b
UNIVERSITÄT
BERN

Software Metrics and Problem Detection

Mircea Lungu

Roadmap

- **Measurements**
- Software Metrics
  - Size / Complexity Metrics
  - Quality Metrics
  - Schedule / Cost
- Metric-Based Problem Detection
  - Detecting Outliers
  - Encoding Design Problems
- Discussion

Measurements

A measurement is a mapping

- domain
- range
- rules

A measure is a numerical value or a symbol assigned during mapping

In Software:
measurements = **metrics**

Estimation of quantity owes its existence to Measurement Calculation to Estimation of quantity Balancing of chances to Calculation and Victory to Balancing of chances.

Measurement Scales

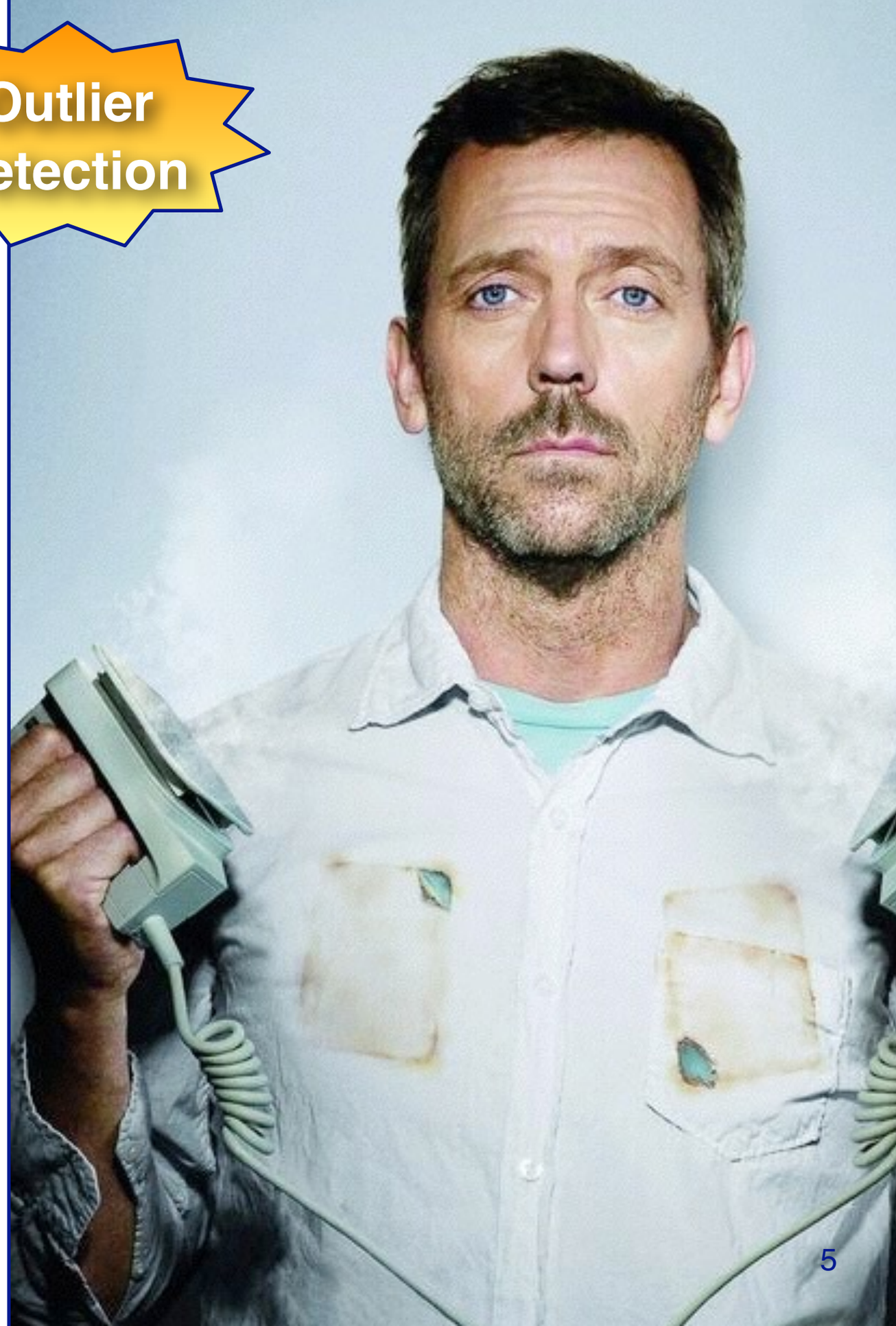
- Nominal
- Ordinal
- Interval
- Ratio

- Analysis should take scales into account

Estimation of quantity owes its existence to Measurement Calculation to Estimation of quantity Balancing of chances to Calculation and Victory to Balancing of chances.

Medical Markers are used in diagnositcs based on statistical data

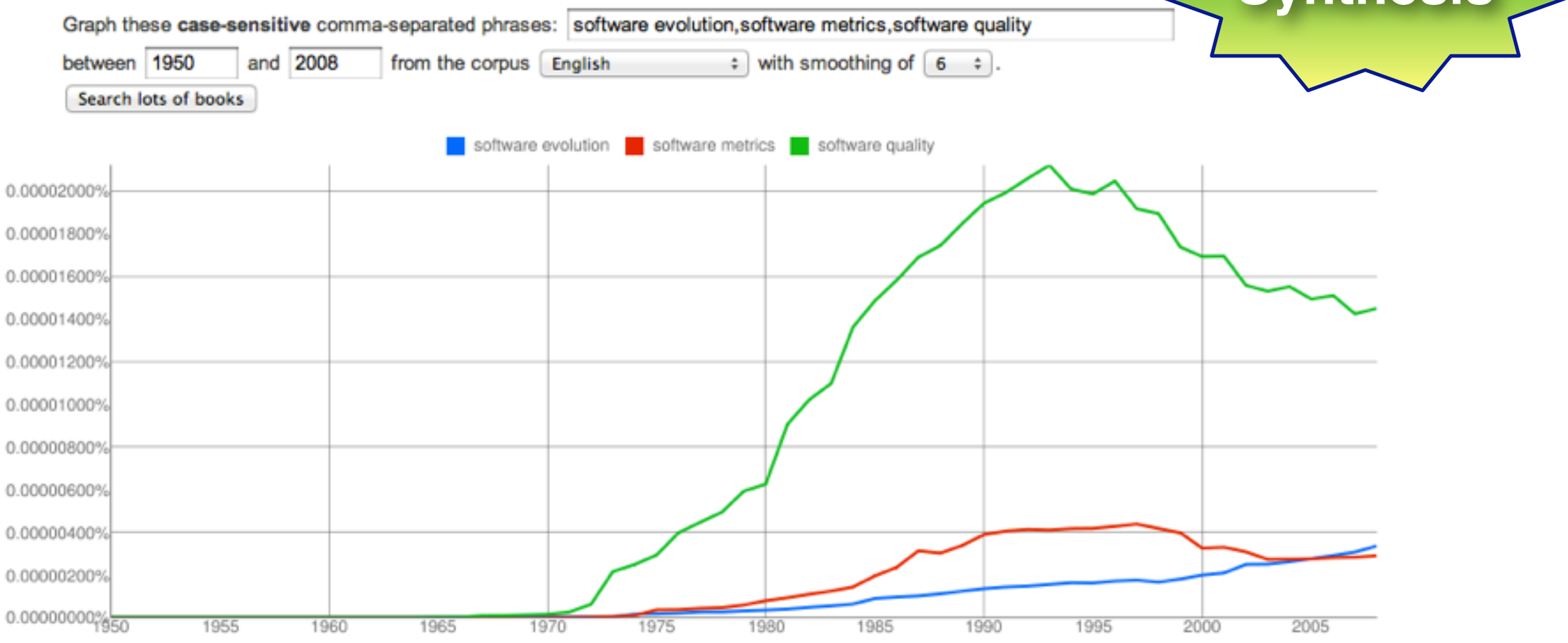
- Potassium Levels
- Red Blood Cell Count
- Glucose Levels
- etc.



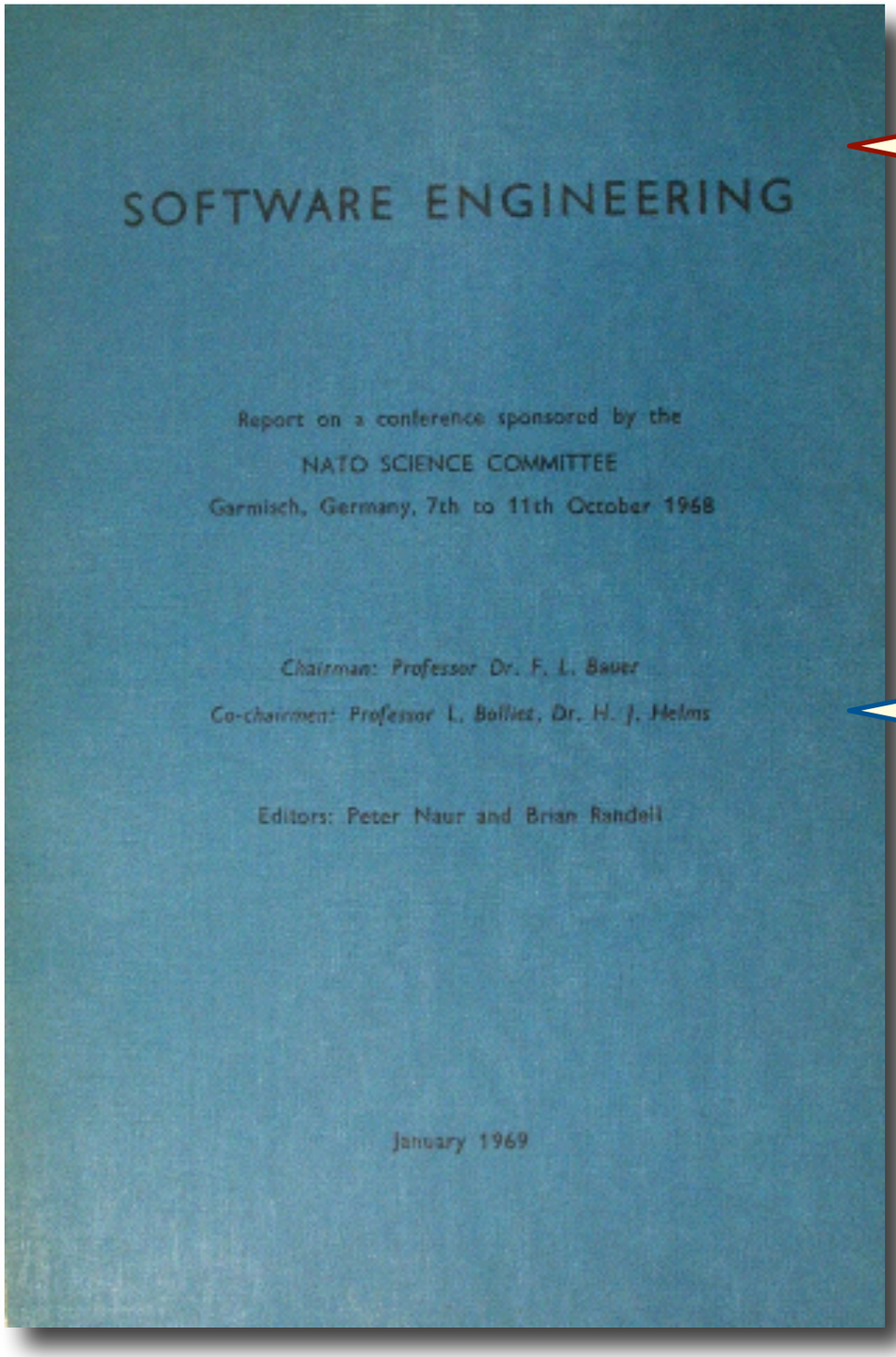
Google Measures N-gram Frequencies

> What do you do when you want to digitize and report about 5 million books but can not because of copyright?

