J-Spring

16 april 2008 Spant! - Bussum

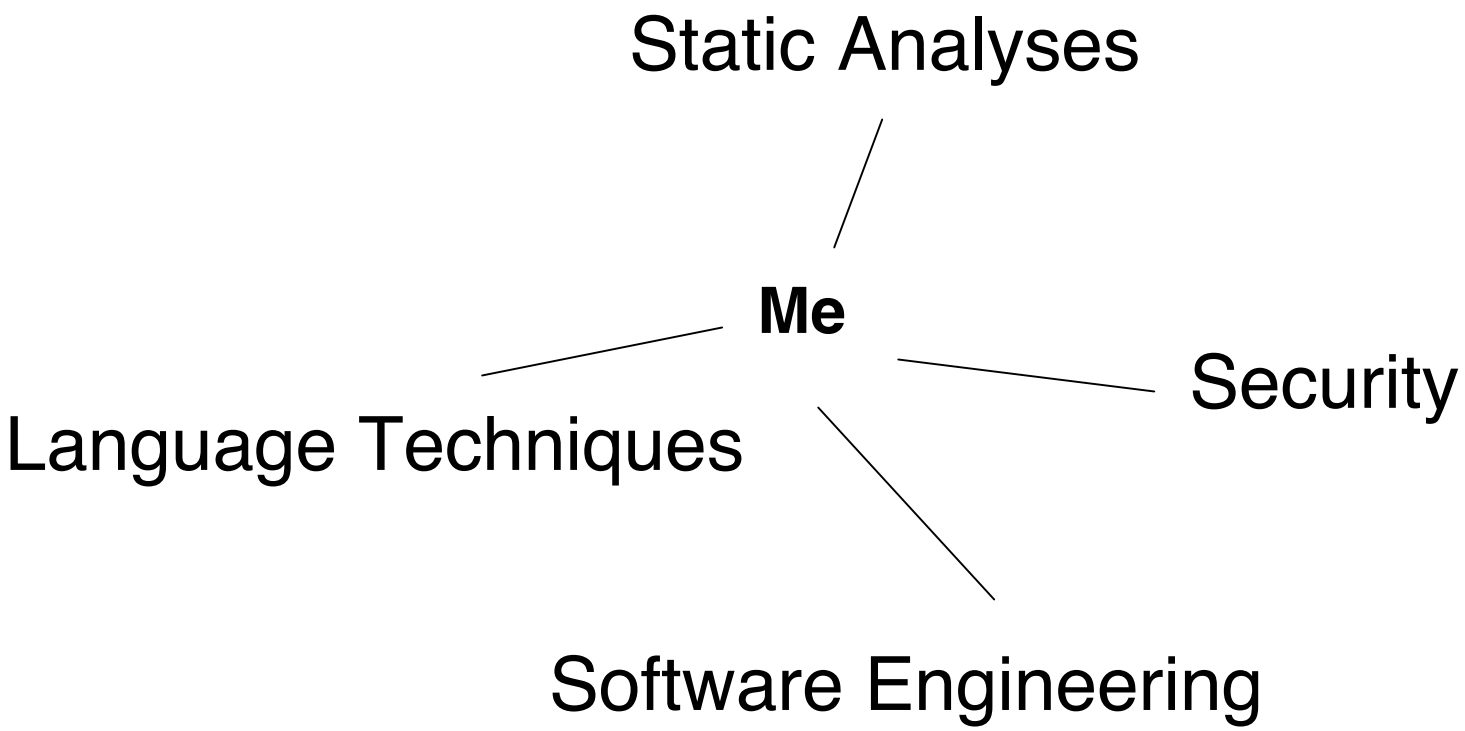
Putting Fluent Interfaces to the Test

Eric Bouwers

Software Improvement Group

Eric Bouwers

Software Improvement Group



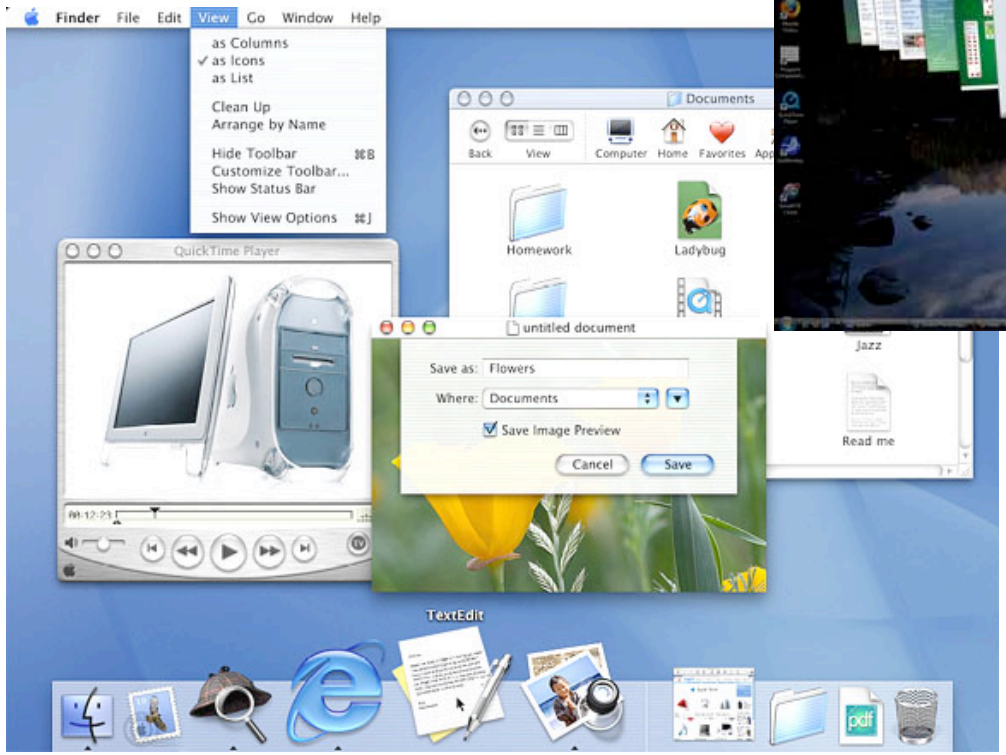
Software Improvement Group

Software Risk Assessment

Software Monitoring

DocGen

Fluent Interface



Software Improvement Group

History of Fluent Interfaces

2005 Now

http://martinfowler.com/bliki/FluentInterface.html

Definition (wikipedia)

In software engineering, a fluent interface is an objected oriented construct that defines a behavior capable of relaying the instruction context of a subsequent call

Classic example


```
private void makeUsualOrder(Customer customer) {
  Order o = new Order();
  customer.addOrder(o);
  Product p1 = new Product(1,Product.find(“Billy”));
  o.addProduct(p1);
  Product p2 = new Product(2,Product.find(”Janso"));
  o.addProduct(p2);
  Product p3 = new Product(4,Product.find(“Traby"));
  o.addProduct(p3);
   p2.setSkippable(true);
   o.setPriorityRush(true);
}
```

Classic example cont'd


```
private void makeFluentUsualOrder(Customer customer) {
   customer.newOrder()
        .with(1, “Billy”)
        .with(2, "Janso").skippable()
        .with(4, ”Traby")
        .priorityRush();
}
```

Definition (revised)

A Fluent Interface is a way to present your API as simply as possible while making its usage easy to read and write.

Definition (revised)

A Fluent Interface is a way to present your API as simply as possible while making its usage easy to read and write.

Definition (revised)

A Fluent Interface is a way to present your API as simply as possible while making its usage easy to read and write.

