# real-world refactoring

**NEAL FORD** software architect / meme wrangler

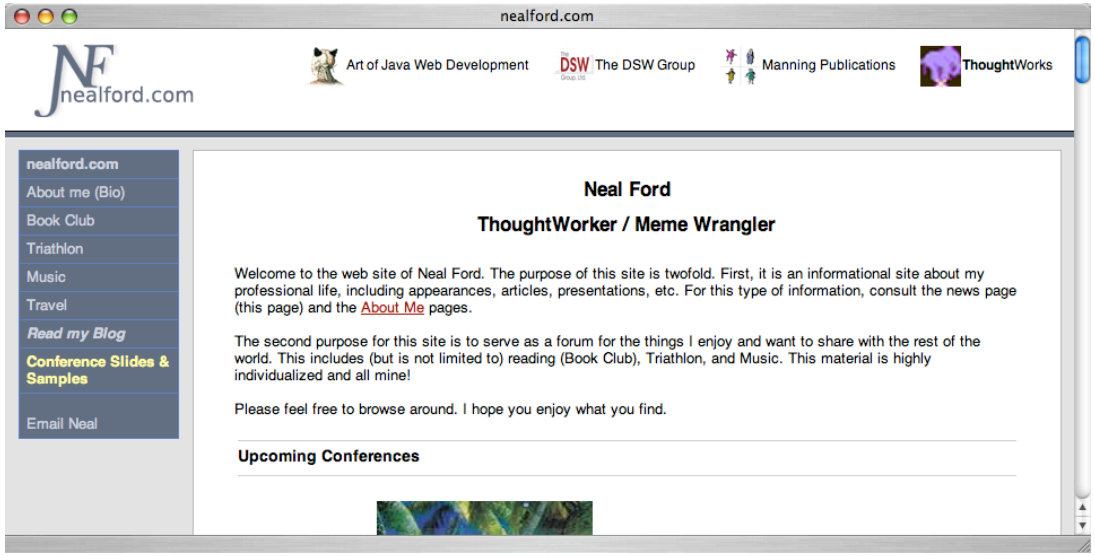
**ThoughtWorks**

**nford@thoughtworks.com**
**3003 Summit Boulevard, Atlanta, GA 30319**
www.nealford.com
www.thoughtworks.com
memeagora.blogspot.com

