Inquiring the usage of aspect-oriented programming: an empirical study

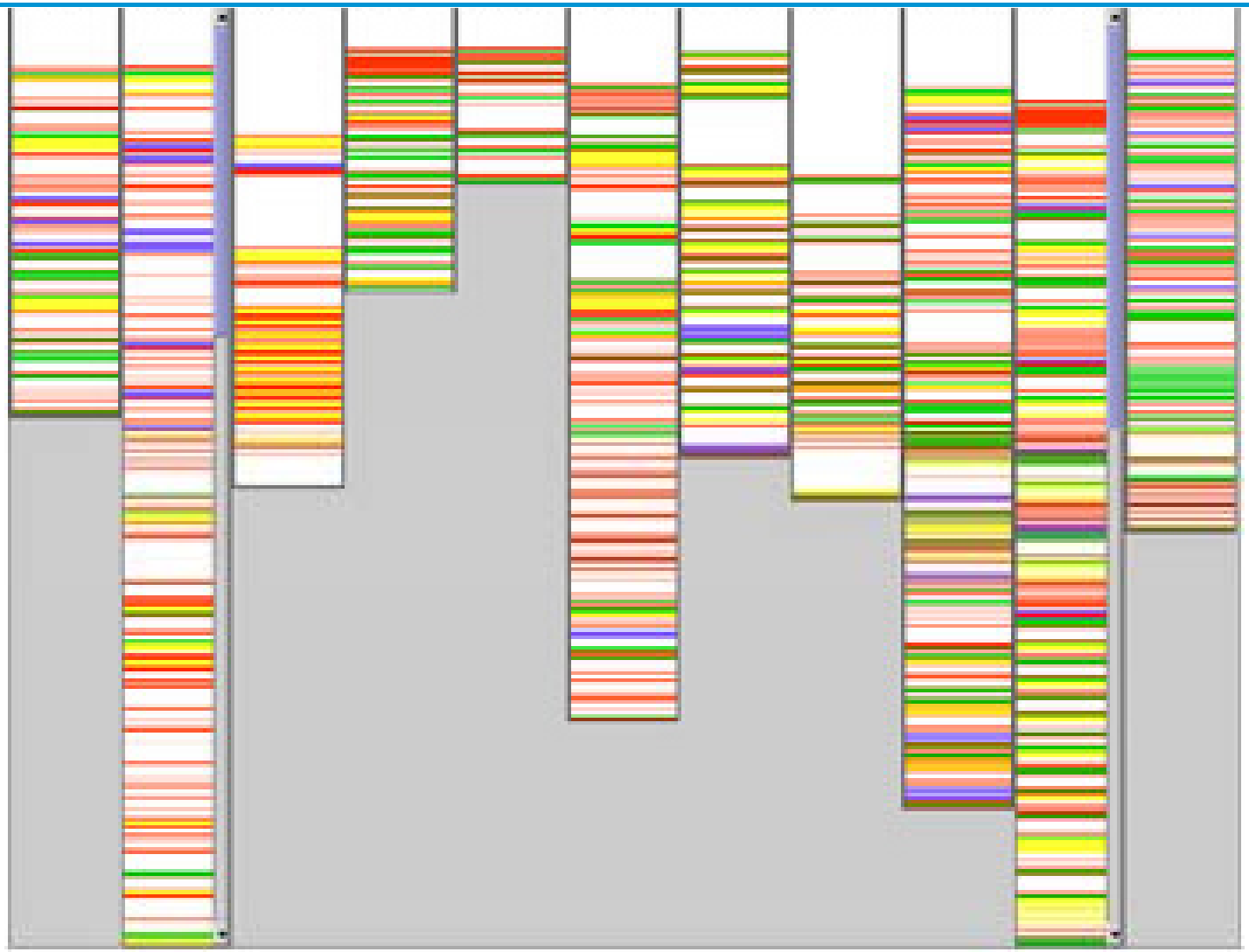
Freddy Muñoz, Benoit Baudry, Romain Delamare

INRIA Rennes – Bretagne Atlantique

Yves Le Traon

IT-Telecom Bretagne

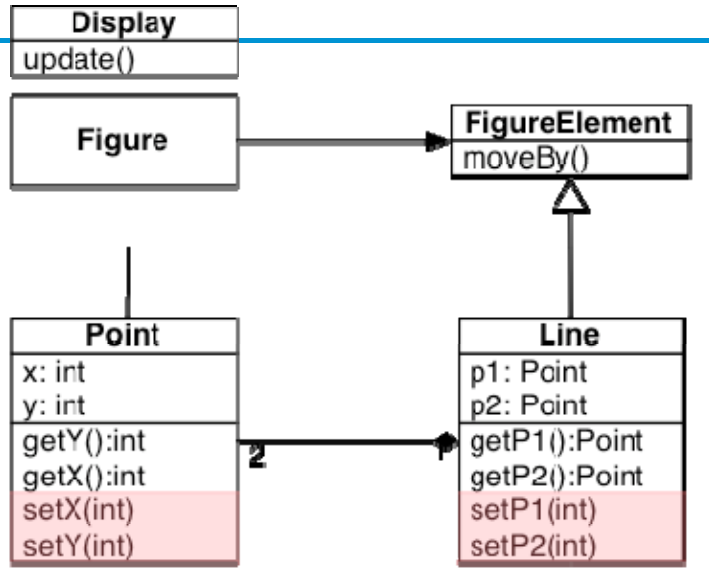
Aspect oriented programming



Aspect oriented programming

```
class Point implements FigureElement {
   private int x = 0, y = 0;
   int getX() { return x;}
   int getY() { return y;}
   void setX(int x) {
        this.x = x;
        Display.update();
   }
   void setY(int y) {
        this.y = y;
        Display.update();
   }
   void moveBy(int dx, int dy) {
        ...
        Display.update();
   }
}
```

```
class Line implements
FigureElement{
    private Point p1,
    Point getP1() {
        return p1;}
    Point getP2() {
        return p2;}
    void setP1(Point p1) {
        this.p1 = p1;
        Display.update();
    }
    void setP2(Point p2) {
        this.p2 = p2;
        Display.update();
    }
a    void moveBy(int dx, int dy)
{
        ...
        Display.update();
}}
```



Aspect oriented programming

```
class Point implements FigureElement {
  private int x = 0, y = 0;

   int getX() { return x; }
   int getY() { return y; }
   void setX(int x) {
        this.x = x;
   }
   void setY(int y) {
        this.y = y;
   }
   void moveBy(int dx, int dy) {
        ...
   }
}
```

```
 class Line implements FigureElement{
    private Point p1, p2;

     Point getP1() { return p1; }
     Point getP2() { return p2; }
     void setP1(Point p1) {
          this.p1 = p1;
     }
     void setP2(Point p2) {
          this.p2 = p2;
     }
     void moveBy(int dx, int dy) {
          ...
     }
 }
```

```
aspect DisplayUpdating {
    pointcut move():
         call(void FigureElement.moveBy(int, int))||
         call(void Line.setP1(Point))              ||
         call(void Line.setP2(Point))              ||
         call(void Point.setX(int))                ||
         call(void Point.setY(int));
    after(): move() {
         Display.update();
    }
}
```

Aspect oriented programming

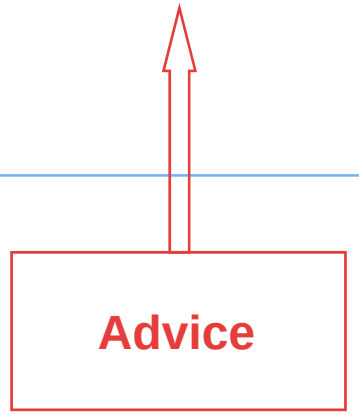
```
class Point implements FigureElement {
  private int x = 0, y = 0;

   int getX() { return x; }
   int getY() { return y; }
   void setX(int x) {
        this.x = x;
   }
   void setY(int y) {
        this.y = y;
   }
   void moveBy(int dx, int dy) {
        ...
   }
}
```

```
 class Line implements FigureElement{
    private Point p1, p2;

     Point getP1() { return p1; }
     Point getP2() { return p2; }
     void setP1(Point p1) {
          this.p1 = p1;
     }
     void setP2(Point p2) {
          this.p2 = p2;
     }
     void moveBy(int dx, int dy) {
          ...
     }
 }
```

```
aspect DisplayUpdating {
    pointcut move():
         call(void FigureElement.moveBy(int, int))||
         call(void Line.setP1(Point))              ||
         call(void Line.setP2(Point))              ||
         call(void Point.setX(int))                ||
         call(void Point.setY(int));
    after(): move() {
         Display.update();
    }
}
```



Aspect oriented programming

```
class Point implements FigureElement {
  private int x = 0, y = 0;

   int getX() { return x; }
   int getY() { return y; }
   void setX(int x) {
       this.x = x;
   }
   void setY(int y) {
       this.y = y;
   }
   void moveBy(int dx, int dy) {
       ...
   }
}
```

```
 class Line implements FigureElement{
    private Point p1, p2;

     Point getP1() { return p1; }
     Point getP2() { return p2; }
     void setP1(Point p1) {
         this.p1 = p1;
     }
     void setP2(Point p2) {
         this.p2 = p2;
     }
     void moveBy(int dx, int dy) {
         ...
     }
 }
```

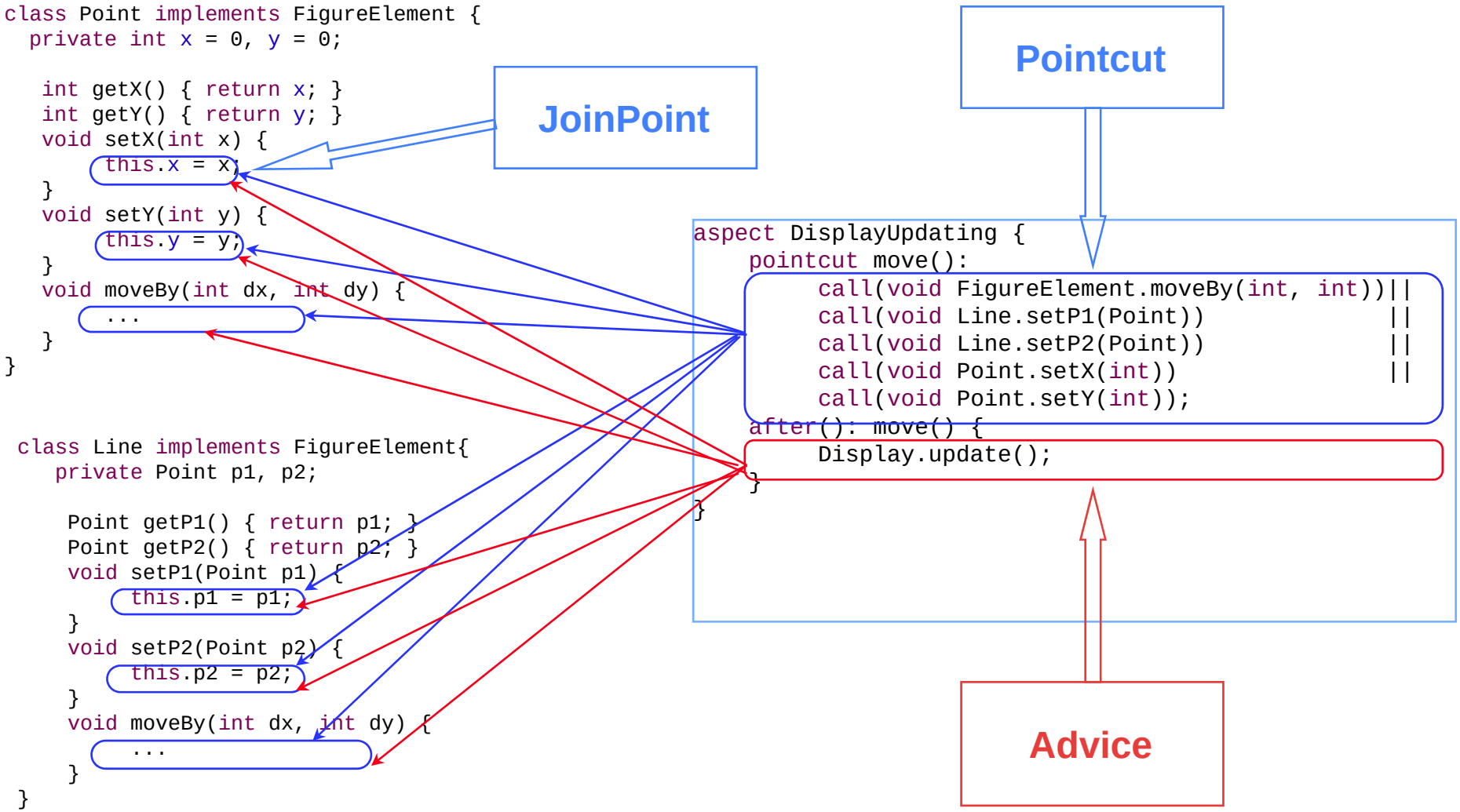
JoinPoint

Pointcut

```
aspect DisplayUpdating {
    pointcut move():
        call(void FigureElement.moveBy(int, int))||
        call(void Line.setP1(Point))              ||
        call(void Line.setP2(Point))              ||
        call(void Point.setX(int))                ||
        call(void Point.setY(int));
    after(): move() {
        Display.update();
    }
}
```

