Università
della
Svizzera
italiana

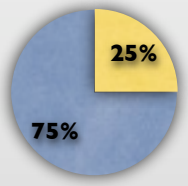
**Facoltà
di scienze
informatiche**

Spyware-ridden software development

Romain Robbes
Michele Lanza

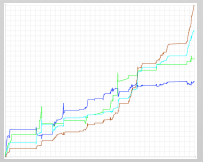
ESUG
2006

SpyWare is a research prototype built for my Ph.D.



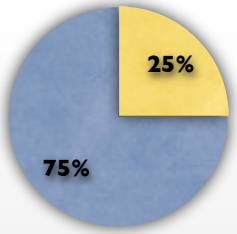
Context

What is SpyWare?



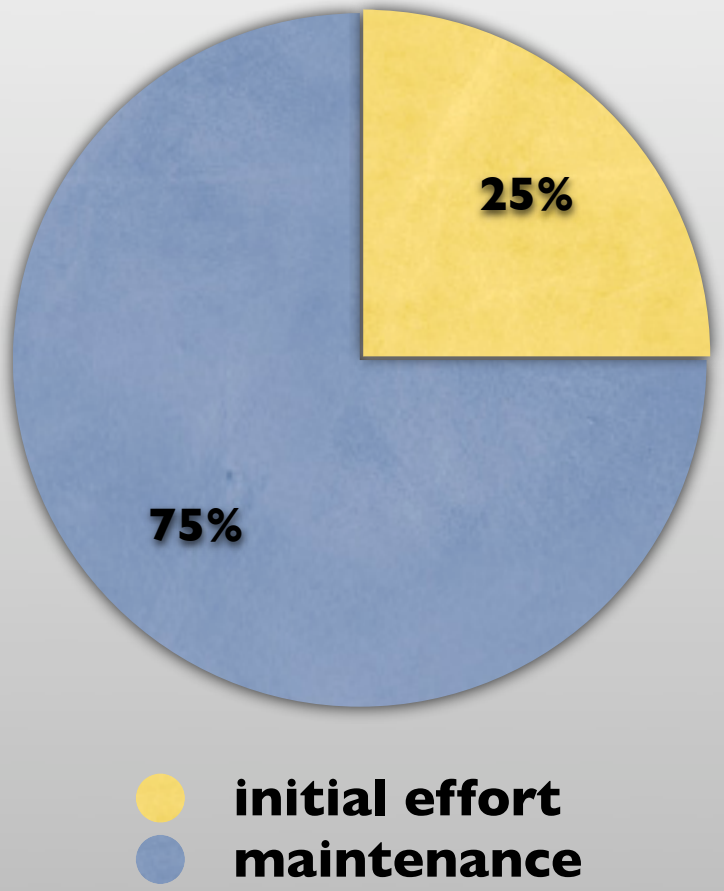
Results & demo

Further possibilities

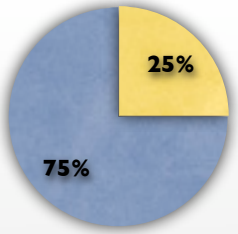


More than 3/4 of the cost of software is maintenance*

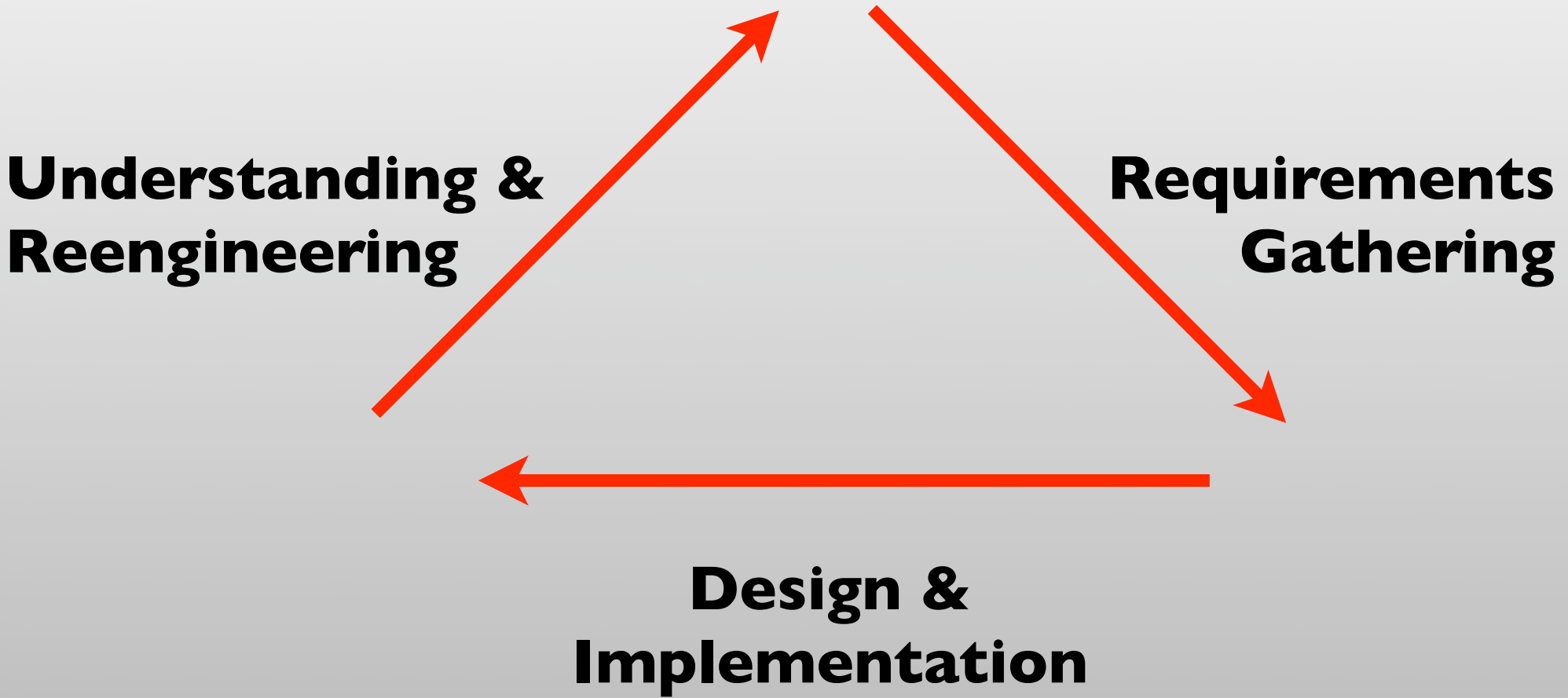
"A program that is used in a real-world environment must change, or become progressively less useful in that environment. "
– Lehman's laws

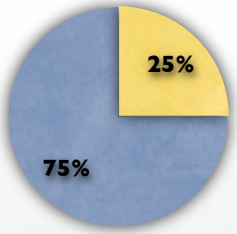


* from: http://www.cs.jyu.fi/~koskinen/smcosts.htm

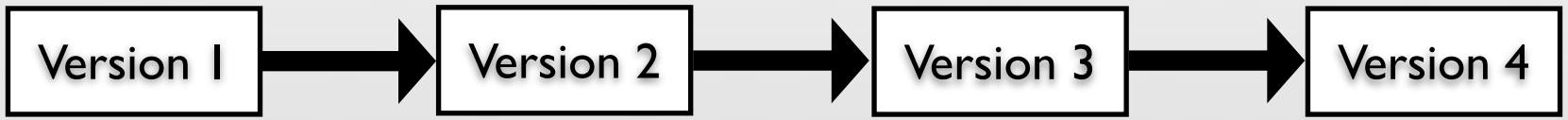


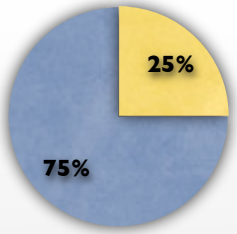
The lifecycle of software



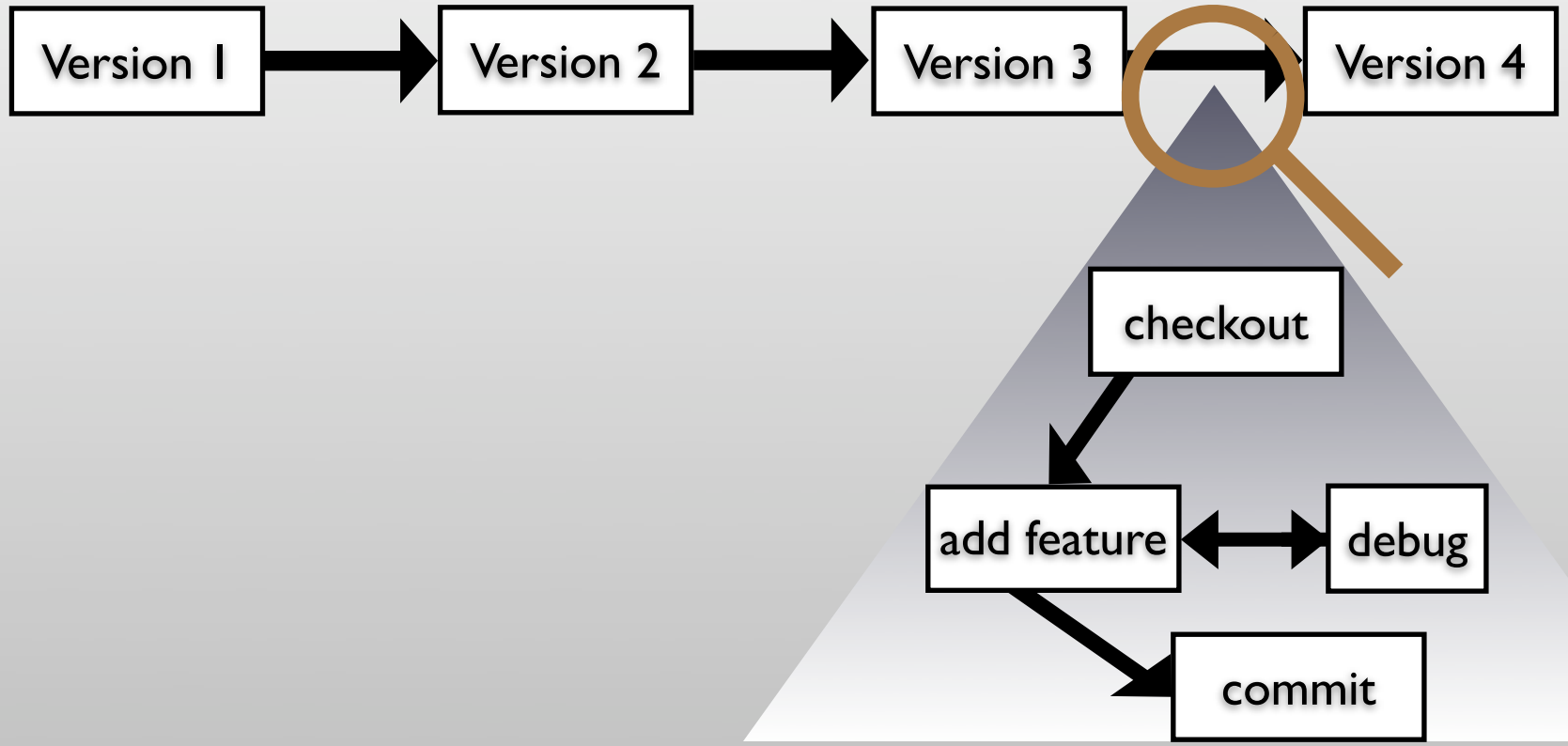


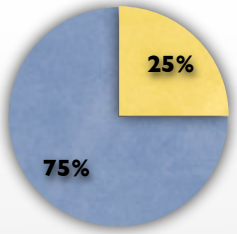
Software development is incremental





Software development is incremental



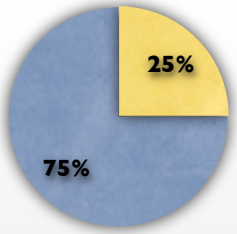


Software development is incremental

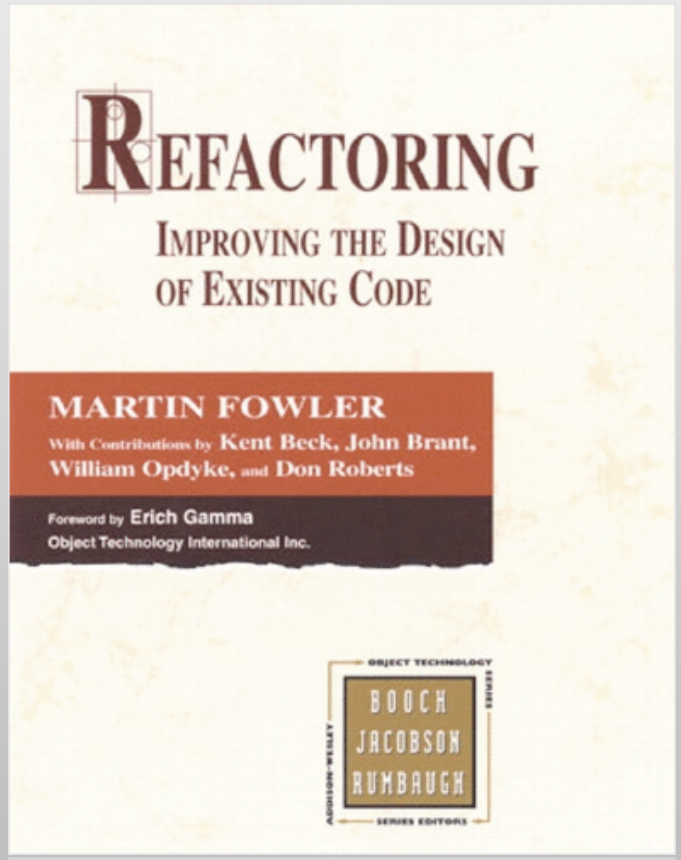
Version 1 → Version 2 → Version 3 → Version 4

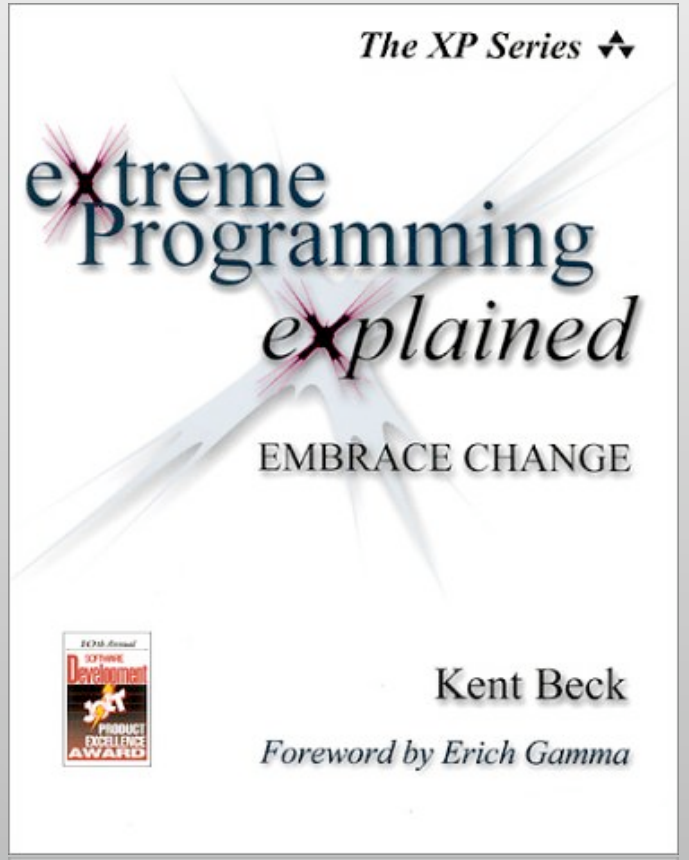
add class
add method
add method

checkout
add feature
debug
commit

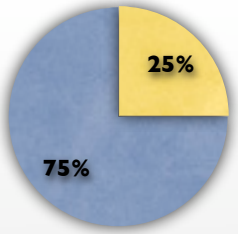


Languages and methodologies target this problem

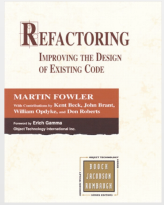




The holy trinity?

