Architecture and Architecture Views

James Ivers and Mario Benitez

September 13, 2023

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

[Distribution Statement A] Approved for public release and unlimited distribution.

Copyright 2023 Carnegie Mellon University.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.

Please see Copyright notice for non-US Government use and distribution.
This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

Architecture Tradeoff Analysis Method® and ATAM® are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM23-0943

Architecture – A Few Definitions

"The software architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both."

- Clements, et al. *Documenting Software Architectures: Views and Beyond* (2nd edition), Addison-Wesley, 2010.

"All architecture is design but not all design is architecture. Architecture represents the significant design decisions that shape a system, where significant is measured by cost of change."

- Grady Booch

Extensive list of definitions at http://www.sei.cmu.edu/architecture/definitions.html

© 2020 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

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

Key to system ***evolution***

- manage future uncertainty
- assure cost-effective agility

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

Architecture aids discussion of many **concerns** with many **stakeholders**

- Guiding development
- Gauging the difficulty of potential changes
- Identifying stress points for testing
- Anticipating deployment challenges
- Onboarding new team members
- …

© 2020 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

Quality Attributes

Quality attributes

- properties of work products or goods by which stakeholders judge their quality
- stem from business and mission goals
- need to be characterized in a measurable, system-specific way

Quality attributes include

- Performance
- Availability
- Security
- Deployability
- Usability
- Scalability
- Modifiability
- Interoperability
- Portability
- ...

© 2020 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

Important Design Decisions

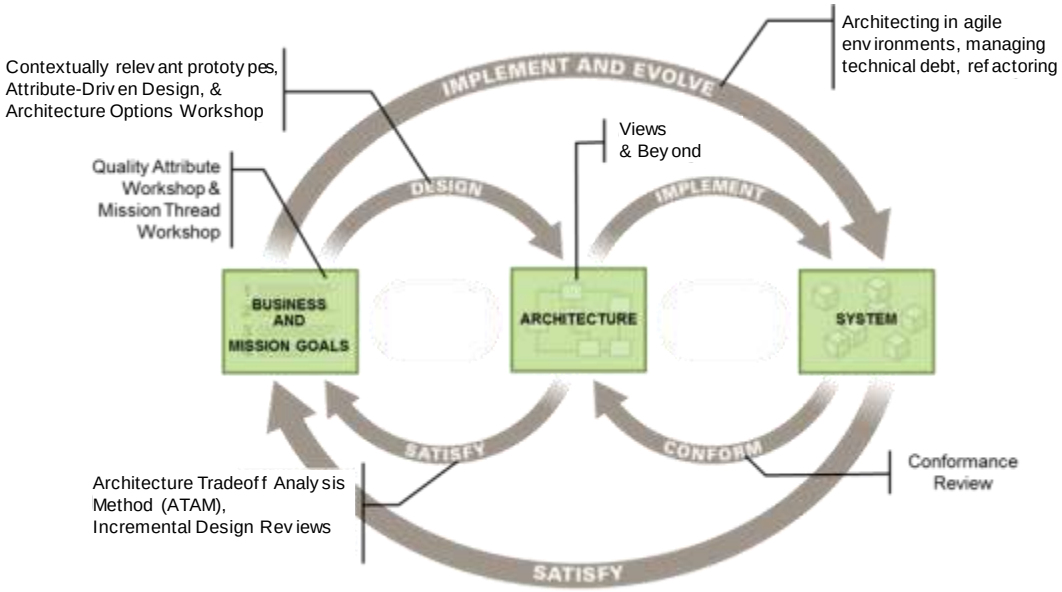
Architecture permits/precludes the *achievement of a system's desired quality attributes*. The strategies for achieving them are architectural.

