Architecting High Quality Software: The Role of Software Architecture in System Development and Evolution

Linda Northrop

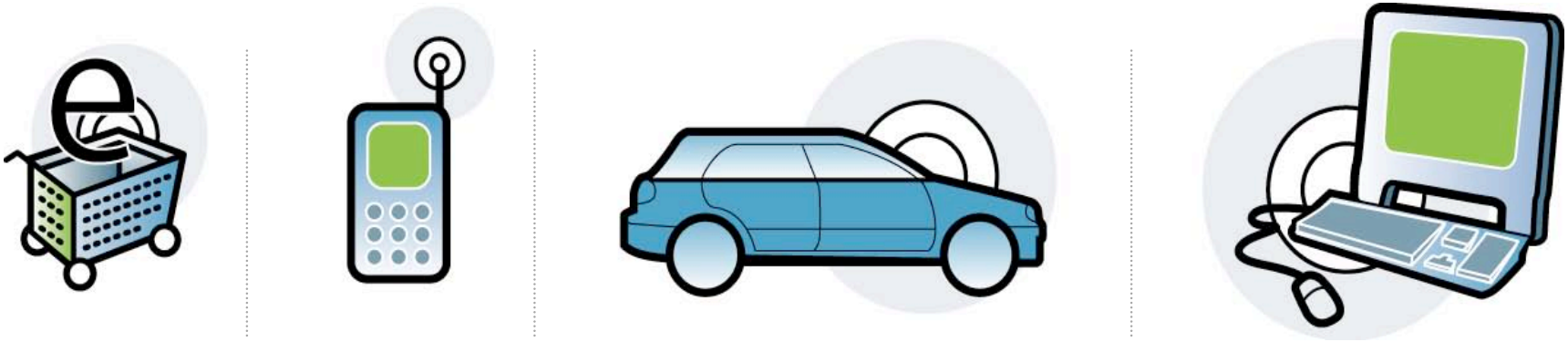
Director, Product Line Systems Program

SEI

Software

Society's Dependence on Software

Business Success Requires Software Prowess

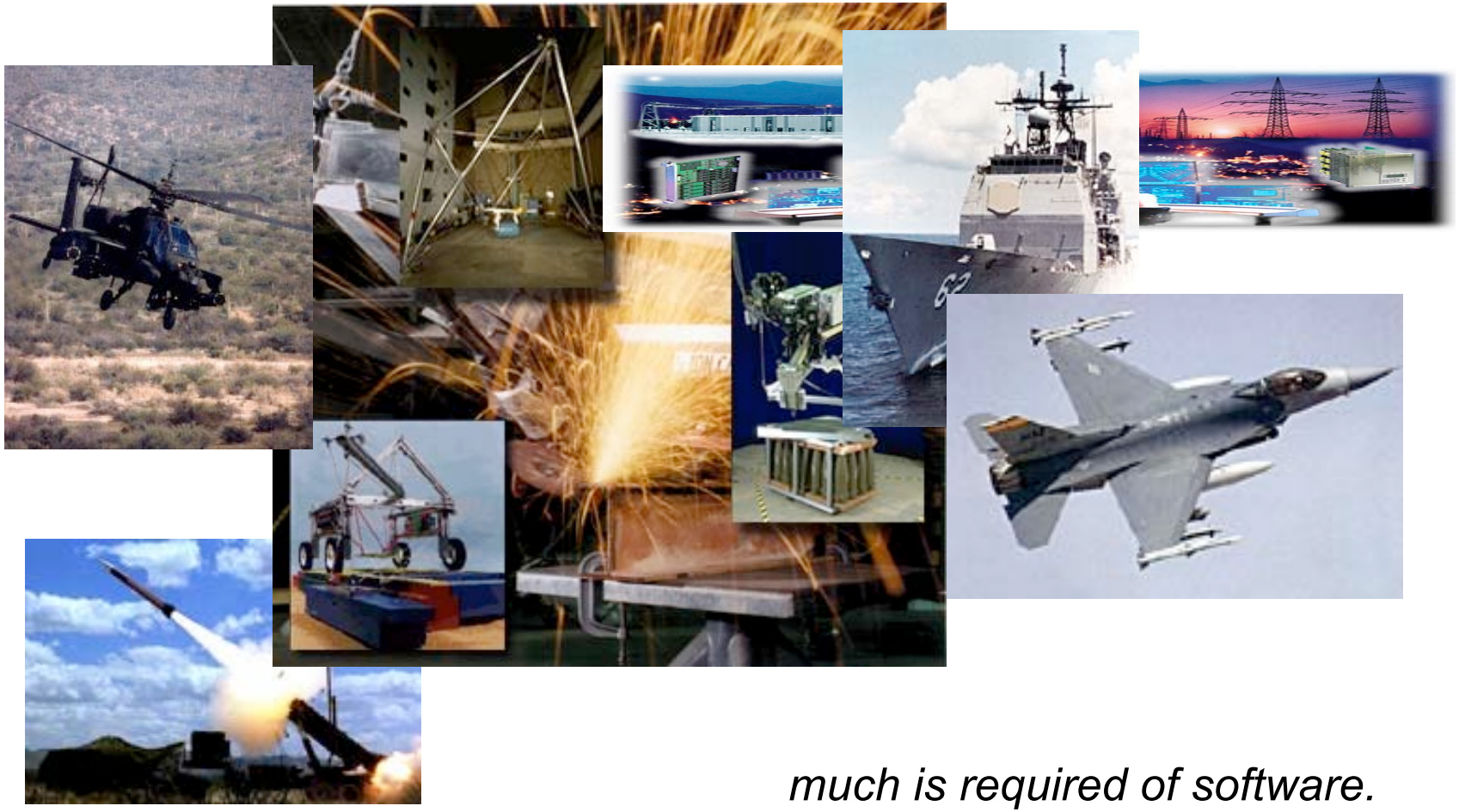


Software pervades every sector.

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

Cell Phone Today	Cell Phone in 2010	This year's cars	Cars in 2010
~2 million lines of code	~ 10 million lines of code	~35 million lines of code	~ 100 million lines of code

Where Behavior Counts Most



much is required of software.

Quality

Quality

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

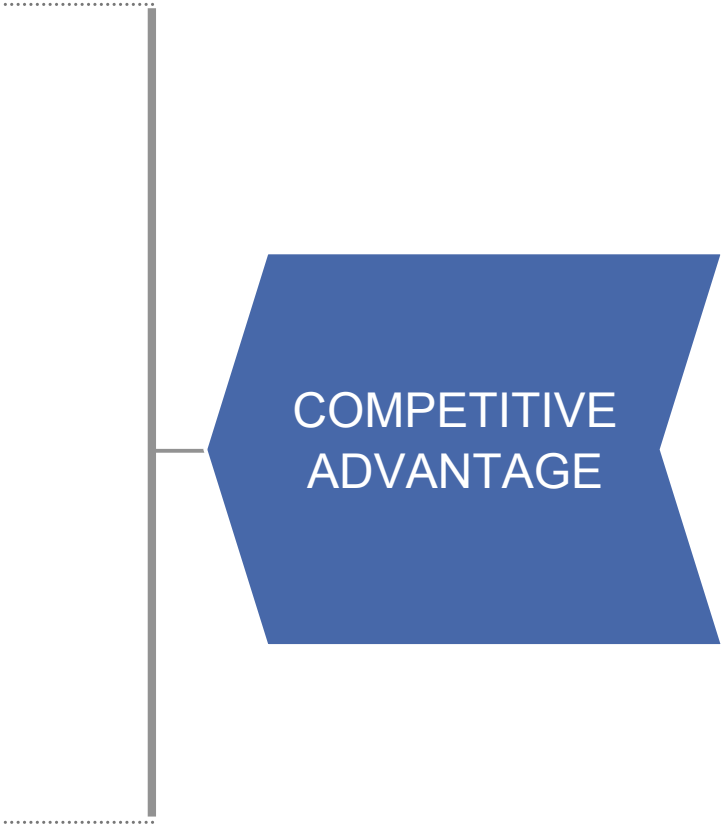
High Quality

High quality software meets business goals and user needs.

