

*Borrowed from the companion website to Code Complete 2nd Ed. by Steve McConnell at <http://www.cc2e.com/0323>*

## **Requirements Checklist**

The requirements checklist contains a list of questions to ask yourself about your project's requirements. This book doesn't tell you how to do good requirements development, and the list won't tell you how to do one either. Use the list as a sanity check at construction time to determine how solid the ground that you're standing on is—where you are on the requirements Richter scale.

Not all of the checklist questions will apply to your project. If you're working on an informal project, you'll find some that you don't even need to think about. You'll find others that you need to think about but don't need to answer for-mally. If you're working on a large, formal project, however, you may need to consider every one.

---

### **Specific Functional Requirements**

- Are all the inputs to the system specified, including their source, accuracy, range of values, and frequency?
- Are all the outputs from the system specified, including their destination, accuracy, range of values, frequency, and format?
- Are all output formats specified for web pages, reports, and so on?
- Are all the external hardware and software interfaces specified?
- Are all the external communication interfaces specified, including handshaking, error-checking, and communication protocols?
- Are all the tasks the user wants to perform specified?
- Is the data used in each task and the data resulting from each task specified?

### **Specific Non-Functional (Quality) Requirements**

- Is the expected response time, from the user's point of view, specified for all necessary operations?
- Are other timing considerations specified, such as processing time, data-transfer rate, and system throughput?
- Is the level of security specified?
- Is the reliability specified, including the consequences of software failure,

the vital information that needs to be protected from failure, and the strategy for error detection and recovery?

- Is maximum memory specified?
- Is the maximum storage specified?
- Is the maintainability of the system specified, including its ability to adapt to changes in specific functionality, changes in the operating environment, and changes in its interfaces with other software?
- Is the definition of success included? Of failure?

### **Requirements Quality**

- Are the requirements written in the user's language? Do the users think so?
- Does each requirement avoid conflicts with other requirements?
- Are acceptable trade-offs between competing attributes specified-for example, between robustness and correctness?
- Do the requirements avoid specifying the design?
- Are the requirements at a fairly consistent level of detail? Should any requirement be specified in more detail?
- Should any requirement be specified in less detail?
- Are the requirements clear enough to be turned over to an independent group for construction and still be understood?
- Is each item relevant to the problem and its solution? Can each item be traced to its origin in the problem environment?
- Is each requirement testable? Will it be possible for independent testing to determine whether each requirement has been satisfied?
- Are all possible changes to the requirements specified, including the likelihood of each change?

### **Requirements Completeness**

- Where information isn't available before development begins, are the areas of incompleteness specified?
- Are the requirements complete in the sense that if the product satisfies every requirement, it will be acceptable?
- Are you comfortable with all the requirements? Have you eliminated requirements that are impossible to implement and included just to appease your customer or your boss?