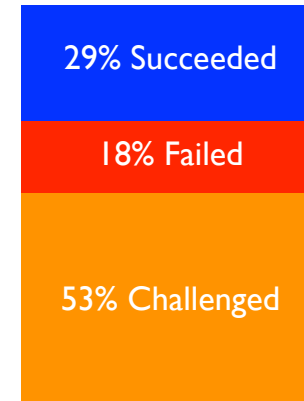


Metrics and Problem Detection

(slides courtesy of Tudor Girba - Uni. Bern www.tudorgirba.com)

Software is complex.

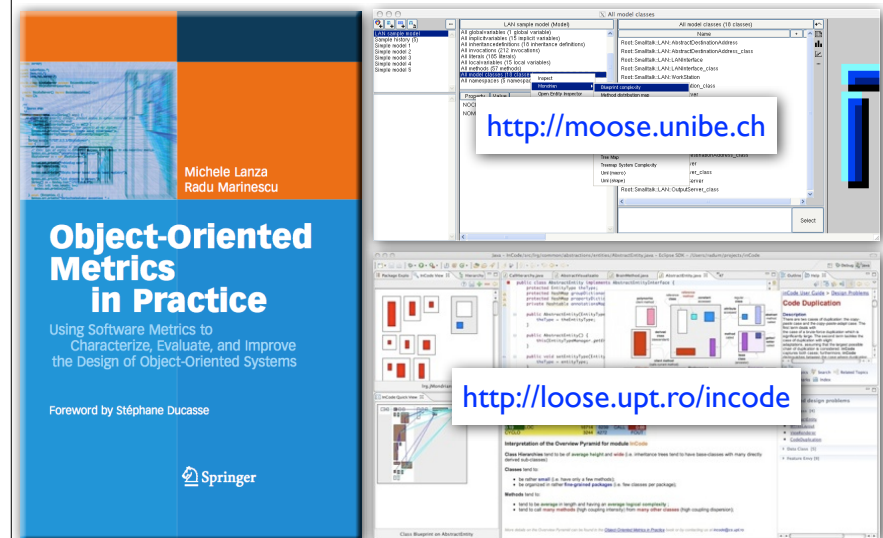


The Standish Group, 2004

How to judge its quality?



A large system contains lots of details.



1

Metrics

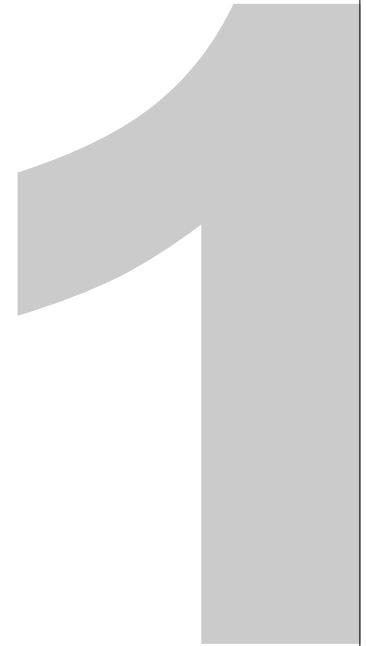
2

Design
Problems

3

Code Duplication

Metrics



You **cannot control**
what you **cannot measure.**

Tom de Marco

Metrics are functions that assign **numbers** to
products, processes and **resources.**

Software metrics are measurements which relate to software systems, processes or related documents.

Metrics compress system traits into numbers.

Let's see some examples...

Examples of size metrics

NOM - number of methods

NOA - number of attributes

LOC - number of lines of code

NOS - number of statements

NOC - number of children

McCabe cyclomatic complexity (**CYCLO**) counts the number of independent paths through the code of a function.

McCabe, 1977

✓ it reveals the minimum number of tests to write

✗ interpretation can't directly lead to improvement action

Weighted Method Count (**WMC**) sums up the complexity of class' methods (measured by the metric of your choice; usually CYCLO).

Chidamber, Kemerer, 1994

✓ it is configurable, thus adaptable to our precise needs

✗ interpretation can't directly lead to improvement action

Depth of Inheritance Tree (**DIT**) is the (maximum) depth level of a class in a class hierarchy.

Chidamber, Kemerer, 1994

✓ inheritance is measured

✗ only the potential and not the real impact is quantified

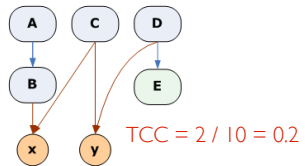
Coupling between objects (**CBO**) shows the number of classes from which methods or attributes are used.