Advice

Aspect oriented programming

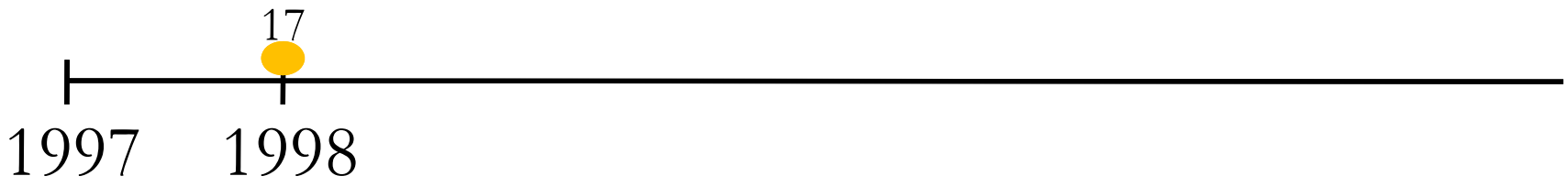


AOP in the last decade

Kiczales, et al. (1997).
Aspect-Oriented Programming. ECOOP'97

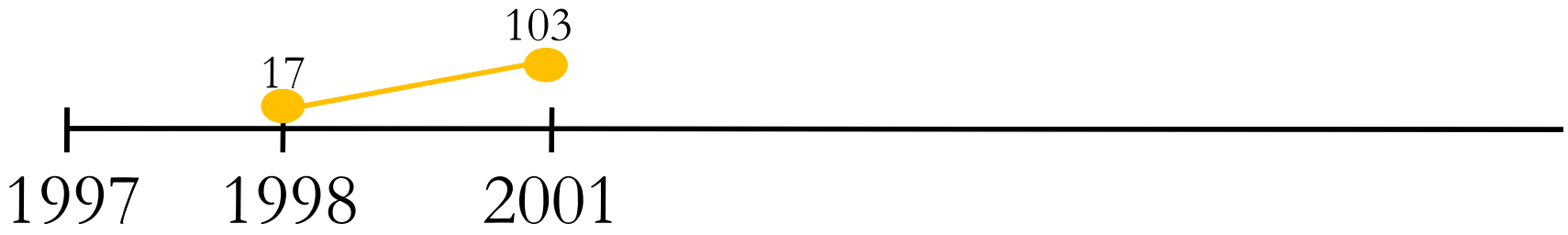
1997

AOP in the last decade

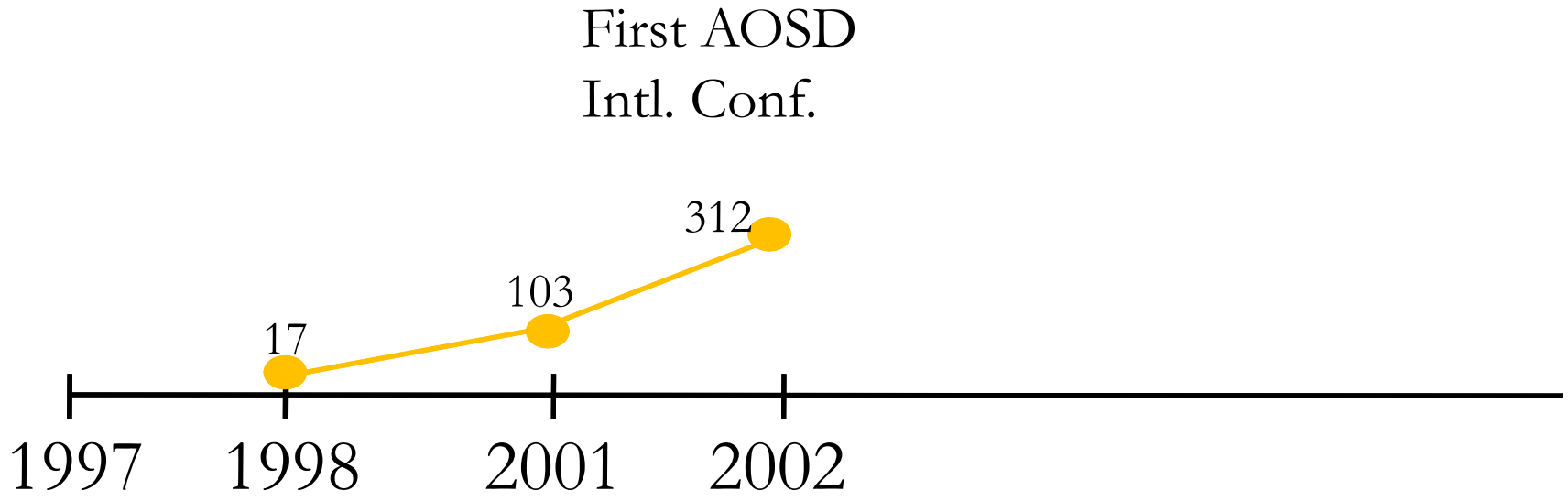


AOP in the last decade

MIT announces AOP
as a key technology for
the future 10 years

