approved, and then serve to constrain subsequent decision making and construction. For software systems, these design decisions are behavioral and structural.

External behavioral descriptions show how the product will interface with its users, other systems, and external devices, and should take the form of requirements. Structural descriptions show how the product is divided into parts and the relations between those parts. Internal behavioral descriptions are needed to describe the interfaces between components. Structural descriptions often show several distinct views of the same part because it is impossible to put all the information in one drawing or document in a meaningful way. A component in one view may be a part of a component in another.

Software architectures are often presented as layered hierarchies that tend to commingle several different structures in one diagram. In the 1970s Parnas pointed out that the term "hierarchy" had become a buzzword, and then precisely defined the term and gave several different examples of structures used for different purposes in the design of different systems (Parnas 1974). Describing the structures of an architecture as a set of *views*, each of which addresses different concerns, is now accepted as a standard architecture practice (Clements et al. 2003; IEEE 2000). We will use the word "architecture" to refer to a set of annotated diagrams and functional descriptions that specify the structures used to design and construct a system. In the software development community there are many different forms used, and proposed, for such diagrams and descriptions. See Hoffman and Weiss (2000, chaps. 14 and 16) for some examples.

> **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.**
>
> **"Externally visible" properties are those assumptions other elements can make of an element, such as its provided services, performance characteristics, fault handling, shared resource usage, and so on.**
>
> —*Len Bass, Paul Clements, and Rick Kazman,* Software Architecture in Practice, *Second Edition*

Architecture Versus Design

Architecture is a part of the design of the system; it highlights some details by abstracting away from others. Architecture is thus a subset of design. A developer focused on implementing a component of the system may not be very aware of how all the components fit together, but rather is primarily concerned with the design and development of a small number of component(s), including the architectural constraints that they must obey and the rules they can use. As such, the developer is working on a different aspect of the system design than the architect.