It has the right features and the right attributes.

Universal Business Goals

- High quality
- Quick (or right) time to market
- Increased market share
- Effective use of limited resources
- Product alignment
- Low-cost production
- Low-cost maintenance
- Market agility
- Mind share



The Ultimate Universal Goal

User Needs

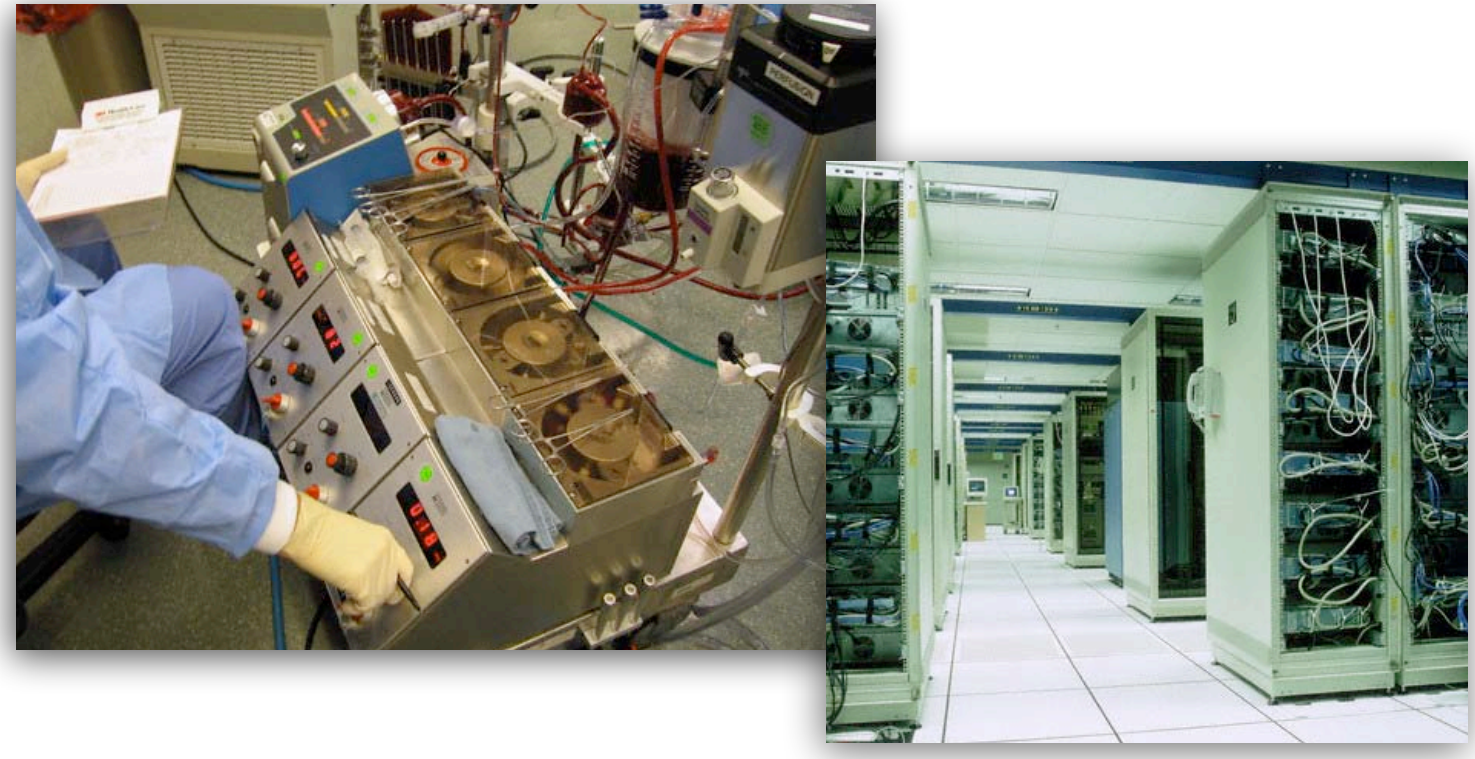
Required capability
Low learning threshold
Ease of use
Predictable behavior
Dependable service
Timely response
Timely throughput
Protection from unintended intruders and viruses
……

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

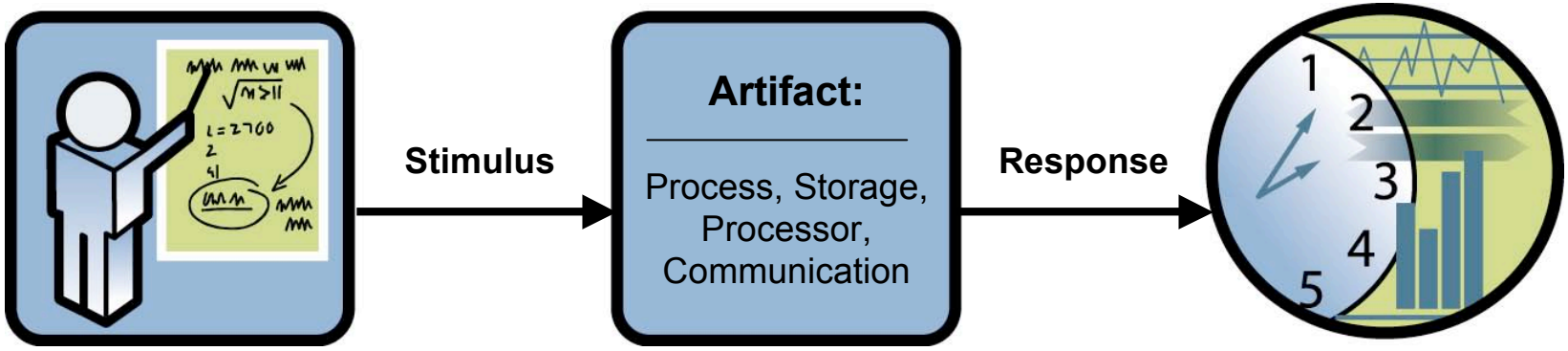
- Performance
- Availability
- Usability
- Modifiability
- Security
- Etc.

Quality attribute requirements stem from business goals and user needs.

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



SOURCE **ENVIRONMENT** **RESPONSE MEASURE**

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.