Chidamber, Kemerer, 1994

✓ it takes into account real dependencies not just declared ones

✗ no differentiation of types and/or intensity of coupling

Tight Class Cohesion (**TCC**) counts the relative number of method-pairs that access attributes of the class in common.



Bieman, Kang, 1995

- ✓ interpretation can lead to improvement action
- ✓ ratio values allow comparison between systems

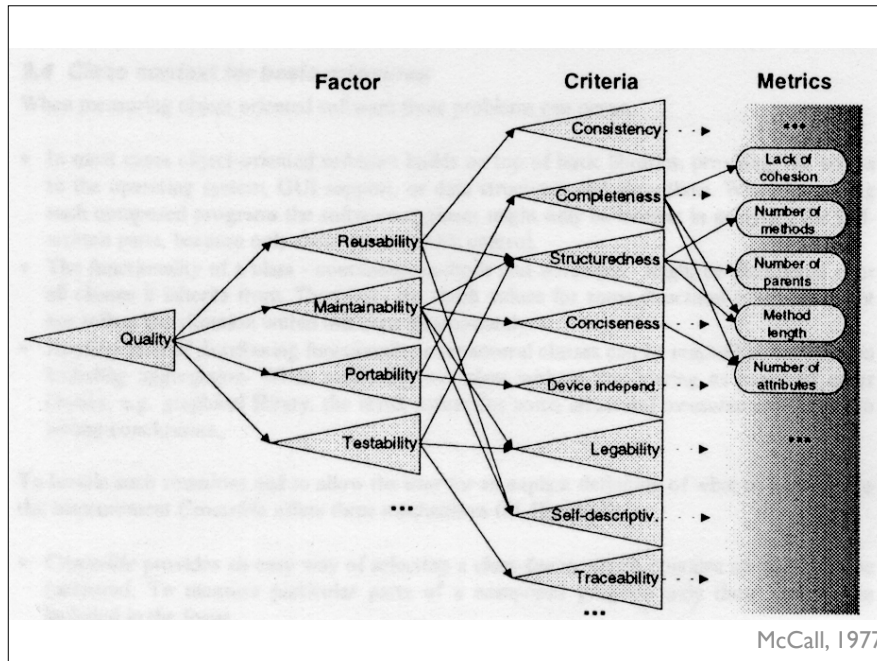
Access To Foreign Data (**ATFD**) counts how many attributes from other classes are accessed directly from a measured class.

Marinescu 2006



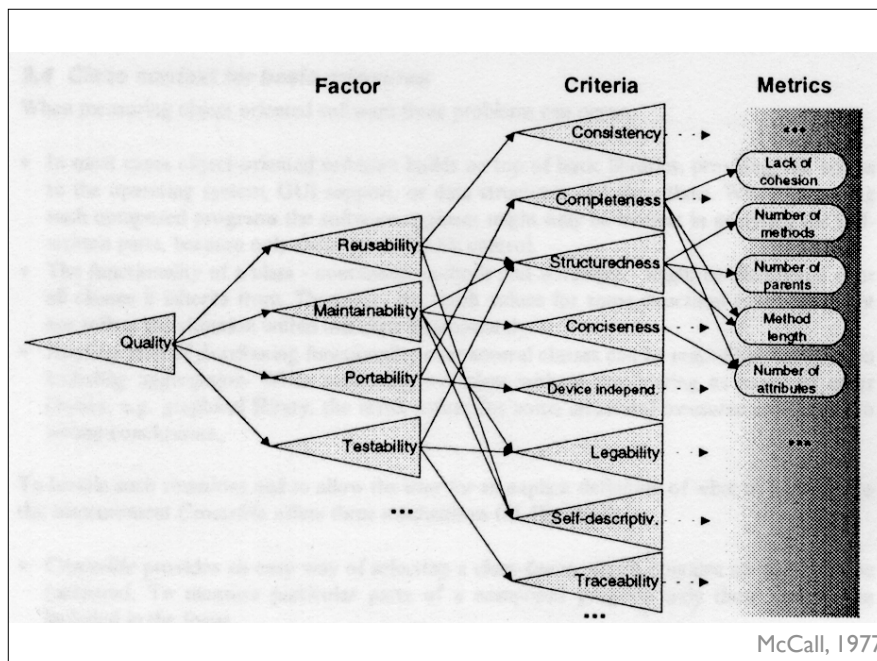
Design Problems





Metrics Assess and Improve Quality!