# housekeeping

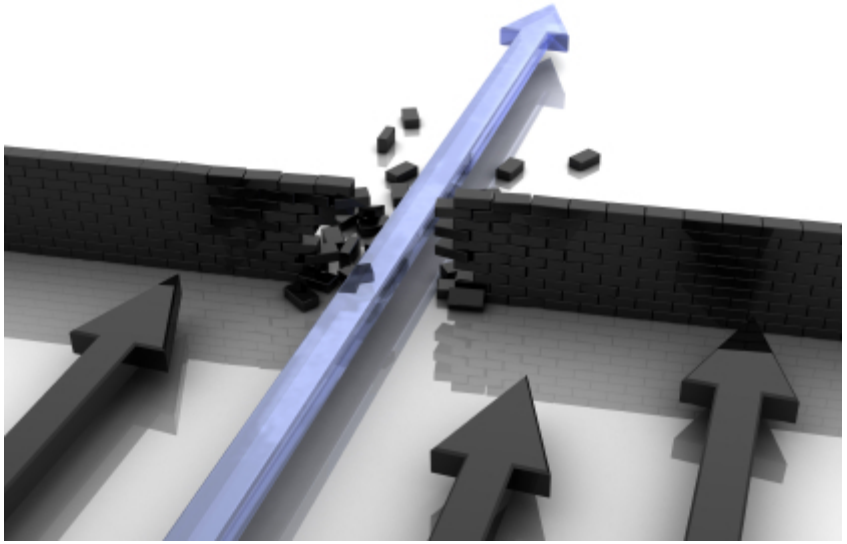
ask questions anytime

download slides from nealford.com




download samples from `github.com/nealford`

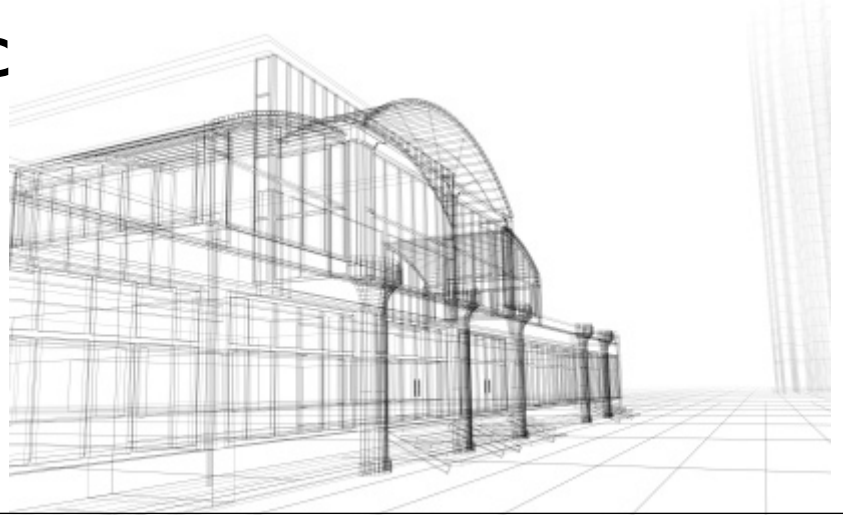
# emergent design family



emergent design &
evolutionary architec

real
world
refactoring

test-driven
design



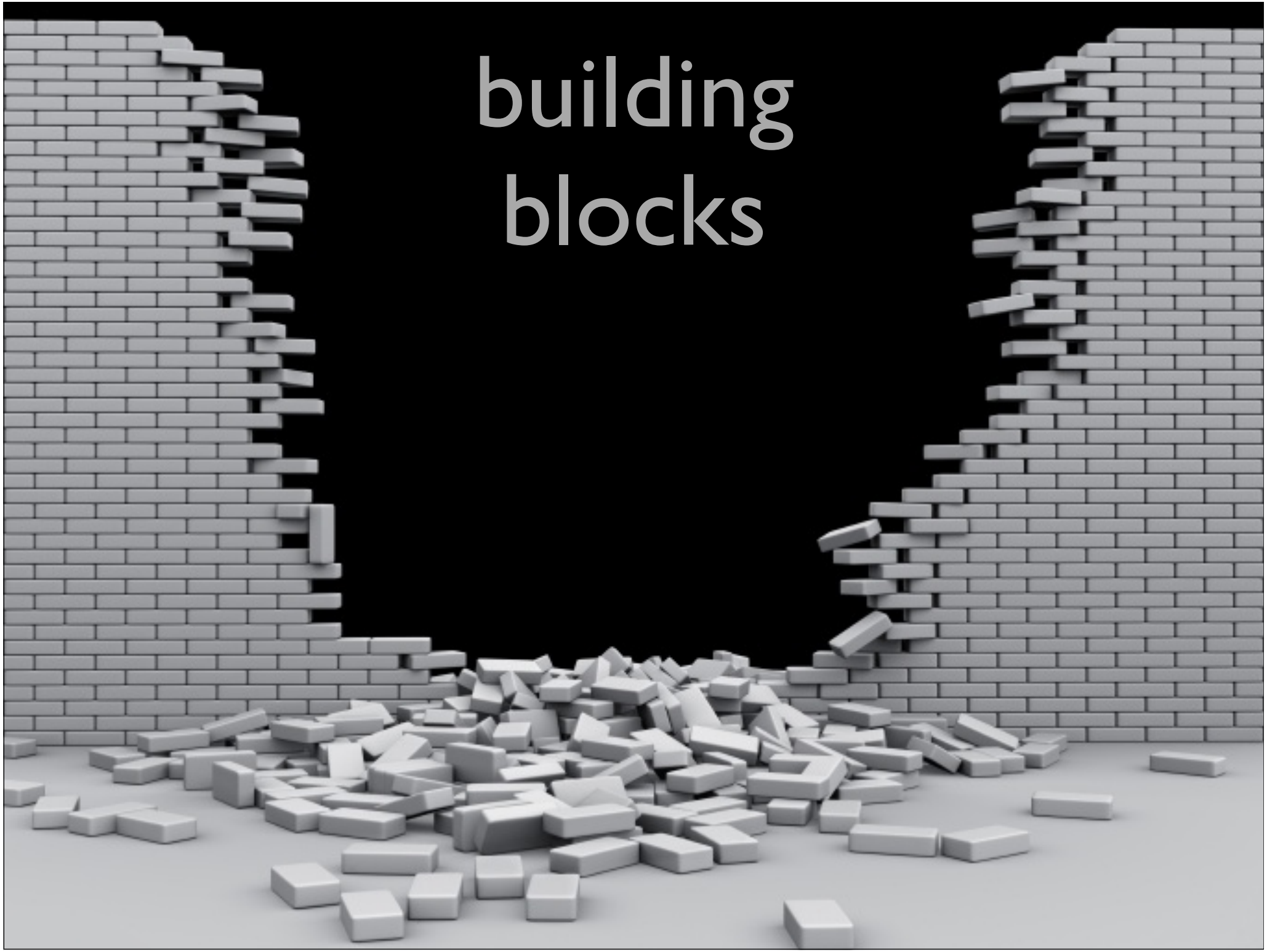
# what i cover

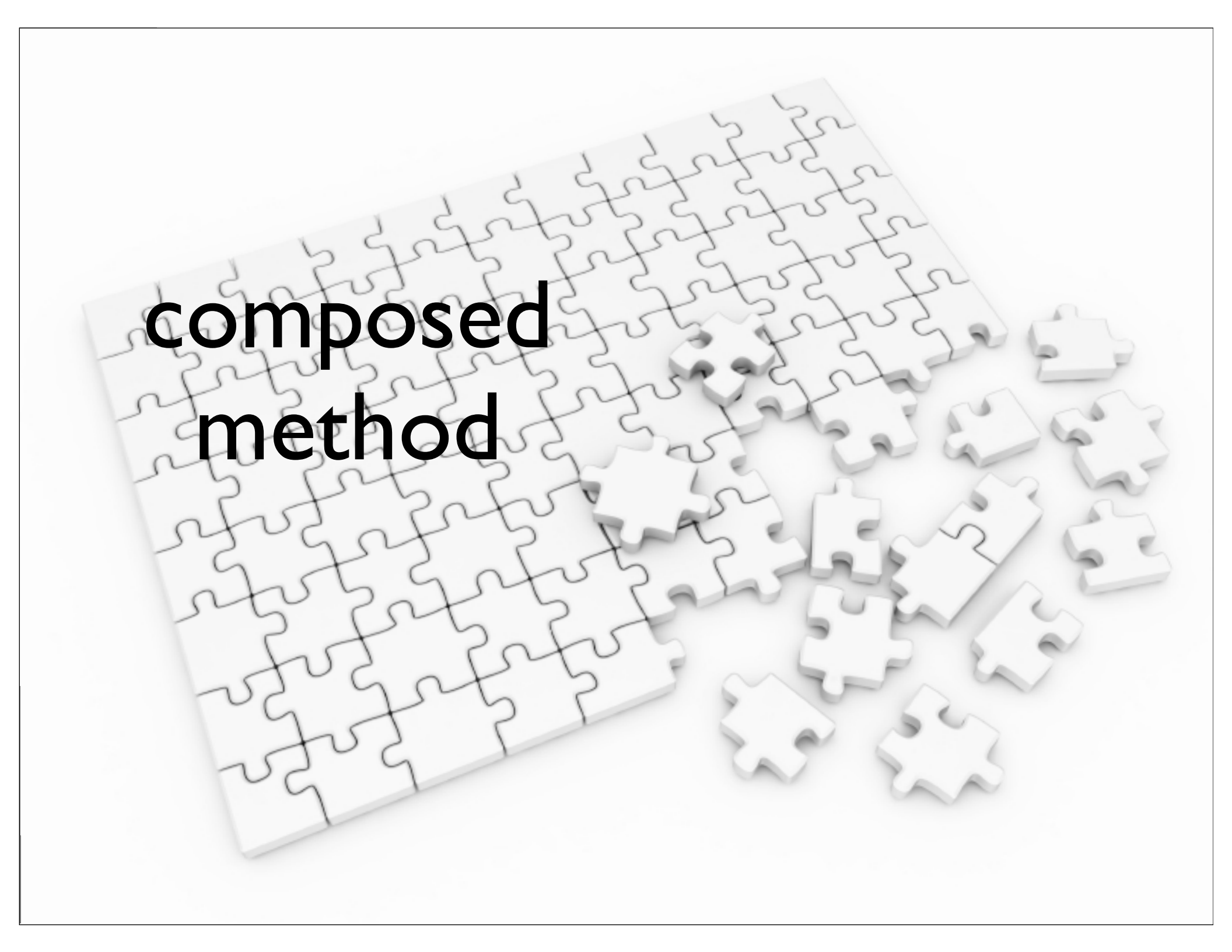
building blocks

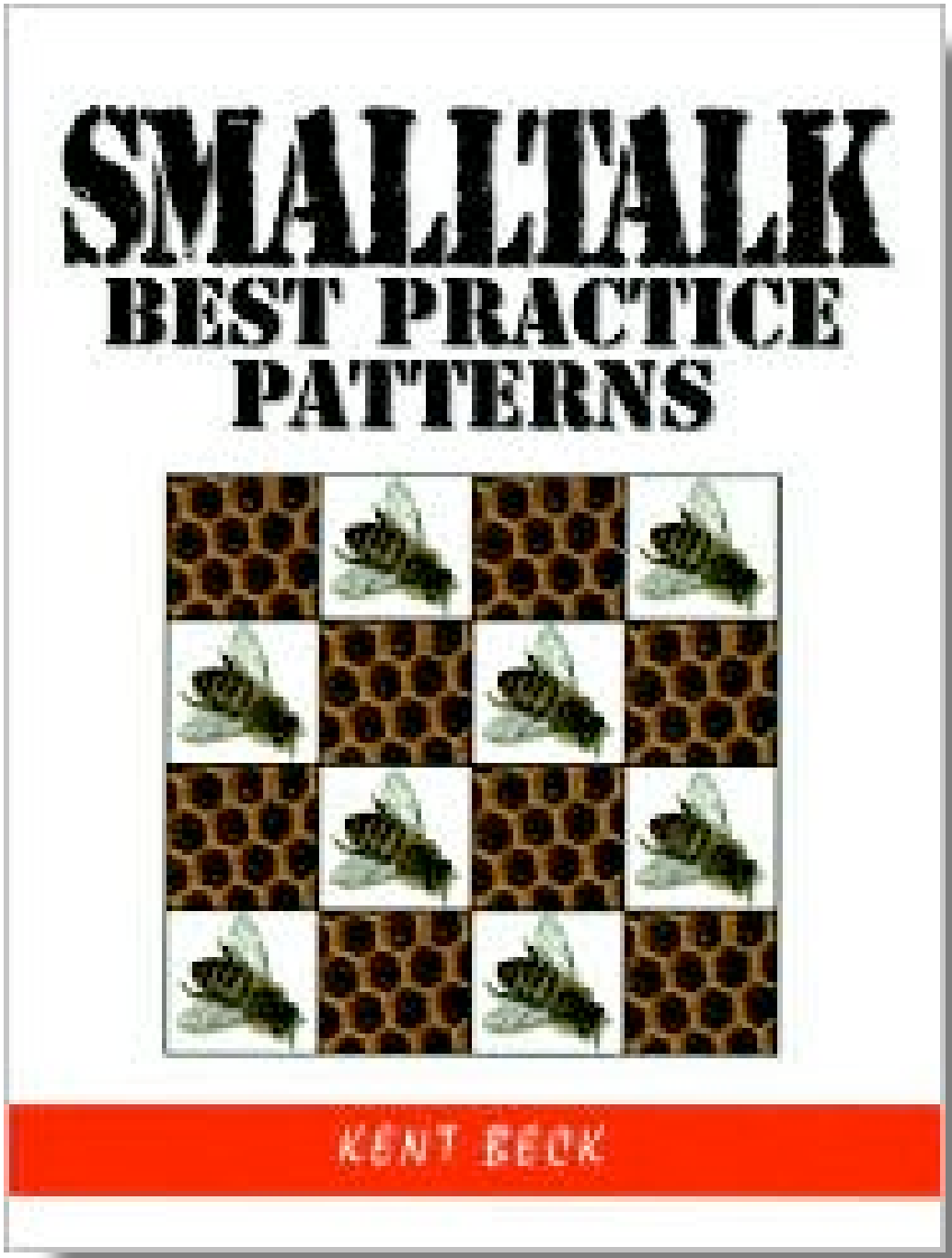
- composed method
- the slap principle

refactoring code, databases, and build files

advice on why & when to refactor


building
blocks


composed
method


SMALLTALK
BEST PRACTICE
PATTERNS
KENT BECK

# composed method

***Divide your program into methods that perform one identifiable task.***

Keep all of the operations in a method at the same level of abstraction.

This will naturally result in programs with many small methods, each a few lines long.

# refactoring to composed method

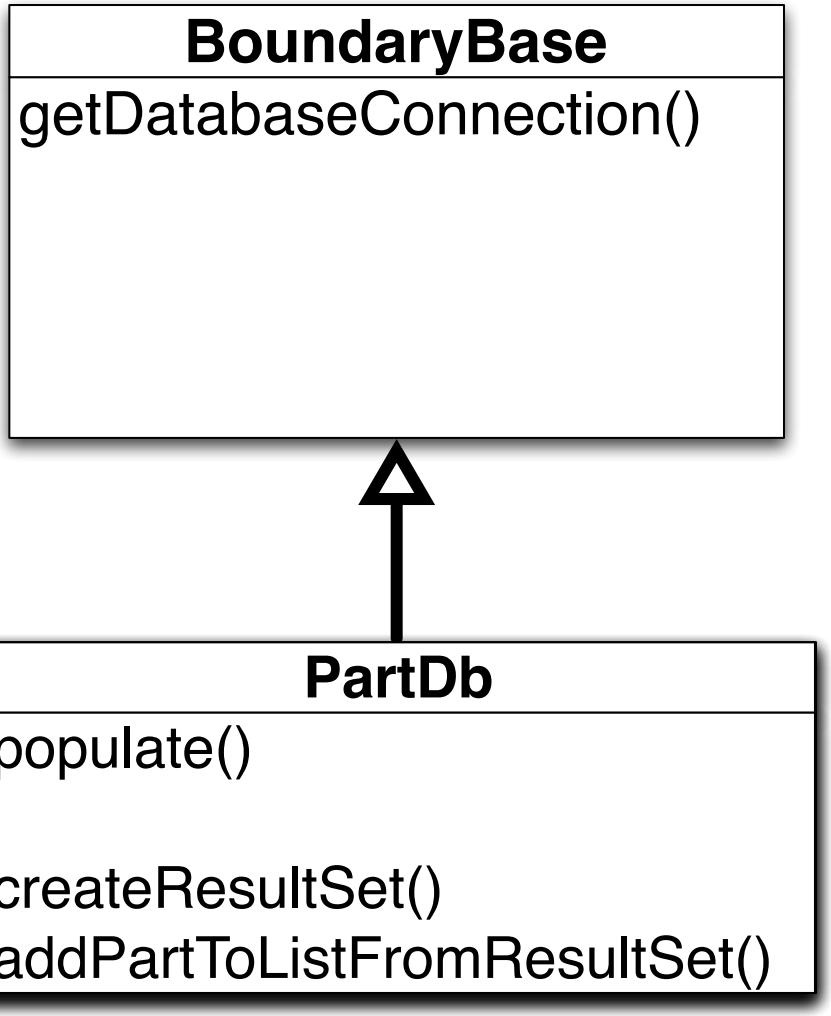
```
public void populate() throws Exception  {
    Connection c = null;
    try {
        Class.forName(DRIVER_CLASS);
        c = DriverManager.getConnection(DB_URL, USER, PASSWORD);
        Statement stmt = c.createStatement();
        ResultSet rs = stmt.executeQuery(SQL_SELECT_PARTS);
        while (rs.next()) {
            Part p = new Part();
            p.setName(rs.getString("name"));
            p.setBrand(rs.getString("brand"));
            p.setRetailPrice(rs.getDouble("retail_price"));
            partList.add(p);
        }
    } finally {
        c.close();
    }
}
```

```
private void addPartToListFromResultSet(ResultSet rs)
        throws SQLException {
    Part p = new Part();
    p.setName(rs.getString("name"));
    p.setBrand(rs.getString("brand"));
    p.setRetailPrice(rs.getDouble("retail_price"));
    partList.add(p);
}
```

