



# A Validation of Martin's Metric

26.09.2009 | Sami Hyrynsalmi (sthyry@utu.fi)



# Table of contents

1. Software package metrics
2. Martin's metric
3. Theoretical validation
4. Experimental evaluation
5. Conclusion and future work



# Software package metrics

- The most common objective of (package) metrics is to identify weak parts of software
- Package is a set of classes which have a **reason** to be in the same set
  - A package can be an architectural component or a part of one component
  - In Java, packages are explicitly named so the automatization of package measures is easy
  - In C++, packages are implicit and the metrics software might need external help to recognize the packages (e.g. folder structure)



# Software package metrics

- The most widely known package metric is presented by R.C. Martin (1995; 2002)
- Also Ducasse *et al.* (2005), Ponisio (2006), Melton and Tempero (2007), and Zhou *et al.* (2008) have presented metrics for package level
- Martin's metric is widely used, but there is a lack of theoretical and empirical evaluation



## Martin's basic package measurements

$C_a$  = number of classes outside of the package which depend on the package

$C_e$  = number of classes outside of the package on which the package depends upon

$$I \text{ (instability)} = \frac{C_e}{C_a + C_e}$$

$N_a$  = number of abstract classes, interfaces etc. in the package

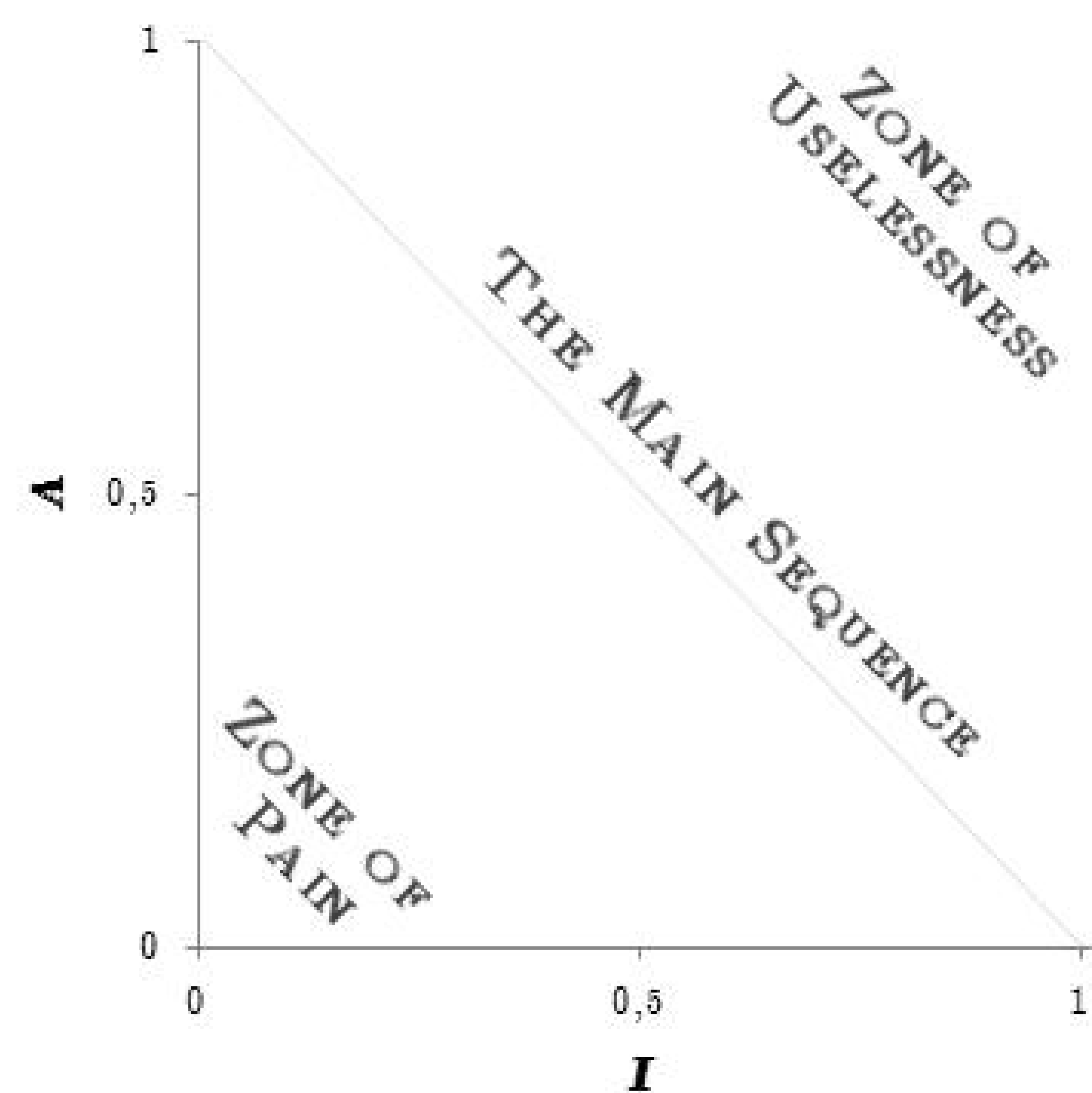
$N_c$  = number of all classes in the package

$$A \text{ (abstractness)} = \frac{N_a}{N_c}$$



# Martin's metric

- Martin defined IA-graph where the line from (1,0) to (0,1) is called the main sequence
  - Packages in the main sequence are in the right proportions of stability and abstractness
- Martin's metric is the package's distance from the main sequence
  - $D$  is distance and  $D'$  is normalized distance
- The objective of the metric is to find packages which are hard to maintain and reuse



$$D = \frac{|A + I - 1|}{\sqrt{2}}$$

$$D' = |A + I - 1|$$



# Martin's cohesion metric

- R.C. Martin (2002) also presented the cohesion metric  $H$  which does not belong in the original metric suite
- $H = \frac{\rho + 1}{N_c}$ 
  - Where  $\rho$  is the number of internal class-to-class relationships in the package and  $N_c$  is the number of classes
- Martin didn't define what are the criteria of coupling or what are the internal relationships