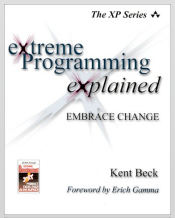


The change rate of the software increases



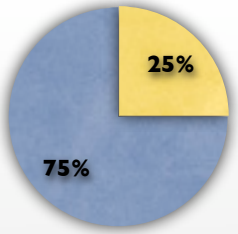
It is easier to lose reference points

Evolvability and understandability are at odds



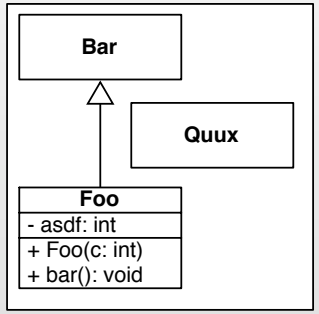
XP states that the software is in maintenance 100% of the time ...

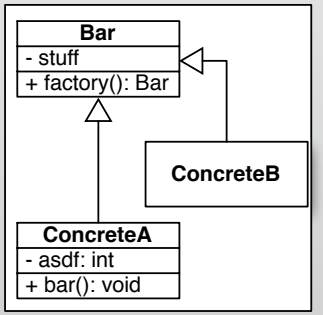
Are conventional reengineering tools adapted to agile development?