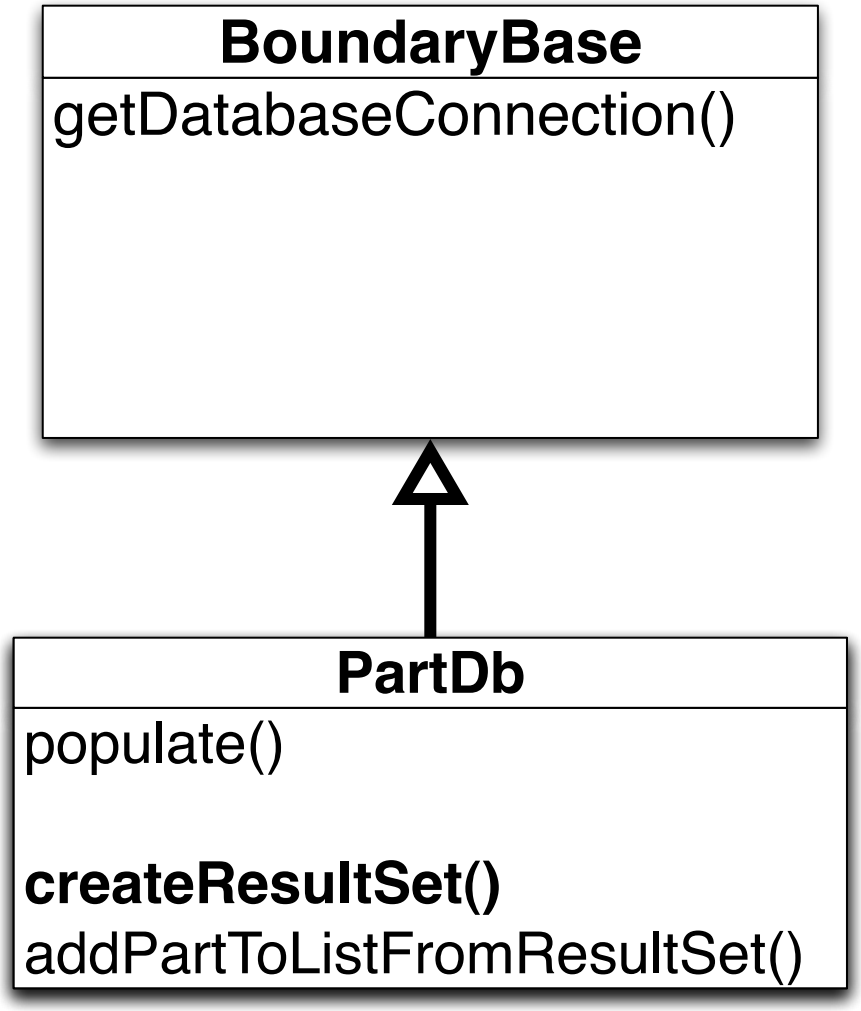
```
public void populate() throws Exception {
    Connection c = null;
    try {
        c = getDatabaseConnection();
        ResultSet rs = createResultSet(c);
        while (rs.next())
            addPartToListFromResultSet(rs);
    } finally {
        c.close();
    }
}
```

```
private ResultSet createResultSet(Connection c)
        throws SQLException {
    return c.createStatement().
            executeQuery(SQL_SELECT_PARTS);
}
```

```
private Connection getDatabaseConnection()
        throws ClassNotFoundException, SQLException {
    Connection c;
    Class.forName(DRIVER_CLASS);
    c = DriverManager.getConnection(DB_URL,
            "webuser", "webpass");
    return c;
}
```




```
private Connection getDatabaseConnection()
        throws ClassNotFoundException, SQLException {
    Connection c;
    Class.forName(DRIVER_CLASS);
    c = DriverManager.getConnection(DB_URL,
            "webuser", "webpass");
    return c;
}
```




```
private ResultSet createResultSet(Connection c)
        throws SQLException {
    return c.createStatement().
            executeQuery(SQL_SELECT_PARTS);
}
```

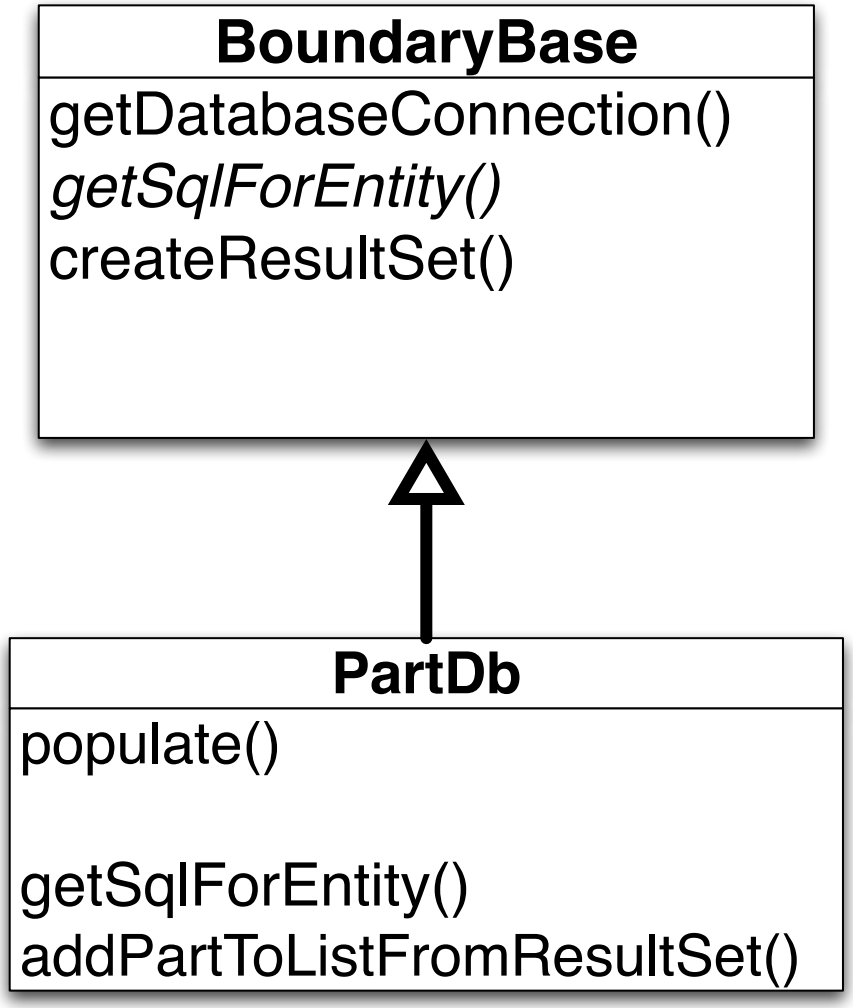
# BoundaryBase

```
abstract protected String getSqlForEntity();

protected ResultSet createResultSet(Connection c) throws SQLException {
    Statement stmt = c.createStatement();
    return stmt.executeQuery(getSqlForEntity());
}
```

```
private ResultSet createResultSet(Connection c)
        throws SQLException {
    return c.createStatement().
            executeQuery(SQL_SELECT_PARTS);
}
```

# PartDb

```
protected String getSqlForEntity() {
    return SQL_SELECT_PARTS;
}
```

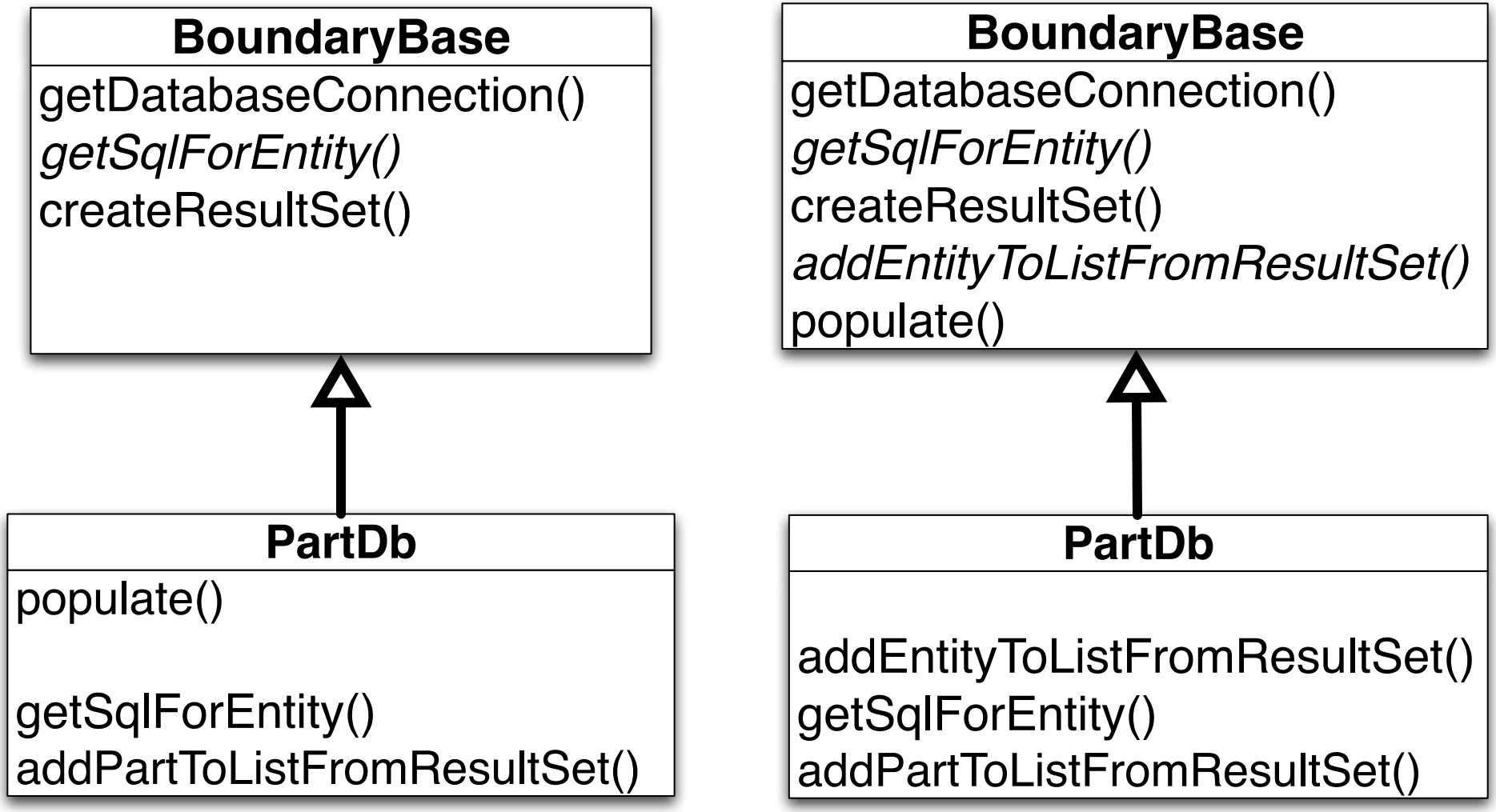

BoundaryBase
getDatabaseConnection()
getSqlForEntity()
createResultSet()
PartDb
populate()
getSqlForEntity()
addPartToListFromResultSet()

old version
(in PartDB)

```
public void populate() throws Exception {
    Connection c = null;
    try {
        c = getDatabaseConnection();
        ResultSet rs = createResultSet(c);
        while (rs.next())
            addPartToListFromResultSet(rs);
    } finally {
        c.close();
    }
}
```

```
abstract protected void addEntityToListFromResultSet(ResultSet rs)
        throws SQLException;

public void populate() throws Exception {
    Connection c = null;
    try {
        c = getDatabaseConnection();
        ResultSet rs = createResultSet(c);
        while (rs.next())
            addEntityToListFromResultSet(rs);
    } finally {
        c.close();
    }
}
```

refactored version
(in BoundaryBase)


