

Decomposing Systems into Modules

- Paper One: On the Criteria To Be Used in Decomposing Systems into Modules
-- by D. L. Parnas
- Paper Two: The Modular Structure of Complex Systems
-- by D.L. Parnas, P.C. Clements and D.M. Weiss

Overview

- ♦ Purpose
- ♦ A system design problem – KWIC index
- ♦ Conventional approach
- ♦ Unconventional approach
- ♦ Comparison
- ♦ Criteria used in decomposing the system
- ♦ Another system design problem – OFP
- ♦ Conclusion

Purpose

- ♦ Modularization:
 - a mechanism for enhancing flexibility and comprehensibility
 - Allowing shortening of development time
- ♦ Effectiveness of modularization
 - Depend on the criteria used in decomposing the system

Philosophy of Modular Programming

- ♦ A 1970 textbook by Gouthier and Pont

A well-defined segmentation of the project effort ensures system modularity. Each task forms a separate, distinct program module. At implementation time each module and its inputs and outputs are well-defined, there is no confusion in the intended interface with other system modules. At checkout time the integrity of the module is tested independently; there are few scheduling problems in synchronizing the completion of several tasks before checkout can begin. Finally, the system is maintained in modular fashion, system errors and deficiencies can be traced to specific system modules, thus limiting the scope of the detailed error searching.

Major Advancement in the area of Modular Programming

- ♦ The development of coding techniques and assemblers allows
 - one module to be rewritten with little knowledge of the code in another module
 - Modules to be reassembled and replaced without reassembly of the entire system

Primary Goal of Decomposing the System into Modules

- ✓ Managerial
 - Little need for communication
 - When implementing individual modules
 - Or changing individual modules
 - ✓ Product flexibility
 - Change to one module will not cause change in another
 - ✓ Comprehensibility
 - Whole system can be better understood
 - Better designed
- All in all, lower cost

Example

- ◆ A KWIC index system is studied

Some basic concepts used in the analysis

Concepts

- ◆ Module
 - Responsibility assignment
- ◆ Modularization
 - System description
 - Includes design decisions
 - System level decisions

Now the KWIC index system

- ◆ Input
 - Accepts an ordered sets of lines
 - Each line an ordered set of words
 - Each word an ordered set of characters
- ◆ Any line can be circularly shifted
- ◆ Output
 - a listing of all circular shifts of all lines
 - In alphabetical order

How does it work

- ◆ Input

All work
And no play
Makes Jack a dull boy

How does it work II

- ◆ Output

work All
play And no
no play And
Makes Jack a dull boy
Jack a dull boy Makes
dull boy Makes Jack a
boy Makes Jack a dull
And no play
All work
a dull boy Makes Jack

Cool and Useful

<http://www.utdallas.edu/~agi012000/kwic3/index.html>

Modularization One (Mod I)

- ◆ Input
- ◆ Circular shift
- ◆ Alphabetizing
- ◆ Output
- ◆ Master control

Mod I: Input Module

- ◆ Reads the data from the input medium
- ◆ Stores them in core
- ◆ The characters are packed four to a word
- ◆ End of word indicator
- ◆ Index shows the starting address of each line

Mod I: Circular Shift

- ◆ Prepares an index giving the starting address of each circular shift
- ◆ Original index of the line in Module I
- ◆ Output of this module saved in core

Mod I: Alphabetizing

- ◆ Takes as input the arrays produced by previous two modules
- ◆ Outputs an index giving the starting address of circular shifts in alphabetical order
- ◆ Same format as of Module II

Mod I: Output Module

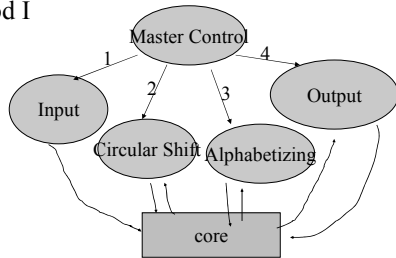
- ◆ Using arrays in Module III and I
- ◆ Generates an output listing all the circular shifts

Mod I: Master Control

- ◆ Control the sequencing among the four modules
- ◆ May handle error messages
- ◆ Space allocation
- ◆ And things like that

Diagram One

Mod I



Modularization Two (Mod II)

- ◆ Line Storage
- ◆ Input
- ◆ Circular Shifter
- ◆ Alphabetizer
- ◆ Output
- ◆ Master Control

Looks familiar

But is it the same as Mod I?

Mod II: Line Storage

- ◆ Consists of functions or subroutines
- ◆ Provides means by which user of the module may call on it
- ◆ Examples:
 - CHAR(r,w,c)
 - SETCHAR(rp,v, c, d)
 - WORDS(r)
- ◆ Restrictions in calling the routines
- ◆ Additional routines
 - Number of words in any line
 - Number of lines currently stored
 - Number of characters in any word
 - DELINE
 - DELWRD

Mod II: Input Module

- ◆ Reads the original lines
- ◆ Calls line storage module
- ◆ Have them store internally

Mod II: Circular Shifter

- ◆ Functions
 - analogs of functions in module 1
 - CSCHAR(l,w,c)
 - CSSETUP
- ◆ create a line holder containing all of the circular shifts of lines

Mod II: Alphabetizer

- ◆ Consists of two functions
- ◆ ALPH
 - Called first
 - Alphabetizing
- ◆ ITH
 - Serves as an index
 - ITH(i) gives the index of the circular shift which comes ith in the alphabetical ordering