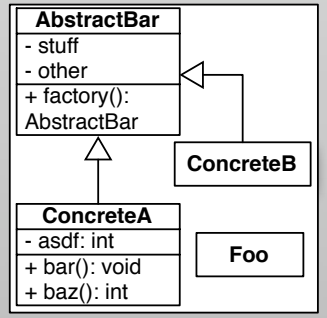


Software evolution analysis helps reengineering

Versioning system

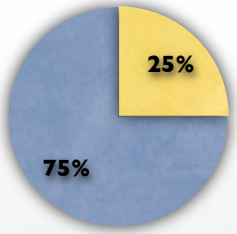




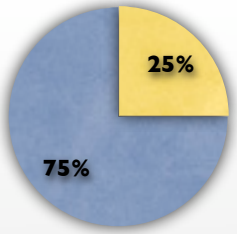


Metrics & trends

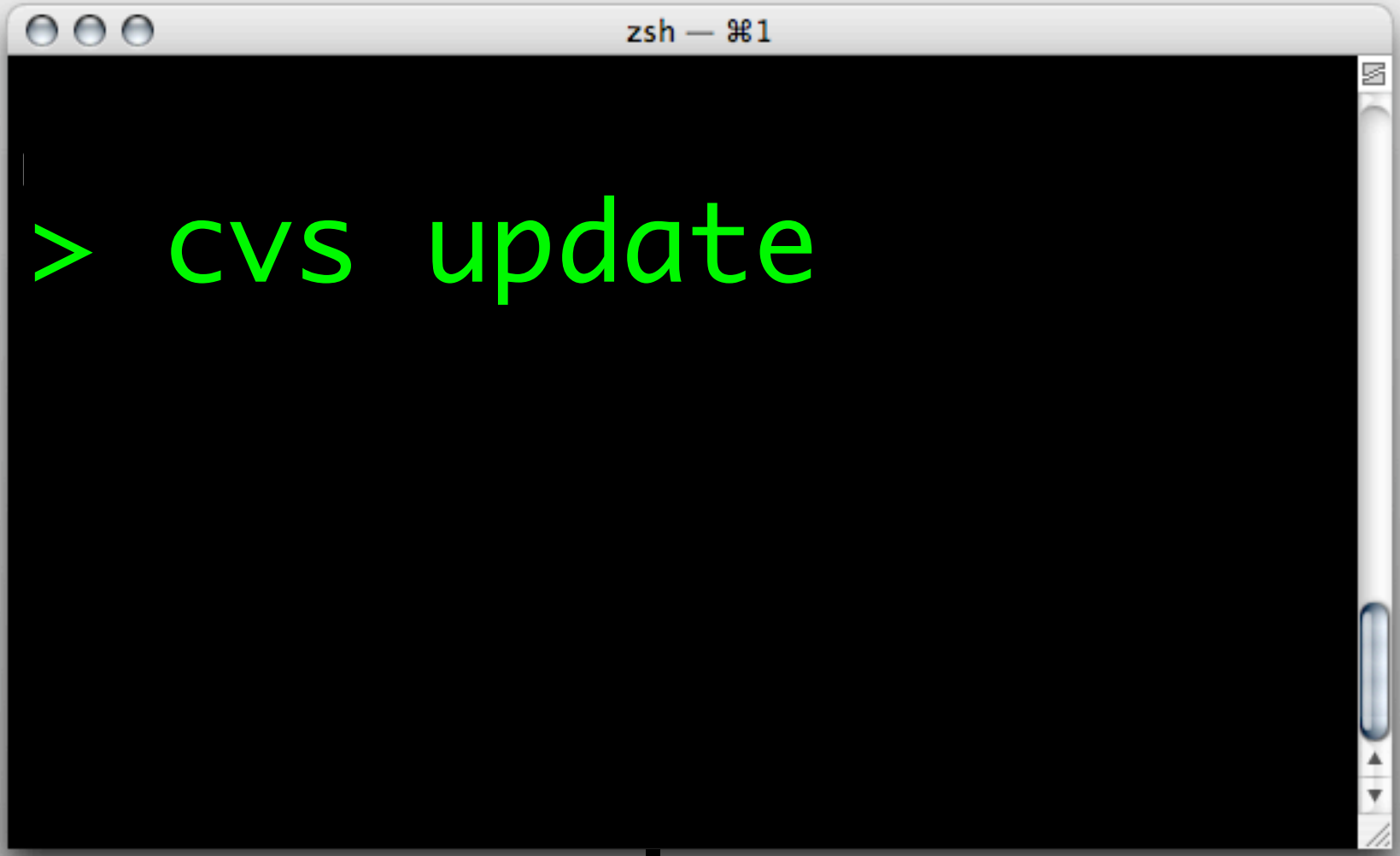
History holds useful information

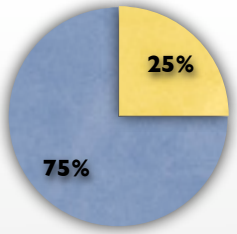


Versioning systems lose information

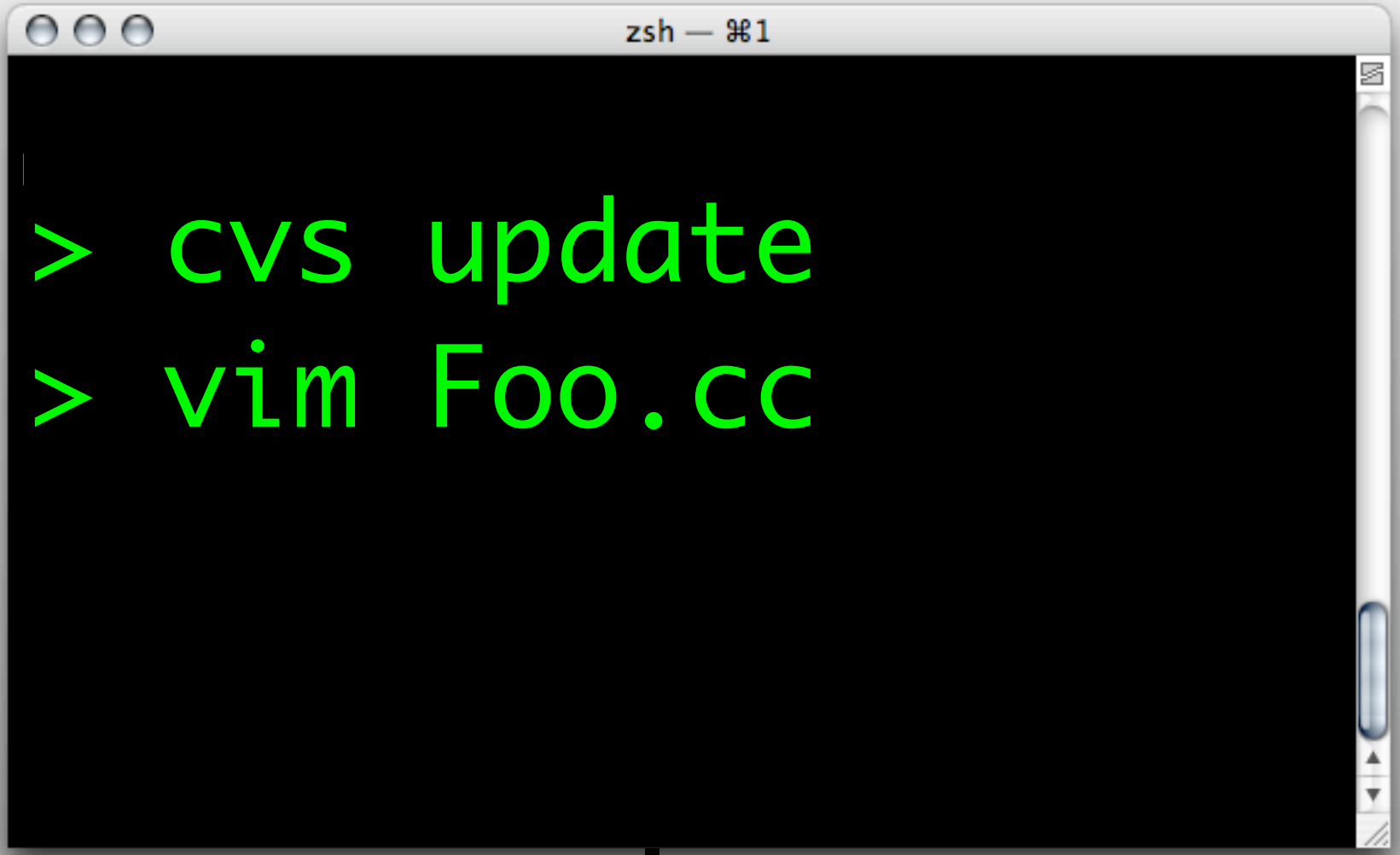


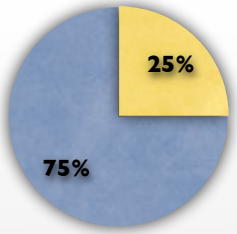
Versioning systems lose information



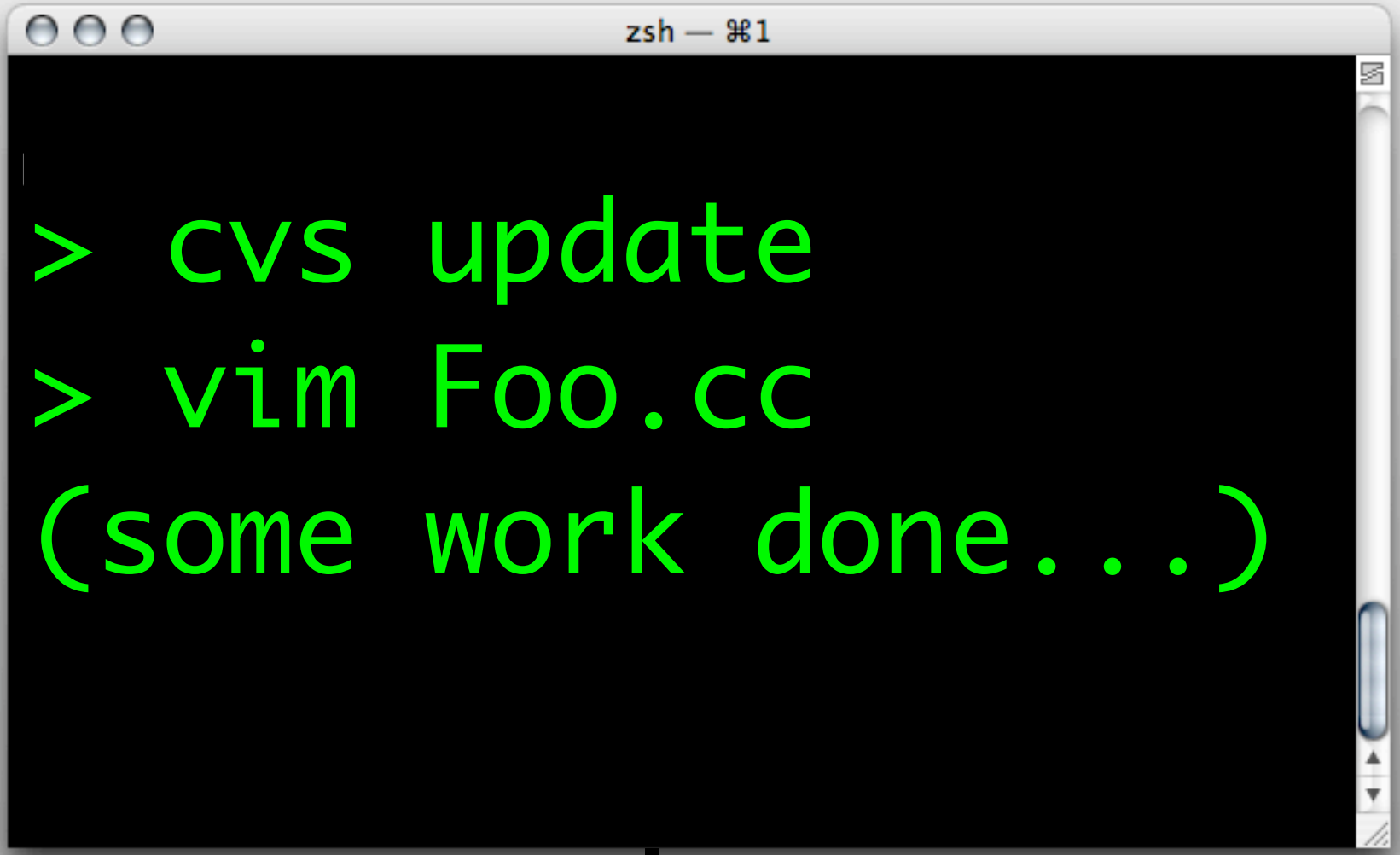


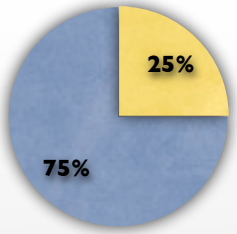
Versioning systems lose information



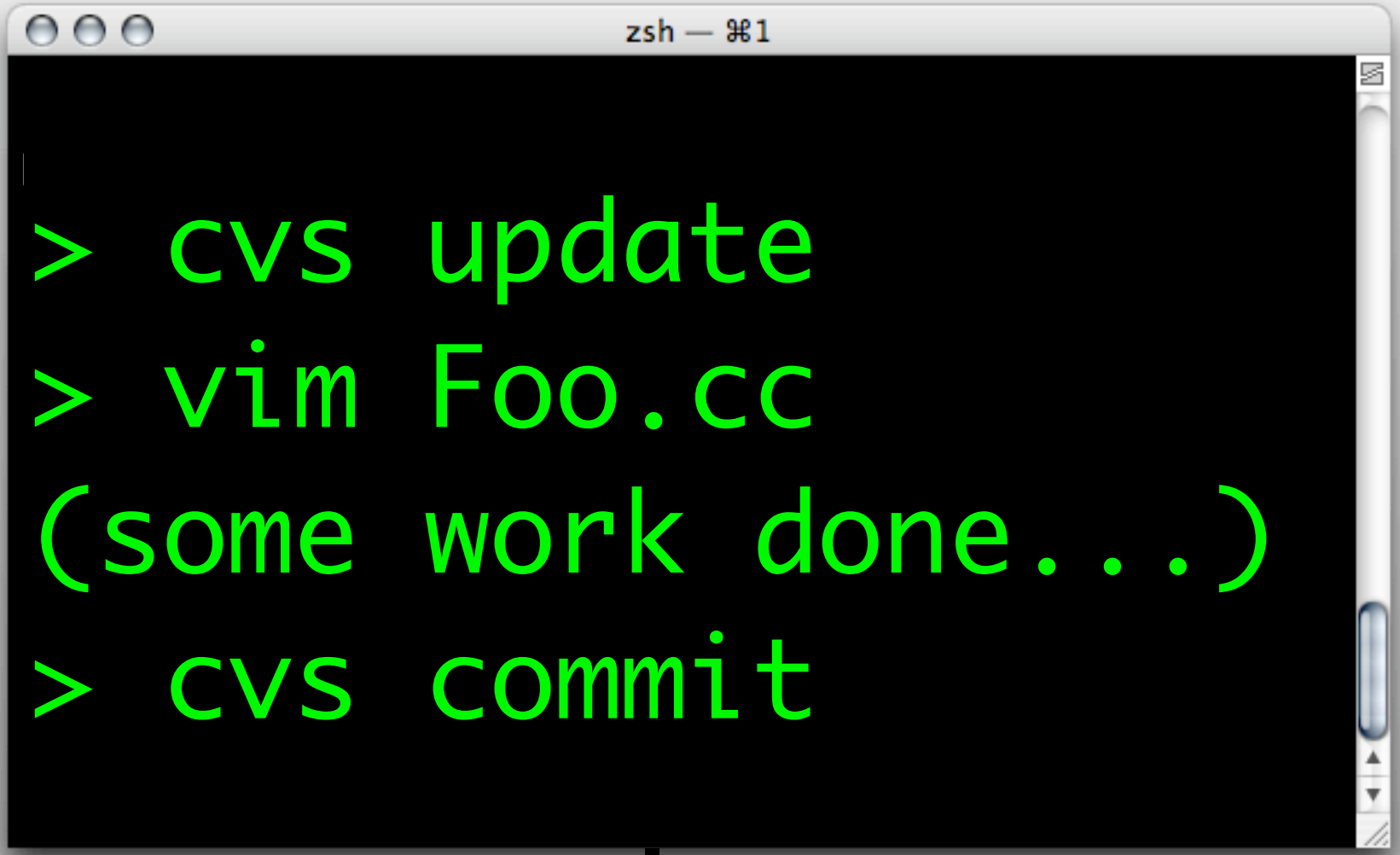


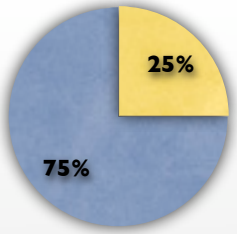
Versioning systems lose information





Versioning systems lose information

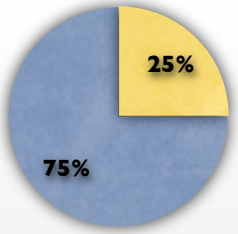




Versioning systems lose information

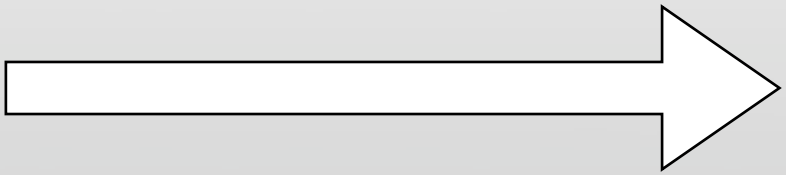
```
zsh — ⌘1

> cvs update
> vim Foo.cc
(some work done...)
> cvs commit
```

?

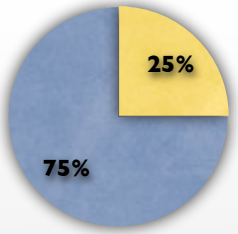


Our paranoid programmer saves every 5 minutes

```
class Foo {
   public int x;
   public int y;

   public doFoo() {
       blah.blah(blah);
       z = x + y;
       blu = blu * 2;
       t = blurg(z);
       bli[t] = blu;
       return t;
   }

   public quux() {
      return y + 4;
   }

   public asdf() {
      return x * 8 + y;
   }
}


 f = new Foo();
 f.doFoo();
 print f.x + f.y;
```



```
class Foo {
   private int x;
   private int y;

   public getX()  { return x; }
   public setX(newX) { x = newX; }

   public getY() { return y; }
   public setY(newY) { y = newY; }

   public baz() {
       blah.blah(blah);
       z = getX() + getY();
       return bar(z);
   }

     public quux() {
      return getY() + 4;
   }

   public asdf() {
      return getX() * 8 + getY();
   }

    private bar(z) {
       blu = blu * 2;
       t = blurg(z);
       bli[t] = blu;
       return t;
   }


}


 f = new Foo();
 f.baz();
 print f.getX() + f.getY();
```

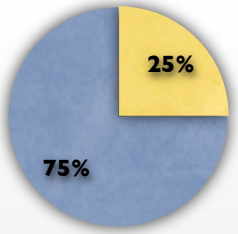


Our paranoid programmer saves every 5 minutes

```
class Foo {
   public int x;
   public int y;

   public doFoo() {
      blah.blah(blah);
      z = x + y;
      blu = blu * 2;
      t = blurg(z);
      bli[t] = blu;
      return t;
   }

   public quux() {
      return y + 4;
   }

   public asdf() {
      return x * 8 + y;
   }
}


 f = new Foo();
 f.doFoo();
 print f.x + f.y;
```

1.Extract method

```
class Foo {
   private int x;
   private int y;

   public getX()  { return x; }
   public setX(newX) { x = newX; }

   public getY() { return y; }
   public setY(newY) { y = newY; }

   public baz() {
      blah.blah(blah);
      z = getX() + getY();
      return bar(z);
   }

   public quux() {
      return getY() + 4;
   }

   public asdf() {
      return getX() * 8 + getY();
   }

   private bar(z) {
      blu = blu * 2;
      t = blurg(z);
      bli[t] = blu;
      return t;
   }

}


 f = new Foo();
 f.baz();
 print f.getX() + f.getY();
```

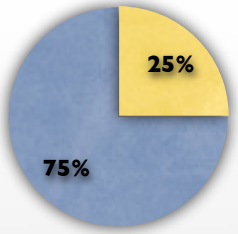


Our paranoid programmer saves every 5 minutes

```
class Foo {
   public int x;
   public int y;

   public doFoo() {
       blah.blah(blah);
       z = x + y;
       blu = blu * 2;
       t = blurg(z);
       bli[t] = blu;
       return t;
   }

   public quux() {
      return y + 4;
   }

   public asdf() {
      return x * 8 + y;
   }
}


 f = new Foo();
 f.doFoo();
 print f.x + f.y;
```

1.Extract method

```
class Foo {
   private int x;
   private int y;

   public getX()  { return x; }
   public setX(newX) { x = newX; }

   public getY() { return y; }
   public setY(newY) { y = newY; }

   public baz() {
       blah.blah(blah);
       z = getX() + getY();
       return bar(z);
   }

     public quux() {
      return getY() + 4;
   }

   public asdf() {
      return getX() * 8 + getY();
   }

    private bar(z) {
       blu = blu * 2;
       t = blurg(z);
       bli[t] = blu;
       return t;
   }


}


 f = new Foo();
 f.baz();
 print f.getX() + f.getY();
```



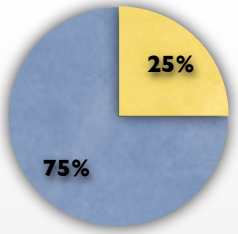
Our paranoid programmer saves every 5 minutes

```
class Foo {
   public int x;
   public int y;

   public doFoo() {
      blah.blah(blah);
      z = x + y;
      blu = blu * 2;
      t = blurg(z);
      bli[t] = blu;
      return t;
   }

   public quux() {
      return y + 4;
   }

   public asdf() {
      return x * 8 + y;
   }
}


f = new Foo();
f.doFoo();
print f.x + f.y;
```

1.Extract method

2.Rename method

```
class Foo {
   private int x;
   private int y;

   public getX()  { return x; }
   public setX(newX) { x = newX; }

   public getY() { return y; }
   public setY(newY) { y = newY; }

   public baz() {
      blah.blah(blah);
      z = getX() + getY();
      return bar(z);
   }

     public quux() {
      return getY() + 4;
   }

   public asdf() {
      return getX() * 8 + getY();
   }

    private bar(z) {
      blu = blu * 2;
      t = blurg(z);
      bli[t] = blu;
      return t;
   }


}


f = new Foo();
f.baz();
print f.getX() + f.getY();
```



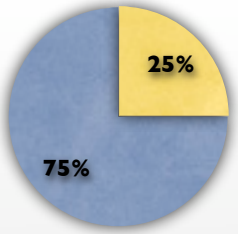
Our paranoid programmer saves every 5 minutes

```
class Foo {
   public int x;
   public int y;

   public doFoo() {
       blah.blah(blah);
       z = x + y;
       blu = blu * 2;
       t = blurg(z);
       bli[t] = blu;
       return t;
   }

   public quux() {
      return y + 4;
   }

   public asdf() {
      return x * 8 + y;
   }
}


 f = new Foo();
 f.doFoo();
 print f.x + f.y;
```

1.Extract method

2.Rename method

```
class Foo {
   private int x;
   private int y;

   public getX()  { return x; }
   public setX(newX) { x = newX; }

   public getY() { return y; }
   public setY(newY) { y = newY; }

   public baz() {
       blah.blah(blah);
       z = getX() + getY();
       return bar(z);
   }

     public quux() {
      return getY() + 4;
   }

   public asdf() {
      return getX() * 8 + getY();
   }

    private bar(z) {
       blu = blu * 2;
       t = blurg(z);
       bli[t] = blu;
       return t;
   }


}


 f = new Foo();
 f.baz();
 print f.getX() + f.getY();
```

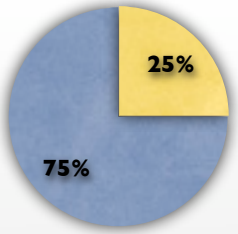


Our paranoid programmer saves every 5 minutes

```
class Foo {
   public int x;
   public int y;

   public doFoo() {
      blah.blah(blah);
      z = x + y;
      blu = blu * 2;
      t = blurg(z);
      bli[t] = blu;
      return t;
   }

   public quux() {
      return y + 4;
   }

   public asdf() {
      return x * 8 + y;
   }
}


f = new Foo();
f.doFoo();
print f.x + f.y;
```

1. Extract method
2. Rename method
3. Create accessors

```
class Foo {
   private int x;
   private int y;

   public getX()  { return x; }
   public setX(newX) { x = newX; }

   public getY() { return y; }
   public setY(newY) { y = newY; }

   public baz() {
      blah.blah(blah);
      z = getX() + getY();
      return bar(z);
   }

   public quux() {
      return getY() + 4;
   }

   public asdf() {
      return getX() * 8 + getY();
   }

   private bar(z) {
      blu = blu * 2;
      t = blurg(z);
      bli[t] = blu;
      return t;
   }


}


f = new Foo();
f.baz();
print f.getX() + f.getY();
```

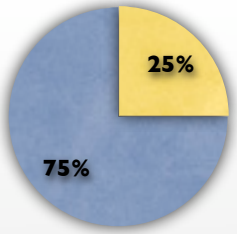


Our paranoid programmer saves every 5 minutes

```
class Foo {
   public int x;
   public int y;

   public doFoo() {
       blah.blah(blah);
       z = x + y;
       blu = blu * 2;
       t = blurg(z);
       bli[t] = blu;
       return t;
   }

   public quux() {
      return y + 4;
   }

   public asdf() {
      return x * 8 + y;
   }
}


 f = new Foo();
 f.doFoo();
 print f.x + f.y;
```

1. Extract method
2. Rename method
3. Create accessors

```
class Foo {
   private int x;
   private int y;

   public getX()  { return x; }
   public setX(newX) { x = newX; }

   public getY() { return y; }
   public setY(newY) { y = newY; }

   public baz() {
       blah.blah(blah);
       z = getX() + getY();
       return bar(z);
   }

     public quux() {
      return getY() + 4;
   }

   public asdf() {
      return getX() * 8 + getY();
   }

    private bar(z) {
       blu = blu * 2;
       t = blurg(z);
       bli[t] = blu;
       return t;
   }


}


 f = new Foo();
 f.baz();
 print f.getX() + f.getY();
```



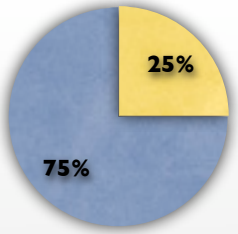
Our paranoid programmer saves every 5 minutes

```
class Foo {
   public int x;
   public int y;

   public doFoo() {
       blah.blah(blah);
       z = x + y;
       blu = blu * 2;
       t = blurg(z);
       bli[t] = blu;
       return t;
   }

   public quux() {
       return y + 4;
   }

   public asdf() {
       return x * 8 + y;
   }
}


 f = new Foo();
f.doFoo();
print f.x + f.y;
```

CVS: +18 / -11

1. Extract method
2. Rename method
3. Create accessors

```
class Foo {
   private int x;
   private int y;

   public getX()  { return x; }
   public setX(newX) { x = newX; }

   public getY() { return y; }
   public setY(newY) { y = newY; }

   public baz() {
       blah.blah(blah);
       z = getX() + getY();
       return bar(z);
   }

    public quux() {
       return getY() + 4;
   }

   public asdf() {
       return getX() * 8 + getY();
   }

    private bar(z) {
       blu = blu * 2;
       t = blurg(z);
       bli[t] = blu;
       return t;
   }


}


 f = new Foo();
f.baz();
print f.getX() + f.getY();
```



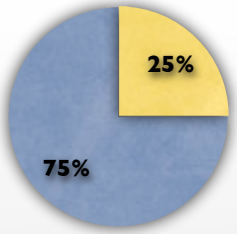
Our paranoid programmer saves every 5 minutes

```
class Foo {
   public int x;
   public int y;

   public doFoo() {
       blah.blah(blah);
       z = x + y;
       blu = blu * 2;
       t = blurg(z);
       bli[t] = blu;
       return t;
   }

   public quux() {
      return y + 4;
   }

   public asdf() {
      return x * 8 + y;
   }
}


 f = new Foo();
 f.doFoo();
 print f.x + f.y;
```

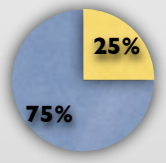
CVS: +18 / -11

1. Extract method
2. Rename method
3. Create accessors

```
class Foo {
   private int x;
   private int y;

   public getX()  { return x; }
   public setX(newX) { x = newX; }

   public getY() { return y; }
   public setY(newY) { y = newY; }

   public baz() {
       blah.blah(blah);
       z = getX() + getY();
       return bar(z);
   }

     public quux() {
      return getY() + 4;
   }

   public asdf() {
      return getX() * 8 + getY();
   }

    private bar(z) {
       blu = blu * 2;
       t = blurg(z);
       bli[t] = blu;
       return t;
   }


}


 f = new Foo();
 f.baz();
 print f.getX() + f.getY();
```



Recapitulation



Software evolution is hard but necessary

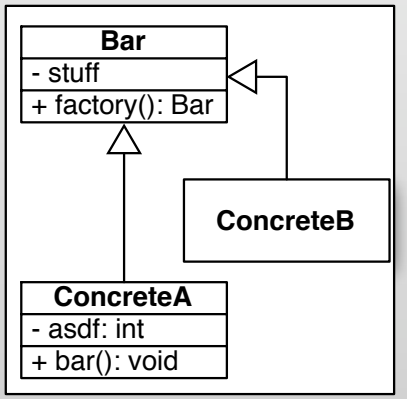
Agile practices make it change even faster

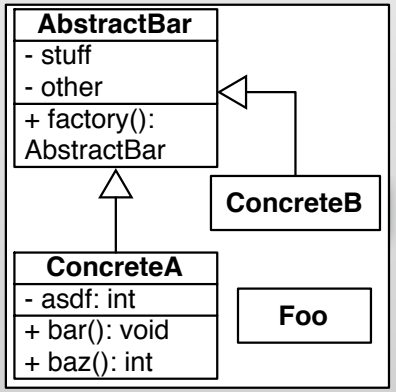
Versioning systems are inappropriate sources of information for SE tools

The principles behind SpyWare

1. **Model changes, not versions**
2. **Use the IDE, not the code repository**
3. **Provide tools in the IDE**

