Bass, Clements, and Kazman. Software Architecture in Practice, Addison-Wesley 1997:

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

By "externally visible" properties, we are referring to those assumptions other components can make of a component, such as its provided services, performance characteristics, fault handling, shared resource usage, and so on. The intent of this definition is that a software architecture must abstract away some information from the system (otherwise there is no point looking at the architecture, we are simply viewing the entire system) and yet provide enough information to be a basis for analysis, decision making, and hence risk reduction.

Let's look at some of the implications of this definition in more detail.

First, architecture defines components. The architecture embodies information about how the components interact with each other. This means that architecture specifically omits content information about components that does not pertain to their interaction.

Second, the definition makes clear that systems can comprise more than one structure, and that no one structure holds the irrefutable claim to being the architecture. By intention, the definition does not specify what architectural components and relationships are. Is a software component an object? A process? A library? A database? A commercial product? It can be any of these things and more.

Third, the definition implies that every software system has an architecture, because every system can be shown to be composed of components and relations among them.

Fourth, the behavior of each component is part of the architecture, insofar as that behavior can be observed or discerned from the point of view of another component. This behavior is what allows components to interact with each other, which is clearly part of the architecture. Hence, most of the box-and-line drawings that are passed off as architectures are in fact not architectures at all. They are simply box-and-line drawings.

Architecture defined (yet again)

Mary Shaw, CMU
Grady Booch,
Philippe Kruchten,
Rich Reitman
Kurt Bittner, Rational

- Software architecture encompasses the set of significant decisions about the organization of a software system
  - selection of the structural elements and their interfaces by which a system is composed
  - behavior as specified in collaborations among those elements
  - composition of these structural and behavioral elements into larger subsystem
  - architectural style that guides this organization

RATIONAL SOFTWARE

Architecture defined (continued)

Mary Shaw, CMU
Grady Booch,
Philippe Kruchten,
Rich Reitman
Kurt Bittner, Rational

- Software architecture also involves
  - usage
  - functionality
  - performance
  - resilience
  - reuse
  - comprehensibility
  - economic and technology constraints and tradeoffs
  - aesthetic concerns

RATIONAL SOFTWARE

Architectural style

- An architecture style defines a family of systems in terms of a pattern of structural organization.
- An architectural style defines
  - a vocabulary of components and connector types
  - a set of constraints on how they can be combined
  - one or more semantic models that specify how a system's overall properties can be determined from the properties of its parts

RATIONAL SOFTWARE

Architecture metamodel

RATIONAL SOFTWARE

Representing System Architecture

26

RATIONAL SOFTWARE

Models, Views, and Diagrams

A ***model*** is a complete description of a system from a particular perspective

38

RATIONAL SOFTWARE

Component Diagram

- Captures the physical structure of the implementation

RATIONAL SOFTWARE

Architectural design workflow

- Select scenarios: criticality and risk
- Identify main classes and their responsibility
- Distribute behavior on classes
- Structure in subsystems, layers, define interfaces
- Define distribution and concurrency
- Implement architectural prototype
- Derive tests from use cases
- Evaluate architecture

Iterate

Use case view

Logical view

Implementation view

Deployment view

Process view

70

RATIONAL SOFTWARE

Deployment Diagram

- Captures the topology of a system's hardware

RATIONAL SOFTWARE