Software Metrics

By - Nisheeth Bhat

Need for Software Metrics

- Quantify the project
  - Quantifying schedule , performance ,work effort , project status
  - Helps software to be compared and evaluated
- Better Estimates
  - Use the measure of your current performance to improve your future work estimates
- Resolve Software crises
  - Such as wrong cost estimates, slow productivity rate and so on

Goals of software metrics

- Estimate the cost & schedule of future projects
- Evaluate the productivity impacts of new tools and techniques
- Establish productivity trends over time
- Improve software quality
- Forecast future staffing needs
- Anticipate and reduce future maintenance needs

What are Software Metrics ?

- Software metrics are units of measurement of software product and the process by which it is developed
- Metrics and measures form the basis for numerous models of the software development process

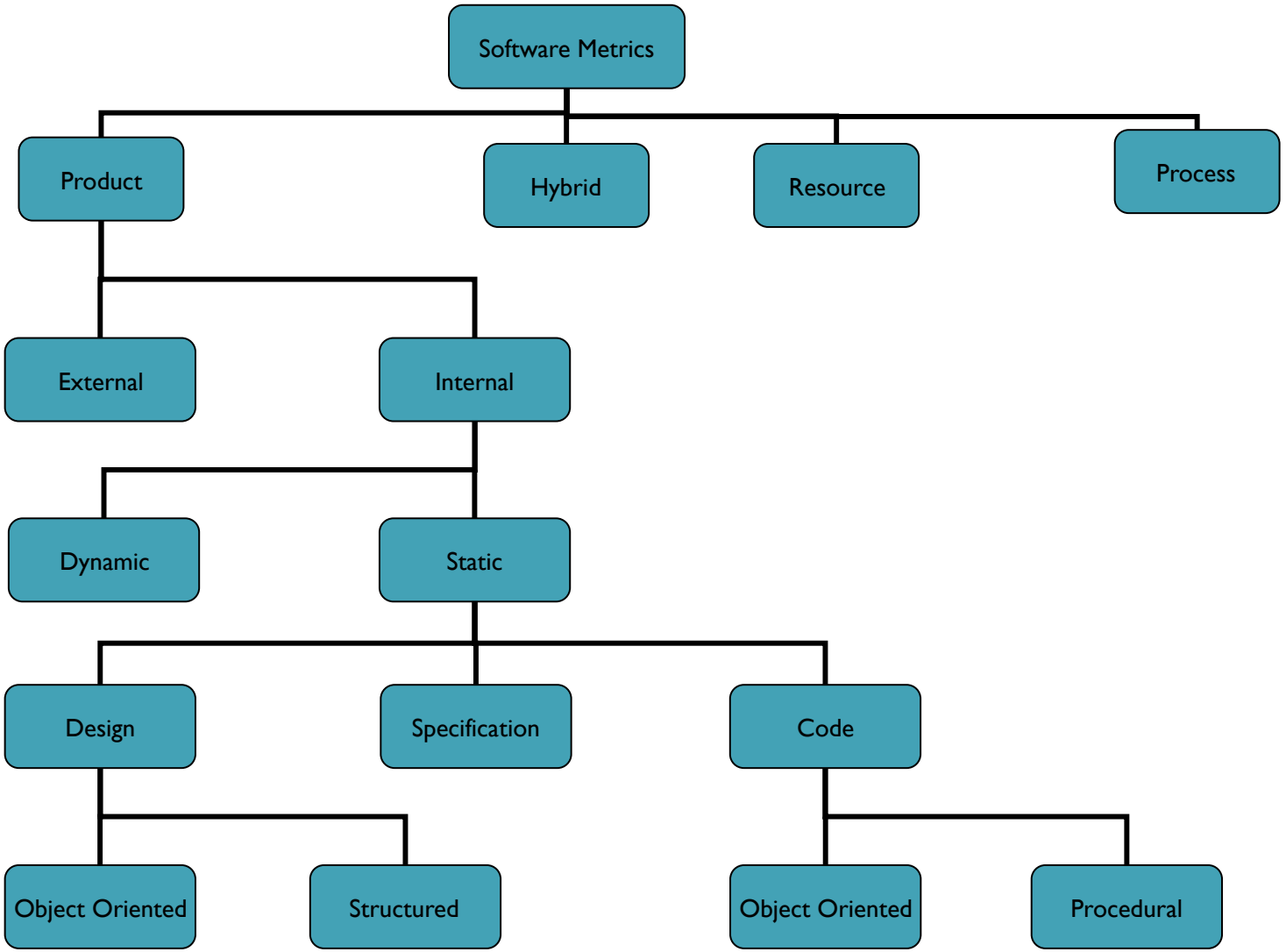
Metric Classification (1)