Synthesis



Can you assess unknown code without reading it?



Fraser:
One of the problems that is central to the software production process is to identify the nature of progress and **to find some way of measuring it**.

McIlroy:
In programming efforts [...] clarity and style seem to count for nothing — the only thing that counts is whether the program works when put in place. It seems to me that it is important that we should **impose these types of aesthetic standards**.

Roadmap

- Measurements
- **Software Metrics**
  - Size / Complexity Metrics
  - Quality Metrics
  - Schedule / Cost
- Metric-Based Problem Detection
  - Detecting Outliers
  - Encoding Design Problems
- Discussion

Roadmap

- Measurements
- Software Metrics
  - **Size / Complexity Metrics**
  - Quality Metrics
  - Schedule / Cost
- Metric-Based Problem Detection
  - Detecting Outliers
  - Encoding Design Problems
- Discussion

Size Measures

LOC
NOM
NOA
NOC
NOP
... etc.

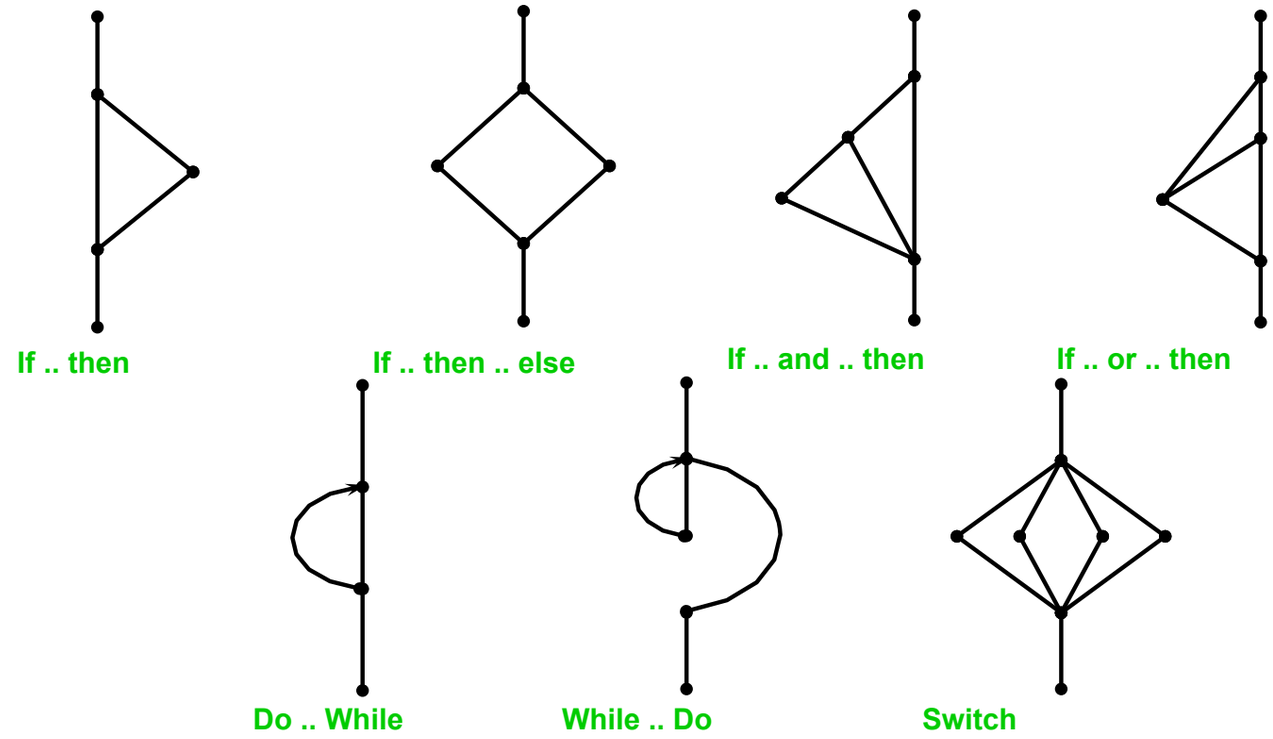
Lorenz, Kidd, 1994
Chidamber, Kemerer, 1994

Cyclomatic Complexity (CYCLO)

The number of independent linear paths through a program.

(McCabe ’77)

+ Measures minimum effort for testing



Weighted Methods per Class (WMC)

The complexity of a class by summing the complexity of its methods, usually using CYCLO.

(Chidamber & Kemerer ’94)

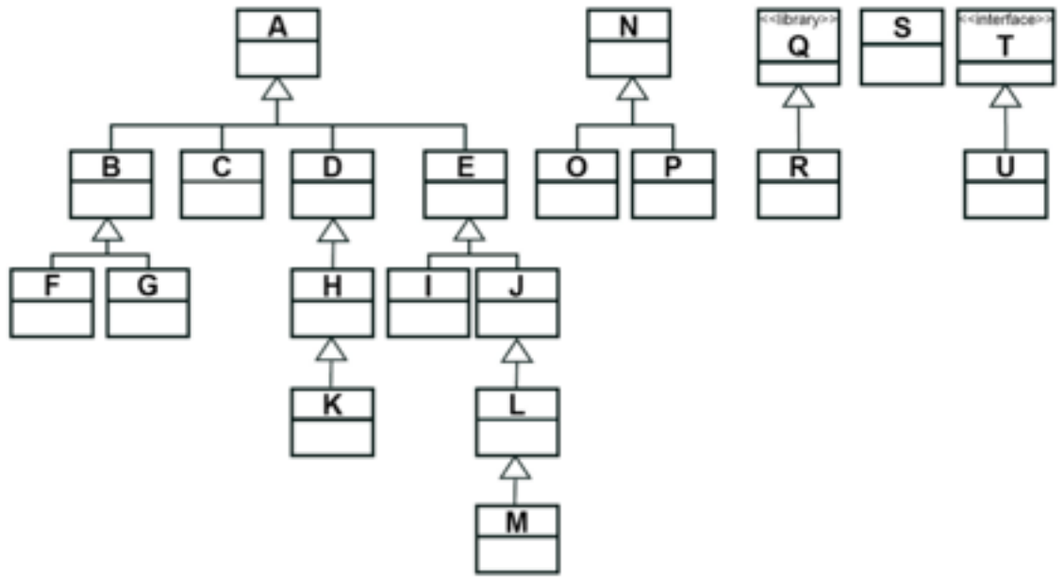
+ A proxy for the time and effort required to maintain a class

Depth of Inheritance Tree (DIT)

The maximum depth level of a class in a hierarchy.

(Chidamber & Kemerer ’94)

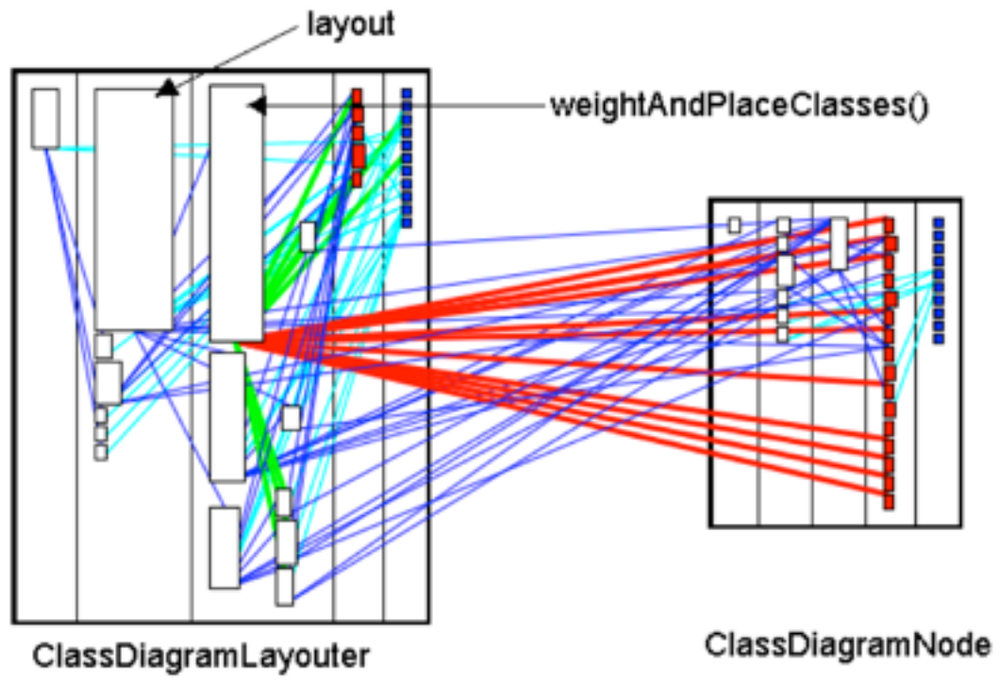
+ Inheritance depth is a good proxy for complexity



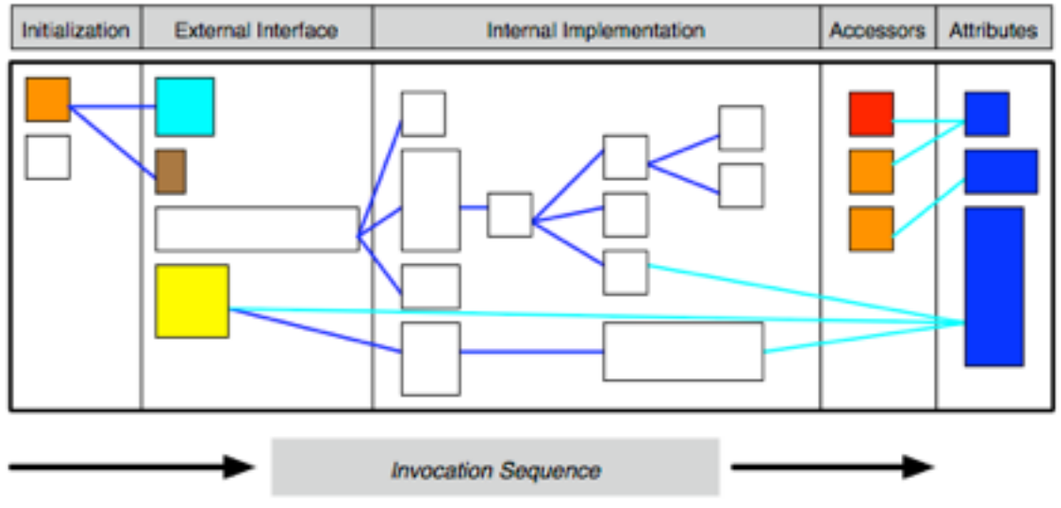
Access To Foreign Data (ATFD)

ATFD counts how many attributes from other classes are accessed directly from a given class.

