1. Important Points
   1. “Architectural descriptions tend to be informal and idiosyncratic: box-and-line diagrams convey essential system structure, with accompanying prose explaining the meaning of the symbols.” [249]
   2. “Much of [architectural description’s] power comes from use of idiomatic architectural terms, such as ‘client-server system,’ ‘layered system,’ or ‘blackboard organization.’” [249]
   3. “In its simplest form, object-oriented design lets us encapsulate data and behavior in discrete objects that provide explicit interfaces to other objects.” [250]
   4. “[Object-oriented design] is not by itself well suited to describing complex interactions between groups of objects.” [250]
   5. “Design patterns have become an increasingly popular choice for addressing OOD’s limitations.” [250]
   6. “Architectural design fulfills two primary roles. First, it provides a level of abstraction at which software system designers can reason about system behavior: function, performance, reliability and so on.” [250] “Second, an architectural design serves as the ‘conscience’ for a system as it evolves.” [250]
   7. “Architectural descriptions are primarily concerned with the following basic issues: System structure,” [250] “rich abstractions for interaction,” [250] and “global properties.” [250]
   8. “An architectural style characterizes a family of systems that are related by shared structural and semantic properties.” [251]
   9. “Styles typically provide the following four things: A vocabulary of design elements,” [251] design rules or constraints,” [251] “semantic interpretation,” [251] and “analysis that can be performed on systems built in that style.” [251]
   10. “[Architectural style] promotes design reuse,” [251] “can lead to significant code reuse,” [251] “is easier for others to understand,” [251] “supports interoperability,” [251] and “permits specialized, style-specific analysis.” [251]
   11. “Software architecture concepts allow an architect to describe multiple, rich interfaces to a component and to describe and encapsulate complex protocols of component interaction that are difficult to describe using traditional object-oriented concepts and notation.” [252]
   12. “The pattern approach lets us describe relatively complex protocols of interactions between objects that we want to encapsulate, but don’t want to encapsulate within a single class.” [254]
   13. “Architectural design is concerned with composing systems from components, and the interactions between these components.” [255]
   14. “The basic idea behind design patterns is that common idioms are found repeatedly in software designs and that these patterns should be made explicit, codified, and applied appropriately to similar problems.” [255]
   15. “There are three fundamental requirements for specifying and reusing software design patterns: the design domain must be well understood, it must support the encapsulation of design elements, and it must have evolved a collection of well-known and proven design idioms.” [256]
   16. “We see patterns and architectural styles as complementary mechanisms for encapsulating design expertise.” [256]
   17. “Architectural design patterns and object-oriented design patterns are simply instances of the more general class of all design patterns.” [257]
   18. “A specific architectural style is better thought of as a language for building patterns than as an instance of a design pattern itself.” [258]
2. Disagreements
   1. “Software architecture concepts allow an architect to describe multiple, rich interfaces to a component and to describe and encapsulate complex protocols of component interaction that are difficult to describe using traditional object-oriented concepts and notation.” [252]

Above, the authors state that encapsulating complex protocols using traditional object-oriented concepts and notation is difficult. I disagree with their idea. Object-oriented design patterns are a traditional object-oriented concept. As such, complex component interaction can easily be described using traditional object-oriented concepts.

1. Questions and Clarifications
   1. Is Wright notation common? [252]
   2. What is an OMT-based design? [253]
   3. What is the difference between a first-class and second-class object? [253]