Really?



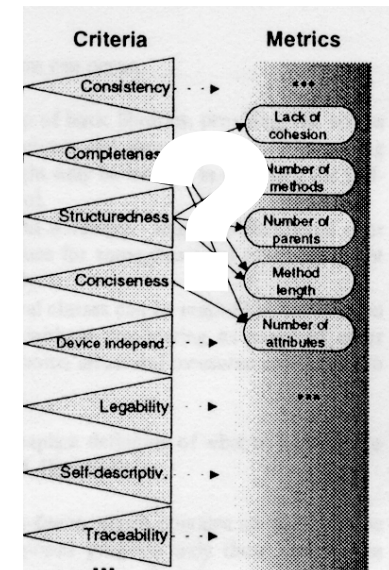
Problem 1: metrics granularity

capture **symptoms**, not causes of problems

in **isolation**,
they don't lead to **improvement** solutions

Problem 2: implicit mapping

we don't reason in terms of **metrics**,
but in terms of **design principles**



2 big obstacles in using metrics:

Thresholds make metrics hard to interpret

Granularity make metrics hard to use in isolation

How do I improve code?

Quality is more than 0 bugs.

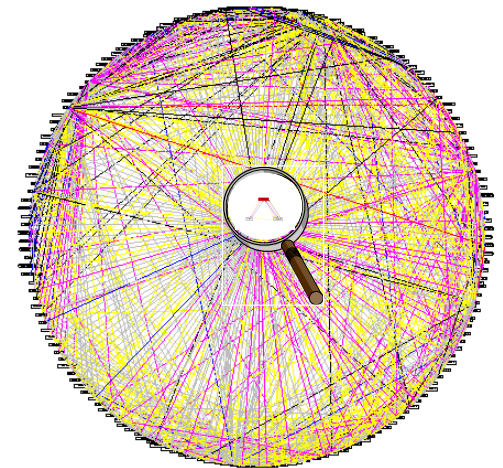
Breaking design principles, rules and best practices

deteriorates the code;

it leads to design problems.

Imagine changing just a small design fragment

and 33%
of all classes
would require changes



Design problems are expensive
frequent
unavoidable

How to detect and eliminate them?

Metrics should be used in a goal-oriented fashion

Goal-Question-Metric Approach [Basili&Rombach, 1988]

- Define a **Goal**
- Formulate **Questions**
- Find suitable **Metrics**

God Classes tend to centralize the intelligence of the system, to do everything and to use data from small data-classes.

Riel, 1996

God Classes tend
to centralize the intelligence of the system,
to do everything and
to use data from small data-classes.

God Classes

centralize the intelligence of the system,
do everything and
use data from small data-classes.

God Classes

are complex,
are not cohesive,
access external data.

God Classes

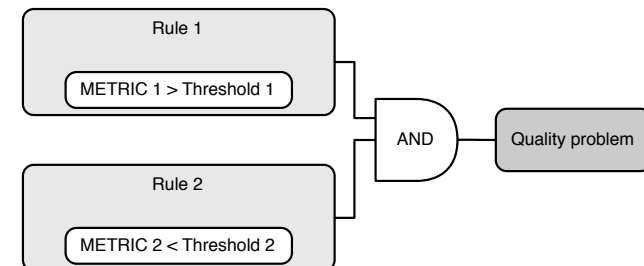
are complex,
are not cohesive,
access external data.

WMC is high
TCC is low
ATFD more than few

Compose metrics into queries using
logical operators

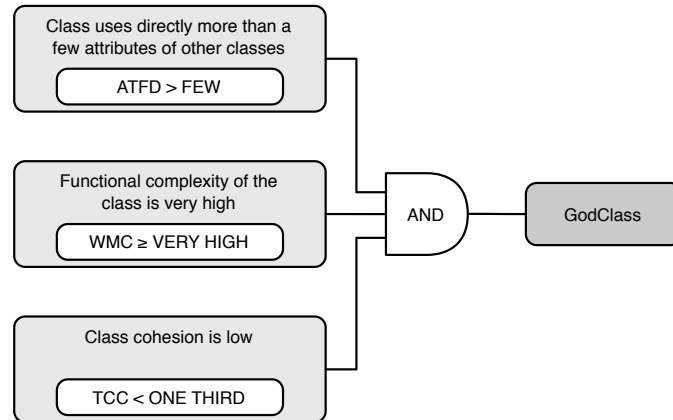
Detection Strategies are metric-based queries to
detect design flaws.

