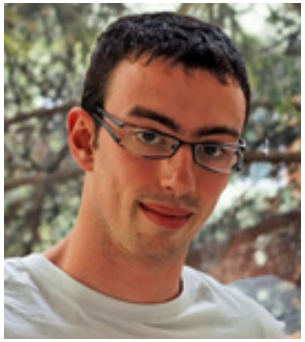
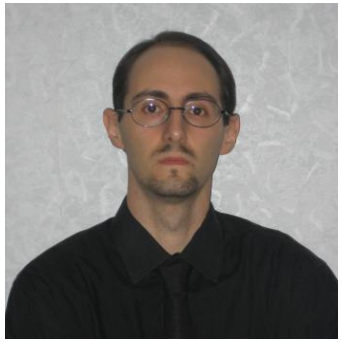
An *Empirical Study* on the Developers' Perception of Software Coupling



Gabriele Bavota

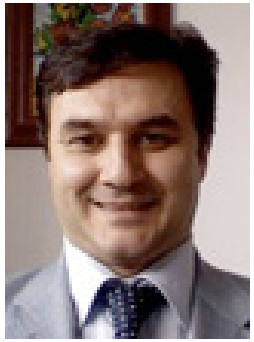
Bogdan Dit

Rocco Oliveto



Massimiliano Di Penta

Denys Poshyvanyk



Andrea De Lucia

SEER U

WILLIAM & MARY

Coupling

"the measure of the *strength* of association established by a connection from one module to another" [Stevens et al., 1974]

Coupling

Coupling

Measure coupling…

Coupling

Measure coupling…

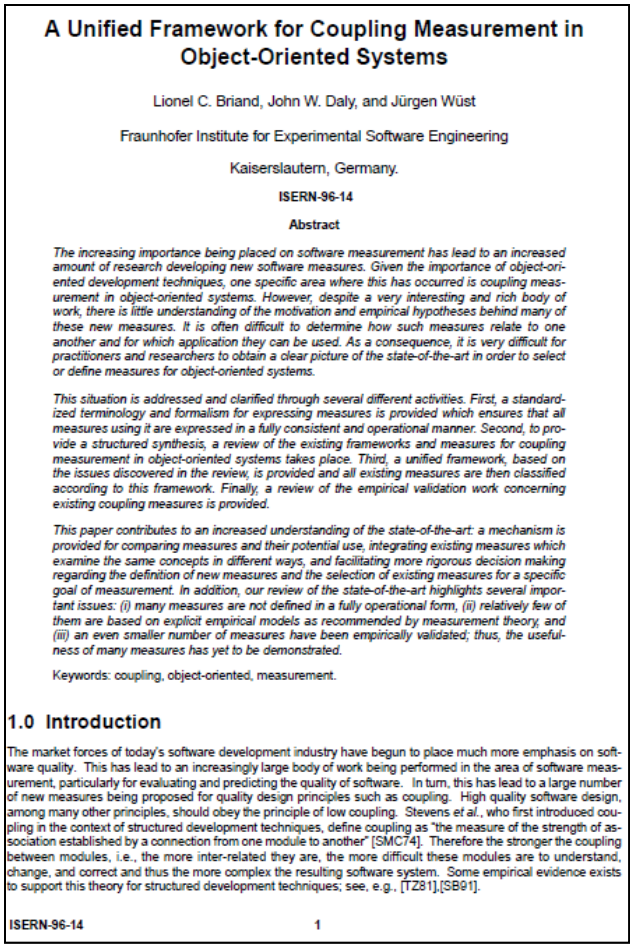
Use coupling…

- predicting fault proneness
- change impact analysis
- software remodularization
- software reuse
- change propagation
- etc.

What are the Typical Types of Information used for Coupling?

What are the Typical Types of Information used for Coupling?

[Briand et al., TSE'99]:
overview of ***structural*** metrics



A Unified Framework for Coupling Measurement in Object-Oriented Systems

Lionel C. Briand, John W. Daly, and Jürgen Wüst

Fraunhofer Institute for Experimental Software Engineering

Kaiserslautern, Germany.

ISERN-96-14

Abstract

The increasing importance being placed on software measurement has lead to an increased amount of research developing new software measures. Given the importance of object-oriented development techniques, one specific area where this has occurred is coupling measurement in object-oriented systems. However, despite a very interesting and rich body of work, there is little understanding of the motivation and empirical hypotheses behind many of these new measures. It is often difficult to determine how such measures relate to one another and for which application they can be used. As a consequence, it is very difficult for practitioners and researchers to obtain a clear picture of the state-of-the-art in order to select or define measures for object-oriented systems.

This situation is addressed and clarified through several different activities. First, a standardized terminology and formalism for expressing measures is provided which ensures that all measures using it are expressed in a fully consistent and operational manner. Second, to provide a structured synthesis, a review of the existing frameworks and measures for coupling measurement in object-oriented systems takes place. Third, a unified framework, based on the issues discovered in the review, is provided and all existing measures are then classified according to this framework. Finally, a review of the empirical validation work concerning existing coupling measures is provided.

This paper contributes to an increased understanding of the state-of-the-art: a mechanism is provided for comparing measures and their potential use, integrating existing measures which examine the same concepts in different ways, and facilitating more rigorous decision making regarding the definition of new measures and the selection of existing measures for a specific goal of measurement. In addition, our review of the state-of-the-art highlights several important issues: (i) many measures are not defined in a fully operational form, (ii) relatively few of them are based on explicit empirical models as recommended by measurement theory, and (iii) an even smaller number of measures have been empirically validated; thus, the usefulness of many measures has yet to be demonstrated.

Keywords: coupling, object-oriented, measurement.

1.0 Introduction

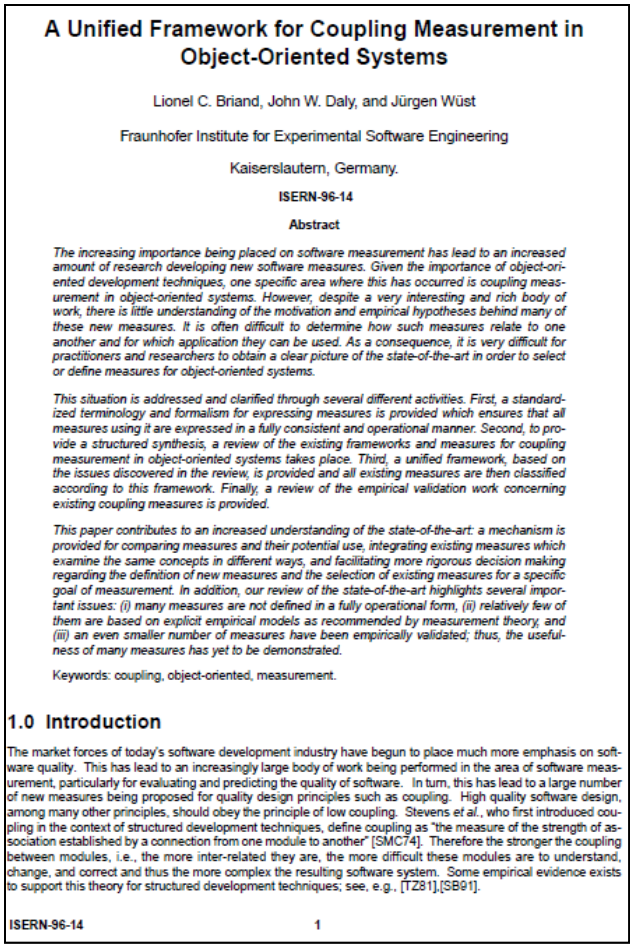
The market forces of today's software development industry have begun to place much more emphasis on software quality. This has lead to an increasingly large body of work being performed in the area of software measurement, particularly for evaluating and predicting the quality of software. In turn, this has lead to a large number of new measures being proposed for quality design principles such as coupling. High quality software design, among many other principles, should obey the principle of low coupling. Stevens *et al.*, who first introduced coupling in the context of structured development techniques, define coupling as "the measure of the strength of association established by a connection from one module to another" [SMC74]. Therefore the stronger the coupling between modules, i.e., the more inter-related they are, the more difficult these modules are to understand, change, and correct and thus the more complex the resulting software system. Some empirical evidence exists to support this theory for structured development techniques; see, e.g., [TZ81],[SB91].

What are the Typical Types of Information used for Coupling?

[Briand et al., TSE'99]:
overview of ***structural*** metrics

Typical types of information:

- **Structural**
- **Dynamic**
- **Semantic**
- **Logical**



A Unified Framework for Coupling Measurement in Object-Oriented Systems

Lionel C. Briand, John W. Daly, and Jürgen Wüst

Fraunhofer Institute for Experimental Software Engineering

Kaiserslautern, Germany.

ISERN-96-14

Abstract

The increasing importance being placed on software measurement has lead to an increased amount of research developing new software measures. Given the importance of object-oriented development techniques, one specific area where this has occurred is coupling measurement in object-oriented systems. However, despite a very interesting and rich body of work, there is little understanding of the motivation and empirical hypotheses behind many of these new measures. It is often difficult to determine how such measures relate to one another and for which application they can be used. As a consequence, it is very difficult for practitioners and researchers to obtain a clear picture of the state-of-the-art in order to select or define measures for object-oriented systems.

This situation is addressed and clarified through several different activities. First, a standardized terminology and formalism for expressing measures is provided which ensures that all measures using it are expressed in a fully consistent and operational manner. Second, to provide a structured synthesis, a review of the existing frameworks and measures for coupling measurement in object-oriented systems takes place. Third, a unified framework, based on the issues discovered in the review, is provided and all existing measures are then classified according to this framework. Finally, a review of the empirical validation work concerning existing coupling measures is provided.

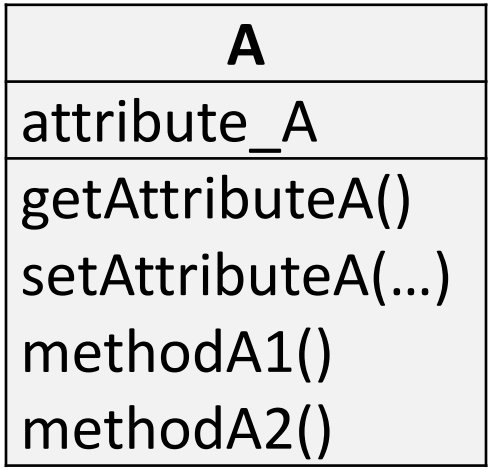
This paper contributes to an increased understanding of the state-of-the-art: a mechanism is provided for comparing measures and their potential use, integrating existing measures which examine the same concepts in different ways, and facilitating more rigorous decision making regarding the definition of new measures and the selection of existing measures for a specific goal of measurement. In addition, our review of the state-of-the-art highlights several important issues: (i) many measures are not defined in a fully operational form, (ii) relatively few of them are based on explicit empirical models as recommended by measurement theory, and (iii) an even smaller number of measures have been empirically validated; thus, the usefulness of many measures has yet to be demonstrated.

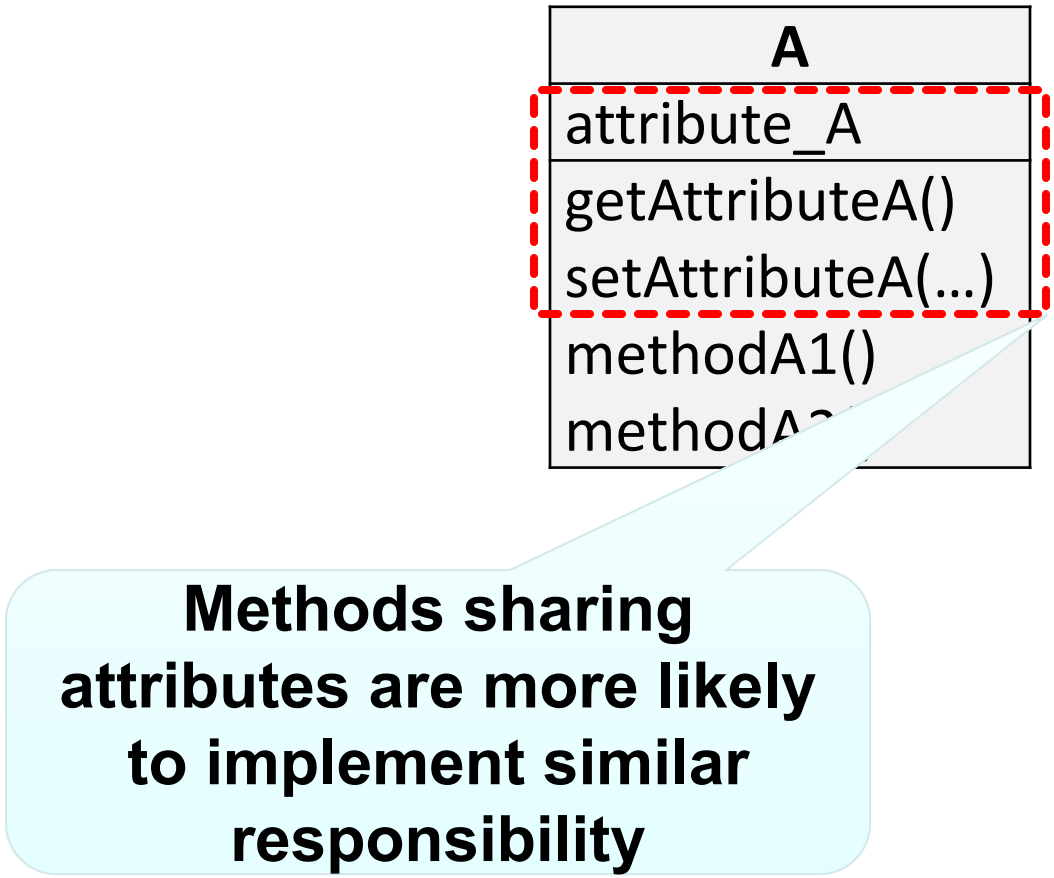
Keywords: coupling, object-oriented, measurement.

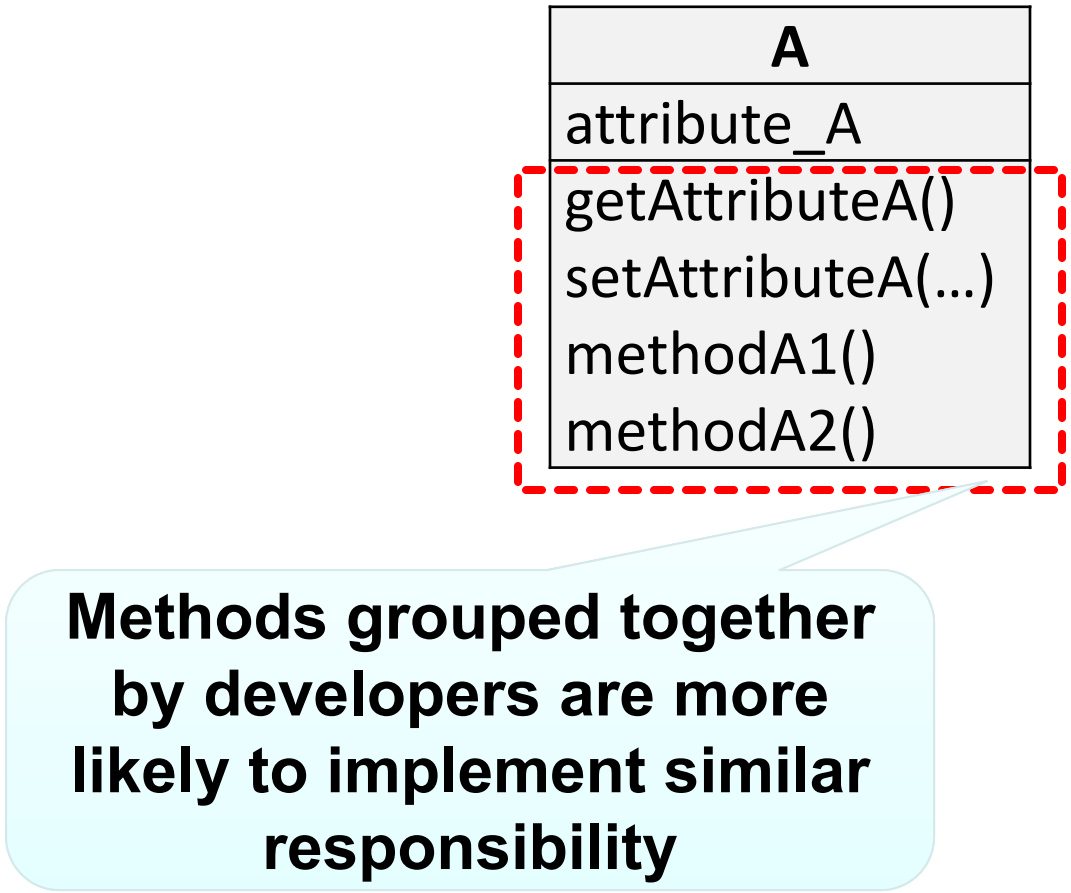
1.0 Introduction

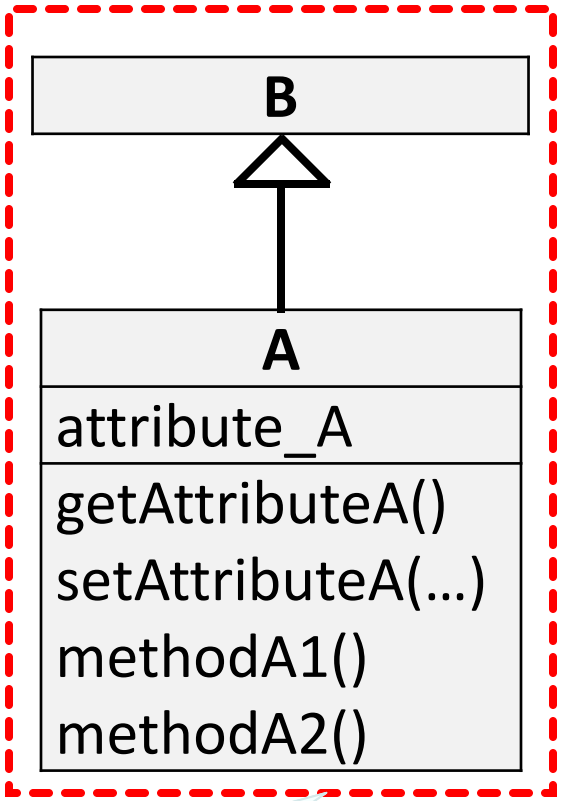
The market forces of today's software development industry have begun to place much more emphasis on software quality. This has lead to an increasingly large body of work being performed in the area of software measurement, particularly for evaluating and predicting the quality of software. In turn, this has lead to a large number of new measures being proposed for quality design principles such as coupling. High quality software design, among many other principles, should obey the principle of low coupling. Stevens *et al.*, who first introduced coupling in the context of structured development techniques, define coupling as "the measure of the strength of association established by a connection from one module to another" [SMC74]. Therefore the stronger the coupling between modules, i.e., the more inter-related they are, the more difficult these modules are to understand, change, and correct and thus the more complex the resulting software system. Some empirical evidence exists to support this theory for structured development techniques; see, e.g., [TZ81],[SB91].

Structural

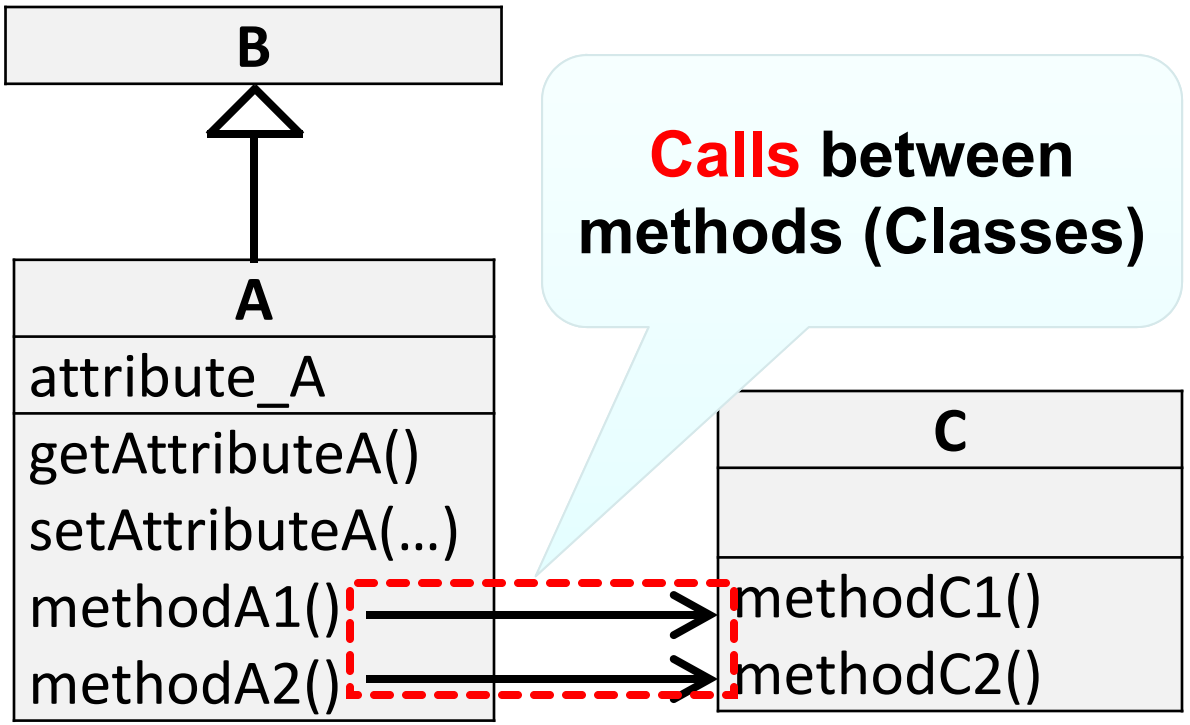

A
attribute_A
getAttributeA()
setAttributeA(…)
methodA1()
methodA2()

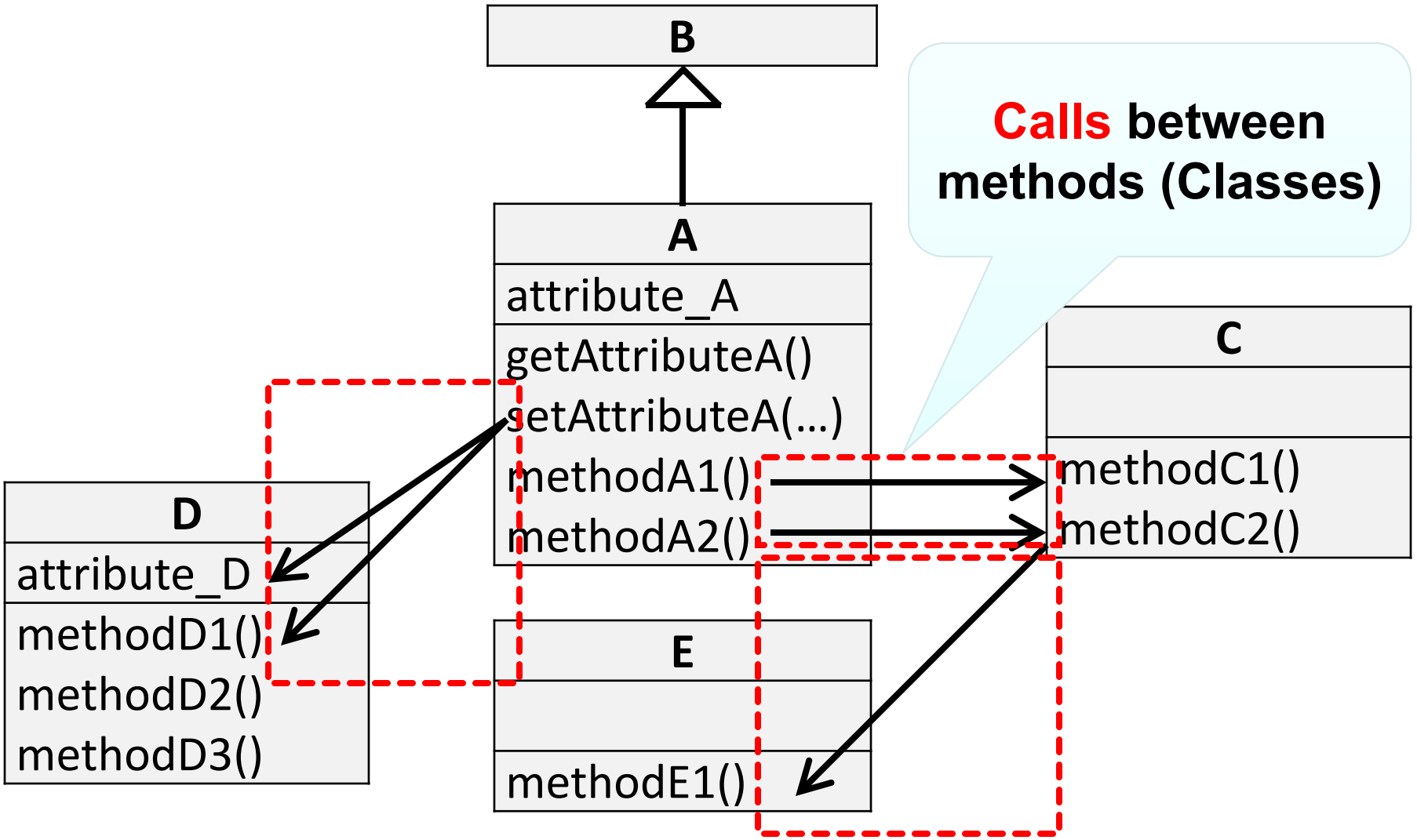

A
attribute_A
getAttributeA()
setAttributeA(…)
methodA1()
Methods sharing attributes are more likely to implement similar responsibility


A
attribute_A
getAttributeA()
setAttributeA(…)
methodA1()
methodA2()
Methods grouped together by developers are more likely to implement similar responsibility



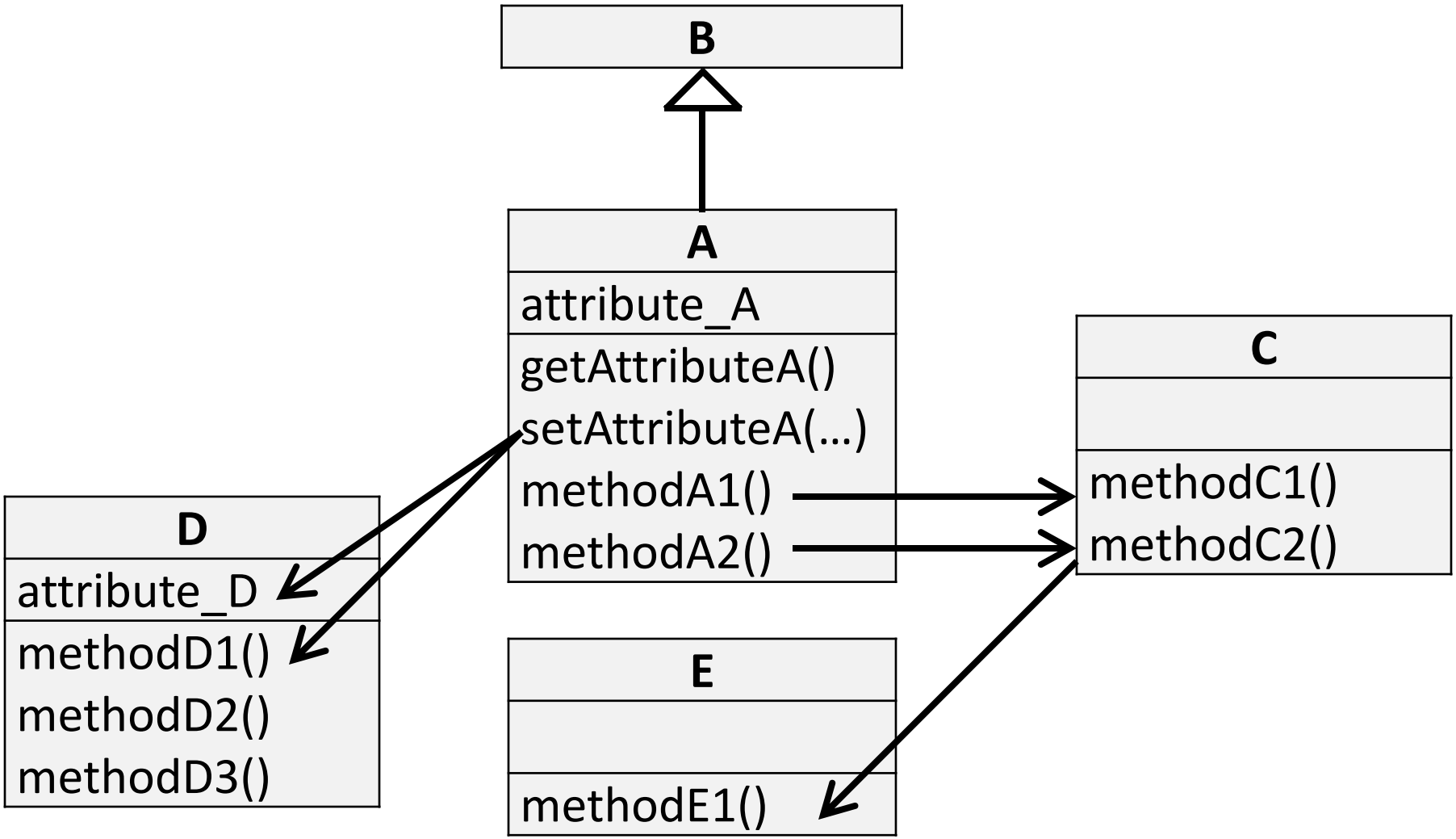
Classes having *inheritance* relations are more likely to implement similar responsibility

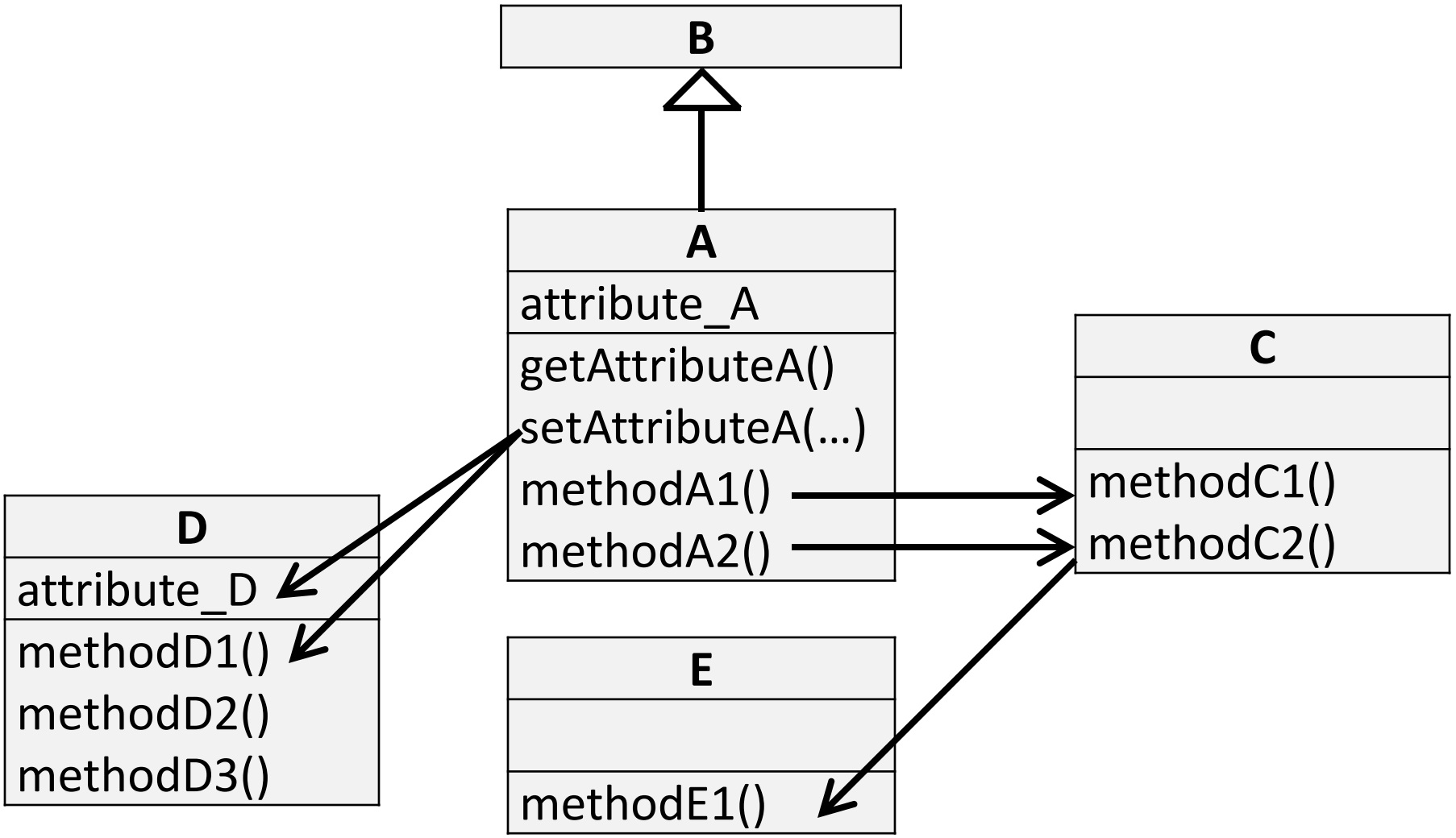

B
A
attribute_A
getAttributeA()
setAttributeA(…)
methodA1()
methodA2()
Calls between methods (Classes)
C
methodC1()
methodC2()