Real world usage

Evolving an Embedded Domain-Specific Language in Java,
Steve Freeman, Nat Pryce,
In Proceedings OOPSLA 2006

Real world usage

Software Improvement Group

```
import org.jmock.Expectations;
import org.jmock.Mockery;

public class JmockTestClass extends TestCase {

    Mockery context = new Mockery();

    public void testMockObject() {

        final Order mockOrder = context.mock(Order.class);

        Expectations expect = new Expectations(){{
            exactly(3).of(mockOrder).addOrder(with(an

        }};
```

```
an(Class<T> type) Matcher<T> - Expectations
any(Class<T> type) Matcher<T> - Expectations
anything() Matcher<T> - Expectations
aNonNull(Class<T> type) Matcher<T> - Expectati
aNull(Class<T> type) Matcher<T> - Expectations
AncestorEvent - javax.swing.event
AncestorIterator - com.sun.org.apache.xml.interna
AncestorIterator - com.sun.org.apache.xml.interna
AncestorListener - javax.swing.event
And - com.sun.org.apache.xpath.internal.operation
Press '^.' to show Template Proposals
```

```
Syntax error on token ",", Expression expected after this token
```

Real world usage

Software Improvement Group

```
t.mock(Order.class);

ctations(){{
ddOrder(with(an
```

an(Class<T> type) Matcher<T> - Expectations
any(Class<T> type) Matcher<T> - Expectations
anything() Matcher<T> - Expectations
aNonNull(Class<T> type) Matcher<T> - Expectati
aNull(Class<T> type) Matcher<T> - Expectations
AncestorEvent - javax.swing.event
AncestorIterator - com.sun.org.apache.xml.interna
AncestorIterator - com.sun.org.apache.xml.interna
AncestorListener - javax.swing.event
And - com.sun.org.apache.xpath.internal.operation

Press '^_' to show Template Proposals

Use cases

Pattern matching & Mdx-queries

Use cases

Pattern matching & Mdx-queries

Demand-driven Specification

Use cases

Pattern matching & Mdx-queries

Demand-driven Specification

Loose Strict

Use case 1

Software Improvement Group

Pattern Matching

Method - Extraction

Software Improvement Group

```
public class Example {
       String name

     public Example(){
          this.name = "default”;}

     public void testExample(int message){
         System.out.println(this.name);
         System.out.println(message);
     }
}
```

Method - Extraction


```
public class Example {
       String name

     public Example(){
          this.name = "default”;}

     public void testExample(int message){
         System.out.println(this.name);
         System.out.println(message);
     }
}
```

Tokenized Source


```
[public,class,Example,{,String,name,public,Exampl
  e,(,),{,this.name,=,"default",;,},public,void,
testExample,(,int,message,),{,System.out.println,
  (,this.name,),;,System.out.println,(,message,),
  ;,},}]
```

Method - Pattern


```
TypePredicate PUBLIC = new TypePredicate(LITERAL_PUBLIC);
TypePredicate PRIVATE = new TypePredicate(LITERAL_PRIVATE);
BackReferenceablePredicate IDENTIFIER = new
    BackReferenceablePredicate(new TypePredicate(IDENT));
TypePredicate RETURN_TYPE = new TypePredicate(TYPE);
TypePredicate VOID = new TypePredicate(LITERAL_VOID);
TypePredicate LCURLYT = new TypePredicate(LCURLY);
TypePredicate RCURLYT = new TypePredicate(RCURLY);

OrPredicate PUBLIC_OR_PRIVATE = new OrPredicate(PUBLIC, PRIVATE);
OrPredicate RETURN_TYPE_OR_VOID = new OrPredicate(RETURN_TYPE, VOID);

Expression method = new Expression(new Object[] {
    PUBLIC_OR_PRIVATE,RETURN_TYPE_OR_VOID, IDENTIFIER,
    new BalancedTextPredicate("(", ")"),
    new BalancedPredicate(LCURLYT, RCURLYT)});
```