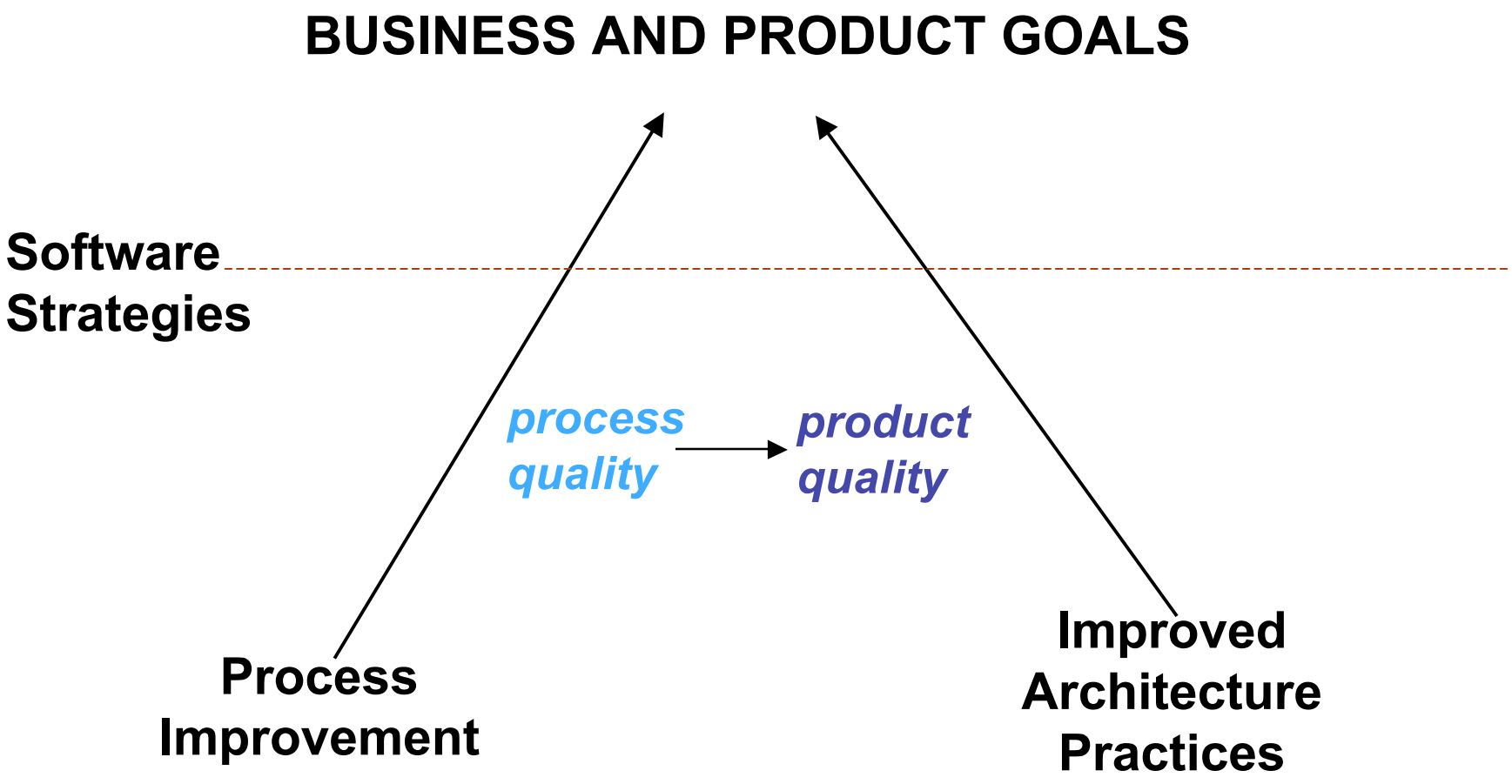
- Modifiability
- Interoperability
- Availability
- Security
- Predictability
- Portability

analysis, design, development

Quality Attribute Drivers → **Software Architecture** → **Software**

has these qualities

Software Strategies Are Needed



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.

Sample Issues Stemming From Architectural Decisions

Availability:
having a single point of failure
not including availability mechanisms
using infrastructure that does not support availability mechanisms

Performance:
not knowing performance requirements
not performing any performance modeling or prototyping
unfamiliarity with infrastructure choices
not using known performance mechanisms

Security:
not using known mechanisms to support security goals

Modifiability:
allocating functionality in a way that jeopardizes portability
supporting the addition and deletion of different devices
lack of attention to potential growth paths

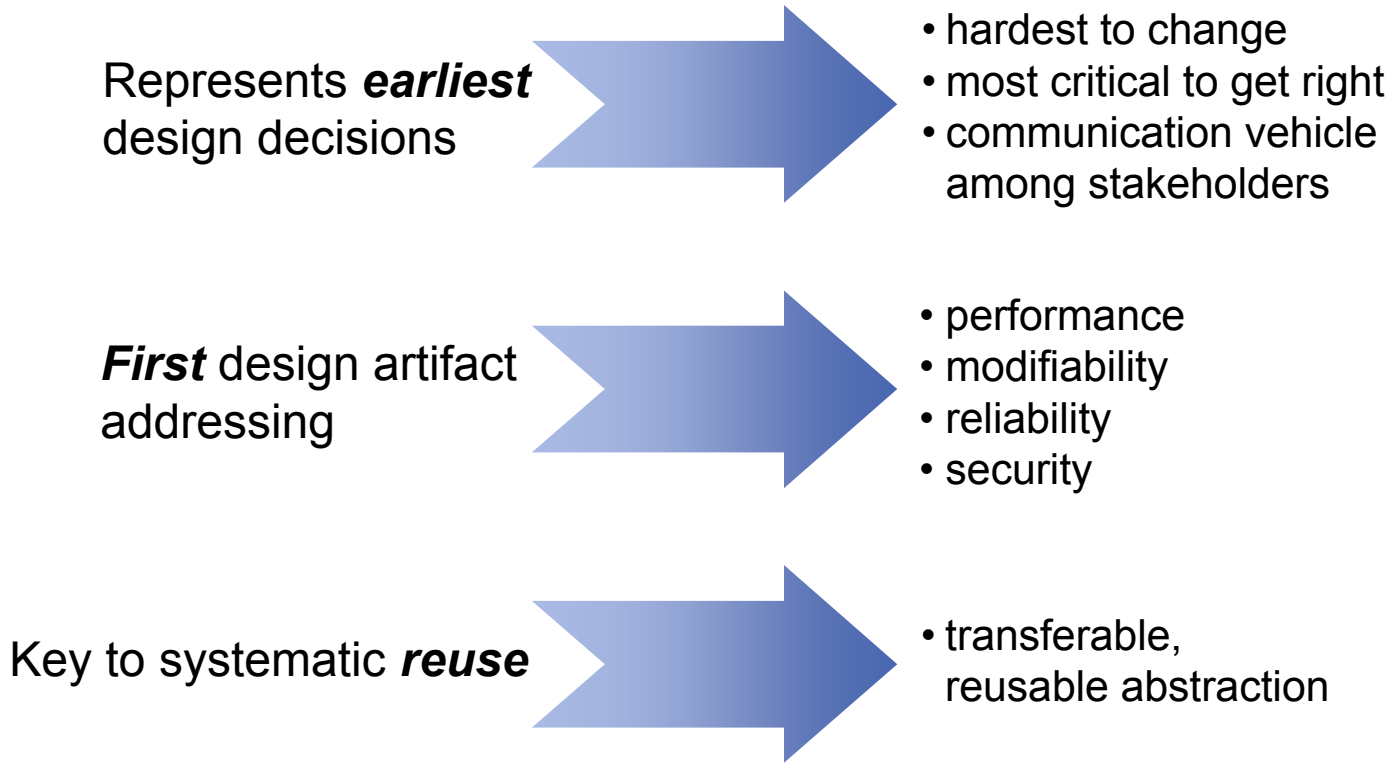
Integration:
problems with migrating legacy systems
lack of uniformity in key areas

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?



