software.pioniere

David L. Parnas
Department of Computing and Software
McMaster University, Hamilton Ontario, Canada

Modularization by Information Hiding

The Secret History of Information Hiding

The concept of "information-hiding" as a software design principle is widely accepted in academic circles. Many successful designs can be seen as successful applications of abstraction or information hiding. On the other hand, most industrial software developers do not apply the idea and many consider it unrealistic. This talk describes how the idea developed, discusses difficulties in ist application, and speculates on why it has not been completely successful.

A Dutch Adventure

Not interested in modularisation when I went to Holland.

Thought the problem of modularisation was obviously solved.

Industrial software revealed that it was not solved.

It takes deliberate attention to get good software structure.

Phrases like "Clean design" or "Beautiful" are not Meaningful

I found the software at Philips quite ugly.

My manager found that meaningless.

My manager asked me to explain what principle was being violated.

My manager asked me for pragmatic criteria.

I had no answers.

The Napkin of Doom

Compiler and data base experts have lunch.

They exchange a control block format on a napkin.

Napkin is punched, copied, and filed.

Format changes but napkin does not.

Components are coupled and don’t work.

They had to do something.

I did not know what they should have done.

Information Distribution in Software

My first paper on ”information hiding” was presented at the IFIP conference in 1971.

The paper did explain that it was information distribution that made systems "dirty" by establishing almost invisible connections between supposedly independent modules.

This was an answer to one of my manager’s questions. I could explain what I meant by "clean".

NO answer for Jan or Johan.

Teaching an Early Software Engineering Course

Asked to teach a new "Software Engineering" course.

Nobody could tell me what the content of that course should be or how it should differ from an advanced programming course.

Class project with limited information distribution to demonstrate a "clean" design.

Defined "module" as, a work assignment for an individual or a group.

5 modules, Divided class into 5 groups of 4.

Every one does one assignment with the same specification

1024 possible combinations, 25 tested and made to work.

Unprecedented! Hard to do today!

What Johan Should have said to Jan

Johan and Jan should have shared a control-block hiding module.

They should have shared only the interface specification.

They needed a specification technique.

Two Journal Articles on Information Hiding

The better of the two articles was first rejected by the journal.

The review stated, "Obviously Parnas does not know what he was talking about because nobody does it that way."

Language - Often the Wrong Issue

Still searching for a good specification notation.

What Really Matters is Design

Limiting the information is the key!

Can formalise the concept of information hiding using Shannon’s theory of information. It doesn’t really help!

From Parnas-Modules to Information-Hiding Modules

SRI International introduced the HDM or Hierarchical Development Method

They referred to the modules as ”Parnas-Modules”.

I requested that they switch to ”information hiding” Modules.

This has caught on.

Information hiding is harder than it looks

Software engineering educational standards are much lower than those for other engineering fields, there are other reasons for failing to apply information hiding that I have observed.

1. The Flowchart Instinct
2. Planning for Change Seems like too much Planning
3. Bad Models
4. Extending Bad Software
5. Assumptions often go unnoticed:
   Major Flaw in my KWIC Index
6. Reflecting the system environment in the software structure.
7. "Oh, too bad we changed it"

Hierarchical Modular Structure for Complex Systems

When there are many modules more structure is needed.

Create a hierarchy using ”part of”.

Write a ”module guide” based on this hierarchy.

There are no "Levels of Abstraction".

Some authors and teachers use the phrase "levels of abstraction"

Terminology seems innocent bit not clearly defined.

The term "levels" can be used properly only if one has defined a relation that is loop free.

None of the authors has defined the relation, "more abstract than",

Must clearly distinguish between modules (collections of programs) and the programs themselves.

Useful to arrange programs in a hierarchical structure based on t "uses".

Not all programs in a module at the same level in the "uses" hierarchy.

Newer Ideas: Abstract Data Types, O-O, Components

Information-hiding is the basis of three more recent concepts.

(1) Abstract Data Types

- Many Copies of the hidden data structure

(2) Object-Oriented Languages

- Could be helpful but not really needed.
- Language is not the issue. Design is what matters
- Some features make it easy to do things wrong.

(3) Component-Oriented Programming

- The problems of the 50s, 60s, and 70s are still with us.

Future Research: Standard Information-Hiding Designs

The principle is now clear.

It is still hard to apply.

Researchers should be publishing proposed standard structures for classes of programs.

Researchers should be publishing proposed standard designs.