B
A
attribute_A
getAttributeA()
setAttributeA(...)
methodA1()
methodA2()
C
methodC1()
methodC2()
D
attribute_D
methodD1()
methodD2()
methodD3()
E
methodE1()
Calls between methods (Classes)

Structural

Dynamic

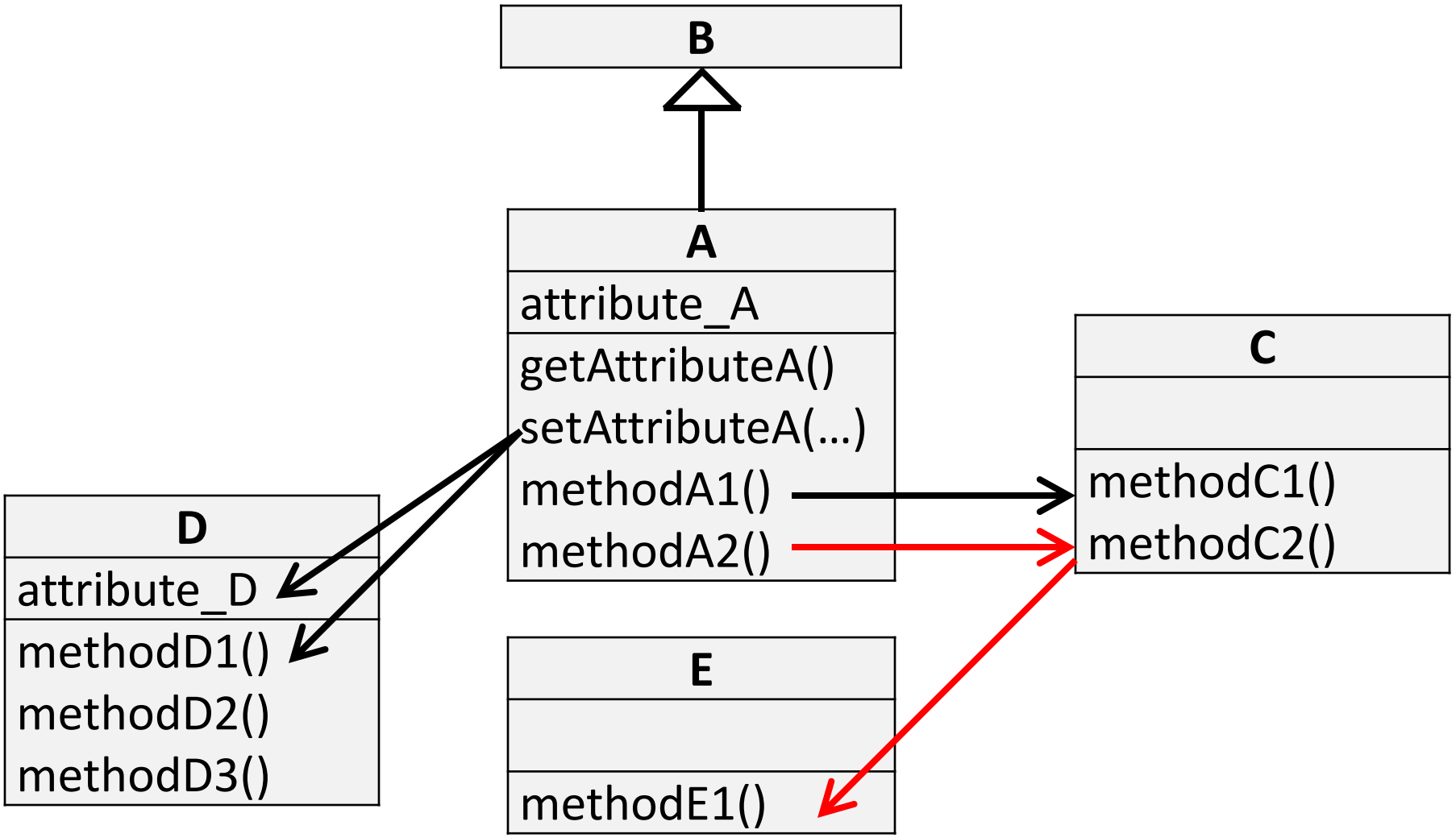

B
A
attribute_A
getAttributeA()
setAttributeA(…)
methodA1()
methodA2()
C
methodC1()
methodC2()
D
attribute_D
methodD1()
methodD2()
methodD3()
E
methodE1()



Execution Trace

A.methodA2() → C.methodC2()

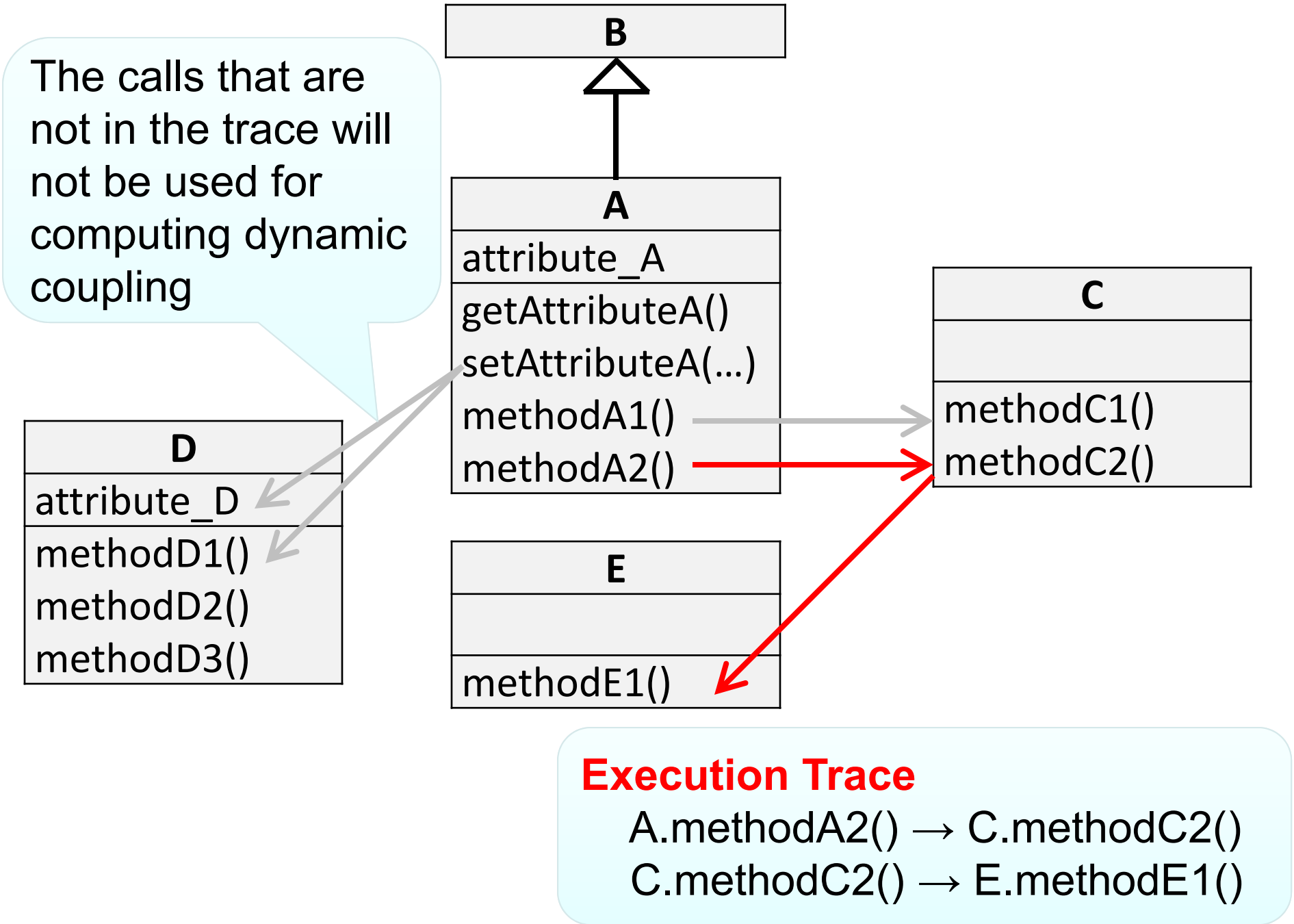
C.methodC2() → E.methodE1()

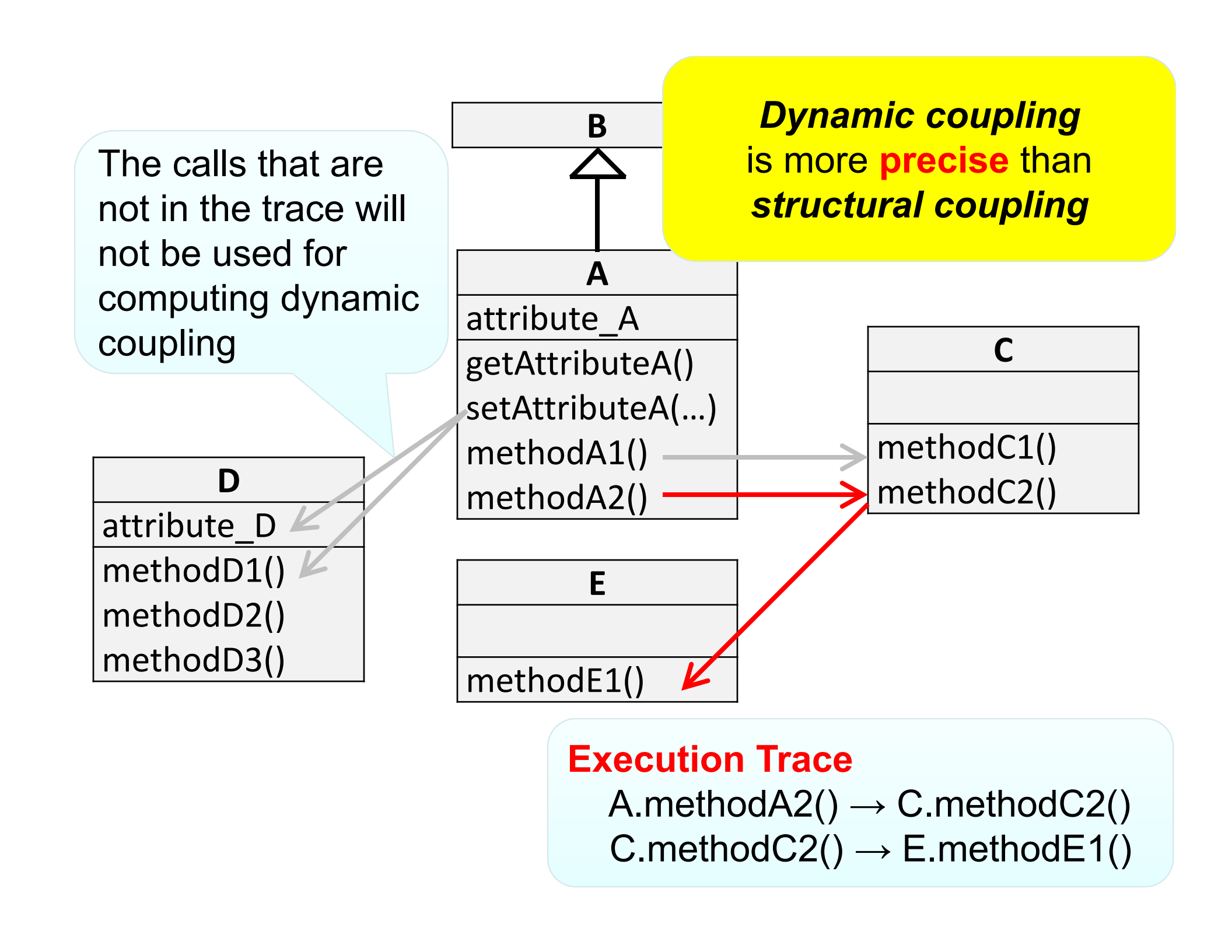


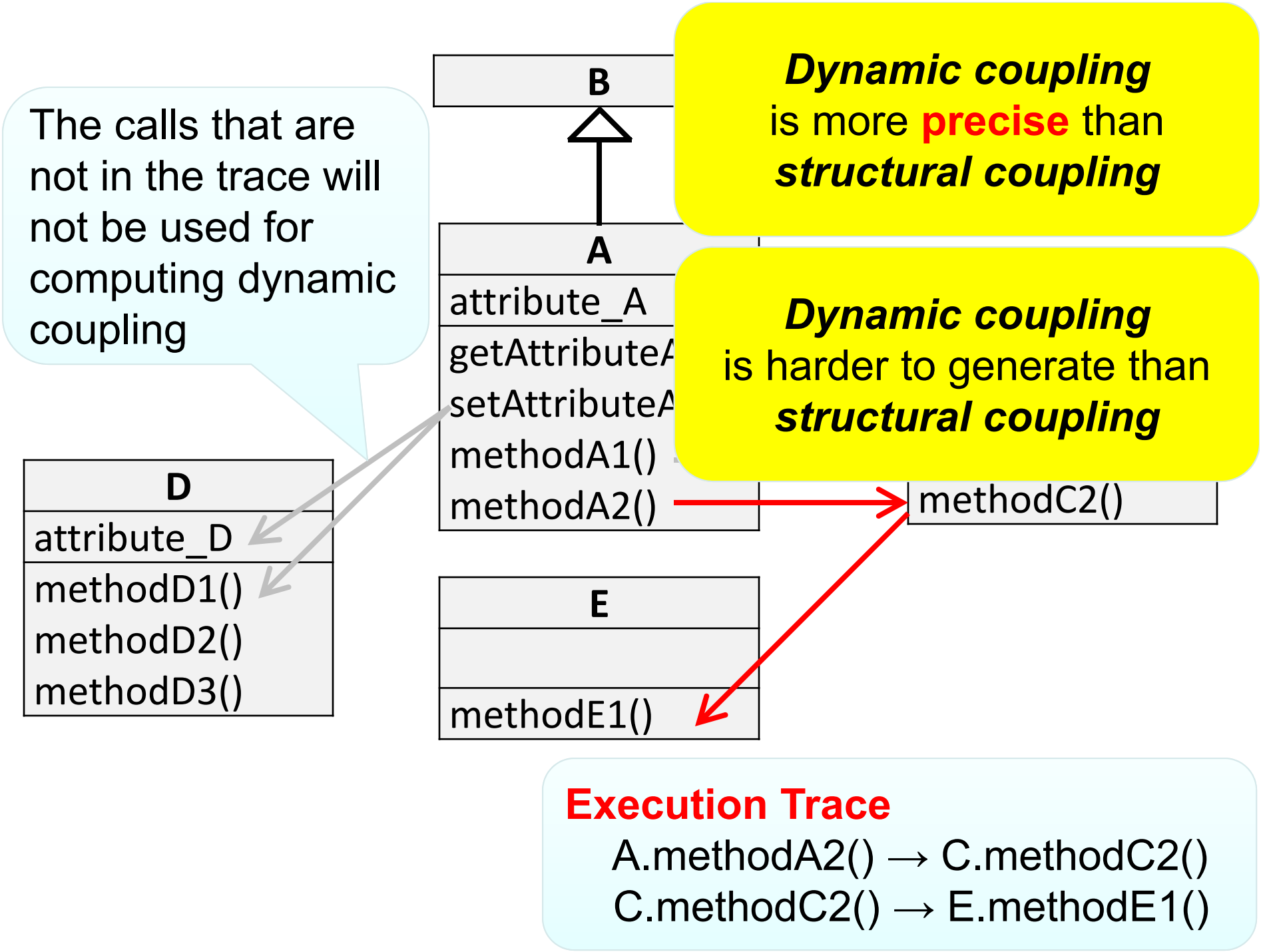
Execution Trace

A.methodA2() → C.methodC2()

C.methodC2() → E.methodE1()


The calls that are not in the trace will not be used for computing dynamic coupling
B
A
attribute_A
getAttributeA()
setAttributeA(...)
methodA1()
methodA2()
C
methodC1()
methodC2()
D
attribute_D
methodD1()
methodD2()
methodD3()
E
methodE1()
Execution Trace
A.methodA2() → C.methodC2()
C.methodC2() → E.methodE1()


B
A
attribute_A
getAttributeA()
setAttributeA(...)
methodA1()
methodA2()
C
methodC1()
methodC2()
D
attribute_D
methodD1()
methodD2()
methodD3()
E
methodE1()
The calls that are not in the trace will not be used for computing dynamic coupling
Dynamic coupling is more precise than structural coupling
Execution Trace
A.methodA2() → C.methodC2()
C.methodC2() → E.methodE1()


B
A
attribute_A
getAttributeA
setAttributeA
methodA1()
methodA2()
methodC2()
D
attribute_D
methodD1()
methodD2()
methodD3()
E
methodE1()
The calls that are not in the trace will not be used for computing dynamic coupling
Dynamic coupling is more precise than structural coupling
Dynamic coupling is harder to generate than structural coupling
Execution Trace
A.methodA2() → C.methodC2()
C.methodC2() → E.methodE1()

Structural

Dynamic

Semantic

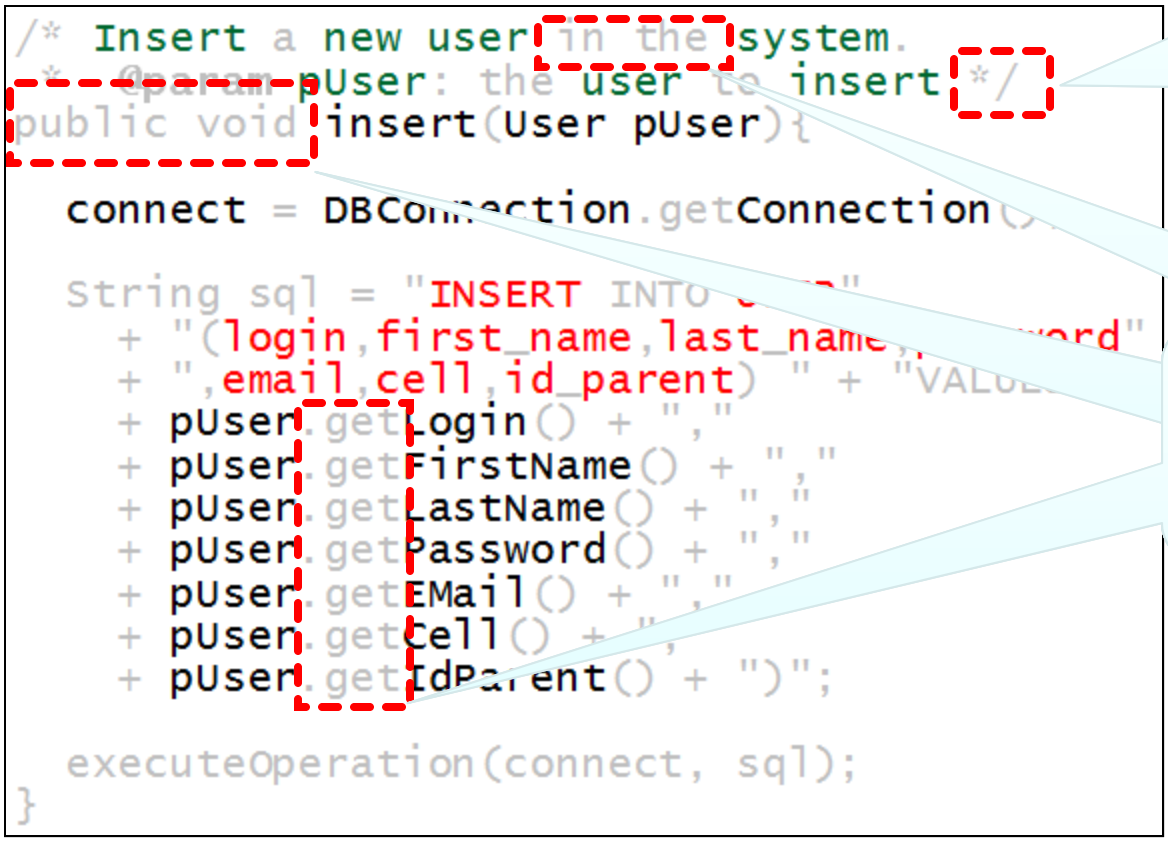
```java
/* Insert a new user in the system.
 *  @param pUser: the user to insert.*/
public void insert(User pUser){

  connect = DBConnection.getConnection();

  String sql = "INSERT INTO USER"
    + "(login,first_name,last_name,password"
    + ",email,cell,id_parent) " + "VALUES ("
    + pUser.getLogin() + ","
    + pUser.getFirstName() + ","
    + pUser.getLastName() + ","
    + pUser.getPassword() + ","
    + pUser.getEMail() + ","
    + pUser.getCell() + ","
    + pUser.getIdParent() + ")";

  executeOperation(connect, sql);
}
```

```java
/* Delete an user from the system.
 *  @param pUser: the user to delete.*/
public void delete(User pUser) {

    connect = DBConnection.getConnection();

    String sql = "DELETE FROM USER "
        + "WHERE id_user = "
        + pUser.getId();

    executeOperation(connect, sql);
}
```

```java
/* Insert a new user in the system.
 *  @param pUser: the user to insert*/
public void insert(User pUser){

  connect = DBConnection.getConnection();

  String sql = "INSERT INTO USER"
    + "(login,first_name,last_name,password"
    + ",email,cell,id_parent) " + "VALUES ("
    + pUser.getLogin() + ","
    + pUser.getFirstName() + ","
    + pUser.getLastName() + ","
    + pUser.getPassword() + ","
    + pUser.getEMail() + ","
    + pUser.getCell() + ","
    + pUser.getIdParent() + ")";

  executeOperation(connect, sql);
}
```

Remove special characters

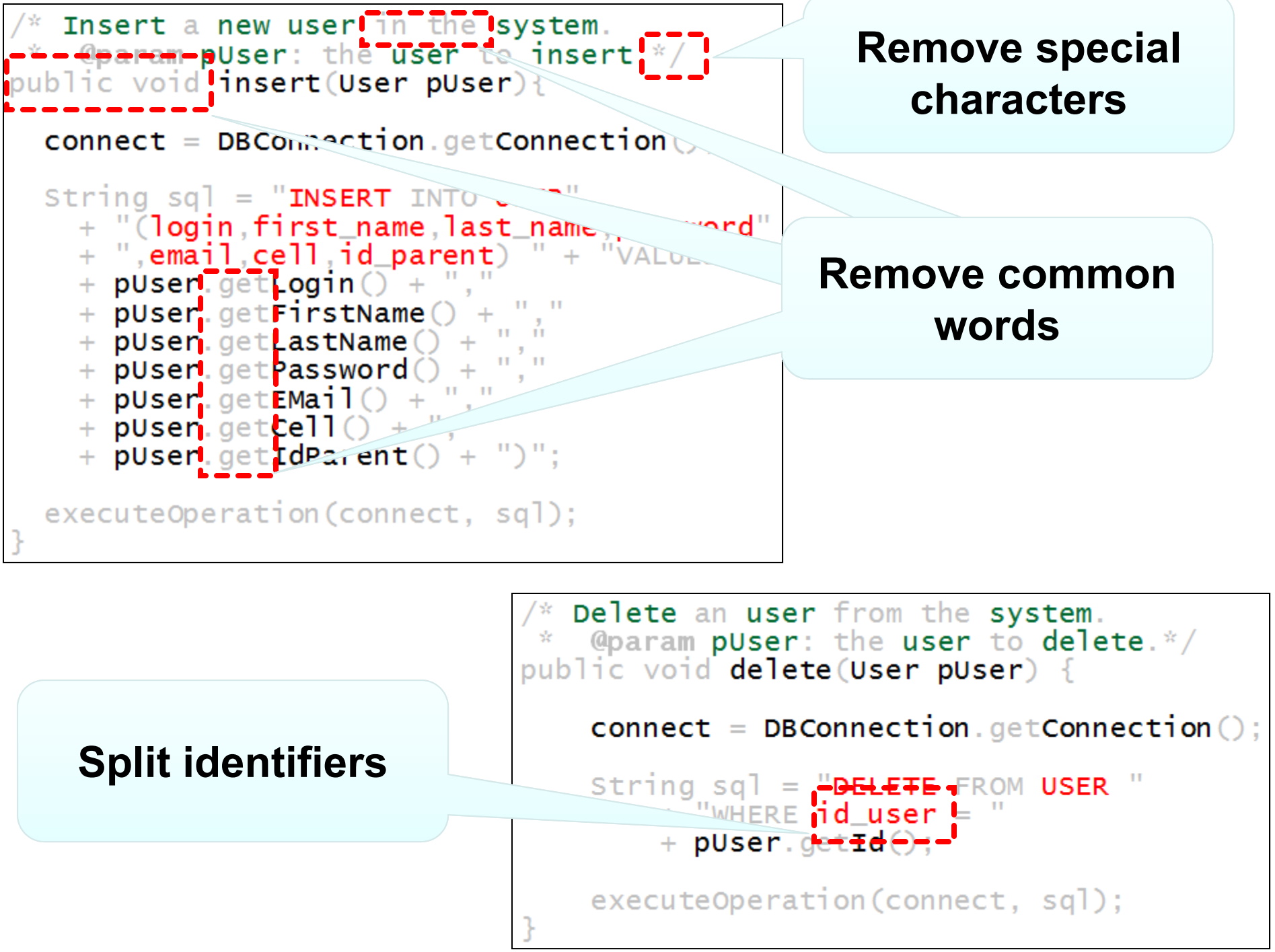
```java
/* Delete an user from the system.
 *  @param pUser: the user to delete.*/
public void delete(User pUser) {

    connect = DBConnection.getConnection();

    String sql = "DELETE FROM USER "
        + "WHERE id_user = "
        + pUser.getId();

    executeOperation(connect, sql);
}
```



Remove special characters

Remove common words

```
/* Delete an user from the system.
 *  @param pUser: the user to delete.*/
public void delete(User pUser) {

    connect = DBConnection.getConnection();

    String sql = "DELETE FROM USER "
        + "WHERE id_user = "
        + pUser.getId();

    executeOperation(connect, sql);
}
```


/* Insert a new user in the system.
* @param pUser: the user to insert */
public void insert(User pUser){
connect = DBConnection.getConnection();
String sql = "INSERT INTO
+ "(login,first_name,last_name,
+ ",email,cell,id_parent) " + "VALU
+ pUser.getLogin() + ","
+ pUser.getFirstName() + ","
+ pUser.getLastName() + ","
+ pUser.getPassword() + ","
+ pUser.getEMail() + ","
+ pUser.getCell() + ","
+ pUser.getIdParent() + ")";
executeOperation(connect, sql);
}
Remove special characters
Remove common words
Split identifiers
/* Delete an user from the system.
* @param pUser: the user to delete.*/
public void delete(User pUser) {
connect = DBConnection.getConnection();
String sql = "DELETE FROM USER "
+ "WHERE id_user = "
+ pUser.getId();
executeOperation(connect, sql);
}

```java
/* Insert a new user in the system.
 *  @param pUser: the user to insert.*/
public void insert(User pUser){

  connect = DBConnection.getConnection();

  String sql = "INSERT INTO USER"
    + "(login,first_name,last_name,password"
    + ",email,cell,id_parent) " + "VALUES ("
    + pUser.getLogin() + ","
    + pUser.getFirstName() + ","
    + pUser.getLastName() + ","
    + pUser.getPassword() + ","
    + pUser.getEMail() + ","
    + pUser.getCell() + ","
    + pUser.getIdParent() + ")";

  executeOperation(connect, sql);
}
```

“Textual” coupling…

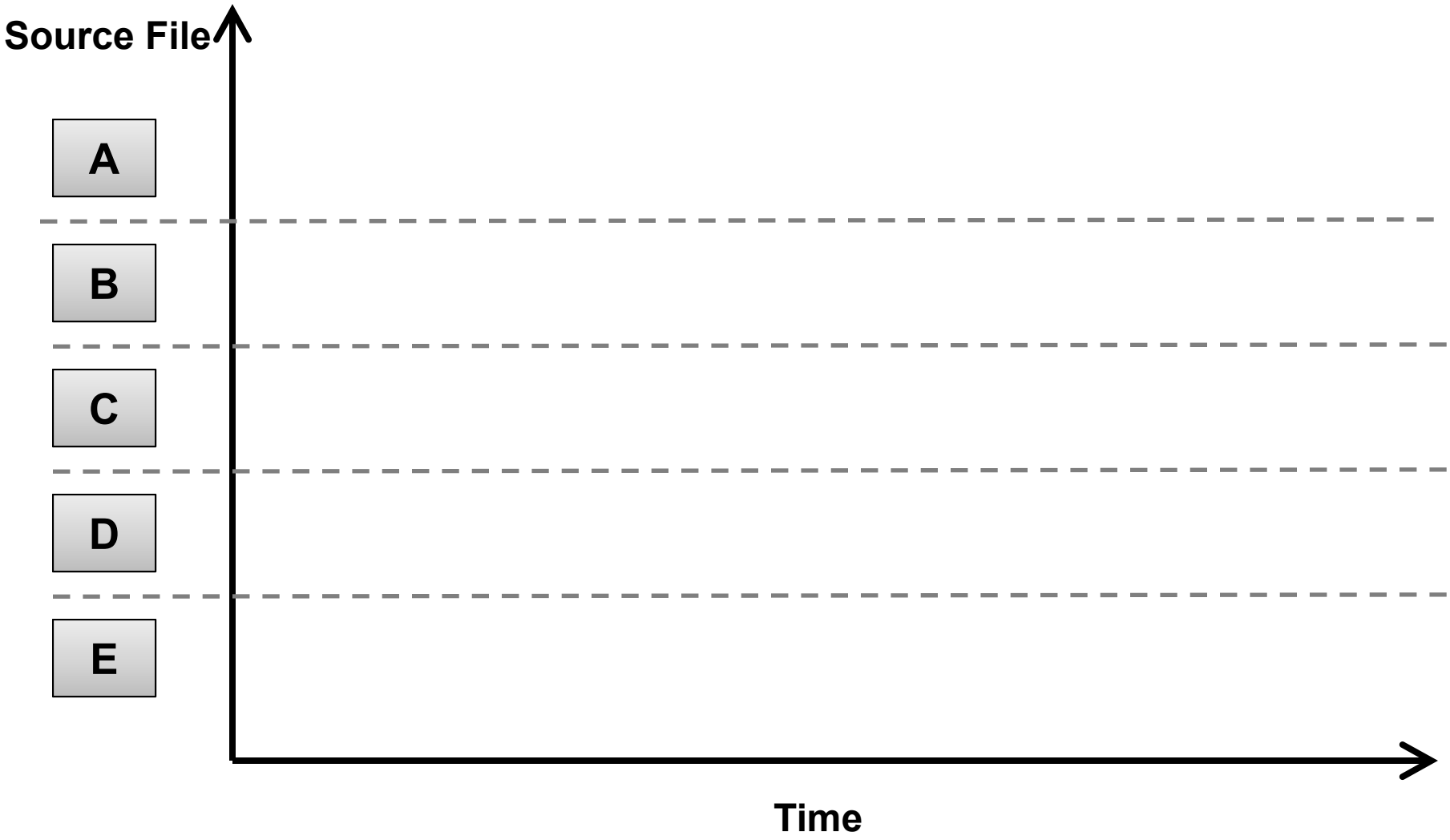
```java
/* Delete an user from the system.
 *  @param pUser: the user to delete.*/
public void delete(User pUser) {

    connect = DBConnection.getConnection();

    String sql = "DELETE FROM USER "
        + "WHERE id_user = "
        + pUser.getId();

    executeOperation(connect, sql);
}
```

