Write Well!

“The intended readers are the system designers, team manager, and, once validated, the programmer.”

Use the active voice.

Bad: “The complex arithmetic expression is broken down and evaluated as a series of precedence-ordered simple arithmetic expressions.”

Good: The Complex Expression Evaluator module breaks down a complex arithmetic expressions into a sequence of precedence-ordered simple arithmetic expressions.

Better: The Complex Expression Evaluator module accepts a complex arithmetic expression as input and returns as output a sequence of precedence-ordered simple arithmetic expressions equivalent to the input expression.

Avoid Programming

Bad: “On encountering a ‘)’ or eos, Get_Char_Main2 calls Pop; this indicates the close of a higher-order operation. In addition, after popping the stack, Pop calls Get_Char_Main2 recursively, passing it what was just popped off the stack. Pop also handles eos, and repeatedly pops the stack until the stack is empty.”

Good: Simplify accepts a string containing a binary expression and returns three strings: the left operand, the operator, and the right operand of the input expression.

But think of the programmer: is the module described with enough detail to implement?

Miscellaneous

Just like in the SRD, if you introduce something in the SDD, you have to define it.

Keep track of the SRD, particularly added requirements.

Concentrate on the architectural and logical design.

Wishful thinking is a powerful design technique.

SRD Structure

Define modules in isolation using component descriptions.

- Purpose: one sentence description of the module's abstraction.
- Function: describe the transformation from input to output.
- Interface: describe the inputs and outputs.
- Logical design.

Use structure charts to tie the modules together and show the abstraction levels.

- Use the existing notation.
- Subordinates are the innards of the superiordinates.
- Architectural design.