Carnegie Mellon
Software Engineering Institute

The Importance of Software Architecture

Linda Northrop
Director, Product Line Systems

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

This work is sponsored by the U.S. Department of Defense.

Focus: Software Architecture

Software is pervasive in today's systems and business operations.

"The only thing you can do with an F-22 that does not require software is take a picture of it" ~ Lt. Gen Fain

Software is the root cause of most of today's system problems.

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

What Is Software Architecture?

Typically software architecture includes ad hoc box-and-line drawing(s) of the system that is intended to solve the problems articulated by the specification.

- Boxes define the elements or “parts” of the system.
- Lines define the interactions or the between parts.

A software architecture is a “first cut” at solving the problem and designing the system.

The Definition of Software Architecture

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

Bass L.; Clements P.; Kazman R. *Software Architecture in Practice 2nd Edition* Reading, MA: Addison-Wesley, 2003.

Implications of Our Definition - 1

Architecture is an abstraction of a system:

- Architecture defines the system elements and how they interact.
- Architecture suppresses purely local information about the elements; private details of the elements are not architectural.

Defines the properties of components

- Properties of components are assumptions that one component can make about another:
  - provided services, required services, performance characteristics, fault handling, resource usage

Implications of Our Definition - 2

Every system *has* an architecture.

- Every system is composed of elements and there are relationships among them.
- In the simplest case, a system is composed of a single element, related only to itself.

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!*

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.

Functional Requirements

Functional requirements define

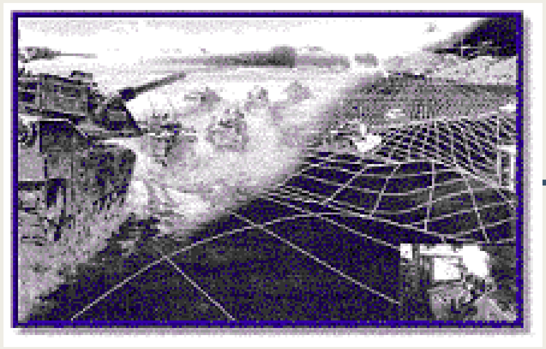
- what the system must do
- how components will interact, cooperate, and synchronize correctly
- what it means to “function” properly

Functionality is largely orthogonal to the structure.

- how modifiable is a system that is functioning properly?

Essential quality attributes are often overlooked or omitted from functional specifications and system descriptions.

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

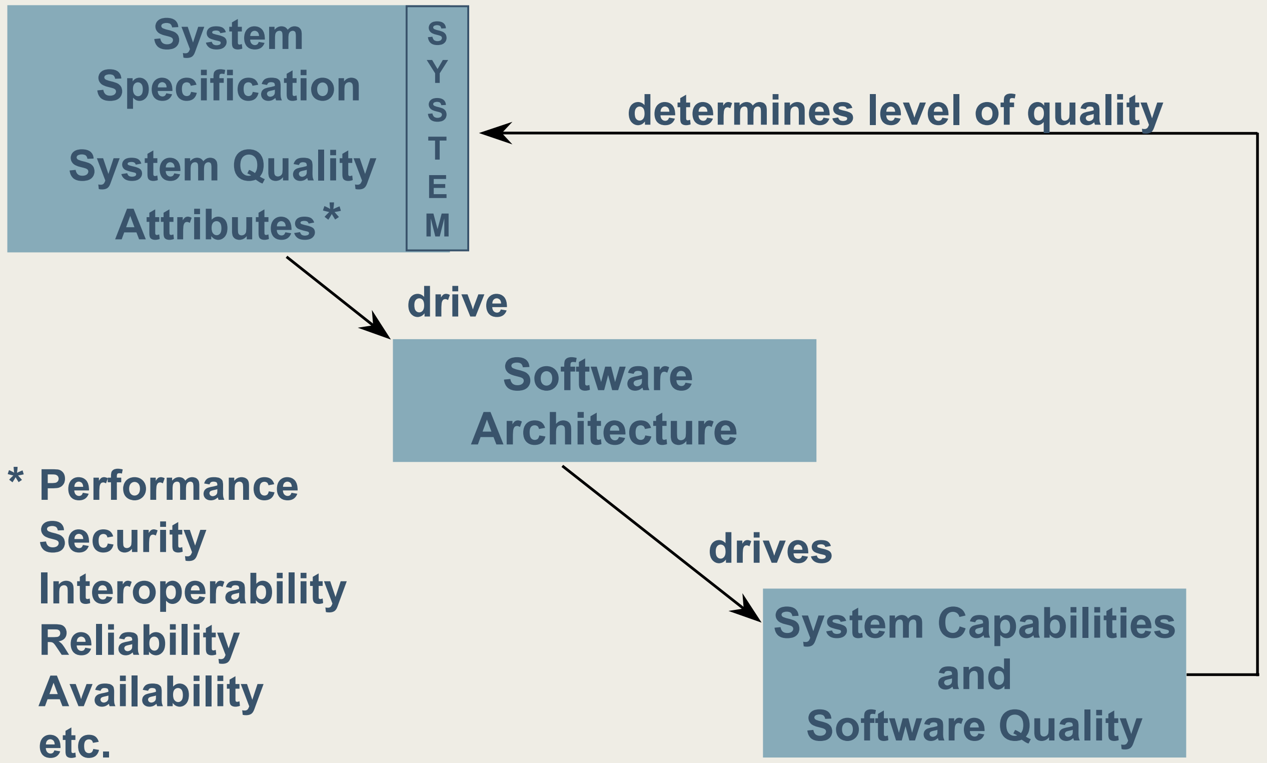
:

