CS 635 Advanced Object-Oriented Design & Programming Spring Semester, 2005 Doc 9 Cohesion

Contents

References

Object Coupling and Object Cohesion, chapter 7 of *Essays on Object-Oriented Software Engineering*, Vol 1, Berard, Prentice-Hall, 1993,

Cohesion

"Cohesion is the degree to which the tasks performed by a single module are functionally related."

IEEE, 1983

"Cohesion is the "glue" that holds a module together. It can be thought of as the type of association among the component elements of a module. Generally, one wants the highest level of cohesion possible."

Bergland, 1981

"A software component is said to exhibit a high degree of cohesion if the elements in that unit exhibit a high degree of functional relatedness. This means that each element in the program unit should be essential for that unit to achieve its purpose."

Sommerville, 1989

Types of Module Cohesion
From Worst to Best

Coincidental (worst)

Logical

Temporal

Procedural

Communication

Sequential

Functional (best)

Module Cohesion
Coincidental

Little or no constructive relationship among the elements of the module

Common Object Occurrence:

> Object does not represent any single object-oriented concept
>
> Collection of commonly used source code as a class inherited via multiple inheritance

```
class Rous
   {
   public static int findPattern( String text, String pattern)
      { // blah}

   public static int average( Vector numbers )
      { // blah}

   public static OutputStream openFile( String fileName )
      { // blah}

   }
```

Module Cohesion
Logical

Module performs a set of related functions, one of which is selected via function parameter when calling the module

Similar to control coupling

Cure:

Isolate each function into separate operations

```
public void sample( int flag )
  {
  switch ( flag )
    {
    case ON:
      // bunch of on stuff
      break;
    case OFF:
      // bunch of off stuff
      break;
    case CLOSE:
      // bunch of close stuff
      break;
    case COLOR:
      // bunch of color stuff
      break;
    }
  }
```

Module Cohesion
Temporal

Elements are grouped into a module because they are all processed within the same limited time period

Common example:

"Initialization" modules that provide default values for objects

"End of Job" modules that clean up

```
procedure initializeData()
   {
   font = "times";
   windowSize = "200,400";
   foo.name = "Not Set";
   foo.size = 12;
   foo.location = "/usr/local/lib/java";
   }
```

Cure: Each object should have a constructor and destructor

```
class foo
   {
   public foo()
      {
      foo.name = "Not Set";
      foo.size = 12;
      foo.location = "/usr/local/lib/java";
      }
   }
```

Sample Configuration File

[Macintosh]	[General]
EquationWindow=146,171,406,661	Zoom=200
SpacingWindow=0,0,0,0	CustomZoom=150
	ShowAll=0
	Version=2.01
[Spacing]	OptimalPrinter=1
LineSpacing=150%	MinRect=0
MatrixRowSpacing=150%	ForceOpen=0
MatrixColSpacing=100%	ToolbarDocked=1
SuperscriptHeight=45%	ToolbarShown=1
SubscriptDepth=25%	ToolbarDockPos=1
LimHeight=25%	
LimDepth=100%	[Fonts]
LimLineSpacing=100%	Text=Times
NumerHeight=35%	Function=Times
DenomDepth=100%	Variable=Times,I
FractBarOver=1pt	LCGreek=Symbol,I
FractBarThick=0.5pt	UCGreek=Symbol
SubFractBarThick=0.25pt	Symbol=Symbol
FenceOver=1pt	Vector=Times,B
SpacingFactor=100%	Number=Times
MinGap=8%	
RadicalGap=2pt	[Sizes]
EmbellGap=1.5pt	Full=12pt
PrimeHeight=45%	Script=7pt
	ScriptScript=5pt
	Symbol=18pt
	SubSymbol=12pt

Call these constructors/destructors from a nonobject-oriented routine that performs a single, cohesive task

Module Cohesion
Procedural

Associates processing elements on the basis of their procedural or algorithmic relationships

Procedural modules are application specific

In context the module seems reasonable

Removed from the context these modules seem strange and very hard to understand

Why is that being done here?

