Refactoring - Program Restructuring

Idaho State University | Computer Science

Isaac Griffith

CS 4423 and CS 5523
Department of Computer Science
Idaho State University

ROAR

Idaho State University
Computer Science

Outcomes

After today's lecture you will:

- Be able to understand and describe various approaches to program restructuring.

ROAR

Software Restructuring

CS 4423/5523

ROAR

Software Restructuring

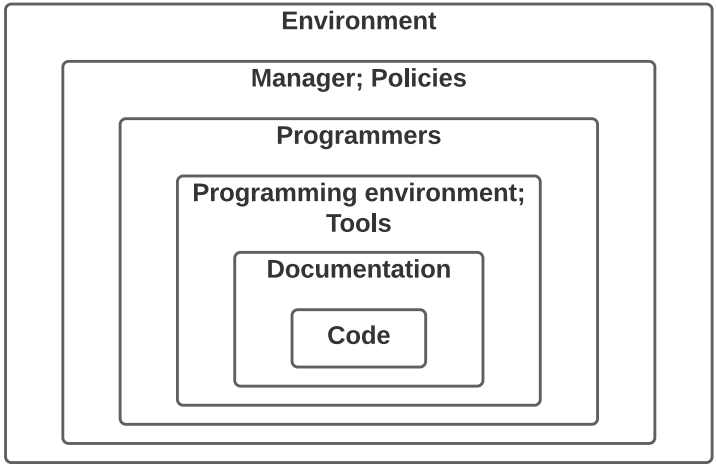
- Software restructuring dates back to the mid 1960s, as soon as programs were written in Fortran.
- Topics of discussion in this section are:
  - Factors influencing software structure
  - Classification of restructuring approaches
  - Restructuring techniques
    - Elimination-of-goto approach
    - Localization and information hiding approach
    - System sandwich approach
    - Clustering approach
    - Program slicing approach

Software Structure Factors

- Software structure is a set of attributes of the software such that the programmer gets a good understanding of software.
- Any factor that can influence the state of software or the programmer's perception might influence software structure.
- One view of the factors that influence software structure has been shown in Fig. 7.9.
    - Code
    - Tools
    - Managers and policies
    - Documentation
    - Programmers
    - Environment

Idaho State University
Computer Science

Software Structure Factors



ROAR

Idaho State University
Computer Science

Software Structure Factors

- Code
  - Code quality at all levels of details (e.g. variables, constants, statements, function, and module) impact code understanding.
  - Adherence to coding standards improves code quality.
  - Adoption of common architectural styles enhances code understanding.
- Documentation
  - Internal documentation (also known as in-line codumentation)
  - External documentation
    - Requirements documents
    - Design documents
    - User manuals
    - Test cases

Software Structure Factors

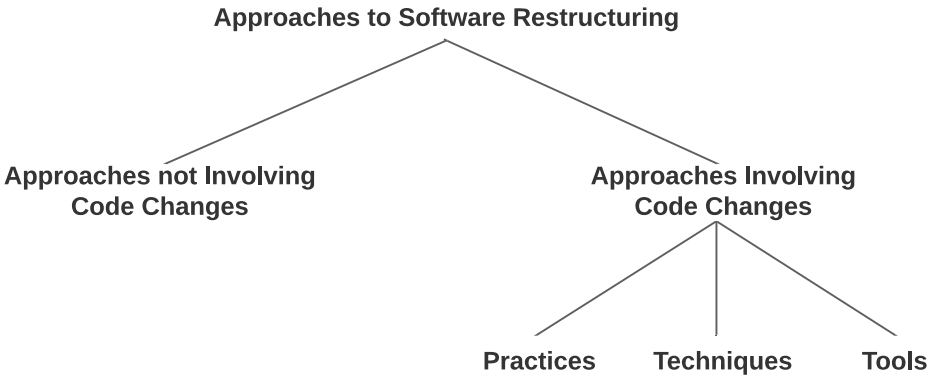
- Tools – Programming environment
  - Development tools help programmers better understand the code.
    - Tracing of source code help in understanding the dynamic behavior of the code.
    - Animation of algorithms help in understanding the dynamic strategy adopted in algorithms.
    - Cross referencing of global variables reveal interactions among modules.
    - Tools can reformat code for better readability via pretty printing, highlighting of key words, and color coding of source code.
- Programmers
  - Qualities of programmers influence their perception of structure.
    - Individual capabilities
    - Education
    - Experience and training
    - Aptitude