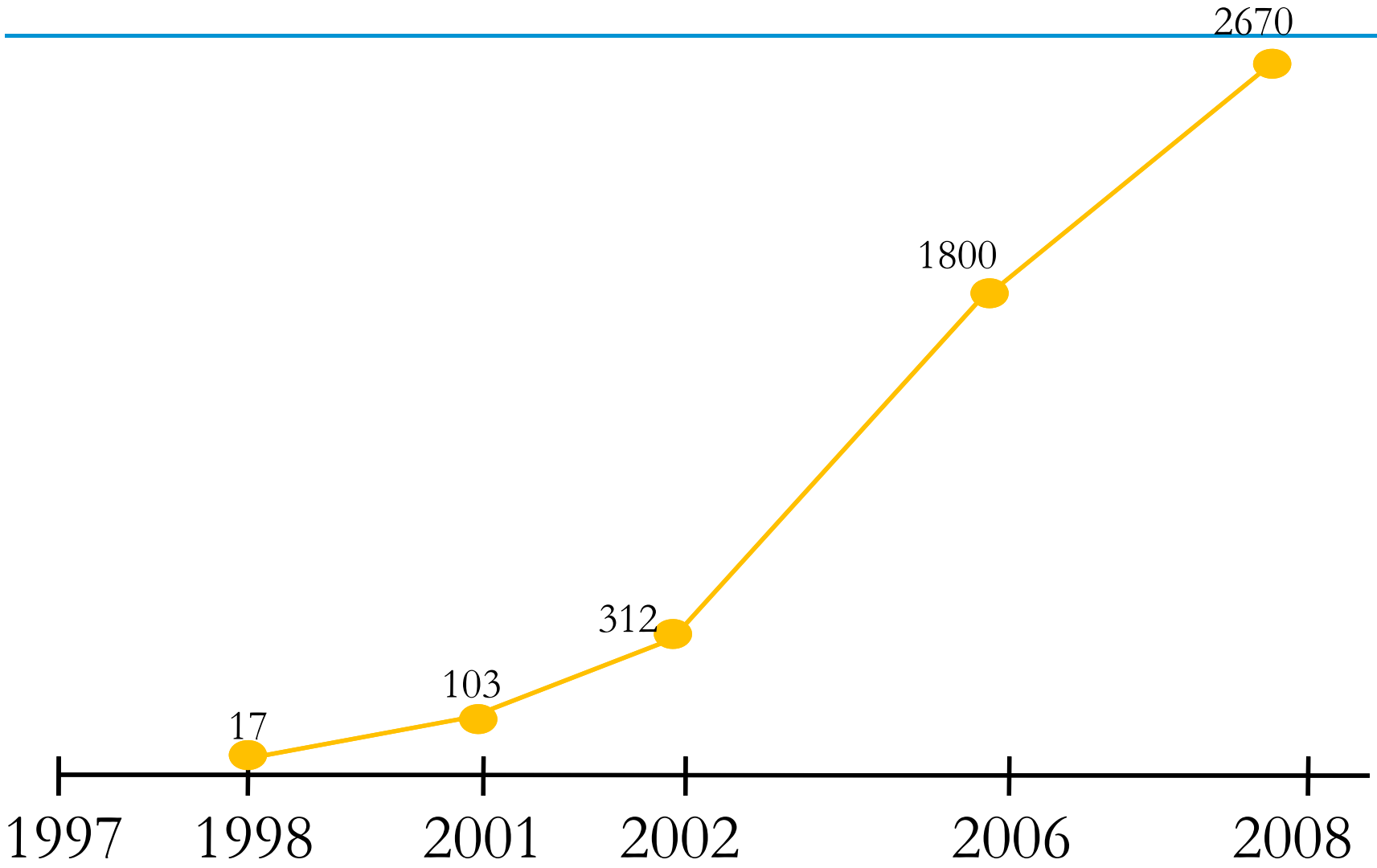


AOP in the last decade

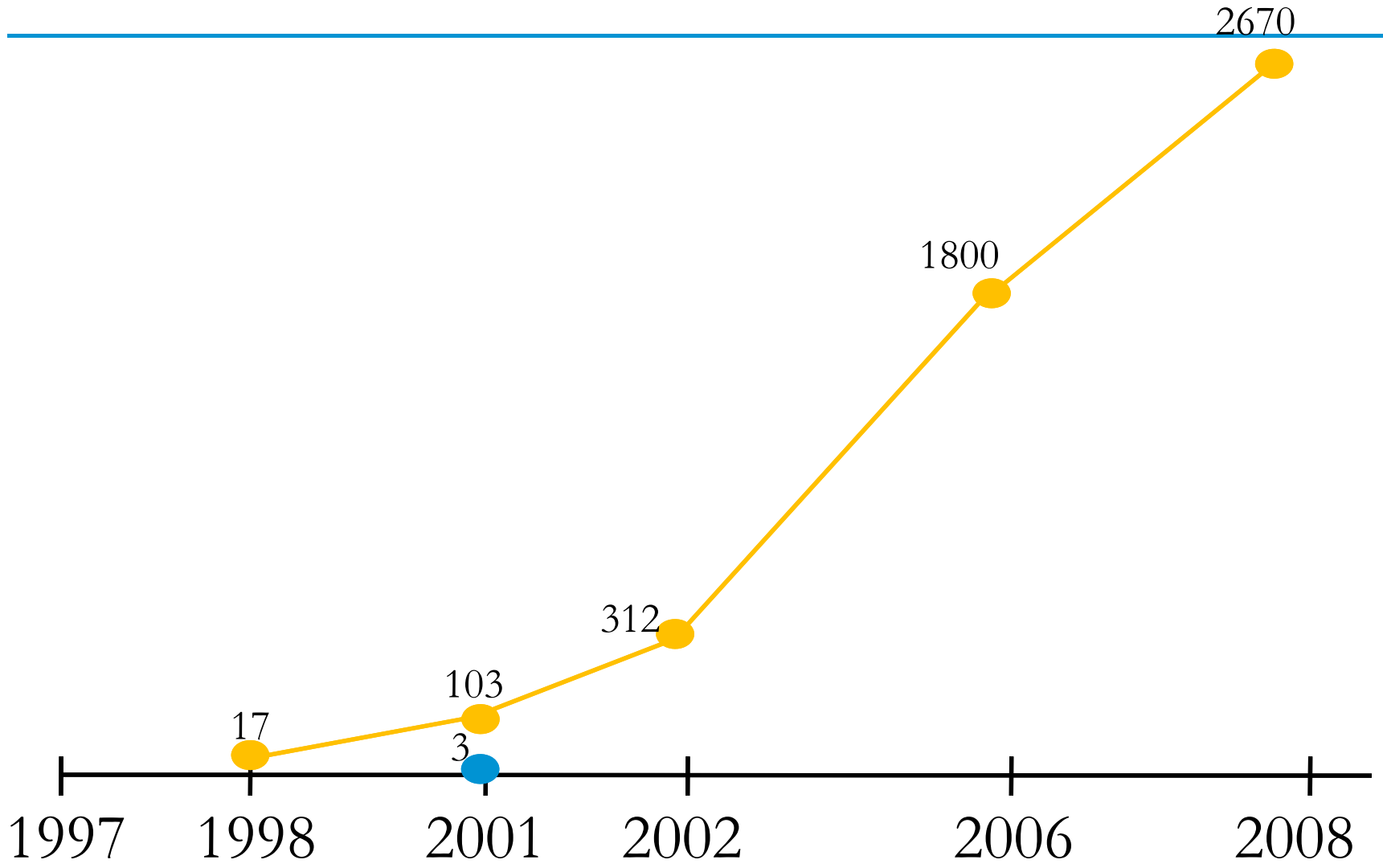


INRIA

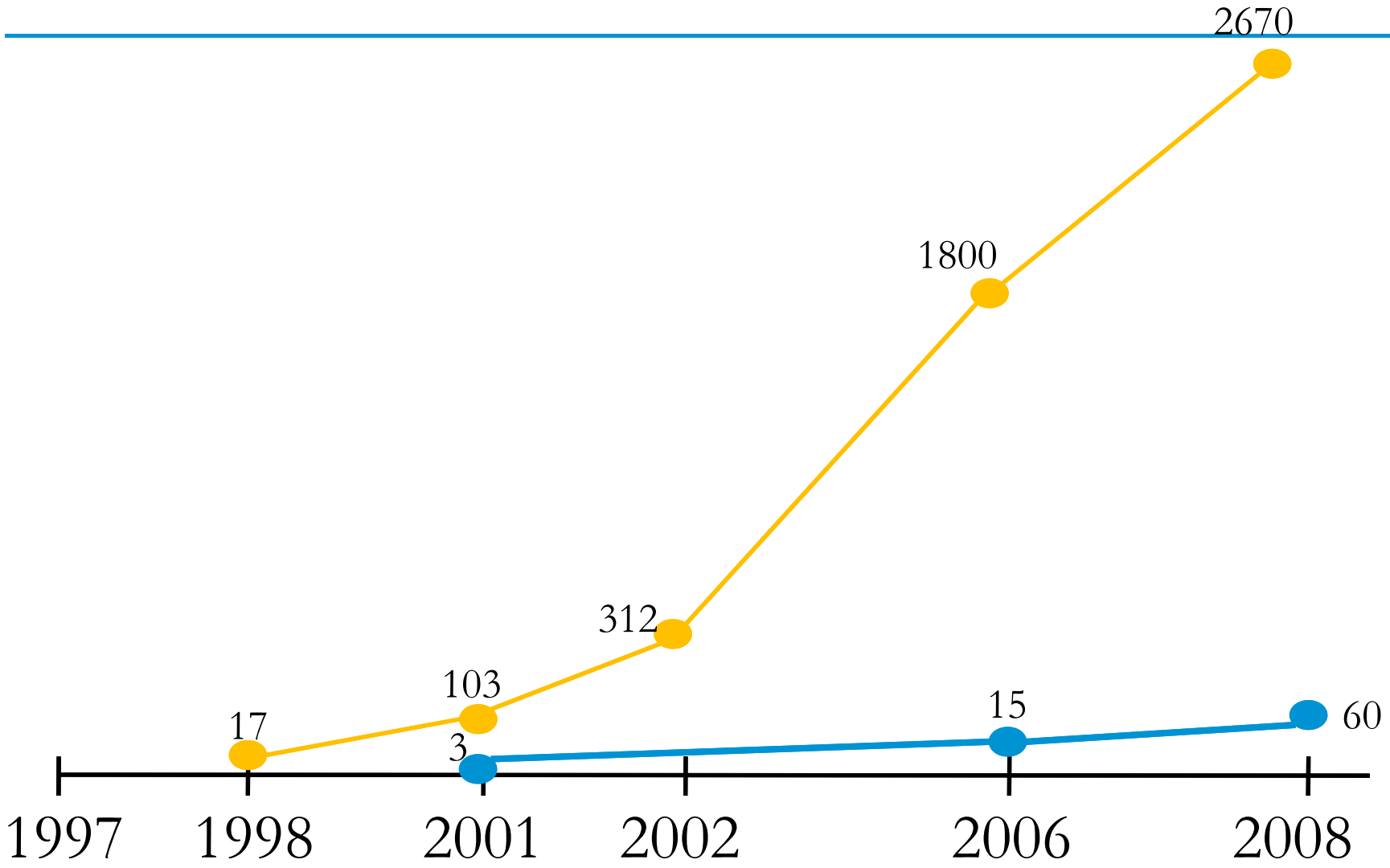
AOP in the last decade



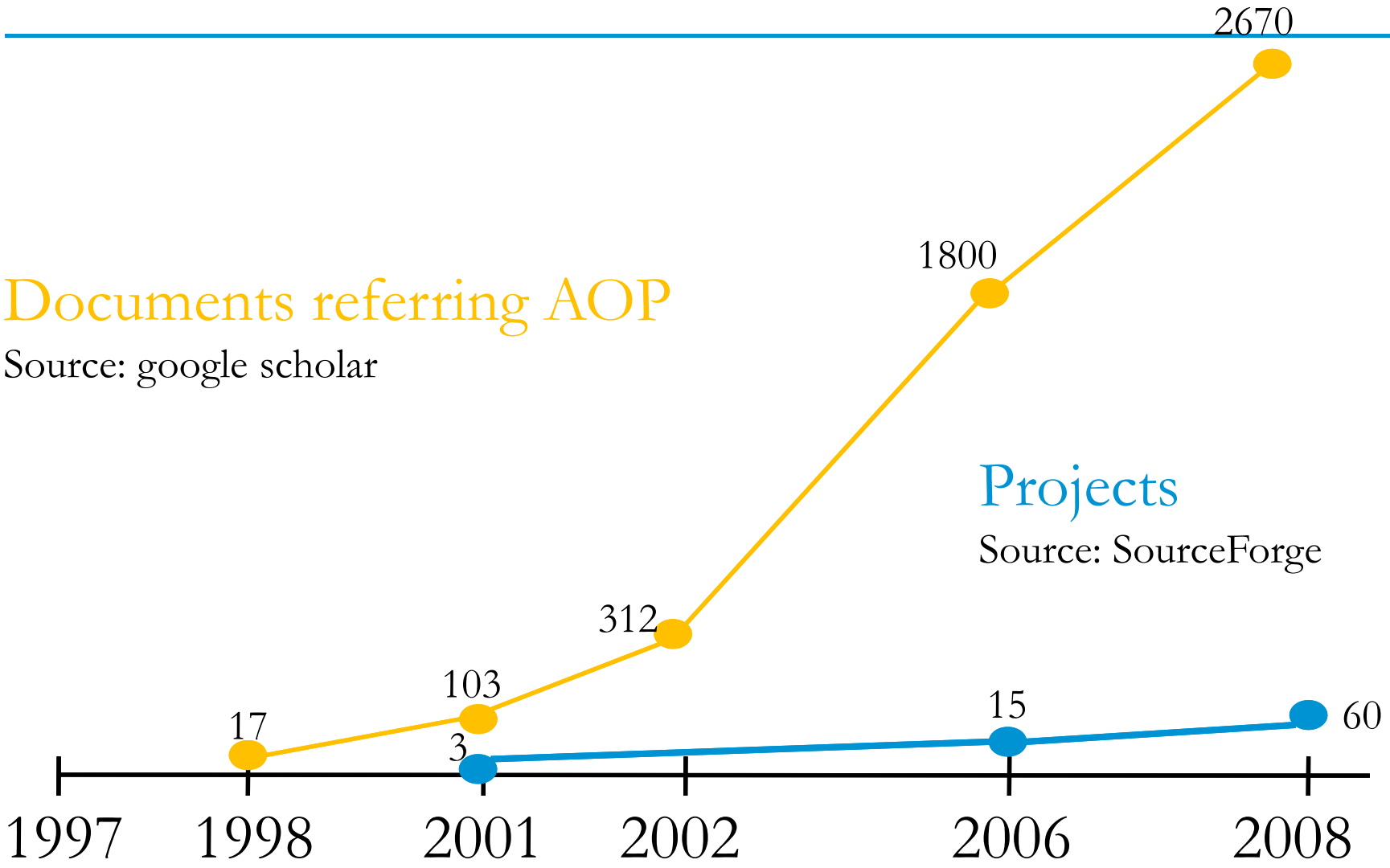
AOP in the last decade



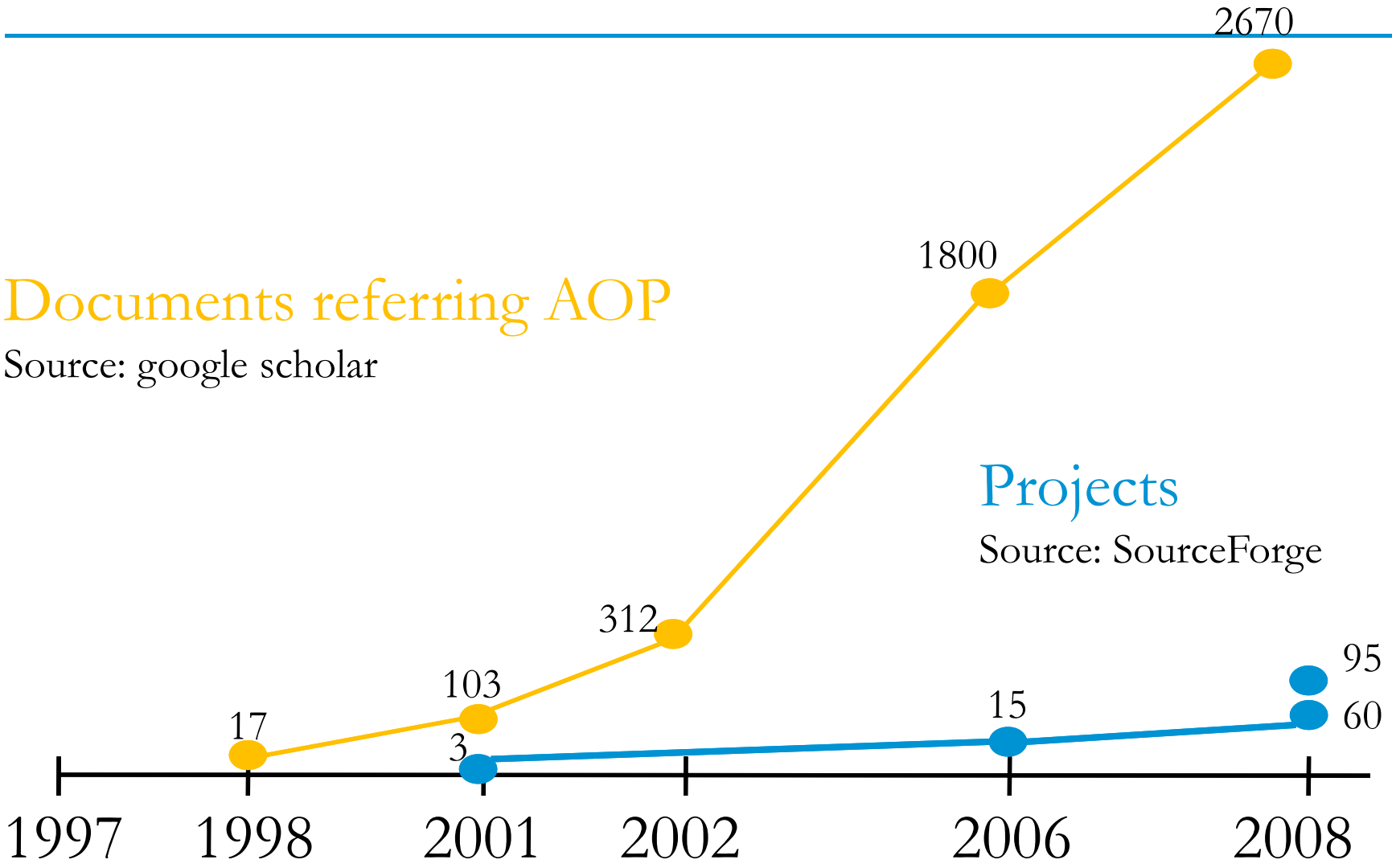
AOP in the last decade



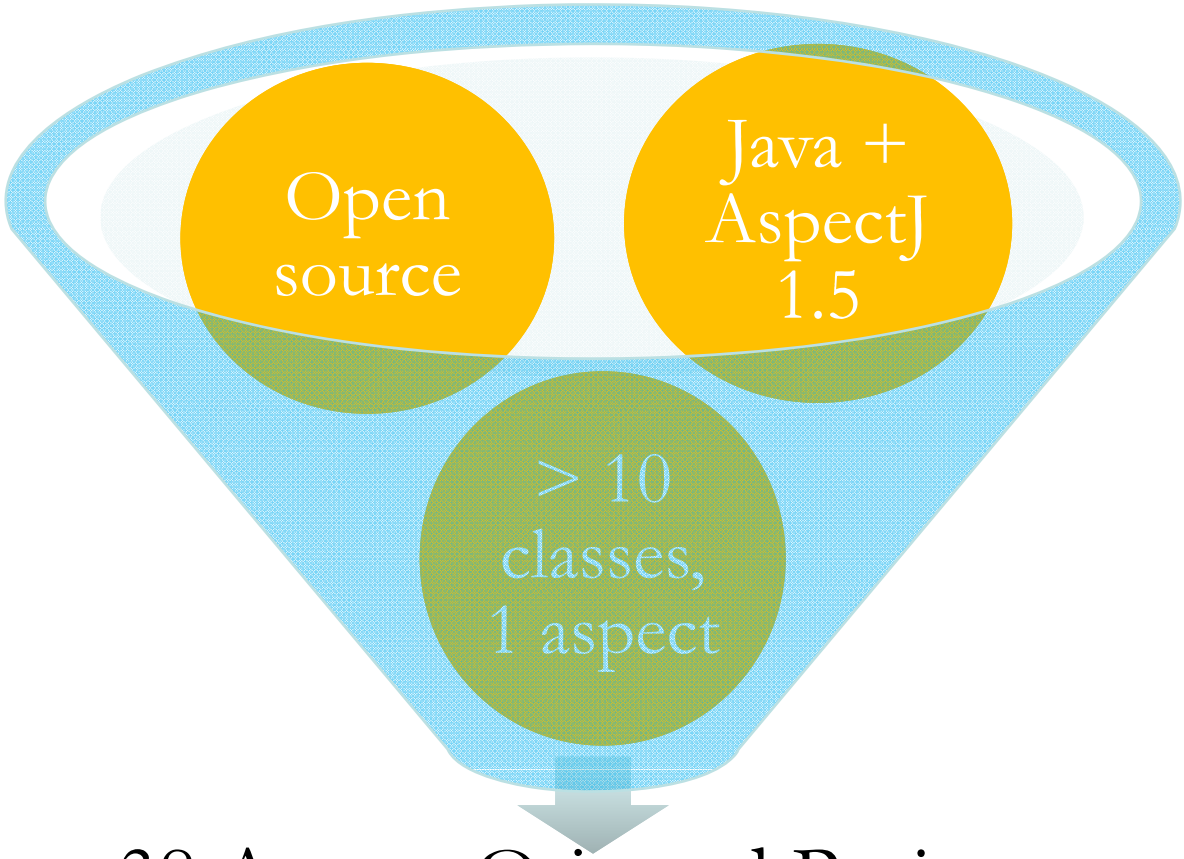