BoundaryBase
getDatabaseConnection()
getSqlForEntity()
createResultSet()
PartDb
populate()
getSqlForEntity()
addPartToListFromResultSet()
BoundaryBase
getDatabaseConnection()
getSqlForEntity()
createResultSet()
addEntityToListFromResultSet()
populate()
PartDb
addEntityToListFromResultSet()
getSqlForEntity()
addPartToListFromResultSet()

```
protected Connection getDatabaseConnection() throws ClassNotFoundException,
        SQLException {
    Connection c;
    Class.forName(DRIVER_CLASS);
    c = DriverManager.getConnection(DB_URL, "webuser", "webpass");
    return c;
}

abstract protected String getSqlForEntity();

protected ResultSet createResultSet(Connection c) throws SQLException {
    Statement stmt = c.createStatement();
    return stmt.executeQuery(getSqlForEntity());
}

abstract protected void addEntityToListFromResultSet(ResultSet rs)
        throws SQLException;

public void populate() throws Exception {
    Connection c = null;
    try {
        c = getDatabaseConnection();
        ResultSet rs = createResultSet(c);
        while (rs.next())
            addEntityToListFromResultSet(rs);
    } finally {
        c.close();
    }
}
```

BoundaryBase

# PartDb

```
public Part[] getParts() {
    return (Part[]) partList.toArray(TEMPLATE);
}

protected String getSqlForEntity() {
    return SQL_SELECT_PARTS;
}

protected void addEntityToListFromResultSet(ResultSet rs) throws SQLException {
    Part p = new Part();
    p.setName(rs.getString("name"));
    p.setBrand(rs.getString("brand"));
    p.setRetailPrice(rs.getDouble("retail_price"));
    partList.add(p);
}
```

# benefits of composed method

shorter methods easier to test

method names become documentation

large number of very cohesive methods

discover reusable assets that you didn't know were there

# SLAP

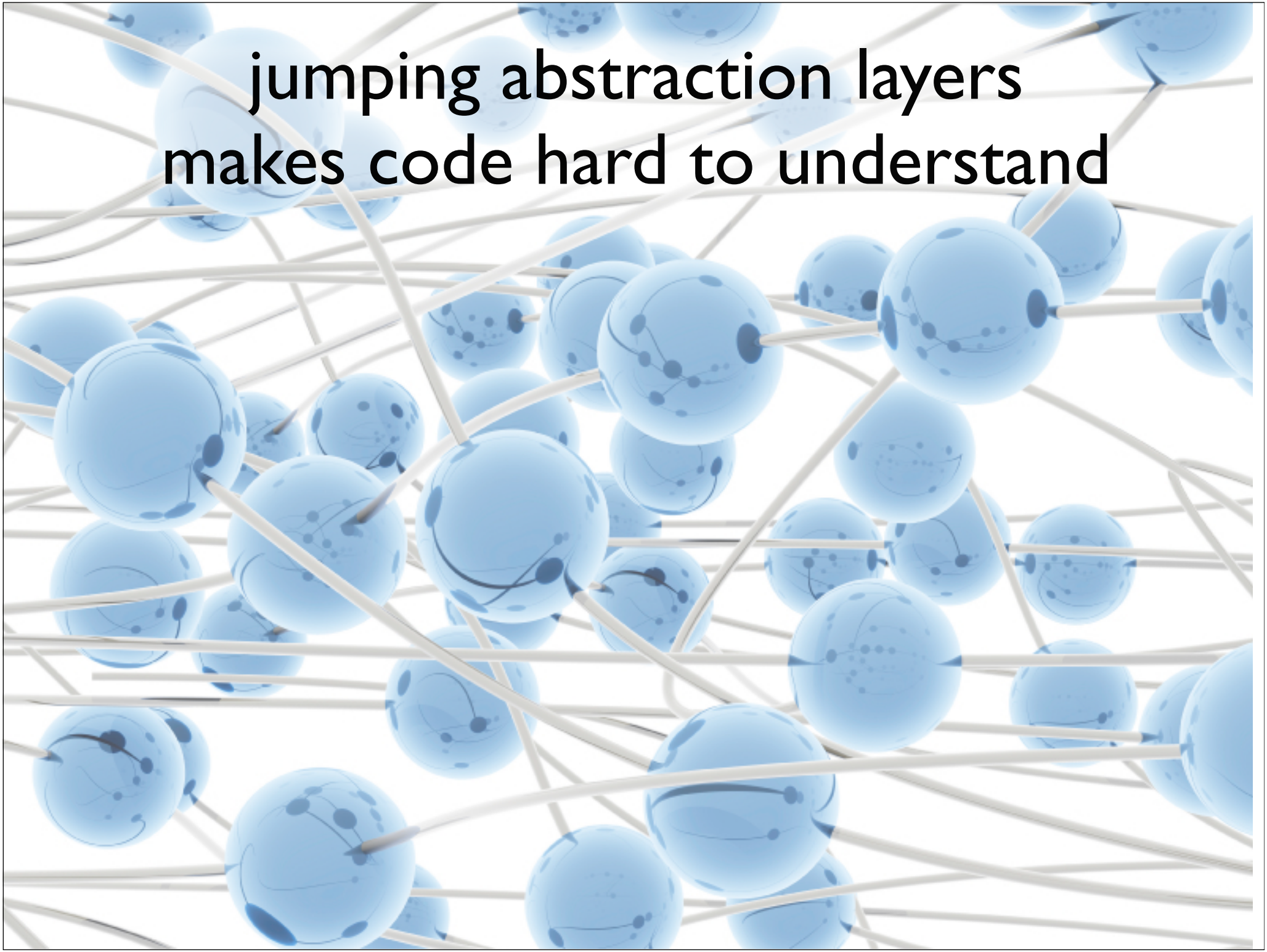
single level of abstraction principle

# composed method

Divide your program into methods that perform one identifiable task.

***Keep all of the operations in a method at the same level of abstraction.***

This will naturally result in programs with many small methods, each a few lines long.


jumping abstraction layers
makes code hard to understand

```
public void addOrder(final ShoppingCart cart, String userName,
                     Order order) throws SQLException {
    Connection c = null; PreparedStatement ps = null;
    Statement s = null; ResultSet rs = null;
    boolean transactionState = false;
    try {
        c = dbPool.getConnection();
        s = c.createStatement();
        transactionState = c.getAutoCommit();
        int userKey = getUserKey(userName, c, ps, rs);
        c.setAutoCommit(false);
        addSingleOrder(order, c, ps, userKey);
        int orderKey = getOrderKey(s, rs);
        addLineItems(cart, c, orderKey);
        c.commit();
        order.setOrderKey(orderKey);
    } catch (SQLException sqlx) {
        s = c.createStatement();
        c.rollback();
        throw sqlx;
    } finally {
        try {
            c.setAutoCommit(transactionState);
            dbPool.release(c);
            if (s != null)  s.close();
            if (ps != null) ps.close();
            if (rs != null) rs.close();
        } catch (SQLException ignored) {
        }
    }
}
```

```
public void addOrder(final ShoppingCart cart, String userName,
                     Order order) throws SQLException {
    Connection connection = null; PreparedStatement ps = null;
    Statement statement = null; ResultSet rs = null;
    boolean transactionState = false;
    try {
        connection = dbPool.getConnection();
        statement = connection.createStatement();
        transactionState = setupTransactionStateFor(connection, transactionState);
        addSingleOrder(order, connection, ps, userKeyFor(userName, connection));
        order.setOrderKey(generateOrderKey(statement, rs));
        addLineItems(cart, connection, order.getOrderKey());
        completeTransaction(connection);
    } catch (SQLException sqlx) {
        rollbackTransactionFor(connection);
        throw sqlx;
    } finally {
        cleanUpDatabaseResources(connection, transactionState, statement, ps, rs);
    }
}
```

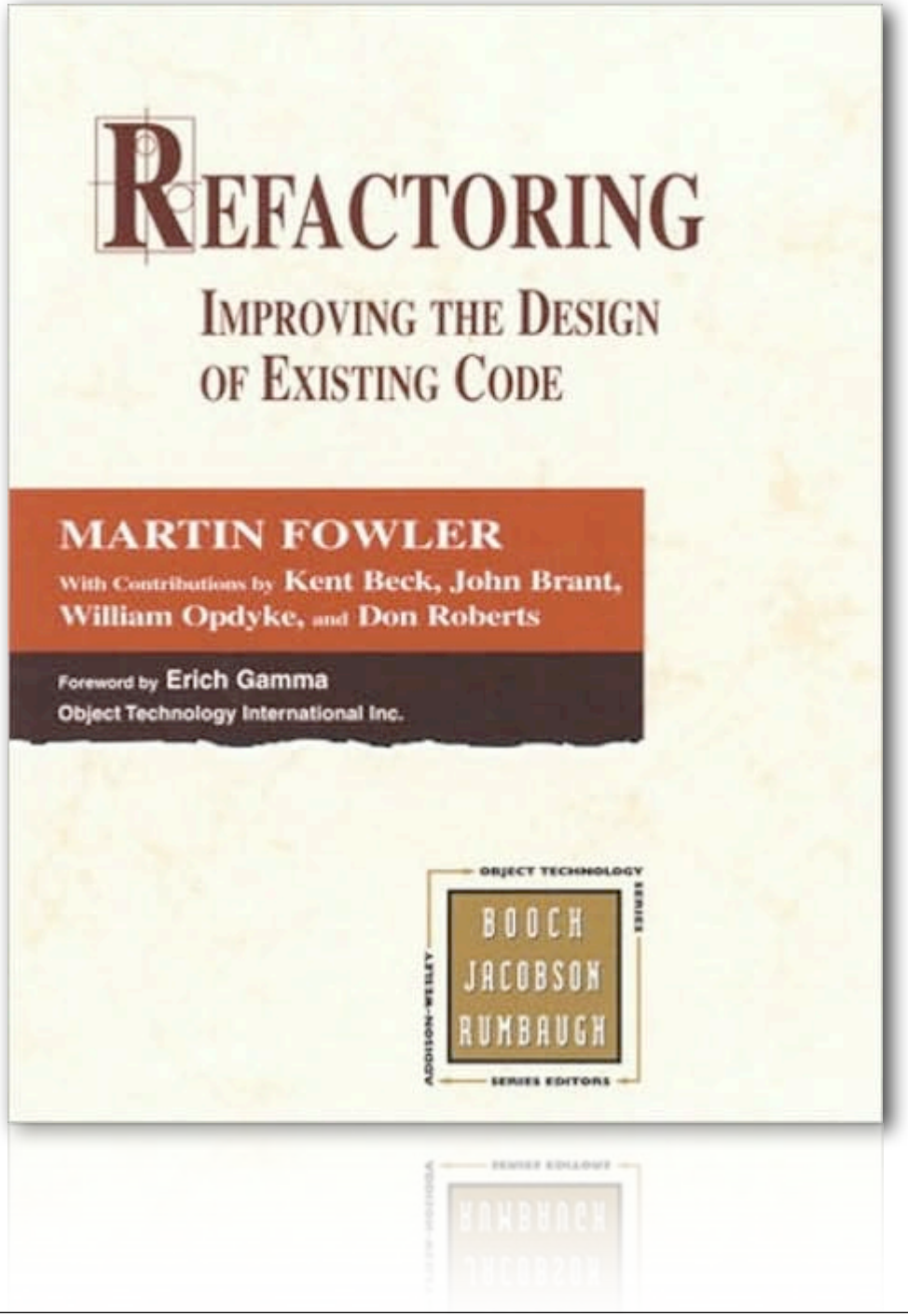
```
public void addOrderFrom(ShoppingCart cart, String userName,
                         Order order) throws SQLException {
    setupDataInfrastructure();
    try {
        add(order, userKeyBasedOn(userName));
        addLineItemsFrom(cart, order.getOrderKey());
    } catch (SQLException sqlx) {
        rollbackTransaction();
        throw sqlx;
    } finally {
        cleanUp();
    }
}
```

```
private void completeTransaction() throws SQLException {
    ((Connection) _db.get("connection")).commit();
}
```