- Products
  - Explicit results of software development activities
  - Requirement specifications documents, design diagrams ,source code listings and the end product
- Processes
  - Activities related to software development life cycle
  - Time, effort and cost

Metric Classification(11)

- Resources
  - Available resources characteristics
  - Number of developers ,their skills and hardware reliability and performance
- Hybrid
  - Mixtures of product and process metrics
  - Cost per function point, and time to deliver and LOC

Software Metrics Taxonomy



External v/s Internal

- External metrics
  - Properties visible to the users
  - Not available till the late stages of the software development life cycle
  - Functionality, quality, reliability, maintainability
- Internal metrics
  - Address properties visible only to the development team
  - Cost, effort, LOC, speed, memory

Dynamic v/s Static

- Static
  - Collected from the static artifacts of the software such as specification
  - Documents, design diagrams and code listings

- Dynamic
  - Collected during the run-time of the software from its executable form
  - Extent of class usage, Dynamic Coupling and Dynamic Lack of Cohesion

Specification v/s Design

- Specification
  - Analyze the product specifications and provides early feedback about the developing software product
  - De Marco's Bang metric would distinguish a system whether the system is function-strong or data-strong
- Design
  - Measured at a bit later stage of the software development process
  - Helps refining the design
  - Henry and Kafura's information flow ,reuse ratio and coupling factor

Code Metric (1)

- Properties of the written code
- Size-Oriented Metrics
  - Size of the software produced
  - LOC - Lines Of Code
  - KLOC - 1000 Lines Of Code
  - SLOC – Statement Lines of Code (ignore whitespace)
- Typical Measures
  - Errors/KLOC, Defects/KLOC, Cost/LOC, Documentation Pages/KLOC

Code Metric (11)

- McCabe's Complexity Measures
  - McCabe's metrics are based on a control flow representation of the program
  - A program graph is used to depict control flow
  - Nodes represent processing tasks (one or more code statements)
  - Edges represent control flow between nodes
  - Applied to individual functions , modules , methods or classes within a program

McCabe's Cyclomatic Complexity

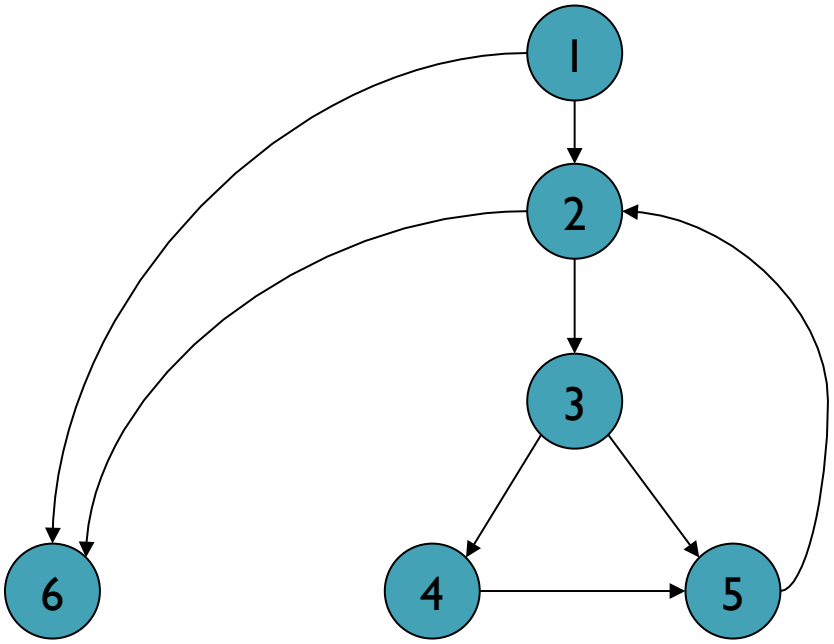
- Count of the number of linearly independent paths through the source code