Mod II: Output Module

Give the desired printing of circular shifts

Mod II: Master Control

Similar to that of modularization one

Diagram Two

Mod II

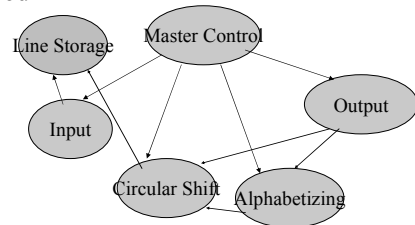
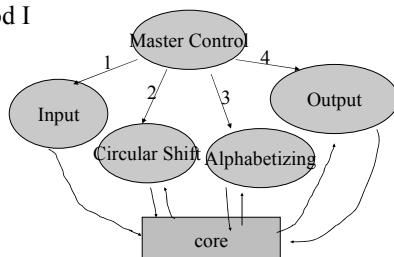


Diagram One

Mod I



Comparison

- ◆ Both will work
- ◆ Algorithms used might be same
- ◆ Identical runnable representation
- ◆ Yet other representation different
 - Changing
 - Documenting
 - understanding

Things that may change

- ◆ Input format
- ◆ Decision to have all lines stored in core
- ◆ The decision to pack 4 characters to a word
- ◆ index circular shifts rather than store them
- ◆ Alphabetize the list once

Differences:Changeability I

- ◆ Input Format change:
 - both affect input module
- ◆ If not store all lines in core
- ◆ Not pack four characters a word
- ◆ Some system uses noise function to get rid of noise words
- ◆ May use symbol table within the storage module
 - Mod I: changes needed in every module
 - Mod II: only affects module one

Differences:Changeability II

- ◆ Index the circular shifts vs store them as such
 - Mod II, change only affects circular shifter
 - Mod I, alphabetizer + output also need to know
- ◆ Alphabetize once
 - Mod I, output module can begin after alphabetizing is done
 - Mod II, no change is need

Differences:Comprehensibility

- ◆ Mod I
 - need to understand other module(s)
 - When understanding one
- ◆ Mod II
 - Each module has the knowledge of their own
 - Hiding from other module

Difference: Independent Development

- ◆ Mod I:
 - Interfaces between the modules are complex formats
 - joint effort among several development groups
- ◆ Mod II:
 - Interface more abstract
 - Consist in the function names
 - The numbers
 - Type of parameters
 - Independent development of modules

Criteria

- ◆ In Mod I
 - Each major step is a module
 - Common approach
 - Flow chart
 - Good for system with 5,000-10,000 instructions
 - Insufficient for larger projects
 - Big Projects, start with a module guide

Criteria

- ◆ Mod II: Information Hiding (IH)
 - No longer correspond to steps in the processing
 - Every module knows its design decision
 - Other modules don't have this knowledge
 - Interface reveals as little as possible

Conclusion

- ◆ Advise against
 - decomposing a system based on flowchart
- ◆ Propose
 - Taking the IH approach
 - Begin with a list of difficult design decisions
 - Or design decisions which are likely to change
 - Each module hides such decision from others

Complex System

- ◆ More about information hiding

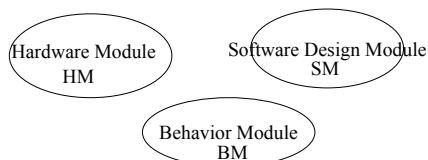
- ◆ An example: OFP for A-7E aircraft

OFP

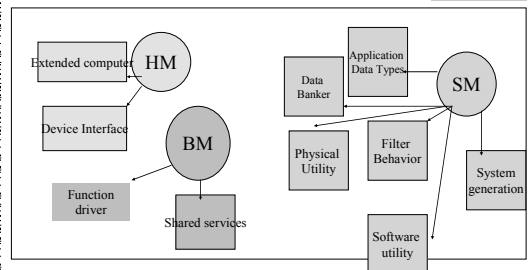
- ◆ Stands for Onboard Flight Program
- ◆ A project for A-7E aircraft
- ◆ SW by the Naval Research Lab
- ◆ A real-time program
- ◆ processes flight data
- ◆ Controls displays for the pilot
- ◆ Computes the aircraft position
- ◆ Using internal navigation system
- ◆ Must be accurate!!

Module Guide

◆ Top Level



Second Level



Design Principle

- ◆ Criterion we mentioned before – IH
- ◆ Likelihood to change
 - System details likely to change independently should be secrets to itself
 - Interfaces between modules include information less likely to change
- ◆ Data structure
 - Only accessible to modules inside itself
 - Others obtain information by calling its access methods

Three Ways to Describe a Modular Structure

- ◆ Roles played by individual modules
- ◆ Secrets associated with each module
- ◆ Facilities provided by each module
 - Also called module specifications
 - How to use the module
 - What does the module do

Two Kinds of Secrets

- ◆ Primary secrets
 - Hidden information
 - Specified to the software designer
- ◆ Secondary secrets
 - Implementation decision
 - When implementing the module

Summary

- ◆ Two system examples
- ◆ Two modularizations
- ◆ IH criterion is advisable
- ◆ Flowchart approach is less desirable
- ◆ Likelihood to change is an measurement of effectiveness of modularization

Resources

- ➔ The Modular Structure of Complex Systems– by D. L. Parnas, P.C. Clements, D. M. Weiss
- ➔ Enhancing Reusability with Information Hiding – by D. L. Parnas, P.C. Clements, D. M. Weiss
- ➔ <http://fxvcl.sourceforge.net/kwicIndex.html>
- ➔ http://www.utdallas.edu/~vxs037000/kwic/class_diagram.htm
- ➔ <http://www.ipsj.or.jp/members/Magazine/Eng/0906/article001.html>
- ➔ <http://www.cacs.louisiana.edu/~arun/courses/453.Fall01/home/assignment2/kwic1.9alpha.doc>

Special Thanks

Help from Dr. Coppit