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.

It is

- the carrier of most system quality attributes
- a forum for resource tradeoffs
- a contract that allows multi-party development
- an essential part of complex systems

Definition of 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.

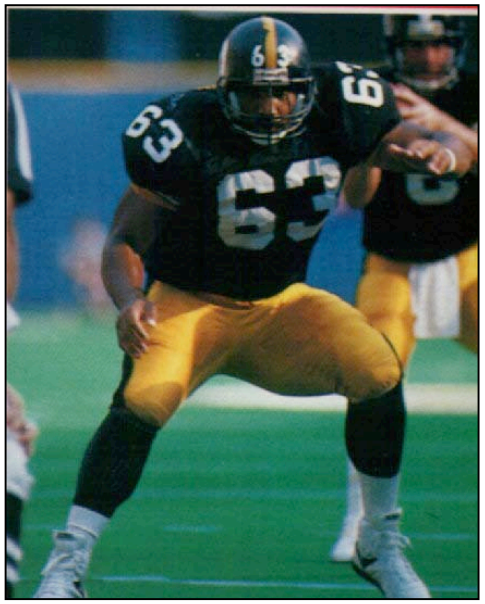
If you don't develop an architecture, you will get one anyway – and you might not like what you get!

Representing Software Architectures

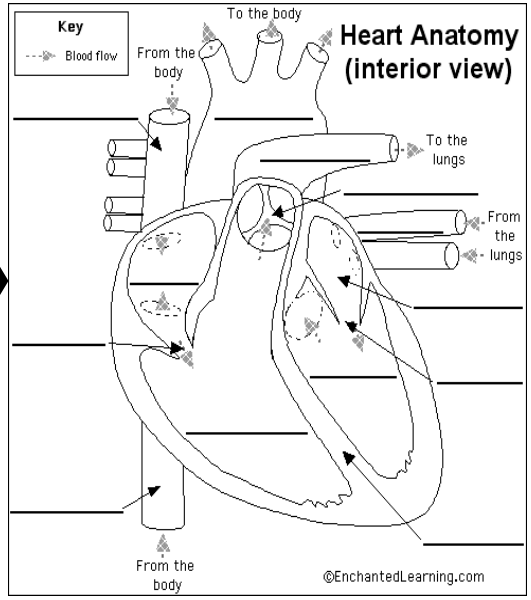
A software architecture is often depicted using an ad hoc box-and-line drawing of the system that is intended to solve the problems articulated by the specification.

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

Structures And Views - 1



A human body comprises multiple *structures.*

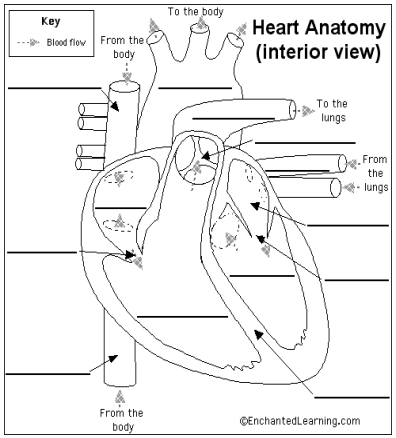


a *static view* of one human *structure*

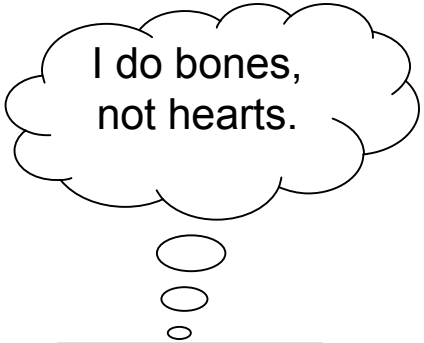
a *dynamic view* of that *structure*

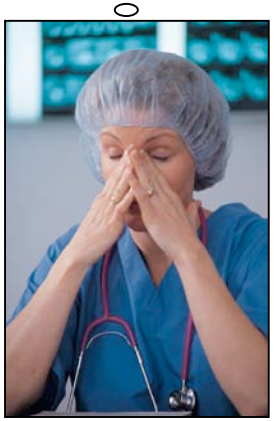
One body has many structures, and those structures have many views.
So it is with software…

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]

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.

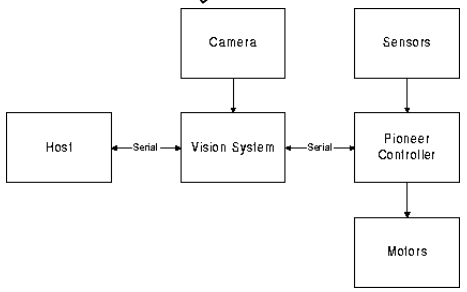
Where Do Architectures Come From?

Requirements Beget Design

Requirements in various forms

Available knowledge

System



Designer

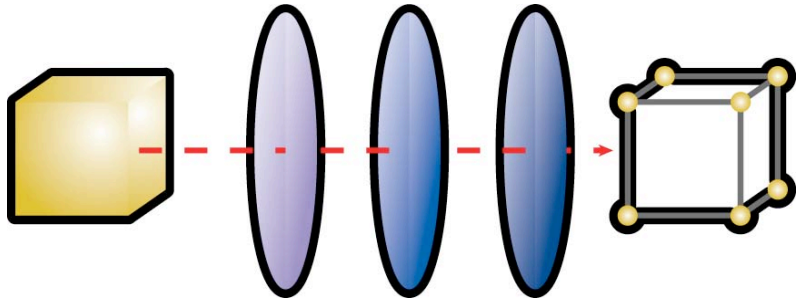
Architecture

Factors Influencing Architectures

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

Architectures are influenced by

- stakeholders of a system
- development organization
- technical environment
- architect's background

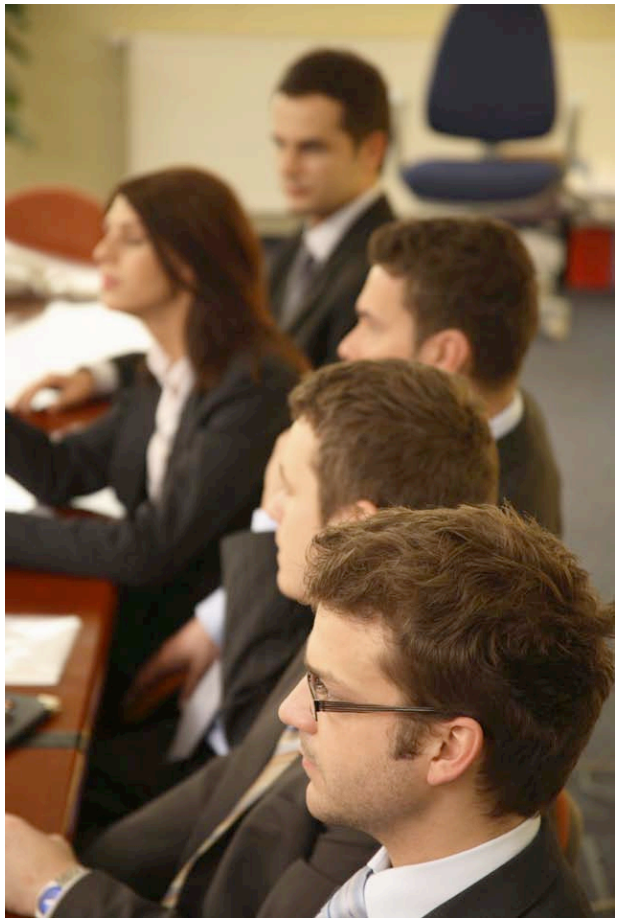


Who Are The System Stakeholders?