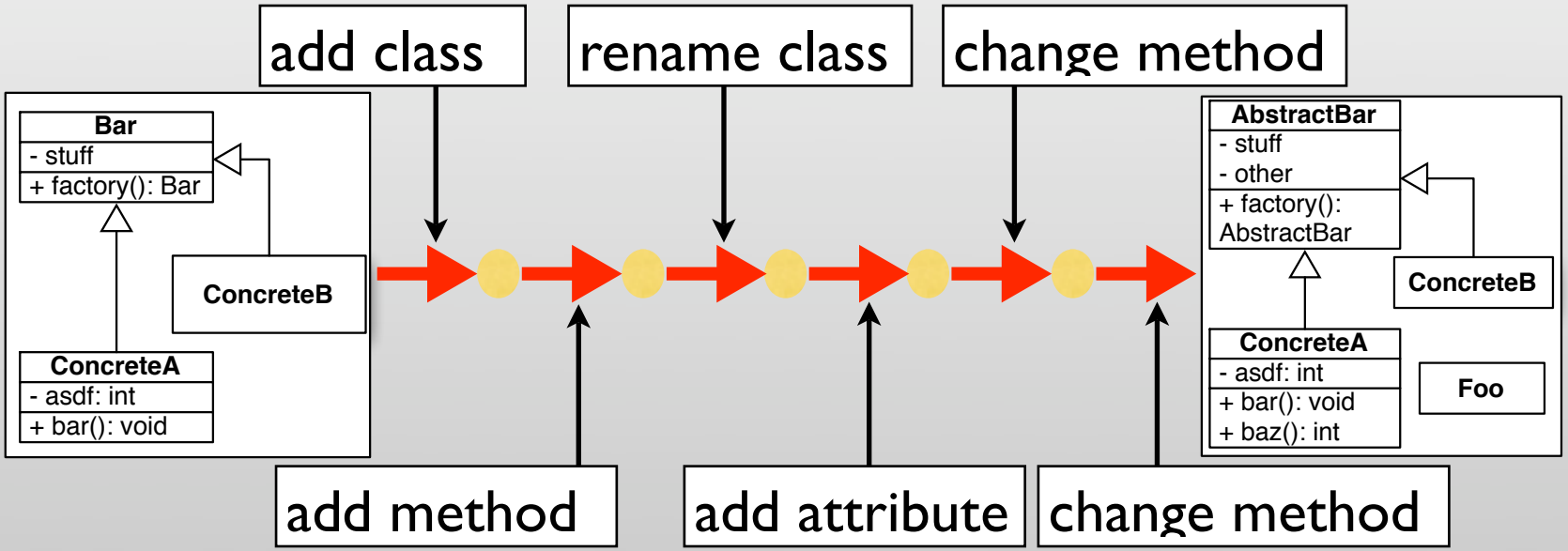
Modelling and capturing change information





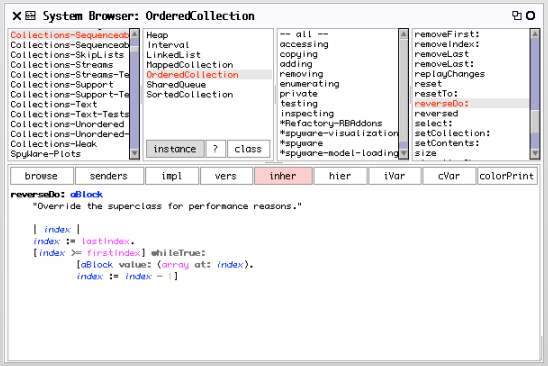
Rather than storing versions, we record the semantic actions of the programmer

Modelling and capturing change information



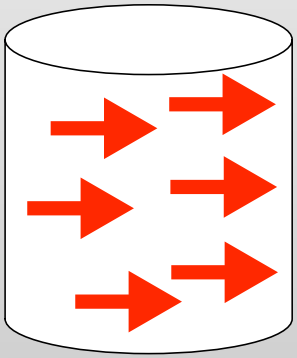
Rather than storing versions, we record the semantic actions of the programmer

The general architecture of SpyWare

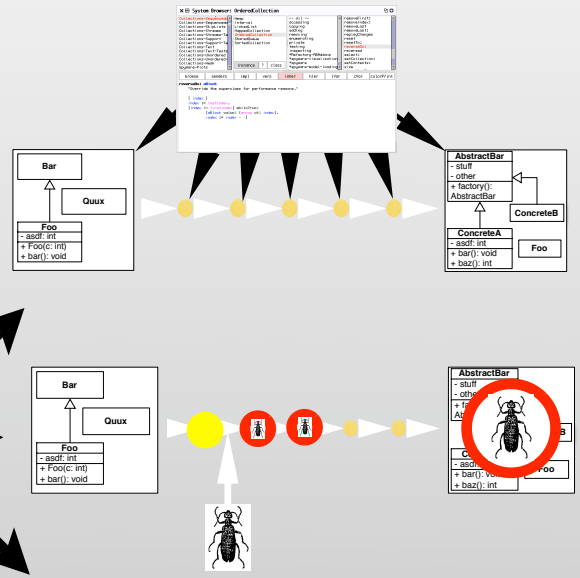


IDE

SpyWare plugin

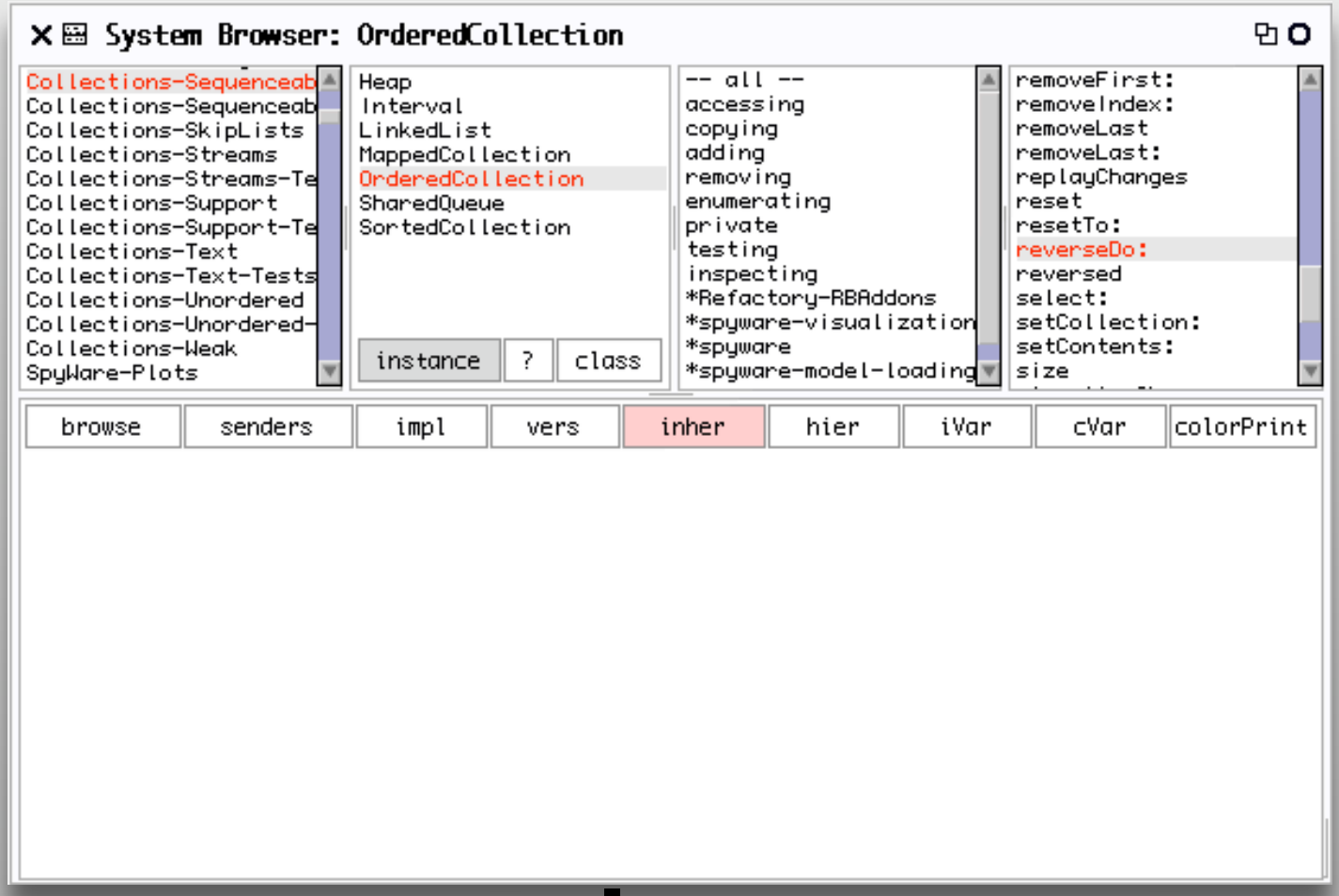


Change repository

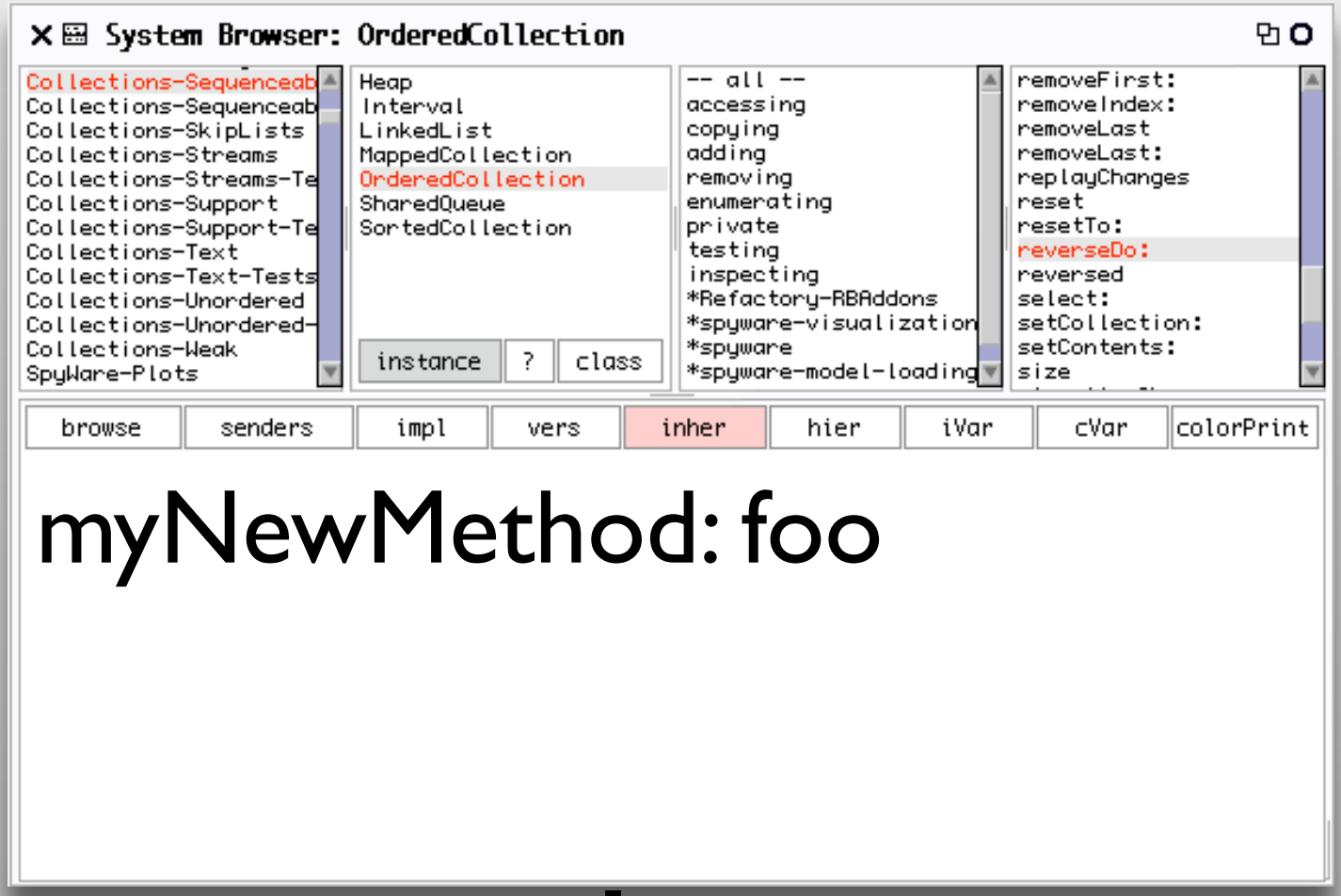


Tools

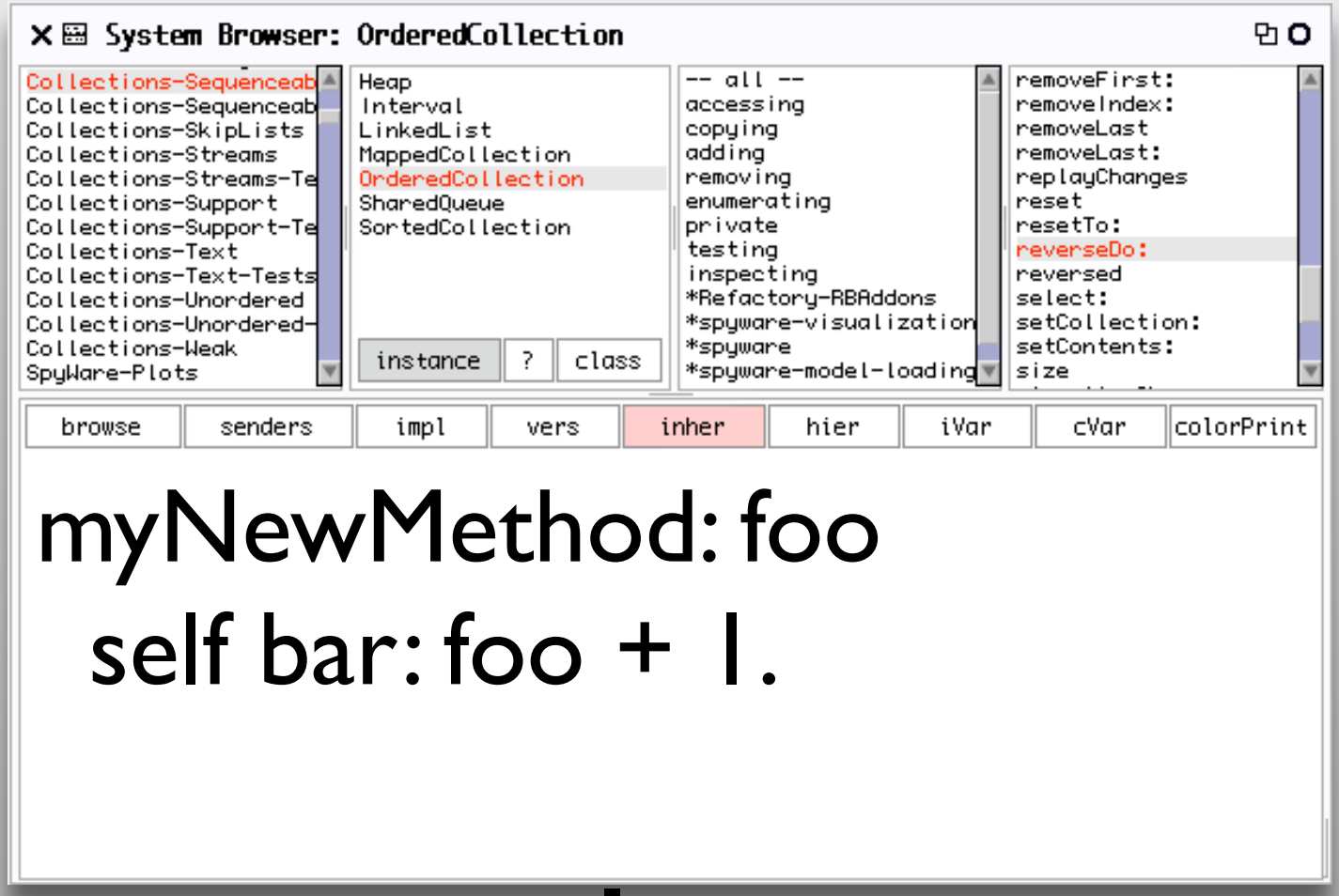
How does SpyWare work?