# Theoretical validation

- The objective is to test if the metric supports (some) theory of the attribute which it tries to measure
- Theoretical validation can reveal flaws from the definition of a metric, but it can't show if the metric is reasonable or practical
- Briand *et al.* (1996) presented 5 requirements for coupling, 4 for cohesion, and 3 for size metrics



# Properties of size

- $Nc$  and  $Na$  are size metrics,  $A$  is a kind of relative size metric
- 1. Non-negativity
  - The value of a size metric is non-negative
  - $Nc$ ,  $Na$ , and  $A$  fulfil the property
- 2. Null value
  - The value of a size metric is null for empty packages
  - $Nc$ ,  $Na$ , and  $A$  fulfil the property
- 3. Module additivity
  - $SizeMetric(P1)+SizeMetric(P2)=SizeMetric(P1+P2)$
  - $Nc$  and  $Na$  fulfil the property, for  $A$  the property does not hold
  - This is not failure of  $A$ , because framework is not intended for *relative sizes*



# Properties of coupling

- $C_e$  and  $C_a$  are coupling metrics
- 1. Non-negativity
  - The value of metric is non-negative
- 2. Null value
  - The value of metric is null when there is no coupling
- 3. Monotonicity
  - $CouplingMetric(P) \geq CouplingMetric(P')$
- 4. Merging of connected systems
  - $CouplingMetric(P1) + CouplingMetric(P2) \geq CouplingMetric(P1+P2)$
- 5. Merging of unconnected systems
  - $CouplingMetric(P1) + CouplingMetric(P2) = CouplingMetric(P1+P2)$
- $C_e$  and  $C_a$  fulfil the properties



# Properties of cohesion

- $H$  is a cohesion metric

## 1. Non-negativity and normalization

- The metric's range should be  $[0, max]$
- $H$  does not fulfil the property

$$H = \frac{\rho + 1}{Nc}$$

## 2. Null value and maximum value

- When the set of internal relationship is empty, the value of metric is null. When the set is maximal, the value of metric is  $max$ .
- $H$  does not fulfil the property

## 3. Monotonicity

- $CohesionMetric(P) \leq CohesionMetric(P')$
- The property holds for  $H$

## 4. Cohesive modules

- $max\{CohesionMetric(P1), CohesionMetric(P2)\} \geq CohesionMetric(P1+P2)$
- The property holds



# Proposed variation

- $H$  does not fulfil two properties
- One of the problem is that there is no fixed upper or lower bound
  - Additional 1 in numerator prevents reaching the value 0.
  - $\rho$  depends polynomially on  $N_c$ , and thus there is no fixed upper bound.
- Proposed variation  $H'$  is valid and quite naturally resembles the cohesion metric  $C_o'$  defined by Briand *et al.* (1998)

$$H = \frac{\rho + 1}{N_c}$$

$$H' = \begin{cases} 1, & N_c = 1 \\ \frac{2\rho}{N_c(N_c - 1)}, & N_c \neq 1 \end{cases}$$