(Lanza & Marinescu ’06)



+ ATFD summarizes the interaction of a class with its environment



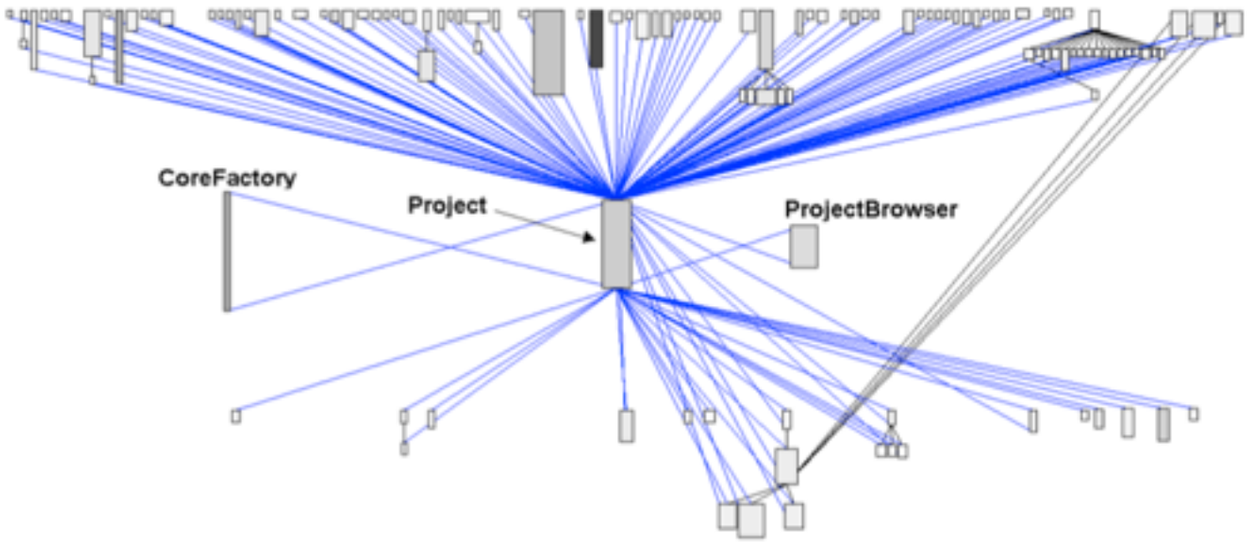
Roadmap

- Measurements
- Software Metrics
  - Size / Complexity Metrics
  - **Quality Metrics**
  - Schedule / Cost
- Metric-Based Problem Detection
  - Detecting Outliers
  - Encoding Design Problems
- Discussion

Coupling Between Object Classes (CBO)

CBO for a class is the number of other classes to which it is coupled.

(Chidamber & Kemerer ’94)

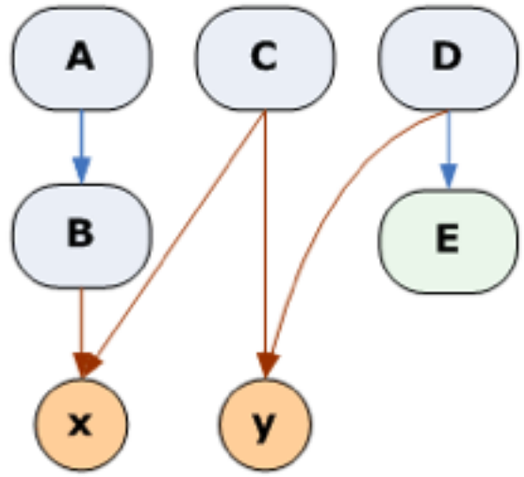


+ Meant to assess modular design and reuse

Tight Class Cohesion (TCC)

TCC counts the relative number of method-pairs that access attributes of the class in common.

(Bieman & Kang, 95)



TCC = 2 / 10 = 0.2

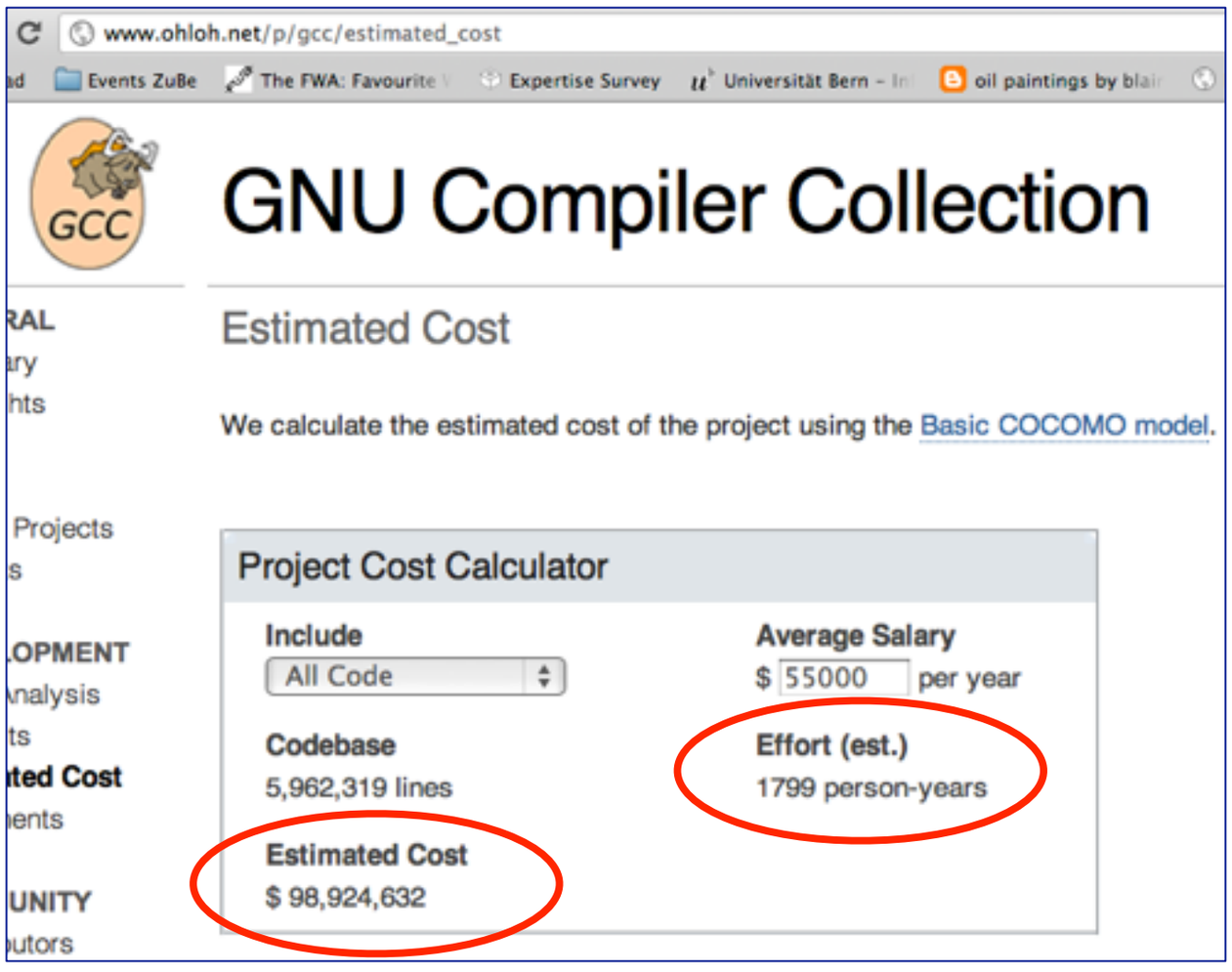
+ Can lead to improvement action

Roadmap

- Measurements
- Software Metrics
  - Size / Complexity Metrics
  - Quality Metrics
  - **Schedule / Cost**
- Metric-Based Problem Detection
  - Detecting Outliers
  - Encoding Design Problems
- Discussion

Man-Month/Year

The amount of work performed by an average developer in a month/year.



Function Point (FP)

FP is a unit of measurement to express the amount of functionality an information system provides to a user.

- Risks hiding the internal functions (algorithms)

The Measurement Process

The Goal-Question-Metric model proposes three steps to finding the correct metrics.

(Victor Basili)

1) Establish the **goals** of your maintenance or development project.

2) Derive, for each goal, **questions** that allow you to verify its accomplishment.

3) Find what should be **measured** in order to quantify the answer to the questions.

Targets without clear goals will not achieve their goals clearly.

Gilb's Principle

Roadmap

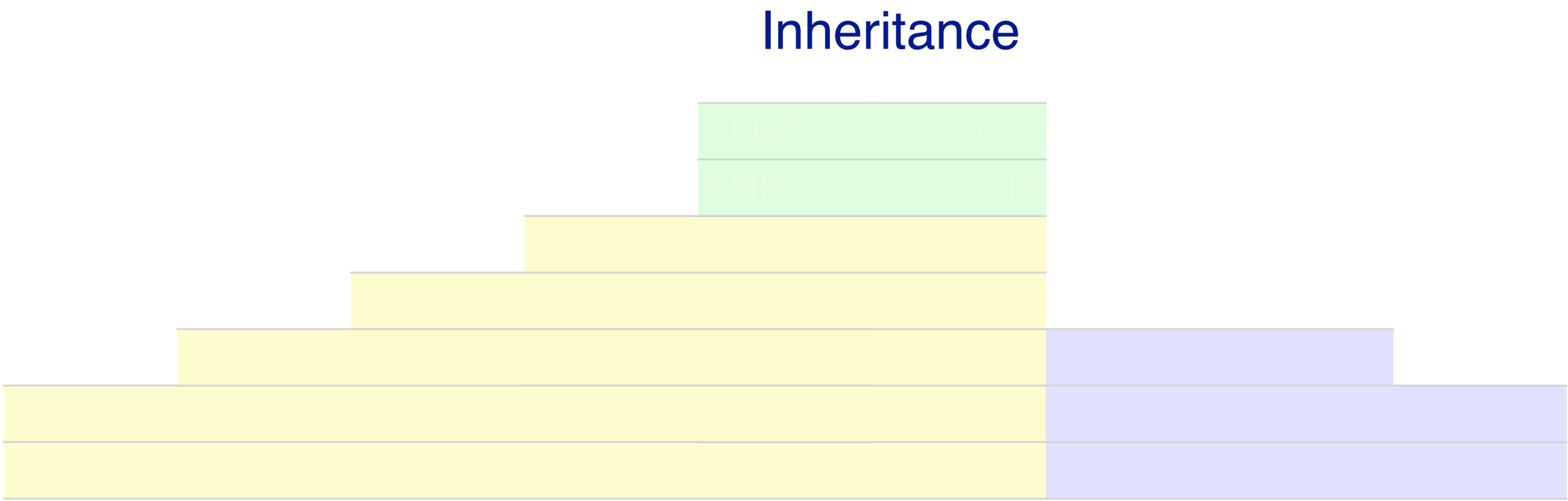
- Measurements
- Software Metrics
  - Size / Complexity Metrics
  - Quality Metrics
  - Schedule / Cost
- **Metric-based Problem Detection**
  - Detecting Outliers
  - Encoding Design Problems
- Discussion

