Object-Oriented Software Metrics

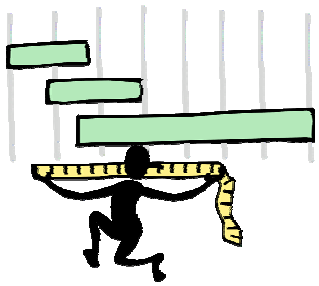
Radu Marinescu

LOOSE Research Group
Politehnica University of Timişoara
radu.marinescu@cs.upt.ro

loose.upt.ro

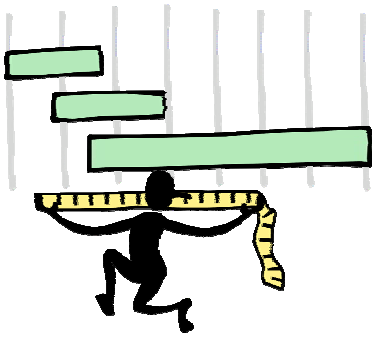
Outline

What are Metrics?



- Functions, that assign a precise numerical value
- to
  - Products (Software) ,
  - Resources (Staff, Tools, Hardware) or
  - Processes (Software development).

Object-Oriented Product Metrics



- Size & Structural Complexity
- Inheritance
- Coupling
- Cohesion

Weighted Method Count (WMC)

- Definition [Chidamber&Kemerer, 1994]:

$$WMC = \sum_{i=1}^{n} c_i$$ where c_i = complexity of method m_i

- Interpretation:
  - Time and effort for **maintenance**
  - The higher the WMC for a class, the higher the influence on the subclasses
  - A high WMC reduces the **reuse** probability for the class

GOOD: Metric is configurable!

BAD: Interpretation can't directly lead to improvement action!

Depth of Inheritance Tree (DIT)

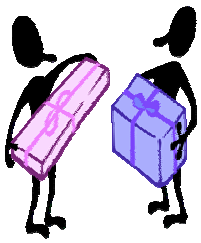
- Definition [Chidamber&Kemerer, 1994]:
  - depth of a class in the inheritance graph

- Interpretation:
  - the higher DIT, the lower the **understandability** of the class
  - higher DIT, class more **complex**
    - harder to test
  - the higher DIT, the higher the **potential reuse** from the superclasses

BAD: Nothing about real reuse!

Change Dependency Between Classes (CDBC)

- Definition [Hitz&Montazeri, 1996]:
  - number of methods in a ***client-class*** (CC) that depend on a ***server-class*** (SC)
- Characteristics:
  - defined on a pair of classes
  - stability of the server class
  - differentiates between types of coupling
- Interpretation:
  - the higher CDBC, the bigger the ***maintenance*** impact on CC, by a change in SC

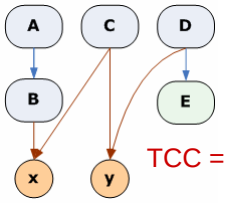


CDBC differentiates between types of coupling

	α = Number of methods of CC potentially affected by a change
SC is not used by CC at all	0
SC is the class of an instance variable of CC	n
Local variables of type SC are used within j methods of CC	j
SC is a superclass of CC	n
SC is the parameter type of j methods of CC	j
CC accesses a global variable of class SC	n

Tight Class Cohesion (TCC)

- Definition [Bieman&Kang, 1995]:
  - the relative number of method-pairs that access an attribute of the class
- Example:



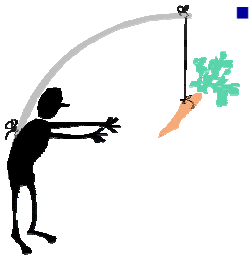
TCC = 2 / 10 = 0.2

- Interpretation:
  - higher TCC ➔ tighter the connection between the methods
  - lower TCC ➔ probably class implements more than one functionality

GOOD: Interpretation can lead to improvement action!

GOOD: Ratio values allow comparison between systems!

Obstacles in Using Metrics



- Interpretation of metrics is hard
  - many confusing and redundant definitions
  - issue of thresholds
    - need statistical data
  - hard to compare the results
    - normalize!
- Applying metrics is hard
  - issue of granularity
    - metrics need to be used in combination
      - quality models
      - detection strategies
      - polymetric views

Issue of thresholds exemplified

Metric	Value
LOC	35.000
NOM	3.600
NOC	380

- Let's play a game:
  - Want brief overview of the code of an OO system never seen before
  - Want to find out how hard it will be to understand the code

We need statistical data for thresholds

Several questions remain unanswered...

- Is it "normal" to have...
  - ...380 classes in a system with 3.600 methods?
  - ...3.600 methods in a system with 35.000 lines of code?