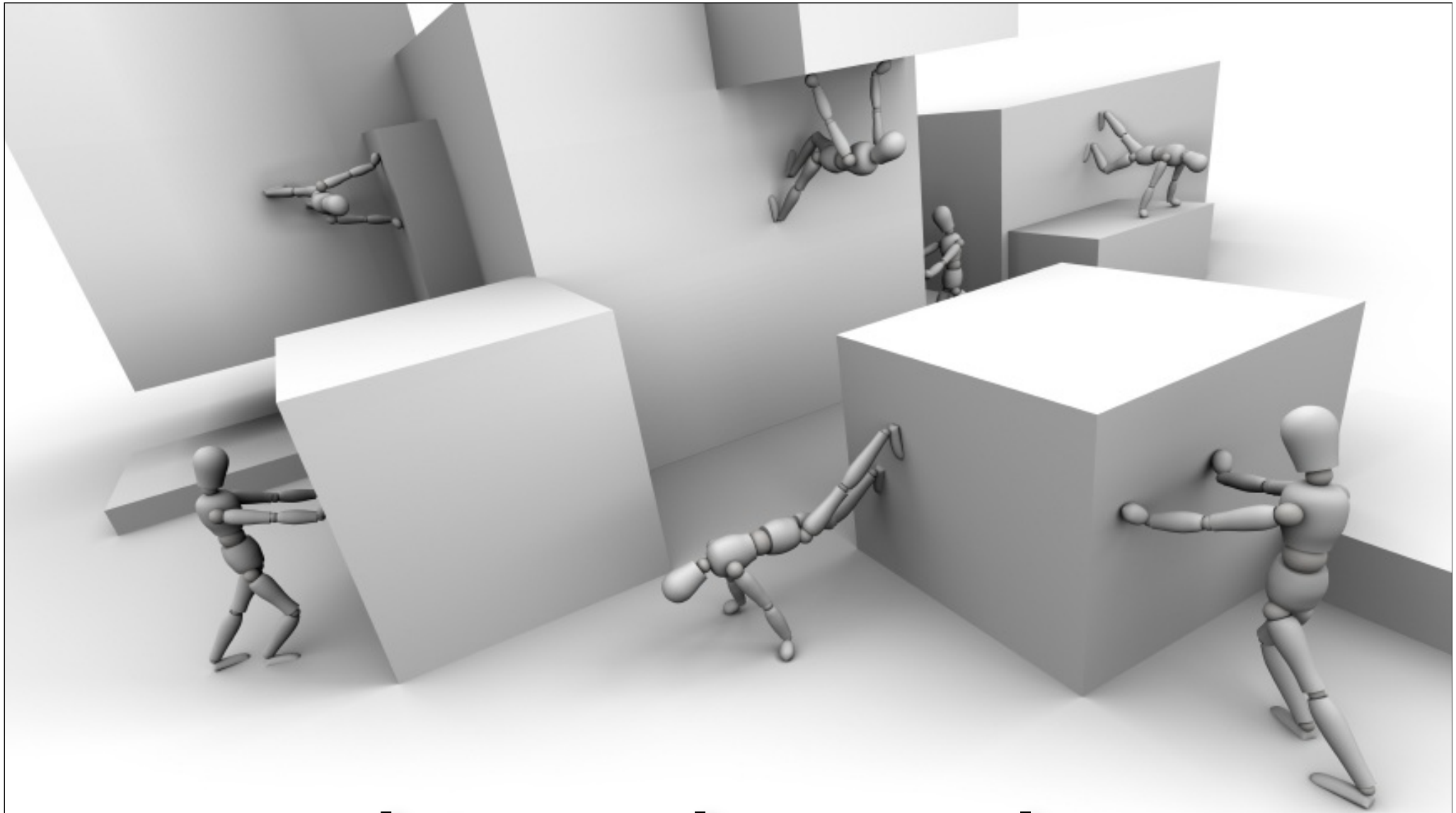
```
public void addOrder(ShoppingCart cart, String userName,
                     Order order) throws SQLException {
    Connection c = null;
    PreparedStatement ps = null;
    Statement s = null;
    ResultSet rs = null;
    boolean transactionState = false;
    try {
        s = c.createStatement();
        transactionState = c.getAutoCommit();
        int userKey = getUserKey(userName, c, ps, rs);
        c.setAutoCommit(false);
        addSingleOrder(order, c, ps, userKey);
        int orderKey = getOrderKey(s, rs);
        addLineItems(cart, c, orderKey);
        c.commit();
        order.setOrderKeyFrom(orderKey);
    } catch (SQLException sqlx) {
        s = c.createStatement();
        c.rollback();
        throw sqlx;
    } finally {
        try {
            c.setAutoCommit(transactior
            dbPool.release(c);
            if (s != null)
                s.close();
            if (ps != null)
                ps.close();
            if (rs != null)
                rs.close();
        } catch (SQLException ignored)
        }
    }
}
```

```
public void addOrderFrom(ShoppingCart cart, String userName,
                         Order order) throws SQLException {
    setupDataInfrastructure();
    try {
        add(order, userKeyBasedOn(userName));
        addLineItemsFrom(cart, order.getOrderKey());
        completeTransaction();
    } catch (SQLException sqlx) {
        rollbackTransaction();
        throw sqlx;
    } finally {
        cleanUp();
    }
}
```