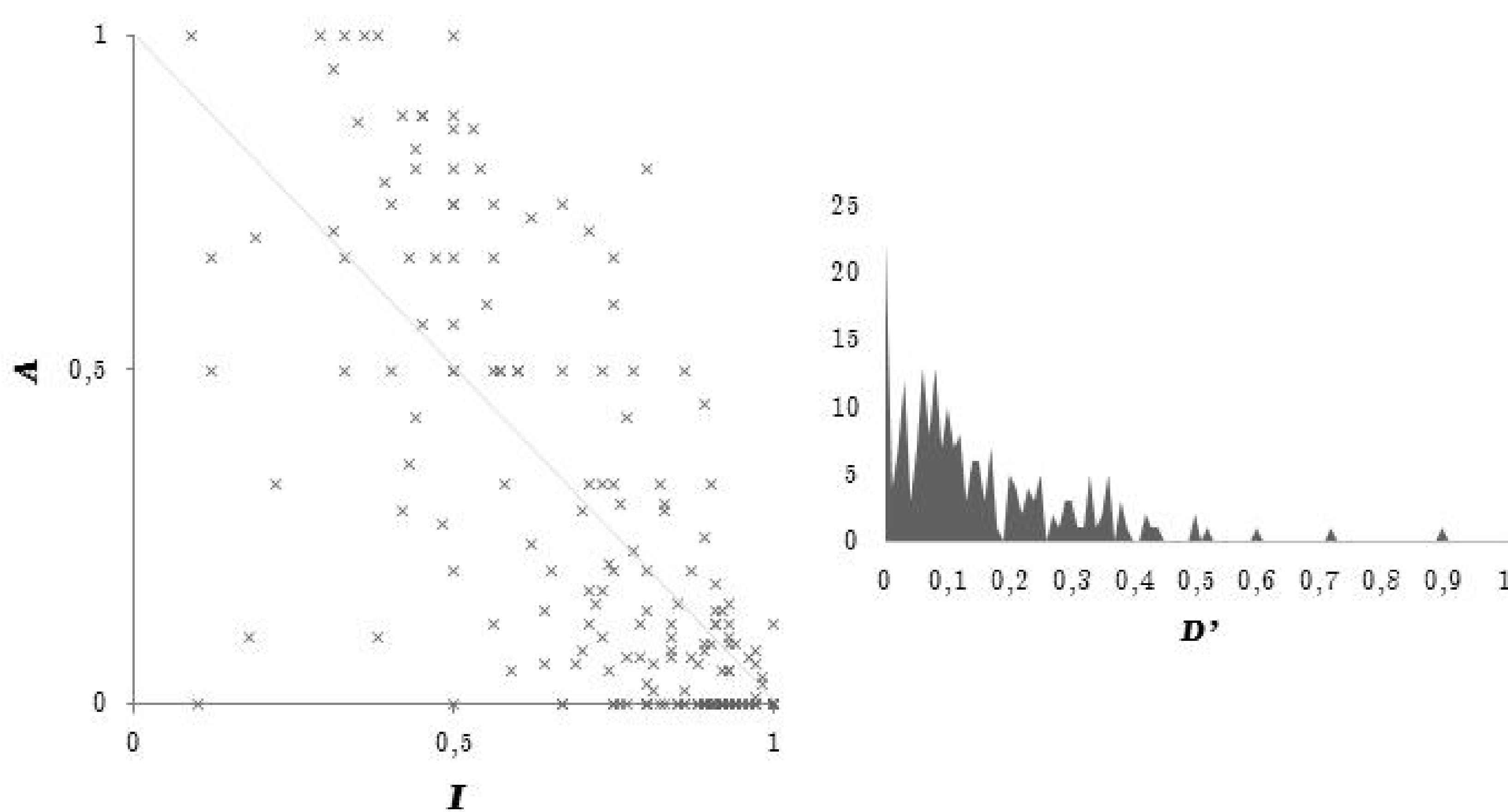


# Experimental evaluation

- If applications believed to be well-made perform well according to Martin's metric, the metric might capture some principles of good design
- Five open source software were measured with JDepend metric software
- Only packages which have been produced by the project itself are measured