AOP in the last decade



AOP in the last decade

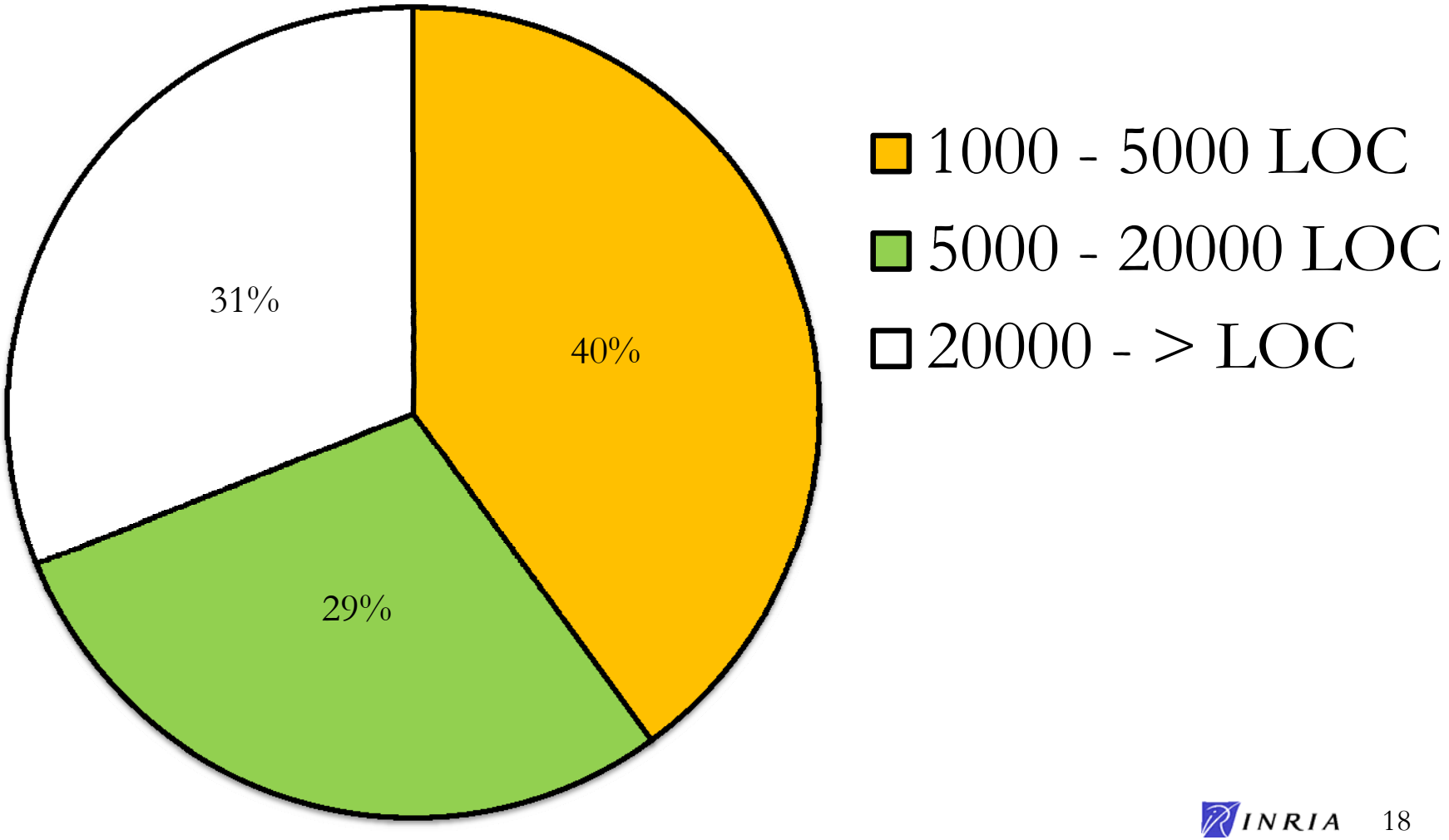


Experimental data selection

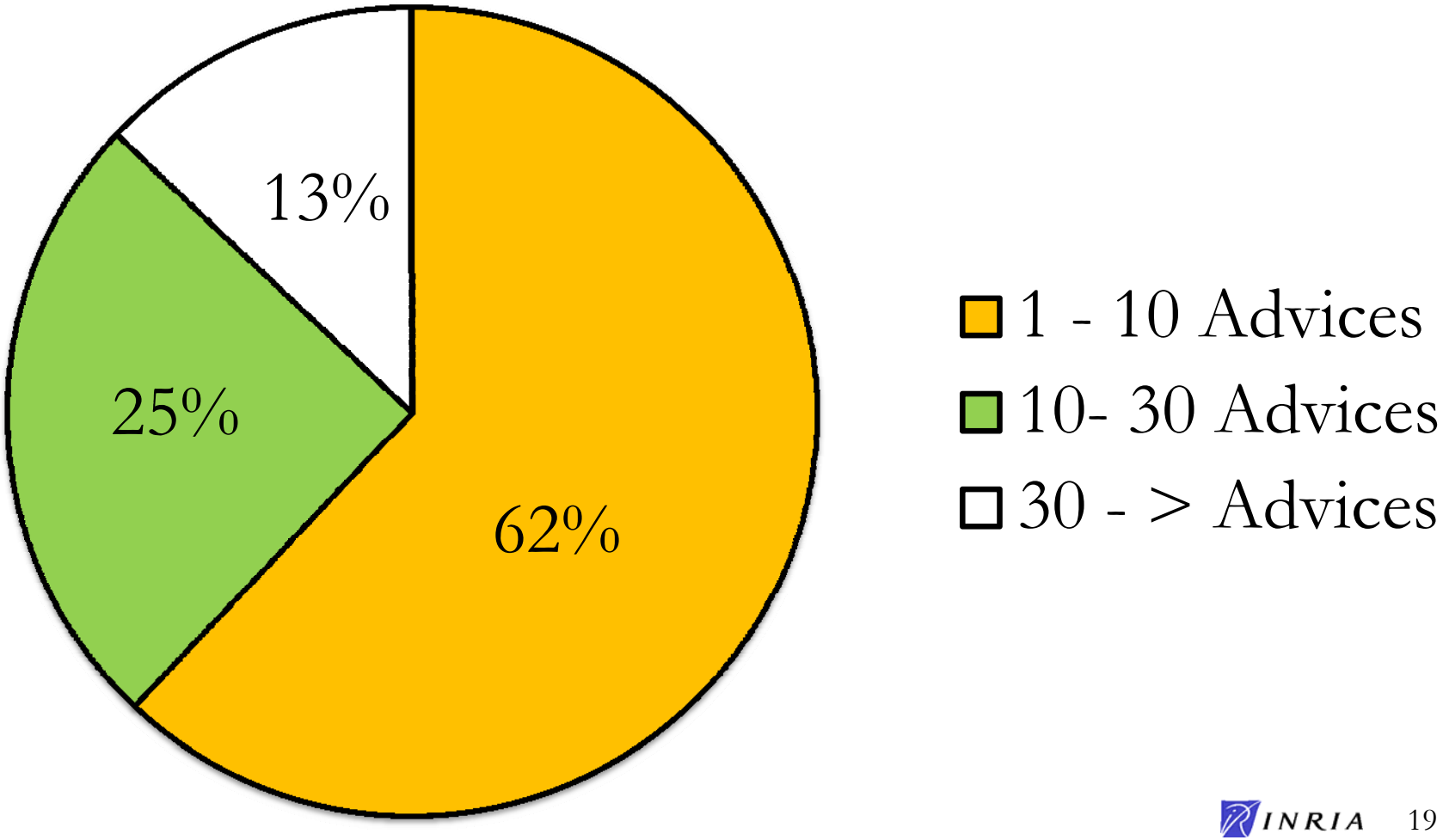


38 Aspect-Oriented Projects
(1116 – 80818 LOC)

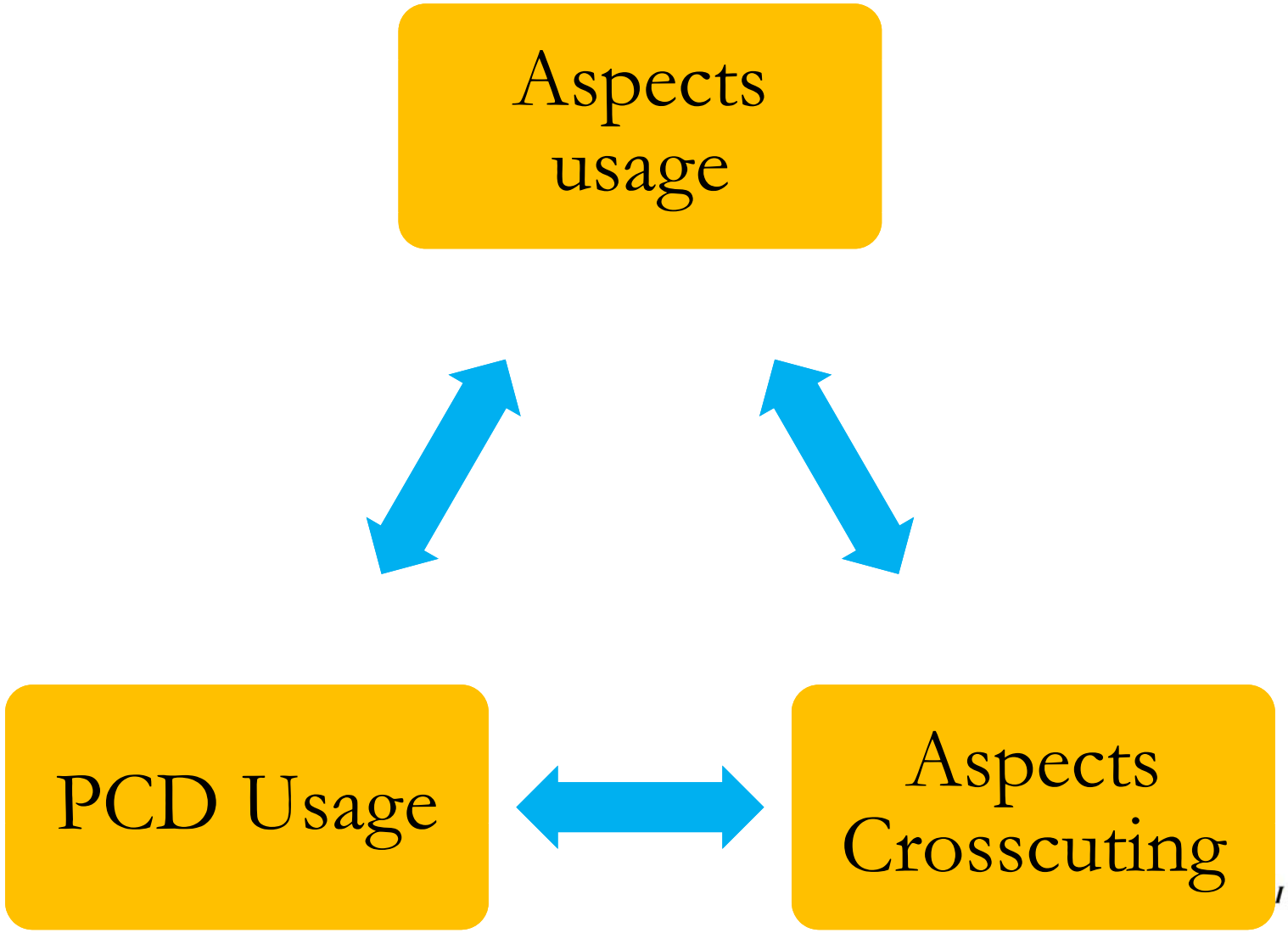
Experimental data - Projects



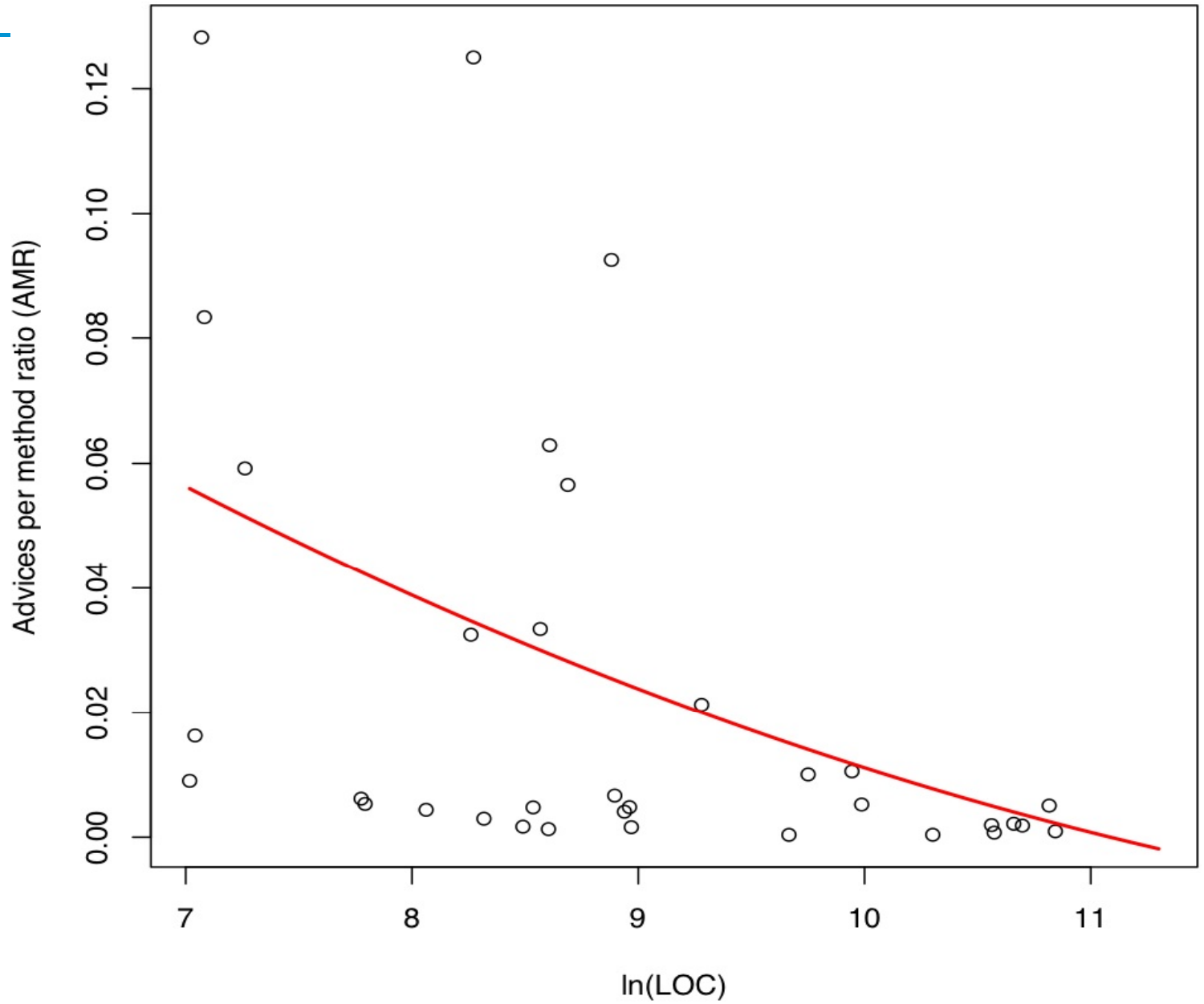
Experimental data