Software Structure Factors

- Managers and policies
  - Management can play an influencing role in having a good initial structure and sustain it by designing policies and allocating resources.
  - Examples
    - Management can design general policies to adhere to standards.
    - Management can tie the annual performance review with the programmer's adherence to standards.
- Environment
  - This refers to the general working environment of programmers.
  - Example: Physical facilities and availability of resources when needed

Restructuring Approaches

- A broad classification of software restructuring approaches



Restructuring Approaches

- Approaches not involving code changes
  - Train programmers in software engineering, including architectural styles and modularization techniques.
  - Upgrade documentation
    - Make in-line comments more accurate and readable.
    - Update comments to reflect code changes.
    - Update external documentations to make them consistent with code, accurate, and complete.

Restructuring Approaches

- Approaches involving code changes
  - **Practices:**
    - Restructuring code with preprocessors.
    - Making code understandable by means of inspection.
    - Formatting code by adhering to standards and style guidelines.
    - Restructuring code for reusability.
  - **Techniques:**
    - Incremental restructuring
    - Goto-less approach
    - Case-statement approach
    - Boolean flag approach
    - Clustering approach
  - **Tools**
    - Eclipse IDE, IntelliJ IDEA, jFactor, Refactorit, and Clone Doctor

Restructuring Techniques

- Restructuring techniques
  - Those were developed in the mid-70s, before object-oriented programming.
  - The techniques are applied at different levels of abstractions.
- Example of restructuring techniques
  - Elimination-of-goto Approach
  - Localization and Information Hiding Approach
  - System Sandwich Approach
  - Clustering Approach
  - Program Slicing Approach

Elimination-of-goto Approach

- Before the onset of structured programming, much code was written in the '70s with `goto` statements.
- Structured programming puts emphasis on the following control constructs: `for`, `while`, `until`, and, `if-then-else`.
- Those constructs make occurrences of loop and branching clear.
- It has been shown that every flowchart program with `goto` statements can be transformed into a functionally equivalent `goto-less` program by using `while` statements.

Localization Approach

- Localization
  - It is a process of collecting the logically related computational resources in one physical module.
    - Functions, procedures, operations, and data types are computational resources.
  - By localizing computational resources into separate modules, programmers can restructure a program into a loosely coupled system of sufficiently independent modules.
  - Sometimes, localization is difficult to achieve.
    - A variable may be imported into a module by means of the include statement.
    - Data sharing among functions is not explicitly represented in source code.

Localization Approach

- A restructuring process based on localization of variables and functions
  - Localization of variables
    - Organize global variables and functions which refer to those global variables into package-like groups.
    - This organization can be achieved by applying the concept of closure of functions to a set of global variables.
    - This leads to groups of functions and global variables referred to by those functions.
  - Localization of functions
    - Put locally called functions and the calling function in the same group.

Information Hiding Approach

- Information Hiding
  - The details of implementations of computational resources can be hidden to make it easier to understand the program.
  - For example, a queue is a high level concept which can be implemented by means of a variety of low level data structures.
    - Singly linked list
    - Doubly linked list
    - Arrays
  - A programmer can design a function by using enqueue and dequeue calls without any concern for their actual implementations.

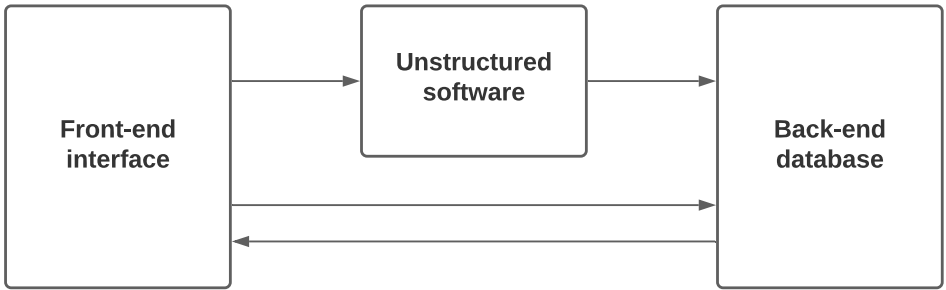
Information Hiding Approach

- A restructuring process based on localization of variables and functions
  - Information hiding and hierarchical structuring
    - Organize groups of functions into hierarchical package structures based on the visibility of functions within groups.
    - Those functions and variables which are only externally referable and visible to other packages constitute the package specification.