Roadmap

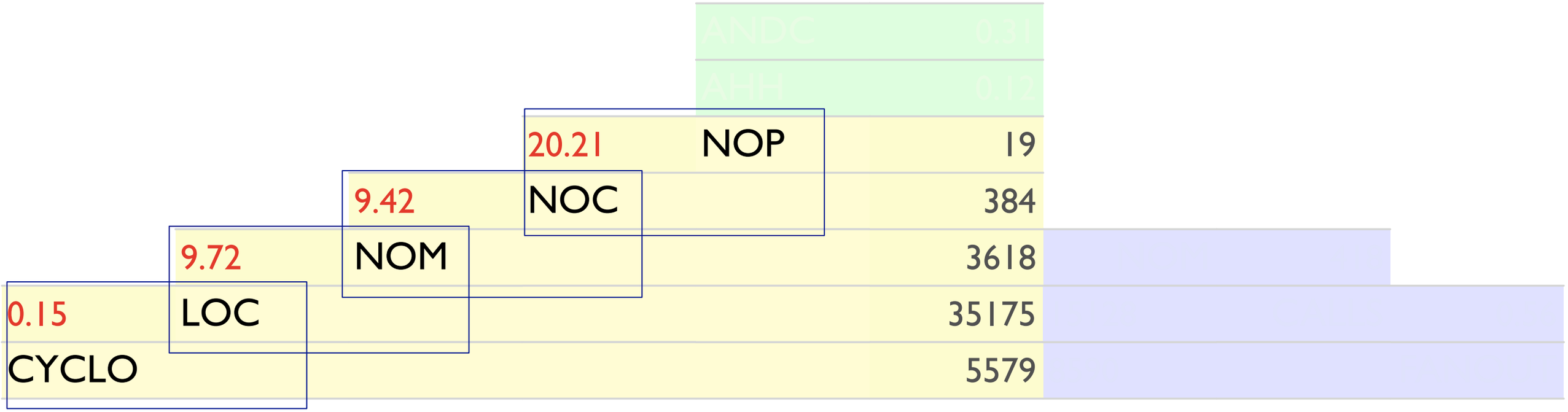
- Measurements
- Software Metrics
  - Size / Complexity Metrics
  - Quality Metrics
  - Schedule / Cost
- Metric-based Problem Detection
  - **Detecting Outliers**
  - Encoding Design Problems
- Discussion

The Overview Pyramid provides a metrics overview.

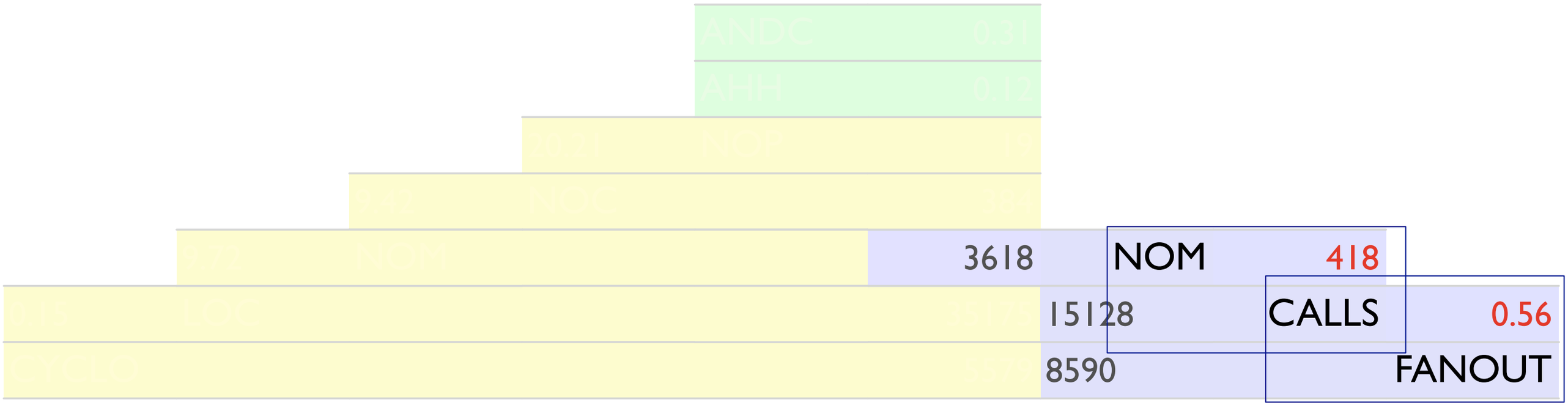
Lanza, Marinescu
2006



The Overview Pyramid provides a metrics overview.

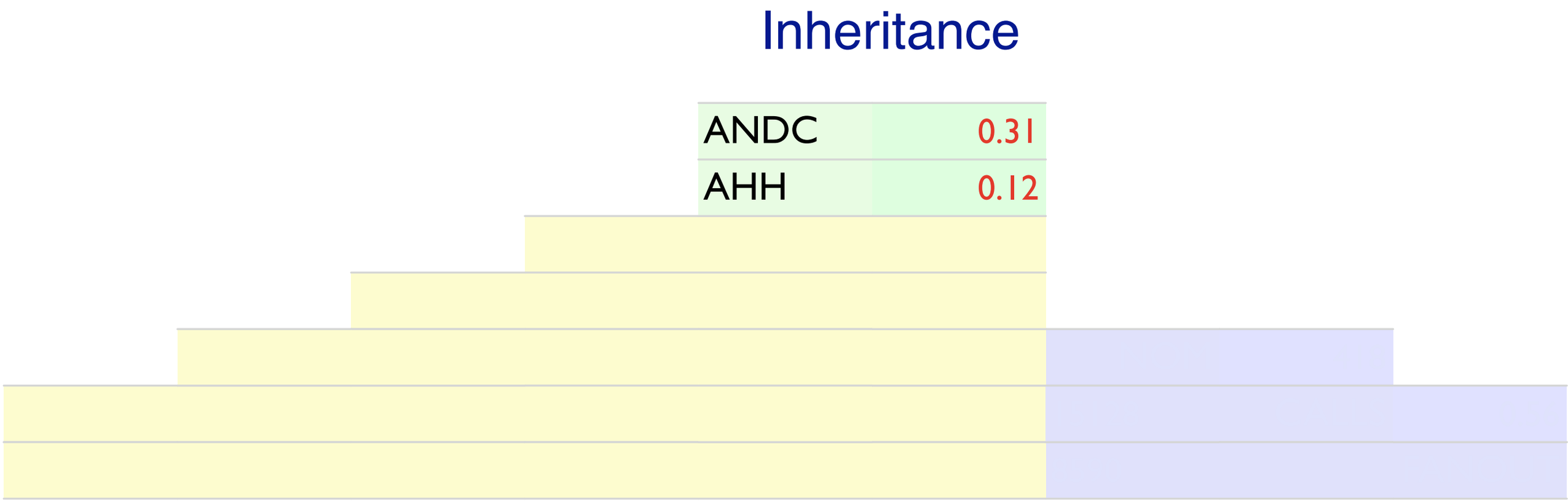


The Overview Pyramid provides a metrics overview.



Communication

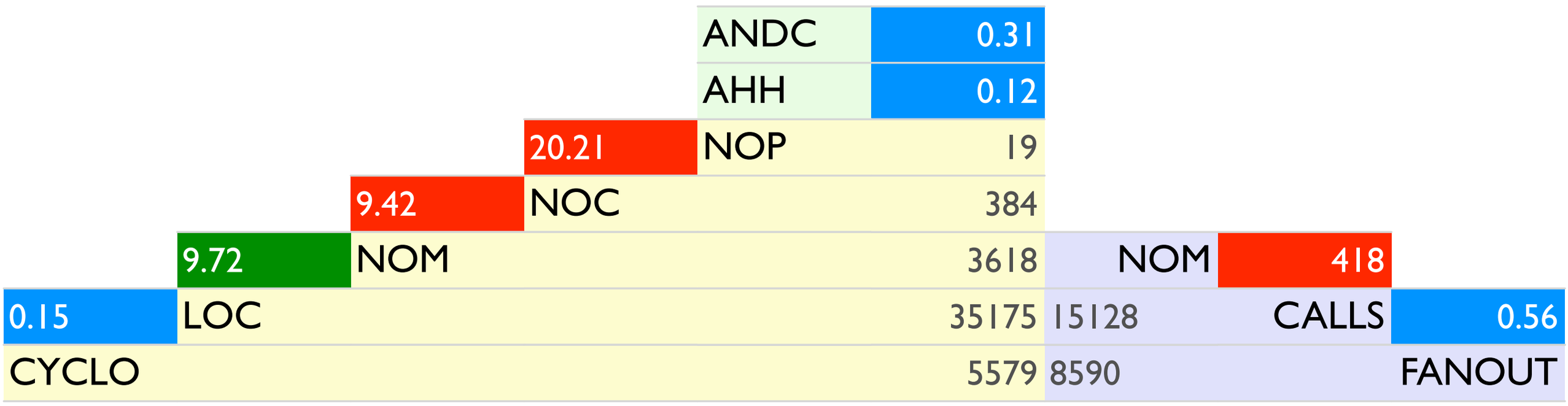
The Overview Pyramid provides a metrics overview.



The Overview Pyramid provides a metrics overview.

			ANDC	0.31		
			AHH	0.12		
		20.21	NOP	19		
	9.42	NOC		384		
	9.72	NOM	3618	NOM	418	
0.15	LOC		35175	15128	CALLS	0.56
CYCLO			5579	8590		FANOUT

The Overview Pyramid provides a metrics overview.



close to high

close to average

close to low

The Overview Pyramid provides a metrics overview.

Outlier Detection

close to high

close to average

close to low

How to obtain the thresholds?

	Java			C++		
	LOW	AVG	HIGH	LOW	AVG	HIGH
CYCLO/LOC	0.16	0.20	0.24	0.20	0.25	0.30
LOC/NOM	7	10	13	5	10	16
NOM/NOC	4	7	10	4	9	15
...						