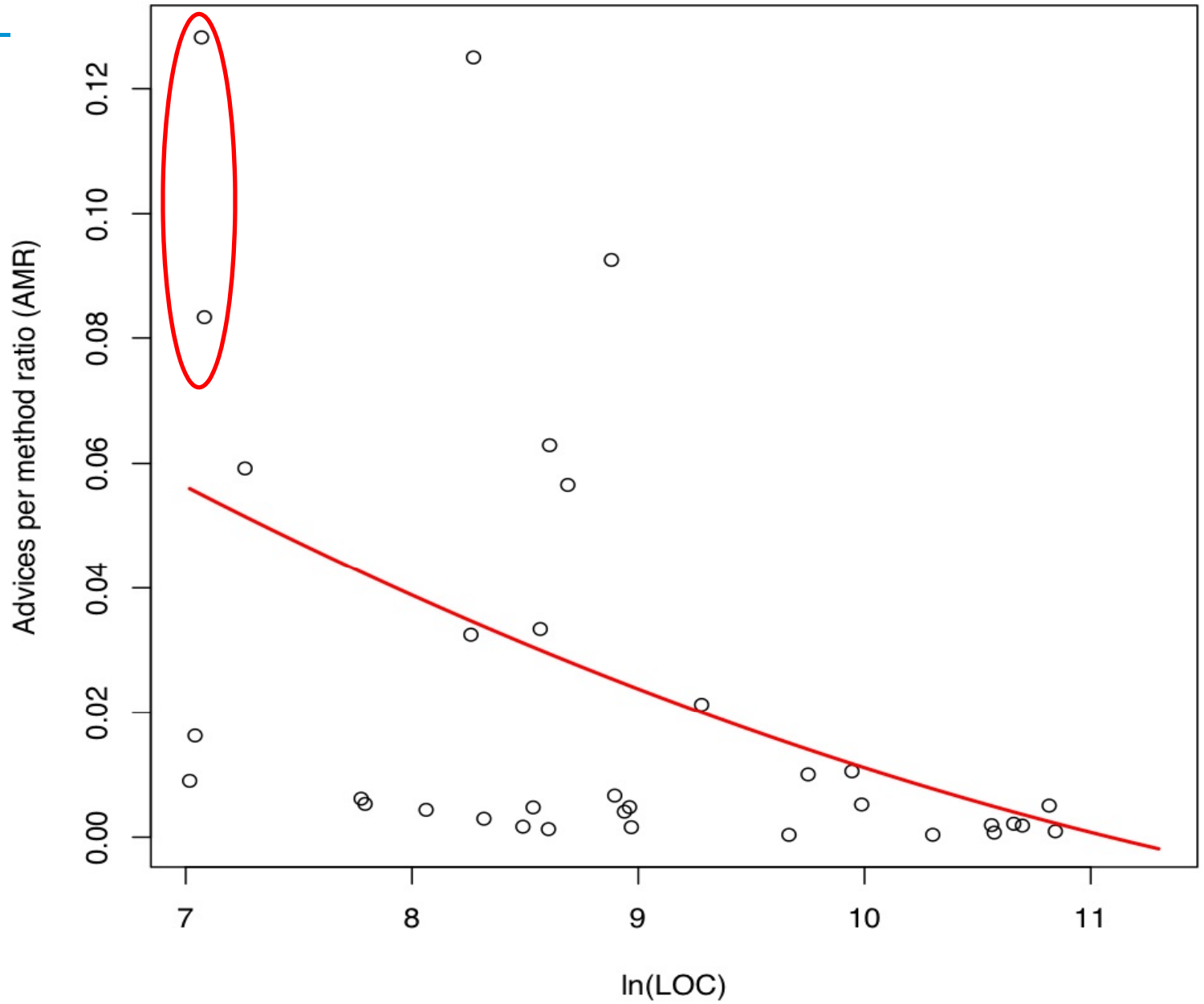
Research Inquiry



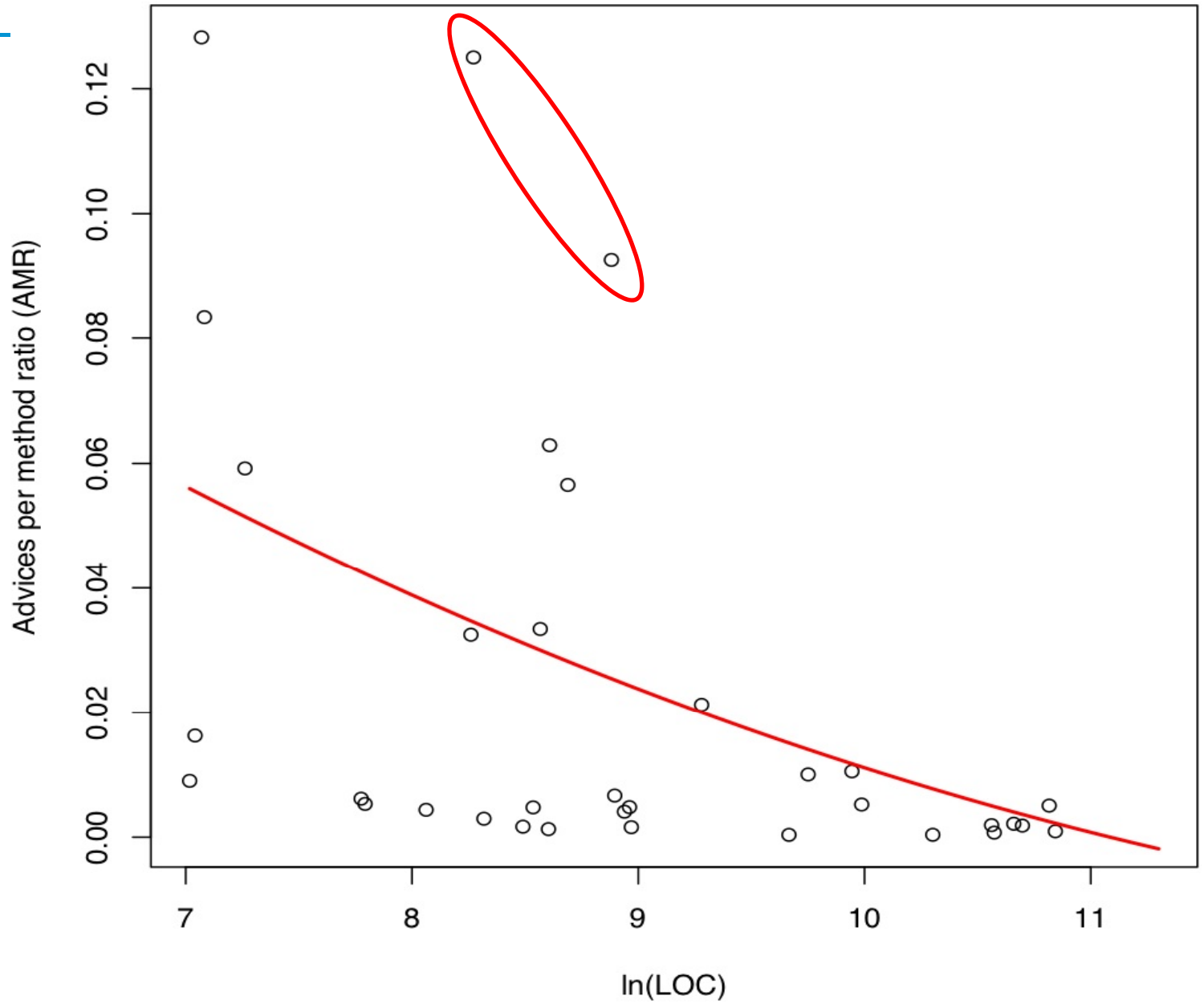
Aspect Usage



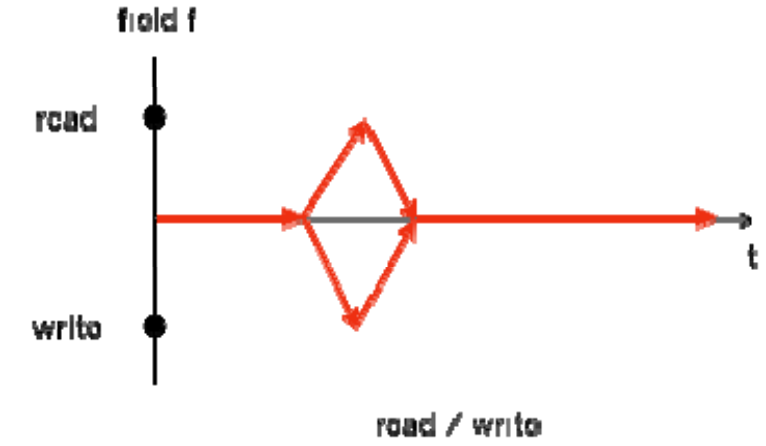
Aspect Usage

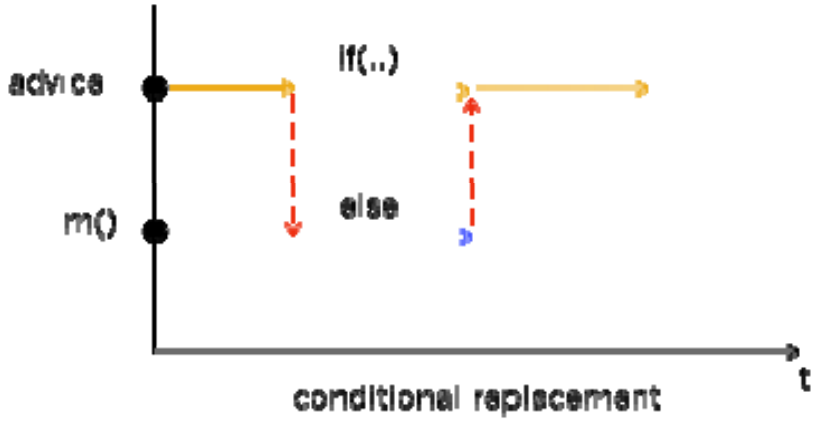


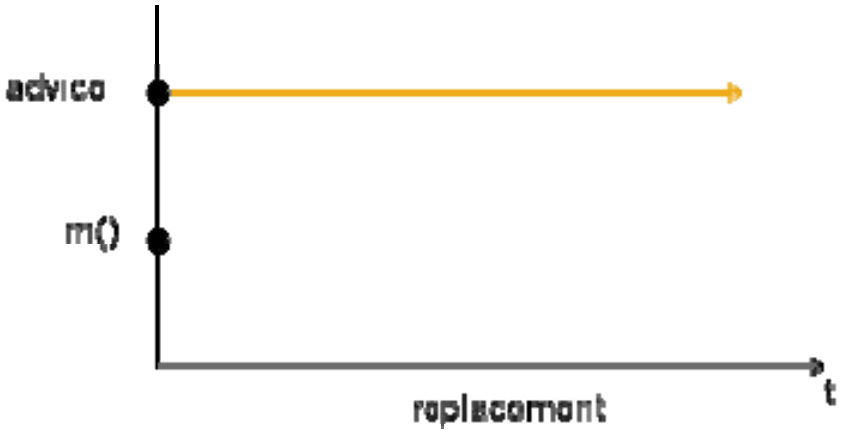
Aspect Usage

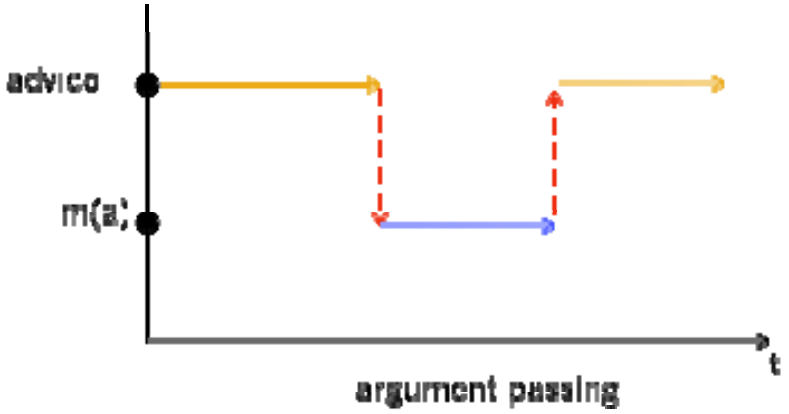


AspectJ interaction patterns

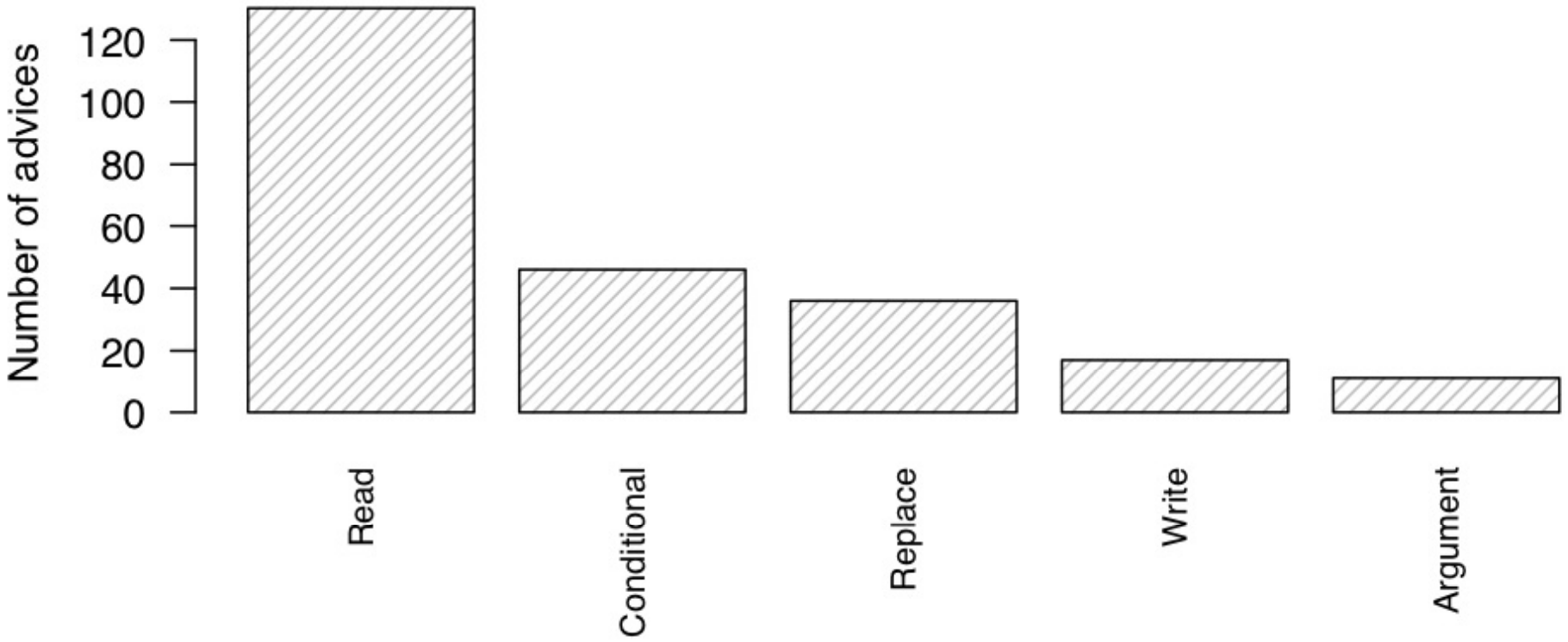