Method - Pattern

Software Improvement Group

```
TypePredicate PUBLIC = new TypePredicate(LITERAL_PUBLIC);
TypePredicate PRIVATE = new TypePredicate(LITERAL_PRIVATE);
BackReferenceablePredicate IDENTIFIER = new
    BackReferenceablePredicate(new TypePredicate(IDENT));
TypePredicate RETURN_TYPE = new TypePredicate(TYPE);
TypePredicate VOID = new TypePredicate(LITERAL_VOID);
TypePredicate LCURLYT = new TypePredicate(LCURLY);
TypePredicate RCURLYT = new TypePredicate(RCURLY);

OrPredicate PUBLIC_OR_PRIVATE = new OrPredicate(PUBLIC, PRIVATE);
OrPredicate RETURN_TYPE_OR_VOID = new OrPredicate(RETURN_TYPE, VOID);

Expression method = new Expression(new Object[] {
    PUBLIC_OR_PRIVATE,RETURN_TYPE_OR_VOID, IDENTIFIER,
    new BalancedTextPredicate("(", ")"),
    new BalancedPredicate(LCURLYT, RCURLYT)});
```

Method - Pattern


```
TypePredicate PUBLIC = new TypePredicate(LITERAL_PUBLIC);
TypePredicate PRIVATE = new TypePredicate(LITERAL_PRIVATE);
BackReferenceablePredicate IDENTIFIER = new
    BackReferenceablePredicate(new TypePredicate(IDENT));
TypePredicate RETURN_TYPE = new TypePredicate(TYPE);
TypePredicate VOID = new TypePredicate(LITERAL_VOID);
TypePredicate LCURLYT = new TypePredicate(LCURLY);
TypePredicate RCURLYT = new TypePredicate(RCURLY);

OrPredicate PUBLIC_OR_PRIVATE = new OrPredicate(PUBLIC, PRIVATE);
OrPredicate RETURN_TYPE_OR_VOID = new OrPredicate(RETURN_TYPE, VOID);

Expression method = new Expression(new Object[] {
    PUBLIC_OR_PRIVATE,RETURN_TYPE_OR_VOID, IDENTIFIER,
    new BalancedTextPredicate("(", ")"),
    new BalancedPredicate(LCURLYT, RCURLYT)});
```

Method - Pattern


```
TypePredicate PUBLIC = new TypePredicate(LITERAL_PUBLIC);
TypePredicate PRIVATE = new TypePredicate(LITERAL_PRIVATE);
BackReferenceablePredicate IDENTIFIER = new
    BackReferenceablePredicate(new TypePredicate(IDENT));
TypePredicate RETURN_TYPE = new TypePredicate(TYPE);
TypePredicate VOID = new TypePredicate(LITERAL_VOID);
TypePredicate LCURLYT = new TypePredicate(LCURLY);
TypePredicate RCURLYT = new TypePredicate(RCURLY);

OrPredicate PUBLIC_OR_PRIVATE = new OrPredicate(PUBLIC, PRIVATE);
OrPredicate RETURN_TYPE_OR_VOID = new OrPredicate(RETURN_TYPE, VOID);

Expression method = new Expression(new Object[] {
    PUBLIC_OR_PRIVATE,RETURN_TYPE_OR_VOID, IDENTIFIER,
    new BalancedTextPredicate("(", ")"),
    new BalancedPredicate(LCURLYT, RCURLYT)});
```

Method - Pattern


```
TypePredicate PUBLIC = new TypePredicate(LITERAL_PUBLIC);
TypePredicate PRIVATE = new TypePredicate(LITERAL_PRIVATE);
BackReferenceablePredicate IDENTIFIER = new
    BackReferenceablePredicate(new TypePredicate(IDENT));
TypePredicate RETURN_TYPE = new TypePredicate(TYPE);
TypePredicate VOID = new TypePredicate(LITERAL_VOID);
TypePredicate LCURLYT = new TypePredicate(LCURLY);
TypePredicate RCURLYT = new TypePredicate(RCURLY);

OrPredicate PUBLIC_OR_PRIVATE = new OrPredicate(PUBLIC, PRIVATE);
OrPredicate RETURN_TYPE_OR_VOID = new OrPredicate(RETURN_TYPE, VOID);

Expression method = new Expression(new Object[] {
    PUBLIC_OR_PRIVATE,RETURN_TYPE_OR_VOID, IDENTIFIER,
    new BalancedTextPredicate("(", ")"),
    new BalancedPredicate(LCURLYT, RCURLYT)});
```