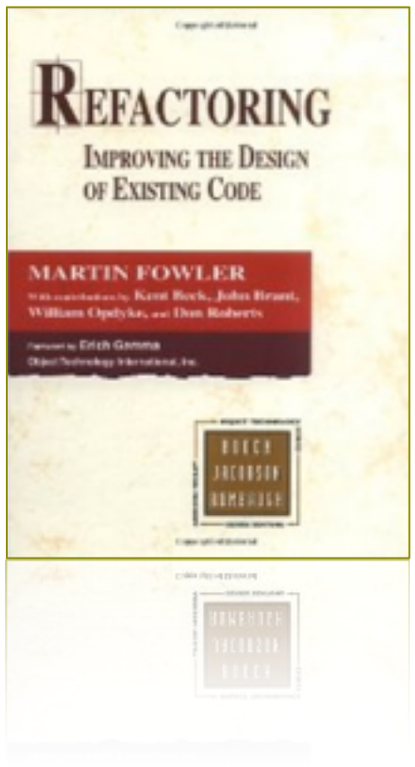
Statistical static analysis of reference systems

Context is important (e.g. programming language)

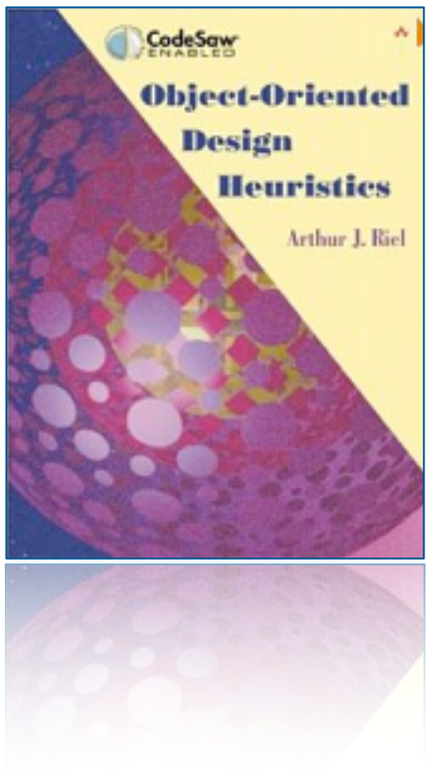
Roadmap

- Measurements
- Software Metrics
  - Size / Complexity Metrics
  - Quality Metrics
  - Schedule / Cost
- Metric-based Problem Detection
  - Detecting Outliers
  - **Encoding Design Problems**
- Discussion

Design Problems and Principles



Bad Smells
Comments
Switch Statement
Shotgun Surgery
...



Design Heuristics
Encapsulation
Minimize Coupling
Class Coherence
Inheritance Depth
...

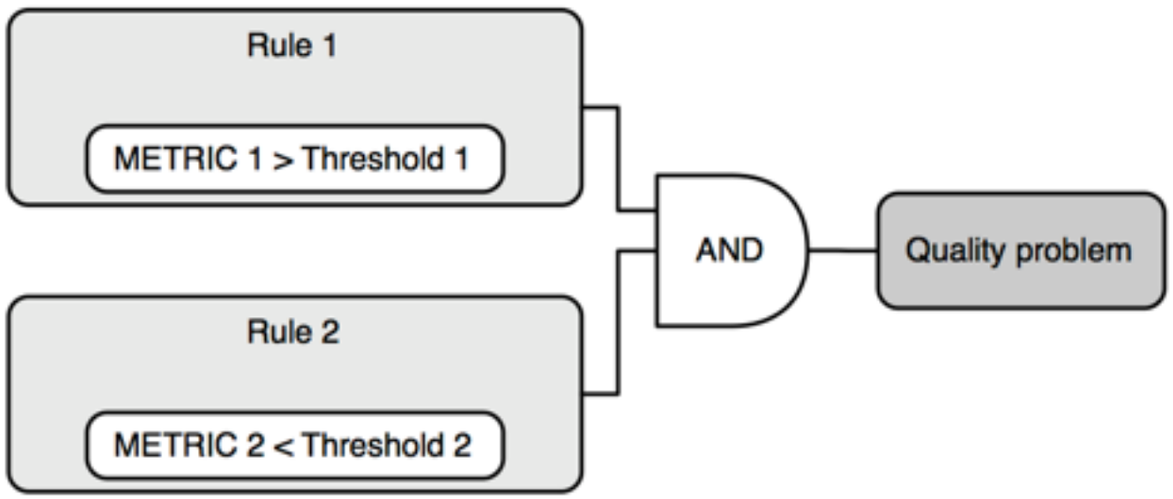
Design principles come in prose - how to measure them?

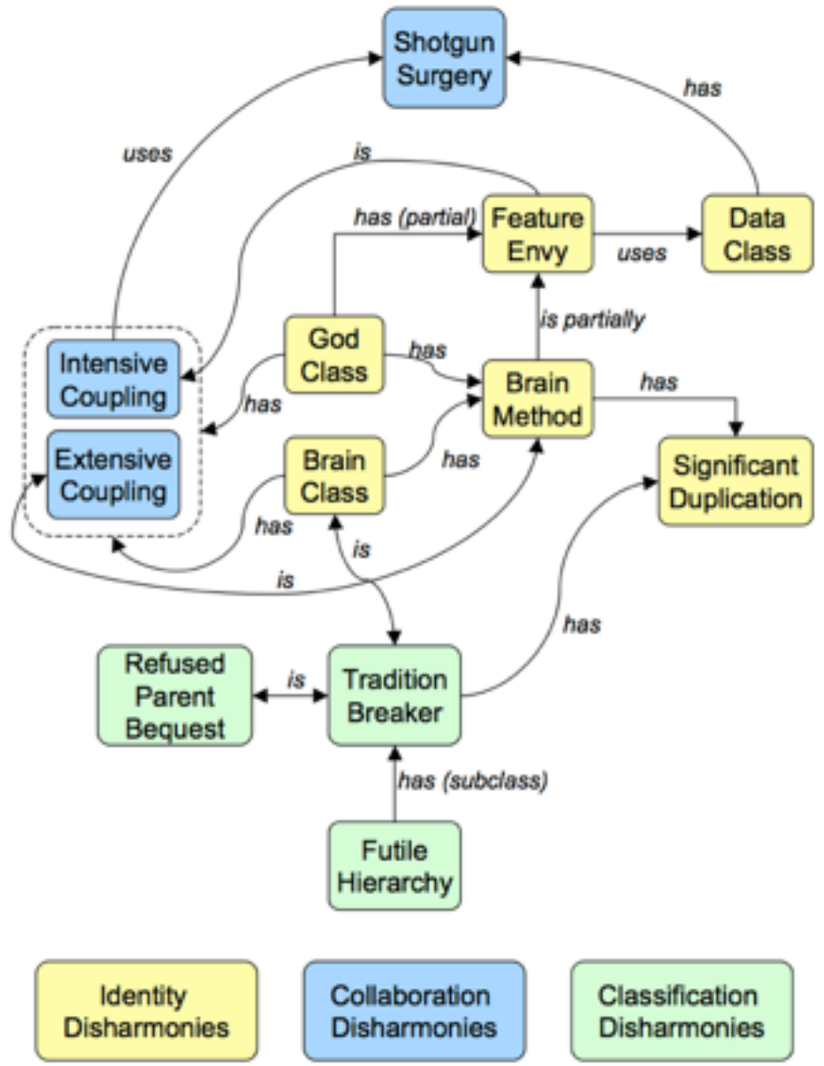
Rarely a single metric is sufficient >>> Detection Strategies

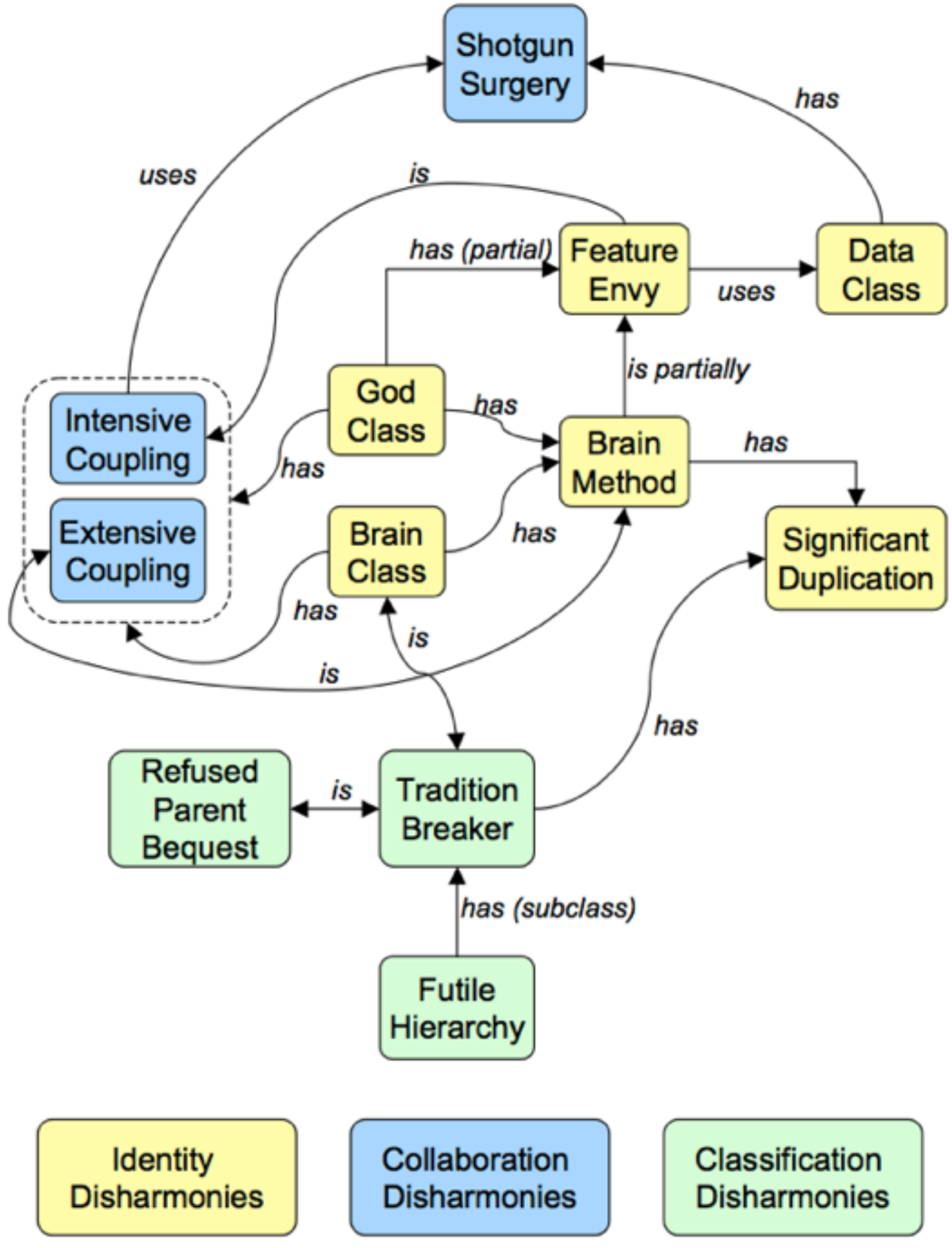
Detection Strategies...

... are metric based queries
for detecting design problems

(Lanza & Marinescu 2002)






Shotgun Surgery
has
uses
is
has (partial)
Feature Envy
uses
Data Class
is partially
God Class
has
has
Brain Method
has
Intensive Coupling
Extensive Coupling
Brain Class
has
has
Significant Duplication
is
is
has
Refused Parent Bequest
is
Tradition Breaker
has (subclass)
Futile Hierarchy
Identity Disharmonies
Collaboration Disharmonies
Classification Disharmonies

God Classes ...

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

God Classes ...