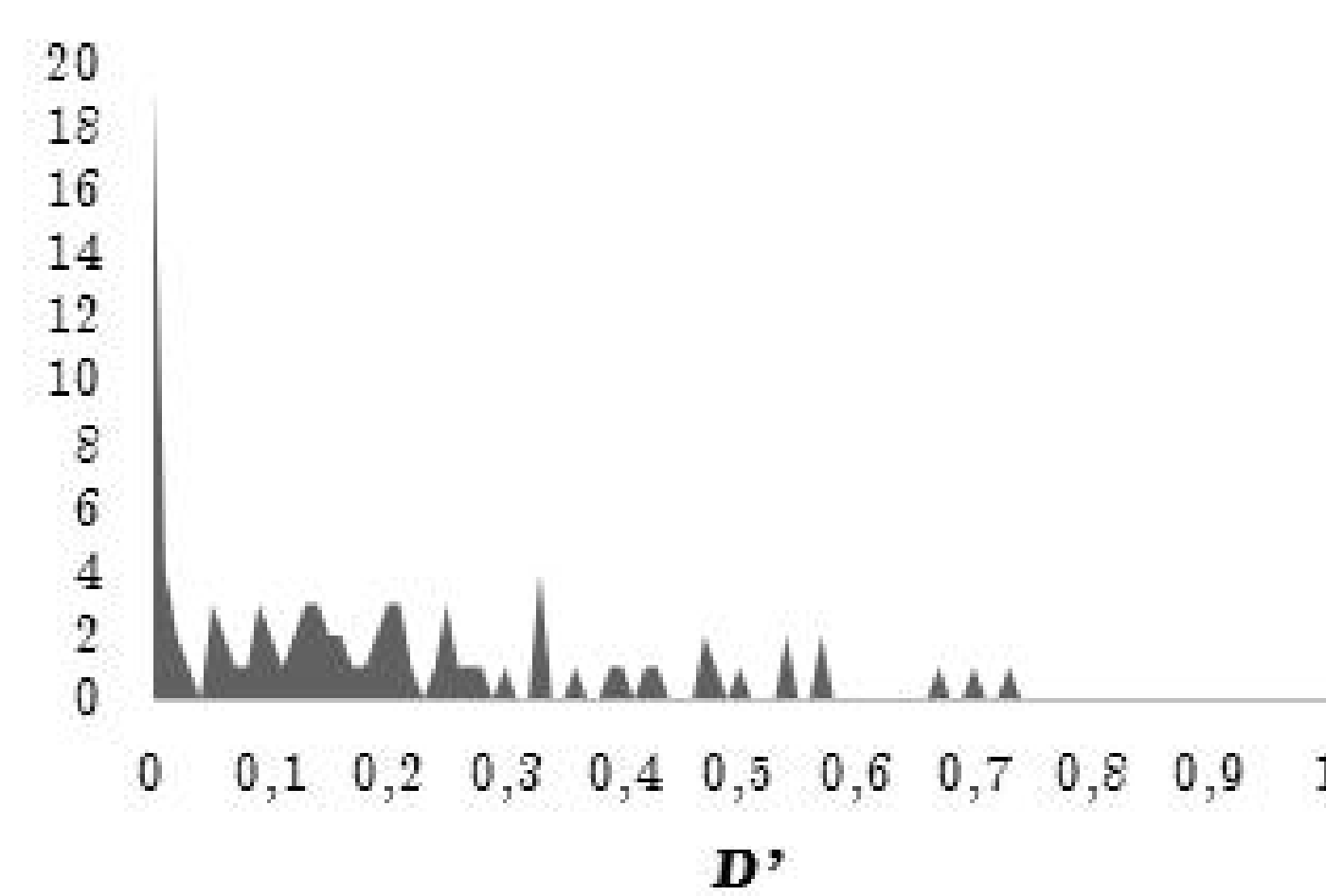
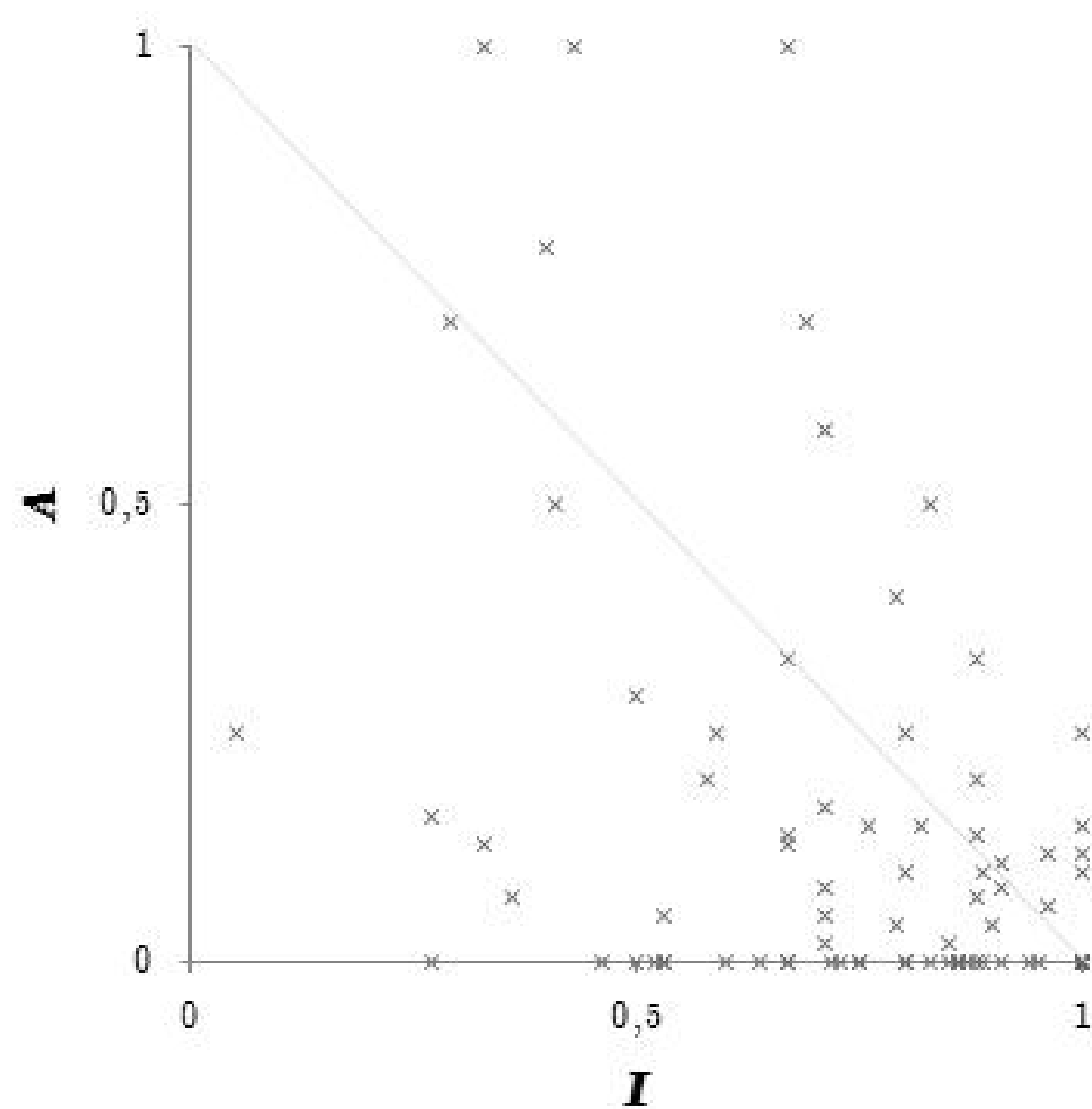


# Vuze



- $N = 208$
- 22 packages (10%) are in the main sequence
- 156 (75%) are no further than 0.21 from the line
- Only 6 packages'  $D' \geq 0.5$