Lanza, Marinescu 2006



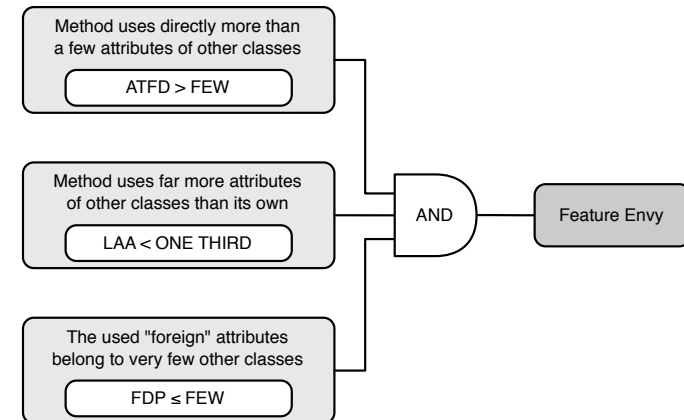
A **God Class** centralizes too much intelligence in the system.

Lanza, Marinescu 2006



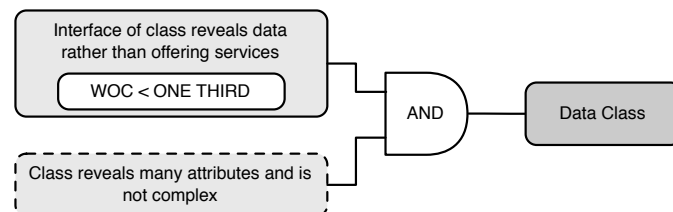
An **Envious Method** is more interested in data from a handful of classes.

Lanza, Marinescu 2006



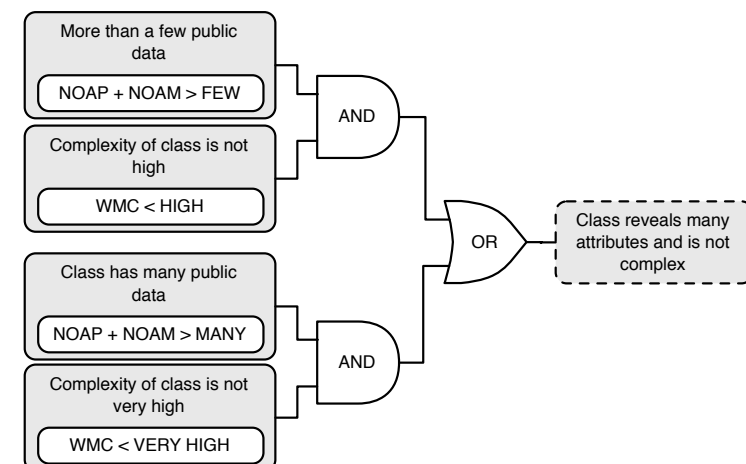
Data Classes are dumb data holders.

Lanza, Marinescu 2006



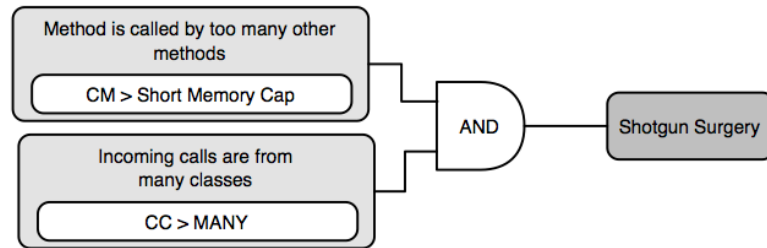
Data Classes are dumb data holders.