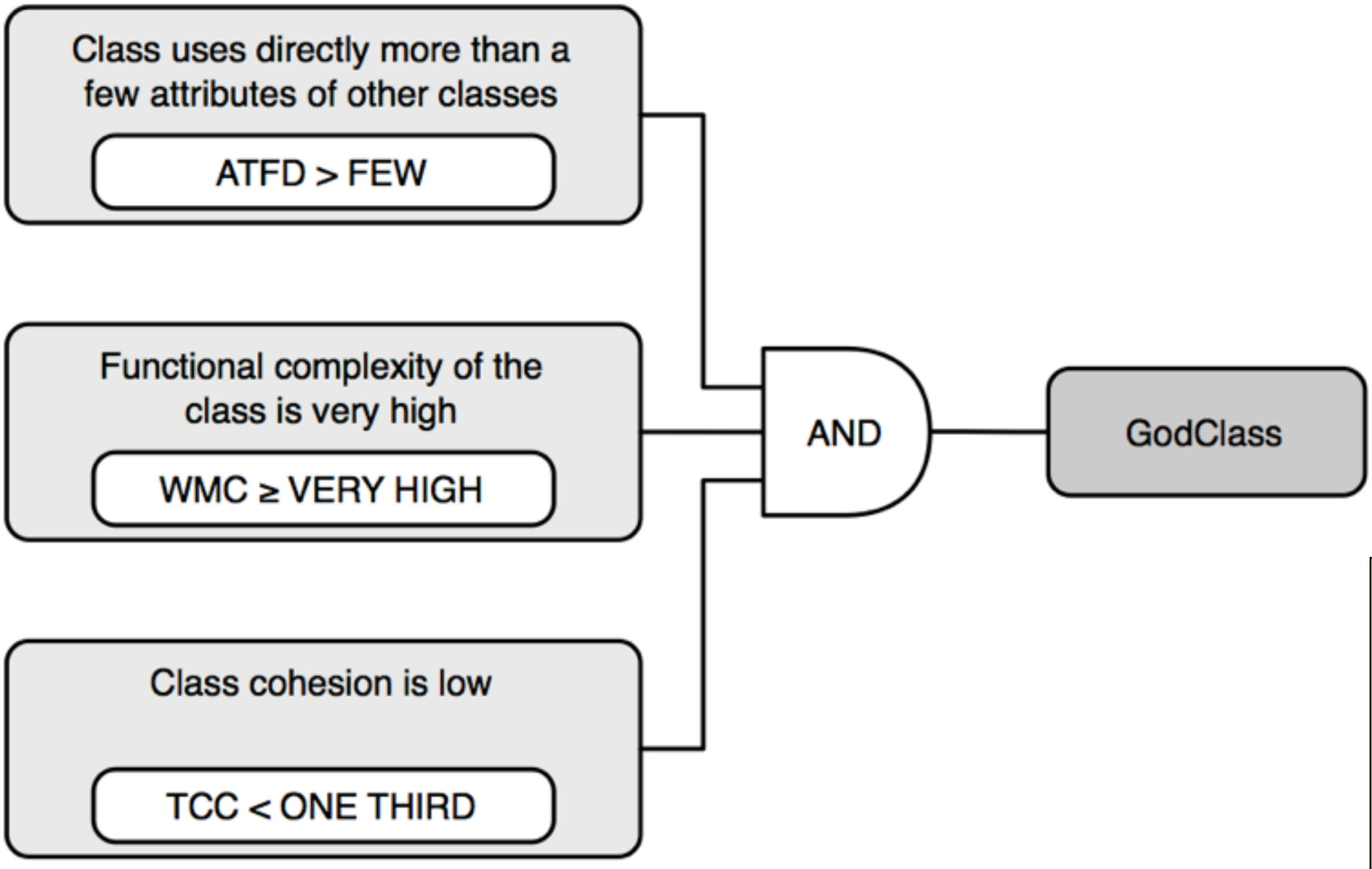
Complexity (WMC)

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

Lack of cohesion (TCC)

Foreign data usage (ATFD)

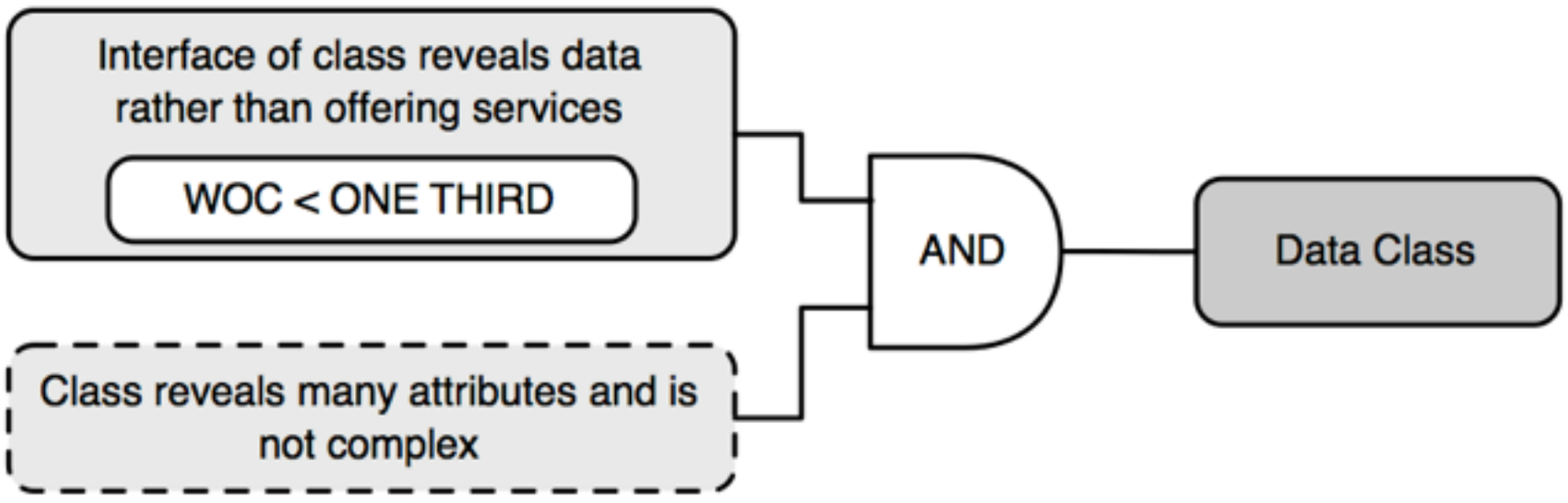
God Classes



Quantifiers

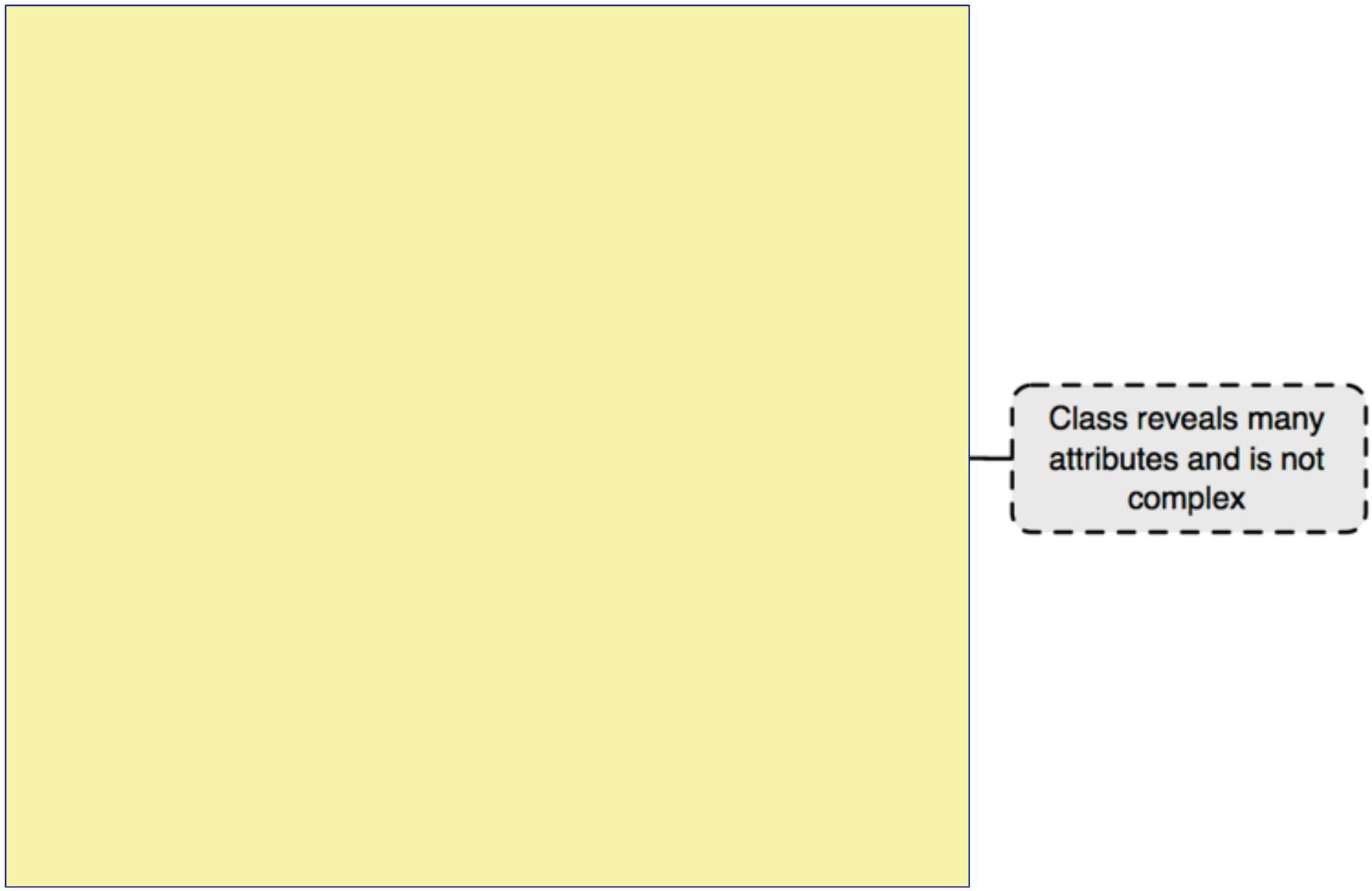
- FEW
- MANY
- TOP
- HIGH
- ONE THIRD ...