Can not understand module without understanding the program and the conditions existing when module is called

Makes module hard to modify, understand

Class Builder verse **Program writer**

Cure:

Redesign the system

If a module is necessary, remove it from objects

Module Cohesion
Communication

Operations of a module all operate upon the same input data set and/or produce the same output data

Cure:

Isolate each element into separate modules

Rarely occurs in object-oriented systems due to polymorphism

Module Cohesion
Sequential

Sequential association the type in which the output data from one processing element serve as input data for the next processing element

A module that performs multiple sequential functions where the sequential relationship among all of the functions is implied by the problems or application statement and where there is a data relationship among all of the functions

Cure:

Decompose into smaller modules

Module Cohesion
Functional

If the operations of a module can be collectively described as a single specific function in a coherent way, the module has functional cohesion

If not, the module has lower type of cohesion

In an object-oriented system:

- Each operation in public interface of an object should be functional cohesive
- Each object should represent a single cohesive concept

Module Cohesion
Informational Strength

Myers states:

"The purpose of an informational-strength module is to hide some concept, data structure, or resource within a single module.

An informational-strength module has the following definition:

- It contains multiple entry points
- Each entry point performs a single specific function
- All of the functions are related by a concept, data structure, or resource that is hidden within the module"

Object Cohesion

The degree to which components of a class are tied together

Evaluating cohesion requires:

- Technical knowledge of the application domain
- Some experience in building, modifying, maintaining, testing and managing applications in the appropriate domain
- Technical background in and experience with reusability

Questions to probe cohesiveness of an object

Does the object represent a complete and coherent concept or does it more closely resemble a partial concept, or a random collection of information?

Does the object directly correspond to a "real world entity," physical or logical?

Is the object characterized in very non-specific terms?

Collection of data, statistics, etc.

Do each of the methods in the public interface for the object perform a single coherent function?

If the object (or system of objects) is removed from the context of the immediate application, does it still represent a coherent and complete object-oriented concept?

For objects that are "system of objects"

Does the system represent an object-oriented concept?

Do all the objects directly support, or directly contribute to the support of, the object-oriented concept that the system represents?

Are there missing objects?

Objects in Isolation

Isolation means without considering any hierarchy that may contain the object or class

Does not discuss non-objects:

- Object with only functions
- Objects with only data

Individual Objects

A **primitive method** is any method that cannot be implemented simply, efficiently, and reliably without knowledge of the underlying implementation of the object

A **composite method** is any method constructed from two or more primitive methods – sometimes from different objects

A **sufficient set of primitive methods** for an object is a minimum set of primitive methods to accomplish all necessary work with on the object

A sufficient set of primitive methods has two major problems:

- Some tasks may be awkward and/or difficult with just a sufficient set of primitive methods
- A sufficient set of primitive methods may not allow us to fully capture the abstraction represented by the object

A **complete set of primitive methods** is a set of primitive methods that both allows us to easily work with the object, and fully captures the abstraction represented by the object.

An object is not as cohesive as it could be if the public interface contains:

- Only primitive methods, but does not fully capture the abstraction represented by the object

- Primitive and composite methods, but does not fully capture the abstraction represented by the object

- A sufficient set of primitive methods with composite methods

- No primitive methods, just composite methods

Note

- Objects with a sufficient set of primitive methods with composite methods is more cohesive than objects with out a sufficient set of primitive methods

- All public methods must directly support the abstraction represented by the object. The methods must make sense when object is removed from the application

Composite Objects

A **composite object** is an object that is conceptually composed of two, or more, other objects, which are externally discernable.

Component objects are those that make up the composite object.

Component objects are **externally discernable** if

- Component objects can be directly queried or changed via methods in the public interface of the composite object and/or

- The externally discernible state of the object is directly affected by the presence or absence of one or more component objects

Ranking of Cohesion of Composite Objects Increasing order of Goodness

- Externally discernible component objects not related

- Some externally discernible component objects are related, the group component objects does not make sense

- The group component objects does not represent a single stable object-oriented concept, but are all bound together some how in an application

- A majaroity of the externally discernible component objects support a single, coherent, object-oriented concept, but at least one does not

- All of the externally discernible component objects support a single, coherent, object-oriented concept, but at least one needed is missing

- All of the externally discernible component objects support a single, coherent, object-oriented concept, and none are missing

Accessing Cohesion of an Individual Object

Assessment of the public methods/public non-methods/component objects

Are all the items appropriate for the given object?

Do we have at least a minimally sufficient set of items?

Do we have extra or application-specific items?