# Apache Tomcat & ArgoUML

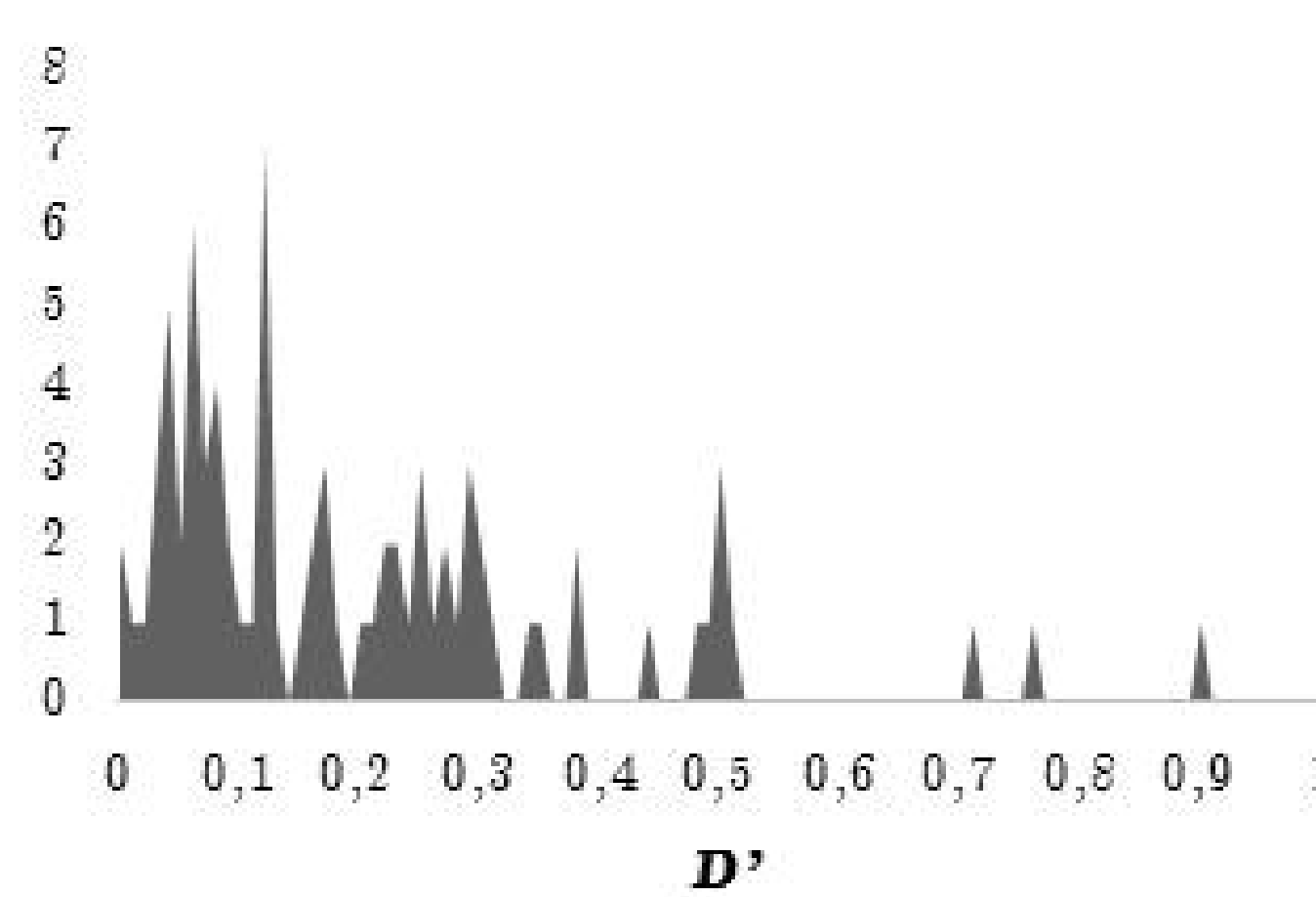
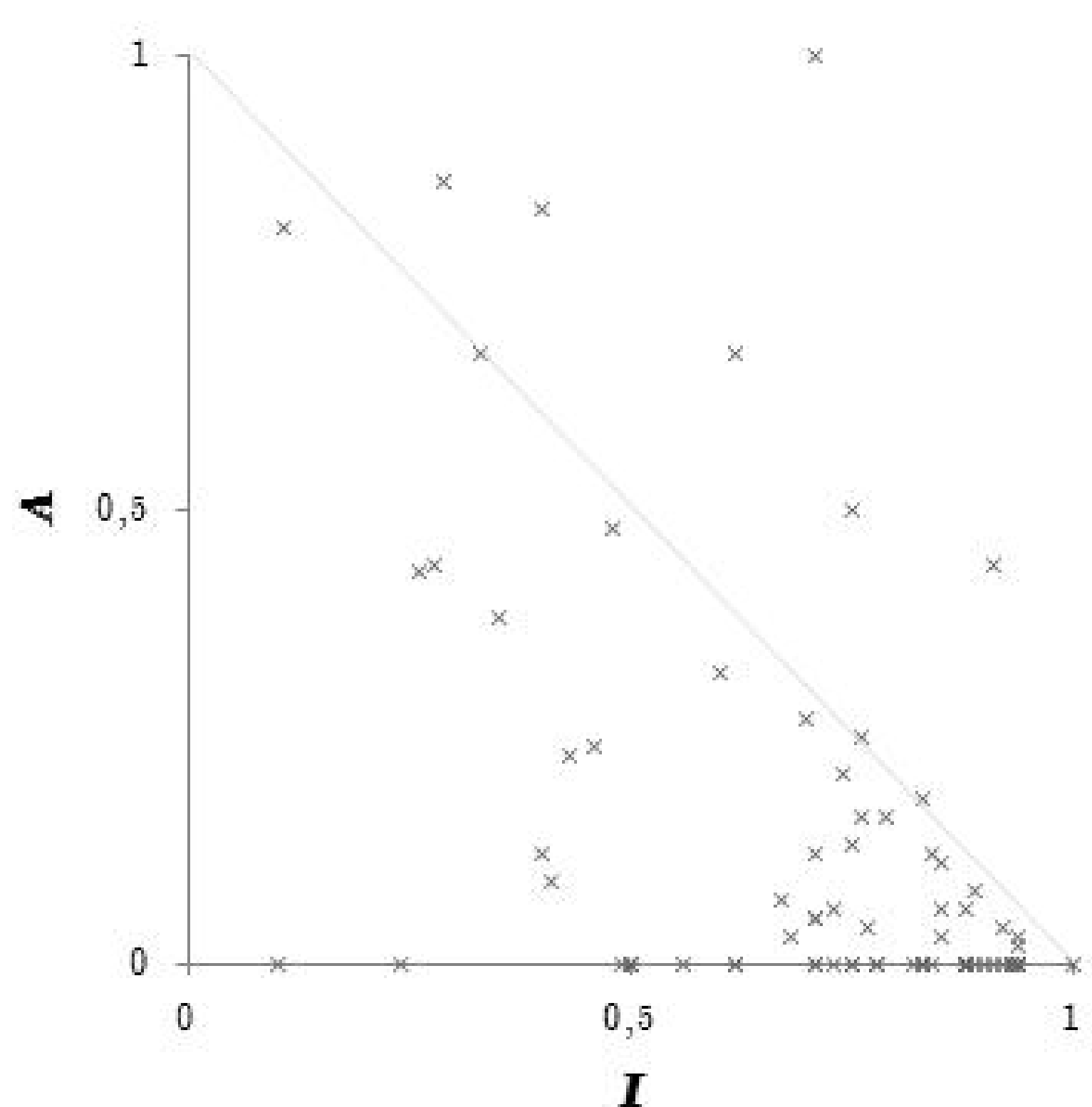