Data Classes are dumb data holders



WOC - Weight Of a Class	
Definition	The number of "functional" public methods divided by the total number of public members (Mar02a)

Data Classes are dumb data holders



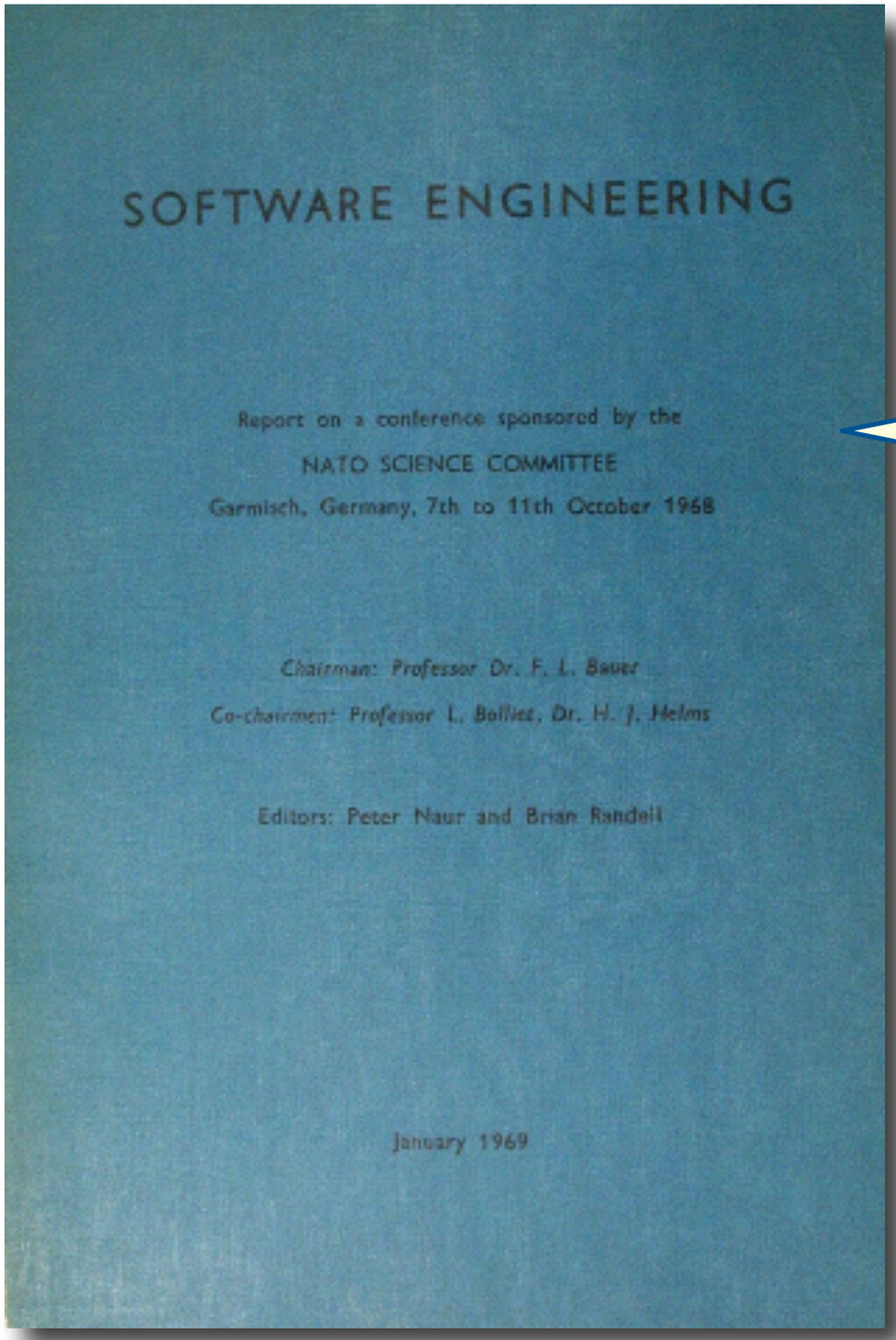
NOAP = #Public Attributes,
NOAM = #Accessor Methods

Feature Envy is ...

This one you find in the Lanza-Marinescu Book!

Roadmap

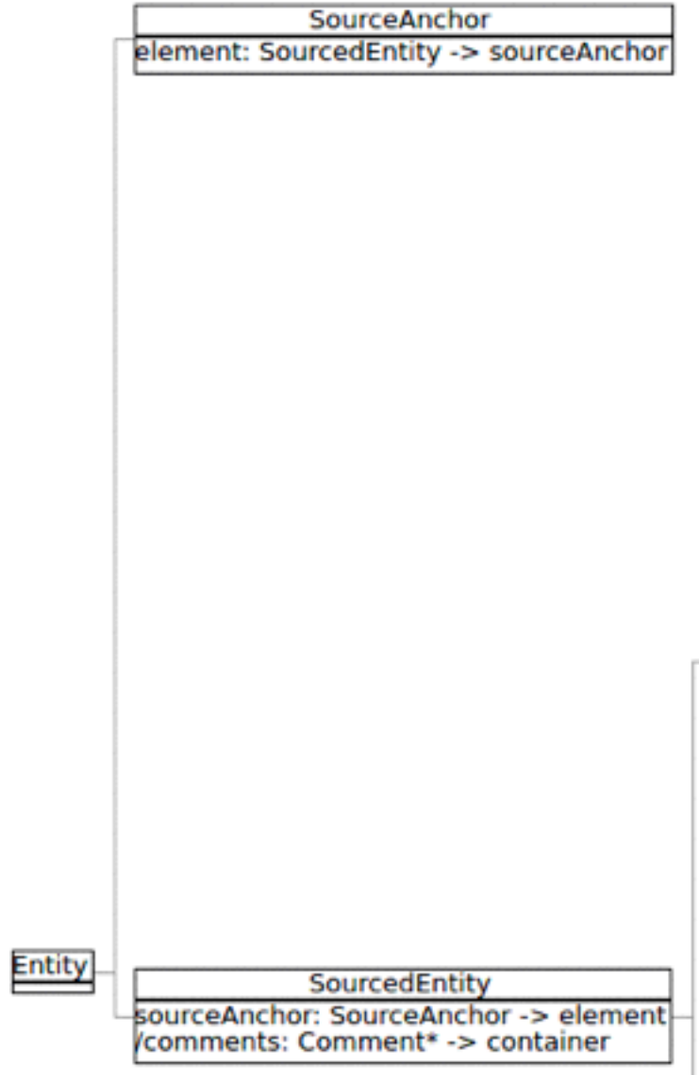
- Measurements
- Software Metrics
  - Size / Complexity Metrics
  - Quality Metrics
  - Schedule / Cost
- Metric-based Problem Detection
  - Detecting Outliers
  - Encoding Design Problems
- **Discussion**

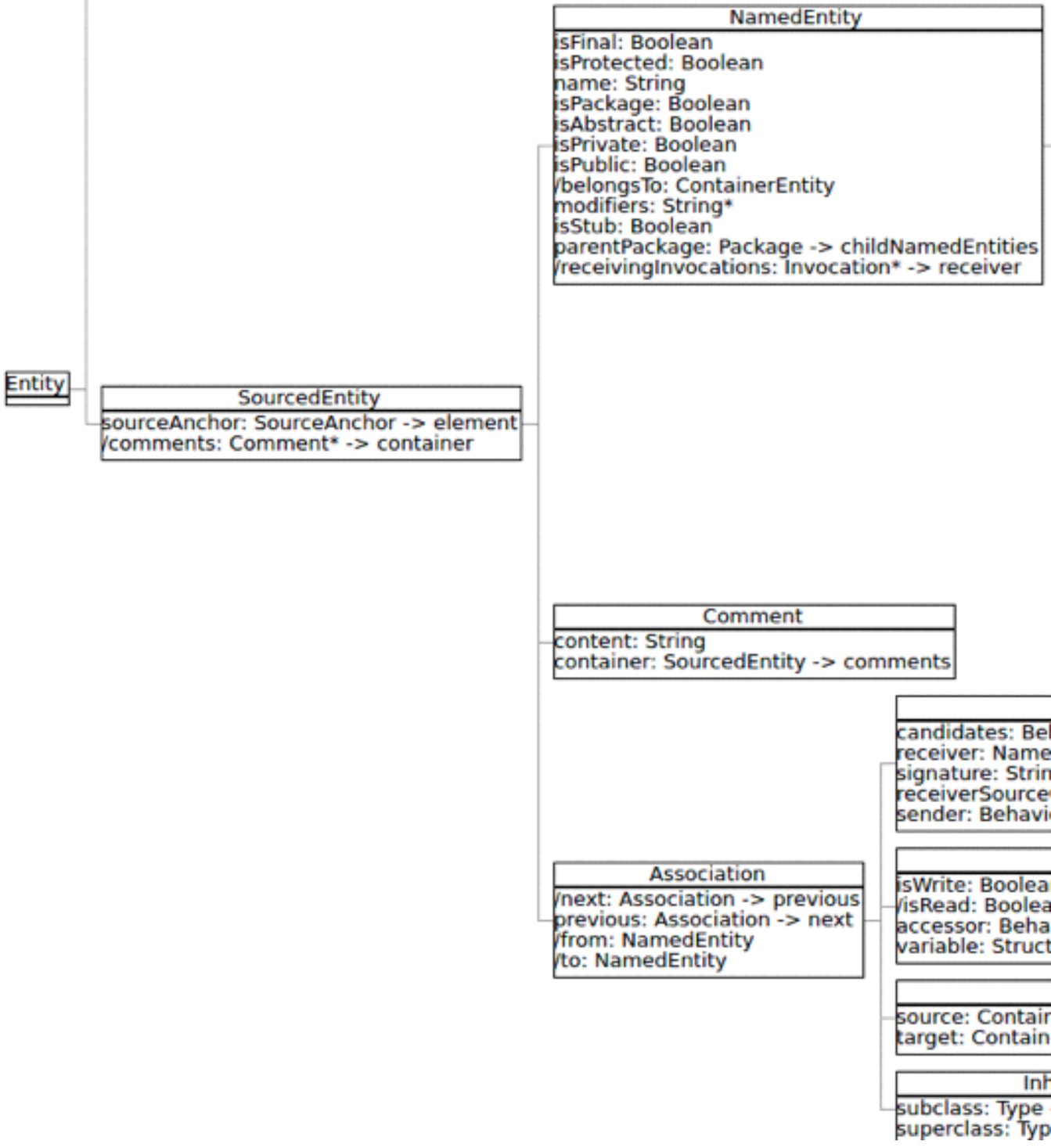


McClure:
I know of one organisation that attempts to apply time and motion standards to the output of programmers. They judge a programmer by the amount of code he produces. This is guaranteed to produce insipid code — code which does the right thing but which is twice as long as necessary.

