A Philosophy of Software Design *by John Ousterhout*

Sheet by Martin Hafskjold Thoresen

Design Principles

Here are the most important software design principles discussed in this book:

1. Complexity is incremental: you have to sweat the small stuff.
2. Working code isn't enough.
3. Make continual small investments to improve system design.
4. Modules should be deep.
5. Interfaces should be designed to make the most common usage as simple as possible.
6. It's more important for a module to have a simple interface than a simple implementation.
7. General-purpose modules are deeper.
8. Separate general-purpose and special-purpose code.
9. Different layers should have different abstractions.
10. Pull complexity downward.
11. Define errors (and special cases) out of existence.
12. Design it twice.
13. Comments should describe things that are not obvious from the code.
14. Software should be designed for ease of reading, not ease of writing.
15. The increments of software development should be abstractions, not features.

Red Flags

Here are a few of of the most important red flags discussed in this book. The presence of any of these symptoms in a system suggests that there is a problem with the system's design:

Shallow Module: The interface for a class or method isn't much simpler than its implementation.

Information Leakage: A design decision is reflected in multiple modules.

Temporal Decomposition: The code structure is based on the order in which operations are executed, not on information hiding.

Overexposure: An API forces callers to be aware of rarely used features in order to use commonly used features.

Pass-Through Method: A method does almost nothing except pass its arguments to another method with a similar signature.

Repetition: A nontrivial piece of code is repeated over and over.

Special-General Mixture: Special-purpose code is not cleanly separated from general purpose code.

Conjoined Methods: Two methods have so many dependencies that its hard to understand the implementation of one without understanding the implementation of the other.

Comment Repeats Code: All of the information in a comment is immediately obvious from the code next to the comment.

Implementation Documentation Contaminates Interface: An interface comment describes implementation details not needed by users of the thing being documented.

Vague Name: The name of a variable or method is so imprecise that it doesn't convey much useful information.

Hard to Pick Name: It is difficult to come up with a precise and intuitive name for an entity.

Hard to Describe: In order to be complete, the documentation for a variable or method must be long.

Nonobvious Code: The behavior or meaning of a piece of code cannot be understood easily.