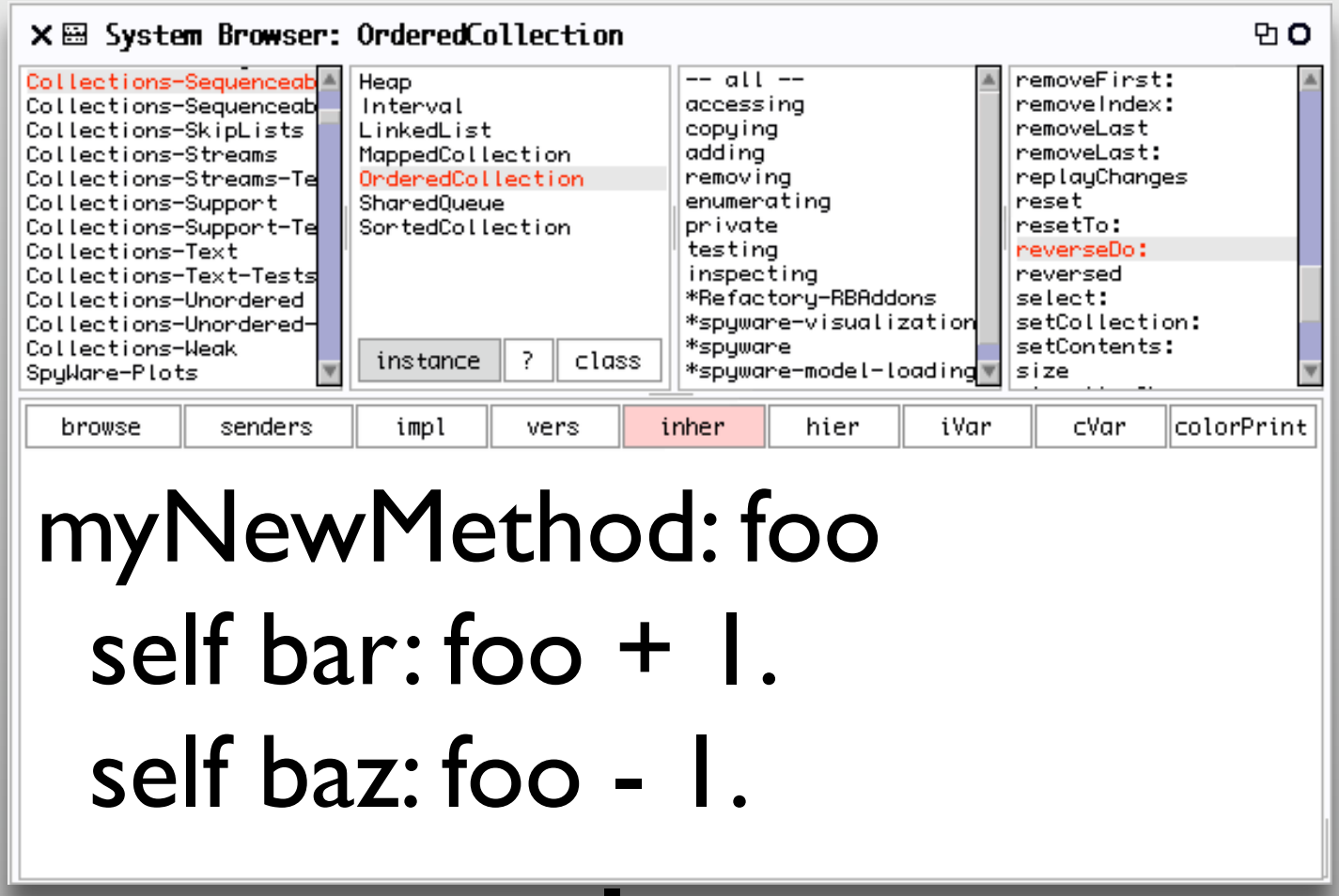
How does SpyWare work?



How does SpyWare work?

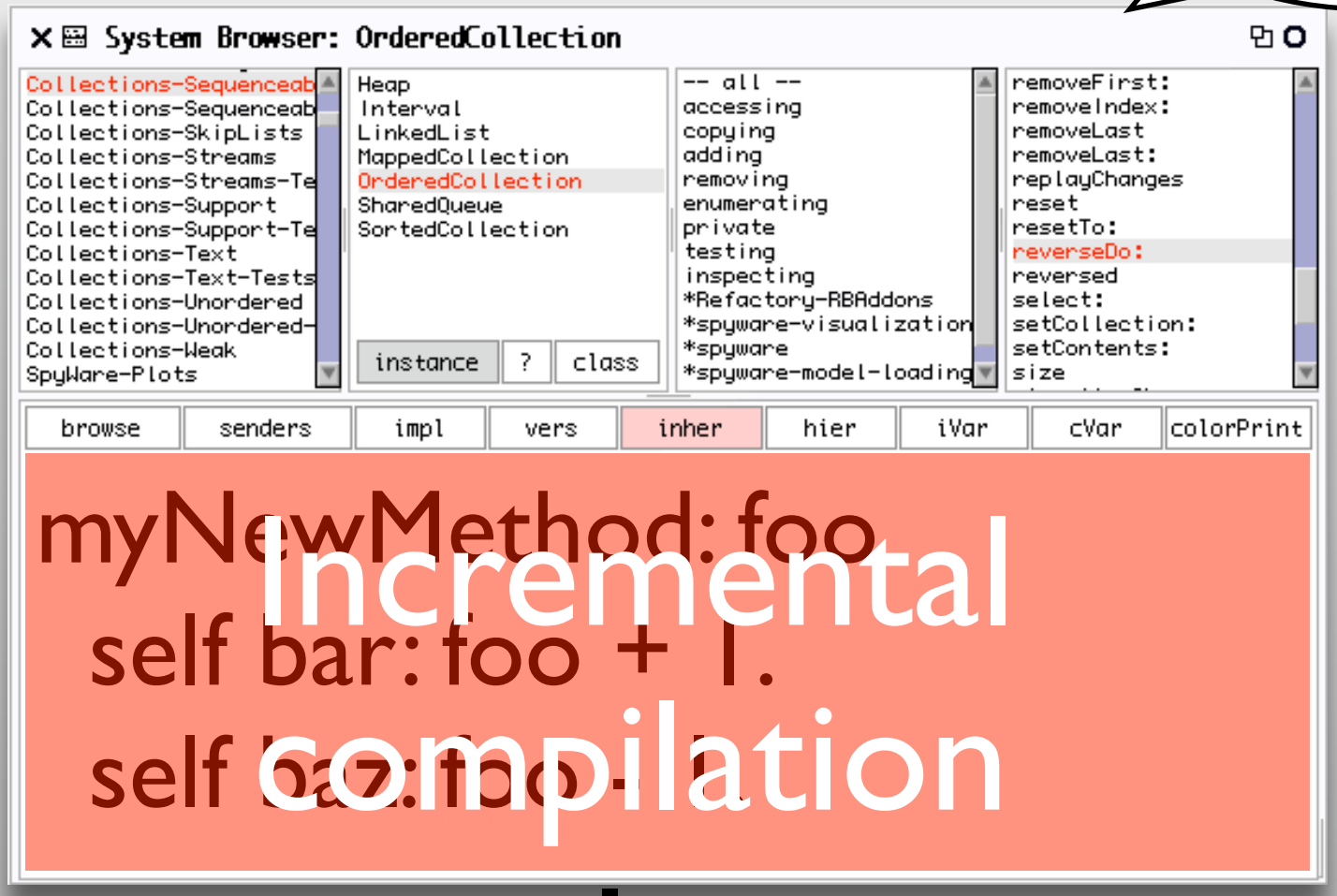


How does SpyWare work?

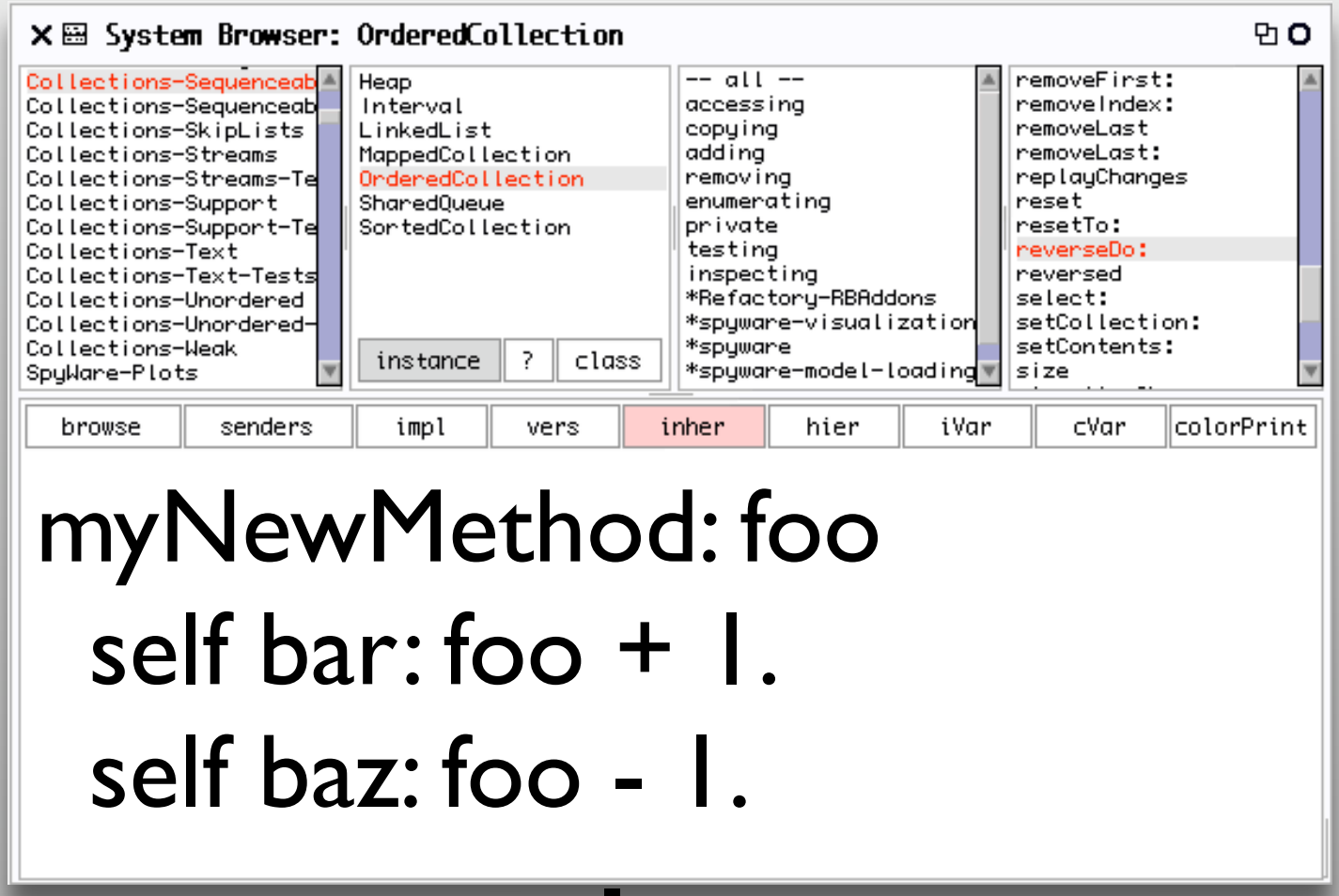


How does SpyWare work?

Method compiled!



How does SpyWare work?



The change builder infers change information

The change builder infers change information

Method compiled!

The change builder infers change information

Is it a new one?

Yes!

Who? when?