Method - Fluent Pattern


```
PatternBuilder pb =
   matchA(LITERAL_PUBLIC).or(LITERAL_PRIVATE)
.followedByA(IDENT)
        .withReference("method_name")
.followedByA(balanced("(", ")"))
.followedByA(balanced(LCURLY, RCURLY));
```

Method - Fluent Pattern

Software Improvement Group

```
PatternBuilder pb =
   matchA(LITERAL_PUBLIC).or(LITERAL_PRIVATE)
.followedByA(IDENT)
        .withReference("method_name")
.followedByA(balanced("(", ")"))
.followedByA(balanced(LCURLY, RCURLY));
```

Method - Fluent Pattern


```
PatternBuilder pb =
   matchA(LITERAL_PUBLIC).or(LITERAL_PRIVATE)
.followedByA(IDENT)
        .withReference("method_name")
.followedByA(balanced("(", ")"))
.followedByA(balanced(LCURLY, RCURLY));
```

Method - Fluent Pattern

Software Improvement Group

```
PatternBuilder pb =
    matchA(LITERAL_PUBLIC).or(LITERAL_PRIVATE)
.followedByA(IDENT)
        .withReference("method_name")
.followedByA(balanced("(", ")"))
.followedByA(balanced(LCURLY, RCURLY));
```

Implementation

PatternBuilder
- 25 public - 8 static - 250 LOC

Implementation


```
public PatternBuilder followedBy(Object o) {
    addToList(o);
    return this;
}
```

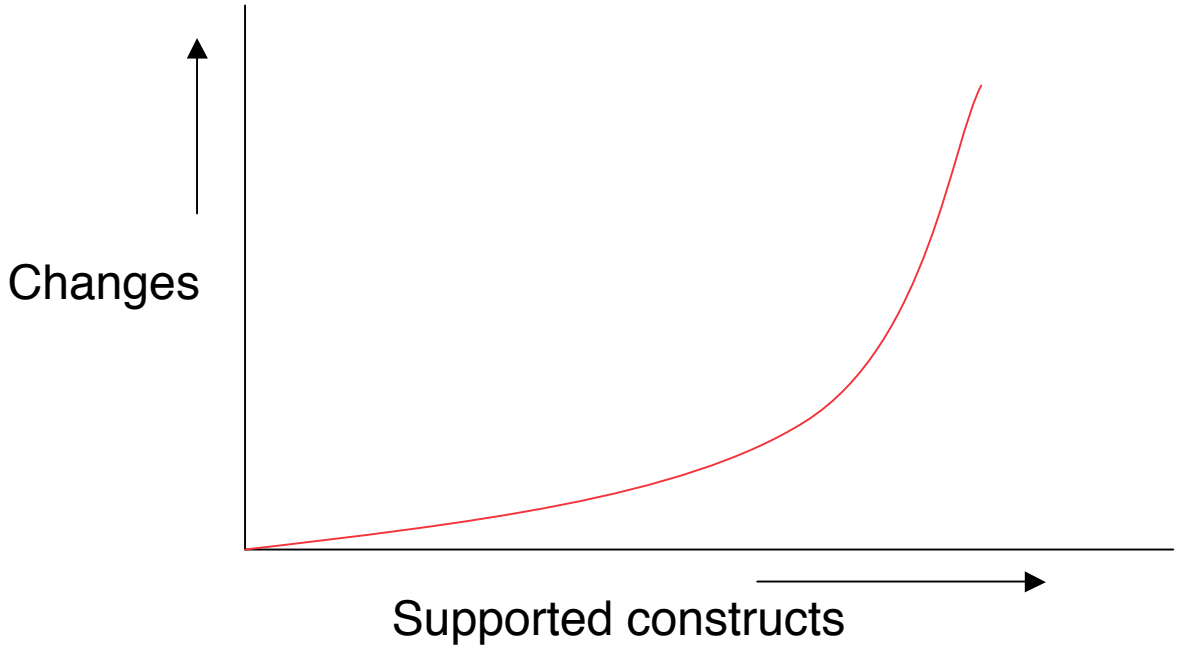
PatternBuilder
- 25 public - 8 static - 250 LOC

