clements, James Ivers, Larry Jones, Rick Kazman, Mark Klein, Reed Little, Paulo Merson, Robert Nord, William O'Brien, Ipek Ozkaya, Rob Wojcik, Bill Wood

# THANKS TO MY DEAR FRIEND AND COLLEAGUE

Coach

*I dedicate this keynote to Jim Tomayko. Jim was a leading contributor to software engineering education and a role model to all educators.*

# THANK YOU!

It has been my honor and pleasure to spend this time with you.

**Linda Northrop**
Director
Product Line Systems Program
Telephone: 412-268-7638
Email: lmn@sei.cmu.edu

**For more information:**
http://www.sei.cmu.edu/architecture/sat_init.html

# REFERENCES


***Evaluating Software Architectures: Methods and Case Studies***
Clements, P.; Kazman, R.; & Klein, M. Reading, MA: Addison- Wesley, 2002.

***Software Architecture in Practice, Second Edition***
Bass, L.; Clements, P.; & Kazman, R. Reading, MA: Addison-Wesley, 2003.

***Documenting Software Architectures: Views and Beyond***
Clements, P.; Bachmann, F.; Bass, L.; Garlan, D.; Ivers, J.; Little, R.; Nord, R.; & Stafford, J. Reading, MA: Addison-Wesley, 2002.

***Software Product Lines: Practices and Patterns***
Clements, P.; Northrop, L. Reading, MA: Addison-Wesley, 2001.