If you want…	you need to pay attention to…
high performance	minimizing the frequency and volume of inter-element communication
modifiability	limiting interactions between elements
security	managing and protecting inter-element communication
reusability	minimizing inter-element dependencies
subsetability	controlling the dependencies between subsets and, in particular, avoiding circular dependencies
availability	the properties and behaviors that elements must have and the mechanisms you will employ to address fault detection, fault prevention, and fault recovery
and so on	…

© 2020 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

Establishing a Discipline of Software Architecture



Range of methods and practices applicable at different points in the development lifecycle

- Domains of expertise include IT, C2, tactical, and health informatics
- Technology expertise includes IoT, big data, digital twin, cloud, and machine learning

8+ courses, available in a mix of public, on-site, and eLearning options

2 professional certificates

A collection of books for wide dissemination

© 2020 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

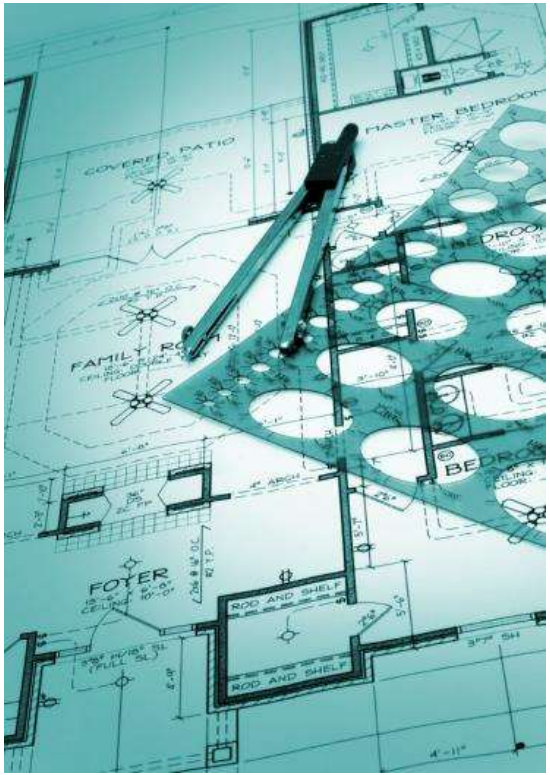
Why Document an Architecture?

Every system has an architecture, intentional or accidental, documented or not.

Architecture documentation has three fundamental uses.

- **Education**, introducing people to the system: new members of the team, external analysts, the customer, or even a new architect.
- **Communication** vehicle among stakeholders and to/from the architect.
- **Analysis**, especially for the quality attributes that the architecture is designed to provide.

An architecture document should be helpful to the people who depend on it to do their work.



© 2020 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

Architecture Views

An architecture is a multidimensional construct, too involved to be seen all at once. Systems comprise many structures that show

- how components and connectors work at runtime
- processes and how they synchronize
- programs and how they call or send data to each other
- composition/decomposition of modules
- mapping of modules to implementation units
- how software is deployed on hardware
- how teams cooperate to build the system
- and much more

Views helps us manage complexity.

© 2020 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

A Short History of Views

- 1974: Parnas observed that software is composed of many structures.
- 1992: Perry and Wolf recognized that, as for buildings, different views of a system are required.
- 1995: Kruchten proposed the “4+1 Views” approach to software architecture.
- 1995: Hofmeister, Nord, and Soni defined the “Siemens Four Views” approach to software.
- 2000: The ANSI/IEEE 1471-2000[1] standard prescribed a view-based approach to architecture description.
- 2003: SEI published *Documenting Software Architectures* (1st Edition), providing descriptions and examples of 13 common views.

[1] This standard was later adopted as ISO/IEC/IEEE 42010.

© 2020 Carnegie Mellon University
[Distribution Statement A] Approved for public release and unlimited distribution.