Romain Robbes @ 21h32

The change builder infers change information

create method

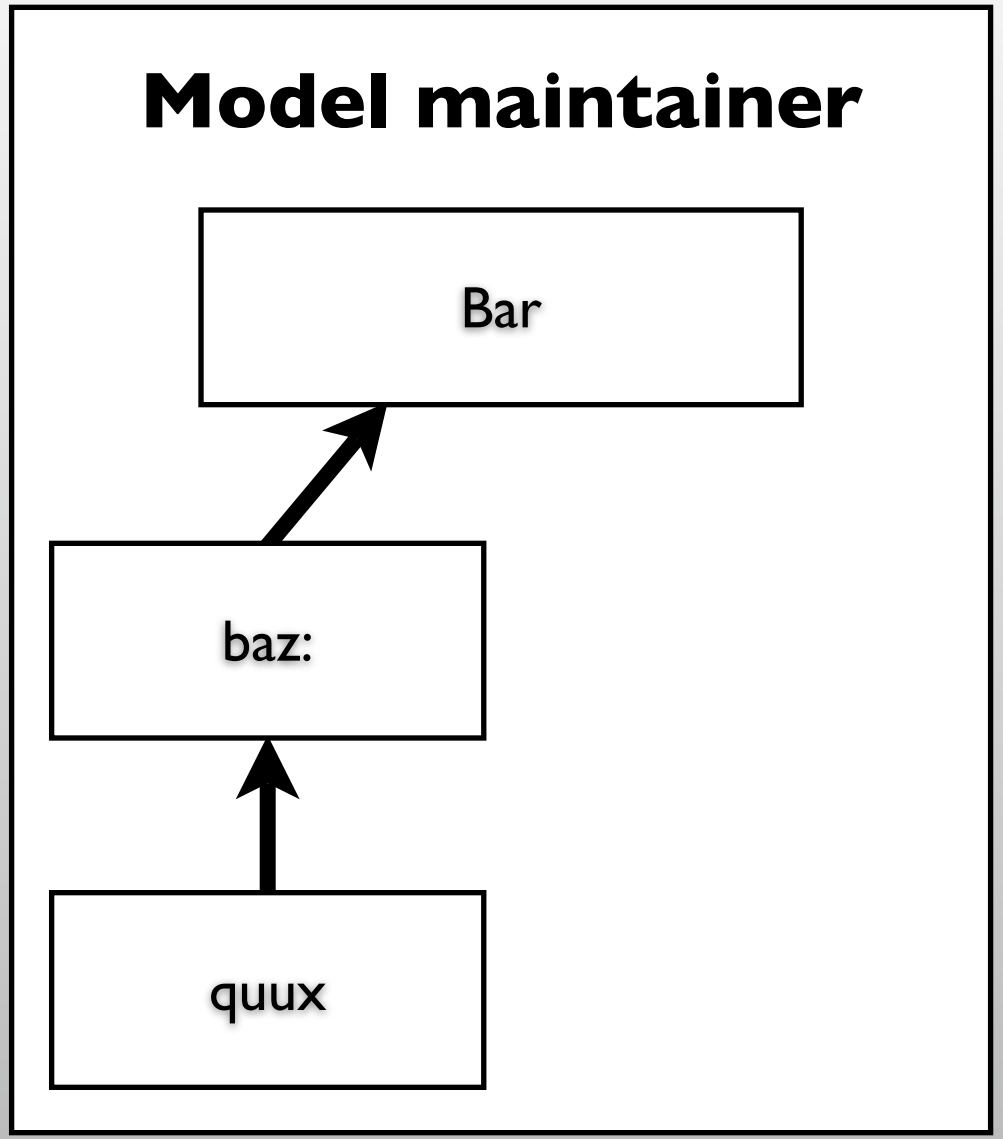
add to class

create variable

add to method

The change builder infers change information

Tools exploit the changes



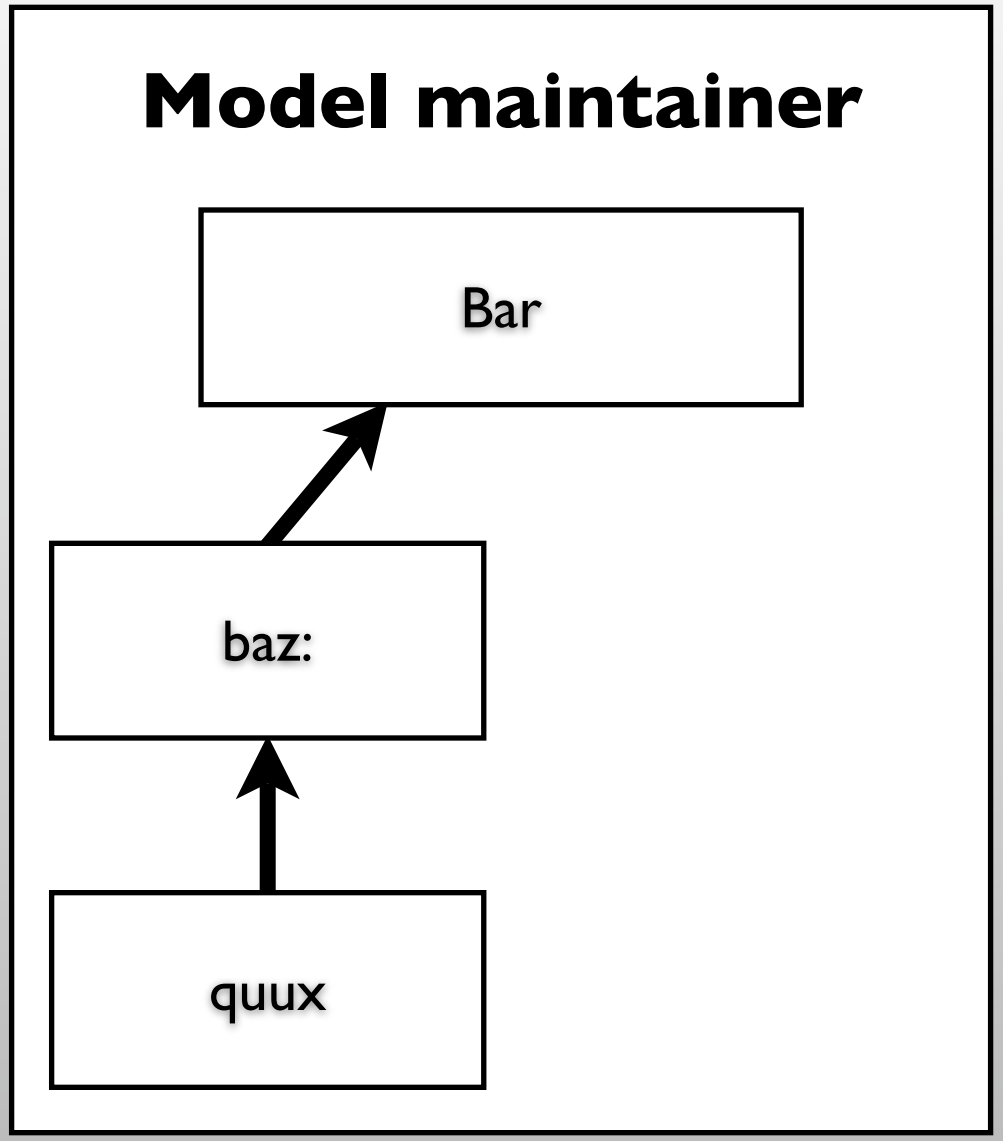
Tools exploit the changes

create method

add to class

create variable

add to method



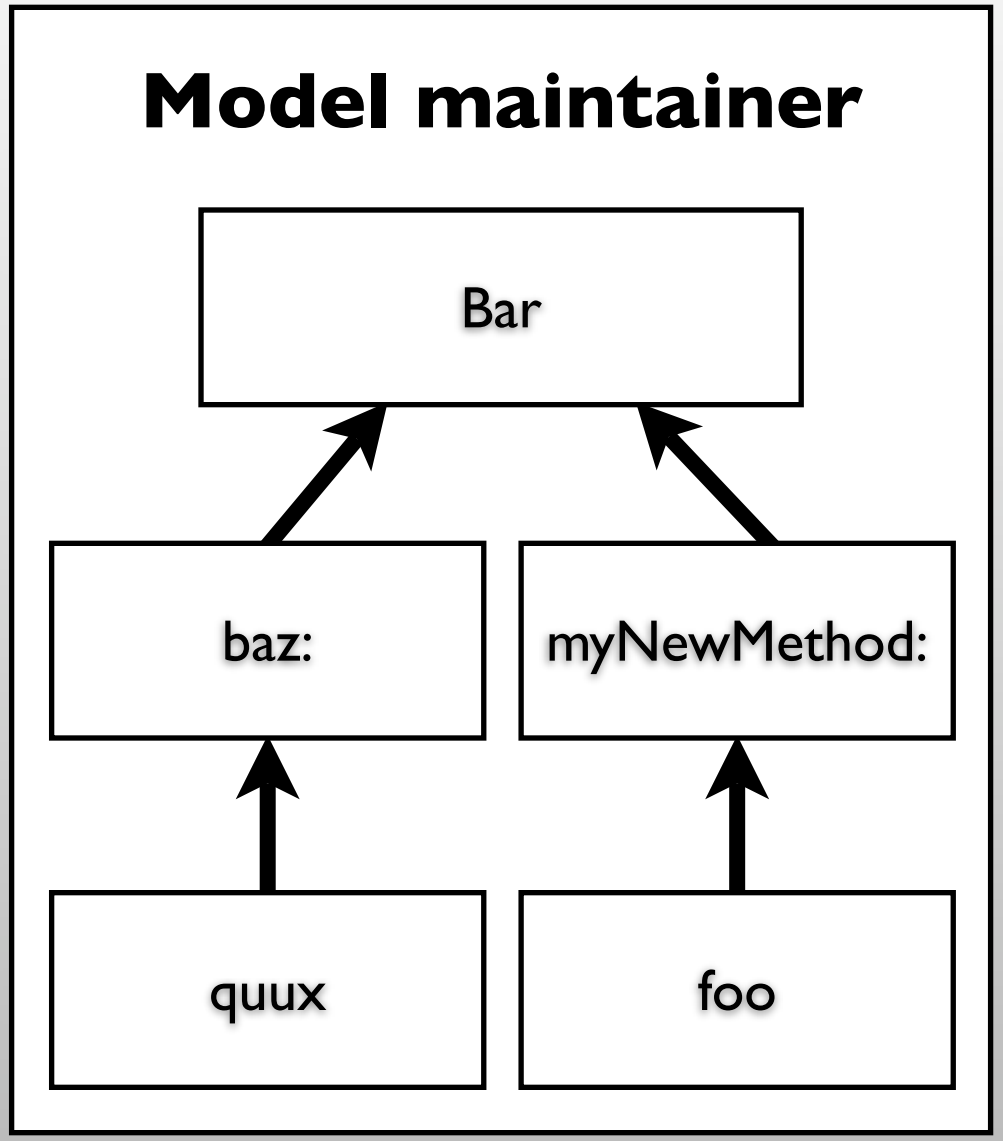
Tools exploit the changes

create method

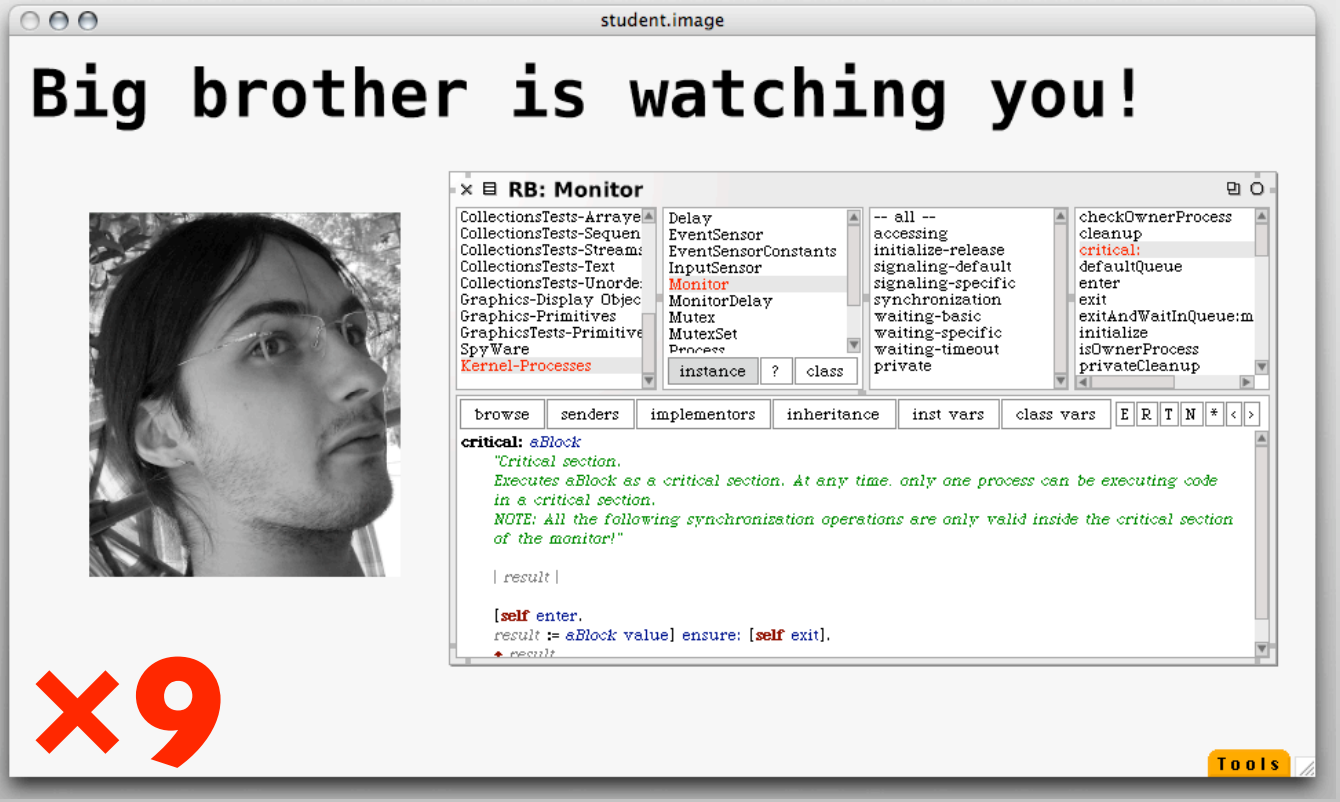
add to class

create variable

add to method



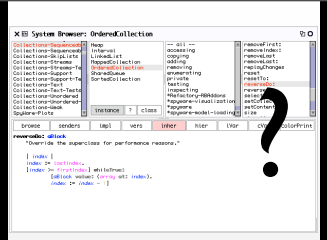
SpyWare case studies:



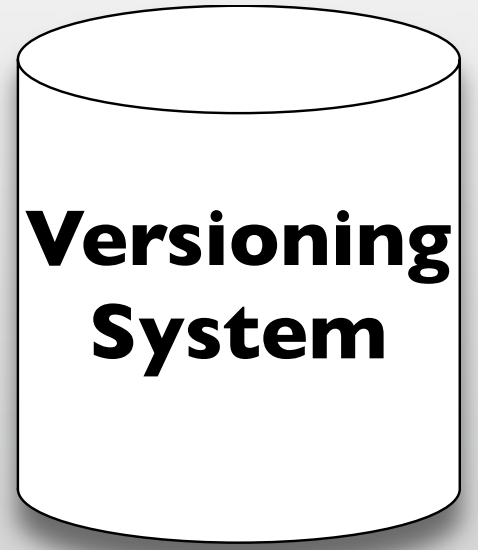
Short SpyWare demo

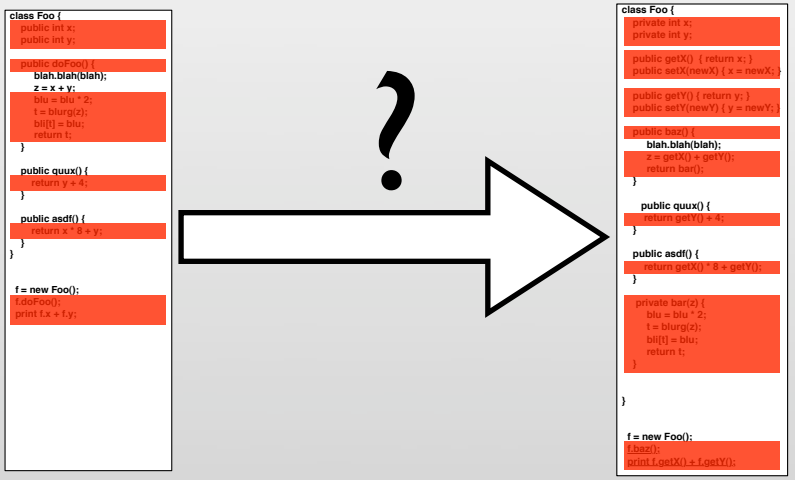
Future possibilities

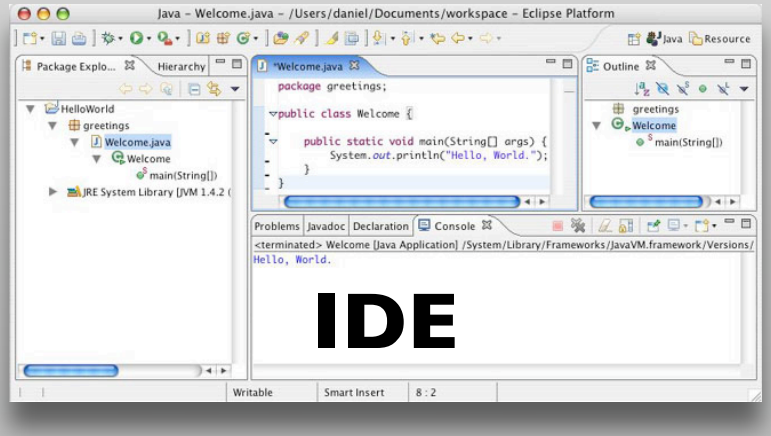
Several tools can be implemented on top of this mechanism: here are a few.

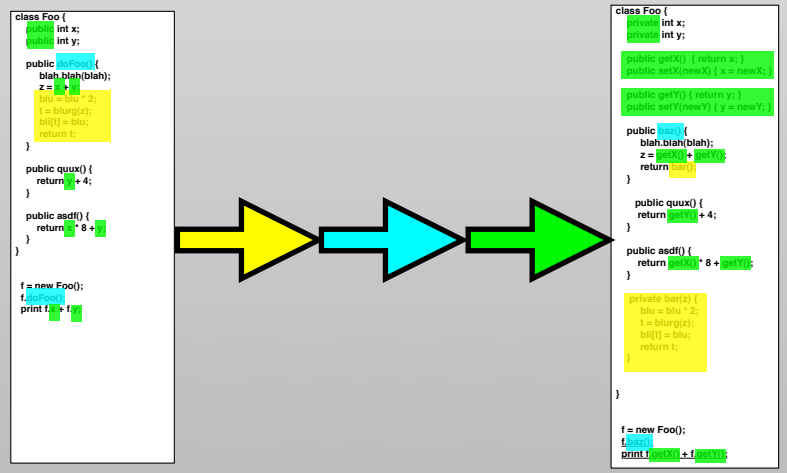


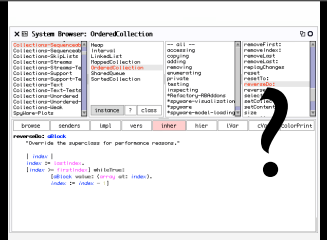
Fine-grained software evolution analysis



