**Checklist: Architecture**

Here’s a list of issues that a good architecture should address. The list isn’t intended to be a comprehensive guide to architecture but to be a pragmatic way of evaluating the nutritional content of what you get at the programmer’s end of the software food chain. Use this checklist as a starting point for your own checklist. As with the requirements checklist, if you’re working on an informal project, you’ll find some items that you don’t even need to think about. If you’re working on a larger project, most of the items will be useful.

**Specific Architectural Topics**

❑ Is the overall organization of the program clear, including a good architectural overview and justification?

❑ Are major building blocks well defined, including their areas of responsibility and their interfaces to other building blocks?

❑ Are all the functions listed in the requirements covered sensibly, by neither too many nor too few building blocks?

❑ Are the most critical classes described and justified?

❑ Is the data design described and justified?

❑ Is the database organization and content specified?

❑ Are all key business rules identified and their impact on the system described?

❑ Is a strategy for the user interface design described?

❑ Is the user interface modularized so that changes in it won’t affect the rest of the program?

❑ Is a strategy for handling I/O described and justified?

❑ Are resource-use estimates and a strategy for resource management described and justified for scarce resources like threads, database connections, handles, network bandwidth, and so on?

❑ Are the architecture’s security requirements described?

❑ Does the architecture set space and speed budgets for each class, subsystem, or functionality area?

❑ Does the architecture describe how scalability will be achieved?

❑ Does the architecture address interoperability?

❑ Is a strategy for internationalization/localization described?

❑ Is a coherent error-handling strategy provided?

❑ Is the approach to fault tolerance defined (if any is needed)?

❑ Has technical feasibility of all parts of the system been established?

❑ Is an approach to overengineering specified?

❑ Are necessary buy-vs.-build decisions included?

❑ Does the architecture describe how reused code will be made to conform to other architectural objectives?

❑ Is the architecture designed to accommodate likely changes?

**General Architectural Quality**

❑ Does the architecture account for all the requirements?

❑ Is any part overarchitected or underarchitected? Are expectations in this area set out explicitly?

❑ Does the whole architecture hang together conceptually?

❑ Is the top-level design independent of the machine and language that will be used to implement it?

❑ Are the motivations for all major decisions provided?

❑ Are you, as a programmer who will implement the system, comfortable with the architecture?