FAMIX 3.0

- Meta-model
- Core - independent of programming language
- Implemented in Moose




NamedEntity
isFinal: Boolean
isProtected: Boolean
name: String
isPackage: Boolean
isAbstract: Boolean
isPrivate: Boolean
isPublic: Boolean
/belongsTo: ContainerEntity
modifiers: String*
isStub: Boolean
parentPackage: Package -> childNamedEntities
/receivingInvocations: Invocation* -> receiver
Entity
SourcedEntity
sourceAnchor: SourceAnchor -> element
/comments: Comment* -> container
Comment
content: String
container: SourcedEntity -> comments
Association
/next: Association -> previous
previous: Association -> next
/from: NamedEntity
/to: NamedEntity

name: String
isPackage: Boolean
isAbstract: Boolean
isPrivate: Boolean
isPublic: Boolean
/belongsTo: ContainerEntity
modifiers: String*
isStub: Boolean
parentPackage: Package -> childNamedEntities
/receivingInvocations: Invocation* -> receiver

ContainerEntity
/incomingReferences: Reference* -> tar
/types: Type* -> container
/outgoingReferences: Reference* -> sou

Entity

SourcedEntity
sourceAnchor: SourceAnchor -> element
/comments: Comment* -> container

Comment
content: String
container: SourcedEntity -> comments

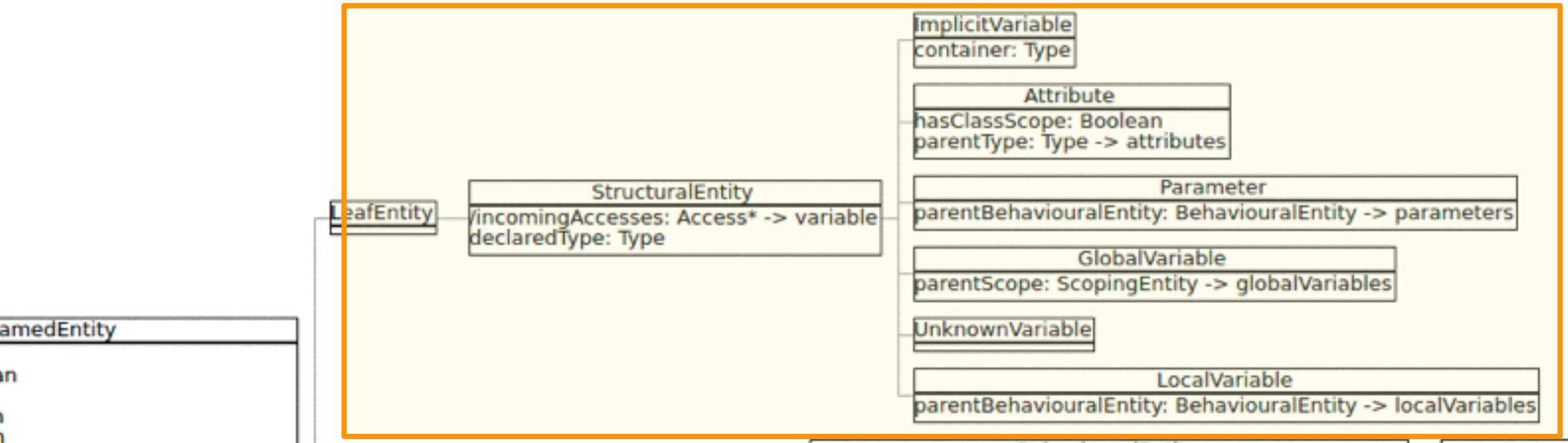
Invocation
candidates: BehaviouralEntity* -> incomingInvocations
receiver: NamedEntity -> receivingInvocations
signature: String
receiverSourceCode: String
sender: BehaviouralEntity -> outgoingInvocations

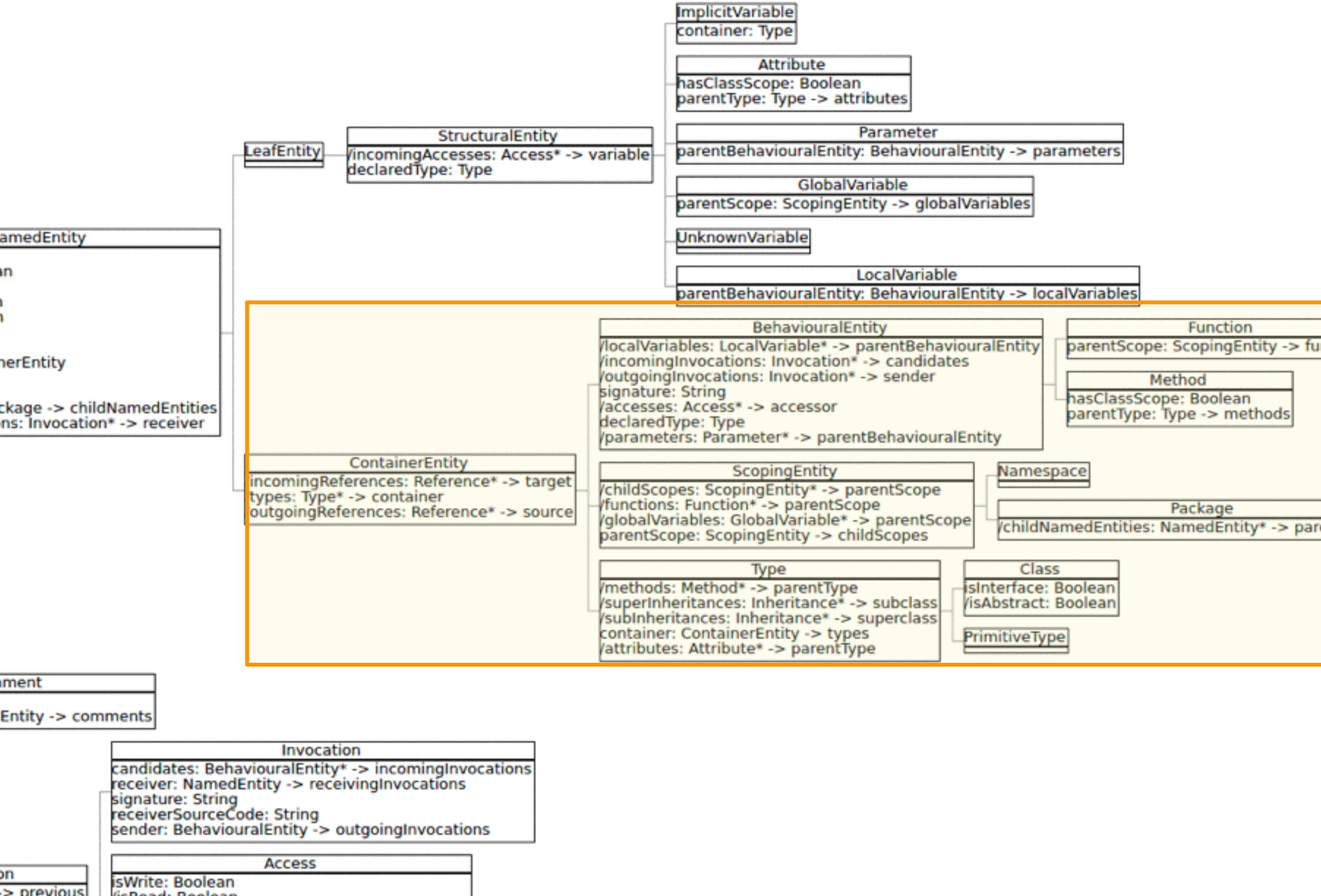
Association
/next: Association -> previous
previous: Association -> next
/from: NamedEntity
/to: NamedEntity

Access
isWrite: Boolean
/isRead: Boolean
accessor: BehaviouralEntity -> accesses
variable: StructuralEntity -> incomingAccesses

Reference
source: ContainerEntity -> outgoingReferences
target: ContainerEntity -> incomingReferences

Inheritance
subclass: Type -> superInheritances
superclass: Type -> subInheritances


ImplicitVariable
container: Type
Attribute
hasClassScope: Boolean
parentType: Type -> attributes
Parameter
parentBehaviouralEntity: BehaviouralEntity -> parameters
GlobalVariable
parentScope: ScopingEntity -> globalVariables
UnknownVariable
LocalVariable
parentBehaviouralEntity: BehaviouralEntity -> localVariables
LeafEntity
StructuralEntity
/incomingAccesses: Access* -> variable
declaredType: Type
amedEntity
an
nerEntity
ckage -> childNamedEntities
ns: Invocation* -> receiver
BehaviouralEntity
/localVariables: LocalVariable* -> parentBehaviouralEntity
/incomingInvocations: Invocation* -> candidates
/outgoingInvocations: Invocation* -> sender
signature: String
/accesses: Access* -> accessor
declaredType: Type
/parameters: Parameter* -> parentBehaviouralEntity
Function
parentScope: ScopingEntity -> fu
Method
hasClassScope: Boolean
parentType: Type -> methods
ContainerEntity
/incomingReferences: Reference* -> target
/types: Type* -> container
/outgoingReferences: Reference* -> source
ScopingEntity
/childScopes: ScopingEntity* -> parentScope
/functions: Function* -> parentScope
/globalVariables: GlobalVariable* -> parentScope
parentScope: ScopingEntity -> childScopes
Namespace
Package
/childNamedEntities: NamedEntity* -> par
Type
/methods: Method* -> parentType
/superInheritances: Inheritance* -> subclass
/subInheritances: Inheritance* -> superclass
container: ContainerEntity -> types
/attributes: Attribute* -> parentType
Class
isInterface: Boolean
/isAbstract: Boolean
PrimitiveType
ment
Entity -> comments
Invocation
candidates: BehaviouralEntity* -> incomingInvocations
receiver: NamedEntity -> receivingInvocations
signature: String
receiverSourceCode: String
sender: BehaviouralEntity -> outgoingInvocations
Access
isWrite: Boolean
/isRead: Boolean
on
-> previous
on -> next


ImplicitVariable
container: Type
Attribute
hasClassScope: Boolean
parentType: Type -> attributes
Parameter
parentBehaviouralEntity: BehaviouralEntity -> parameters
GlobalVariable
parentScope: ScopingEntity -> globalVariables
UnknownVariable
LocalVariable
parentBehaviouralEntity: BehaviouralEntity -> localVariables
LeafEntity
StructuralEntity
/incomingAccesses: Access* -> variable
declaredType: Type
amedEntity
an
nerEntity
ckage -> childNamedEntities
ns: Invocation* -> receiver
BehaviouralEntity
/localVariables: LocalVariable* -> parentBehaviouralEntity
/incomingInvocations: Invocation* -> candidates
/outgoingInvocations: Invocation* -> sender
signature: String
/accesses: Access* -> accessor
declaredType: Type
/parameters: Parameter* -> parentBehaviouralEntity
Function
parentScope: ScopingEntity -> fu
Method
hasClassScope: Boolean
parentType: Type -> methods
ContainerEntity
incomingReferences: Reference* -> target
types: Type* -> container
outgoingReferences: Reference* -> source
ScopingEntity
/childScopes: ScopingEntity* -> parentScope
/functions: Function* -> parentScope
/globalVariables: GlobalVariable* -> parentScope
parentScope: ScopingEntity -> childScopes
Namespace
Package
/childNamedEntities: NamedEntity* -> par
Type
/methods: Method* -> parentType
/superInheritances: Inheritance* -> subclass
/subInheritances: Inheritance* -> superclass
container: ContainerEntity -> types
/attributes: Attribute* -> parentType
Class
isInterface: Boolean
/isAbstract: Boolean
PrimitiveType
ment
Entity -> comments
Invocation
candidates: BehaviouralEntity* -> incomingInvocations
receiver: NamedEntity -> receivingInvocations
signature: String
receiverSourceCode: String
sender: BehaviouralEntity -> outgoingInvocations
on
-> previous
on -> next
Access
isWrite: Boolean
/isRead: Boolean

What you should know!

- Software metrics are measurements
- Every scale allows certain operations and analyses
- Detection strategies are queries for design problem detection
- The Goal Question Metric model has three phases
- Bad smells encode bad OO practices
- Design heuristics encode good OO practices

Can you answer these questions?

- How do you compute TCC for a given class?
- Can you explain how the God Class detection strategy works?
- Can you list several of the elements of the FAMIX meta-model?
- What are three metrics appropriate for OO systems but not be appropriate for procedural systems?
- Can you give examples of three bad smells?
- Why are comments a bad smell? But switch clauses?
- Can you give examples of three design heuristics?

Further Reading

- *Cohesion and Reuse in Object Oriented Systems*, by Bieman & Kang
- *OOMIP* by Lanza and Marinescu (Sections 5.3 - 5.5)
- http://sourcemaking.com/refactoring/bad-smells-in-code
- http://scg.unibe.ch/staff/mircea/sde/60-design-heuristics