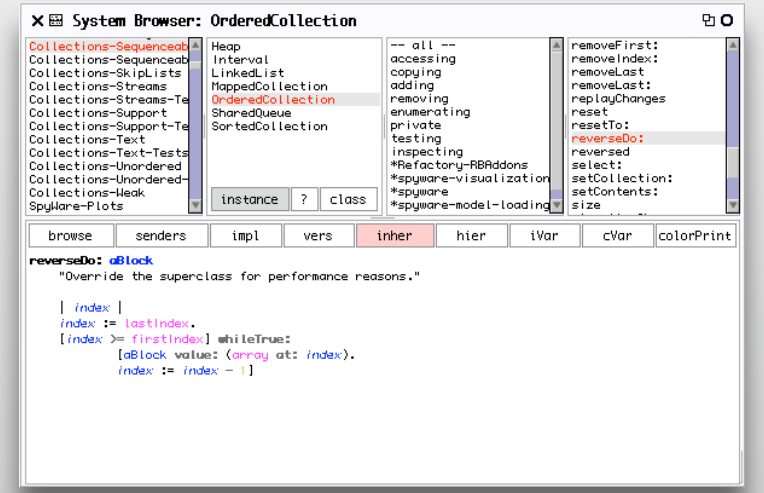


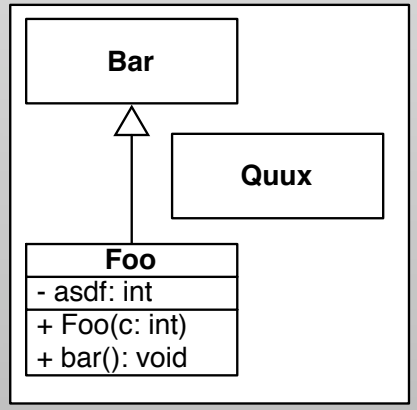


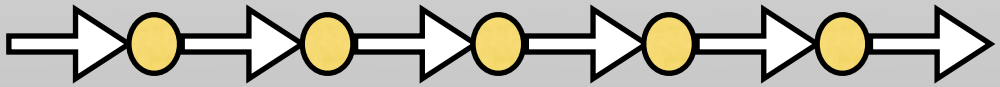


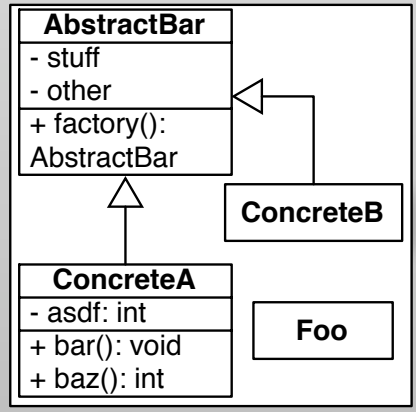


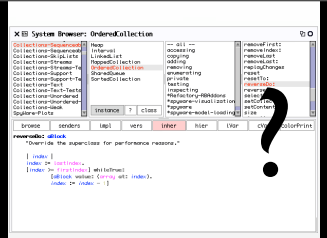
Query and understand recent changes



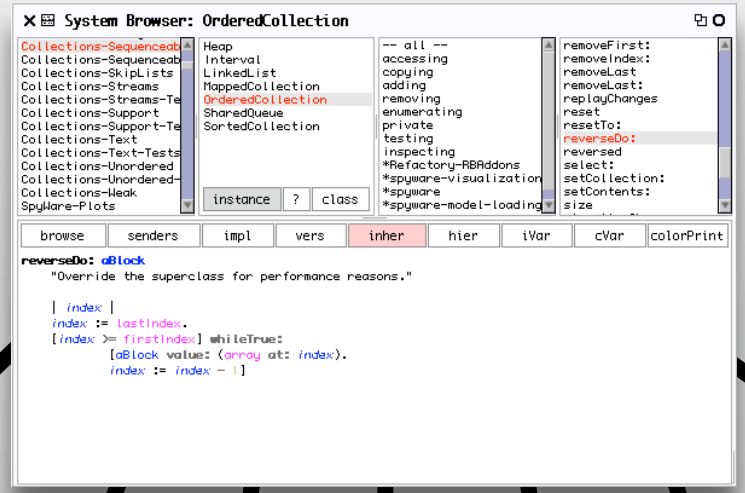


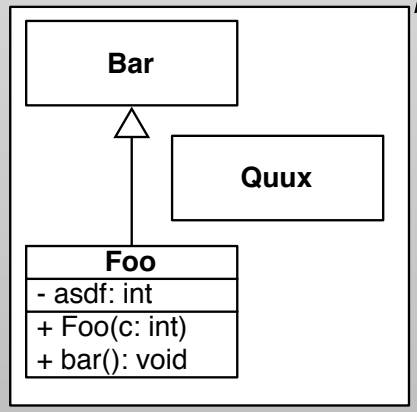


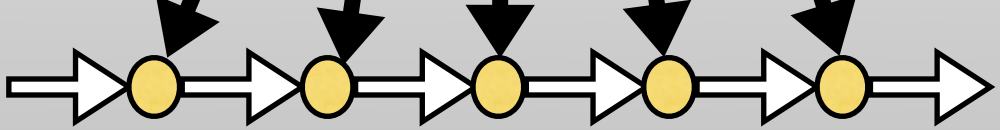


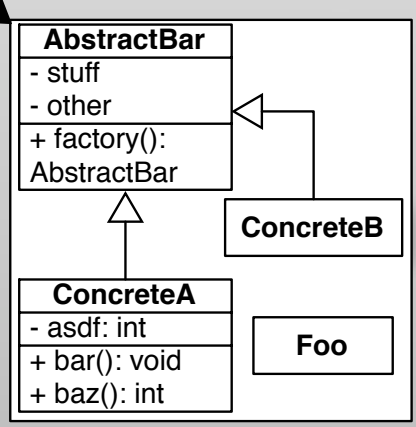


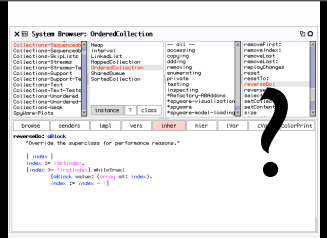
Query and understand recent changes



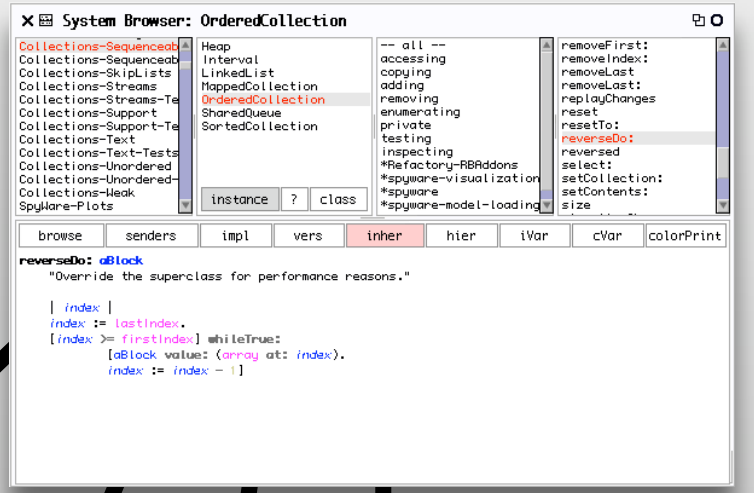


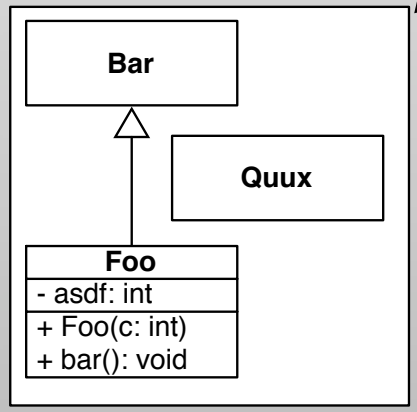


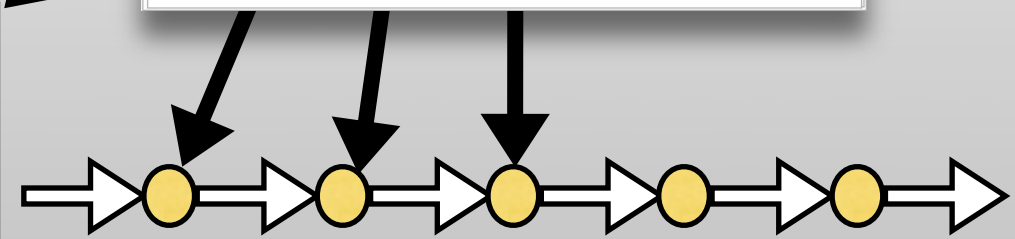


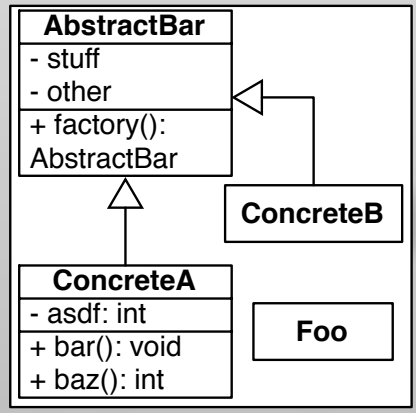


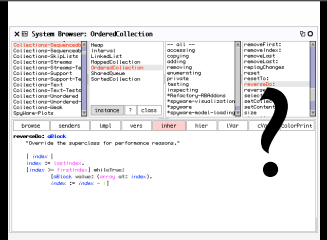
Query and understand recent changes



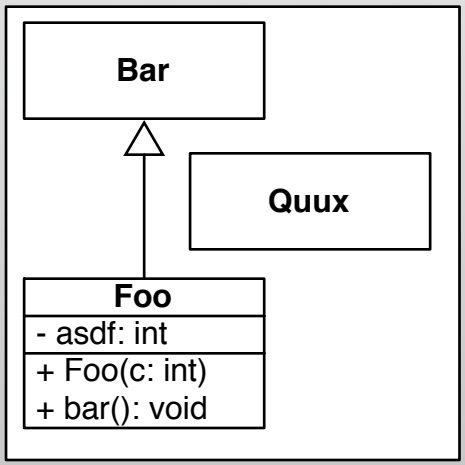


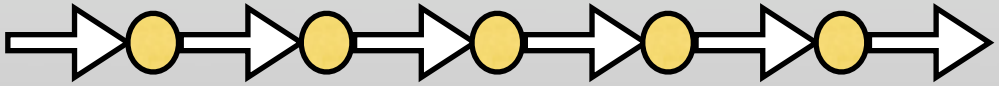


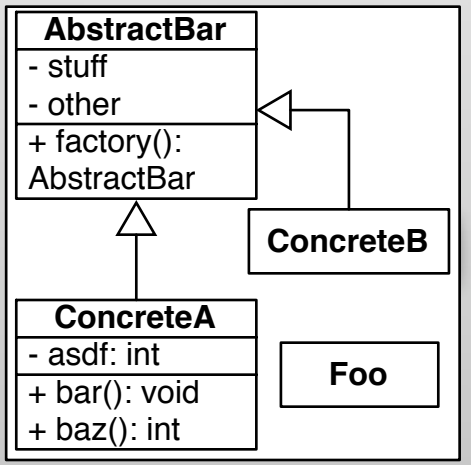


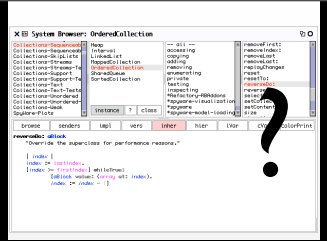


Find precise causes of bugs

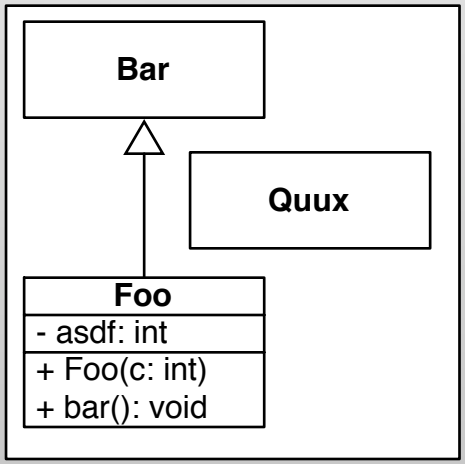


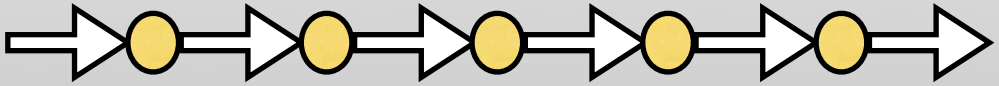


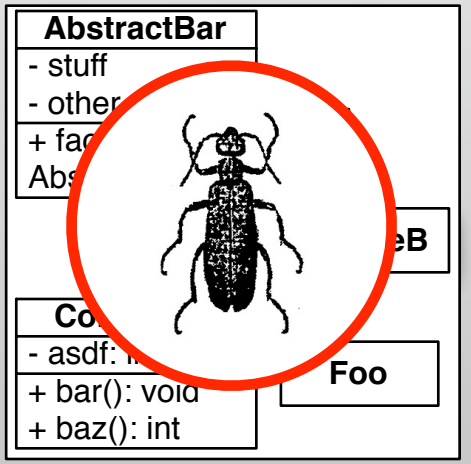


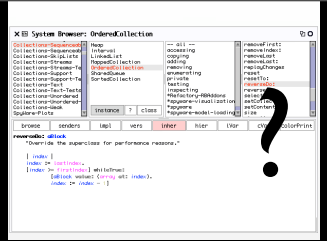


Find precise causes of bugs

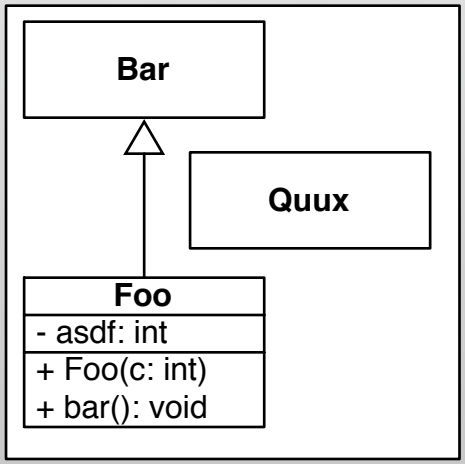


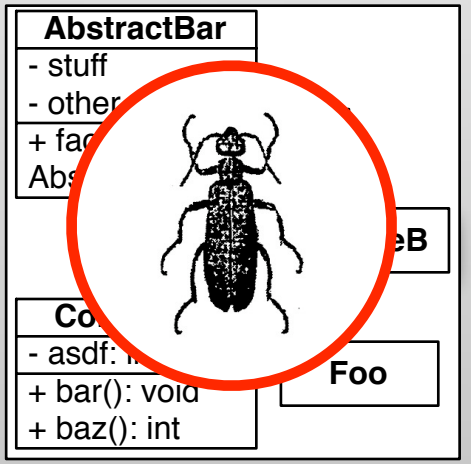


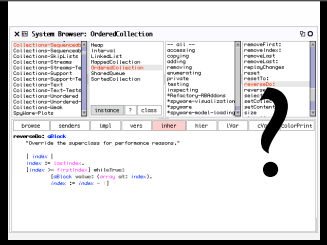




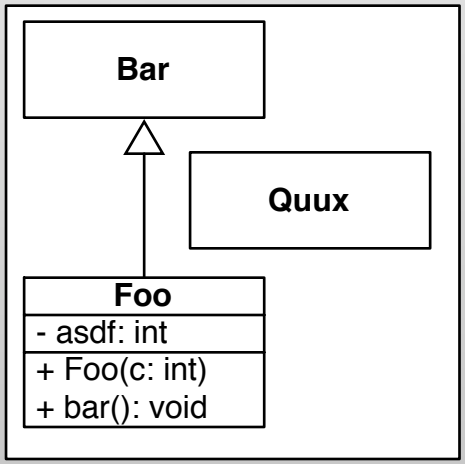