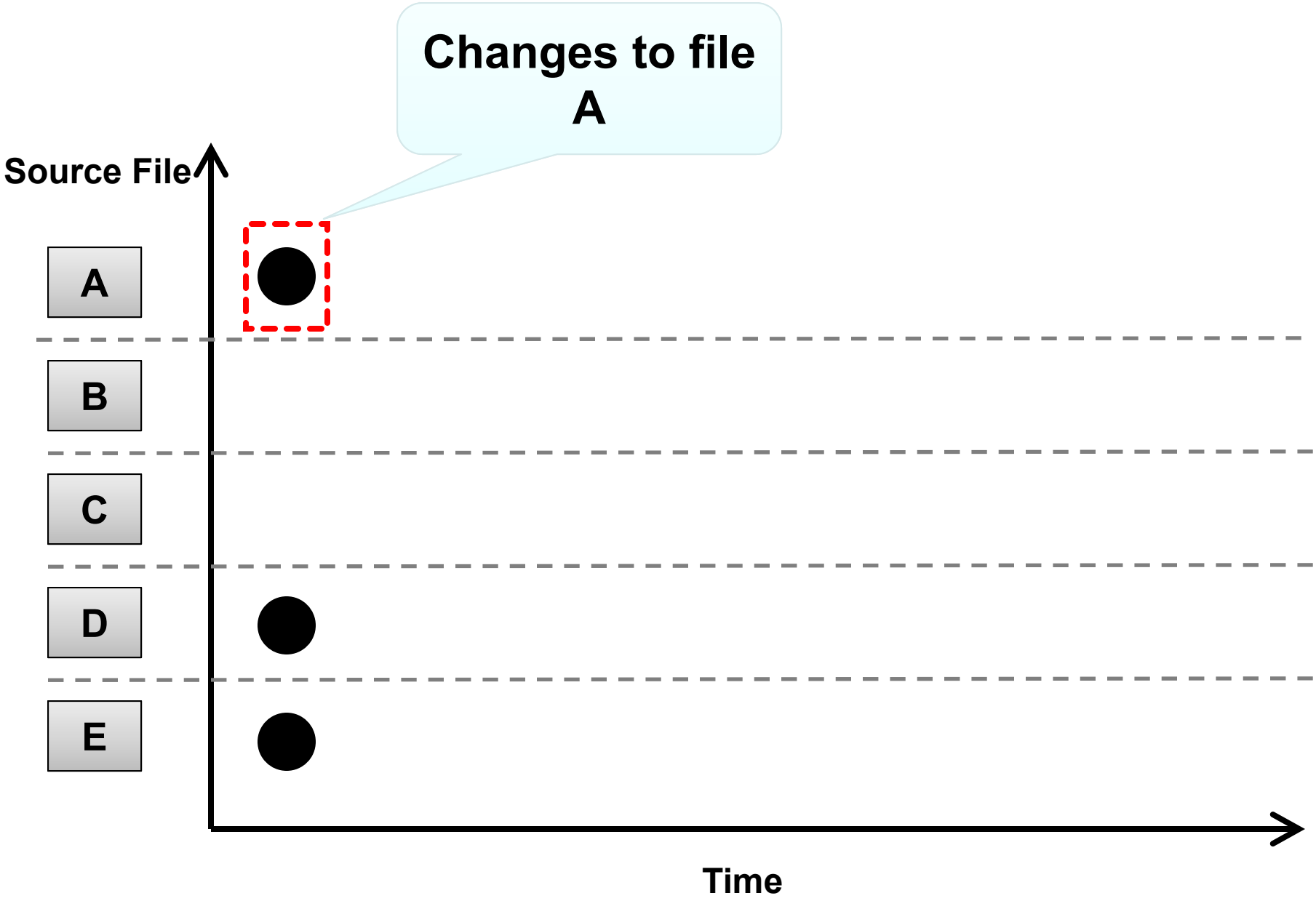
Structural

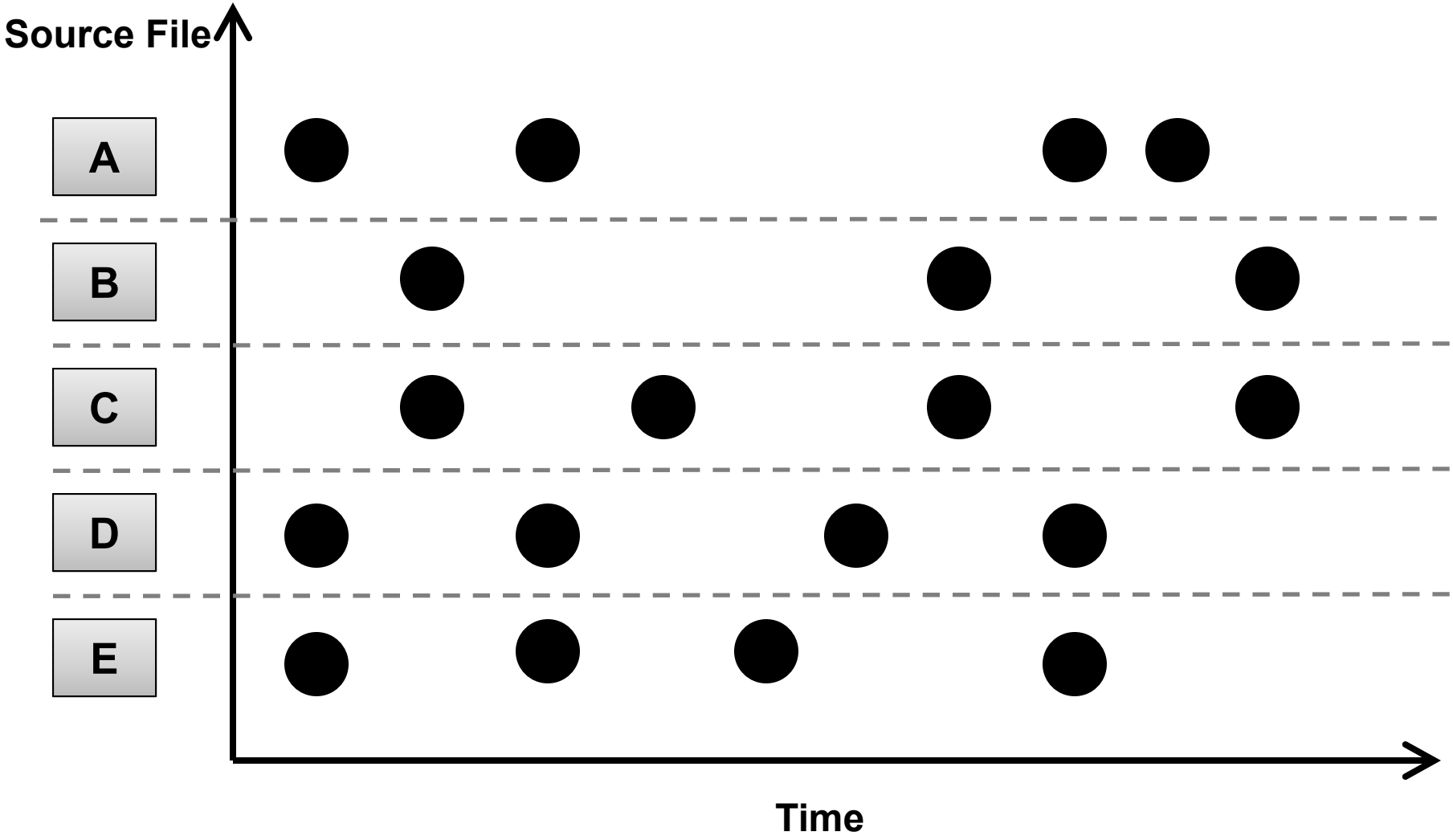
Dynamic

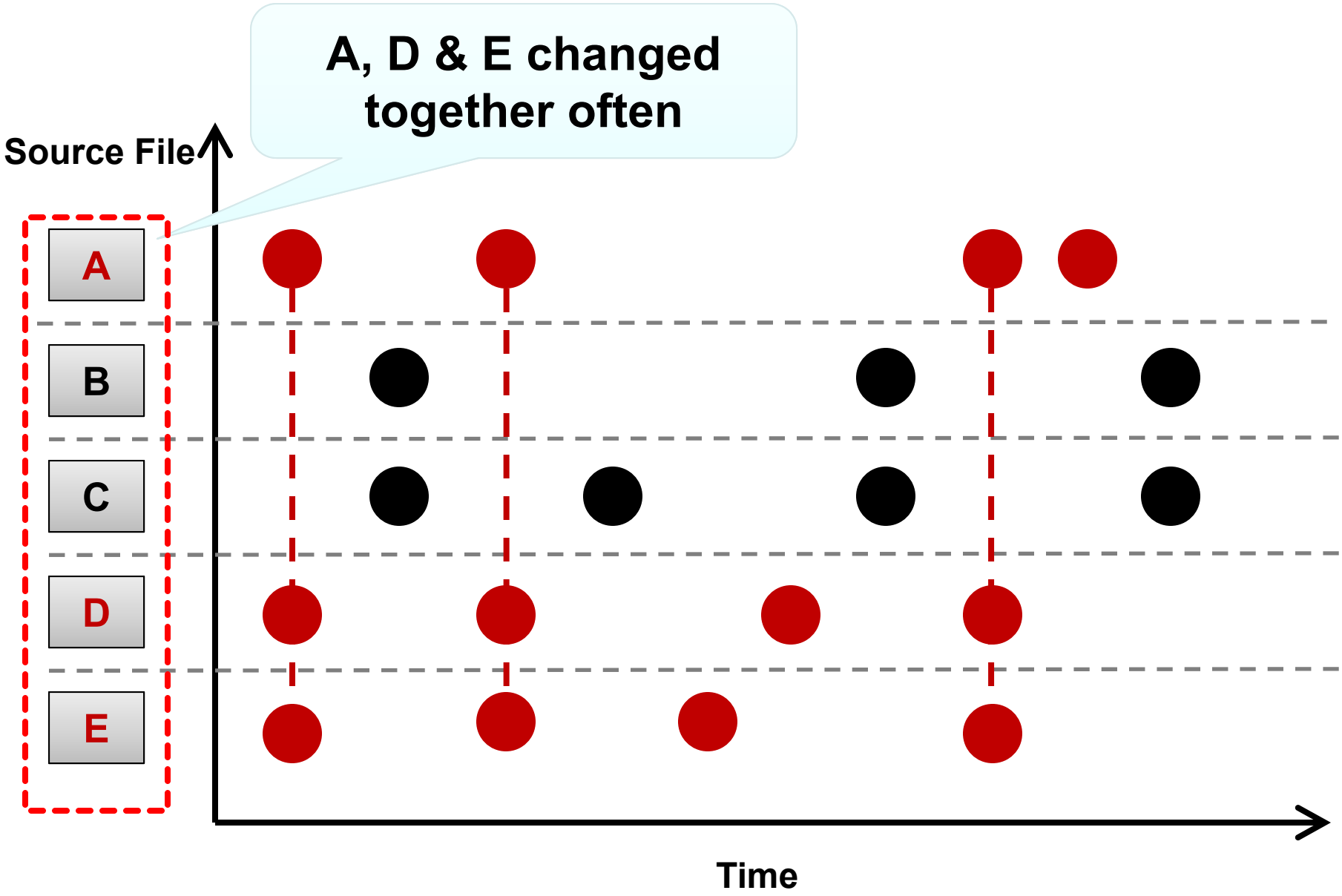
Semantic

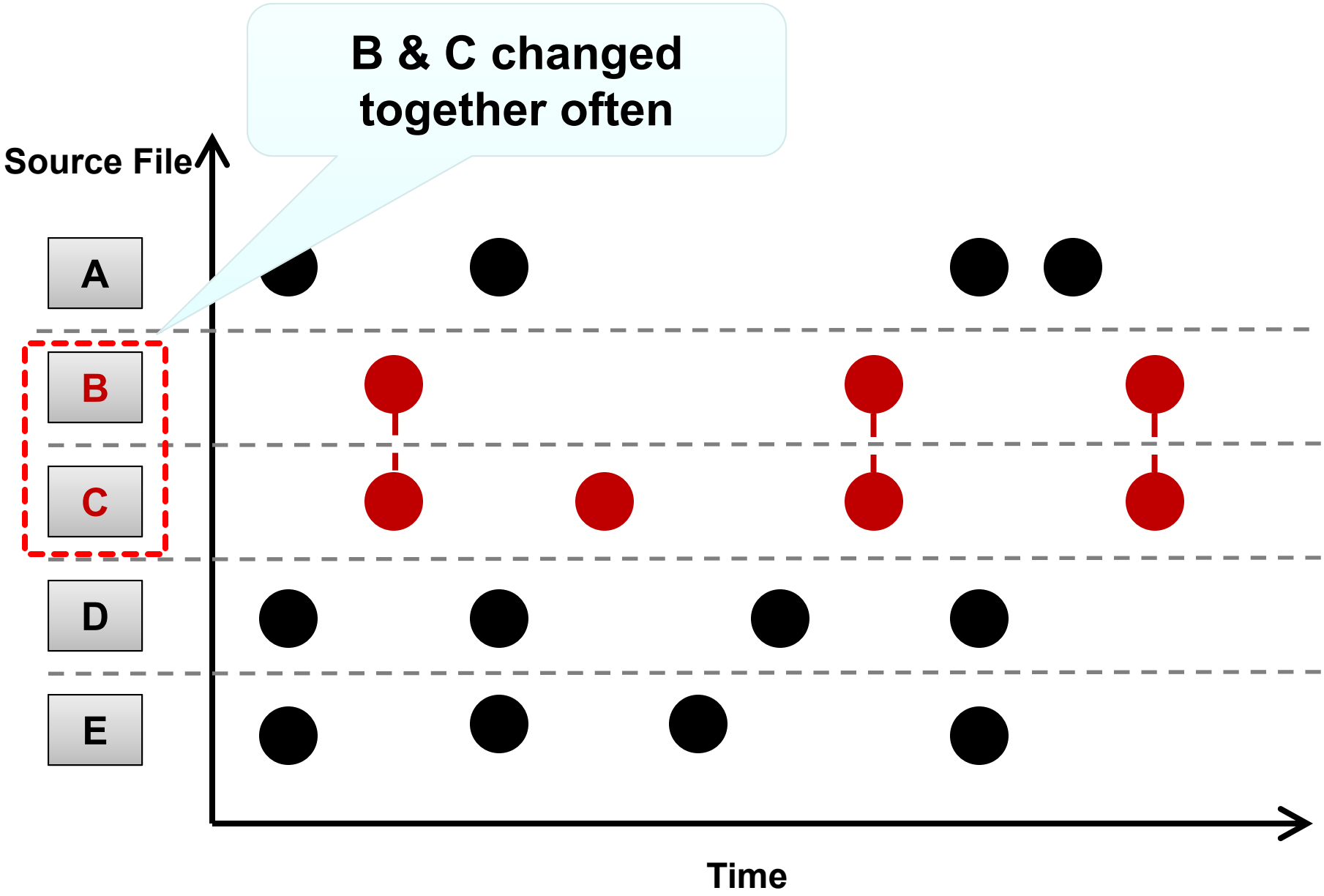
Logical (Evolutionary)

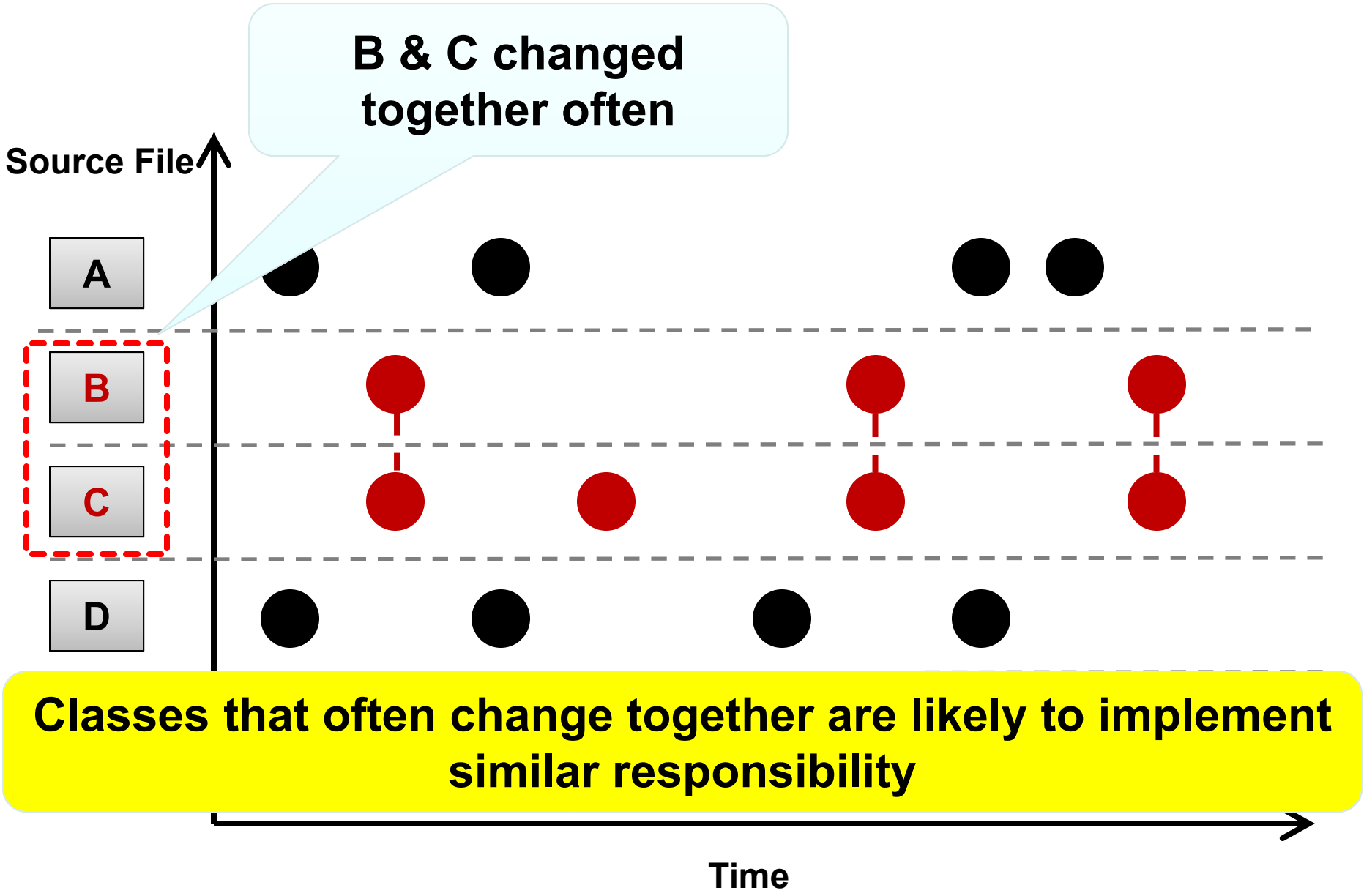

Source File
A
B
C
D
E
Time


Changes to file A
Source File
A
B
C
D
E
Time


Source File
A
B
C
D
E
Time


A, D & E changed together often
Source File
A
B
C
D
E
Time


B & C changed
together often
Source File
A
B
C
D
E
Time


B & C changed together often
Source File
A
B
C
D
Classes that often change together are likely to implement similar responsibility
Time

Which type of ***coupling*** aligns better with developers' perception?

Which type of ***coupling*** aligns better with developers' perception?

Empirical study with 76 developers

Choosing Software Systems

JHotDraw

JHotDraw

Size	200 KLOC	29 KLOC	28 KLOC

JHotDraw

Size	200 KLOC	29 KLOC	28 KLOC
# of classes	1,889	289	245
Vocabulary	7,961	2,834	2,418

JHotDraw

Size	200 KLOC	29 KLOC	28 KLOC
# of classes	1,889	289	245
Vocabulary	7,961	2,834	2,418

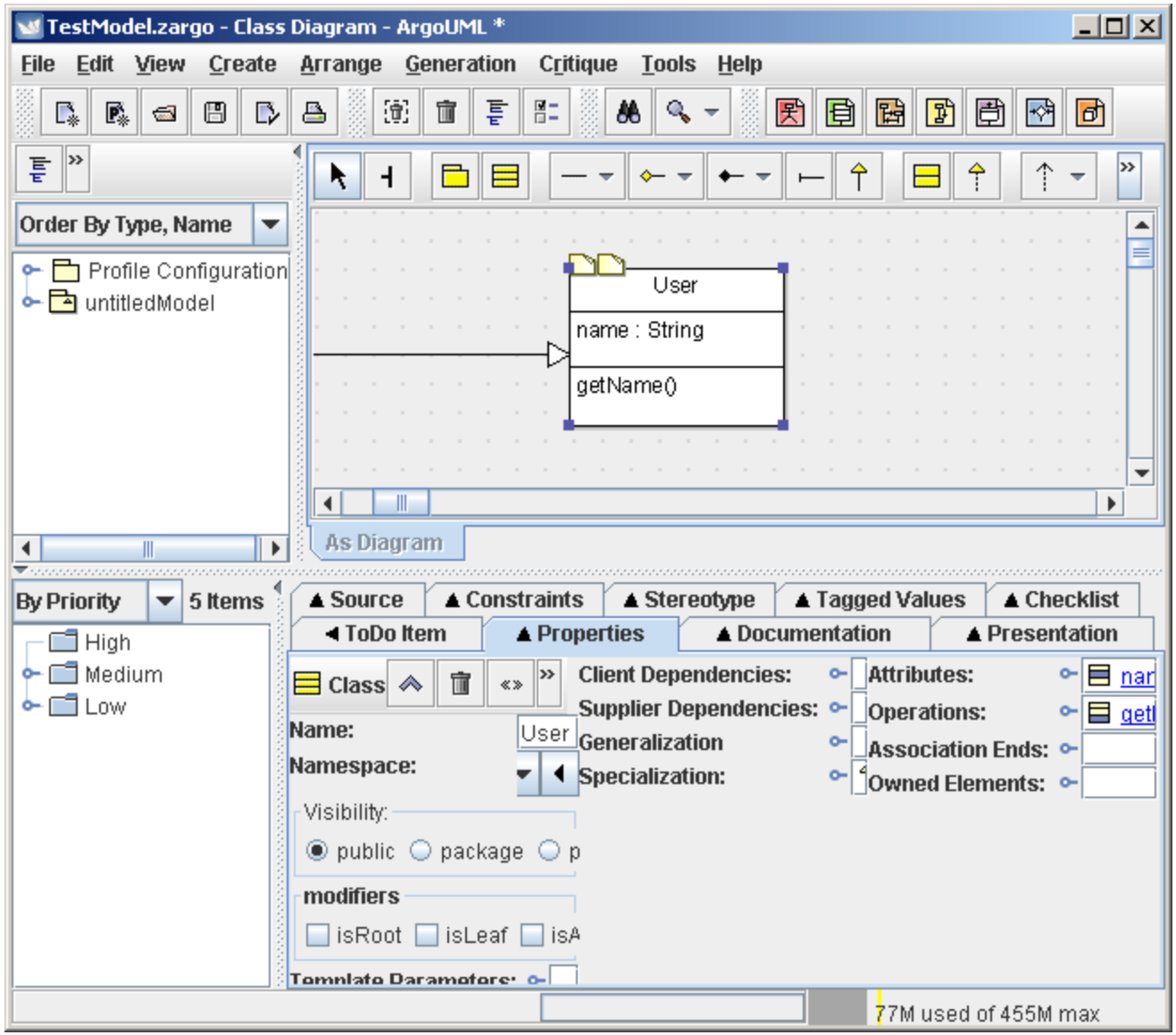
Historical	8 years	5 years	12 years

JHotDraw

Size	200 KLOC	29 KLOC	28 KLOC
# of classes	1,889	289	245
Vocabulary	7,961	2,834	2,418
Historical	8 years	5 years	12 years
Execution Traces Coverage			

No test cases available…

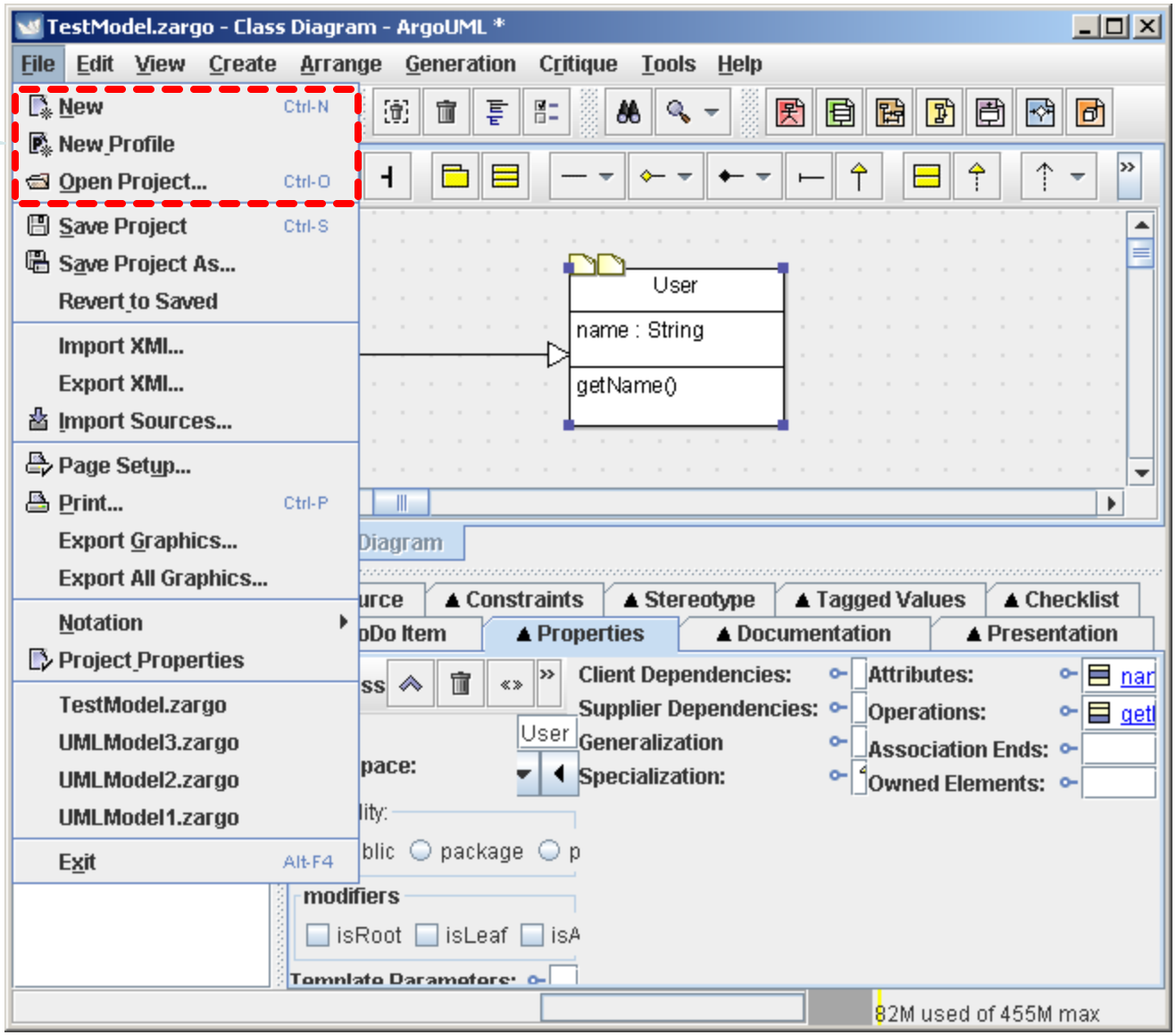
ArgoUML



No test cases available…

ArgoUML

Trace 1

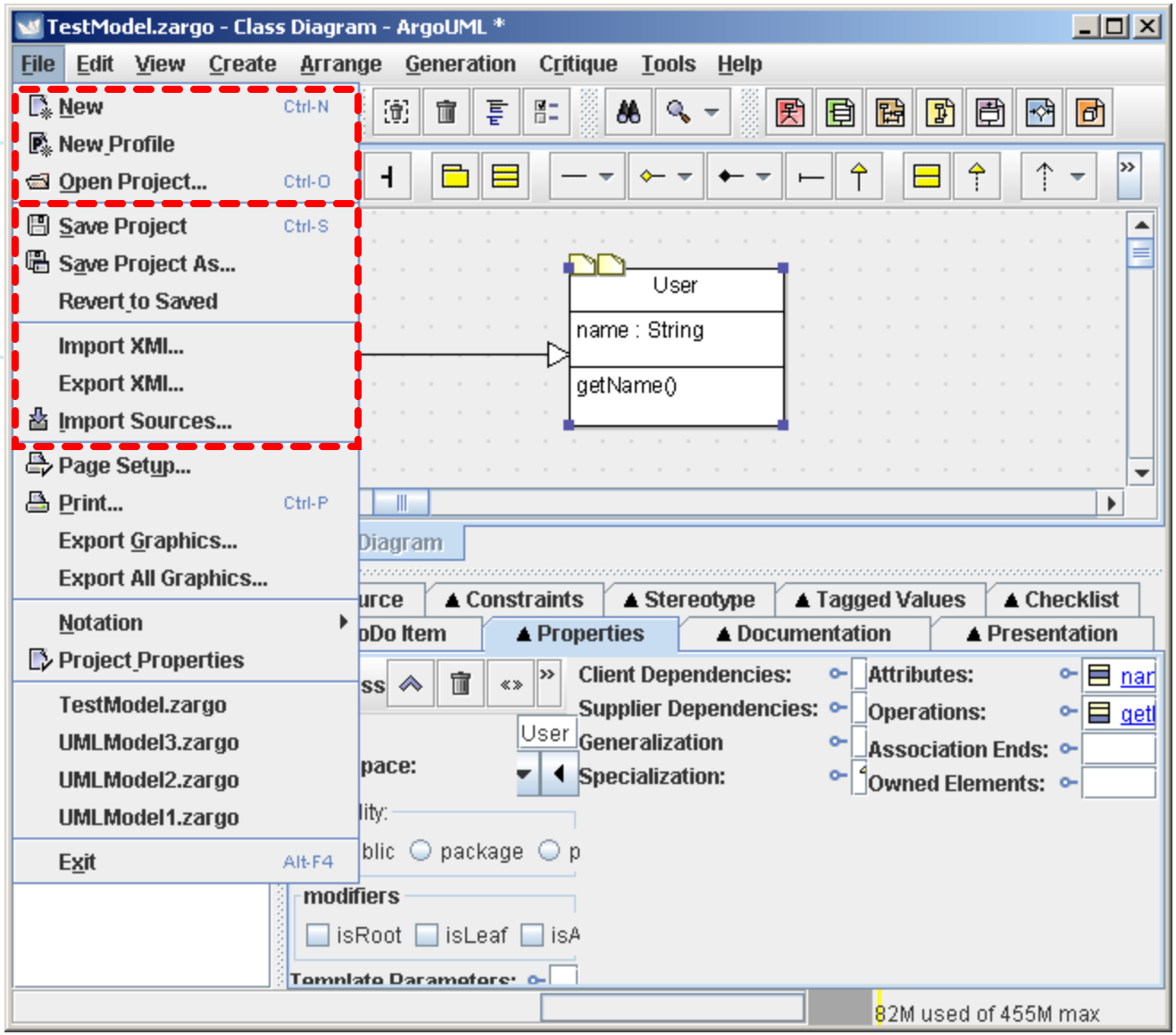


No test cases available…

ArgoUML

Trace 1

Trace 2



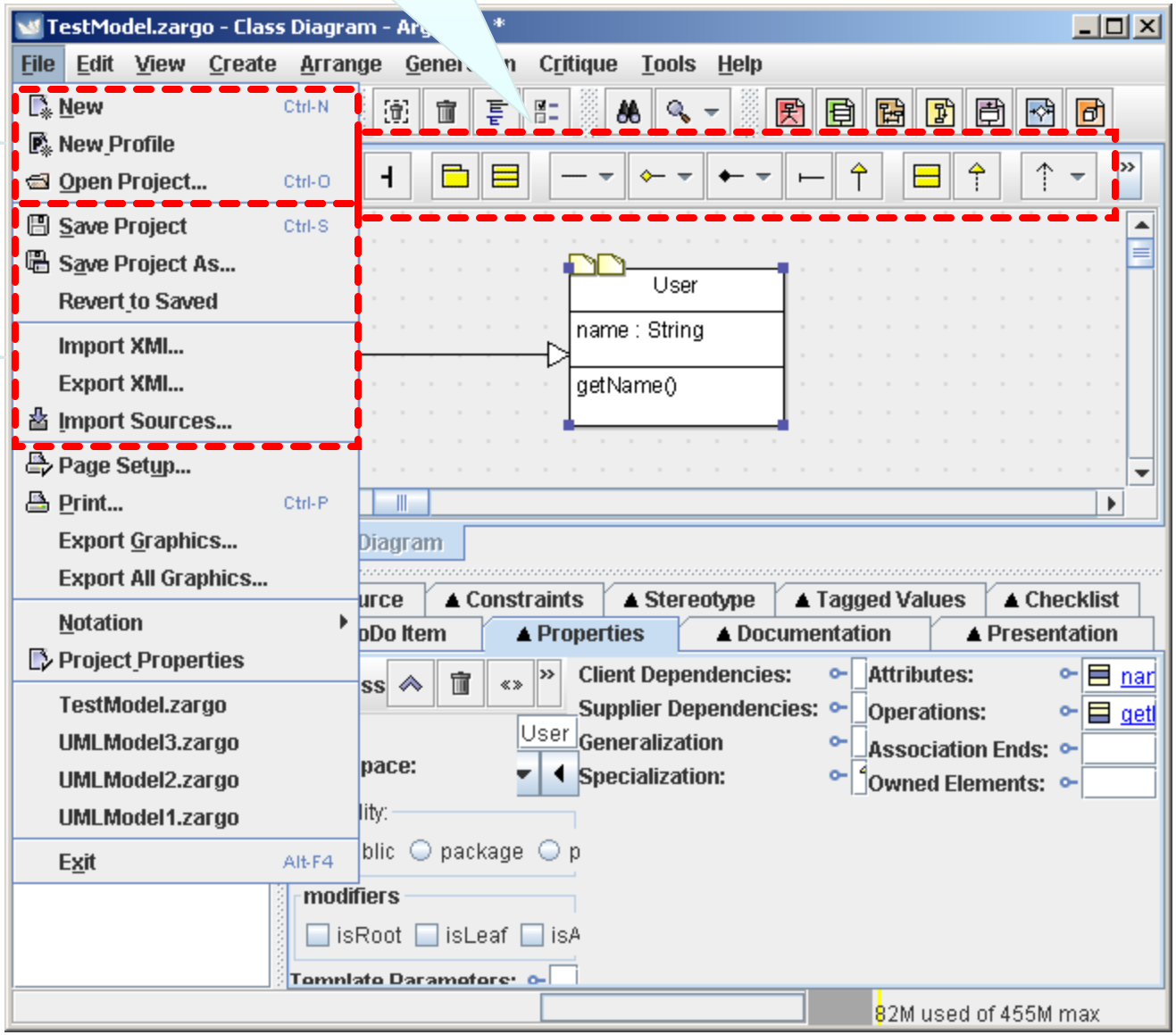
No test cases available…

Trace i

Trace 1

Trace 2

ArgoUML



JHotDraw

Size	200 KLOC	29 KLOC	28 KLOC
# of classes	1,889	289	245
Vocabulary	7,961	2,834	2,418
Historical	8 years	5 years	12 years
Execution Traces Coverage	66%	67%	86%

