Effective Modular Design

- Modular design
  - Reduces complexity
  - Facilitates change
  - Results in easier implementation by supporting parallel development of different parts of the system.

- Functional independence is achieved by developing modules with:
  - Single minded function
  - An aversion to excessive interaction with other modules.

- Independent modules are easier to maintain and test
  - Secondary effects caused by design/code modification are limited
  - Error propagation is reduced
  - Re-use is increased
Two qualitative criteria

- **Cohesion**
  A measure of the relative functional strength of a module
  
  ![Func A-1, Func A-2, Func A-3 vs. Func B-1, Func B-2, Func B-3]

  *High Cohesion (good)*

- **Coupling**
  A measure of the relative interdependence among modules.

  ![Diagram of coupling with arrows]

  *High coupling (bad)*
## Metrics

<table>
<thead>
<tr>
<th>SOURCE</th>
<th>METRIC</th>
<th>OBJECT-ORIENTED CONSTRUCT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traditional</td>
<td>Cyclomatic complexity (CC)</td>
<td>Method</td>
<td>Traditional</td>
</tr>
<tr>
<td>Traditional</td>
<td>Lines of Code (LOC)</td>
<td>Method</td>
<td></td>
</tr>
<tr>
<td>Traditional</td>
<td>Comment percentage (CP)</td>
<td>Method</td>
<td></td>
</tr>
<tr>
<td>Object-Oriented</td>
<td>Weighted methods per class (WMC)</td>
<td>Class/Method</td>
<td>Architecture</td>
</tr>
<tr>
<td>Object-Oriented</td>
<td>Response for a class (RFC)</td>
<td>Class/Message</td>
<td></td>
</tr>
<tr>
<td>Object-Oriented</td>
<td>Lack of cohesion of methods (LCOM)</td>
<td>Class/Cohesion</td>
<td></td>
</tr>
<tr>
<td>Object-Oriented</td>
<td>Coupling between objects (CBO)</td>
<td>Coupling</td>
<td></td>
</tr>
<tr>
<td>Object-Oriented</td>
<td>Depth of inheritance tree (DIT)</td>
<td>Inheritance</td>
<td>Tree structure</td>
</tr>
<tr>
<td>Object-Oriented</td>
<td>Number of children (NOC)</td>
<td>Inheritance</td>
<td></td>
</tr>
</tbody>
</table>

### Cyclomatic Complexity (CC)

![Diagram of cyclomatic complexity](image)

\[
CC = \text{edges} - \text{nodes} + 2
\]

- Evaluates the complexity of an algorithm in a method.
- Calculate the cyclomatic complexity. How? (See notes on whitebox testing).
- A method with a low cyclomatic complexity is generally better. This may imply decreased testing and increased understandability or that decisions are deferred through message passing, not that the method is not complex.

[http://satc.gsfc.nasa.gov/support/STC_APR98/apply_oo/apply_oo.html](http://satc.gsfc.nasa.gov/support/STC_APR98/apply_oo/apply_oo.html)
**Weighted Methods per Class (WMC)**

WMC for *Clothing_dept* = 1  
WMC for *Appliance_dept* = 4

- Counts the methods implemented within a class or the sum of the complexities of the methods (method complexity is measured by cyclomatic complexity).
- Classes with large numbers of methods are likely to be more application specific, limiting the possibility of reuse.

---

**Methods per class**

High number of methods may have greater impact on children through inheritance  
May also indicate application specific, decreasing reusability.

---

http://satc.gsfc.nasa.gov/support/STC_APR98/apply_oo/apply_oo.html

www.software.org/metrics99/roenberg.ppt
Weighted methods per class

Complexity of a method: Ideal 1-5, but 10 is acceptable.
Number of methods in a class: 1-20
WMC: < 100 (5 complexity * 20 methods); should not exceed 200 (10 * 20)

Method Complexity (NASA data)
Complexity vs. Size (NASA)

Size to Complexity (RISK components)
Complexity vs. Size (NASA)

![Complexity vs. Size Chart]

Response for a Class (RFC)

- The RFC is the count of the set of all methods that can be invoked in response to a message to an object of the class or by some method in the class.
  - Includes all methods accessible within the class hierarchy.
  - Looks at the combination of the complexity of a class through the number of methods and the amount of communication with other classes.
  - The more methods that can be invoked from a class through messages, the greater the complexity of the class.
  - Increases complexity of testing and debugging as it requires a greater level of understanding on the part of the tester.
  - A worst case value for possible responses will assist in the appropriate allocation of testing time.

[www.software.org/metrics99/rosenberg.ppt](http://www.software.org/metrics99/rosenberg.ppt)

[http://satc.gsfc.nasa.gov/support/STC_APR98/apply_oo/apply_oo.html](http://satc.gsfc.nasa.gov/support/STC_APR98/apply_oo/apply_oo.html)
Response for a Class (RFC)

Response for a Class < 50, acceptable up to 100. 
> 100 ===> greater complexity and decreased understandability, changes become very difficult due to the potential for a ripple effect.

www.software.org/metrics99/rosenberg.ppt

Response for a Class (RFC)

Prime candidates for walkthrough and inspections

Classes near the 'possible' line are classes that do not invoke many outside methods.

http://satc.gsfc.nasa.gov/support/STC_APR98/apply_oo/apply_oo.html
Count of the number of other classes to which a class is coupled.
- Count the number of distinct non-inheritance related class hierarchies on which a class depends.