analysis, design, development

Quality Attribute Drivers → Software Architecture → Software

has these qualities

System Qualities and Software Architecture


System Specification
System Quality Attributes *
SYSTEM
determines level of quality
drive
Software Architecture
drives
System Capabilities and Software Quality

* Performance
Security
Interoperability
Reliability
Availability
etc.

Challenges

What precisely do these quality attributes such as modifiability, security, performance, and reliability mean?

Can a system be analyzed to determine these desired qualities?

How soon can such an analysis occur?

How do you know if software architecture for a system is suitable without having to build the system first?

How do you document an architecture?

Can you recover an architecture from an existing system?

Common Impediments to Achieving Architectural Success

Lack of adequate architectural talent and/or experience.
Insufficient time spent on architectural design and analysis.
Failure to identify the quality drivers and design for them.
Failure to properly document and communicate the architecture.
Failure to evaluate the architecture beyond the mandatory government review.
Failure to understand that standards are not a substitute for a software architecture.
Failure to ensure that the architecture directs the implementation.
Failure to evolve the architecture and maintain documentation that is current.
Failure to understand that a software architecture does not come free with COTS or with the C4ISR Framework.

What Is Architecture-Based Development?

Architecture-based development involves

- understanding the domain requirements
- developing or selecting the software architecture
- representing and communicating the architecture
- analyzing or evaluating the architecture for ability to satisfy requirements
- organizing the work products around the architecture
- implementing the system based the architecture
- ensuring that the implementation conforms to the architecture
- maintaining the architecture

The architecture must be both prescriptive and descriptive.

Needs, Practices, and Requirements

Early risk mitigation in system life cycle

Architectural insights into legacy systems

Early risk mitigation in system modernization and evolution

Identification and characterization of quality drivers

Improved architecture definition

Qualitative architecture evaluation

Improved architecture documentation

Architectural views of legacy systems

Improved skills and knowledge base in software architecture practices

Improved acquisition practices that capitalize on an architecture-centric approach

Higher quality systems that are built predictably

SEI Enablers

Early risk mitigation in system life cycle

Architectural insights into legacy systems

Early risk mitigation in system modernization and evolution

Identification and characterization of quality drivers **QAW**

Improved architecture definition **ADD**

Qualitative architecture evaluation **ATAM**^{SM}

Improved architecture documentation **Documentation Guidelines**

Architectural views of legacy systems **Architecture Reconstruction**

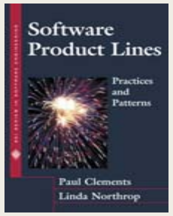
Improved skills and knowledge base in software architecture practices **Books and courses**

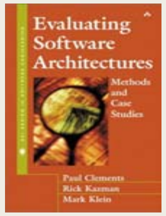
Improved acquisition practices that capitalize on an architecture-centric approach **Guidelines and courses**

Higher quality systems that are built predictably

Carnegie Mellon
Software Engineering Institute

SEI Architecture Technology Summary





Courses and Books

Quality Attribute Workshop (QAW)

Attribute-Driven Design (ADD)

Architecture Documentation Guidelines

Architecture Tradeoff Analysis Method^{SM} (ATAM^{SM})

Architecture Reconstruction (Dali)

Acquisition Guidelines

Contact 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
Pittsburgh, PA 15213-3890

World Wide Web:
http://www.sei.cmu.edu/ata
http://www.sei.cmu.edu/plp

SEI Fax: 412-268-5758