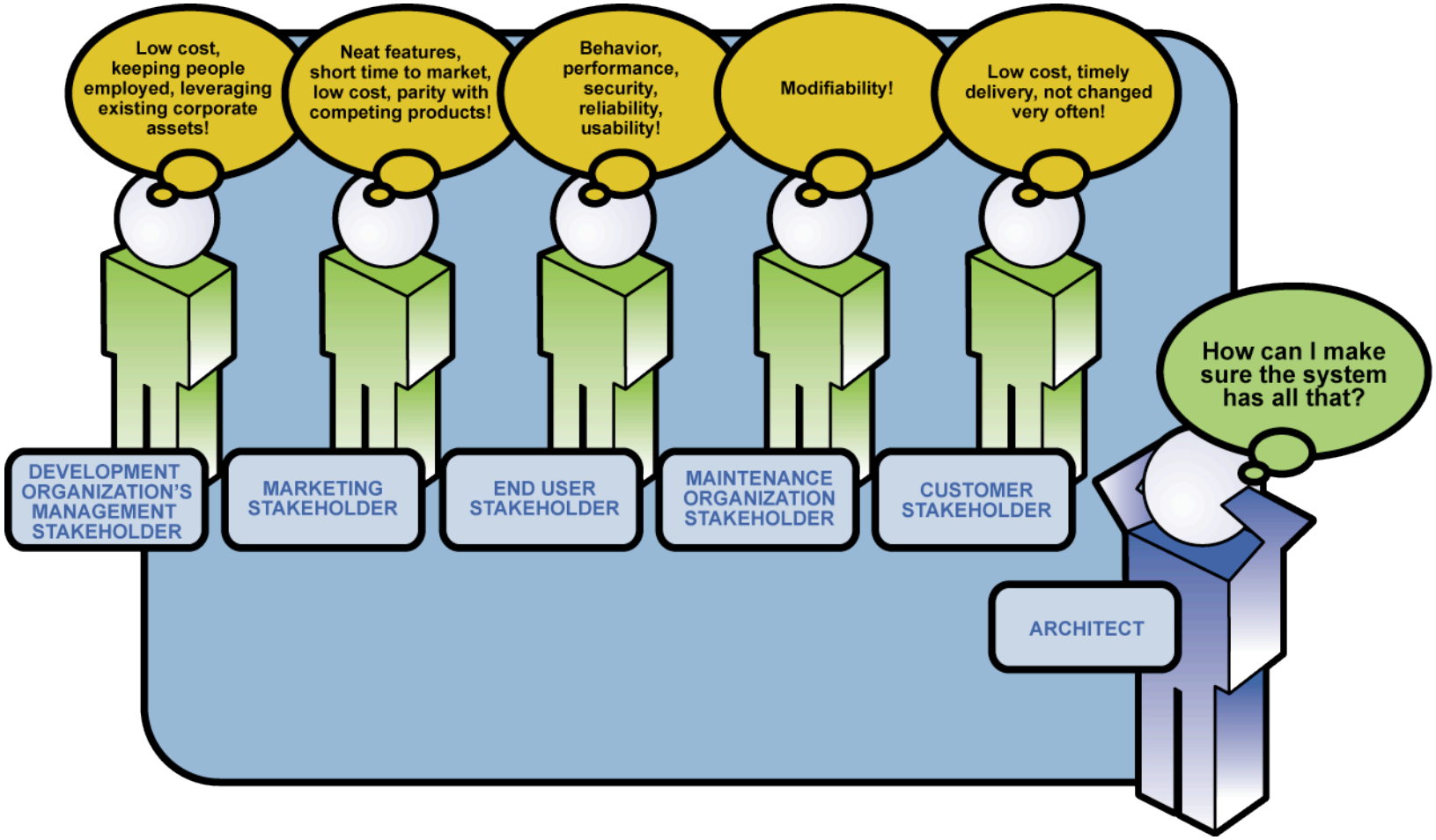
Stakeholders have an interest in the construction of a software system. Stakeholders might include

- customers
- users
- developers
- project managers
- marketers
- Maintainers

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



Influence Of System Stakeholders



Stakeholder Involvement

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

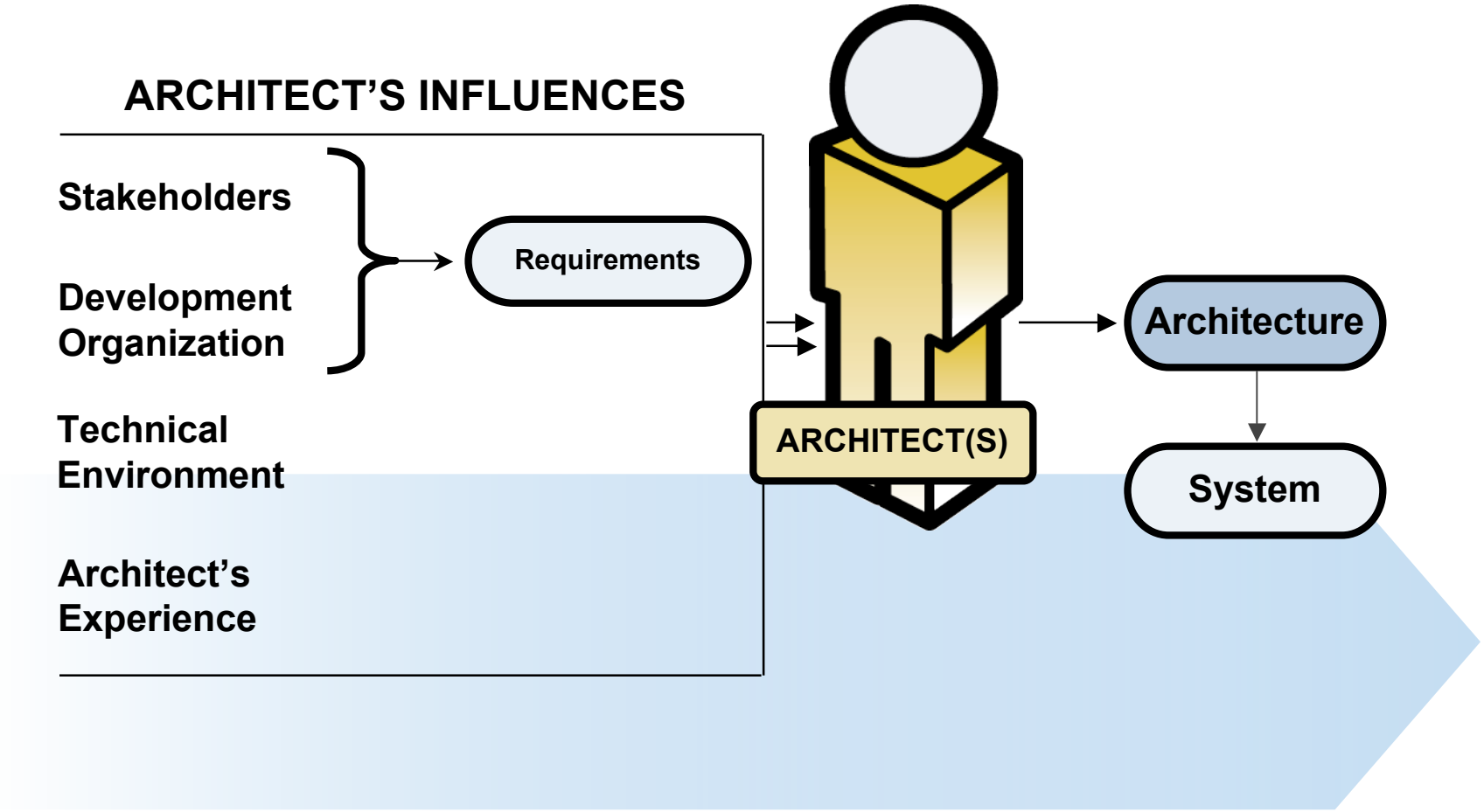
Customer quality attribute requirements are seldom documented, which results in