➔ What means NORMAL?

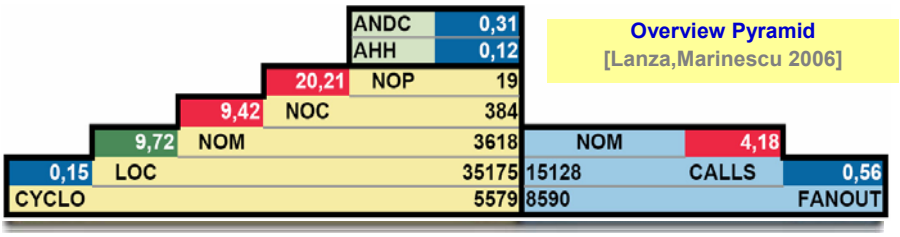
➔ i.e. how do we compare with other projects?

- What about the hierarchies ? What about coupling?

1. We need means of comparison. Thus, proportions are important!

- Collect further relevant numbers; especially coupling and use of inheritance

Issue of thresholds



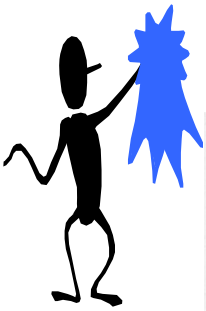
Overview Pyramid
[Lanza,Marinescu 2006]

- Interpretation based on a statistically relevant collection of data
  - collected for Java and C++
  - over 80 systems

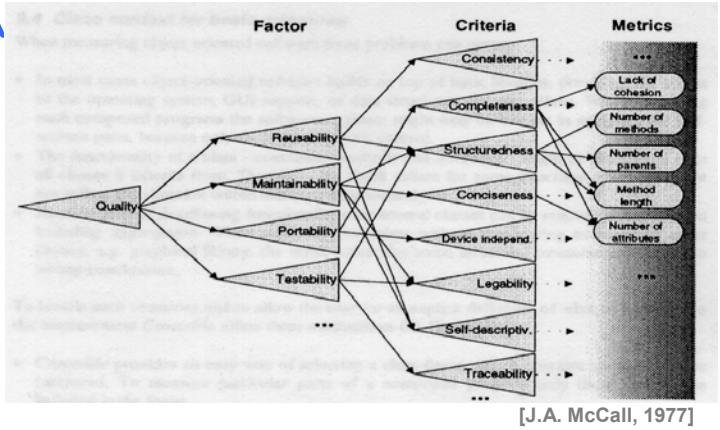
close to AVERAGE
close to HIGH
close to LOW

	Java			C++		
Metric	Low	Average	High	Low	Average	High
CYCLO/Line of code	0.16	0.20	0.24	0.20	0.25	0.30
LOC/Operation	7	10	13	5	10	16
NOM/Class	4	7	10	4	9	15
NOC /Package	6	17	26	3	19	35
CALLS/Operation	2.01	2.62	3.2	1.17	1.58	2
FANOUT /Call	0.56	0.62	0.68	0.20	0.34	0.48
ANDC	0.25	0.41	0.57	0.19	0.28	0.37
AHH	0.09	0.21	0.32	0.05	0.13	0.21

The need to aggregate metrics



Quality Models to correlate quality criteria with concrete measures



[J.A. McCall, 1977]

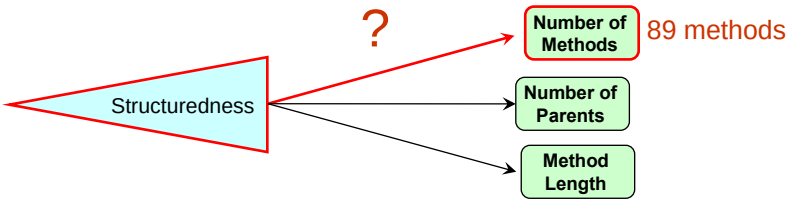
Factor-
Criteria-
Metrics
model

Metrics are too fine-grained indicators

What do we expect from quality models?

- **diagnosis**, **location** and **treatment hints** for quality problems

FCM **locates** classes and methods with abnormal metric values



In FCM models it is hard to find the proper treatment for quality problems because abnormal metric values are rather symptoms than causes of poor quality

Metrics should be used in a goal-oriented fashion

Goal-Question-Metric Approach

[Basili&Rombach, 1988]

