On the Criteria To Be Used in Decomposing Systems into Modules

D.L. Parnas
Carnegie-Mellon University

Presented by:
Loretta Macklem

Parnas D.L., On the Criteria To Be Used in Decomposing Systems into Modules, *Communications of the ACM*, Vol. 15, No. 12, December 1972 pp. 1053 - 1058

Outline

- Problem description
- Two example decompositions
- Comparison of two examples
- General decomposition criteria
- Impact

Problem Statement

- Problem
  - Programs with a large amount of functionality are by nature large and complicated
  - A criteria for decomposing programs is need

Modularization

- Definition from paper
  - "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 detailed error searching." Gouthier and Pont [1,I0.23]

Benefits of Modules

- Managerial
  - development time can be shortened
  - separate groups can work on each module with little need for communication
- Product flexibility
  - drastic changes can be made to one module without changing others
- Comprehensibility
  - it is possible to study the system one module at a time

Benefits for criteria

- In order for the previous benefits to be maximized the modules need to be divided properly

Contribution

- Criteria to be considered when dividing a system into modules
- Previously the criterion was left unmentioned

KWIC Index Production System

- **Input is a set of lines**
  - **Each line is an ordered set of words**
  - **Each word is an ordered set of characters**
- **Each line may be "circularly shifted"**
  - **Remove the first word and append it at the end of the line**
- **Output is a listing of all circular shifts of all lines in alphabetical order**

Example

- Input: abcd efgh ijkl mnop
- Output: abcd efgh ijkl mnop
  efgh ijkl mnop abcd
  ijkl mnop abcd efgh
  mnop abcd efgh ijkl

Decomposition 1

- Module 1: Input
  - Reads in the data lines, store them internally
- Module 2: Circular Shift
  - Produces an array which lists the index of the first character for each circular shift as well as the index of the original starting word
- Module 3: Alphabetizing
  - Produces the same array as module 2, except it is sorted alphabetically
- Module 4: Output
  - Outputs all of the circular shifts in alphabetically order using the array from module 3, and lines stored by module 1
- Module 5: Master Control
  - Controls the sequencing of the other modules

Decomposition 2

- Module 1: Line Storage
  - Provides functions to obtain or set the value of a specific character in a specific word number on a specific line, and to obtain the number of words in a line
- Module 2: Input
  - Reads in input and uses module 1 to store the data internally
- Module 3: Circular Shifter
  - CSCHAR(l,w,c) allows the user to obtain the value of the cth character of the wth word of the lth circular shift
  - CSSetup which sets the values that can be obtained by using the above function
- Module 4: Alphabetizer
  - Gives the ability to determine the alphabetical ordering of the shifts
- Module 5: Output
  - Prints out the set of lines or circular shifts
- Module 6: Master Control
  - Controls the sequencing of the above modules

Comparison

- General
  - Both will produce programs that work
  - The two decompositions may be the same program after they have been compiled

Comparison Cont…

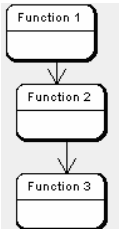
- Changeability
  - Input format change
    - Decomposition 1 and 2 would need to change only one module
  - No longer store all lines in core or whether to pack the characters four to a word
    - Decomposition 1 would need to change all modules, but Decomposition 2 would only need to change module 1

And More…

- Independent Development
  - The interfaces between modules in decomposition 1 are more complex than the interfaces between modules in decomposition 2
- Comprehensibility
  - According to author (I am undecided) 2 is easier to understand than 1

Hierarchical Structure

- Map out the program into a tree structure
  - Functions can only use other functions who are above it in the tree



Hierarchical Structure

- Decomposition 2 as a system has program hierarchy
- Level 1: Symbol table (if exists)
- Level 2: Line Storage
- Level 3: Input and Circular Shifter
- Level 4: Output and Alphabetizer
  - Both output and alphabetizer require circular shifter
  - Line holder and circular shifter are in some sense compatible and the level 4 functionality could perhaps be incorporated into a parameterized version of the functions

Cont…

- Benefits of hierarchy
  - Parts of the system are simplified because they use the services of lower levels
  - The upper levels may be cut off and still have a usable product
    - Could use levels 1 and 2 and produce a question answering system

Independence

- Hierarchal structure and clean decomposition are independent properties
- Although Decomposition 2 produced a hierarchal program, it is conceivable to obtain this also with using the design approach used for decomposition one

Criteria used for Decomposition 1 and 2

- Decomposition 1 used a flowchart approach where each module was a major step in the process
- Decomposition 2 used information hiding as a criterion
  - Each module is characterized by its hiding of the design decision in its implementation
    - By hiding the design decision, any user of the module need not know the inner workings associated with the decision, and thus changing the decision is not costly

General Criteria

1) Data structure, its accessing procedures and modifying procedures are part of a single module

1a) Example: Data is stored in a linked list, the procedures to access and modify the list should be in the same module as the list

2) The necessary steps to call a given routine and the routine itself are part of the same module

2a)Ex.: A program that has a function which only takes in String, and the input currently is in the format of integer, there may be code to turn the integer into a string

Criteria Cont…

3)Formats of control blocks used in queues or similar programs should be hidden within a control block module

3a) Format of control block is usually taken care of by making the formats the interfaces between modules, but this functionality tends to change frequently which makes that decision costly

4)Character codes, alphabetic orderings, and similar data should be in its own module

4a) Increase flexibility

Continued…

5)Sequence in which items will be processed should be in a single module

- Sequencing is very variable

Effect of work

- Modularization is widely used and is taught in most introduction to programming courses
- Object oriented programming languages all exhibit modularization
  - The methods that you decide to include in a class fall under the criteria

3 questions

- Technology and languages have progressed since this paper was published, is there any new criteria that you think should be added to the list?
- Which UML model would you use to convey the division of modules?
- Do you think large scale projects could be done without modularization (is there any alternative method that could work)?

References

- 1.Gauthier, Richard, and Pont, Stephen. *Designing Systems Programs, (C),* Prentice-Hall, Englewood Cliffs, N.J., 1970.