- 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

Summary: Influences On The Architecture



Factors Influenced By Architectures

Structure of the development organization

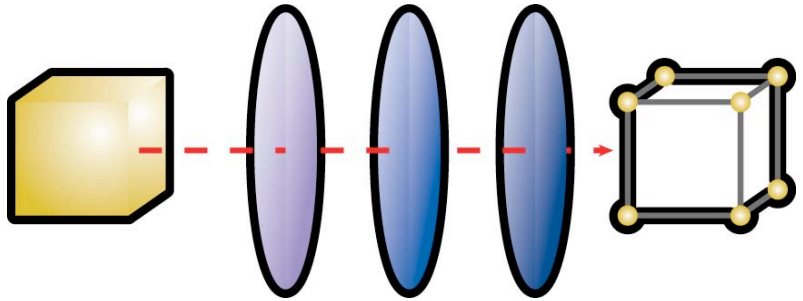
Goals of the development organization

Customer requirements

Architect's experience

Technical environment

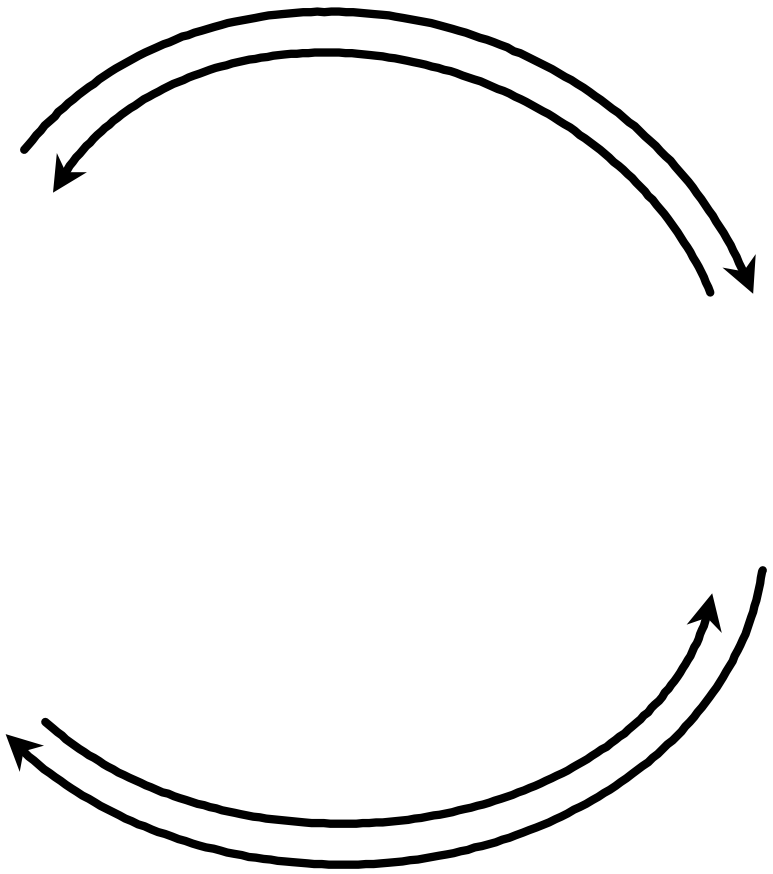
The architecture itself



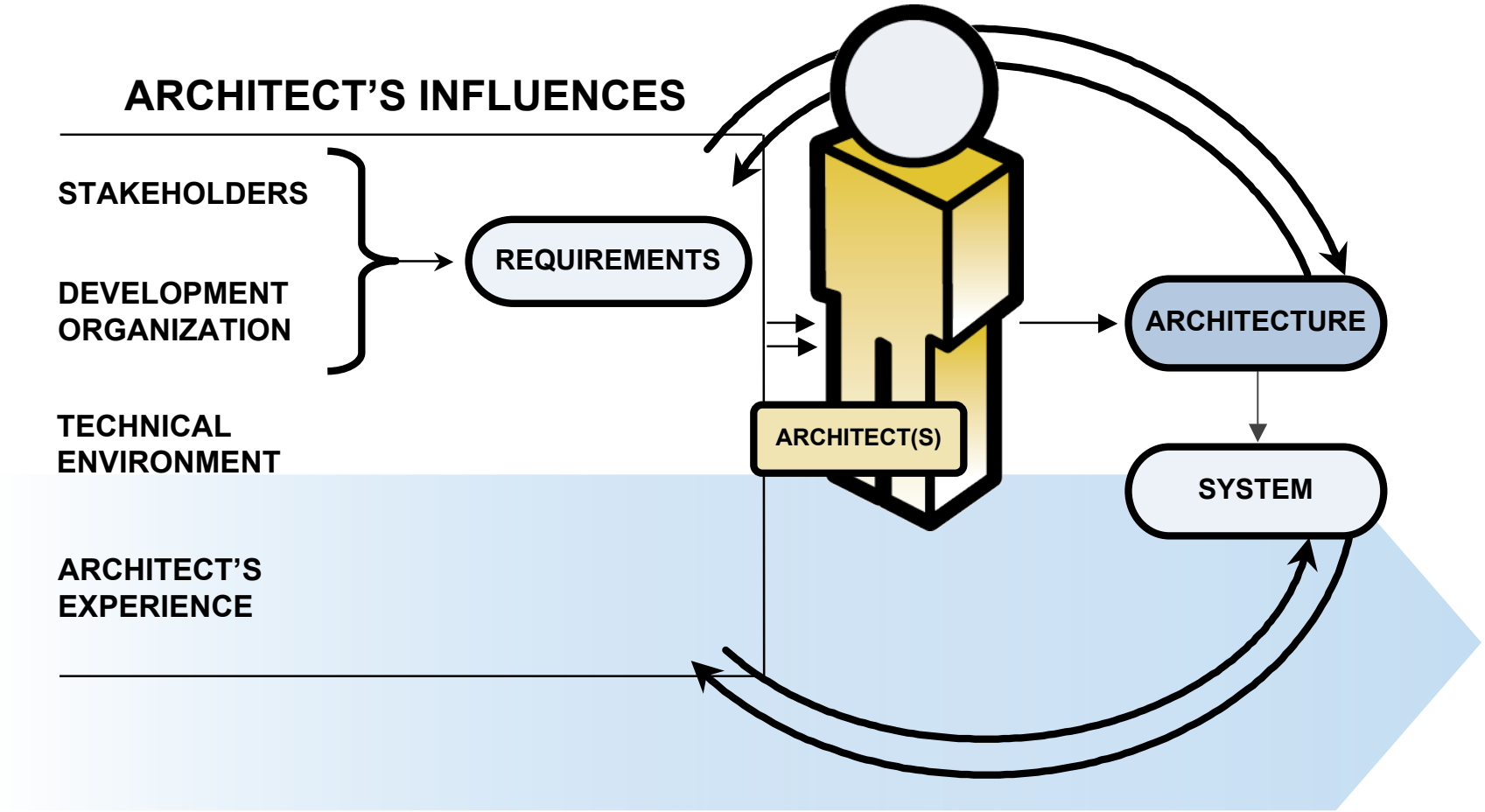
A Cycle Of Influences

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)