Choosing Software Systems

Choosing Participants

JHotDraw

Original

Developers

JHotDraw

Original Developers	6	3	3

JHotDraw

Original Developers

6	3	3

External Developers

64

JHotDraw

Original Developers

6	3	3

External Developers

64

5 with industrial experience
7 faculty
16 PhD students
33 MS students
3 undergraduates

JHotDraw

Original Developers

6	3	3

Original developers evaluated their own system

JHotDraw

External Developers

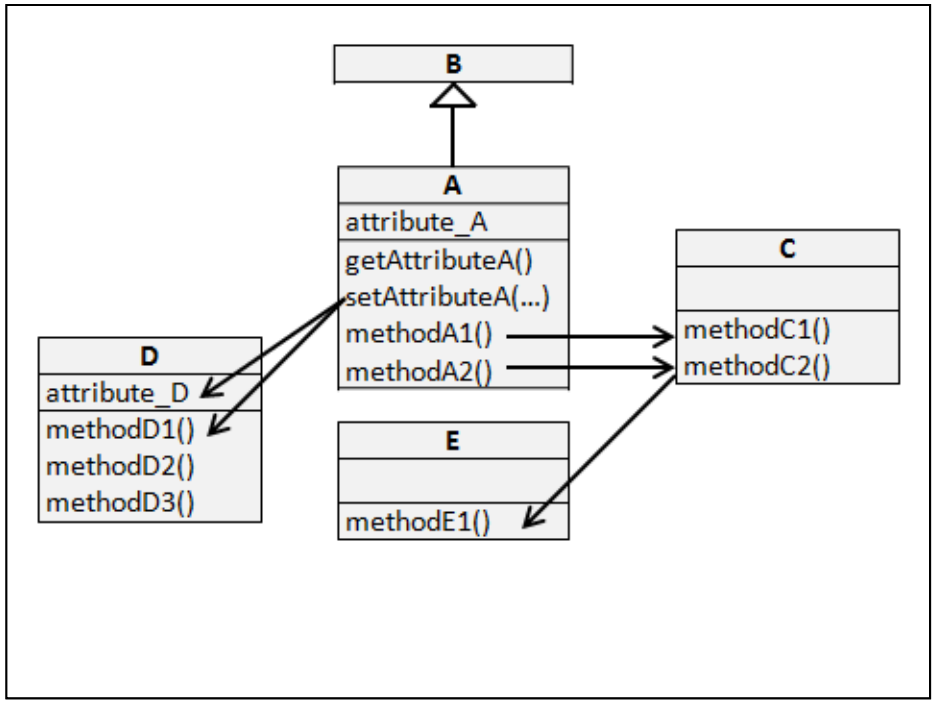
64

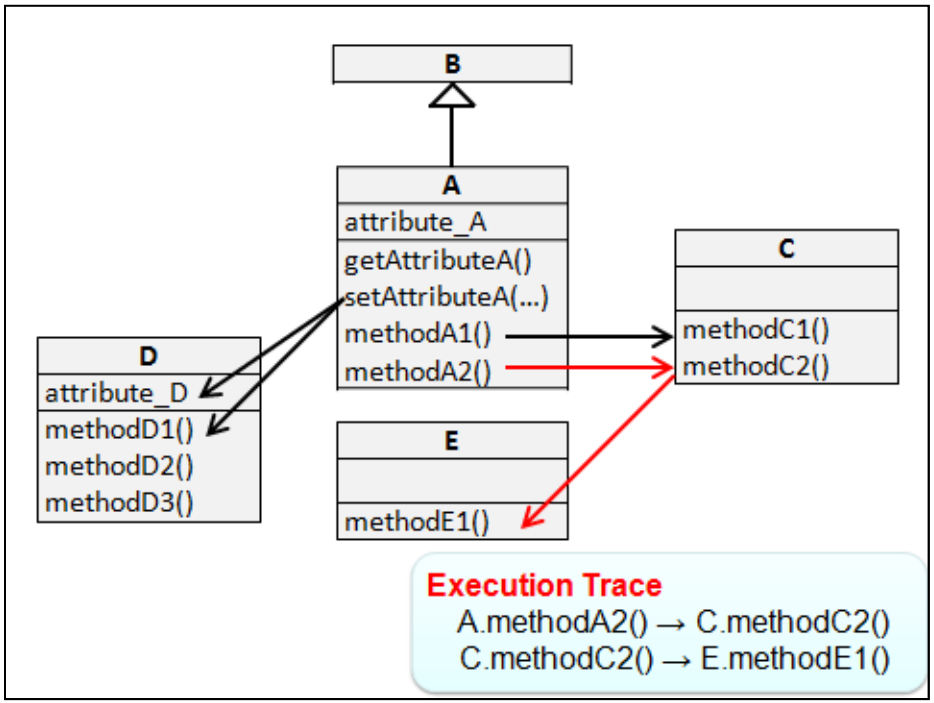
External developers evaluated all systems

Choosing Software Systems

Choosing Participants

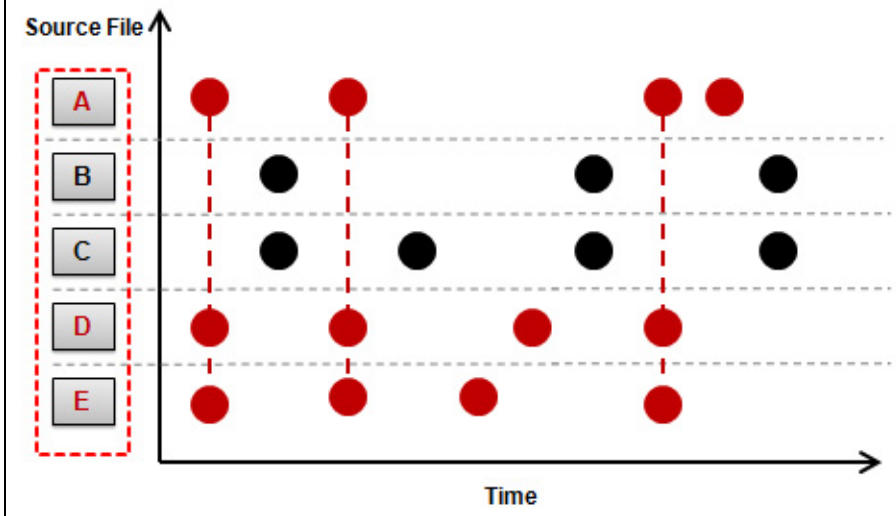
Coupling Metrics



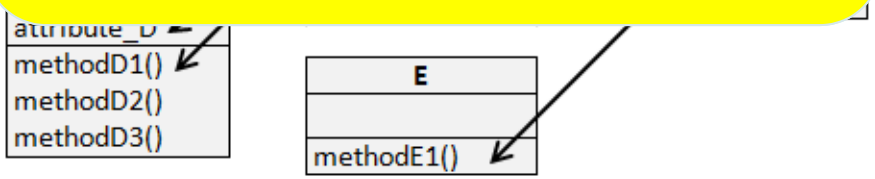



```java
/* Insert a new user in the system.
 *  @param pUser: the user to insert.*/
public void insert(User pUser){

  connect = DBConnection.getConnection();

  String sql = "INSERT INTO USER"
    + "(login,first_name,last_name,password"
    + ",email,cell,id_parent) " + "VALUES ("
    + pUser.getLogin() + ","
    + pUser.getFirstName() + ","
    + pUser.getLastName() + ","
    + pUser.getPassword() + ","
    + pUser.getEMail() + ","
    + pUser.getCell() + ","
    + pUser.getIdParent() + ")";

  executeOperation(connect, sql);
}
```

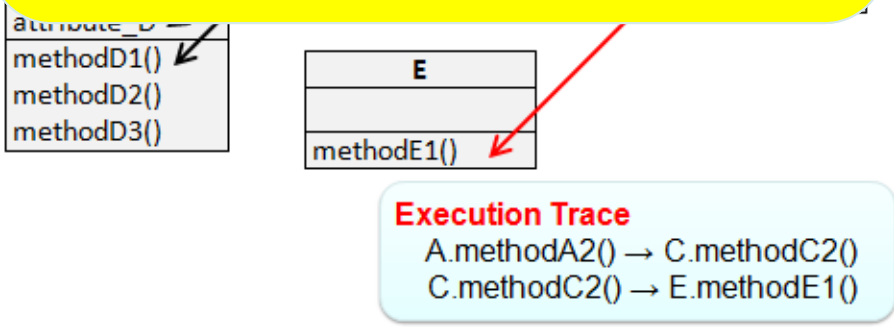
```java
/* Delete an user from the system.
 *  @param pUser: the user to delete.*/
public void delete(User pUser) {

    connect = DBConnection.getConnection();

    String sql = "DELETE FROM USER "
        + "WHERE id_user = "
        + pUser.getId();

    executeOperation(connect, sql);
}
```



Structural Coupling:
ICP: Information flow-based coupling [Lee et al.'95]



Dynamic Coupling:
IC_CD: Import Coupling Class Dynamic Message [Arisholm et al.'04]




```
/* Insert a new user in the system.
 *  @param pUser: the user to insert.*/
public void insert(User pUser){

  connect = DBConnection.getConnection();

  String sql = "INSERT INTO USER"
    + "(login,first_name,last_name,password"
    + ",email,cell,id_parent) " + "VALUES ("
    + pUser.getLogin() + ","
    + pUser.getFirstName() + ","
    + pUser.getLastName() + ","
    + pUser.getPassword() + ","
    + pUser.getEMail() + ","
    + pUser.getCell() + ",'
```

Semantic Coupling:
CCBC: Conceptual Coupling Between Classes [Poshyvanyk et al.'08]

Logical Coupling:
Association rule-based Change Coupling [Ying et al.'04] [Zimmerman et al.'04]

Choosing Software Systems

Choosing Participants

Coupling Metrics

Data Collection

Structural

Structural

Class1	Class2	Coupling
A	B	0.95
C	D	0.90
M	P	0.88
R	L	0.80
…	…	…
N	T	0.25
S	U	0.20
V	B	0.18

Structural

Class1	Class2	Coupling
A	B	0.95
C	D	0.90
M	P	0.88
R	L	0.80
...	...	...
N	T	0.25
S	U	0.20
V	B	0.18

***High Structural* Coupling:**
- Top 2 pairs

Structural

Class1	Class2	Coupling
A	B	0.95
C	D	0.90
M	P	0.88
R	L	0.80
…	…	…
N	T	0.25
S	U	0.20
V	B	0.18

***High Structural* Coupling:**
- Top 2 pairs

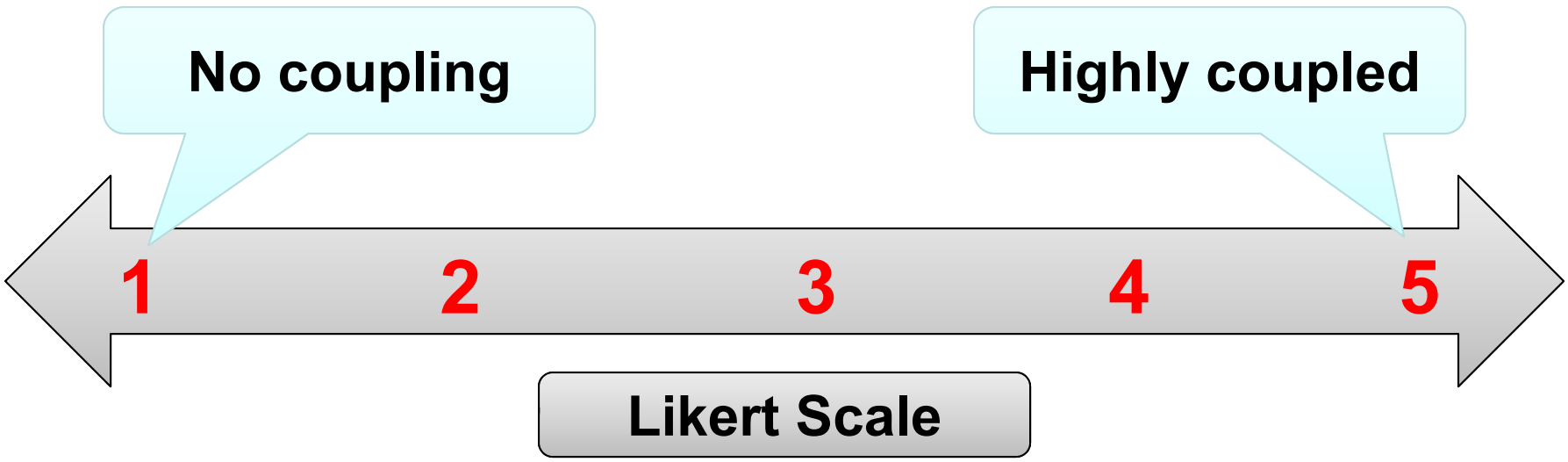
***Low Structural* Coupling:**
- Bottom 2 pairs

```
 * @(#)AWTCursor.java
package org.jhotdraw.standard;

import java.awt.Cursor;

/**
 * Default implementation of the {@link org.jhotdraw.
 * for AWT/Swing.
 *
 * <p>Created on: 08/05/2003.</p>
 *
 * @version $Revision: 1.3 $
 * @author <a href="mailto:ricardo_padilha@users.sour
 * Sangoi Padilha</a>
 * @see org.jhotdraw.framework.Cursor
 */
public class AWTCursor extends Cursor implements org.

    /**
     * Constructor for <code>AWTCursor</code>.
     * @param type
     * @see Cursor#Cursor(int)
     */
    public AWTCursor(int type) {
        super(type);
```

```
 * @(#)Locator.java

package org.jhotdraw.framework;

import org.jhotdraw.util.Storable;

/**
 * Locators can be used to locate a position on a figu
 *
 * <hr>
 * <b>Design Patterns</b><P>
 * <img src="images/red-ball-small.gif" width=6 height
 * <b><a href=../pattlets/sld034.htm>Strategy</a></b><
 * Locator encapsulates the strategy to locate a handl
 *
 * @version <$CURRENT_VERSION$>
 */
public interface Locator extends Storable, Serializabl

    /**
     * Locates a position on the passed figure.
     * @return a point on the figure.
     */
    public Point locate(Figure owner);
}
```

```java
 * @(#)AWTCursor.java
package org.jhotdraw.standard;

import java.awt.Cursor;

/**
 * Default implementation of the {@link org.jhotdraw.
 * for AWT/Swing.
 *
 * <p>Created on: 08/05/2003.</p>
 *
 * @version $Revision: 1.3 $
 * @author <a href="mailto:ricardo_padilha@users.sour
 * Sangoi Padilha</a>
 * @see org.jhotdraw.framework.Cursor
 */
public class AWTCursor extends Cursor implements org.

    /**
     * Constructor for <code>AWTCursor</code>.
     * @param type
     * @see Cursor#Cursor(int)
     */
    public AWTCursor(int type) {
        super(type);
```

```java
 * @(#)Locator.java

package org.jhotdraw.framework;

import org.jhotdraw.util.Storable;

/**
 * Locators can be used to locate a position on a figu
 *
 * <hr>
 * <b>Design Patterns</b><P>
 * <img src="images/red-ball-small.gif" width=6 height
 * <b><a href=../pattlets/sld034.htm>Strategy</a></b><
 * Locator encapsulates the strategy to locate a handl
 *
 * @version <$CURRENT_VERSION$>
 */
public interface Locator extends Storable, Serializabl

    /**
     * Locates a position on the passed figure.
     * @return a point on the figure.
     */
    public Point locate(Figure owner);
}
```



ArgoUML | jEdit | JHotDraw

Structural

ArgoUML

Class1	Class2	Coupling
A	B	0.95
C	D	0.90
M	P	0.88
R	L	0.80
...	...	...
N	T	0.25
S	U	0.20
V	B	0.18

jEdit

Class1	Class2	Coupling
A	B	0.95
C	D	0.90
M	P	0.88
R	L	0.80
...	...	...
N	T	0.25
S	U	0.20
V	B	0.18

JHotDraw

Class1	Class2	Coupling
A	B	0.95
C	D	0.90
M	P	0.88
R	L	0.80
...	...	...
N	T	0.25
S	U	0.20
V	B	0.18

Dynamic

ArgoUML

Class1	Class2	Coupling
A	B	0.95
C	D	0.90
M	P	0.88
R	L	0.80
...	...	...
N	T	0.25
S	U	0.20
V	B	0.18

jEdit

Class1	Class2	Coupling
A	B	0.95
C	D	0.90
M	P	0.88
R	L	0.80
...	...	...
N	T	0.25
S	U	0.20
V	B	0.18

JHotDraw

Class1	Class2	Coupling
A	B	0.95
C	D	0.90
M	P	0.88
R	L	0.80
...	...	...
N	T	0.25
S	U	0.20
V	B	0.18

Semantic

ArgoUML

Class1	Class2	Coupling
A	B	0.95
C	D	0.90
M	P	0.88
R	L	0.80
...	...	...
N	T	0.25
S	U	0.20
V	B	0.18

jEdit

Class1	Class2	Coupling
A	B	0.95
C	D	0.90
M	P	0.88
R	L	0.80
...	...	...
N	T	0.25
S	U	0.20
V	B	0.18

JHotDraw

Class1	Class2	Coupling
A	B	0.95
C	D	0.90
M	P	0.88
R	L	0.80
...	...	...
N	T	0.25
S	U	0.20
V	B	0.18

Logical

ArgoUML

Class1	Class2	Coupling
A	B	0.95
C	D	0.90
M	P	0.88
R	L	0.80
...	...	...
N	T	0.25
S	U	0.20
V	B	0.18

jEdit

Class1	Class2	Coupling
A	B	0.95
C	D	0.90
M	P	0.88
R	L	0.80
...	...	...
N	T	0.25
S	U	0.20
V	B	0.18

JHotDraw

Class1	Class2	Coupling
A	B	0.95
C	D	0.90
M	P	0.88
R	L	0.80
...	...	...
N	T	0.25
S	U	0.20
V	B	0.18

ArgoUML

jEdit

programmer's text editor

JHotDraw

Structural

Class1	Class2	Coupling
A	B	0.95
C	D	0.90
M	P	0.88
R	L	0.80
...	...	...
N	T	0.25
S	U	0.20
V	B	0.18

Class1	Class2	Coupling
A	B	0.95
C	D	0.90
M	P	0.88
R	L	0.80
...	...	...
N	T	0.25
S	U	0.20
V	B	0.18

Class1	Class2	Coupling
A	B	0.95
C	D	0.90
M	P	0.88
R	L	0.80
...	...	...
N	T	0.25
S	U	0.20
V	B	0.18

Dynamic

16 pairs of classes / system

...	...	...
N	T	0.25
S	U	0.20
V	B	0.18

...	...	...
N	T	0.25
S	U	0.20
V	B	0.18

...	...	...
N	T	0.25
S	U	0.20
V	B	0.18

Semantic

Class1	Class2	Coupling
A	B	0.95
C	D	0.90
M	P	0.88
R	L	0.80
...	...	...
N	T	0.25
S	U	0.20
V	B	0.18

Class1	Class2	Coupling
A	B	0.95
C	D	0.90
M	P	0.88
R	L	0.80
...	...	...
N	T	0.25
S	U	0.20
V	B	0.18

Class1	Class2	Coupling
A	B	0.95
C	D	0.90
M	P	0.88
R	L	0.80
...	...	...
N	T	0.25
S	U	0.20
V	B	0.18

Logical

Class1	Class2	Coupling
A	B	0.95
C	D	0.90
M	P	0.88
R	L	0.80
...	...	...
N	T	0.25
S	U	0.20
V	B	0.18

Class1	Class2	Coupling
A	B	0.95
C	D	0.90
M	P	0.88
R	L	0.80
...	...	...
N	T	0.25
S	U	0.20
V	B	0.18

Class1	Class2	Coupling
A	B	0.95
C	D	0.90
M	P	0.88
R	L	0.80
...	...	...
N	T	0.25
S	U	0.20
V	B	0.18

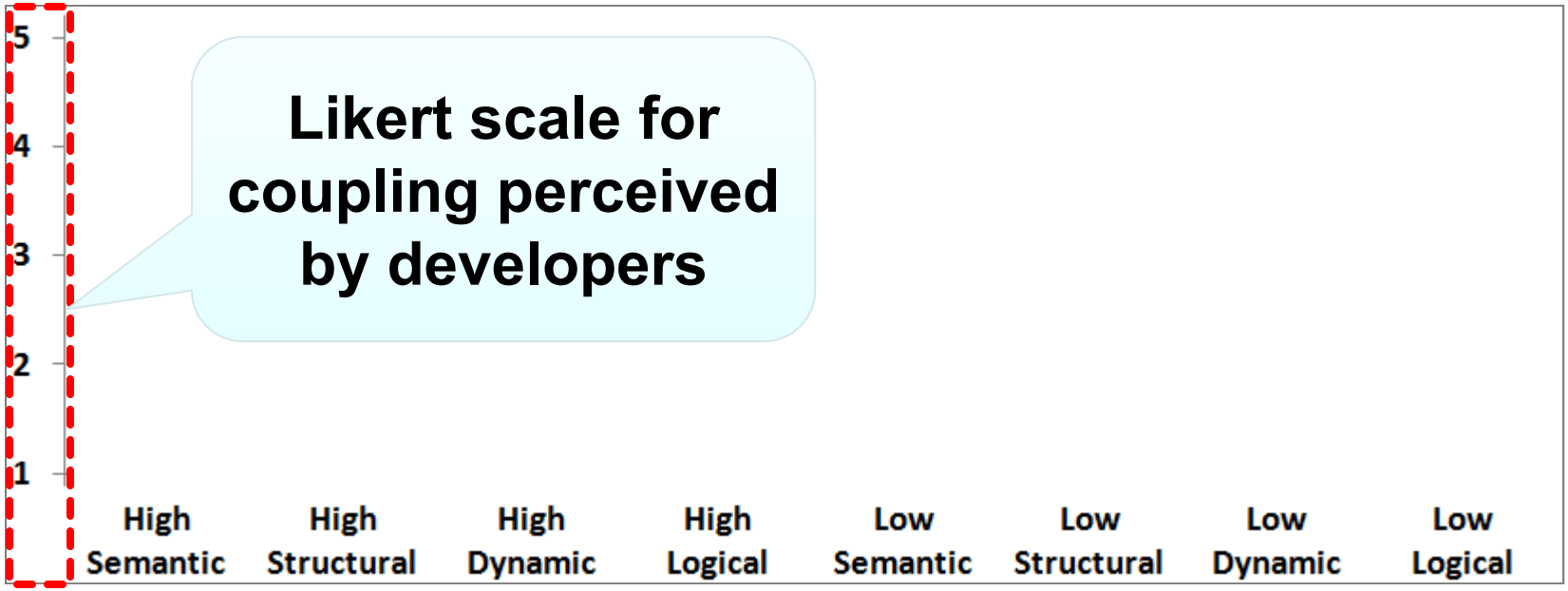
Choosing Software Systems

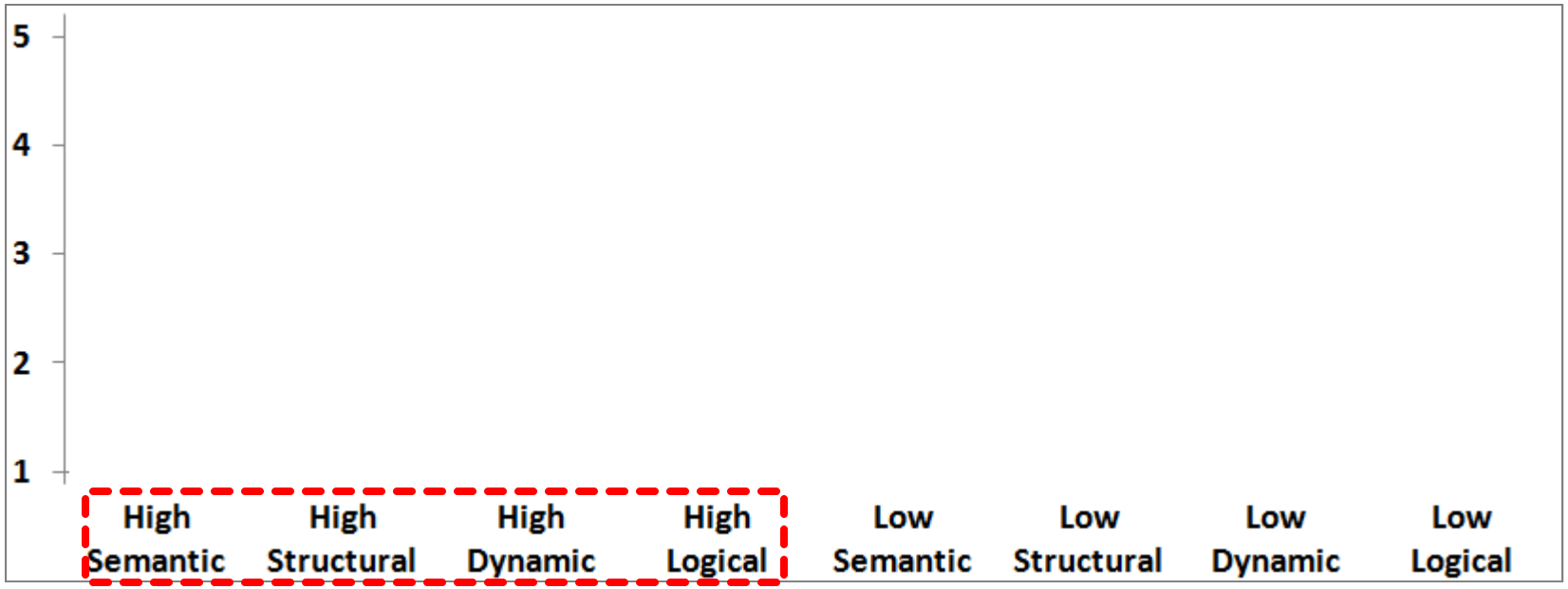
Choosing Participants

Coupling Metrics

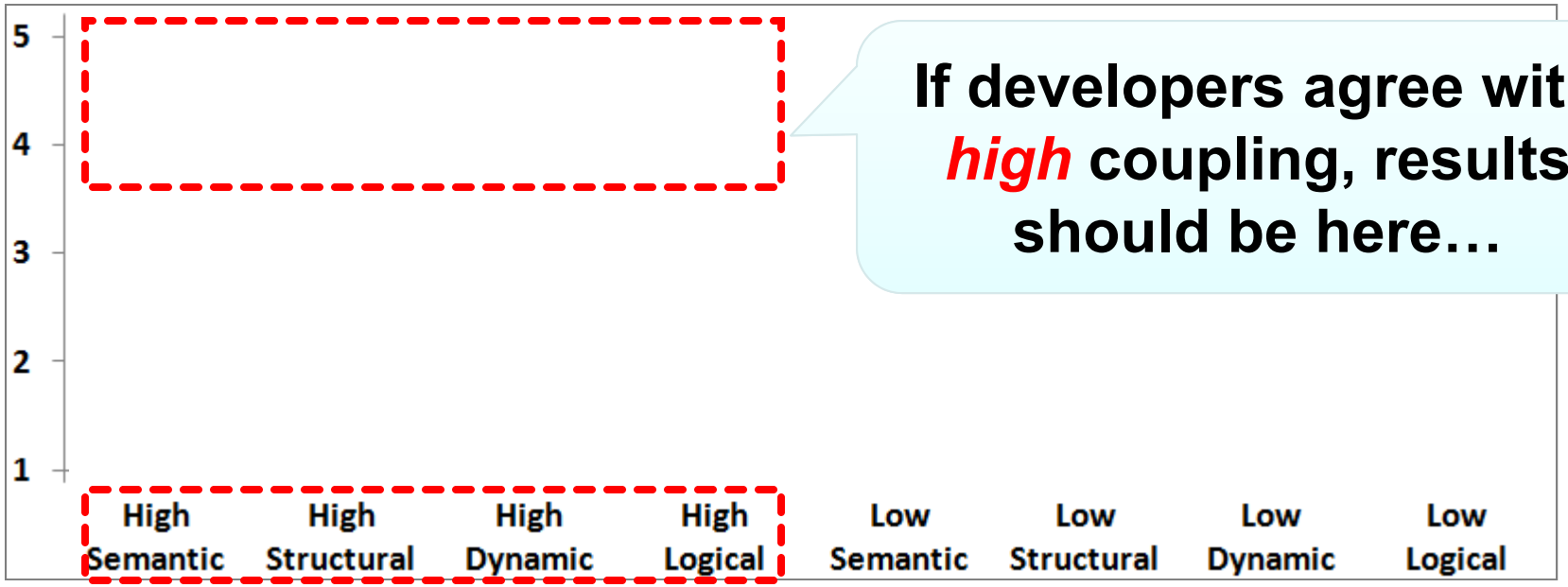
Data Collection

Results


5
4
3
2
1
Likert scale for coupling perceived by developers
High Semantic
High Structural
High Dynamic
High Logical
Low Semantic
Low Structural
Low Dynamic
Low Logical