- Excessive coupling is detrimental to modular design and prevents reuse.
- High COB:
  - Prevents reuse.
  - Increases sensitivity to changes in other parts of the design. Therefore maintenance becomes harder.
  - Understandability decreases.
- Design classes with weak coupling.

http://satc.gsfc.nasa.gov/support/STC APR98/apply_oo/apply_oo.html

Coupling between Object Classes (COB)

Higher CBO indicates classes that my be difficult to understand
Decreased reuse and increased maintenance.

www.software.org/metrics99/rosenberg.ppt
Coupling between Object Classes (COB)

Candidates for revision and inspection.

Depth of Inheritance Tree (DIT)

- The depth of a class within the inheritance hierarchy is the maximum number of steps from the class node to the root of the tree and is measured by the number of ancestor classes.

- The deeper a class is within the hierarchy, the greater the number methods it is likely to inherit making it more complex to predict its behavior.

- Deeper trees constitute greater design complexity, since more methods and classes are involved, but the greater the potential for reuse of inherited methods.

- A support metric for DIT is the number of methods inherited (NMI)
Depth of Inheritance Tree (DIT)

- The number of children is the number of immediate subclasses subordinate to a class in the hierarchy.
- It is an indicator of the potential influence a class can have on the design and on the system.
- The greater the number of children, the greater the likelihood of improper abstraction of the parent and may be a case of misuse of subclassing.
- But the greater the number of children, the greater the reuse since inheritance is a form of reuse.
- If a class has a large number of children, it may require more testing of the methods of that class, thus increase the testing time.
Number of Children

Plot DIT versus NOC

- Higher DIT’s indicate a trade-off between increased complexity and increased reuse.
- Higher NOC’s also indicate reuse, but may require more testing.

http://satc.gsfc.nasa.gov/support/STC_AP98/apply_oo/apply_oo.html
“It Takes a Village”

Any class that meets *at least two* of the following criteria is flagged:

- Response for Class > 100
- Coupling between Objects > 5
- Response for Class > 5 time the number of methods in the class
- Weighted Methods per Class > 100
- Number of Methods > 40

---

### Project Analysis

Use this information to focus testing effort and to pinpoint possible areas for refactoring.

<table>
<thead>
<tr>
<th>Class</th>
<th>#Method</th>
<th>RFC</th>
<th>WMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class4</td>
<td>168</td>
<td>71</td>
<td></td>
</tr>
<tr>
<td>Class3</td>
<td>240</td>
<td>105</td>
<td></td>
</tr>
<tr>
<td>Class7</td>
<td>361</td>
<td>117</td>
<td></td>
</tr>
<tr>
<td>Class8</td>
<td>378</td>
<td>163</td>
<td></td>
</tr>
<tr>
<td>Class9</td>
<td>406</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>Class5</td>
<td>163</td>
<td>63</td>
<td></td>
</tr>
<tr>
<td>Class6</td>
<td>133</td>
<td>79</td>
<td></td>
</tr>
<tr>
<td>Class1</td>
<td>536</td>
<td>176</td>
<td></td>
</tr>
<tr>
<td>Class2</td>
<td>168</td>
<td>71</td>
<td></td>
</tr>
<tr>
<td>Class15</td>
<td>844</td>
<td>281</td>
<td></td>
</tr>
<tr>
<td>Class14</td>
<td>974</td>
<td>325</td>
<td></td>
</tr>
<tr>
<td>Class13</td>
<td>120</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>Class16</td>
<td>164</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>Class17</td>
<td>155</td>
<td>79</td>
<td></td>
</tr>
<tr>
<td>Class18</td>
<td>186</td>
<td>238</td>
<td></td>
</tr>
<tr>
<td>Class19</td>
<td>26</td>
<td>103</td>
<td></td>
</tr>
<tr>
<td>Class12</td>
<td>127</td>
<td>69</td>
<td></td>
</tr>
<tr>
<td>Class11</td>
<td>285</td>
<td>161</td>
<td></td>
</tr>
<tr>
<td>Class10</td>
<td>235</td>
<td>156</td>
<td></td>
</tr>
<tr>
<td>Class9</td>
<td>209</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>Class8</td>
<td>378</td>
<td>163</td>
<td></td>
</tr>
<tr>
<td>Class7</td>
<td>361</td>
<td>117</td>
<td></td>
</tr>
<tr>
<td>Class6</td>
<td>378</td>
<td>163</td>
<td></td>
</tr>
<tr>
<td>Class5</td>
<td>163</td>
<td>63</td>
<td></td>
</tr>
<tr>
<td>Class4</td>
<td>206</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>Class3</td>
<td>168</td>
<td>71</td>
<td></td>
</tr>
<tr>
<td>Class2</td>
<td>168</td>
<td>71</td>
<td></td>
</tr>
<tr>
<td>Class1</td>
<td>536</td>
<td>176</td>
<td></td>
</tr>
</tbody>
</table>

---

[www.software.org/metrics99/rosenberg.ppt](www.software.org/metrics99/rosenberg.ppt)