David Lambertson

**Architectural Styles, Design Patterns and Objects Summary**

**Important Points from Article**

* “Architectural design illuminates such key issues as scaling and portability, the assignment of functionality to design elements, interaction protocols between elements, and global system properties such as processing rates, end-to-end capacities, and overall performance.” [249]
* “The object-oriented paradigm offers another approach to describing system designs.” [250]
* “In practice, an 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]
* “Although interactions may be as simple as procedure calls shared data variables, they often represent more complex forms.” [250]
* “Architectural designs typically describe overall system behavior.” [250]
* “For example, a system architecture might be defined informally as a client-server system, a blackboard system, a pipeline, an interpreter, or a layered system.” [251]
* “An architectural style provides a specialized design language for a specific class of systems.” [251]
* “The use of architectural styles has a number of significant benefits. First, it promotes design reuse… Second, it can lead to significant code reuse… Third, it is easier for others to understand a system’s organizations if conventionalized structures are used.” [251]
* “Fourth, use of standardized styles supports interoperability.” [251]
* “Fifth, as we noted earlier, by constraining the design space, an architectural style often permits specialized, style-specific analyses.” [251]
* “Although OOD concepts can be used to address some architectural design issues, and doing so is popular among software developers, there are significant differences between the capabilities and benefits of object-oriented approaches provided by an emerging class of software architecture design tools and notations.” [252]
* “The Wright style specification describes the semantics of the design elements that can be used in the style (pipes and filters), along with a set of constraints that specify how the design elements can be composed when building systems in the pipe-and-filter style.” [252]
* “As a result, it is difficult for an architectural object to limit the services it can provide based on which aspects of the interface a requester is using and the type of connection between the two objects.” [253-254]
* “One effect of placing a pipe entity between two filters is that the upstream filter no longer knows which downstream filter is receiving and processing its data.” [254]
* “An object-oriented approach to specifying an architectural pipe connector for use in pipe-and-filter style systems, along with rules for how a pipe can be properly instantiated in a design, apparently will require the cooperation of multiple objects.” [254]
* “By itself, however, this design does not completely mitigate the problem of access to inappropriate methods.” [254]
* “As these examples illustrate, architectural designs involve abstractions that may not necessarily be best modeled as a system of objects, at least in the narrow sense of objects as encapsulated data types that interact through method invocation.” [255]
* “Examples include ways of modeling problem domains and requirements, and implementation issues such as designing data structures and algorithms.” [255]
* “Two of the primary limitations of traditional OOD, as described in the previous examples, are the difficulty in specifying how groups of objects interact and in specifying and packaging related collections of objects for reuse.” [255]
* “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]
* “Thus, architectural design patterns and object-oriented design patterns are simply instances of the more general class of all design patterns.” [257]
* “Architectures, architectural styles, objects, and design patterns capture complementary aspects of software design.” [258]

**Things I Didn't Agree With**

“The use of architectural styles has a number of significant benefits. First, it promotes design reuse: routine solutions with well-understood properties can be reapplied to new problems with confidence.” [251]

I do not agree with this because it will not always be reapplied to new problems with confidence. It may apply confidently in most cases, but there will always those problems you have never had to deal with or have never seen and it just will not apply well. These are the problems we need to watch out for the most.

**Things I Did Not Understand**

I understood it all.