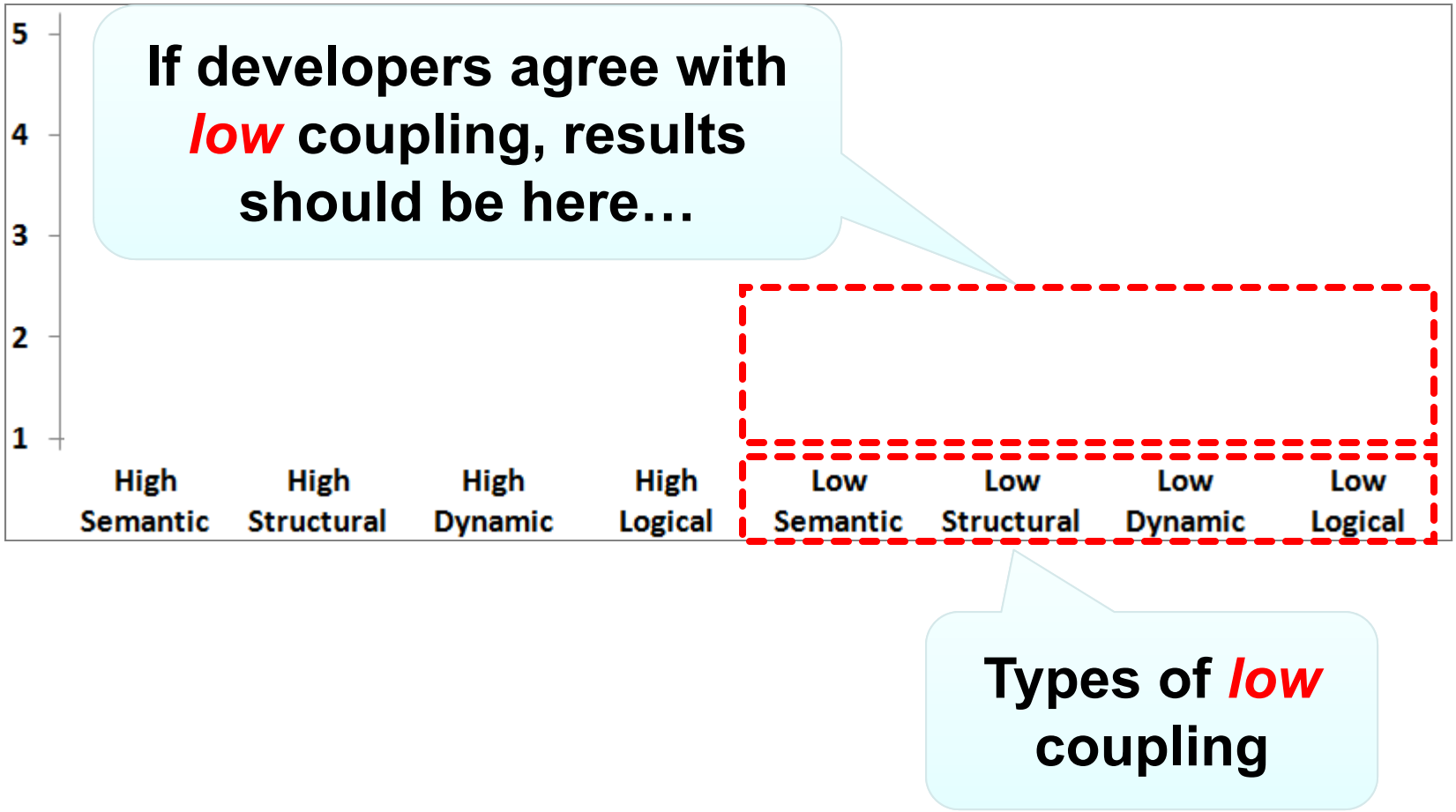


Types of *high* coupling

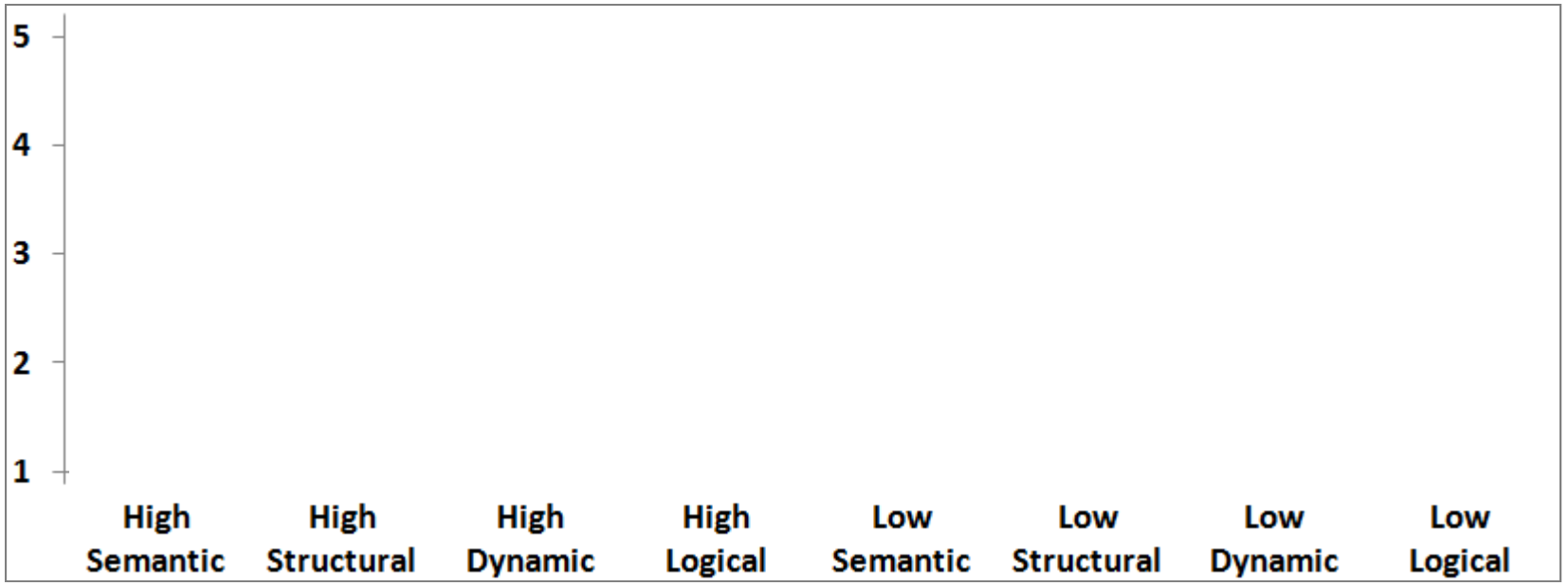


If developers agree with *high* coupling, results should be here…

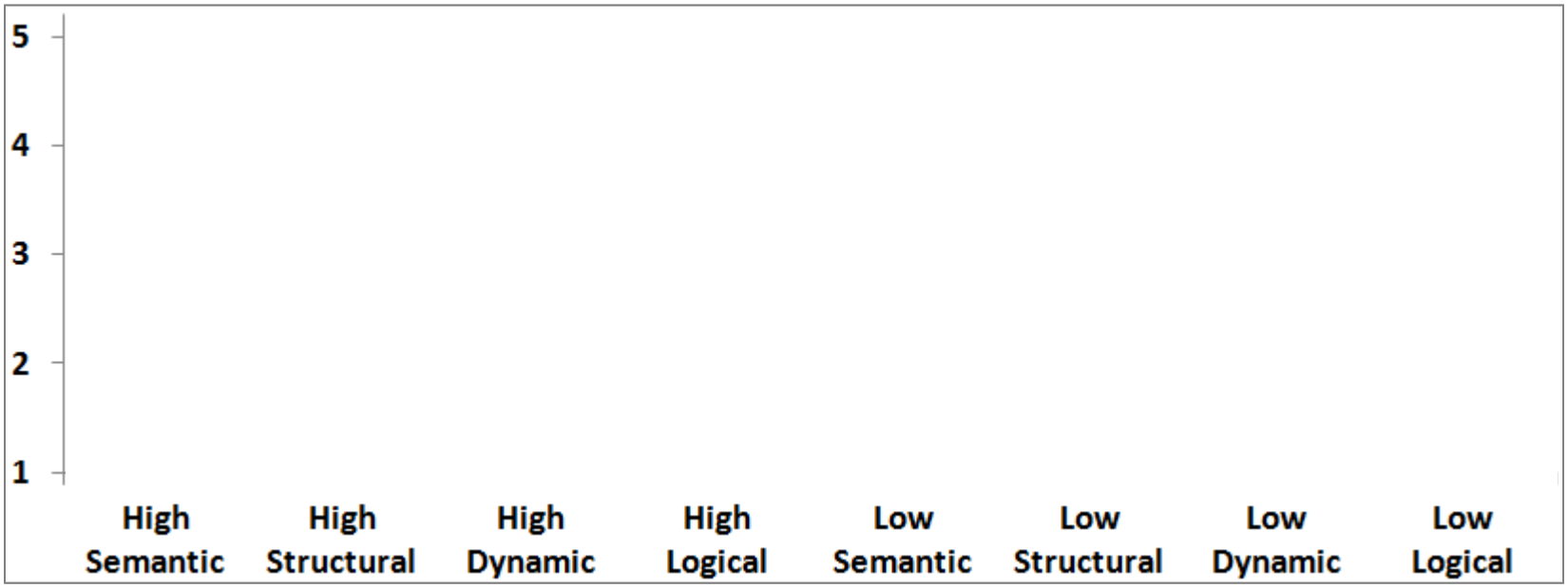
Types of *high* coupling


5
4
3
2
1
If developers agree with *low* coupling, results should be here…
High Semantic
High Structural
High Dynamic
High Logical
Low Semantic
Low Structural
Low Dynamic
Low Logical
Types of *low* coupling

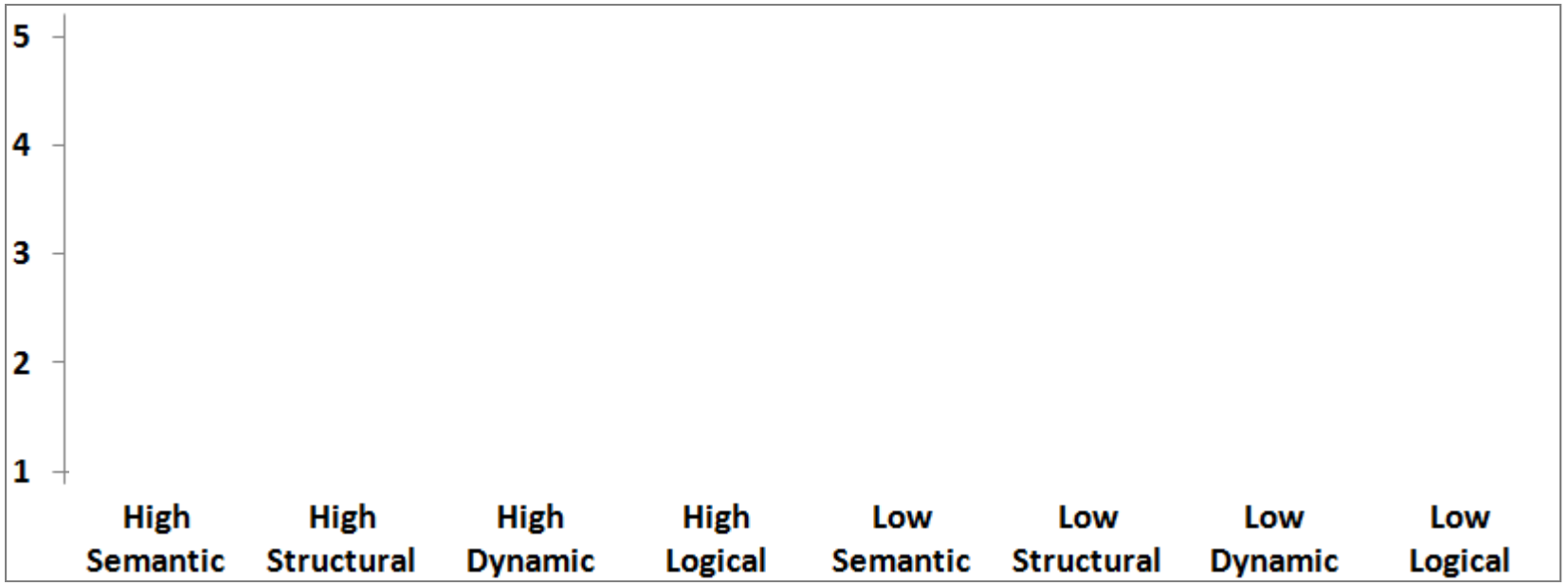
Original Dev.



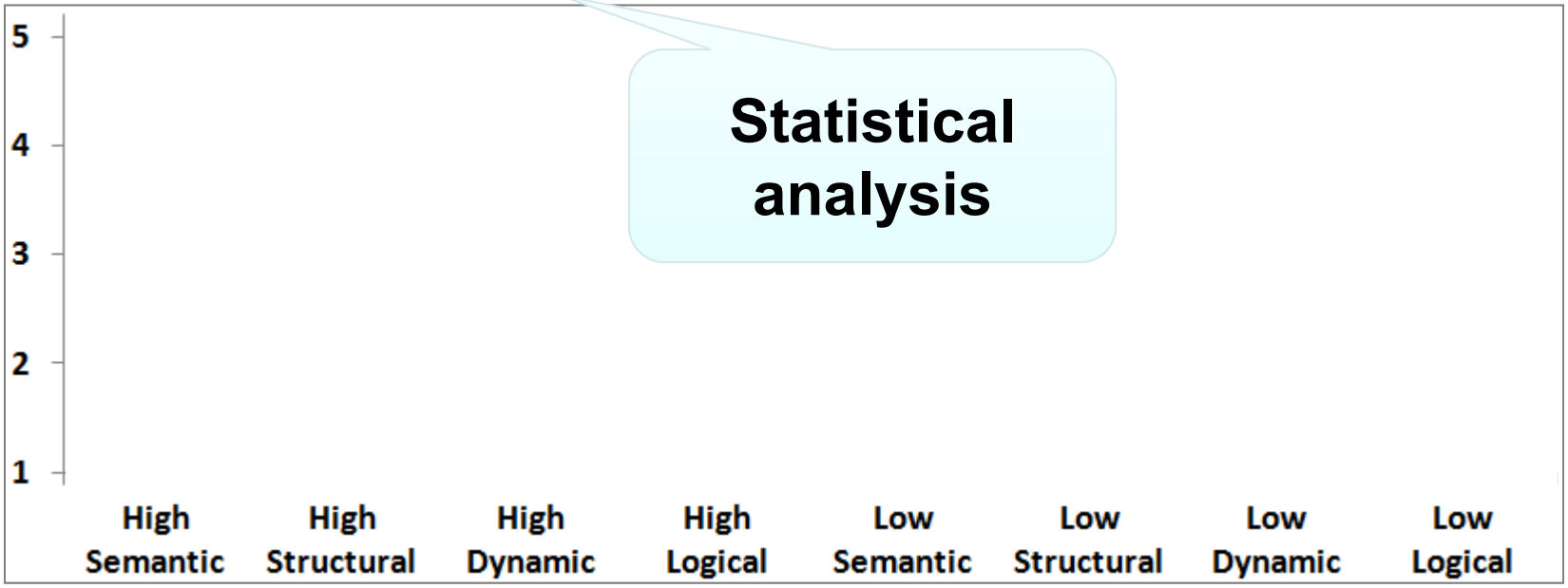
External Dev.



Original Dev.



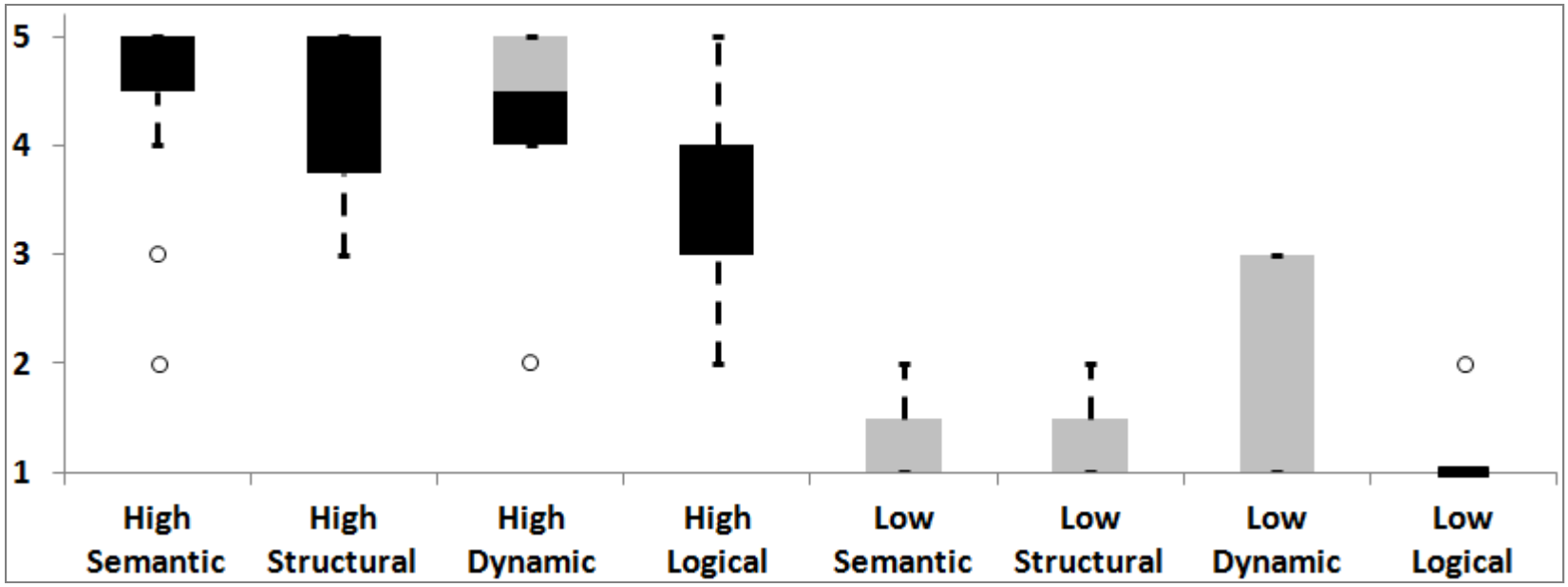
External Dev.



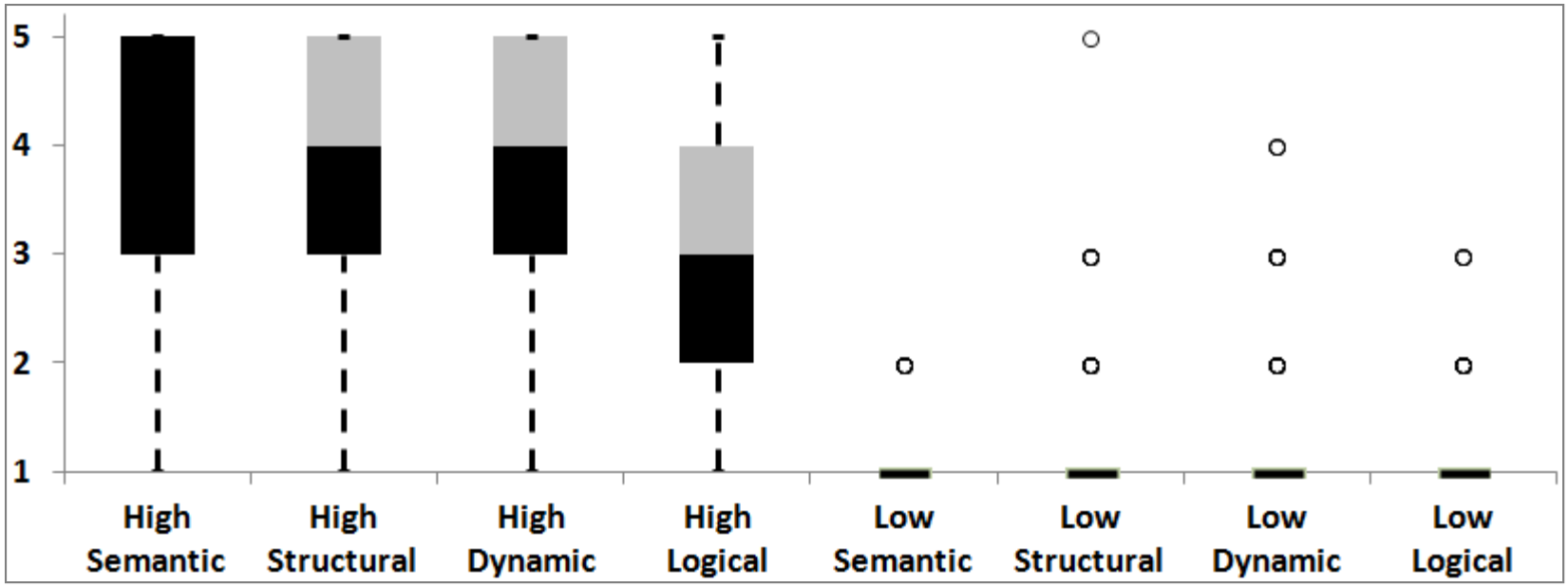
Statistical analysis

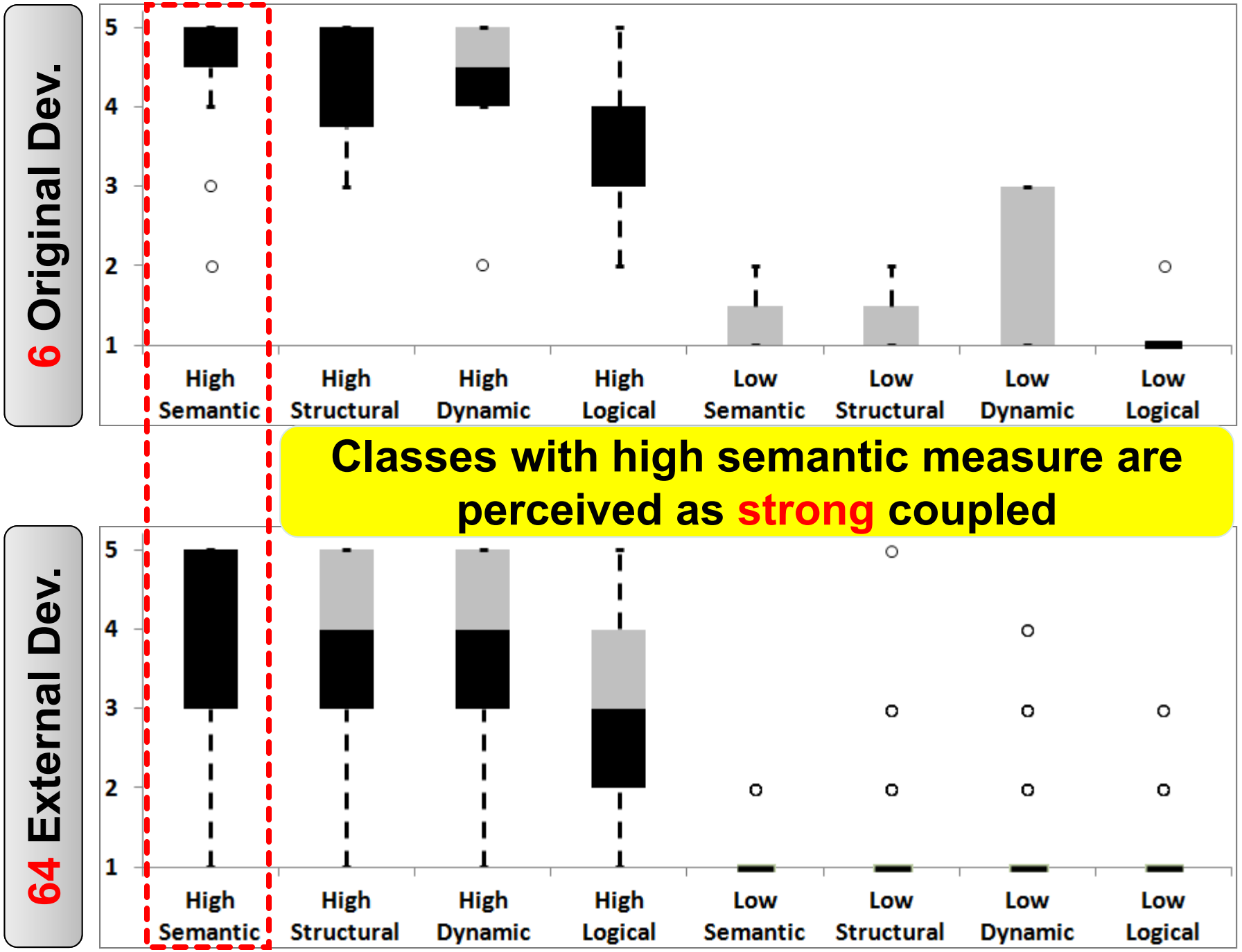

ArgoUML

6 Original Dev.

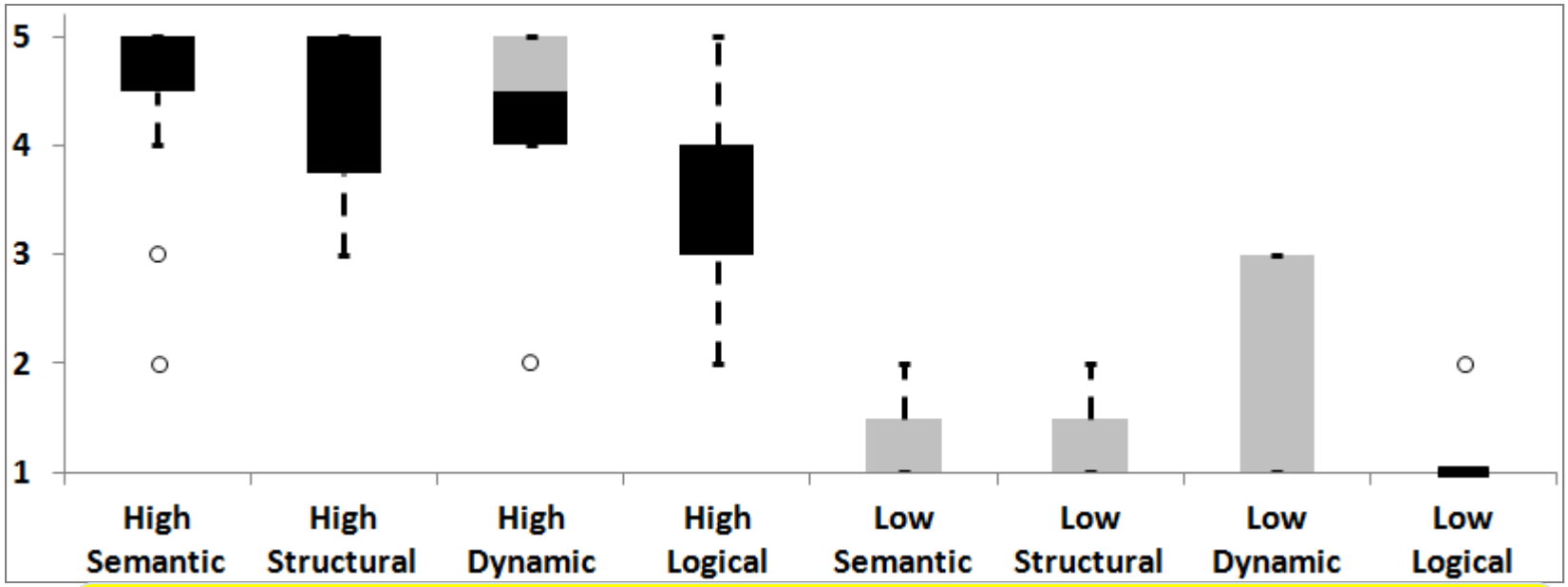


64 External Dev.



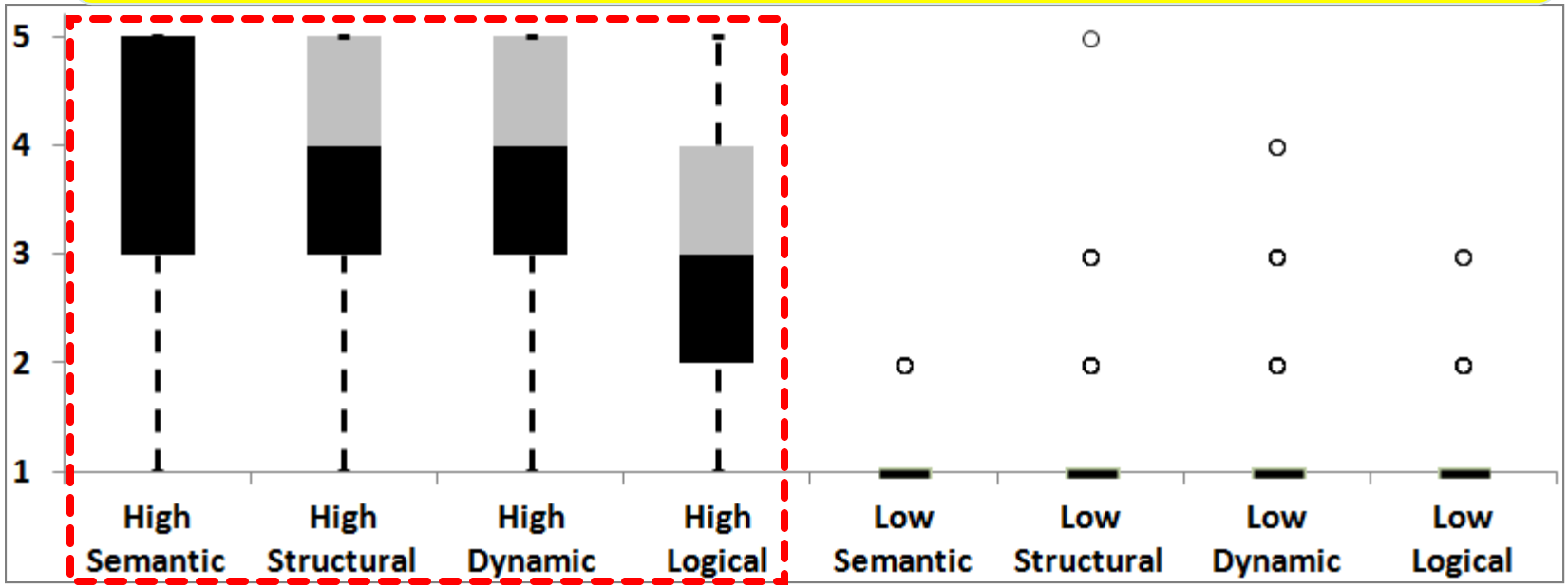

6 Original Dev.
64 External Dev.
5
4
3
2
1
High Semantic
High Structural
High Dynamic
High Logical
Low Semantic
Low Structural
Low Dynamic
Low Logical
Classes with high semantic measure are perceived as strong coupled

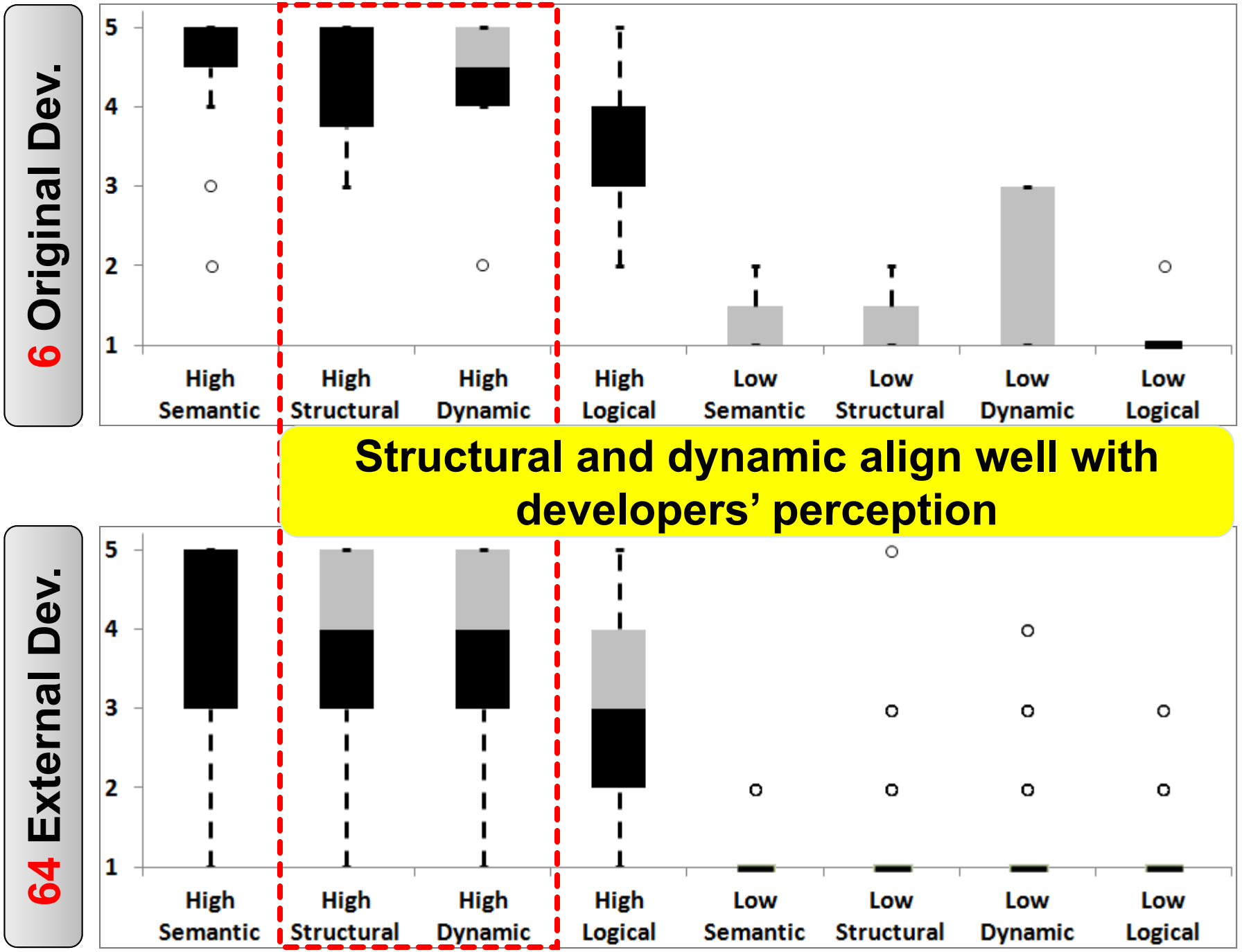
6 Original Dev.