Lanza, Marinescu 2006

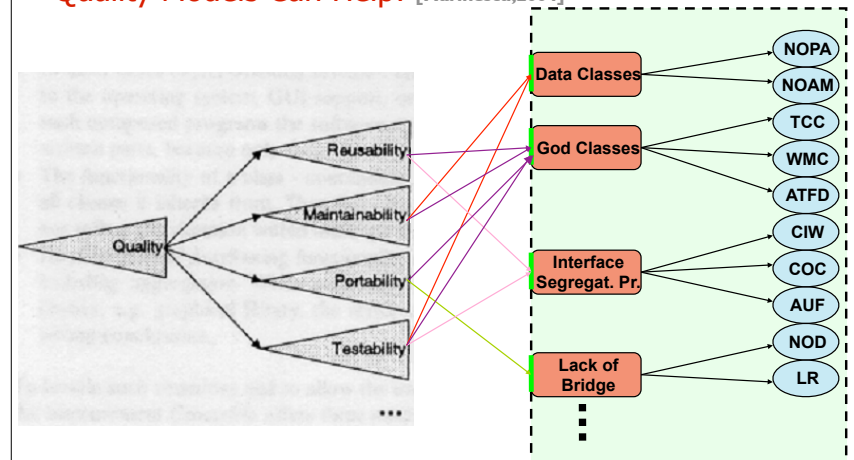


Shotgun Surgery depicts that a change in an operation triggers many (small) in a lot of different operation and classes.

Lanza, Marinescu 2006



Quality Models Can Help! [Marinescu,2004]



Quality — communicates with — **Principles, rules, heuristics**
decomposed in **Factors** quantified in **Detection Strategies**

Dr. Radu Marinescu

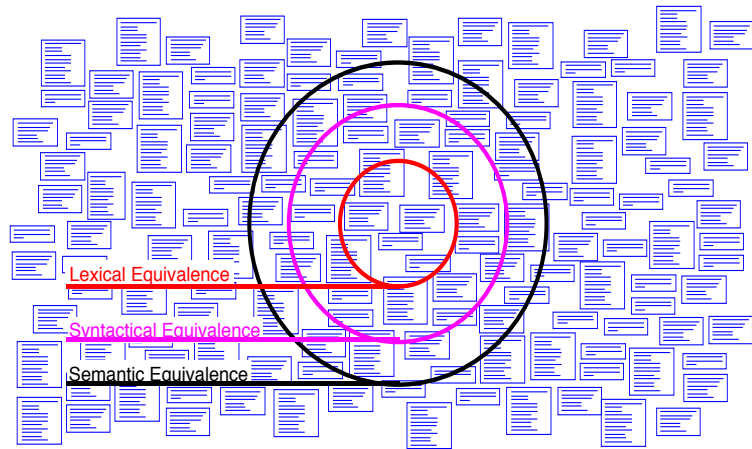
128

Code Duplication

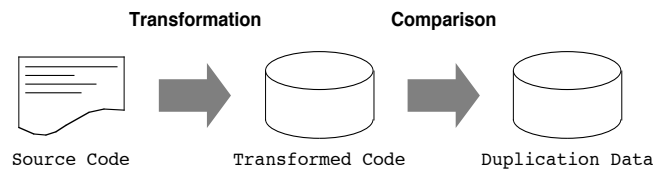
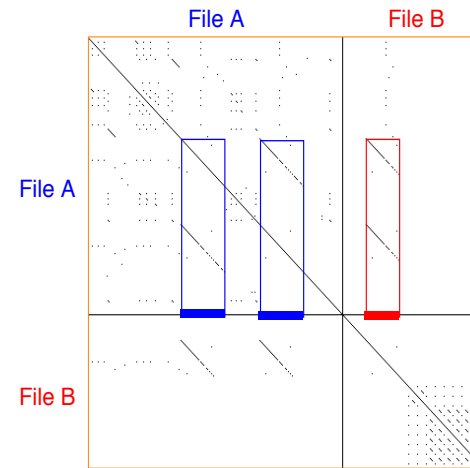
What is Code Duplication?

What are the problems of it?

Code Duplication Detection



Visualization of Copied Code Sequences



Author	Level	Transformed Code	Comparison Technique
Ducasse 99	Lexical	Normalized Strings	String-Matching
Mayrand 96	Syntactical	Metrics Tuples	Discrete comparison
Kontogiannis 97	Syntactical	Metrics Tuples	Euclidean distance
Baxter 98	Syntactical	AST	Tree-Matching