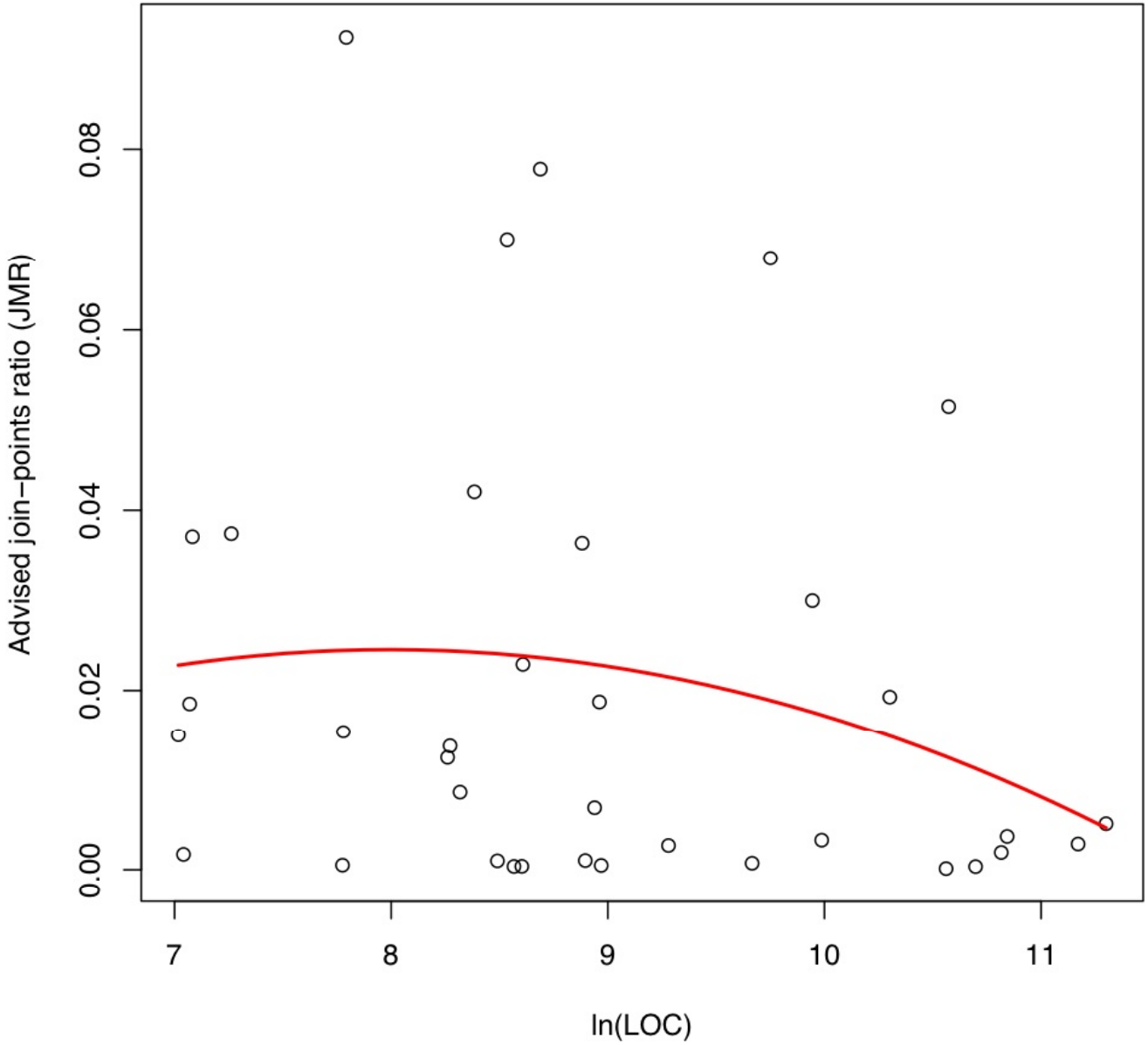




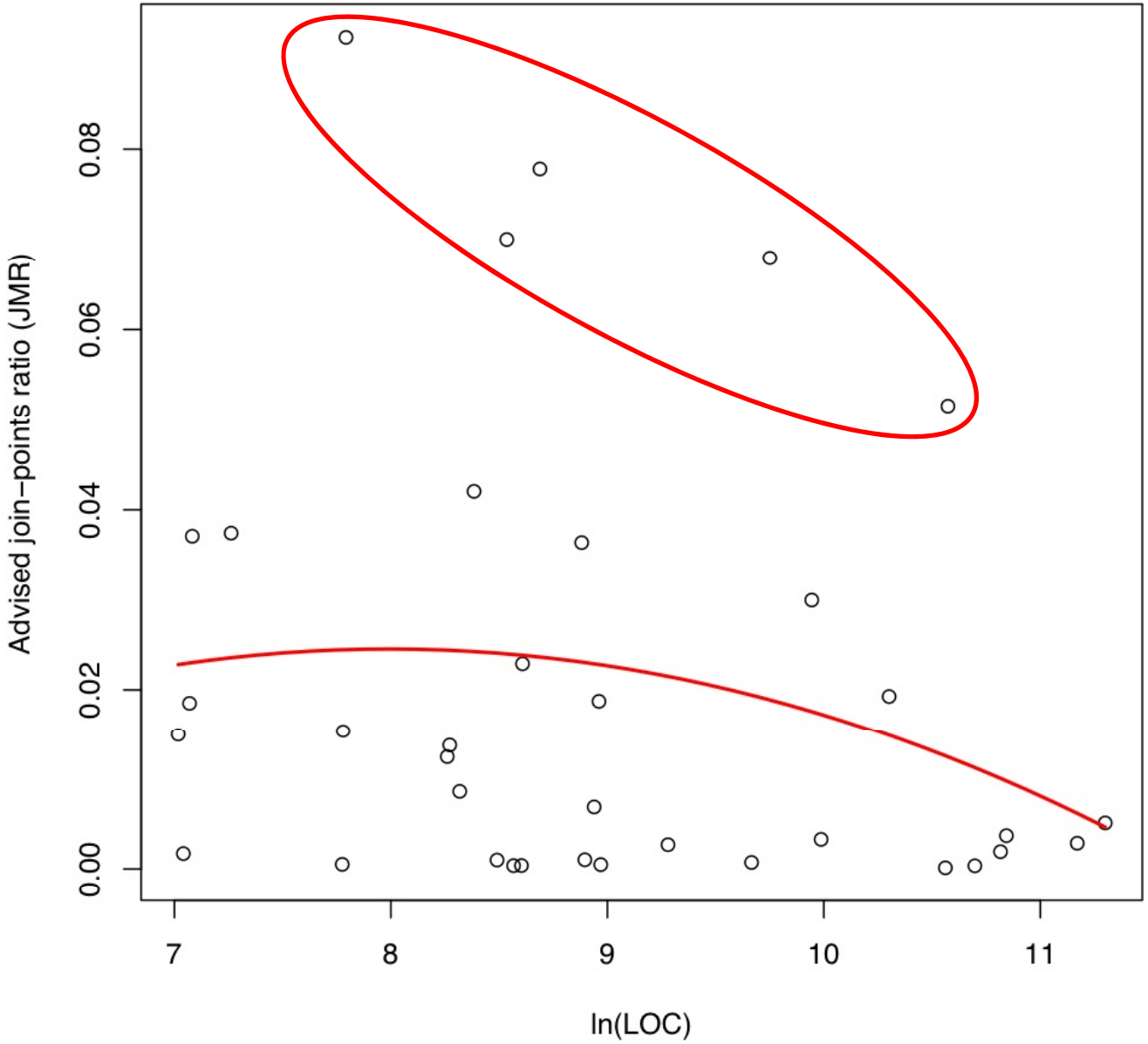
Aspect Usage



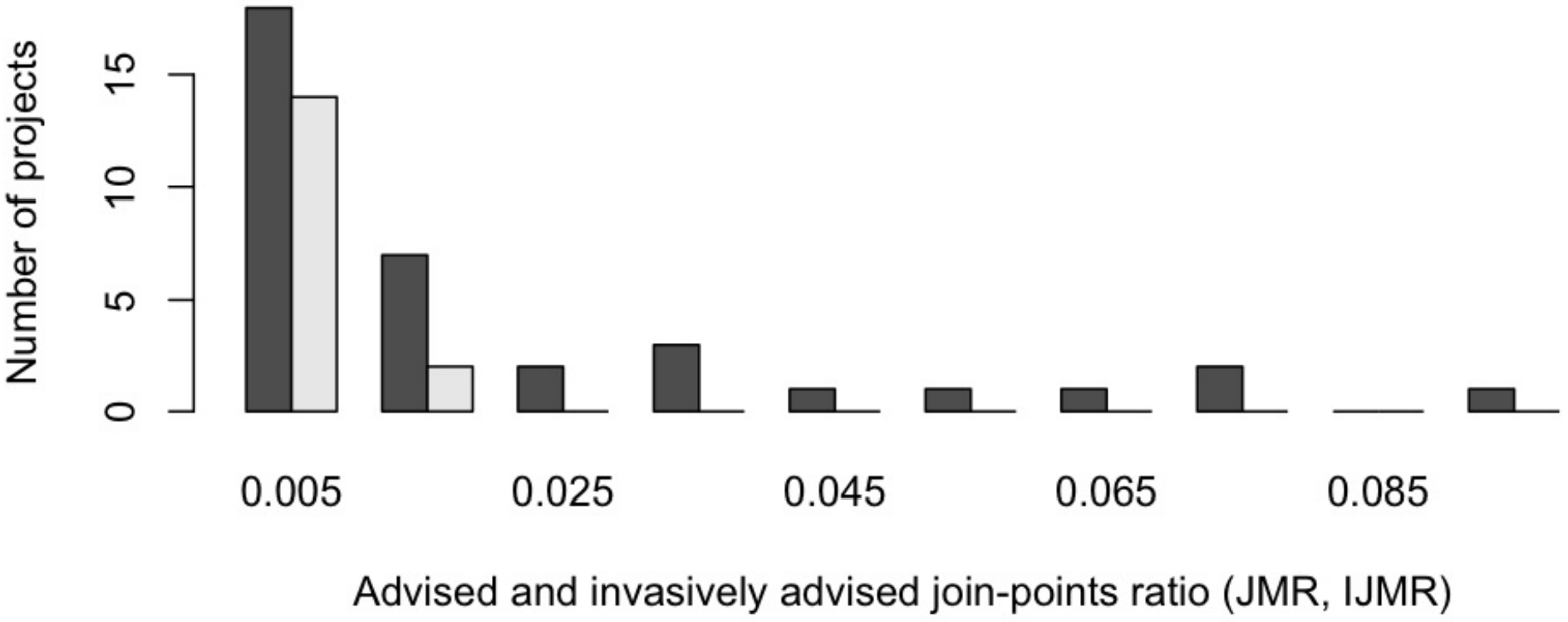
Aspect Crosscutting



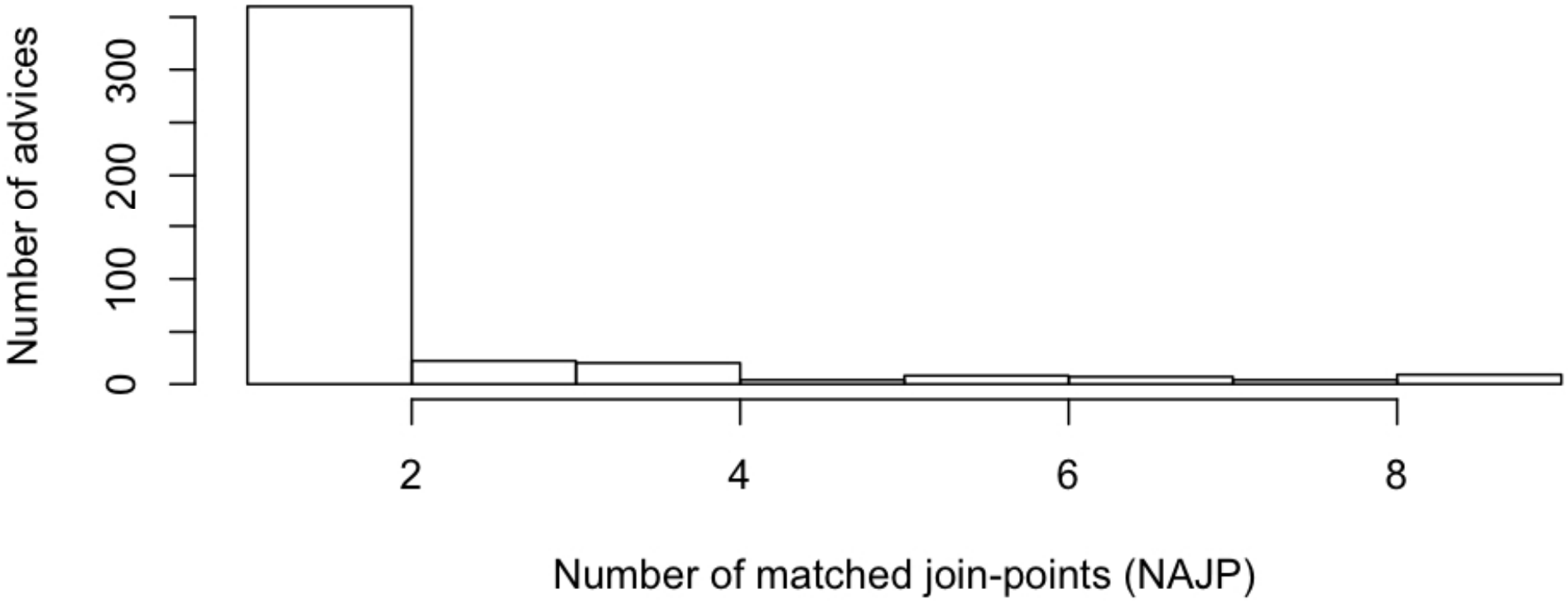
Aspect Crosscutting



Aspect Crosscutting



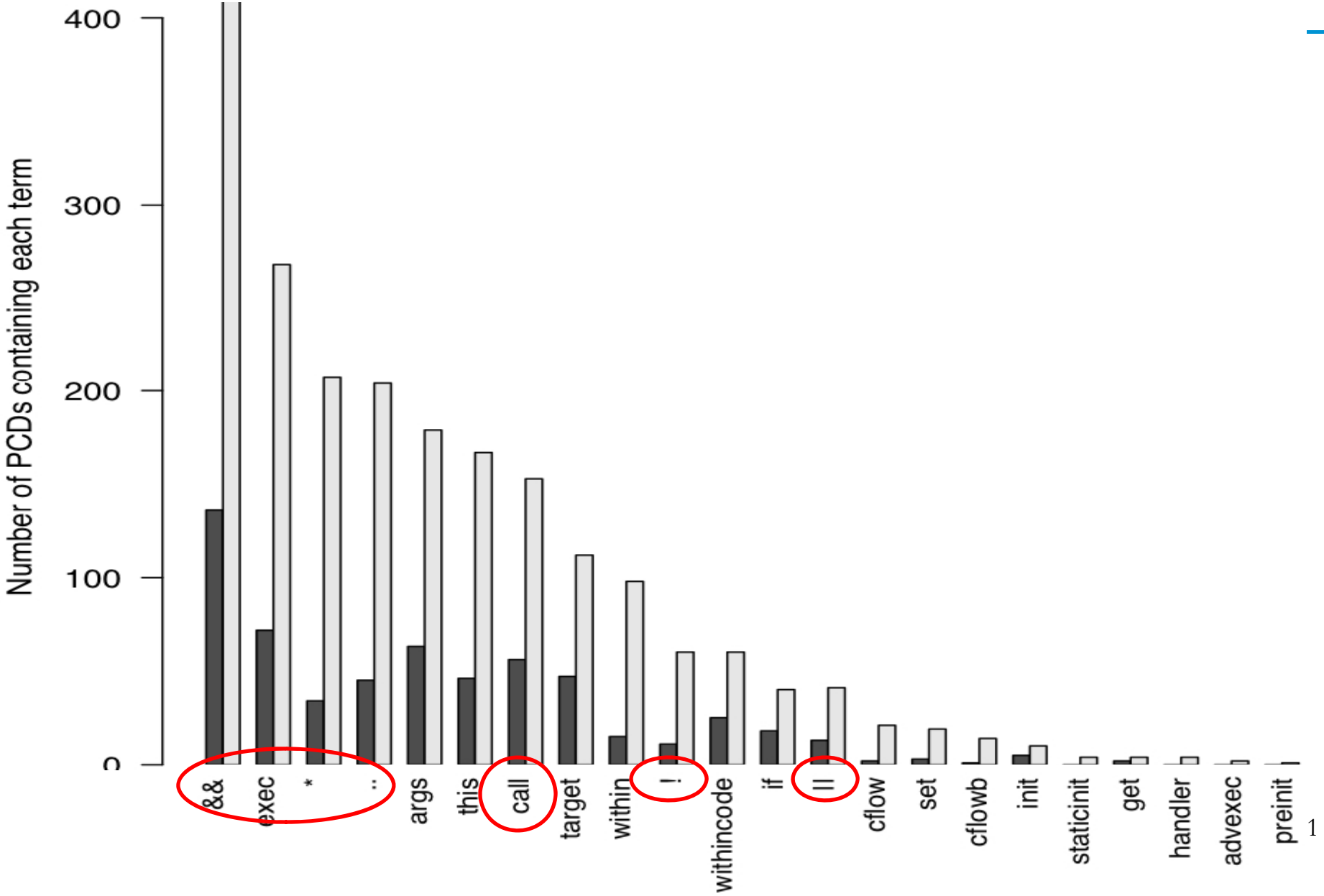
Aspects Crosscutting



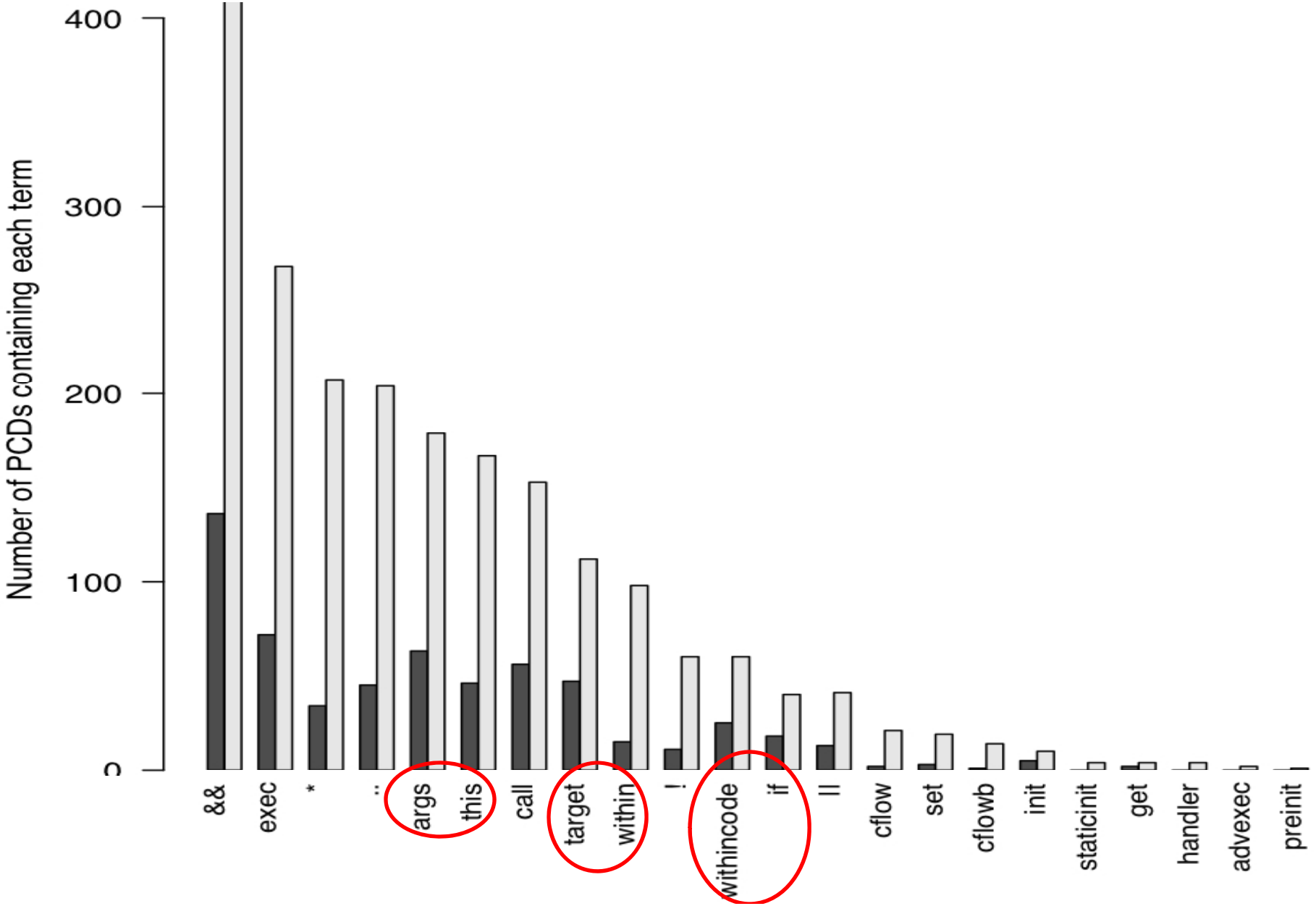
PCD Usage

Number of PCDs containing each term
400
300
200
100
0
&&
exec
*
..
args
this
call
target
within
!
withincode
if
||
cflow
set
cflowb
init
staticinit
get
handler
advexec
preinit

PCD Usage


Number of PCDs containing each term
400
300
200
100
0
&&
exec
*
..
args
this
call
target
within
!
withincode
if
||
cflow
set
cflowb
init
staticinit
get
handler
advexec
preinit

PCD Usage


Number of PCDs containing each term
400
300
200
100
0
&&
exec
*
..
args
this
call
target
within
!
withincode
if
||
cflow
set
cflowb
init
staticinit
get
handler
advexec
preinit

Conclusions

- Few advices at few joinpoints
- Careful interactions with base program
- Technical cross-cutting concerns are encapsulated in aspects