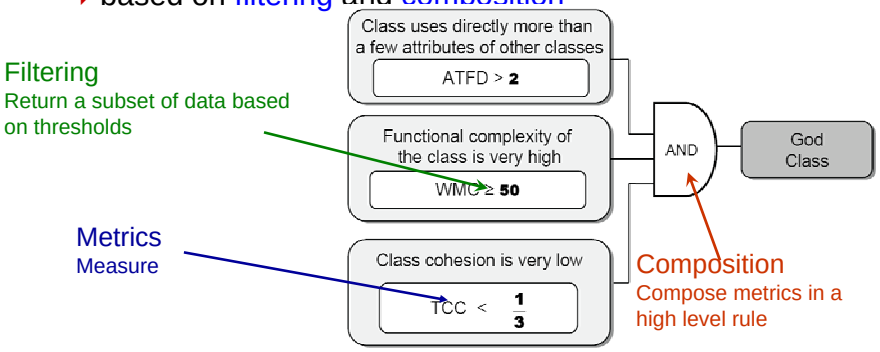
- Define a **Goal**
  - How efficient is the ACME tool
- Formulate **Questions**
  - Who uses the ACME tool?
  - How high is productivity/quality with/without the ACME tool?
- Find suitable **Metrics**
  - Percent of developers that use the ACME tool
  - Experience with ACME
  - Size, complexity, solidity , … of code

Goal-oriented aggregation of metrics

Detection Strategies
[Marinescu 2002, 2004]

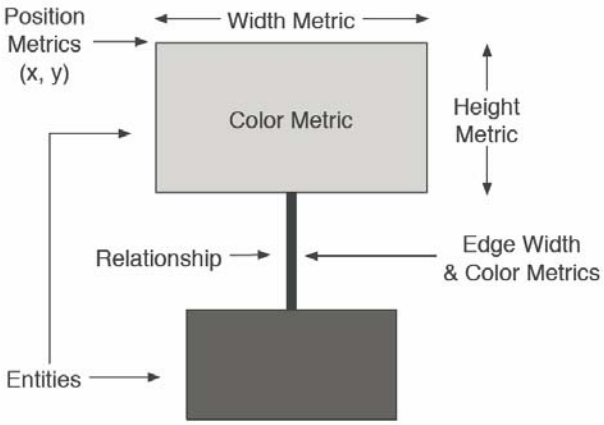
Metrics-based rules that capture quality aspects

- based on filtering and composition

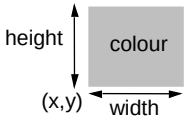


Combine Metrics in a Visual Manner

Polymetric Views
[Lanza,Ducasse 2003]

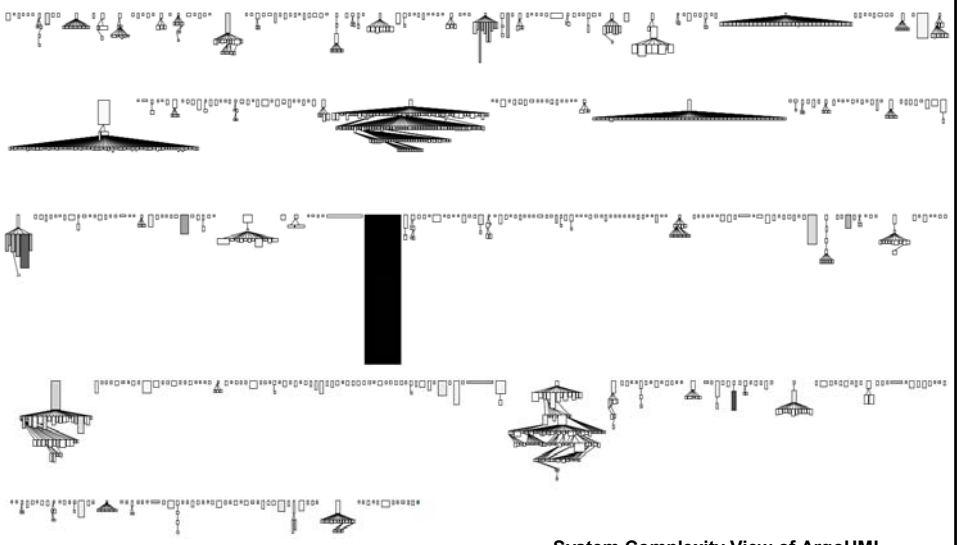


Use *simple* metrics and layout algorithms.



