Sermon 3: Programming and Documentation Style (Discipline and Craftsmanship)

Question is: how to build a large software system in a reasonable amount of time, and make it work reliably?

Programming is a craft: something like a high-tech form of silversmithing. To make big systems work, it takes a tremendous amount of discipline and structure.

Rule #0: simplicity. It's easy to make things complicated, harder to make them simple. Don't accept complexity.

Rule #1: don't over-generalize. Could start by building code that is so flexible, it can do anything! But then code is 10 times larger, more complex than it needs to be, and you'll still need to modify it, because of things you can't predict.

Rule #2: if it's complex, throw it away, start over.

Rule #3: module testing. Every module should be simple enough to be completely testable on its own.

Rule #4: adopt a consistent style and use it everywhere. Decide on file organization, procedure structuring, naming conventions, location of curly braces, everything.

Rule #5: don't litter. Temptation is to make fast fixes that dirty things up. It's crucial to take the time to fix things right at the beginning. You'll never have time to come back to it later!

Rule #6: document carefully as you go. Don't put this off!

Most important things are interfaces: procedure headers and data structures and other things that tie together the parts of the system.

Rule #7: documentation should describe things at a higher level than code (but not **too** high a level).

Examples of pathological cases:

"Add one to i."

"Now we have it the way we want it."

"This is a hack!"

Rule #8: Put documentation near the code: otherwise you forget to change the documentation when you change the code.

Rule #9: Quality, not quantity. Don't need enormous amounts of documentation if what you have is high-quality.

Summary: discipline, craft. Take time up front to save time later. Be willing to cut your losses.