Idaho State University
Computer Science

System Sandwich Approach

- This approach is applied to those software which cannot be restructured with any hope, but need to be retained for their outputs.
- As illustrated, write a new front-end interface and a new back-end data base so that:
  - it is easy to interface with the program; and
  - the program's outputs are recorded in a more structured way.



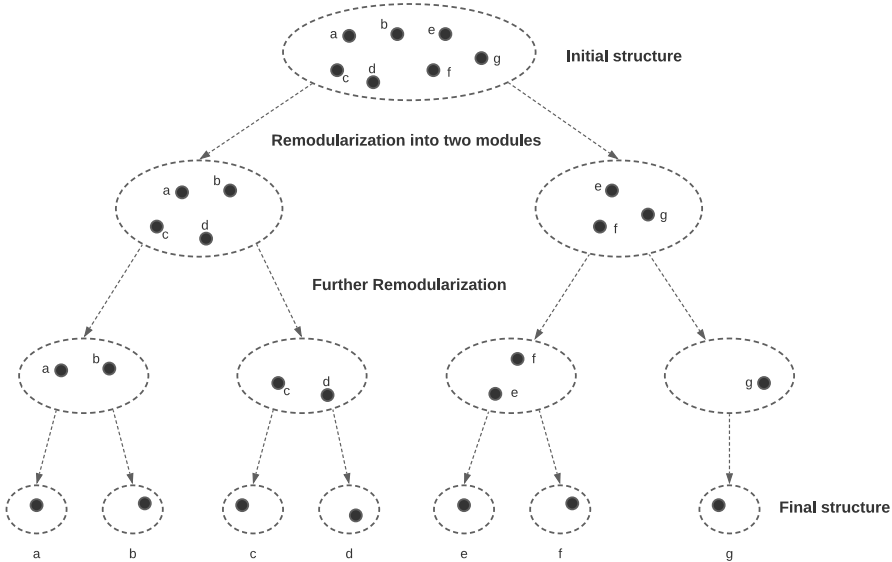
ROAR

Clustering Approach

- Software modularization is an important design step.
- A program can be remodularized in two ways.
  - System level remodularization
    - This is a top-down approach.
    - Partition the program into smaller modules
  - Entity level remodularization
    - This is a bottom-up approach.
    - Group a program's entities to form larger modules.

Idaho State University
Computer Science

Clustering Approach



ROAR

Clustering Approach

- The concept of clustering is key to modularization.
- Clusters are defined as continuous regions of space containing a relatively high density of points, separated from other such regions by regions containing a relatively low density of points.
- Modularization is defined as the clustering of large amount of entities in groups in such a way that the entities in one group are more closely related, based on some similarity metrics.
- While applying the idea of clustering, two factors are taken into account:
  - What similarity metrics to consider?
  - What clustering algorithm to use?

Clustering Approach

- Similarity metrics
  - Distance measures
    - Euclidean distance
    - Manhattan distance
  - Association coefficients
    - Simple matching coefficient
    - Jaccard coefficient

Idaho State University
Computer Science

Association coefficient examples

- Let x and y be two entities. Let:

  a = # of features present for both x and y.

  b = # of features present for x but not y.

  c = # of features present for y but not x.

  d = # of features **not** present for both x and y.

- Simple matching coefficient: $simple(x, y) = (a + d)/(a + b + c + d)$.
- Jaccard coefficient: $Jaccard(x, y) = a/(a + b + c)$.