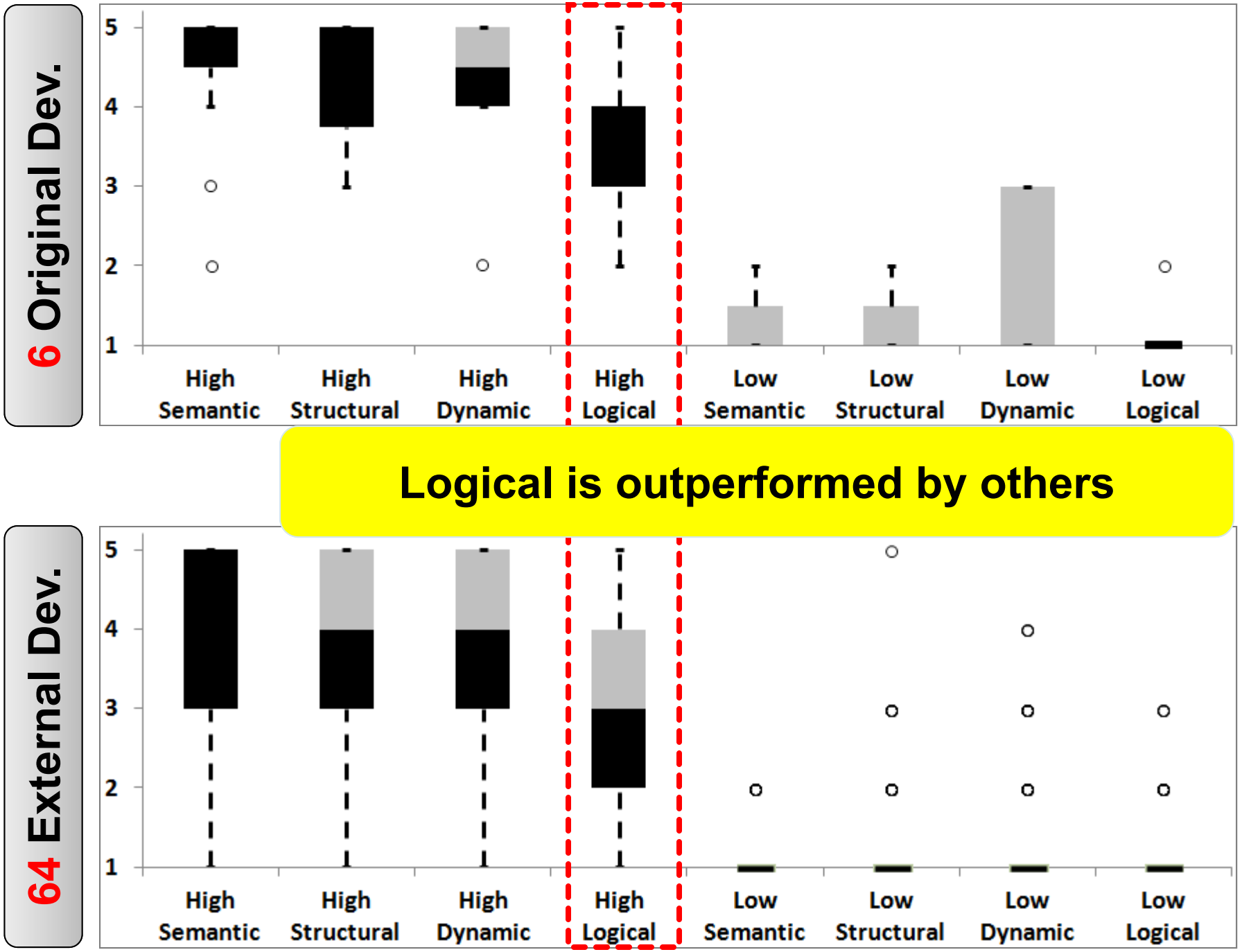


Semantic outperforms others *[p-value<0.05]*

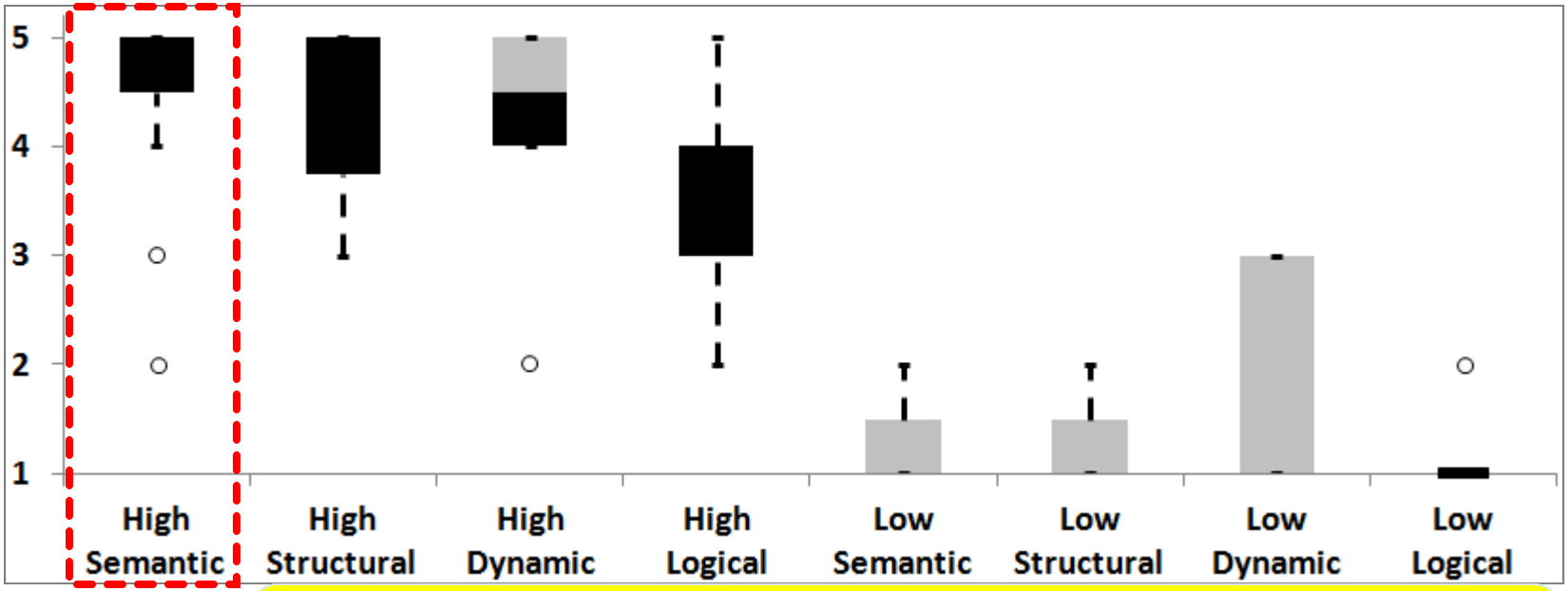
64 External Dev.




6 Original Dev.
64 External Dev.
5
4
3
2
1
High Semantic
High Structural
High Dynamic
High Logical
Low Semantic
Low Structural
Low Dynamic
Low Logical
Structural and dynamic align well with developers' perception

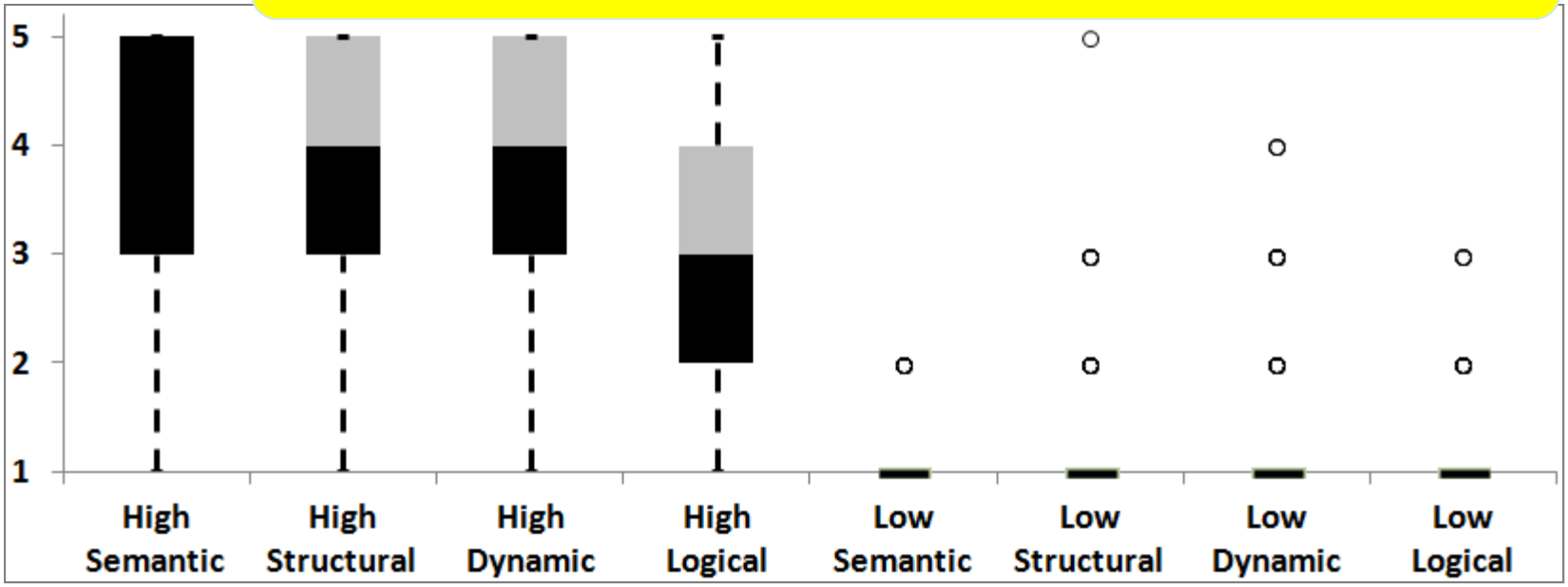

6 Original Dev.
64 External Dev.
5
4
3
2
1
High Semantic
High Structural
High Dynamic
High Logical
Low Semantic
Low Structural
Low Dynamic
Low Logical
Logical is outperformed by others

6 Original Dev.



Qualitative example…

64 External Dev.



ActionVisibilityPrivate

```
@UmlModelMutator
class ActionVisibilityPrivate extends AbstractActionRadioMenuItem

    /**
     * Serial version generated for rev 1.5
     */
    private static final long serialVersionUID = -1342216726253371

    /**
     * The constructor.
     *
     * @param o the target
     */
    public ActionVisibilityPrivate(Object o) {
        super("checkbox.visibility.private-uc", false);
        putValue("SELECTED", Boolean.valueOf(
            Model.getVisibilityKind().getPrivate()
                .equals(valueOfTarget(o))));
    }

    /*
     * @see org.argouml.uml.diagram.ui.AbstractActionRadioMenuItem
     */
    void toggleValueOfTarget(Object t) {
        Model.getCoreHelper().setVisibility(t,
```

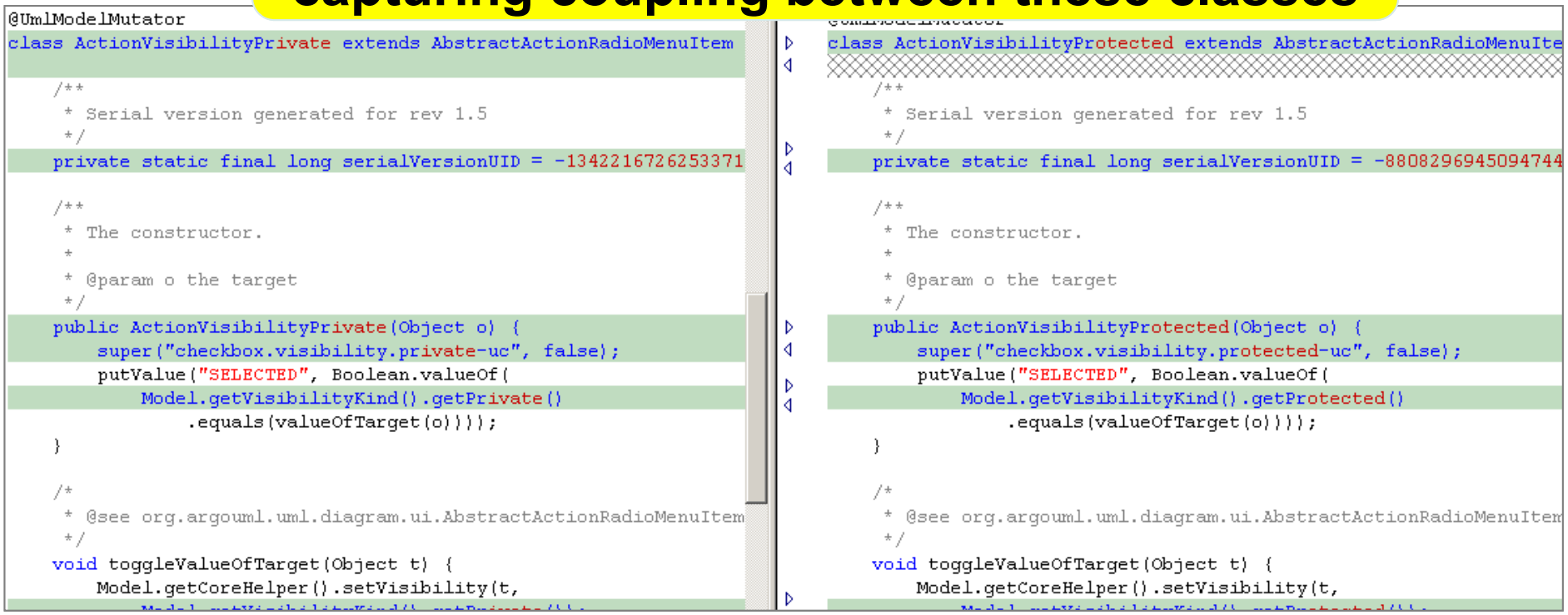
ActionVisibilityProtected

```
@UmlModelMutator
class ActionVisibilityProtected extends AbstractActionRadioMenuIte

    /**
     * Serial version generated for rev 1.5
     */
    private static final long serialVersionUID = -8808296945094744

    /**
     * The constructor.
     *
     * @param o the target
     */
    public ActionVisibilityProtected(Object o) {
        super("checkbox.visibility.protected-uc", false);
        putValue("SELECTED", Boolean.valueOf(
            Model.getVisibilityKind().getProtected()
                .equals(valueOfTarget(o))));
    }

    /*
     * @see org.argouml.uml.diagram.ui.AbstractActionRadioMenuItem
     */
    void toggleValueOfTarget(Object t) {
        Model.getCoreHelper().setVisibility(t,
```

Original developer:

"these classes have a very high coupling even if there is no cooperation between them"

Semantic measure is the only one capturing coupling between these classes

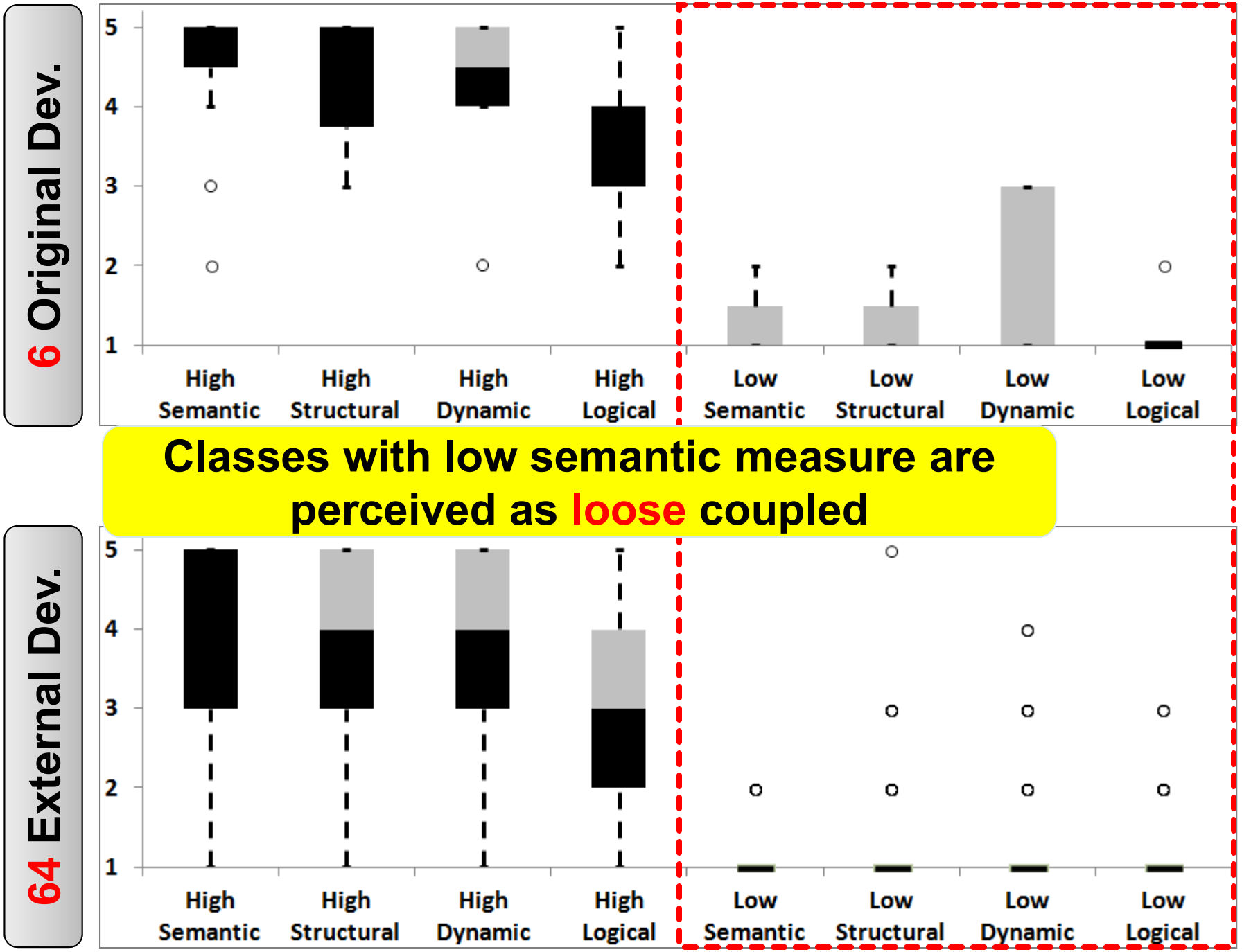


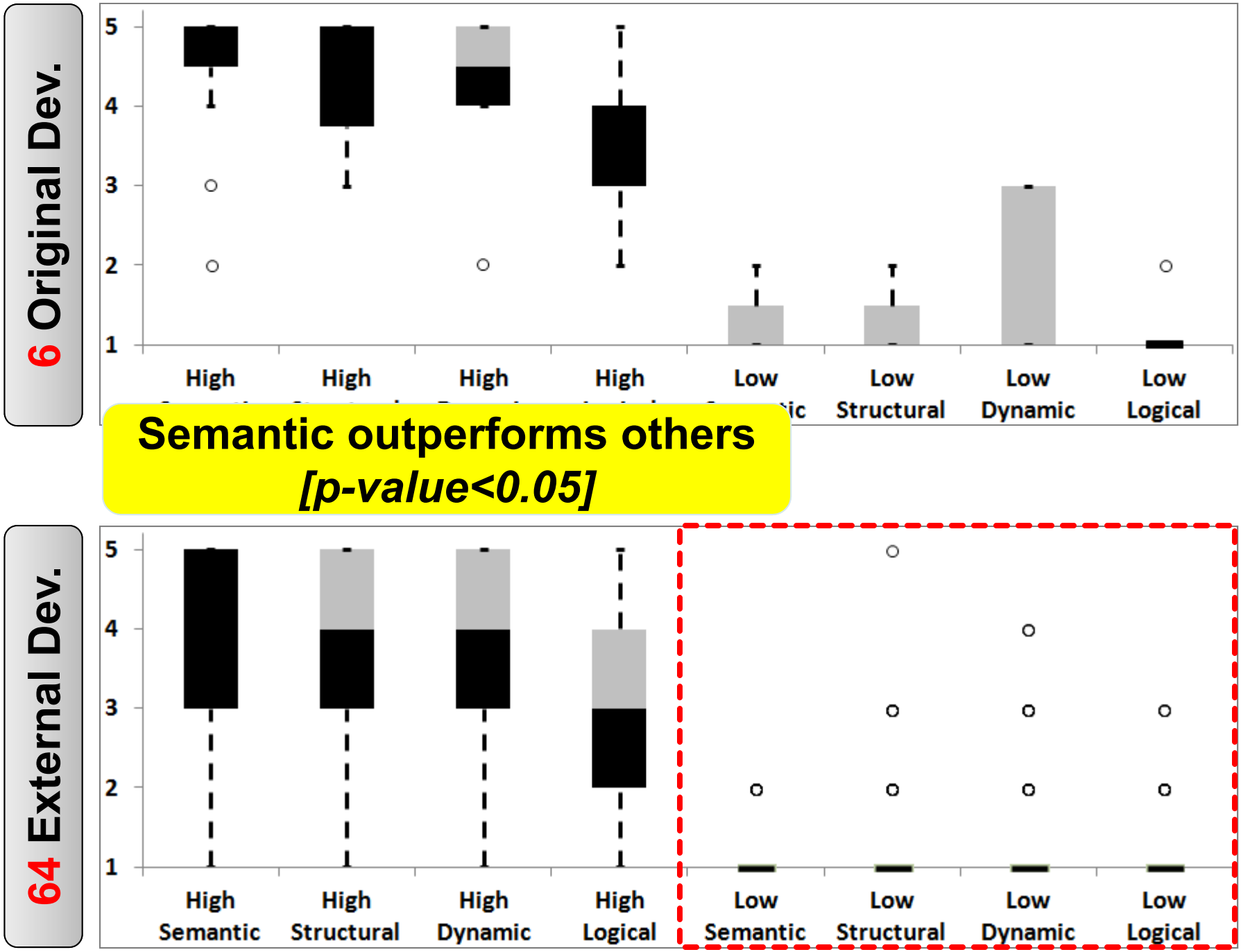
```
@UmlModelMutator
class ActionVisibilityPrivate extends AbstractActionRadioMenuItem

    /**
     * Serial version generated for rev 1.5
     */
    private static final long serialVersionUID = -1342216726253371

    /**
     * The constructor.
     *
     * @param o the target
     */
    public ActionVisibilityPrivate(Object o) {
        super("checkbox.visibility.private-uc", false);
        putValue("SELECTED", Boolean.valueOf(
            Model.getVisibilityKind().getPrivate()
                .equals(valueOfTarget(o))));
    }

    /*
     * @see org.argouml.uml.diagram.ui.AbstractActionRadioMenuItem
     */
    void toggleValueOfTarget(Object t) {
        Model.getCoreHelper().setVisibility(t,
```

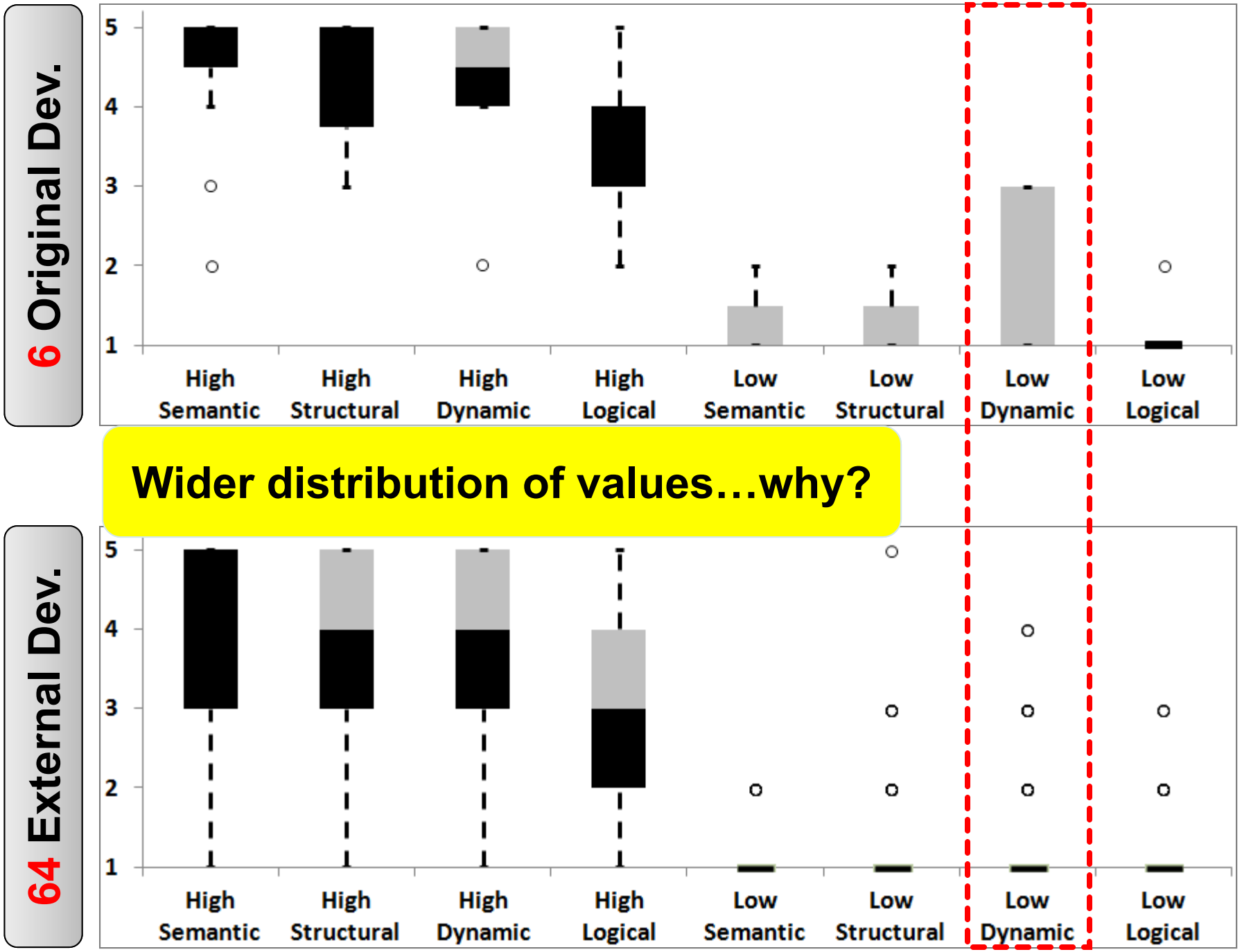
```
class ActionVisibilityProtected extends AbstractActionRadioMenuIte

    /**
     * Serial version generated for rev 1.5
     */
    private static final long serialVersionUID = -8808296945094744

    /**
     * The constructor.
     *
     * @param o the target
     */
    public ActionVisibilityProtected(Object o) {
        super("checkbox.visibility.protected-uc", false);
        putValue("SELECTED", Boolean.valueOf(
            Model.getVisibilityKind().getProtected()
                .equals(valueOfTarget(o))));
    }

    /*
     * @see org.argouml.uml.diagram.ui.AbstractActionRadioMenuItem
     */
    void toggleValueOfTarget(Object t) {
        Model.getCoreHelper().setVisibility(t,
```

Original developer:

"these classes have a very high coupling even if there is no cooperation between them"


6 Original Dev.
64 External Dev.
5
4
3
2
1
High Semantic
High Structural
High Dynamic
High Logical
Low Semantic
Low Structural
Low Dynamic
Low Logical
Classes with low semantic measure are perceived as loose coupled


6 Original Dev.
5
4
3
2
1
High
High
High
High
Low
Low
Low
Low
Structural
Dynamic
Logical
Semantic outperforms others
[p-value<0.05]
64 External Dev.
5
4
3
2
1
High
Semantic
High
Structural
High
Dynamic
High
Logical
Low
Semantic
Low
Structural
Low
Dynamic
Low
Logical


6 Original Dev.
64 External Dev.
Wider distribution of values…why?
5
4
3
2
1
High Semantic
High Structural
High Dynamic
High Logical
Low Semantic
Low Structural
Low Dynamic
Low Logical

AboutBox

```
 */
public class AboutBox extends ArgoDialog {


    /**
     * Insets in pixels.
     */
    private static final int INSET_PX = 3;

    private static Map<String, Component> extraTabs =
        new HashMap<String, Component>();


    private JTabbedPane ta
```

FigCompartmentBox

```
 */
public abstract class FigCompartmentBox extends
FigNodeModelElement {

    private static final Logger LOG = Logger.getLogger(
            FigCompartmentBox.class);


    /**
     * Default bounds for a compartment.
     */
    protected static final Rectangle DEFAULT_COMPARTMENT_BOUNDS
        = new Rectangle(
            X0, Y0 + 20 /* 20 = height of name fig ?*/,
            WIDTH, ROWHEIGHT + 2 /* 2*LINE_WIDTH?  or extra
padding? */ );

                                                    artments =
                                                    ();
```

Classes with low dynamic coupling, but ranked as strong coupled.

AboutBox

```
 */
public class AboutBox extends ArgoDialog {

    /**
     * Insets in pixels.
     */
    private static final int INSET_PX = 3;

    private static Map<String, Component> extraTabs =
        new HashMap<String, Component>();

    private JTabbedPane ta
```

FigCompartmentBox

```
 */
public abstract class FigCompartmentBox extends
FigNodeModelElement {

    private static final Logger LOG = Logger.getLogger(
            FigCompartmentBox.class);

    /**
     * Default bounds for a compartment.
     */
    protected static final Rectangle DEFAULT_COMPARTMENT_BOUNDS
        = new Rectangle(
            X0, Y0 + 20 /* 20 = height of name fig ?*/,
            WIDTH, ROWHEIGHT + 2 /* 2*LINE_WIDTH?  or extra
padding? */ );
```

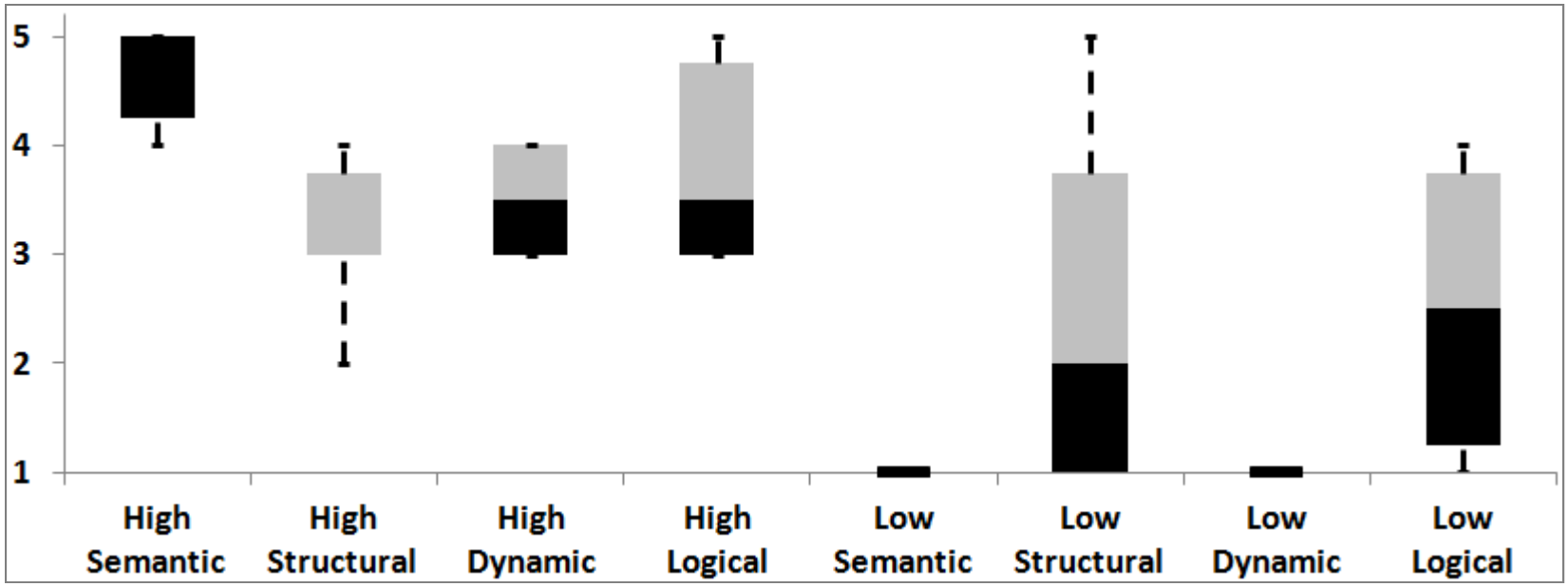
Classes with low dynamic coupling, but ranked as strong coupled.