Implementation


```
public PatternBuilder untilFirst(Integer i) {
    TypePredicate p =
                new TypePredicate(i);
    addToList(
        new ZeroOrMore(
                new NotPredicate(p)));
    addToList(p);
    return this;
}
```

PatternBuilder
- 25 public
- 8 static
- 250 LOC

Lessons Learned



Software Improvement Group

Lessons Learned

A good name, like good will, is got by many actions and lost by one.

Lord Jeffery

Use case 2

Software Improvement Group

Mdx-queries

Use case 2, MDX-queries


```
with
Member [Measures].[Name] as
                        [Time].currentMember.Name
select
   {[Measures].[Name],[Measures].[LINES_OF_CODE]}
ON COLUMNS,
{[Time].Children}
ON ROWS
from [cube]
```

Use case 2, MDX-queries

Software Improvement Group

```
with
Member [Measures].[Name] as
                        [Time].currentMember.Name
select
   {[Measures].[Name],[Measures].[LINES_OF_CODE]}
ON COLUMNS,
{[Time].Children}
ON ROWS
from [cube]
```

Use case 2, MDX-queries

Software Improvement Group

```
with
Member [Measures].[Name] as
                        [Time].currentMember.Name
select
   {[Measures].[Name],[Measures].[LINES_OF_CODE]}
ON COLUMNS,
{[Time].Children}
ON ROWS
from [cube]
```

Use case 2, MDX-queries

Software Improvement Group

```
with
Member [Measures].[Name] as
                        [Time].currentMember.Name
select
   {[Measures].[Name],[Measures].[LINES_OF_CODE]}
ON COLUMNS,
{[Time].Children}
ON ROWS
from [cube]
```

	Name	Lines of Code
2008-01	2008-01	140000
2008-02	2008-02	140500
2008-03	2008-03	100000

Mdx - Normal usage

Software Improvement Group

```
public void normalUsage(MDXDatasource ds) {
  String query =
      "with member [Measures].[Name] as "
    + "            [Time].currentMember.name "
    + "select {[Measures].[Name], "
    + "     [Measures].[LINES_OF_CODE]}"
    + "ON COLUMNS, "
    + "{[Time].Children } "
    + "ON ROWS from [cube]";

  MDXDao.query(ds,query);
}
```

Mdx - Fluent Interface


```
MdxQuery.with()
  .member("Name").onHierarchy("Measures").as()
     .currentMember().name().ofHierarchy("Time").
.onColumns()
    .showFromHierarchy("Measures")
        .member("Name","LINES_OF_CODE")
.onRows()
    .showFromHierarchy("Time")
        .children()
.from("cube");
```

Fluent Interface - broken down


```
WithMember member =
   MdxQuery.with().member("Name")
               .onHierarchy("Measures");
Axis columns =
   member.as().currentMember().name()
               .ofHierarchy("Time").onColumns();
Axis rows =
   columns.showFromHierarchy("Measures")
        .member("Name","LINES_OF_CODE").onRows();
MdxQuery query =
   rows.showFromHierarchy("Time").children()
        .from("cube");
```

Fluent Interface - broken down


```
WithMember member = ...


Axis columns =
   ...

Axis rows =
   ...

MdxQuery query =
    ...
```

Implementation


```
public class Axis {
  protected MdxQuery query;
...
  public Axis(MdxQuery query) {
     this.query = query
  }
...
  public MdxQuery from(String cubeName){
     query.addRows(this);
     query.setName(cubeName);
     return query;
  }
...
}
```

