Requirements vs Specification

* Requirements:
  + Intended for customers in addition to developers
  + Informal descriptions necessary
* Specifcation:
  + For use by developers
  + Formal descriptions necessary

Interface specification

* Contract between component users (clients) and developers (implementers)
* Typically describes demands on users and responsibilities for implementers
* Should present essentials in “user-oriented” terms (abstraction) and hide inessentials (information hiding)

Informal specification

* Straightforward descriptions
* Metaphors
* Implementation details
* Problems:
  + Contradiction
  + Noise
  + Lack of clarity

Formal Interface Specification:

* Communicates precise demands and responsibilities
* Allows independent development of client and implementation components in parallel in team environment
* Minimizes integration costs

Specification Language

* Sone languages design for particular programming languages, some general purpose
* Some integrated with programming constructs
* Few integrate ability to perform formal mathematical reasoning

Mathematical reasoning

* Goal: Proves correctness
* Manual or mechanical
* Consequences:
  + Provides correctness on all valid inputs
  + Can show absence of bugs

Seven sins of specifier

1. Noise
   * Presence in text of element that doesn’t carry relevant information
2. Silence
   * Existence of feature not covered by text
3. Over specification
   * Presence in text that relates to feature of problem not to feature of solution
4. Contradiction
   * Two or more text elements that define feature incompatibly
5. Ambiguity
   * Text that can be interpreted in at least two different ways
6. Forward reference
   * Uses features of problem not yet defined
7. Wishful thinking
   * Text that defines feature of problem in way that a candidate solution cannot realistically be validated with respect to feature