Clustering Approach

- Clustering algorithms: three broad techniques applied.
  - Graph theoretical algorithms
  - Construction algorithms
  - Optimization algorithms (aka iterative and improvement algorithms)
  - Hierarchical algorithms
    - Divisive algorithms
    - Agglomerative algorithms
  - The clustering produced by a hierarchical algorithm can be visualized in a dendogram
    - The dendogram representation of the hierarchy

Clustering Approach

- The general structure of an agglomerative algorithm

```
1. IF there are N entities, begin with N clusters such that each
   cluster contains a unique entity.
   Compute the similarities between the clusters
2. WHILE there is more than a cluster
   DO
      Find the most similar pair of clusters and merge them into
      a single cluster.
      Recompute the similarities between the clusters
   END
```

Divisive Algorithms

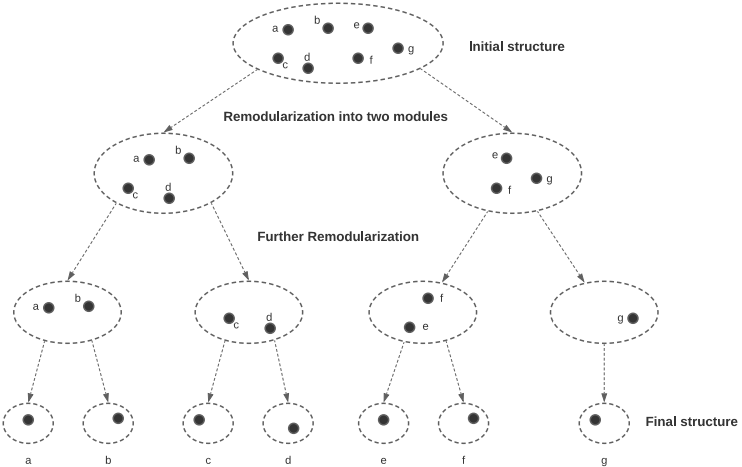
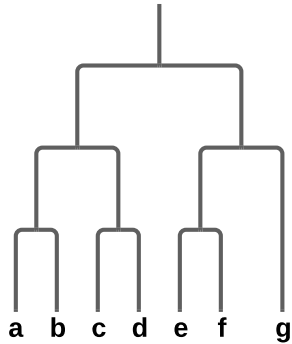


Illustration of system level remodulariation.

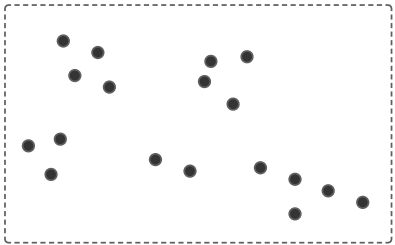
- Bullets represent low level entities
- Dashed shapes represent modules
- Arrows represent progression from one level to the next



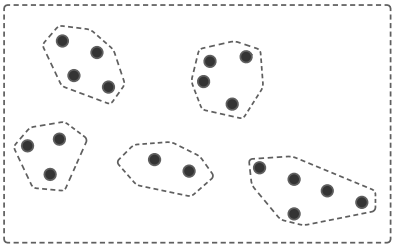
Dendrogram repsentation of the clustering process

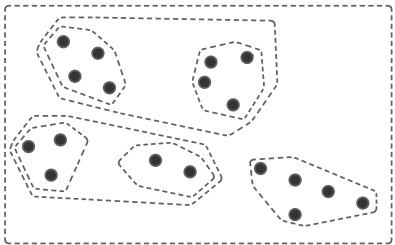
Agglomerative algorithms

A system with one module – the entire system



The system after the first-level of remodularization with five modules





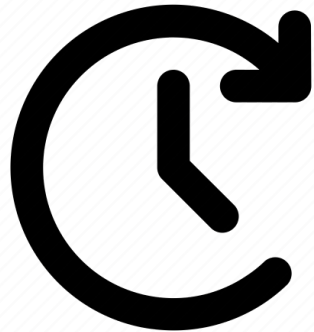
The system after second-level remodularization with three upper-level modules.

Program Slicing Approach

- Two kinds of program slicing
  - **Backward slicing:** The set of statements that can affect the value of a variable at some point of interest in a program is called a backward slice.
  - **Forward slicing:** The set of statements that are likely to be affected by the value of a variable at some point of interest in a program is called a forward slide.
- A key idea in program slicing
  - Identify and extract a cohesive subset of statements from a program.
- Therefore, if a module supports multiple functionalities, a portion of the code can be extracted to form a new module.
- Large functions can be decomposed into smaller functions by means of program slicing to restructure programs.

For Next Time

- Review EVO Chapter 7.5 - 7.6
- Read EVO Chapter 8.1 - 8.2
- Watch Lecture 20



Are there any questions?