# code




*Refactoring is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior.*



# things that make refactoring hard

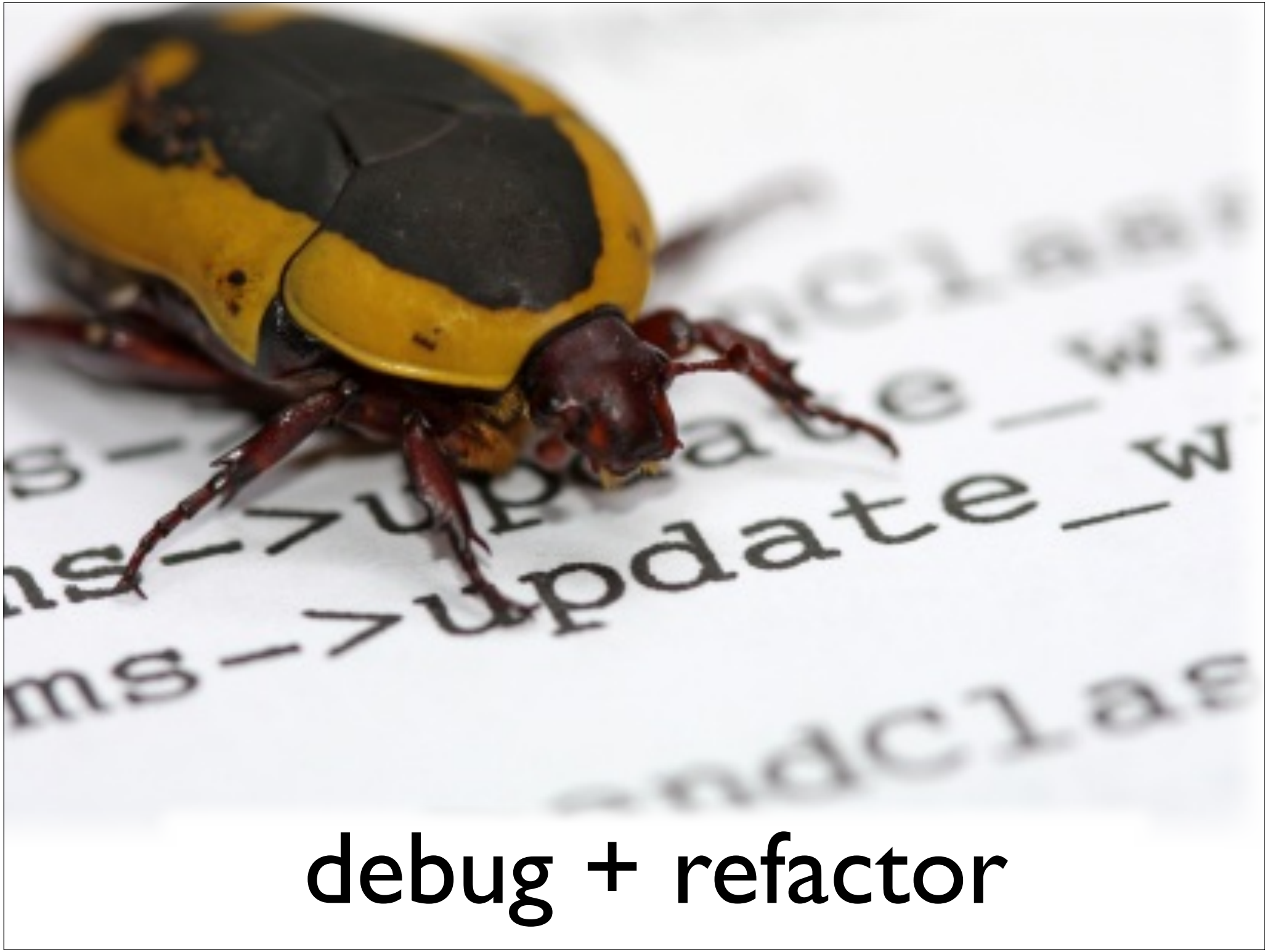

WENGER


fat, ugly methods

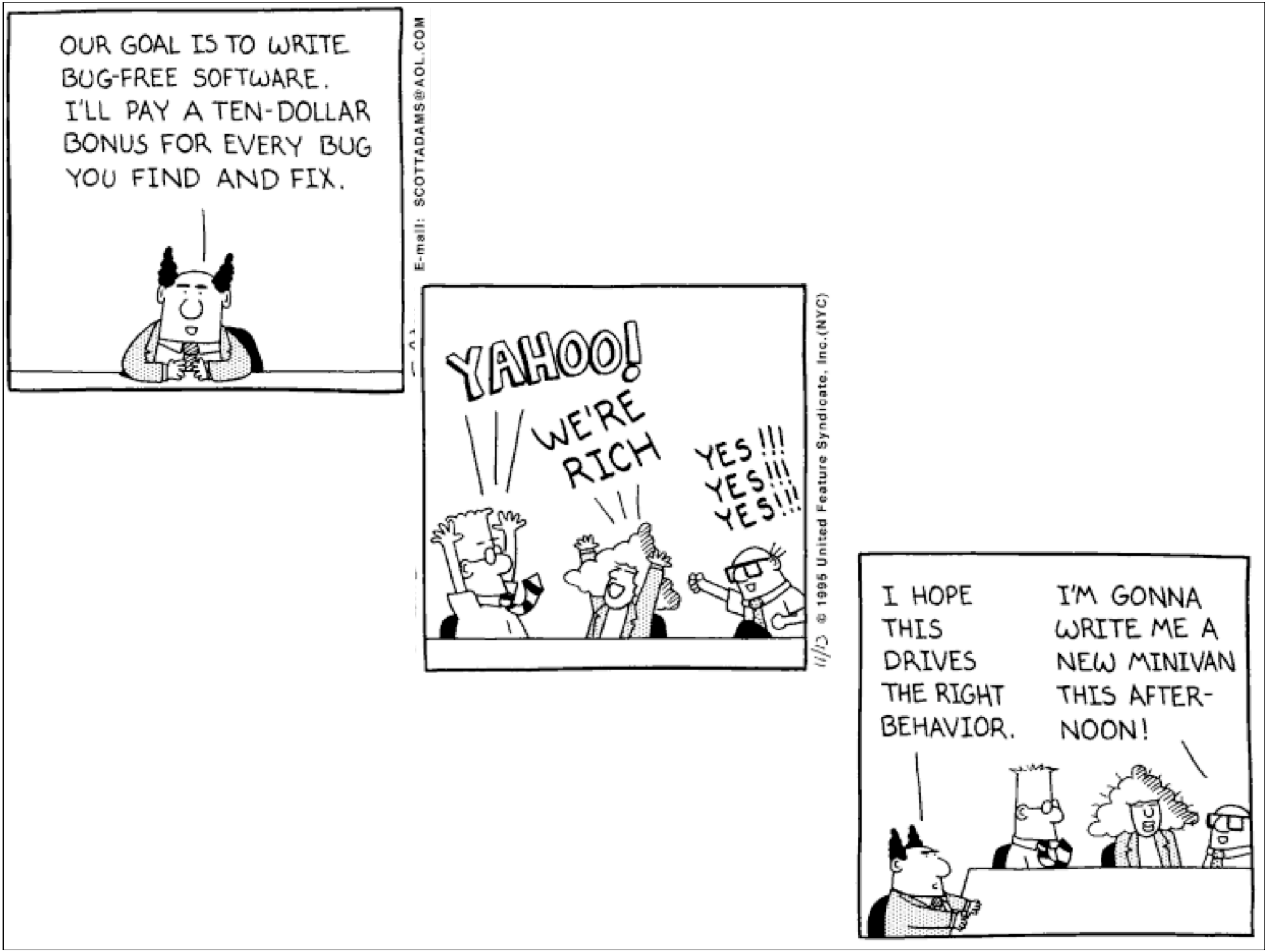
```
public void evaluateExtraParams() {
    super.evaluateExtraParams();

    //Object doubleName = null;

    if (emptyOption != null) {
        addParameter("emptyOption", findValue(emptyOption, Boolean.class));
    }

    if (multiple != null) {
        addParameter("multiple", findValue(multiple, Boolean.class));
    }

    if (size != null) {
        addParameter("size", findString(size));
    }

    if ((headerKey != null) && (headerValue != null)) {
        addParameter("headerKey", findString(headerKey));
        addParameter("headerValue", findString(headerValue));
    }


    if (doubleMultiple != null) {
        addParameter("doubleMultiple", findValue(doubleMultiple, Boolean.class));
    }

    if (doubleSize != null) {
```

from Struts 2.0.11,

**`evaluateExtraParms()`**, 178 lines




# debug + refactor


OUR GOAL IS TO WRITE BUG-FREE SOFTWARE. I'LL PAY A TEN-DOLLAR BONUS FOR EVERY BUG YOU FIND AND FIX.
E-mail: SCOTTADAMS@AOL.COM
YAHOO!
WE'RE RICH
YES !!! YES !!! YES !!!
© 1995 United Feature Syndicate, Inc.(NYC)
I HOPE THIS DRIVES THE RIGHT BEHAVIOR.
I'M GONNA WRITE ME A NEW MINIVAN THIS AFTER-NOON!

# the problem:

aging code base

no tests

lots of bugs

strong desire to refactor...

...plus gut-wrenching fear

# lots of long untested methods

you want to refactor them but...

...you're afraid of what will happen

1. refactor to composed method

2. write tests for the (now) smaller methods

3. write tests when debugging or adding new features

# refactoring attack

refactor to composed method using extract method

(you're debugging anyway)

once you extract the buggy code...

...write tests for it

tests grow around most fragile code first

# how to I get started writing unit tests?

break long methods into small pieces using composed methods

draw a line in the sand: starting next thursday, out code coverage will always rise

every time you find a bug, write a test

write a test for every new feature

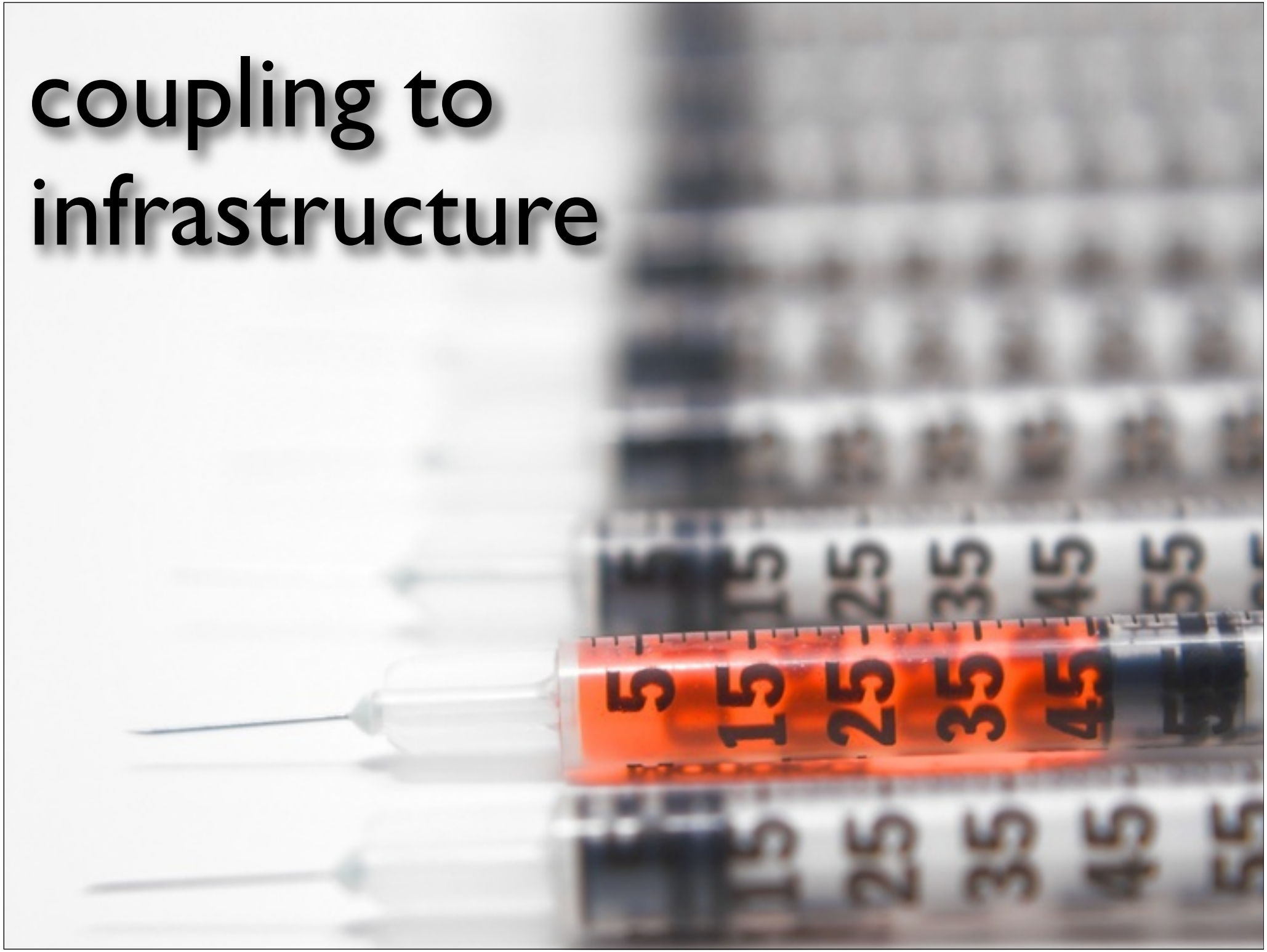
you will slowly grow tests around the part of you code that needs it the most