Apache Tomcat

N = 90

20% in the main sequence

Q3 = 0.27



ArgoUML

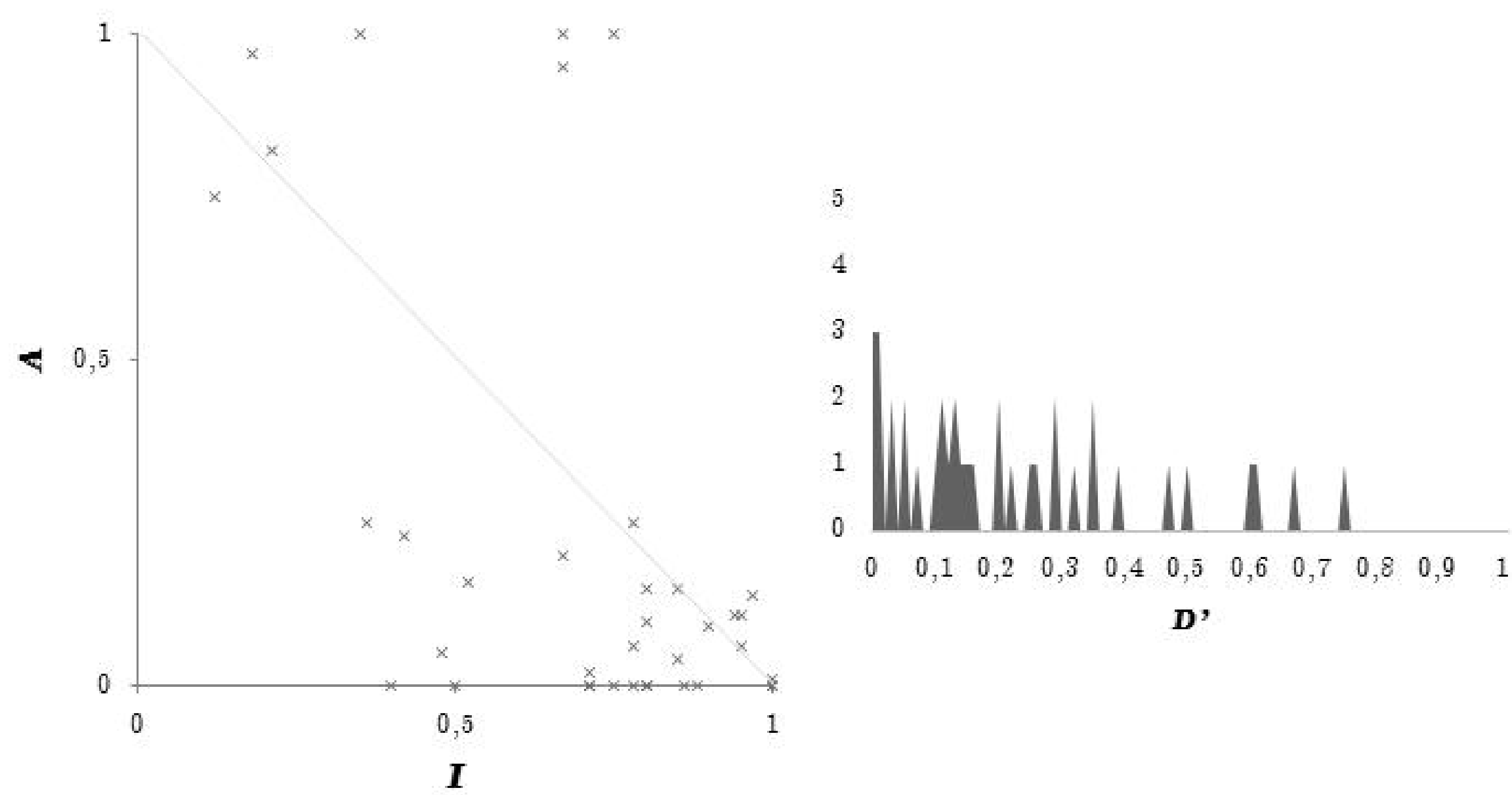
N = 80

The histogram differs notably when compared with the others.