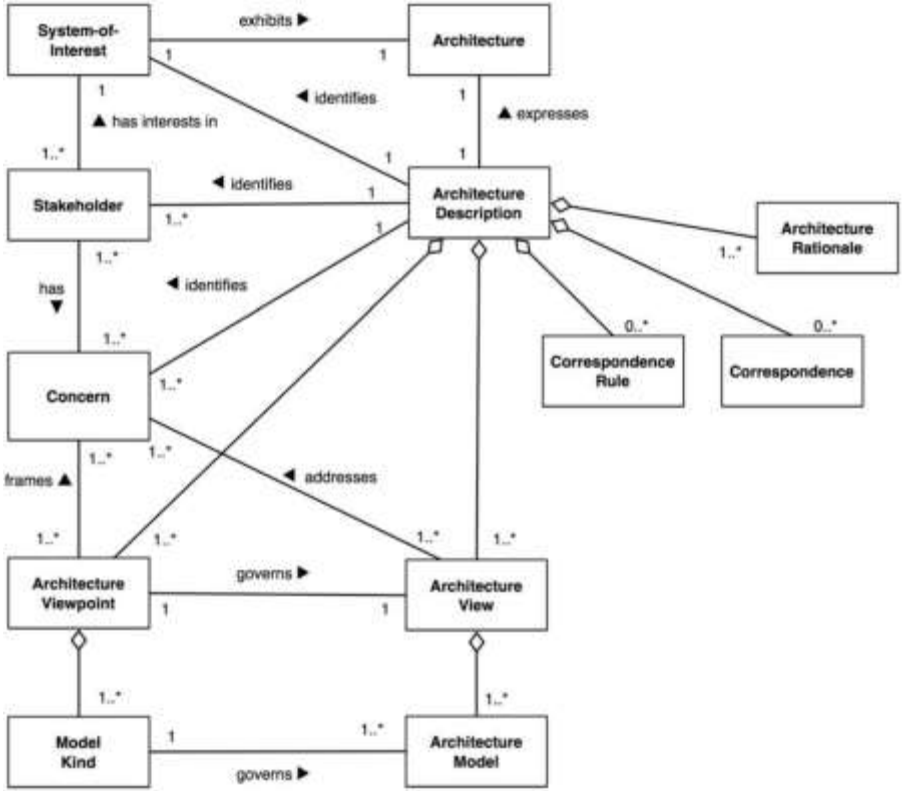
Which Views???

Key keys ideas that emerged from this era

- Choose the best views for each situation.
- Which views are “right” depends on
  - the structures that are inherent in the software
  - who the stakeholders are and how they will use the documentation (their concerns)

An important corollary

- Only document what you need.



Source: ISO/IEC/IEEE 42010.

© 2020 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

Information in a View

A view's primary function is to show the structure that it represents.

Hence, a view will show the **elements** that are in the structure and the **relationships** among them.

A view will also explain what the elements are, what their responsibilities are, and what their important properties are.

Properties are values used to describe the elements to help convey understanding and aid analysis. Properties are often quality-of-service values, such as an element's performance or reliability characteristics.

Architects can choose the properties they wish to document based on what the document's readers will wish to know.

© 2020 Carnegie Mellon University

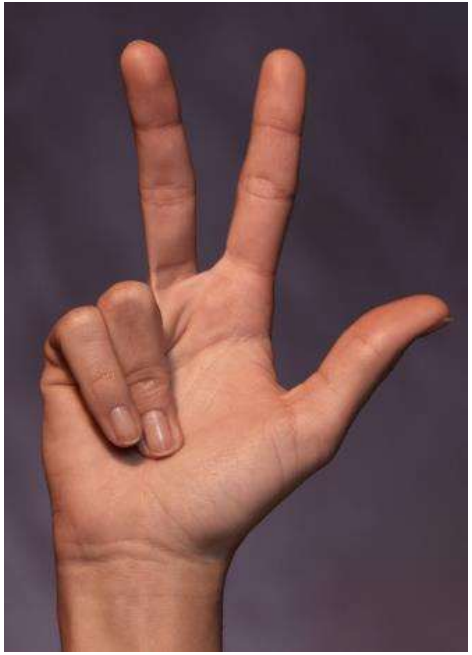
[Distribution Statement A] Approved for public release and unlimited distribution.

Three Types of Views

We commonly observe three kinds of views, each of which shows different kinds of information:

1. **Module views** show how the system is structured as a set of code units.
2. **Component-and-connector views** show how the system is structured as a set of elements with runtime behaviors and interactions.
3. **Allocation views** show how the system relates to non-software structures in its environment.

Every view contains information from at least one of these categories.



© 2020 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

Module Views

Elements: modules. A module is a code unit that implements a set of responsibilities.

Relations: Relations among modules include

- A is part of B. This defines a part-whole relation among modules.
- A depends on B. This defines a dependency relation among modules.
- A is a B. This defines specialization and generalization relations among modules.

Properties: Properties of a module usually include a name, responsibilities, and the visibility of the module and its interface.

These views focus on the organization of code.

What are they good for?

- Development blueprints
- Reasoning about the development-oriented quality attributes
  - Modifiability
  - Maintainability
  - Portability
  - Reusability
- Project management, budgeting, planning, and tracking.