Original developer:

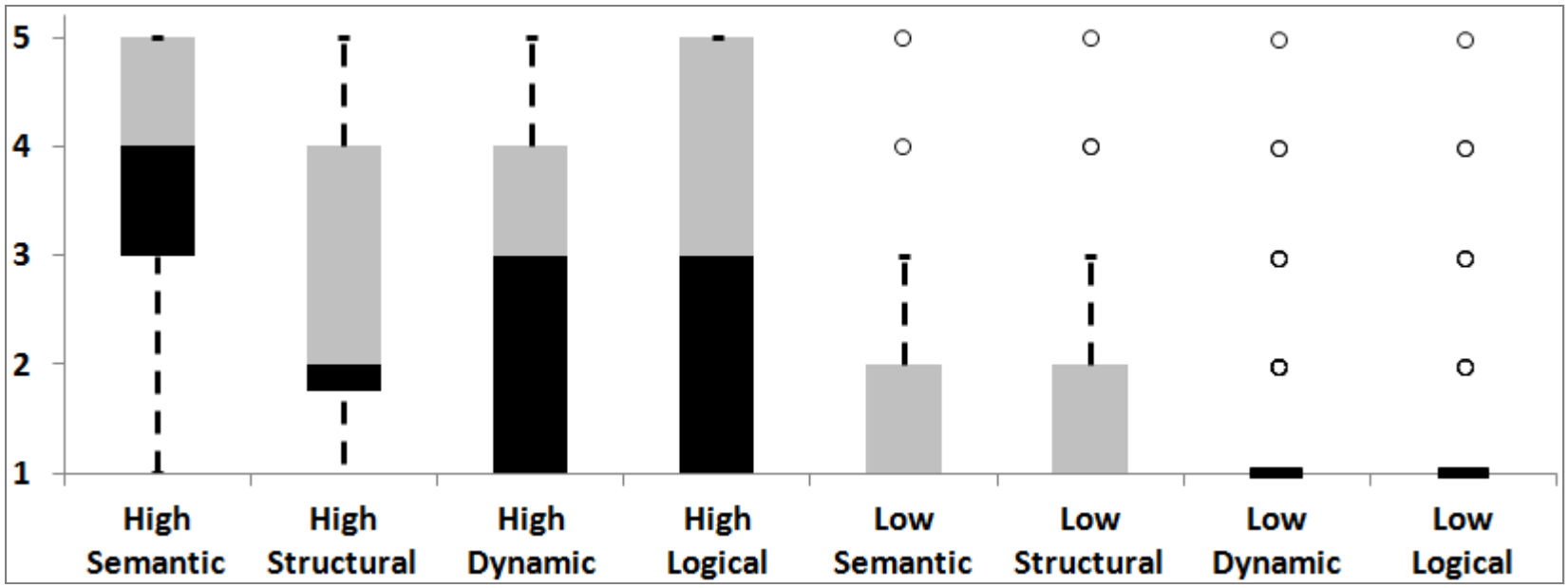
"even if indirectly, the two classes exhibit some form of coupling since they implement similar jobs and changes to the GUI are likely to involve both classes"

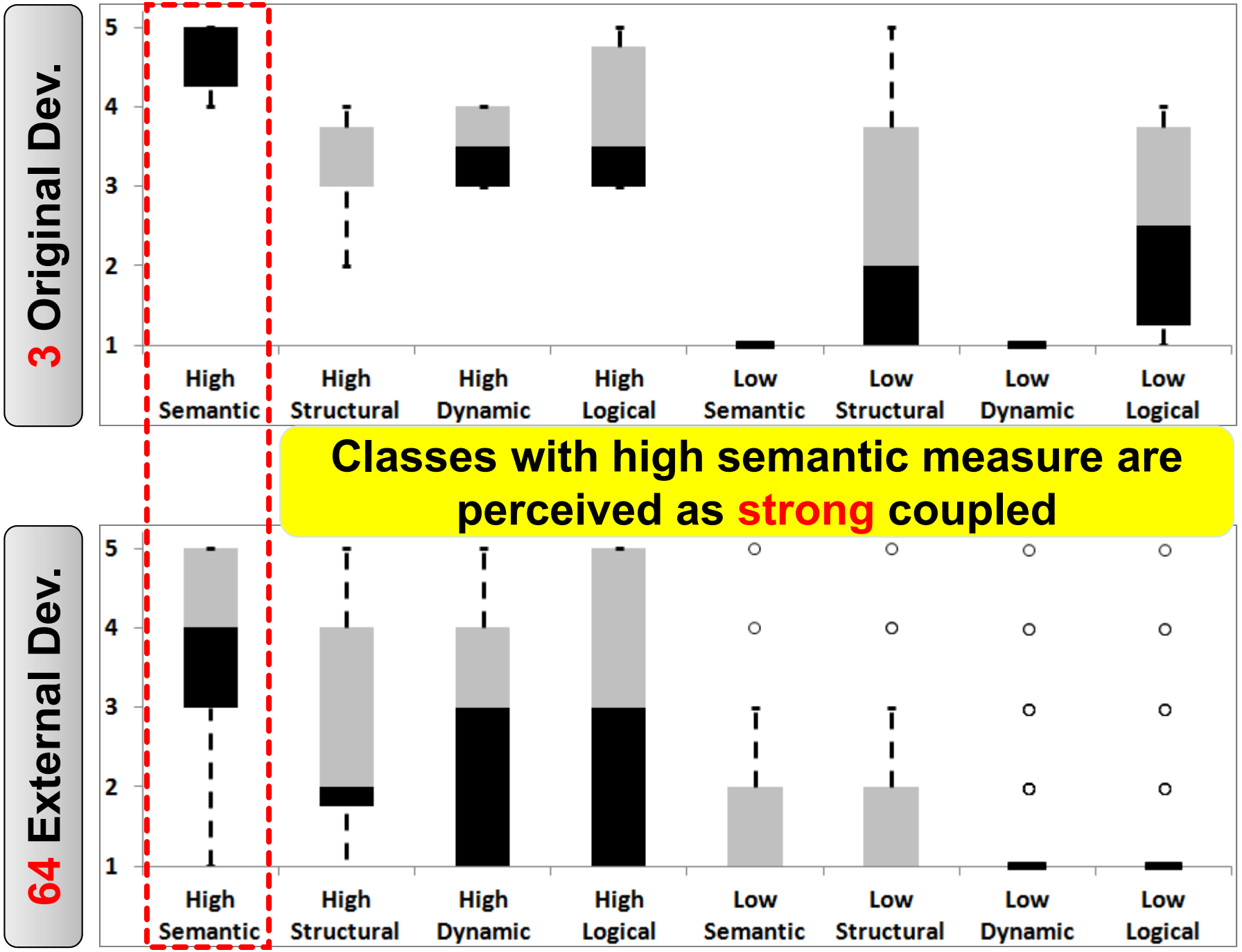
JHotDraw

3 Original Dev.

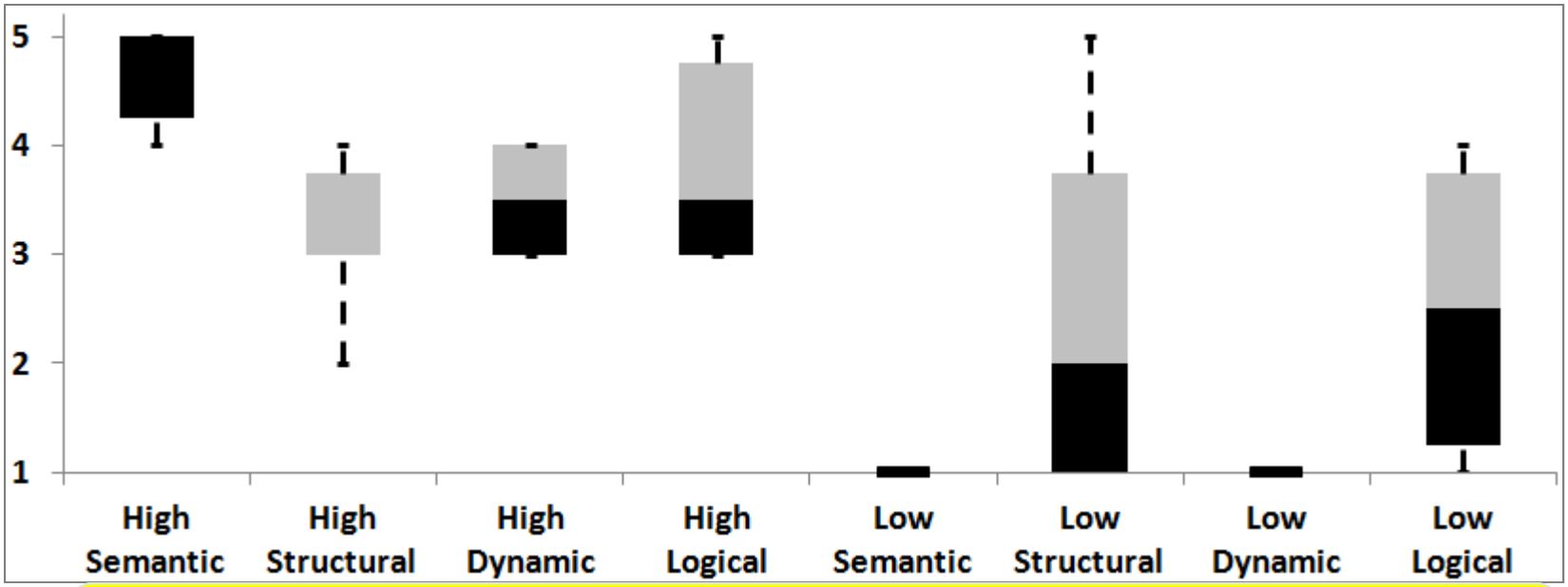


64 External Dev.



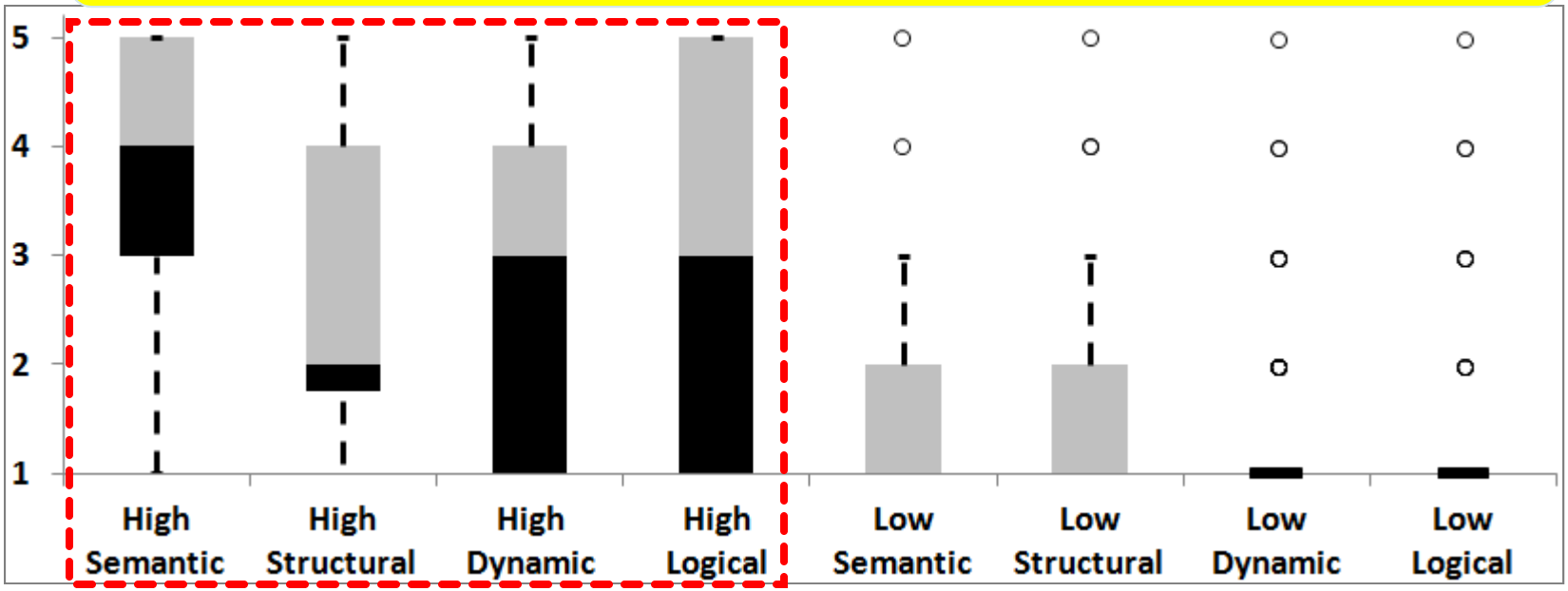

3 Original Dev.
64 External Dev.
5
4
3
2
1
High Semantic
High Structural
High Dynamic
High Logical
Low Semantic
Low Structural
Low Dynamic
Low Logical
Classes with high semantic measure are perceived as strong coupled

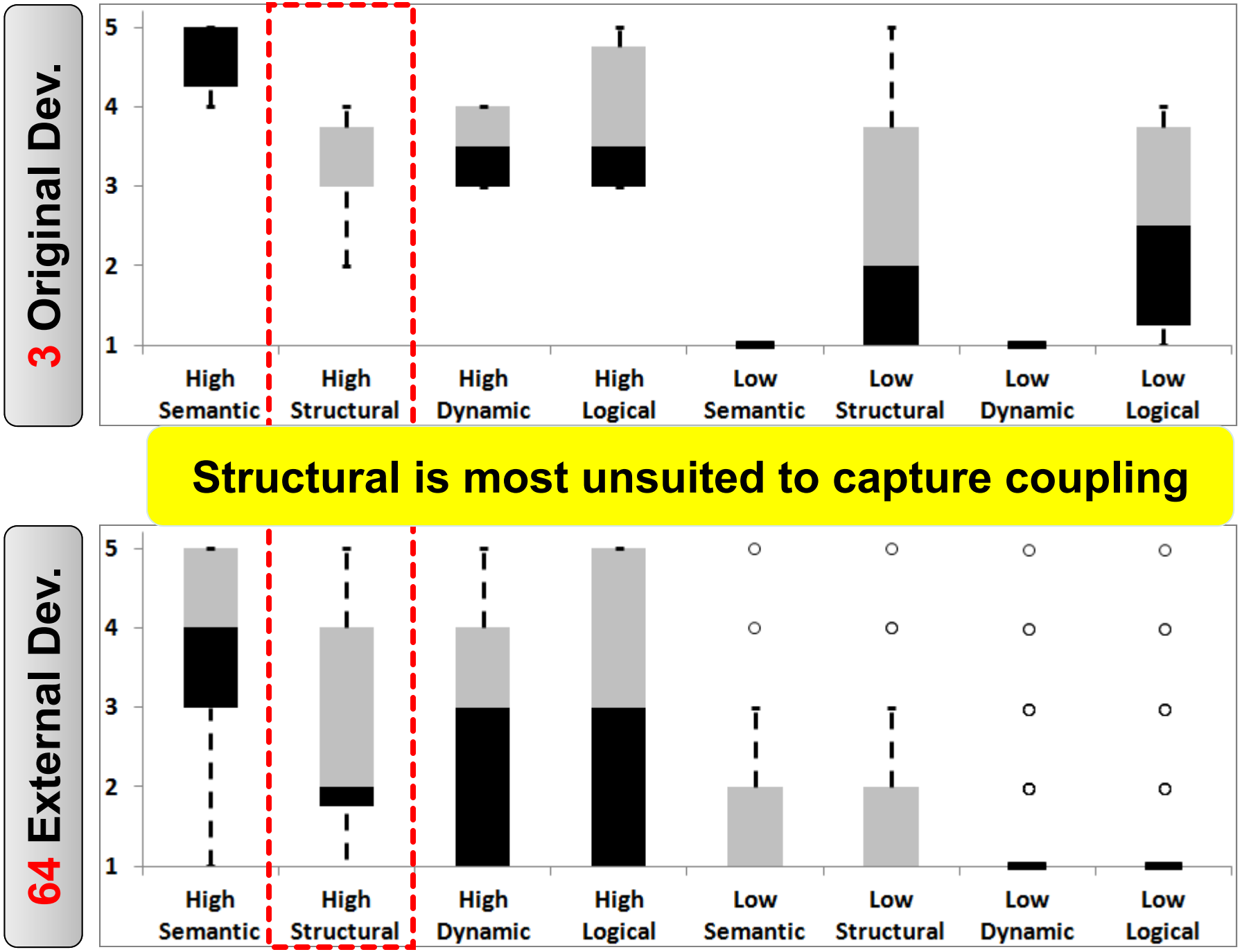
3 Original Dev.

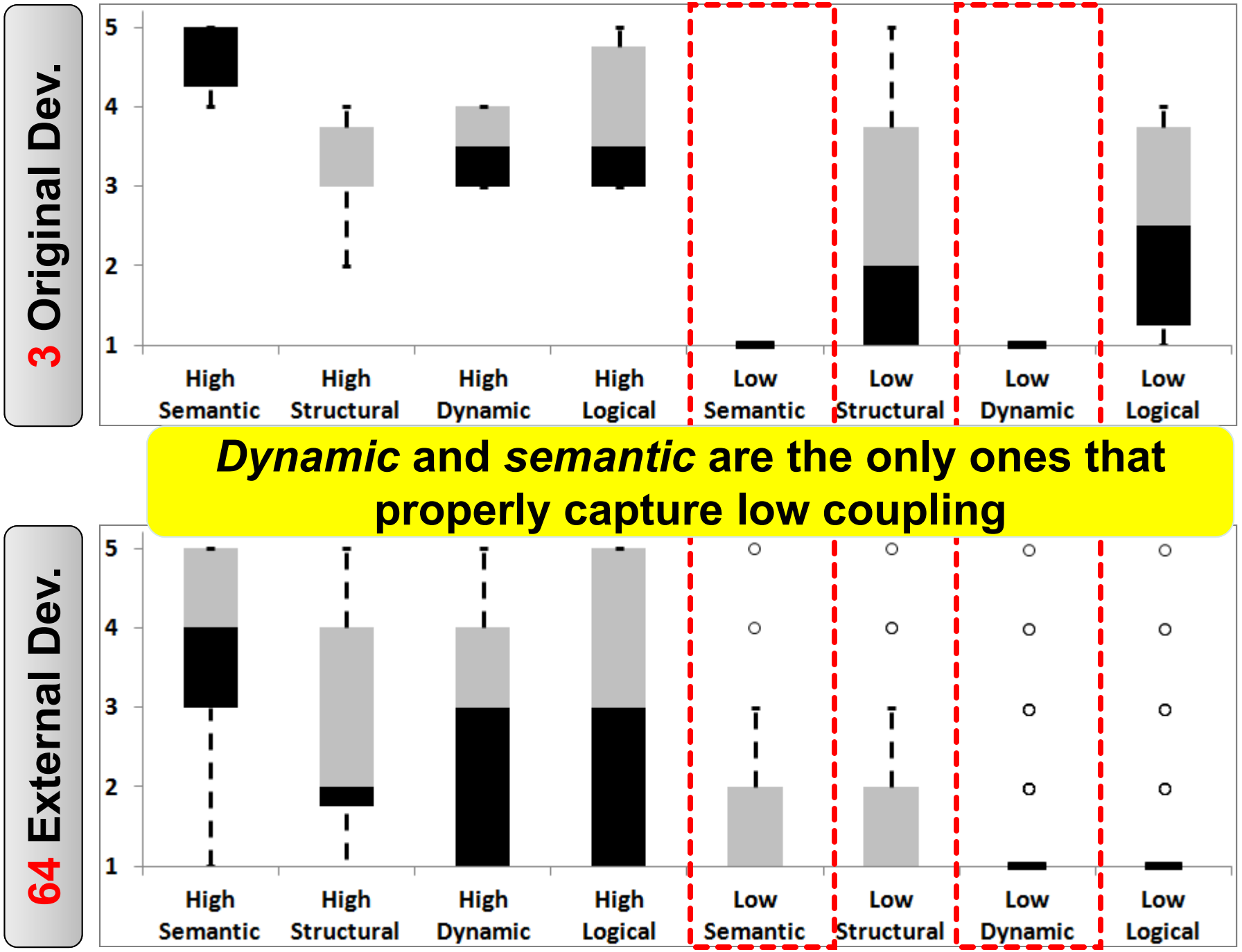


Semantic outperforms others *[p-value<0.05]*

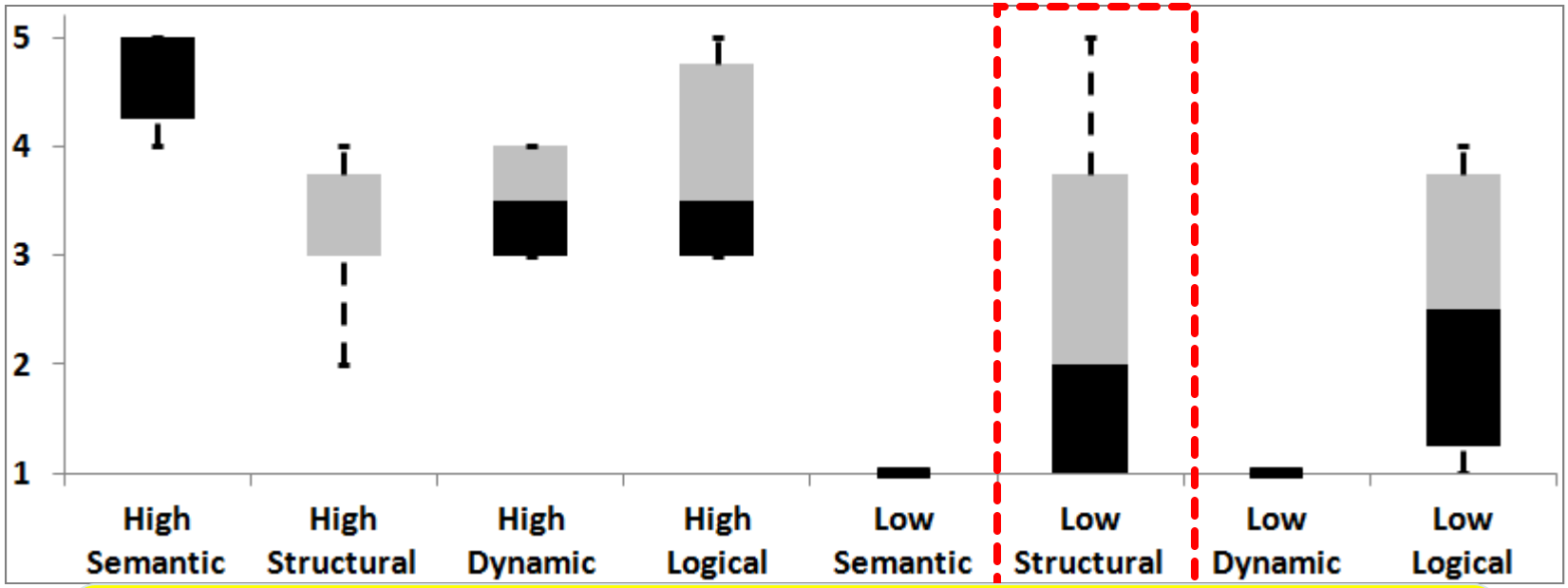
64 External Dev.




3 Original Dev.
5
4
3
2
1
High Semantic
High Structural
High Dynamic
High Logical
Low Semantic
Low Structural
Low Dynamic
Low Logical
Structural is most unsuited to capture coupling
64 External Dev.
5
4
3
2
1
High Semantic
High Structural
High Dynamic
High Logical
Low Semantic
Low Structural
Low Dynamic
Low Logical

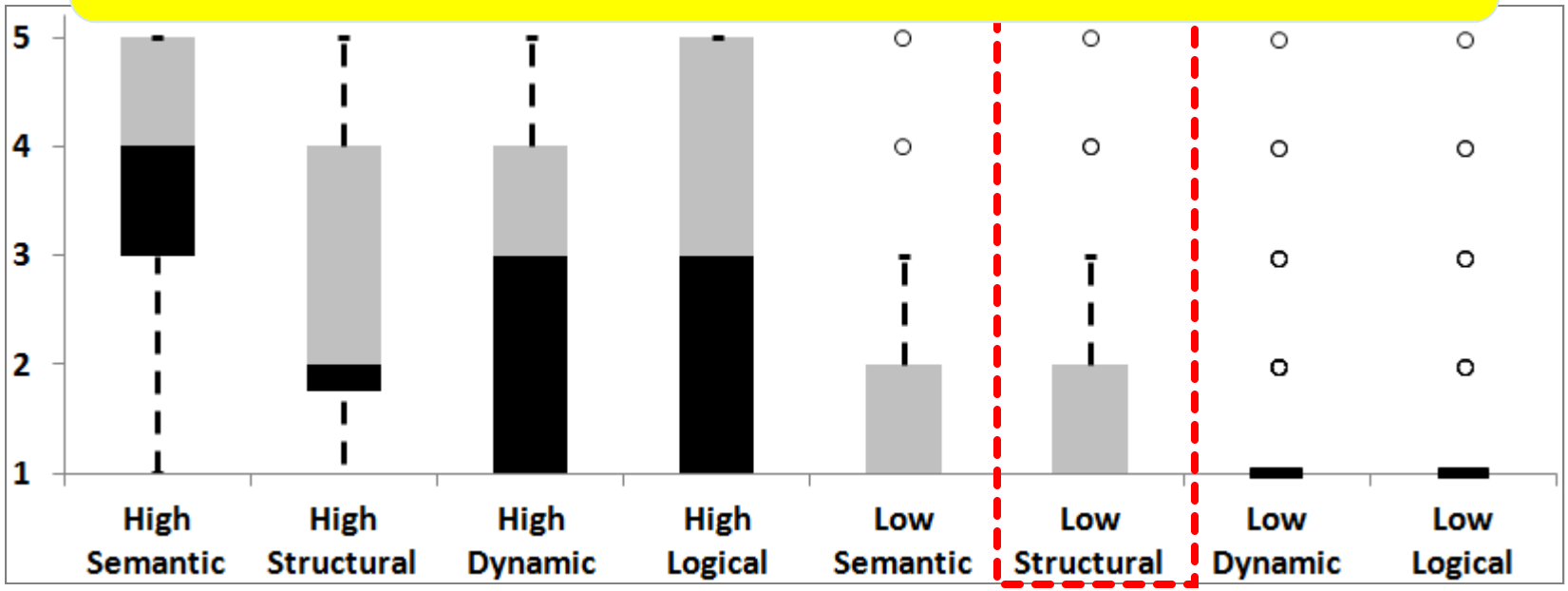

3 Original Dev.
5
4
3
2
1
High Semantic
High Structural
High Dynamic
High Logical
Low Semantic
Low Structural
Low Dynamic
Low Logical
Dynamic and semantic are the only ones that properly capture low coupling
64 External Dev.
5
4
3
2
1
High Semantic
High Structural
High Dynamic
High Logical
Low Semantic
Low Structural
Low Dynamic
Low Logical

3 Original Dev.



Structural produces strange results… Why?

64 External Dev.



ZoomTool

```
public class ZoomTool extends AbstractTool {

    private Tool child;

    public ZoomTool(DrawingEditor editor) {
        super(editor);
    }

    public void mouseDown(MouseEvent e, int x, int y) {
        super.mouseDown(e,x,y);
        //  Added handling for SHIFTed and CTRLed BUTTON3_MASK so
that normal
        //  BUTTON3_MASK does zoomOut, SHIFTed BUTTON3_MASK does
zoomIn
        //  and
        if ((e.g
            if
            }
            view
            child = new ZoomAreaTracker(editor());
            child.mouseDown(e, x, y);
```

ZoomUpdateStrategy

```
public class ZoomUpdateStrategy implements Painter {

    /**
     * The offscreen image
     */
    transient private Image fOffscreen;
    private int fImagewidth = -1;
    private int fImageheight = -1;

    /**
     * Draws the view contents.

     */
                                                    {
                                                    agewidth)
                                                    d.height);
            fImagewidth = d.width;
            fImageheight = d.height;
```

Classes with low coupling => perceived as coupled

ZoomTool

```
public class ZoomTool extends AbstractTool {

    private Tool child;

    public ZoomTool(DrawingEditor editor) {
        super(editor);
    }

    public void mouseDown(MouseEvent e, int x, int y) {
        super.mouseDown(e,x,y);
        //  Added handling for SHIFTed and CTRLed BUTTON3_MASK so
that normal
        //  BUTTON3_MASK does zoomOut, SHIFTed BUTTON3_MASK does
zoomIn
        //  and
        if ((e.g
            if
            }
            view
            child = new ZoomAreaTracker(editor());
```

ZoomUpdateStrategy

```
public class ZoomUpdateStrategy implements Painter {

    /**
     * The offscreen image
     */
    transient private Image fOffscreen;
    private int fImagewidth = -1;
    private int fImageheight = -1;

    /**
     * Draws the view contents.

     */
                                                    {
                                                    agewidth)
                                                    d.height);
            fImagewidth = d.width;
```

Classes with low coupling => perceived as coupled

Original developer:

"these classes are peer features participating in the same context"

ZoomTool

```
public class ZoomTool extends AbstractTool {

    private Tool child;

    public ZoomTool(DrawingEditor editor) {
        super(editor);
    }

    public void mouseDown(MouseEvent e, int x, int y) {
        super.mouseDown(e,x,y);
        //  Added handling for SHIFTed and CTRLed BUTTON3_MASK so
that normal
        //  BUTTON3_MASK does zoomOut, SHIFTed BUTTON3_MASK does
zoomIn
        //  and
        if ((e.g
            if
            }
            view
            child = new ZoomAreaTracker(editor());
```

ZoomUpdateStrategy

```
public class ZoomUpdateStrategy implements Painter {

    /**
     * The offscreen image
     */
    transient private Image fOffscreen;
    private int fImagewidth = -1;
    private int fImageheight = -1;

    /**
     * Draws the view contents.

     */
                                                    {
                                                    agewidth)
                                                    d.height);
            fImagewidth = d.width;
```

Classes with low coupling => perceived as coupled

Original developer:

"these classes are peer features participating in the same context"

Semantic measure captures this coupling (CCBC=0.52)

Semantic measure is not a silver bullet…

AWTCursor

```
public class AWTCursor extends Cursor implements org.jhotdraw.
framework.Cursor {

    /**
     * Constructor for <code>AWTCursor</code>.
     * @param type
     * @see Cursor#Cursor(int)
     */
    public AWTCursor(int type) {
        super(type);
    }
```

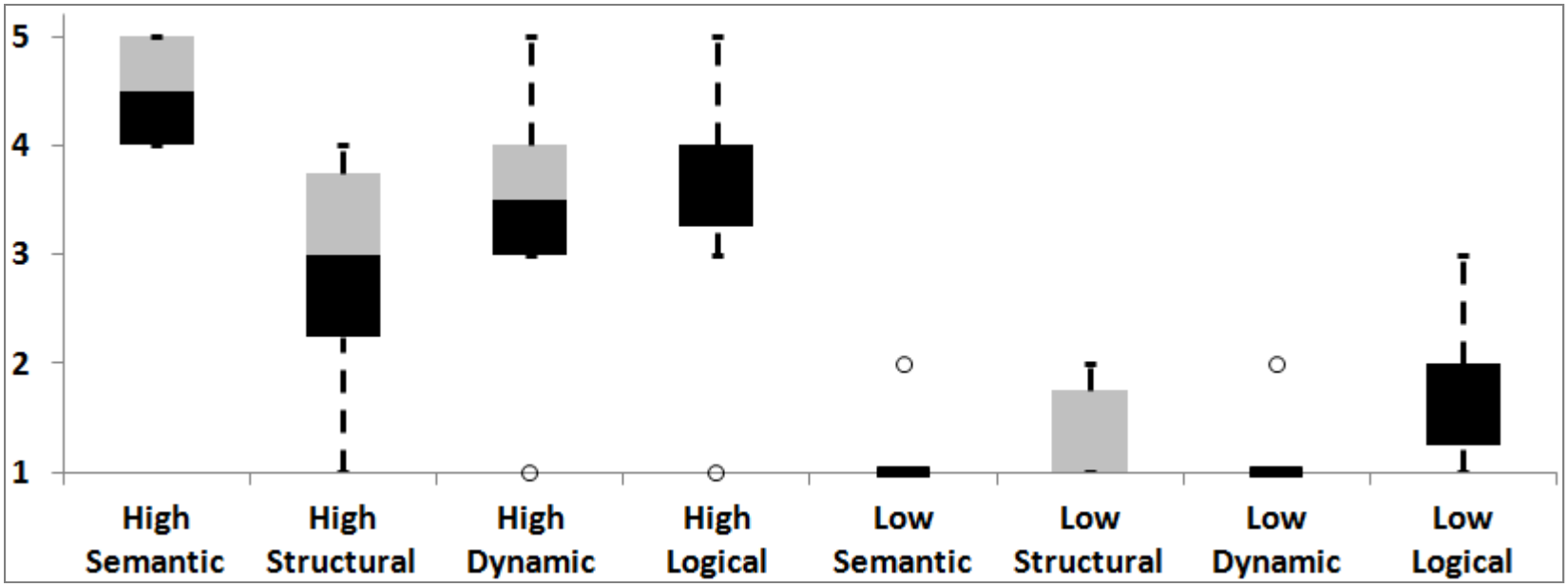
Locator

```
public interface Locator extends Storable, Serializable,
Cloneable {

    /**
     * Locates a position on the passed figure.
     * @return a point on the figure.
     */
    public Point locate(Figure owner);
}
```

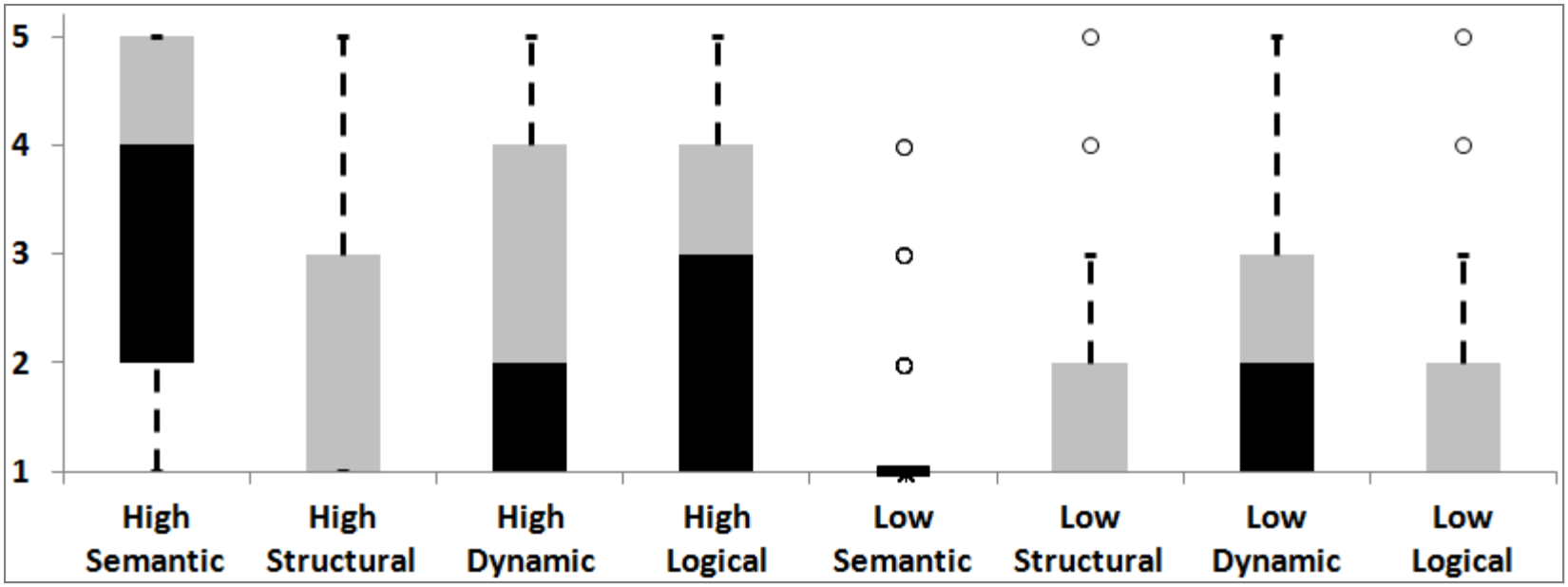
Not coupled, but semantic measure is high (CCBC=0.41)