Architecting

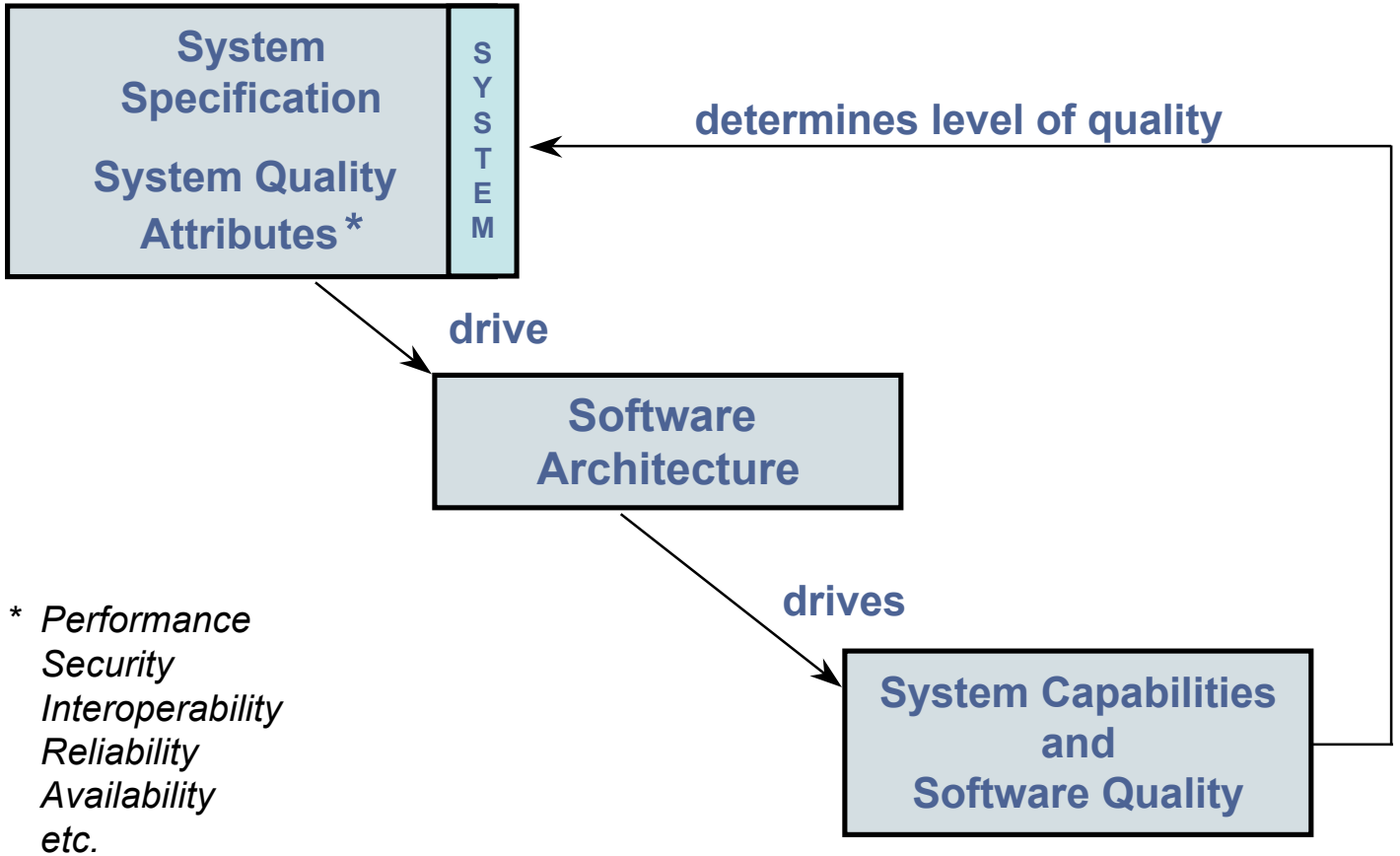
Software Architecture Axioms

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 and evolution 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 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.**
3. Software architecture drives software development and evolution 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 and evolution 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 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
- evolving the architecture so that it **continues to meet business and product goals**

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 which are required to meet business goals.

A system must be buildable within time and budget constraints

- Often the technically “best” solution cannot be created within time, budget, and other constraints.

Process Recommendations

- ✓The architecture should be the product of a single architect or small group of architects with a leader.
- ✓The architect(s) should have the functional requirements for the system and a prioritized list of quality attributes that the system is expected to satisfy.
- ✓The architecture should be well documented with at least one static view and one dynamic view using an agreed to notation that all stakeholders can understand.
- ✓The architecture should be circulated to the system's stakeholders, who should be actively involved in its review.
- ✓The architecture should be analyzed for applicable quantitative measures and formally evaluated for quality attributes before it is too late to change it.
- ✓The architecture should lend itself to incremental implementation via the creation of a "skeletal" system in which the communication paths are exercised, but which at first has minimal functionality.
- ✓The architecture should result in a small, specific set of resource contention areas, the resolution of which are clearly specified, circulated, and maintained.

Structural Recommendations

- ✓The architecture should feature well-defined modules whose functional responsibilities are allocated on the principles of information hiding and separation of concerns.
- ✓Each module should have a well defined interface that encapsulates changeable aspects from other software that uses its facilities, allowing teams to work independently.
- ✓Quality attributes should be achieved using well-known architectural tactics specific to each attribute.
- ✓The architecture should never depend on a particular version of a commercial product or tool.
- ✓Modules that produce data should be separate from modules that consume.
- ✓For parallel-processing systems, the architecture should feature well-defined processes or tasks that do not necessarily mirror the module structure.
- ✓Every task or process should be written so that its assignment to a specific processor can be easily changed, perhaps at runtime.
- ✓The architecture should feature a small number of simple interaction patterns that interact in a consistent way.

Impediments To Architectural Success

Lack of

- adequate architectural talent and/or experience
- time spent on architectural design and analysis
- disciplined use of architecture-centric 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

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?