© 2020 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

Component-and-Connector (C&C) Views

Elements: components and connectors

- Components are principal units of runtime interaction and data stores.
- Connectors are interaction mechanisms.

Relations: attachment of components' ports to connectors' roles (interfaces with protocols)

Properties: Properties of a component or connector often include a name and runtime quality-of-service information to facilitate analysis or prediction of runtime quality attributes.

These views focus on how the software operates at run-time.

What are they good for?

- End-to-end behavioral walk throughs
- Reasoning about the run-time-oriented quality attributes
  - Performance
  - Reliability
  - Security

© 2020 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

Allocation Views

Elements:

- software elements (as defined in module or C&C styles)
- environment elements

Relations: Software elements are "allocated to" environment elements.

Properties: Properties documented in an allocation view are "requires" properties of software elements and "provides" properties of the environment elements. If the properties are compatible, then the allocation is a sound one.

These views focus on how software (modules or C&C) are allocated to environments.

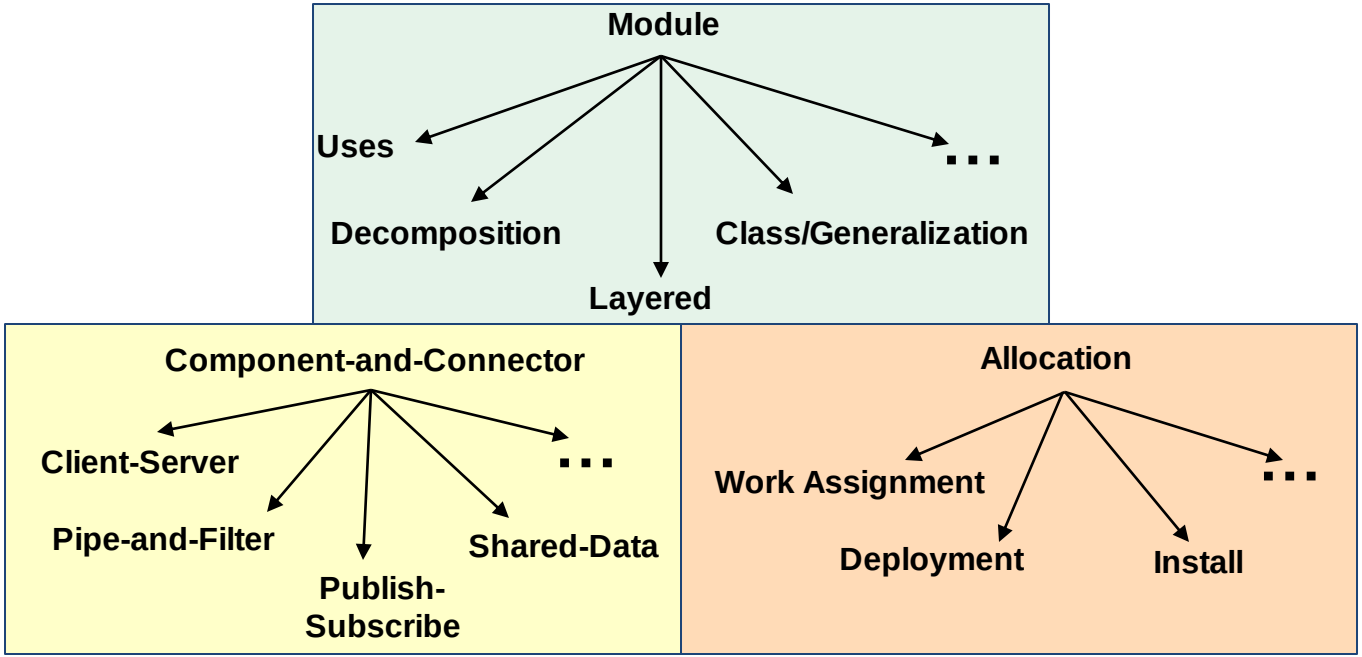
What are they good for?

- Binding elements to specific resources (e.g., hosts or containers)
- Reasoning about quality attributes dependent on resource allocations
  - Scalability
  - Performance

© 2020 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

Summary of Views in DSA



Architects should focus on whatever views will provide them with the most leverage in achieving the desired quality attributes of a system.

© 2020 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.