- M = E − N + 2P where
  - M = cyclomatic complexity
  - E = the number of edges of the graph
  - N = the number of nodes of the graph
  - P = the number of connected components
  - For a single subroutine , P is always equal to 1 and increases with the number of sub-programs in question

Example

```
i = 0;
If (i<n-1) do
  j = i + 1;
  while (j<n) do
    if A[i]<A[j] then
      swap(A[i], A[j]);
  end do;
  i=i+1;
end do;
```

Flow Graph



Computing M

- M = 8 – 6 + 2 = 4
- Basic Set
  - 1, 6
  - 1, 2, 6
  - 1, 2, 3, 4, 5, 6
  - 1, 2, 3, 5,6

- Helps in determining the number of test cases that are necessary to achieve thorough test coverage of a particular module

Meaning

- M is the number of (enclosed) regions/areas of the planar graph
- Number of regions increases with the number of decision paths and loops
- A quantitative measure of testing difficulty and an indication of ultimate reliability
- Experimental data shows value of M should be no more then 10 - testing is very difficulty above this value

Metrics for the Object Oriented

- Weighted Methods per Class
- Method Hiding Factor (MHF)
- Method Inheritance Factor (MIF)
- Polymorphism Factor (PF)
- Coupling Factor (CF)

Weighted Methods per Class

$$WMC = \sum_{i=1}^{n} c_i$$

- c_i is the complexity (e.g. cyclomatic complexity) of each method in a class
- How much time and effort is required to develop and maintain the object
- The larger the number of methods in an object, the greater the potential impact on the children
- Objects with large number of methods are likely to be more application specific, limiting the possible reuse
- Classes with a large Weighted Methods Per Class value can often be refactored into two or more classes

Method Hiding Factor (MHF)

- Measure of the use of the information hiding concept that is supported by the encapsulation mechanism
- A very low MHF indicates an insufficiently abstracted implementation
- Conversely, a high MHF would indicate very little functionality
- MHF Usage:
  - Helps to maintain the modularity of the code
  - Reduce "side-effects" due to implementation refinement
  - Support a top-down approach
  - Test and integrate systems incrementally

Method Inheritance Factor (I)

- Measure of inheritance
- Depth of Inheritance Tree (DIT) is the maximum length from a node to the root/base class
- Lower level subclasses inherit a number of methods making behavior harder to predict
- Deeper trees indicate greater design complexity

Method Inheritance Factor (11)

- Number Of Children (NOC) is the number of subclasses immediately subordinate to a class
- Viewpoints
  - Depth is generally better than breadth in class hierarchy, since it promotes reuse of methods through inheritance
  - Classes higher up in the hierarchy should have more sub-classes then those lower down
  - NOC gives an idea of the potential influence a class has on the design classes with large number of children may require more testing

Method Inheritance Factor (111)

$$MIF = \frac{\sum_{i=1}^{n} M_i(C_i)}{\sum_{i=1}^{n} M_a(C_i)} .$$

- $M_i(C_i)$ is the number of methods inherited and not overridden in C_i
- $M_a(C_i)$ is the number of methods that can be invoked with C_i

Polymorphism Factor (PF)

- Polymorphism potential is measured
- The number of methods that redefines inherited methods, divided by maximum number of possible distinct polymorphic situations

$$PF = \frac{\sum_i M_o(C_i)}{\sum_i \left[M_n(C_i) * DC(C_i) \right]}.$$

- $M_n()$ is the number of new methods
- $M_o()$ is the number of overriding methods
- DC() number of descendent classes of a base class
- Allows refinement of the class taxonomy

Coupling Factor

- Coupling factor represents the number of collaborations between two classes (fan-out of a class)
  - the number of other classes that are referenced in the class
- As collaboration increases reuse decreases
- High fan-outs represent class coupling to other classes/ objects and thus are undesirable
- High fan-ins represent good object designs and high level of reuse
- Not possible to maintain high fan-in and low fan outs across the entire system