MdxQuery
- 6 classes - 380 LOC

Lessons Learned

SELECT <axis_specification> [, <axis_specification>...]
FROM <cube_specification>
WHERE <slicer_specification>

<axis_specification> ::= <set> ON <axis_name>
<axis_name> ::= COLUMNS | ROWS | PAGES
| SECTIONS | CHAPTERS | AXIS(index)

From: http://msdn2.microsoft.com/en-us/library/ms713682(VS.85).aspx

Lessons Learned

Software Improvement Group

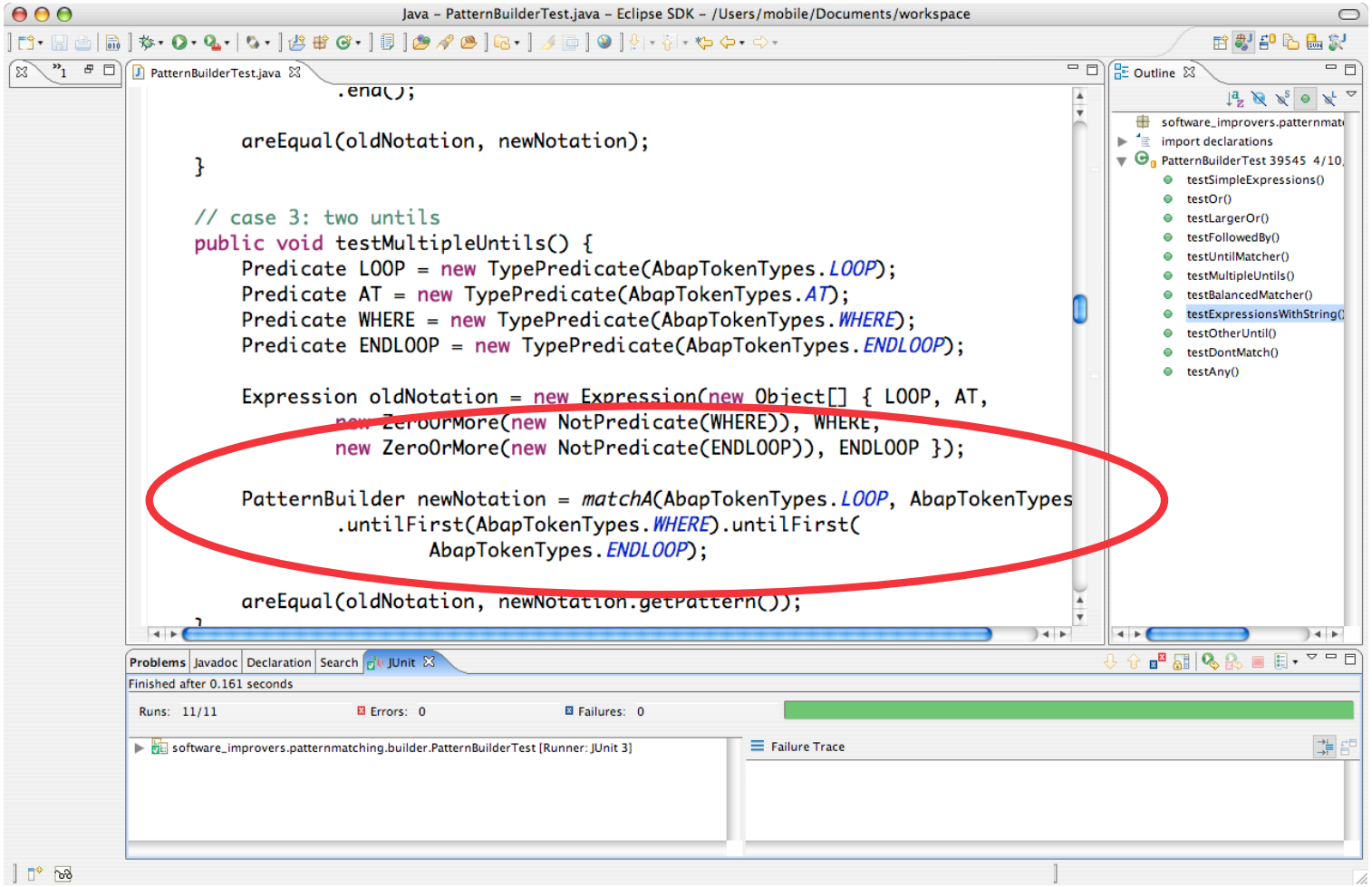
Divide et impera

Overall Lessons


```
import static java.new.feature.VarArgs;
```

Overall Lessons

Software Improvement Group



