"A Rational Design Process: How and Why to Fake it" by D. L. Parnas and P. C. Clements

A Rational Design Process: How and Why to Fake it

"Many have sought a software design process that allows a program to be derived systematically from a precise statement of requirements. … although we will not succeed in designing a real product that way, we can produce documentation that makes it appear that the software was designed by such a process …"

D. L. Parnas

RDP - "Faking It"

- The "rational design process" is an irrational ideal.
- **Question:** If we rarely act in a purely top-down way when we develop software, why do most design methods assume we do?
- **Possible answers:**
  - It's simpler than trying to model real-life.
  - Wide variation in problems, possible solutions.
  - There is utility in the *structure* of the process, and its ongoing documentation.

RPD Payoff

The real payoff comes during maintenance and “the next time around”.

The Role of Documentation

- Documentation plays a major role in the development of any large, long-lived software system **BUT** poor documentation is a monumental, ubiquitous problem.
- "Most programmers regard documentation as a necessary evil, written as an afterthought only because some bureaucrat requires it. They do not expect it to be useful."
- This attitude is a self-fulfilling prophecy!

The Role of Documentation (Cont'd)

- While most documents are incomplete and inaccurate, these problems can be fixed easily. More serious problems are:
  - Poor organization.
  - Boring, redundant, verbose prose:
    - Boredom leads to inattentive reading and undiscovered errors.
  - Confusing and inconsistent terminology.
  - "Myopia", can't see the forest through the trees.
    - Focus is on documenting small details rather than on important design decisions.

So What's to be Done?

- Look at the various stages of the "Rational Design Process", and consider the documents that are to be produced at each step.
- Even if you don't follow the steps in that order, go back and fill in the blanks! Pretend that you did follow the process precisely.
- It isn't just a paper trail you're creating.

Programs vs Proofs

- Designing a software system is a lot like proving a mathematical theorem.
- The construction of an original proof is a painful process; you make lots of mistakes, pursue bad paths, ...
- However, once you've figured out how to get there, you clean up the proof and present it as if no mistakes had ever been made.

Programs vs Proofs

- If you want to prove a similar but different theorem, you can re-use ideas from the first proof!
- The main difference is that during software development, you also record **why** you chose each path, what alternatives were considered, and why other paths were not chosen.