Q3 = 0.28



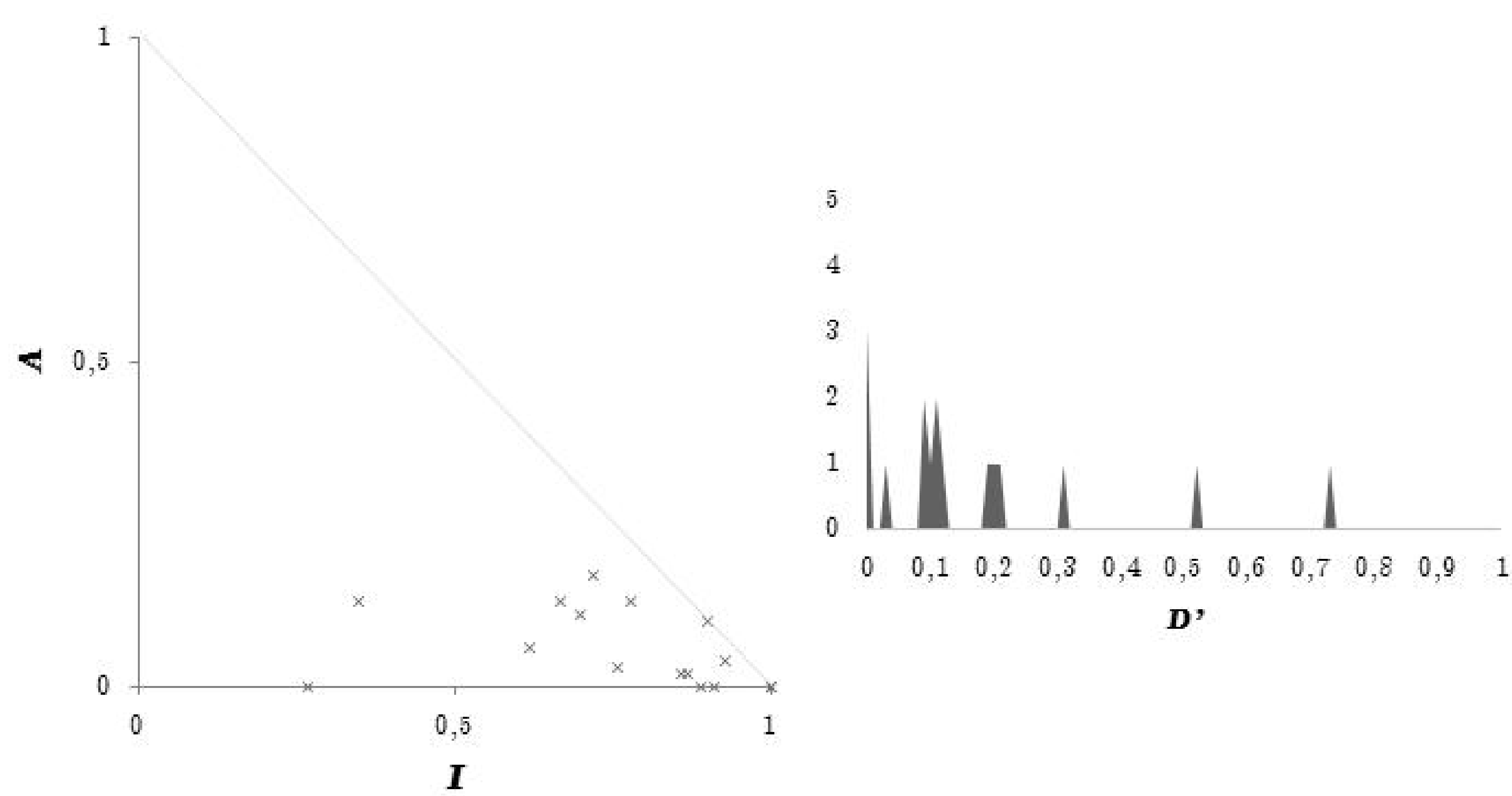
# Apache Xerces2 J & jEdit



Apache Xerces2 J

N = 36

Q3 = 0.32



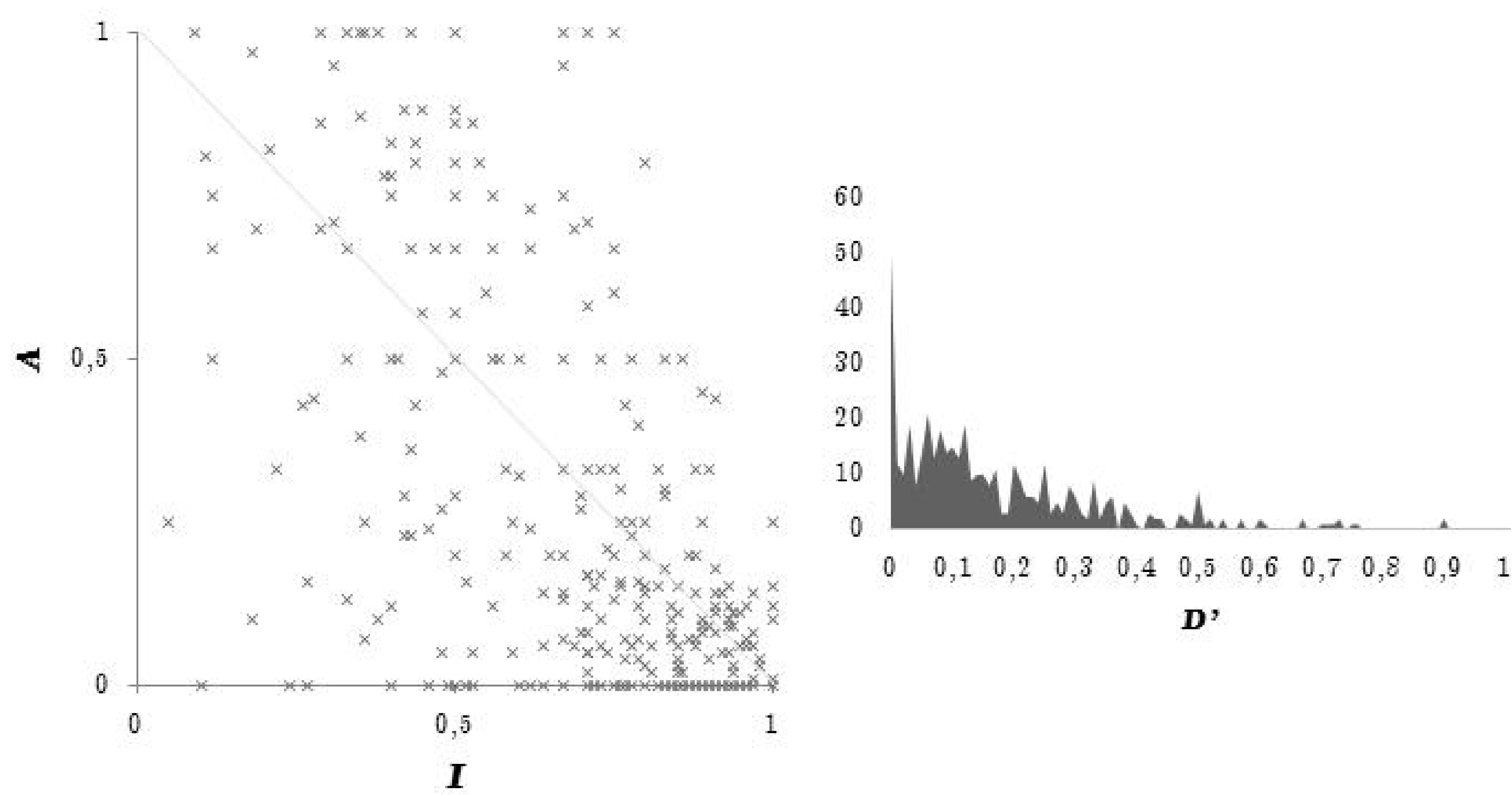
jEdit

N = 16

Q3 = 0.20



# All evaluated packages



	Mean	Median	Min	Q1	Q3	Max	N
Vuze	0.15	0.10	0.0	0.05	0.21	0.90	208
Tomcat	0.18	0.14	0.0	0.01	0.27	0.73	90
ArgoUML	0.20	0.16	0.0	0.07	0.28	0.90	80
Xerces	0.22	0.15	0.0	0.05	0.32	0.75	36
jEdit	0.18	0.11	0.0	0.08	0.20	0.73	16
All	0.17	0.12	0.0	0.05	0.25	0.90	430



# A few observations

- `com.aelitis.azureus.core.util` ( $D'$  is 0.72) and `org.apache.xerces.impl.xml.util` (0.50) offer basic services and data structures. These kinds of classes should be implemented with interfaces if there is *any* possibility of change.
- `org.argouml.application.helpers` (0.76) offers miscellaneous services, and the package could be refactored in two parts



## A few observations

- `org.gtj.sp.jedit.msg` ( $D'$  is 0.72) is in a dependency cycle with `org.gtj.sp.jedit.gui`. The messages should be defined by external interfaces, because messages are volatile.
- `com.aelitis.azureus.login` (0.90) and `org.apache.tomcat.util.res` (0.73) use services only from Java's class library. If dependencies to the class library are ignored, the  $D'$  value of the first package is 0.0 and the second package is 1.0.
  - Clearly, stable services should be ignored



# Conclusion

- Size metrics ( $N_c$  and  $N_a$ ) and coupling metrics ( $C_e$  and  $C_a$ ) are theoretically valid measures. Relative metrics ( $I$  and  $A$ ) can't be tested in the framework.
- The  $H$  is not a valid cohesion metric and we proposed a new version  $H'$  which fulfils the properties
- Five varying size and long developed open source software conforms to Martin's principles
- The metric makes possible to recognize poorly designed packages for refactoring
- The experimental evaluation is only partial, since we couldn't find a huge but badly designed application



# Future work

- How to recognize and handle stable services
  - Java's class library is clearly stable
  - Are all third parties' libraries stable?
- Martin's metric should be statistically tested against some external quality attributes, e.g.:
  - *Maintenance cost of a package*: the best option; unfeasible in practice
  - *Number of changes*: the metric is about the workload of the changes, not the frequency
  - *Number of bugs*: the class level design and coding practices probably cause more bugs than the package level design



# Thanks for attention

- Questions?