Using Metrics

- The Process
  - Select appropriate metrics for problem
  - Utilized metrics on problem
  - Assessment and feedback

- Formulate
- Collect
- Analysis
- Interpretation
- Feedback

Metric tools

- McCabe & Associates (founded by Tom McCabe, Sr.)
  - The Visual Quality ToolSet
  - The Visual Testing ToolSet
  - The Visual Reengineering ToolSet
- Metrics calculated
  - McCabe Cyclomatic Complexity
  - McCabe Essential Complexity
  - Integration Complexity
  - Lines of Code
  - Halstead

CCCC

- A metric analyzer C, C++, Java, Ada-83, and Ada-95 (by Tim Littlefair of Edith Cowan University, Australia)
- Metrics calculated
  - Lines Of Code (LOC)
  - McCabe's cyclomatic complexity
  - Weighted Method per Class, Number of Children, Depth of Inheritance and Coupling
- Generates HTML and XML reports
- Freely available
- http://cccc.sourceforge.net/

Jmetric

- OO metric calculation tool for Java code (by Cain and Vasa for a project at COTAR, Australia)
- Requires Java 1.2 (or JDK 1.1.6 with special extensions)
- Metrics
  - Lines Of Code per class (LOC)
  - Cyclomatic complexity
  - LCOM (by Henderson-Seller)
- Availability
  - Is distributed under GPL
- http://www.it.swin.edu.au/projects/jmetric/products/jmetric/

JMetric tool result

Pr Project | P Package | C Class | M Method | V Variable

```
PACKAGE: com.rolemodelsoft.drawlet.awt

Classes: 9
Public Classes: 9

Lines Of Code: 442
Statement Count: 264

Methods: 101
Public Methods: 92
```

Metric	Average	Std. Dev	Median	Mode	Max	Min	Skew
Class Stats							
Lines of C...	49.1	34.1	27.0	26.0	112.0	19.0	1.9
Statements	29.3	22.6	16.0	15.0	78.0	9.0	1.8
LCOM	0.8	0.2	0.8	0.5	1.0	0.5	-1.0
No. Metho...	11.2	8.7	8.0	8.0	27.0	3.0	1.1
Collaborat...	8.7	4.0	9.0	4.0	17.0	4.0	-0.3
Public							
Methods	10.2	8.0	8.0	3.0	25.0	3.0	0.8
Variables	0.0	0.0	0.0	0.0	0.0	0.0	0.0
No Scope							
Methods	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Variables	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Methods							
Lines of C...	4.0	4.0	2.0	2.0	24.0	1.0	1.5
Statements	2.6	3.2	1.0	1.0	16.0	0.0	1.5
Cyclomati...	1.3	0.6	1.0	1.0	4.0	1.0	1.6
Collaborat...	1.8	1.5	1.0	1.0	9.0	0.0	1.6

More tools on Internet

- A Source of Information for Mission Critical Software Systems, Management Processes, and Strategies

http://www.niwotridge.com/Resources/PM-SWEResources/SWTools.htm

- Defense Software Collaborators (by DACS)

http://www.thedacs.com/databases/url/key.hts?keycode=3

- Metrics tools for C/C++ (Christofer Lott)

http://www.chris-lott.org/resources/cmetrics/

- GEN++ is an application-generator for creating code analyzers for C++ programs

http://www.cs.ucdavis.edu/~devanbu/genp/down-red.html

References

- Software Metrics - A Taxonomy (Faculty of Computers and Information ,Cairo University ,Cairo ,Egypt)
- "Design Metrics for Object-Oriented Software Systems“by Fernando Brito e Abreu, INESC/ISEG
- S. Conte, H. Dunsmore, and V. Shen , "Software Engineering Metrics And Models, 1ST edition." ,Benjamin/Cummings, Menlo Park, CA. , 1986
- R. S. Chidamber and C. F. Kemerer , "A Metrics Suite For Object Oriented Design" , IEEETransactions on Software Engineering, Vol. 20, No. 6. , 1994

Questions?