Find precise causes of bugs

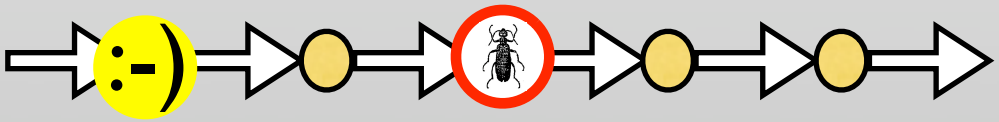


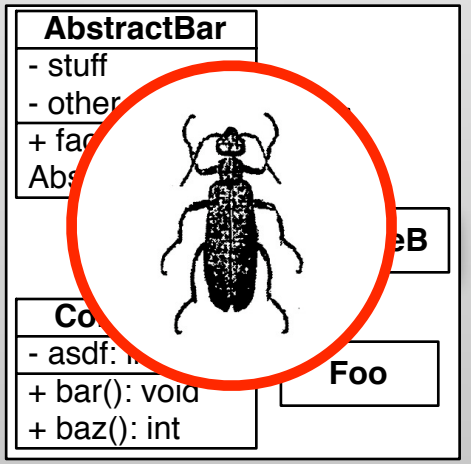


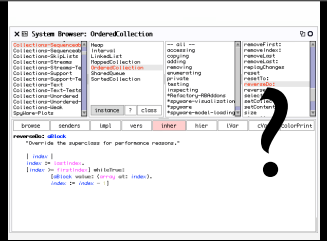


Find precise causes of bugs

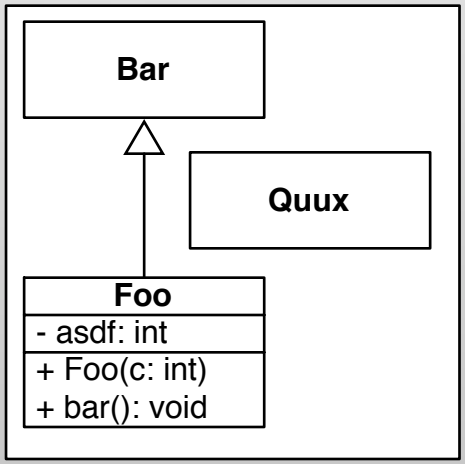


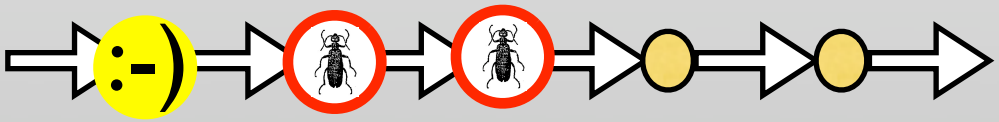


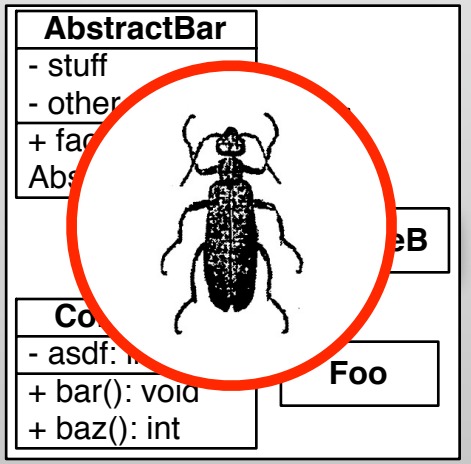


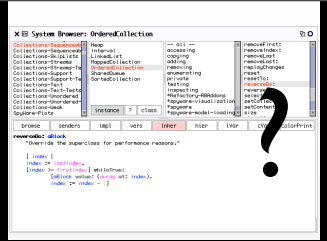


Find precise causes of bugs

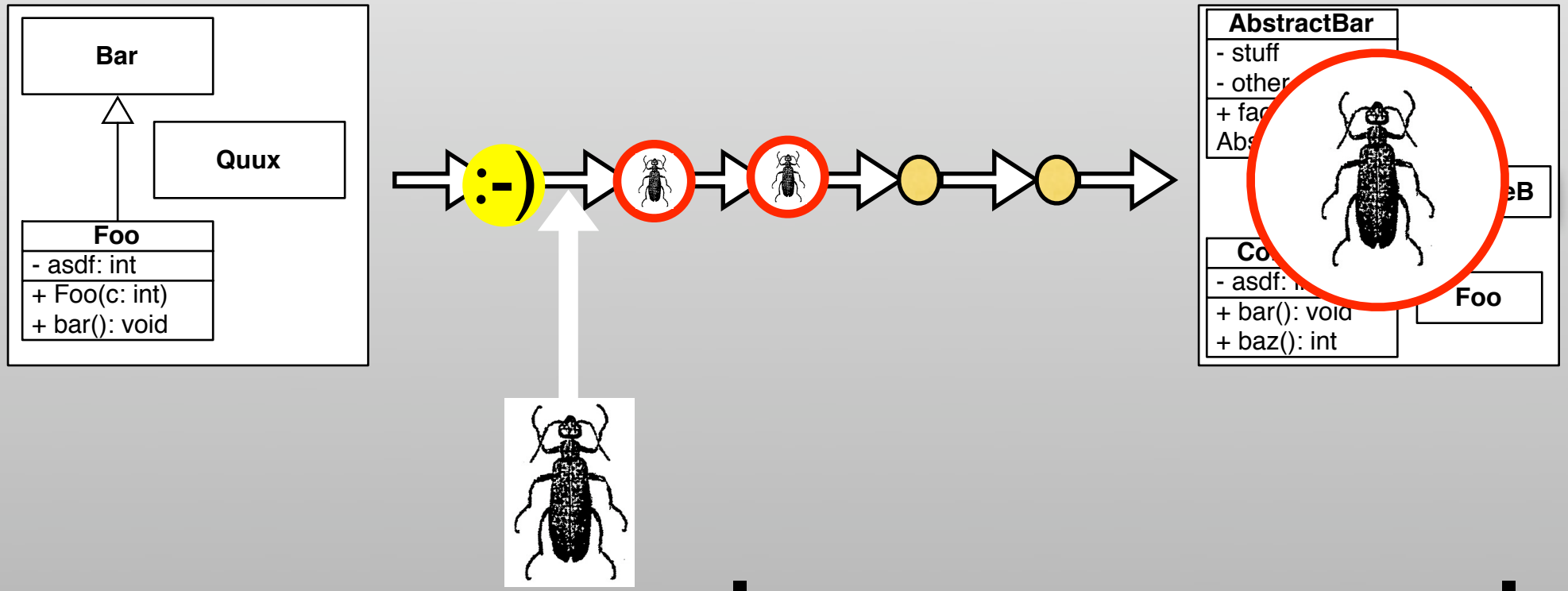


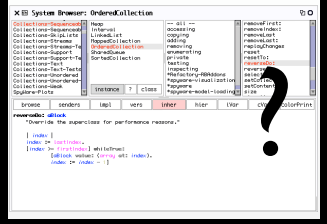




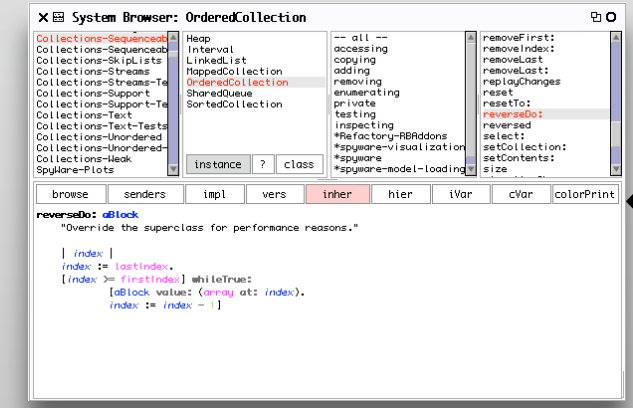


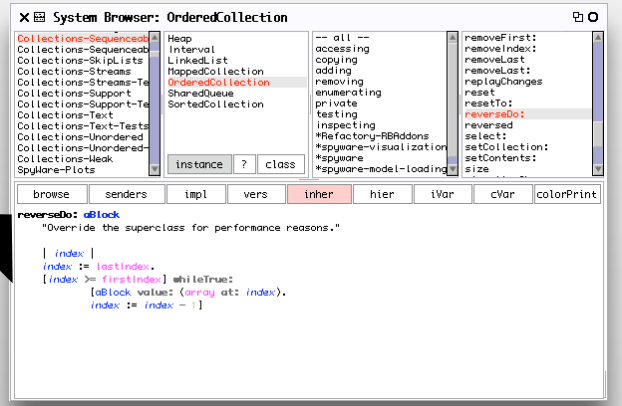
Find precise causes of bugs

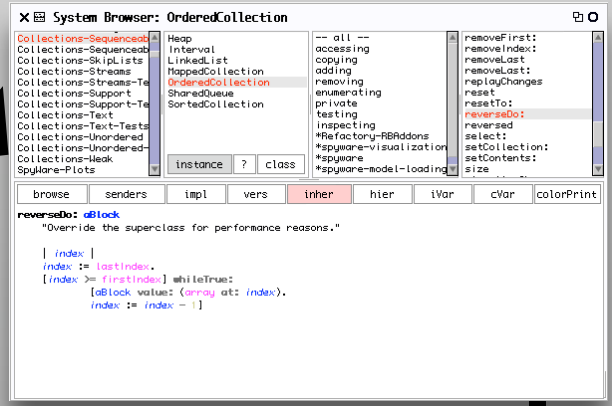


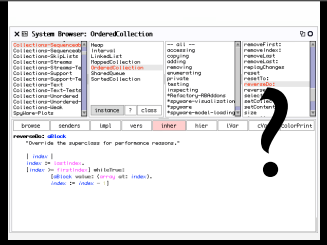


Find merge conflicts faster

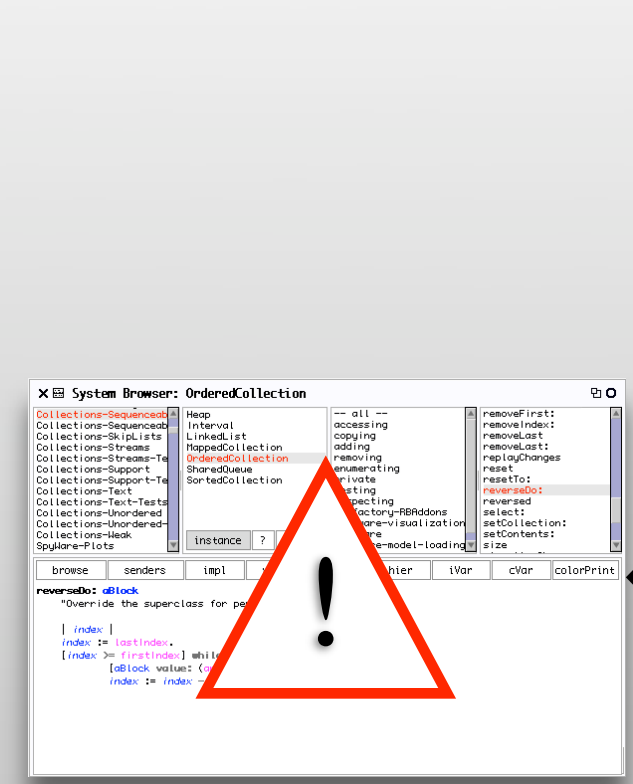






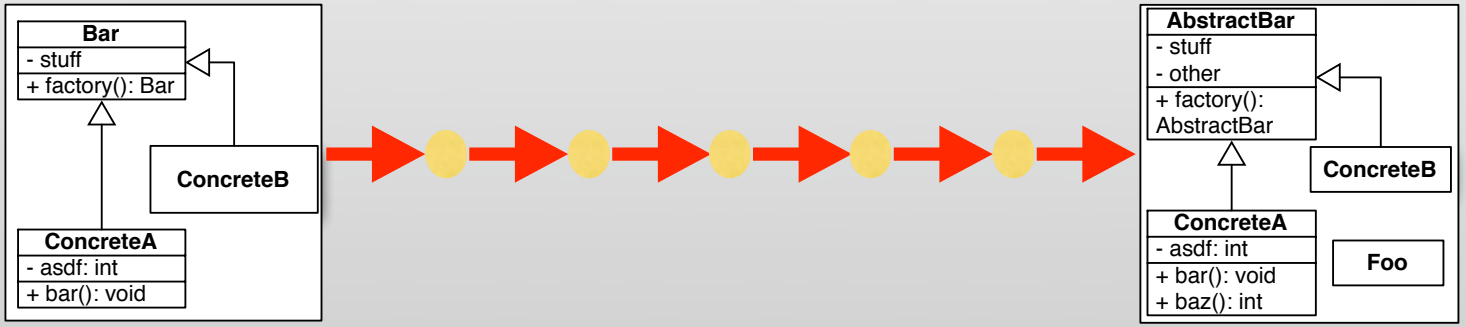


Find merge conflicts faster



Conclusions

SpyWare introduces a model of software changes



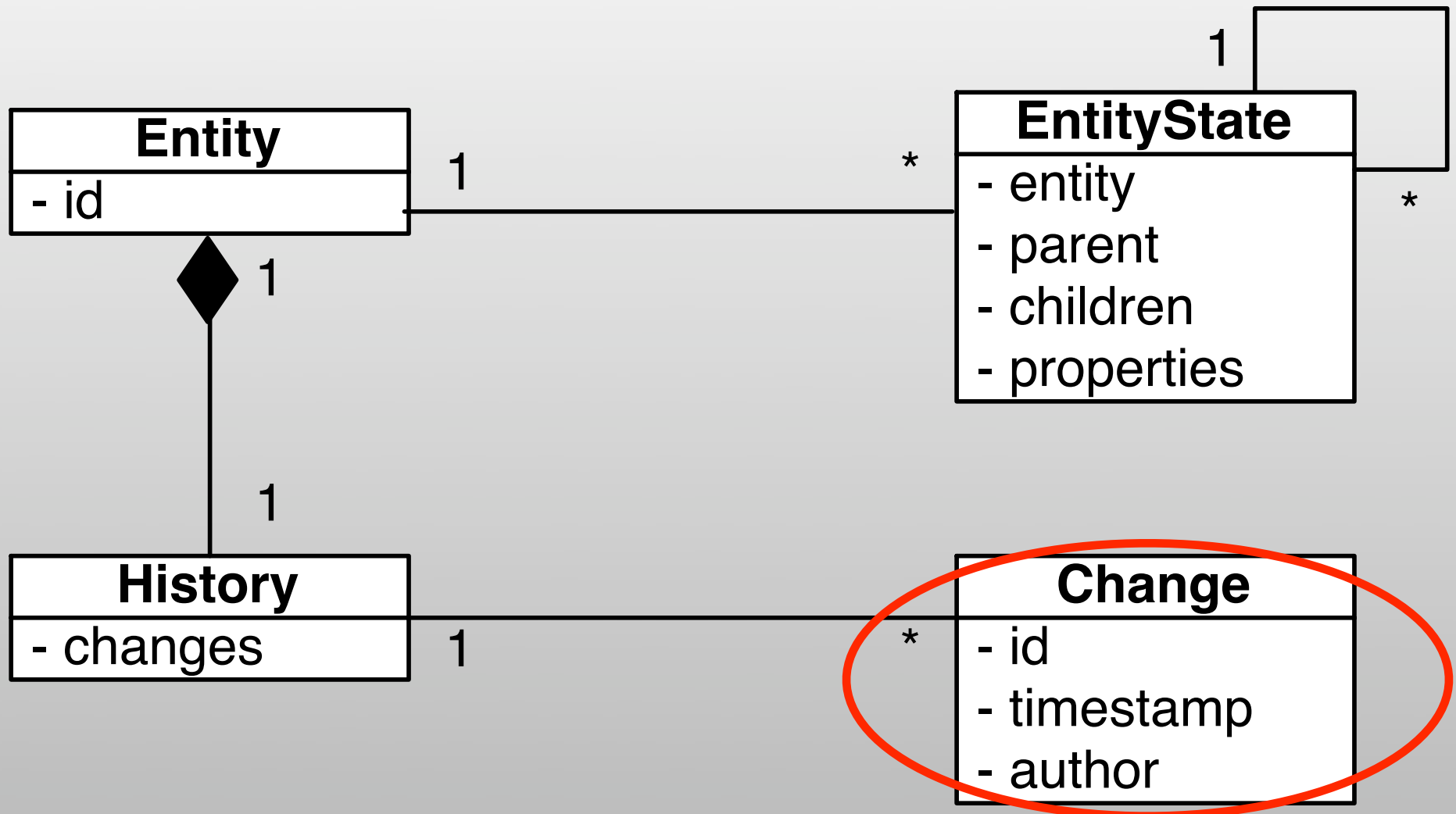
+ no information lost
+ accuracy
+ tool support

- performance?
- space?
- validation?

FOR SPYWARE NOW

http://romain.robb.es/spyware

Our model emphasizes changes over entities



Spyware versus the change log

Ad-hoc format (a bunch of do-its)
Not aware of refactorings
Tied to one image
Data loss because of purges
Exporting with change sets only keeps the last version of a method