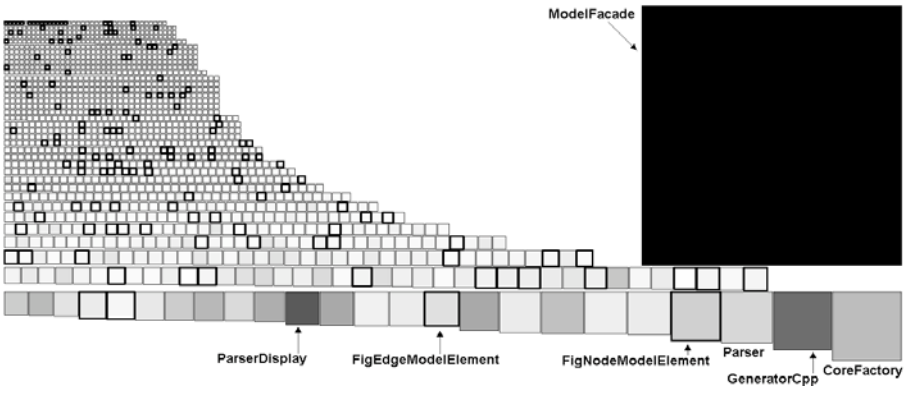
Visualize up to 5 metrics per node

A Picture is Worth a Thousand Words...



System Complexity View of ArgoUML

System Hotspots

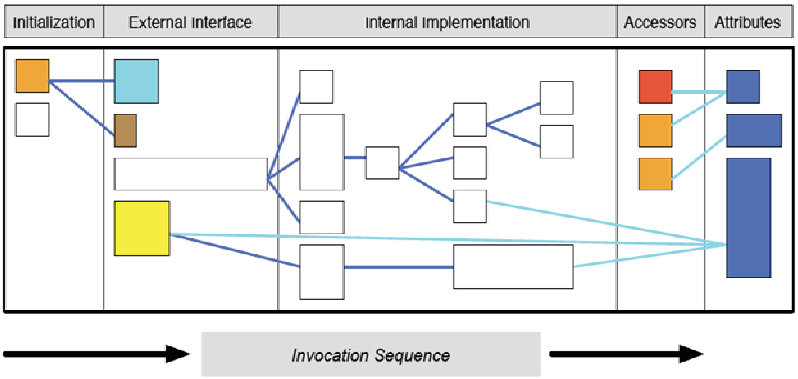


- width = height = NOM
- gray-level = LOC

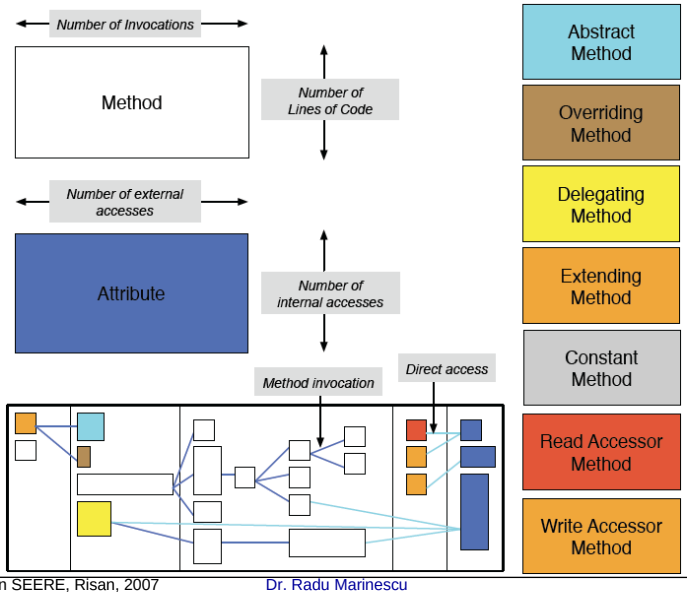
Quickly “Reading” Classes

Class Blueprint
[Lanza,Ducasse 2001]

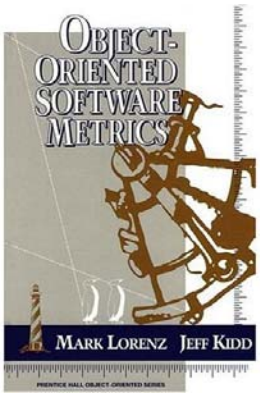
- Visualization Technique
  - serves as code inspection technique
  - reduces the amount of code that must be read



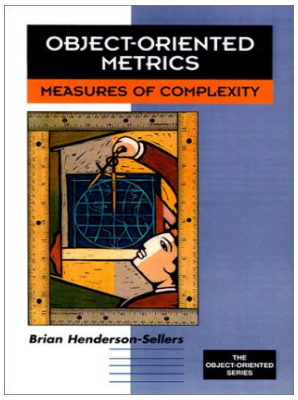
The Class Blueprint



Books on object-oriented metrics



Addison-Wesley, 1994



Prentice-Hall, 1996

Springer, 2006

Instead of conclusions....

- What metrics do we use?
  - It depends…on our measurement goals
- What information to retrieve?
  - It depends… on our objectives
- What entities do we measure?
  - It depends…on the language

Can we understand the beauty of a painting by…
… measuring its frame or counting the colors ?

DISCLAIMER:
Metrics are **not** enough to understand and evaluate design!