# decomposition

just because it's long
doesn't mean it's wrong

look for *accidental* longness

# coupling to infrastructure



# coupling to infrastructure

don't tie yourself into infrastructure

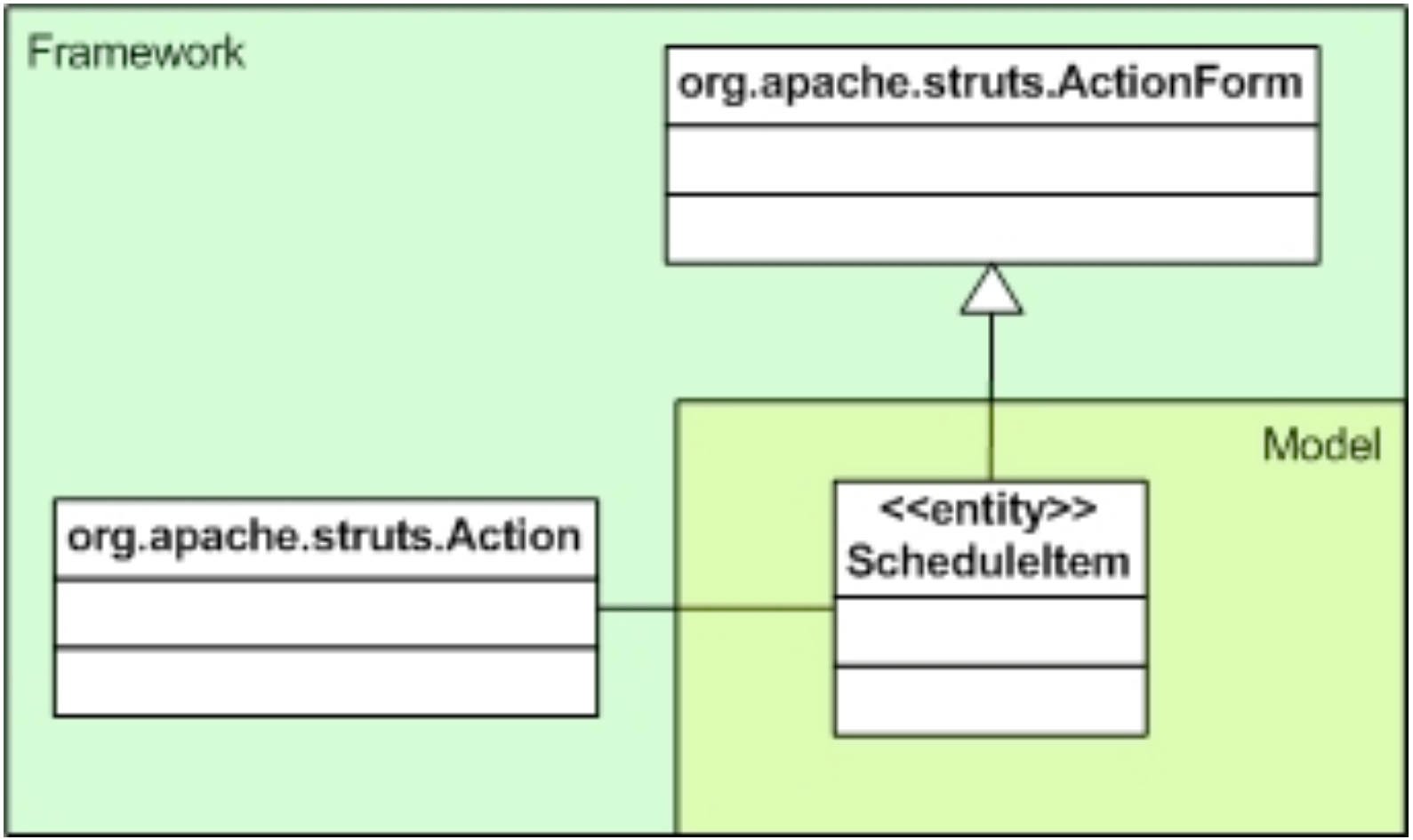
```
import com.giantvendor.seductiveclasses.*
```

pay attention to dependencies

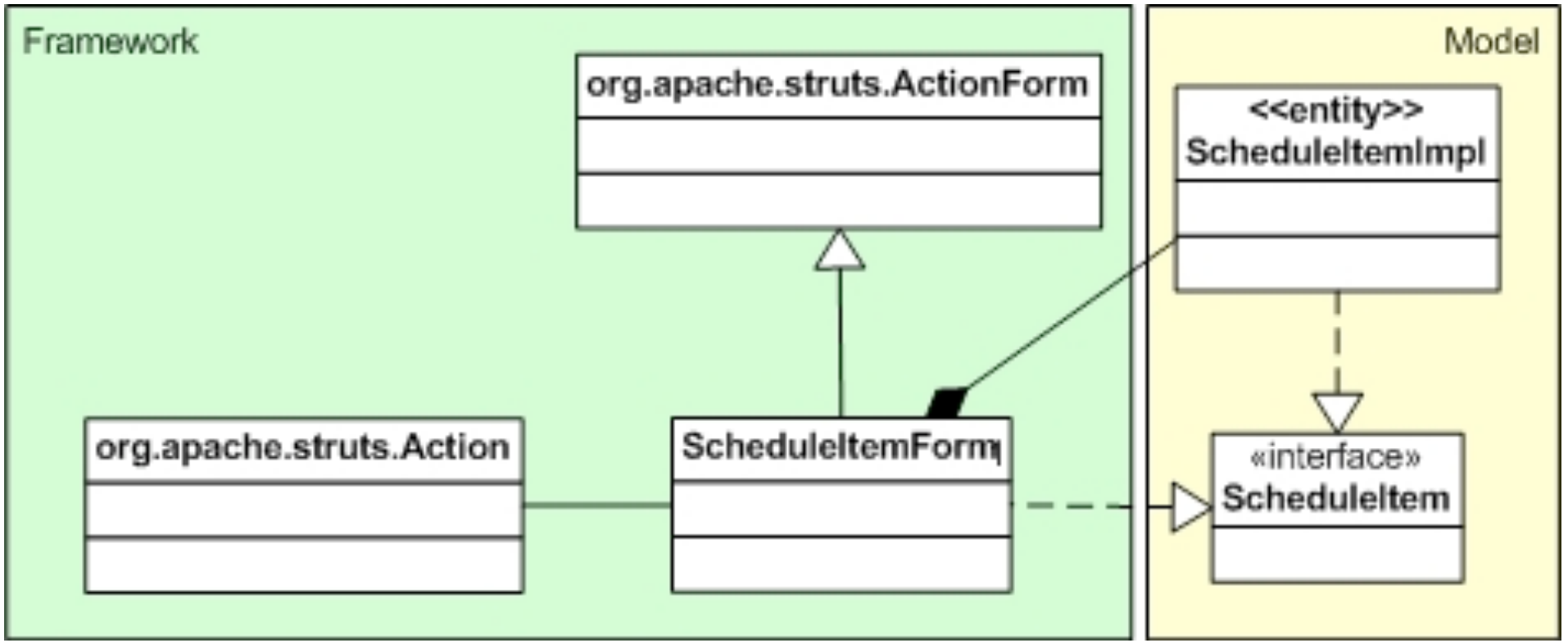
don't extend library/framework classes

*compose instead*

# struts ActionForm



# decoupling from struts



# DRY violations

# copy & paste

# cpd

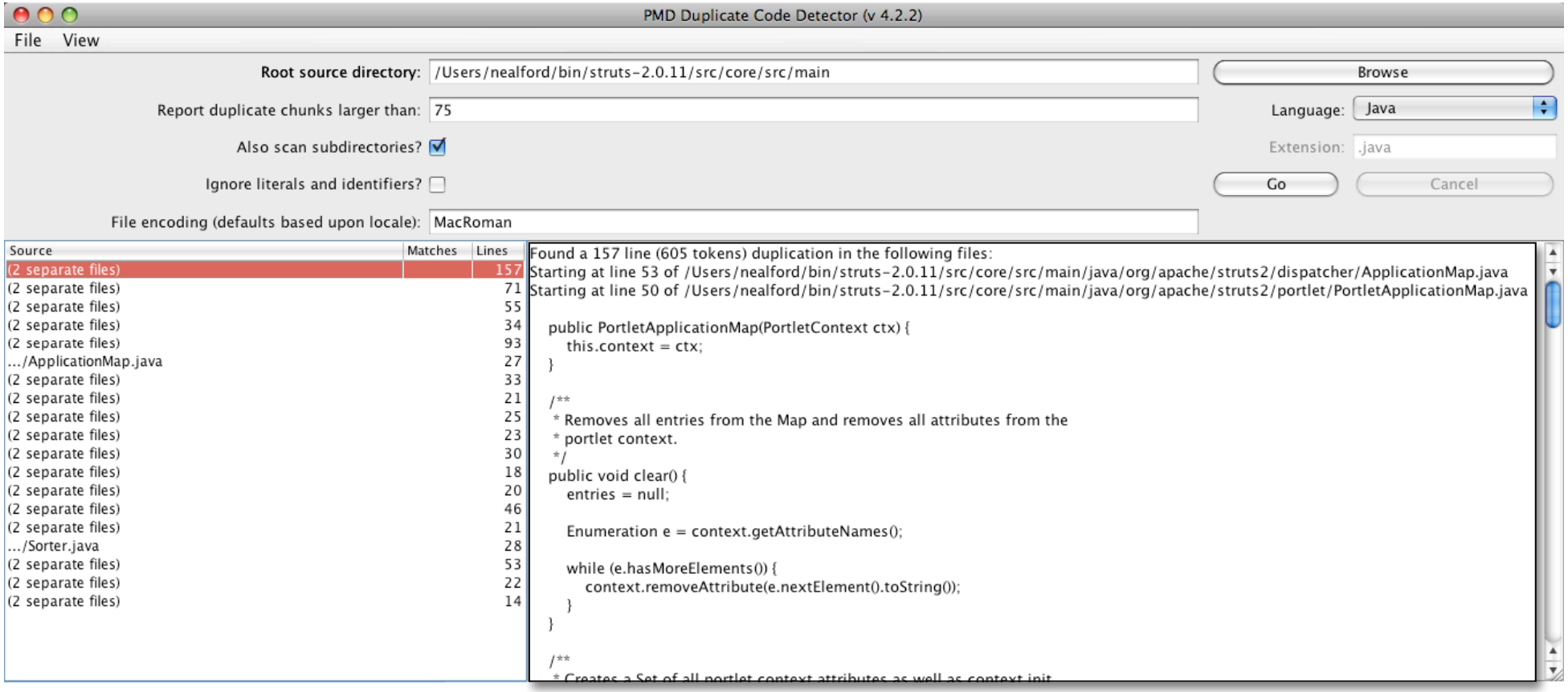
part of the source-code analysis tool **pmd**