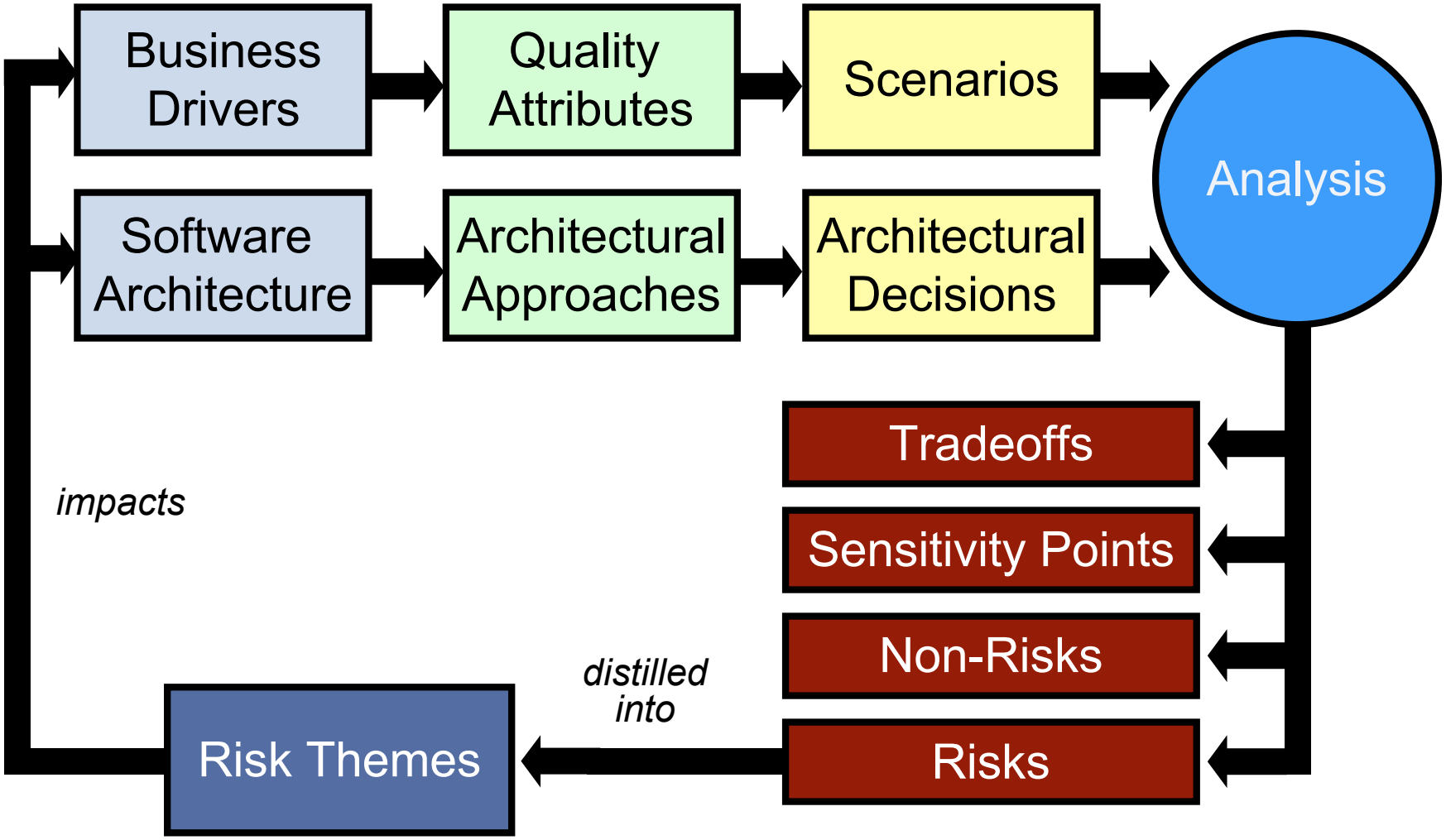
Can you recover an architecture from an existing system?

How do you evolve a software architecture to continue to meet business and product goals?

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*
- evolving the architecture so that it **continues to meet business and product goals**
  - *Architecture Improvement Workshop (AIW) and ArchE*

Conceptual Flow of the ATAM®



Characteristics of SEI Methods

QAW
ADD
ArchE
Views and Beyond
ATAM
CBAM
ARMIN
AIW

- 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

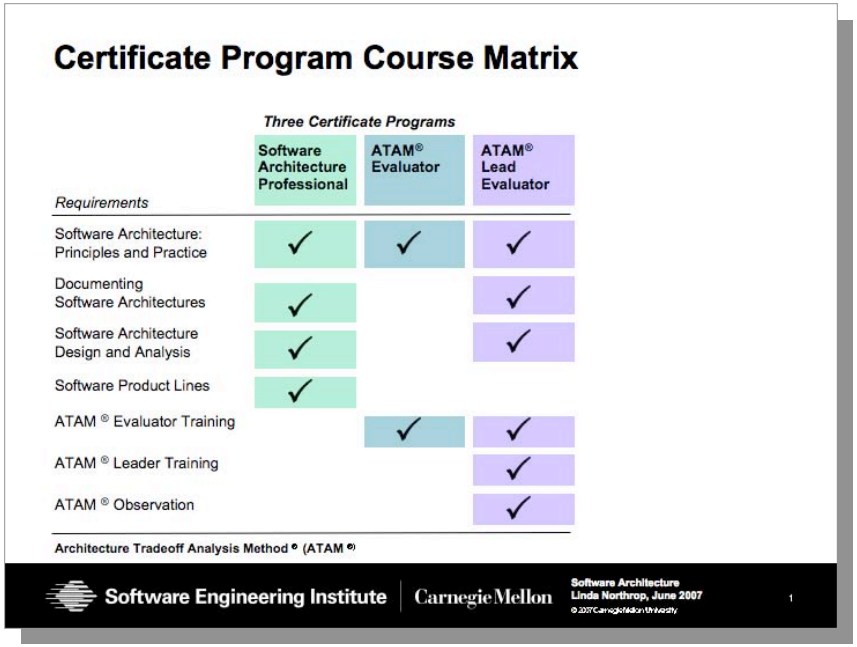
Organizations big and small are recognizing the importance of software architecture. For example,

- 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.



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 depends on disciplined architecture practices.

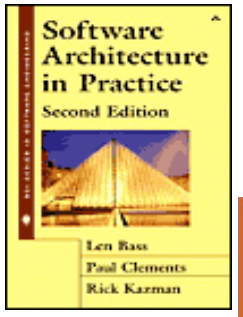
References

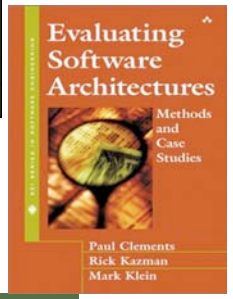
Software Architecture in Practice, Second Edition
Bass, L.; Clements, P.; & Kazman, R. Reading, MA: Addison-Wesley, 2003.

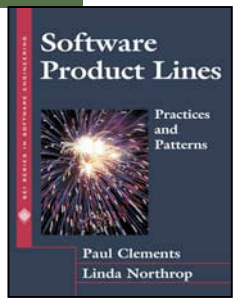
Evaluating Software Architectures: Methods and Case Studies
Clements, P.; Kazman, R.; & Klein, M. Reading, MA: Addison- Wesley, 2002.

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.







Credit To The SEI Software Architecture Team

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

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

For More Information

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

U.S. Mail:
Software Engineering Institute
Carnegie Mellon University
4500 Fifth Avenue
Pittsburgh, PA 15213-3890

World Wide Web:
http://www.sei.cmu.edu/pacc

SEI Fax:
412-268-5758