Noise Elimination

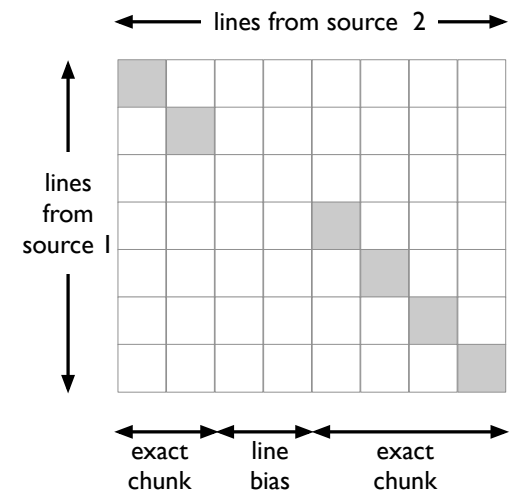
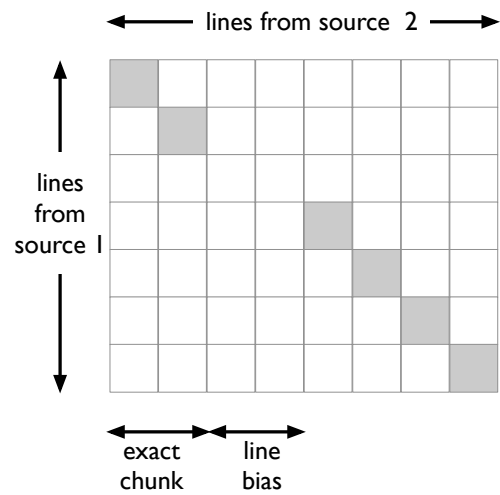
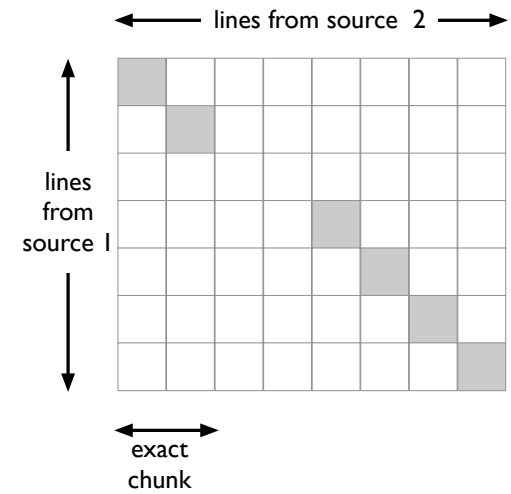
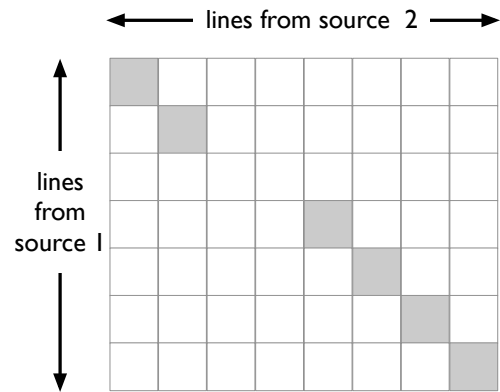
```

...
//assign same fastid as container
fastid = NULL;
const char* fidptr = get_fastid();
if(fidptr != NULL) {
    int l = strlen(fidptr);
    fastid = newchar[ l + 1 ];
}
    
```

→

```

fastid=NULL;
constchar*fidptr=get_fastid();
if(fidptr!=NULL)
    intl=strlen(fidptr)
    fastid = newchar[l+]
    
```