Ideas

Software Improvement Group

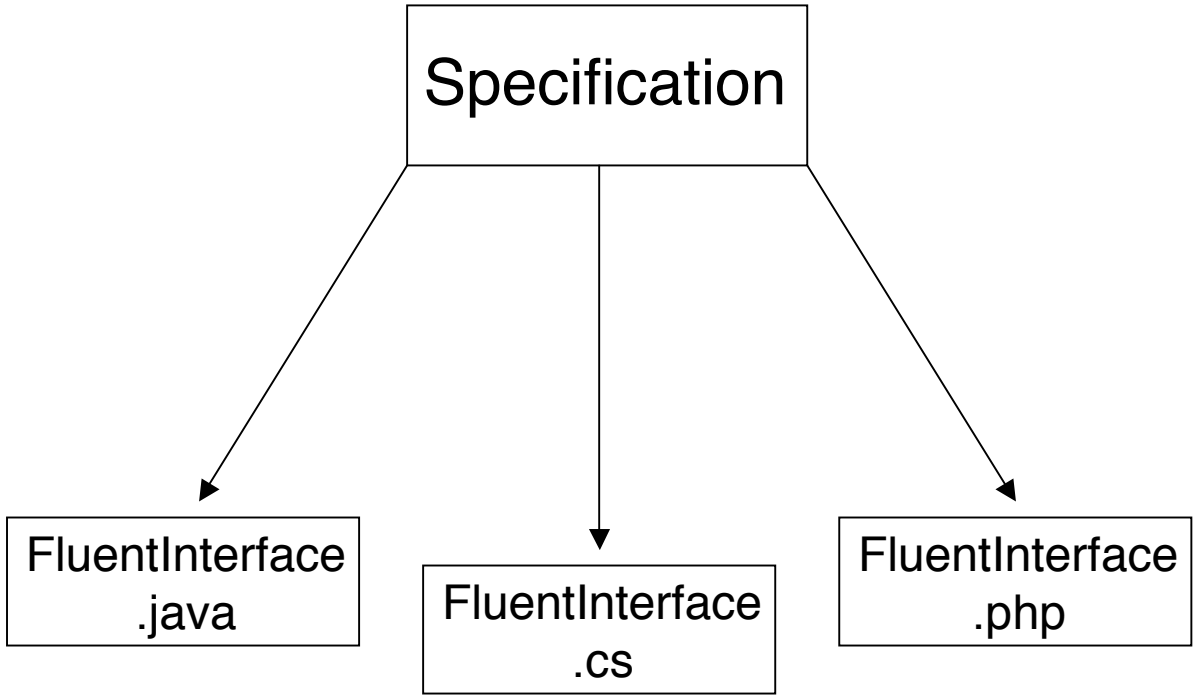
Possibilities

Possibilities


```
public Query valueLike(String userInput){
    escapeEvilThings(userInput);
    updateInternalState(userInput);
    return this;
}
```

Possibilities

Software Improvement Group



Conclusion

- Configuring value objects
- Easy access to library
- Typed embedded languages

Conclusion

- Support everything
- Mix fluent and OO-interfaces

Software Improvement Group

Thank you for your attention

e.bouwers@sig.eu

http://www.sig.eu