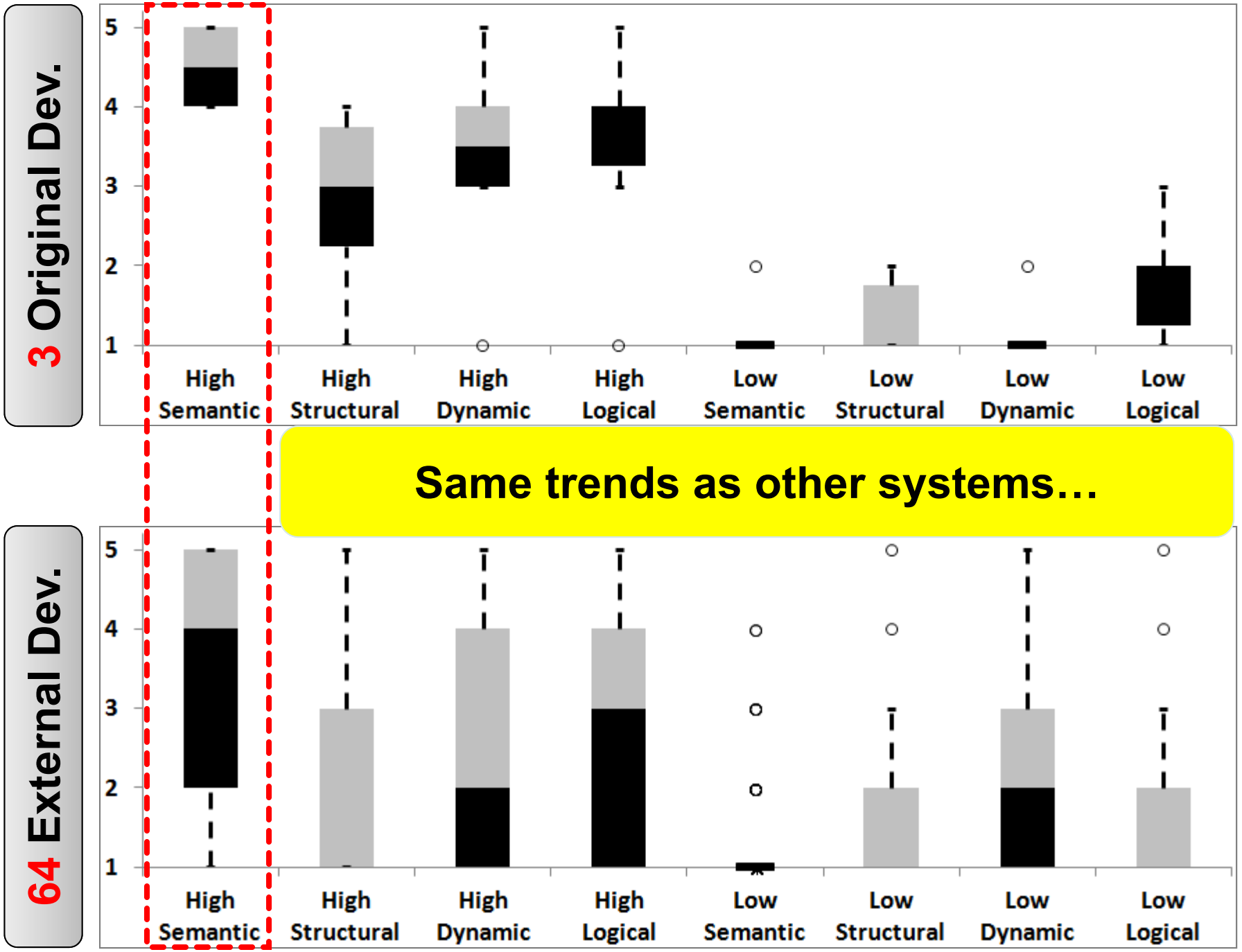

jEdit
programmer's text editor

3 Original Dev.



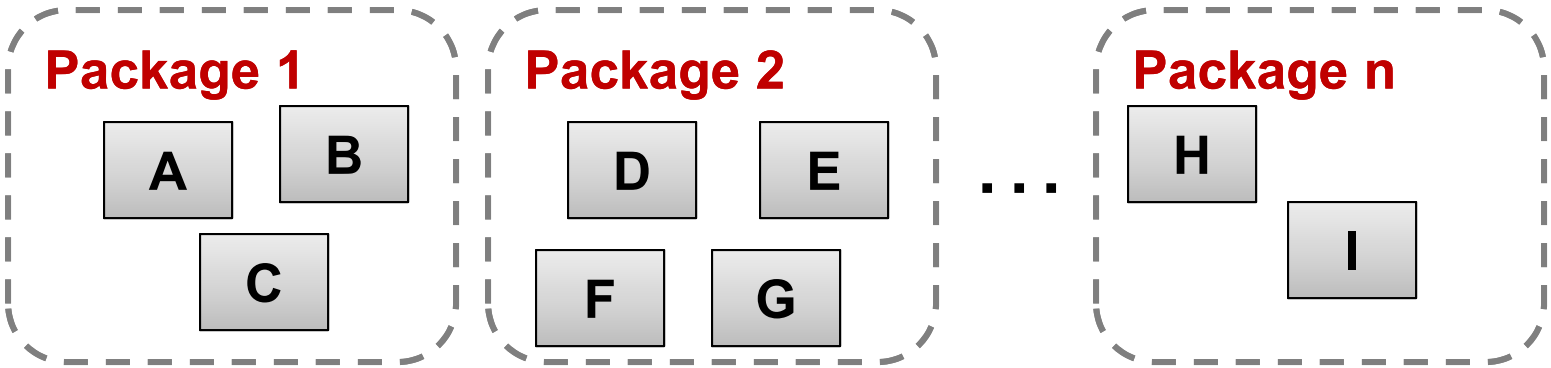
64 External Dev.

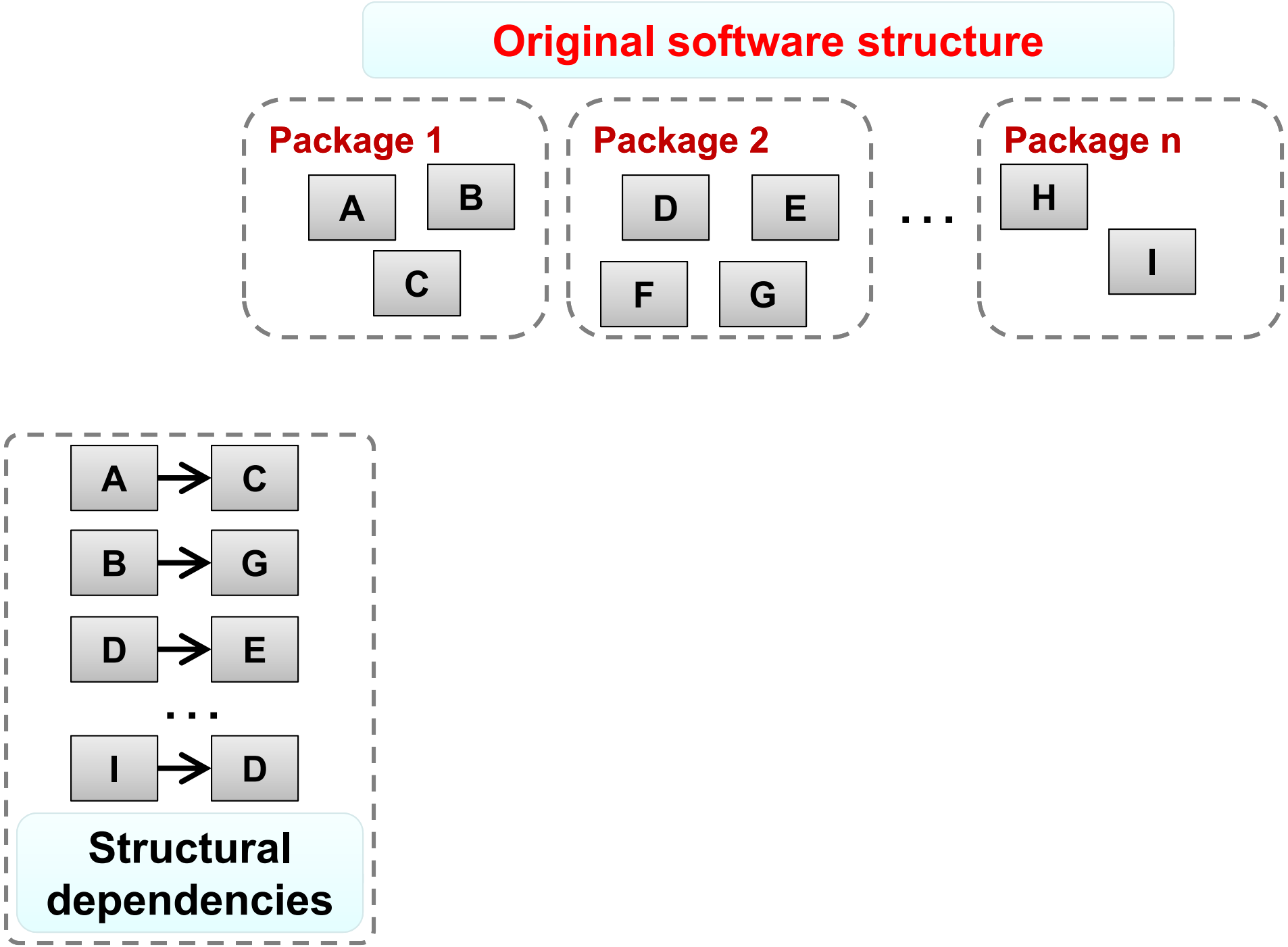


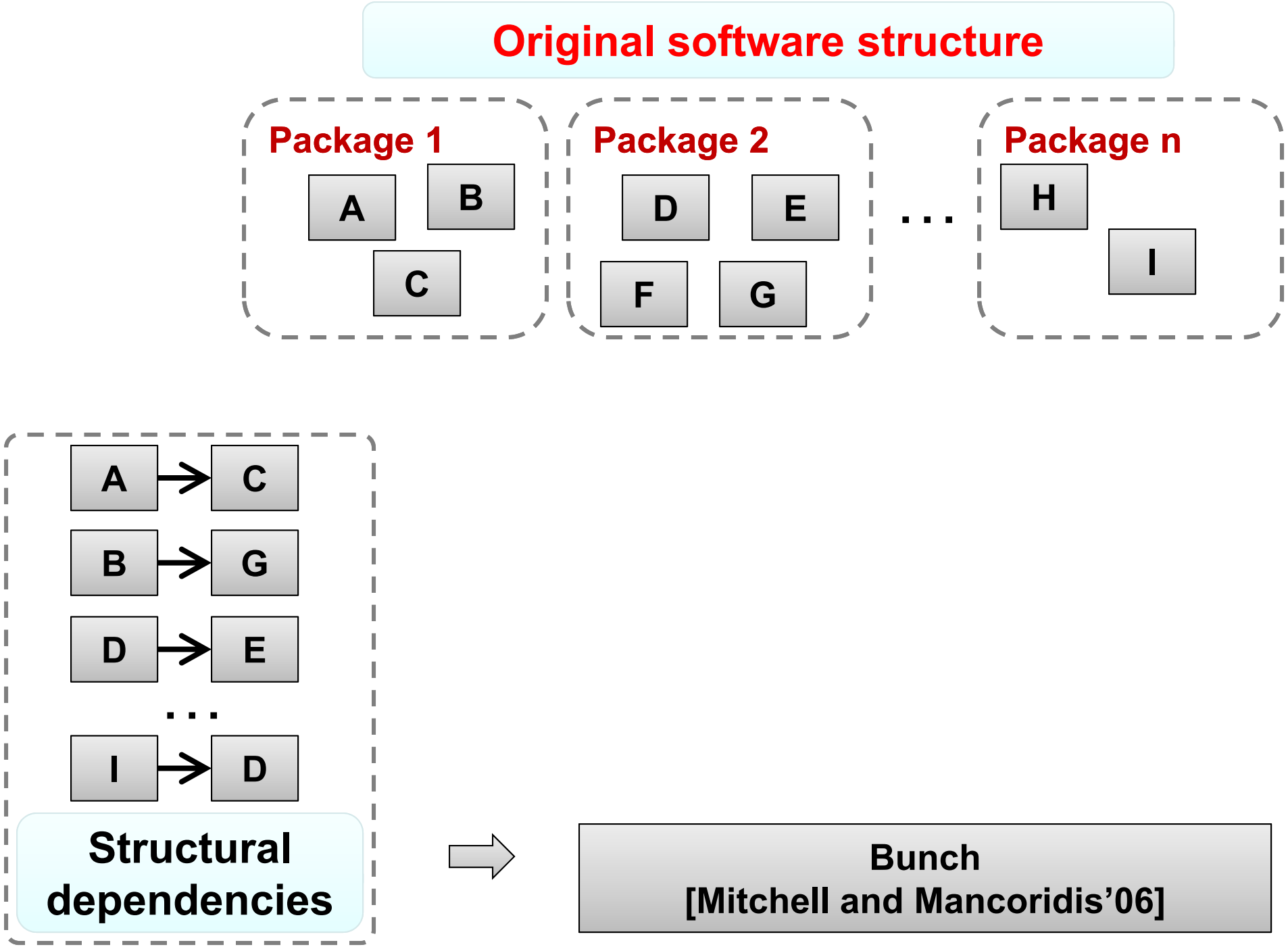

3 Original Dev.
64 External Dev.
5
4
3
2
1
High Semantic
High Structural
High Dynamic
High Logical
Low Semantic
Low Structural
Low Dynamic
Low Logical
Same trends as other systems…

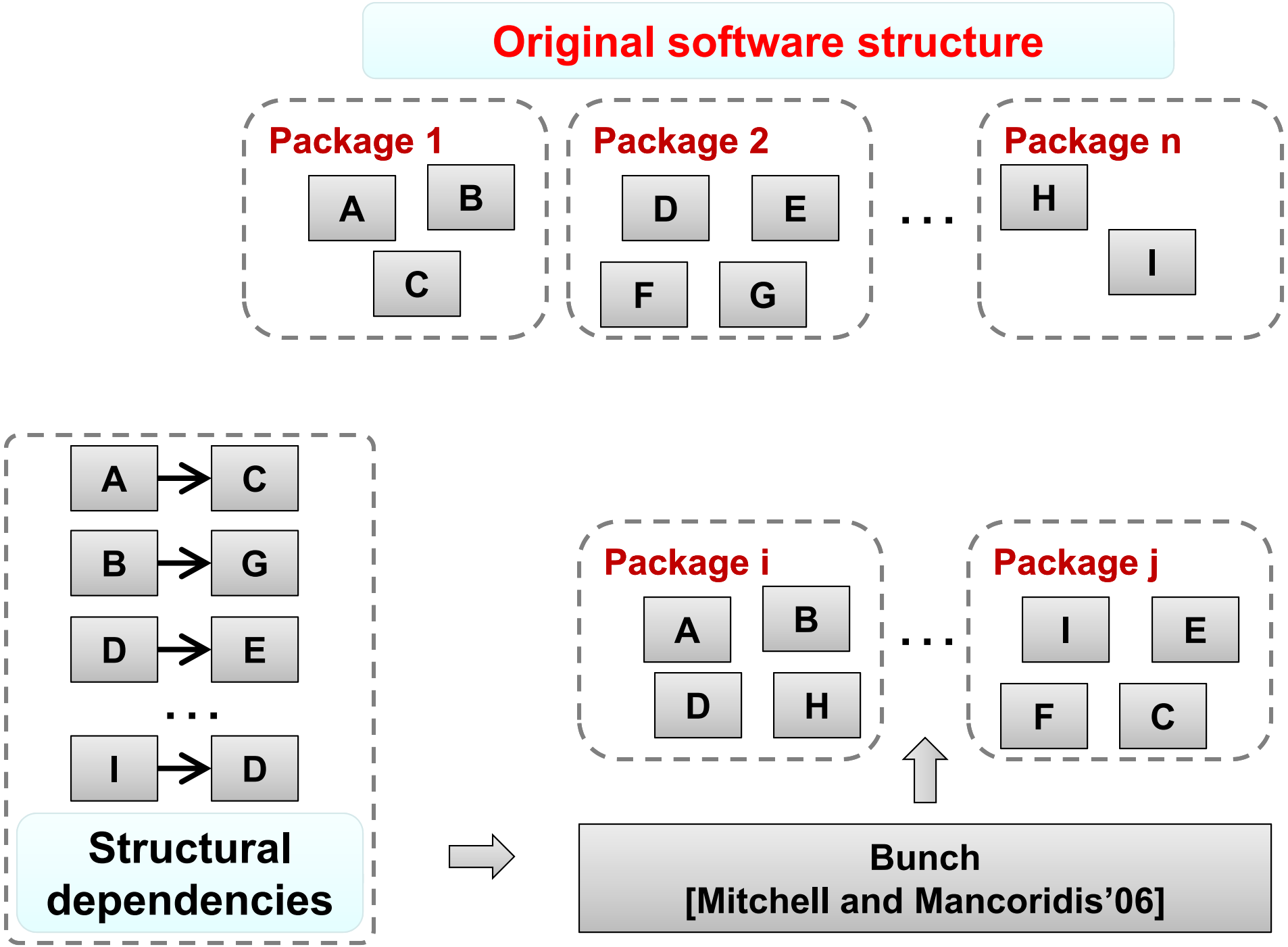
How can these types of coupling affect the remodularization of a software system?

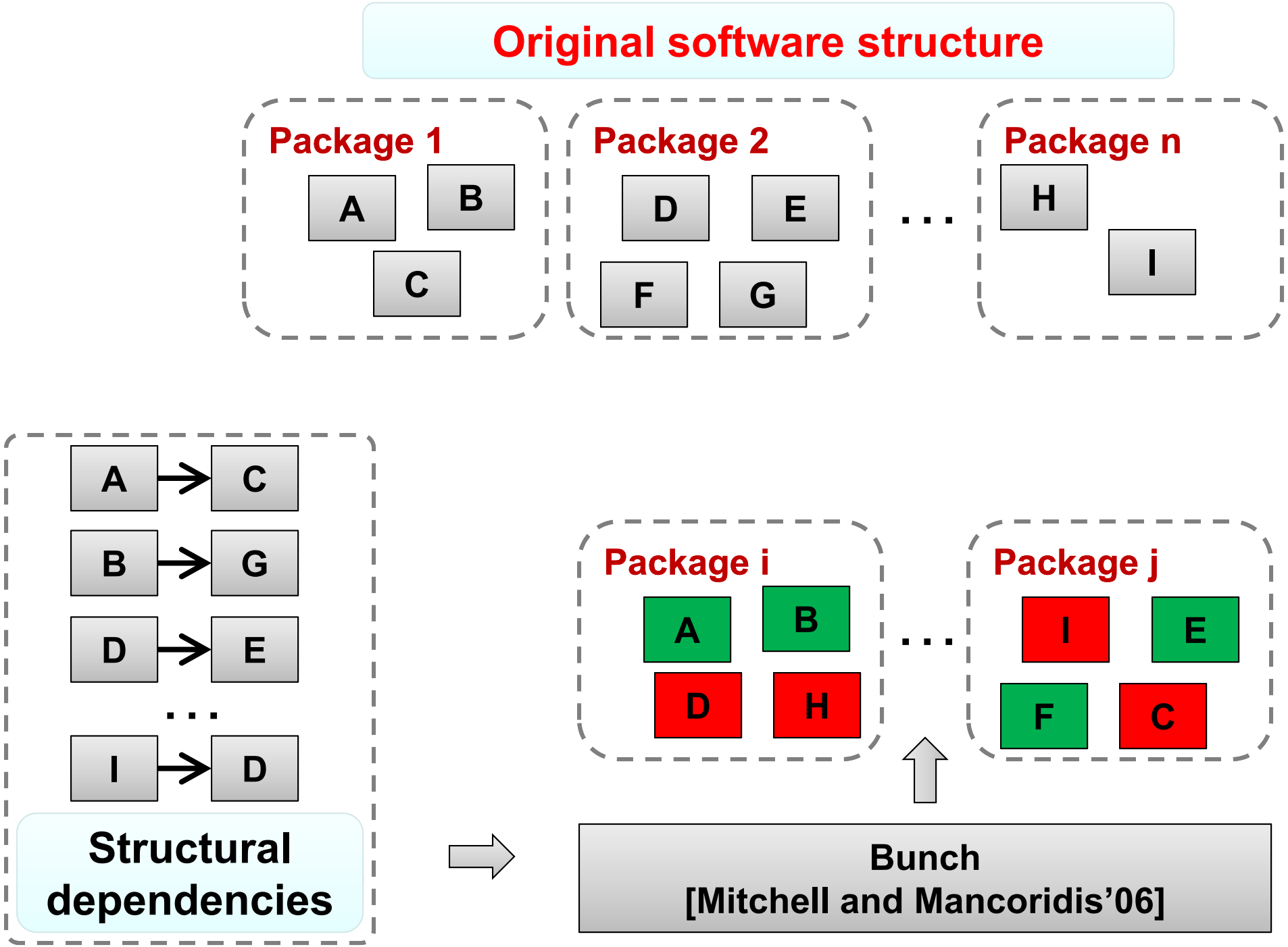
Original software structure

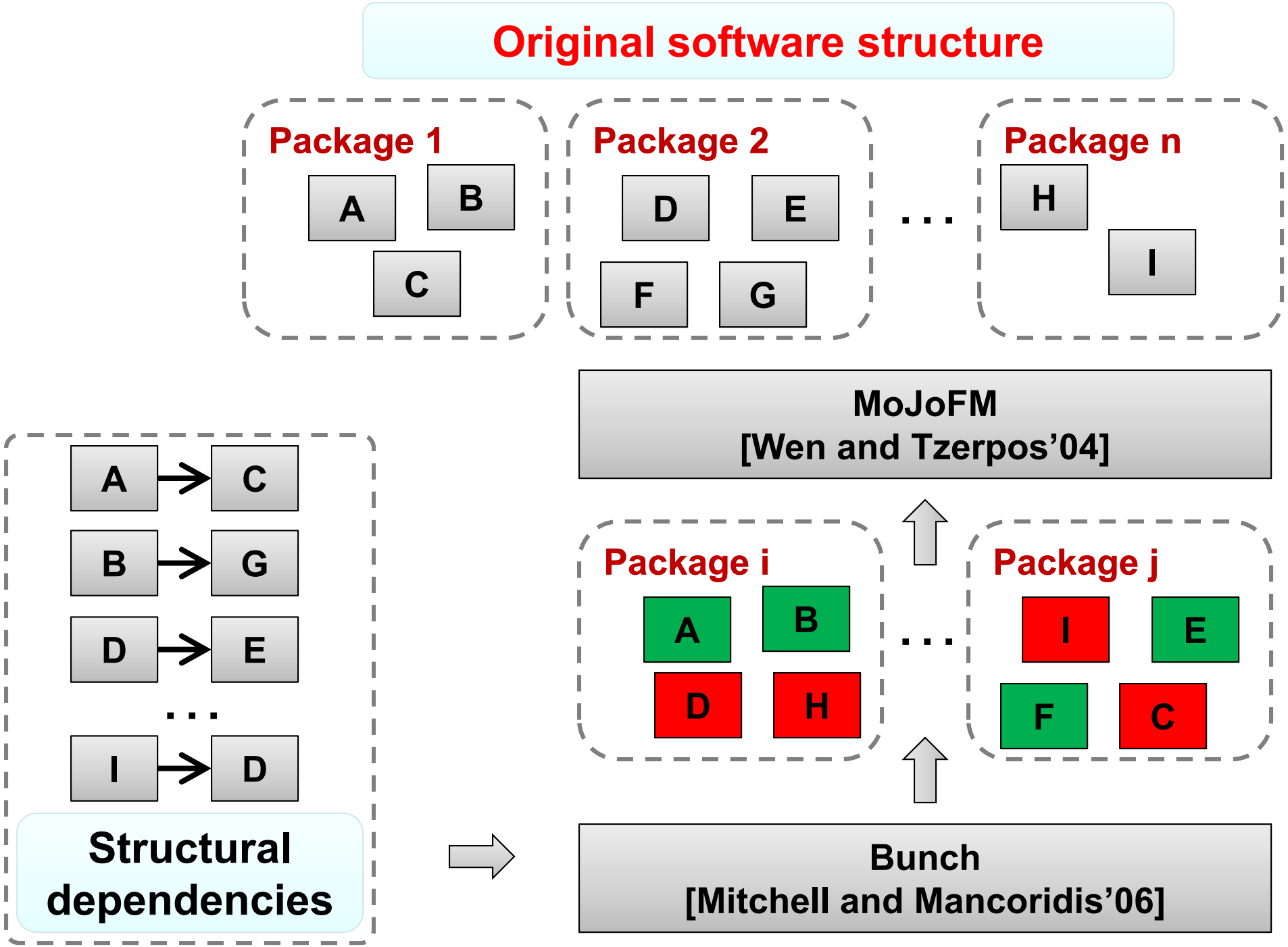


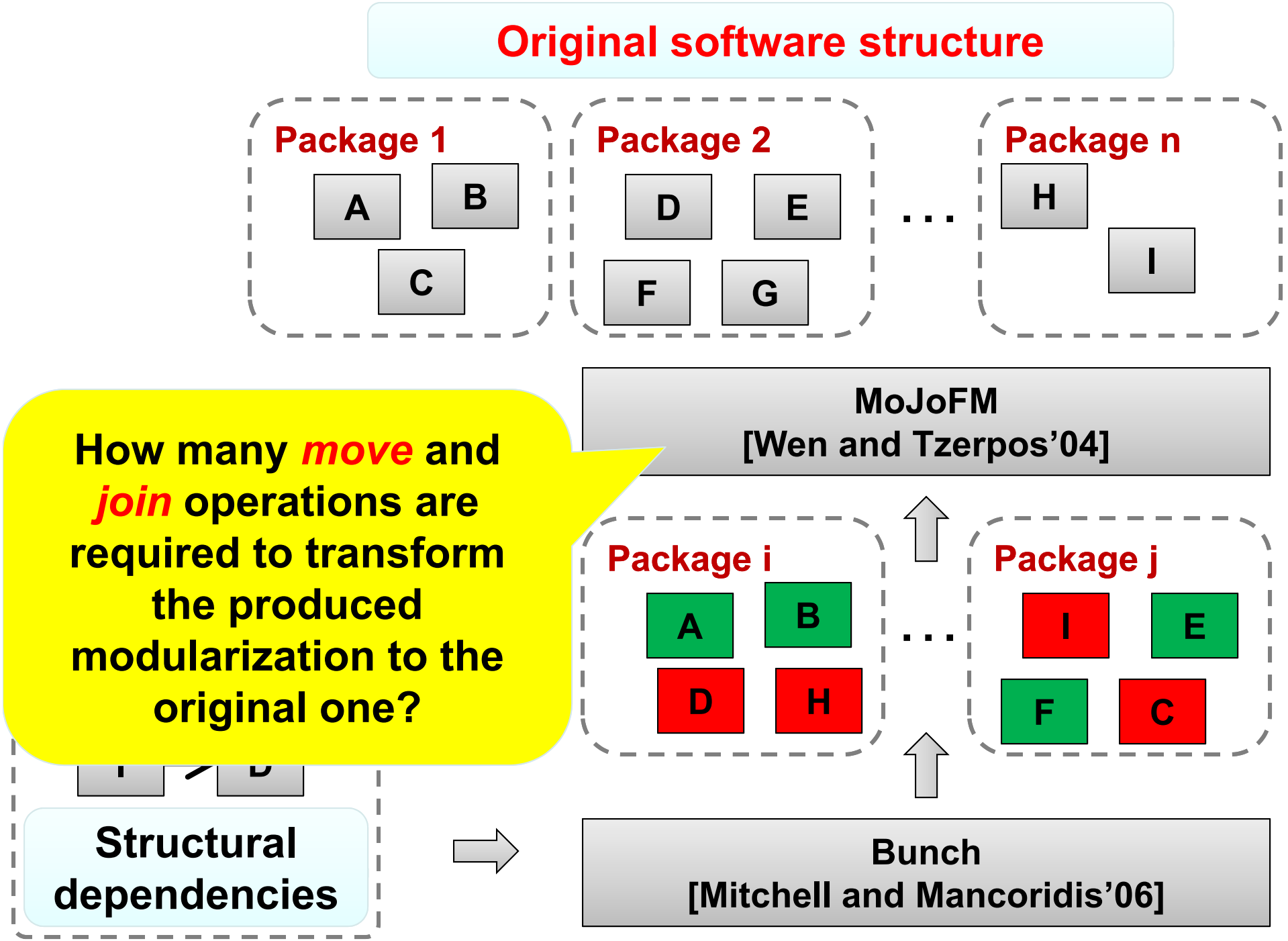

Original software structure
Package 1
A
B
C
Package 2
D
E
F
G
. . .
Package n
H
I
A
C
B
G
D
E
. . .
I
D
Structural dependencies


Original software structure
Package 1
A
B
C
Package 2
D
E
F
G
. . .
Package n
H
I
A
C
B
G
D
E
. . .
I
D
Structural dependencies
Bunch
[Mitchell and Mancoridis'06]


Original software structure
Package 1
A
B
C
Package 2
D
E
F
G
. . .
Package n
H
I
A
C
B
G
D
E
. . .
I
D
Structural dependencies
Package i
A
B
D
H
. . .
Package j
I
E
F
C
Bunch
[Mitchell and Mancoridis'06]


Original software structure
Package 1
A
B
C
Package 2
D
E
F
G
. . .
Package n
H
I
A
C
B
G
D
E
. . .
I
D
Structural dependencies
Package i
A
B
D
H
. . .
Package j
I
E
F
C
Bunch
[Mitchell and Mancoridis'06]

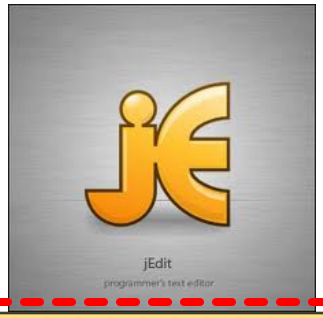

Original software structure
Package 1
A
B
C
Package 2
D
E
F
G
. . .
Package n
H
I
MoJoFM
[Wen and Tzerpos'04]
A
C
B
G
D
E
. . .
I
D
Structural dependencies
Package i
A
B
D
H
. . .
Package j
I
E
F
C
Bunch
[Mitchell and Mancoridis'06]


Original software structure
Package 1
A
B
C
Package 2
D
E
F
G
. . .
Package n
H
I
How many move and join operations are required to transform the produced modularization to the original one?
MoJoFM
[Wen and Tzerpos'04]
Package i
A
B
D
H
. . .
Package j
I
E
F
C
Structural dependencies
Bunch
[Mitchell and Mancoridis'06]

Median MoJoFM for 30 runs of Bunch (to account for randomness)

			JHotDraw
Structural	62%	43%	59%
Semantic	69%	49%	73%
Logical	3%	3%	6%
Dynamic	29%	26%	59%

Median MoJoFM for 30 runs of Bunch (to account for randomness)



JHotDraw

Structural	62%	43%	59%
Semantic	69%	49%	73%
Logical	3%	3%	6%
Dynamic	29%	26%	59%

Structural and semantic coupling produce better remodularity

Median MoJoFM for 30 runs of Bunch (to account for randomness)

JHotDraw

Structural	62%	43%	59%
Semantic	69%	49%	73%
Logical	3%	3%	6%
Dynamic	29%	26%	59%

Maybe not enough data?

Threats to Validity

- Metrics frequently used in the literature
- Performance of *dynamic* and *logical* metrics may have been impacted by insufficient data
- Pairs of classes may not be representative
- Remodularization task: use of current modularization of the system as *oracle*
- Choice of coupling measures

Conclusions

- Empirical study to assess the developer perception on software coupling
  - 76 developers
- Investigate which type of information is more suitable for remodularization
- We need more studies to generalize the results

Future Work

- Consider systems with more dynamic and logical information
- Use different coupling measures
- Consider combinations of coupling metrics

Thank you! Questions?

http://www.distat.unimol.it/reports/coupling

References

- B. S. Mitchell and S. Mancoridis, "On the automatic modularization of software systems using the bunch tool," IEEE Transactions on Software Engineering, vol. 32, no. 3, pp. 193–208, 2006.
- Z. Wen and V. Tzerpos, "An effectiveness measure for software clustering algorithms," in Proceedings of the 12th IEEE International Workshop on Program Comprehension, 2004, pp. 194–203.

Overlap

#Links	#Exclusive links
124,346	48,471
1,078,663	1,002,358
180	0
3,138	89

ArgoUML	Structural	Semantic	Logical	Dynamic
Structural	-	61%	1%	2%
Semantic	7%	-	1%	1%
Logical	24%	88%	-	13%
Dynamic	89%	81%	1%	-

#Links	#Exclusive links
5,029	3,042
9,613	7,665
52	6
358	31

JHotDraw	Structural	Semantic	Logical	Dynamic
Structural	-	37%	1%	5%
Semantic	20%	-	1%	2%
Logical	37%	39%	-	29%
Dynamic	85%	64%	4%	-

#Links	#Exclusive links
1,924	822
21,152	19,953
87	5
453	50

jEdit	Structural	Semantic	Logical	Dynamic
Structural	-	54%	1%	13%
Semantic	5%	-	1%	2%
Logical	52%	40%	-	14%
Dynamic	84%	75%	2%	-