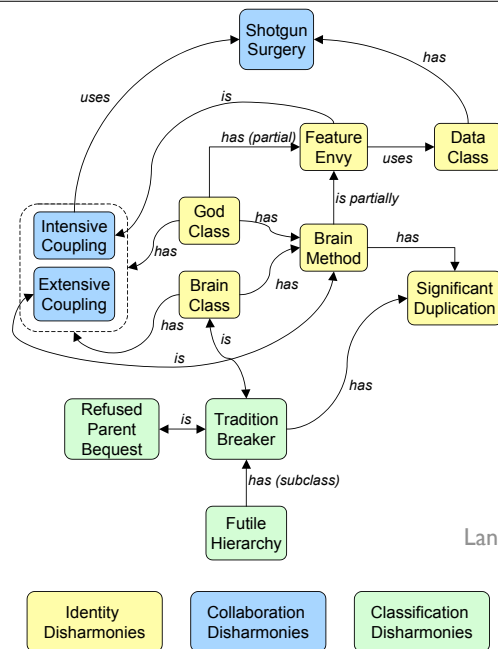
- It is the largest possible duplication chain uniting all exact clones that are close enough to each other.
- The duplication is large enough.

```

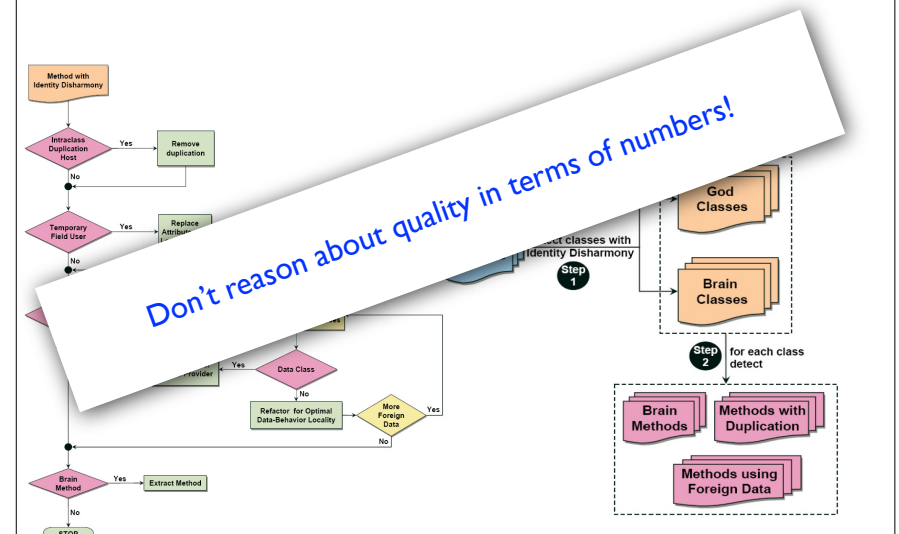
graph LR
    A["Duplication Chain has at least a size of two relevant exact clones  
SDC ≥ 2x(FEW+1)+1"] --> AND
    B["Exact Clones are longer than a few lines of code  
SEC > FEW"] --> AND
    C["Distance between clones is not more than a few lines of code  
LB ≤ FEW"] --> AND
    AND --> D["Significant Duplication Chain"]
    E["Significant 'standalone' exact clone  
SEC > AVERAGE(LOC/Operation)"] --> OR
    D --> OR
    OR --> F["Significant Duplication"]
  
```

2 Design Problems

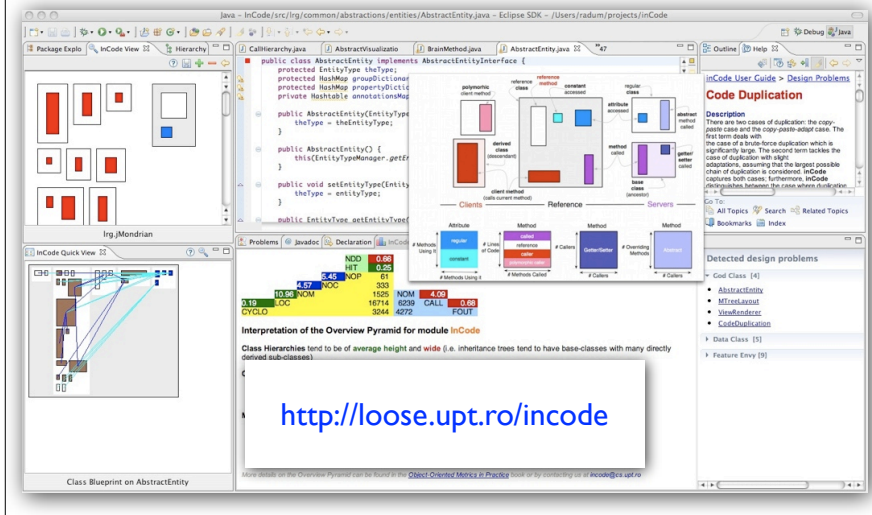
3 Code Duplication



Follow a clear and repeatable process



QA is part of the the Development Process



The screenshot displays the InCode IDE interface. The main window shows a Java class hierarchy for `AbstractEntity`. The left sidebar contains a package explorer and a hierarchy view. The bottom left shows a class blueprint for `AbstractEntity`. The bottom right displays a code duplication report with a table of metrics.

Metric	Value
LOC	1836
CYCLO	0.19
NOM	1522
CAIT	0.69
FOUT	3044

Interpretation of the Overview Pyramid for module InCode

Class Hierarchies tend to be of average height and wide (i.e. inheritance trees tend to have base-classes with many directly derived sub-classes).

<http://loose.upt.ro/incode>

Instead of Conclusions...



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!