configurable window of number of duplicate tokens

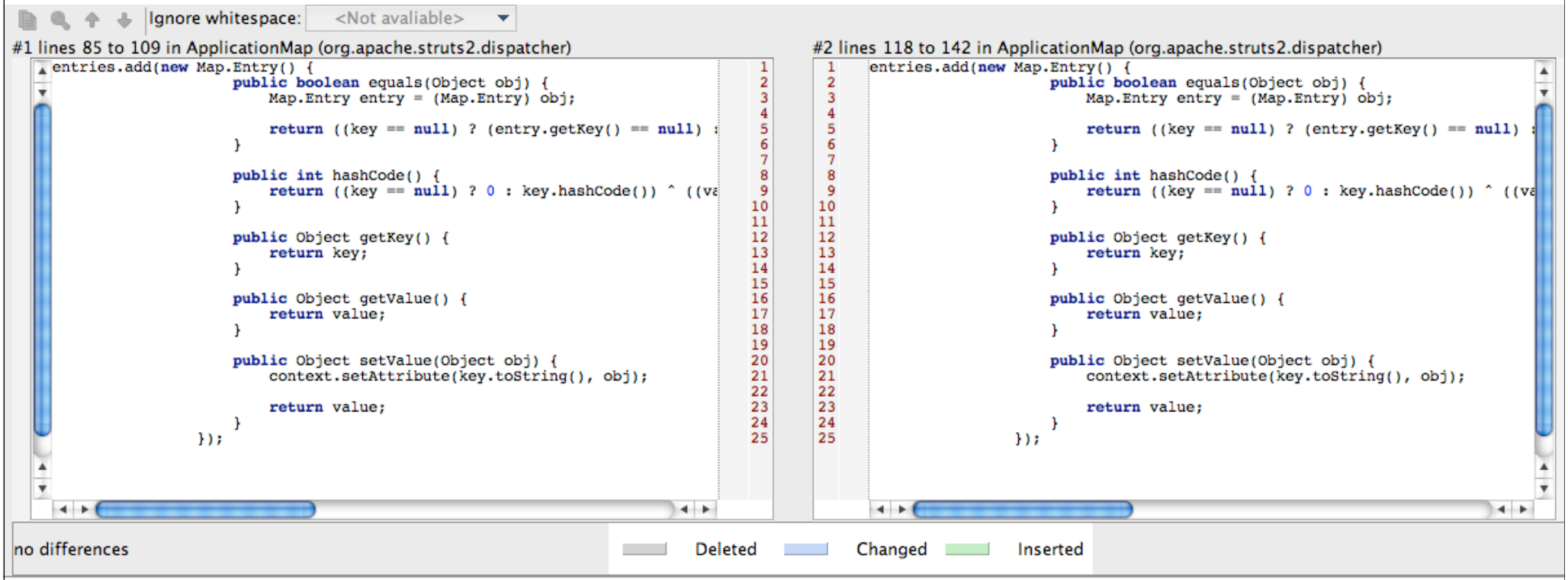
pre-configured with several languages

easy to add new language support

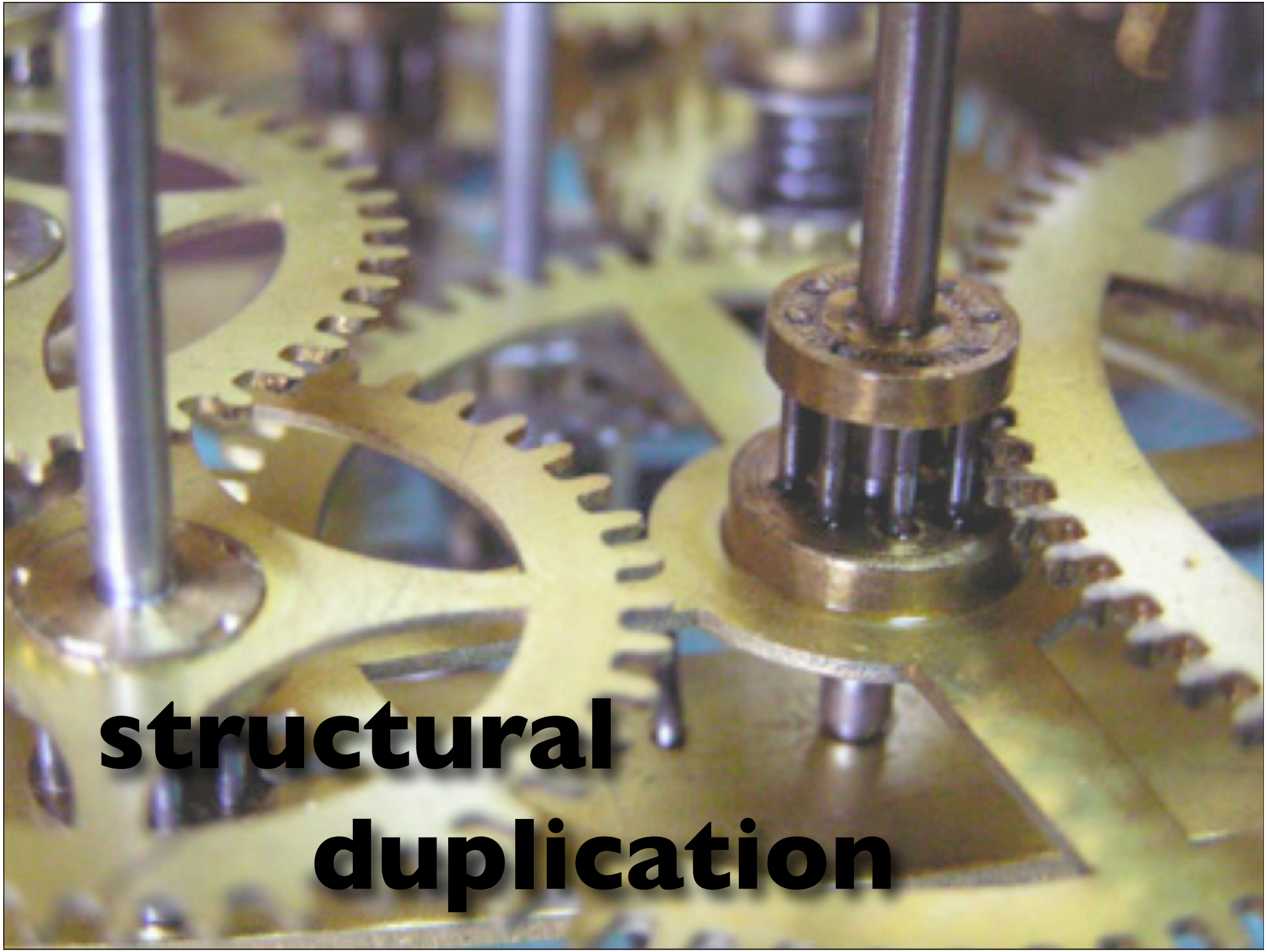
also simian (commercial)


PMD Duplicate Code Detector (v 4.2.2)
File View
Root source directory: /Users/nealford/bin/struts-2.0.11/src/core/src/main
Browse
Report duplicate chunks larger than: 75
Language: Java
Also scan subdirectories?
Extension: .java
Ignore literals and identifiers?
Go
Cancel
File encoding (defaults based upon locale): MacRoman
Source Matches Lines
(2 separate files) 157
(2 separate files) 71
(2 separate files) 55
(2 separate files) 34
(2 separate files) 93
.../ApplicationMap.java 27
(2 separate files) 33
(2 separate files) 21
(2 separate files) 25
(2 separate files) 23
(2 separate files) 30
(2 separate files) 18
(2 separate files) 20
(2 separate files) 46
(2 separate files) 21
.../Sorter.java 28
(2 separate files) 53
(2 separate files) 22
(2 separate files) 14
Found a 157 line (605 tokens) duplication in the following files:
Starting at line 53 of /Users/nealford/bin/struts-2.0.11/src/core/src/main/java/org/apache/struts2/dispatcher/ApplicationMap.java
Starting at line 50 of /Users/nealford/bin/struts-2.0.11/src/core/src/main/java/org/apache/struts2/portlet/PortletApplicationMap.java
public PortletApplicationMap(PortletContext ctx) {
this.context = ctx;
}
/**
* Removes all entries from the Map and removes all attributes from the
* portlet context.
*/
public void clear() {
entries = null;
Enumeration e = context.getAttributeNames();
while (e.hasMoreElements()) {
context.removeAttribute(e.nextElement().toString());
}
}
/**
* Creates a Set of all portlet context attributes as well as context init

# intellij's duplication view



```
while (enumeration.hasMoreElements()) {
    final String key = enumeration.nextElement().toString();
    final Object value = context.getInitParameter(key);
    entries.add(new Map.Entry() {
        public boolean equals(Object obj) {
            Map.Entry entry = (Map.Entry) obj;

            return ((key == null) ?
                (entry.getKey() == null) :
                key.equals(entry.getKey())) && ((value == null) ?
                    (entry.getValue() == null) :
                    value.equals(entry.getValue()));
        }

        public int hashCode() {
            return ((key == null) ? 0 : key.hashCode()) ^ ((value == null) ? 0 : value.hashCode());
        }

        public Object getKey() {
            return key;
        }

        public Object getValue() {
            return value;
        }

        public Object setValue(Object obj) {
            context.setAttribute(key.toString(), obj);

            return value;
        }
    });
}
```


structural
duplication

# given:

```
public class Employee {
    private String name;
    private int salary;
    private int hireYear;

    public Employee(String name, int salary, int hireYear) {
        this.name = name;
        this.salary = salary;
        this.hireYear = hireYear;
    }

    public String getName() { return name; }
    public int getSalary() { return salary;}
    public int getHireYear() { return hireYear; }
}
```

goal:
sort on
any property

# comparator mania!

```
public class EmployeeNameComparator implements Comparator<Employee> {
    public int compare(Employee emp1, Employee emp2) {
        return emp1.getName().compareTo(emp2.getName());
    }
}
```

```
public class EmployeeSalyComparator implements Comparator<Employee> {
    public int compare(Employee emp1, Employee emp2) {
        return emp1.getSalary() - emp2.getSalary();
    }
}
```

same whitespace, different values

```java
public class EmployeeSorter {

    public void sort(List<DryEmployee> employees, String criteria) {
        Collections.sort(employees, getComparatorFor(criteria));
    }

    private Method getSelectionCriteriaMethod(String methodName) {
        Method m;
        methodName = "get" + methodName.substring(0, 1).toUpperCase() +
                methodName.substring(1);
        try {
            m = DryEmployee.class.getMethod(methodName);
        } catch (NoSuchMethodException e) {
            throw new RuntimeException(e.getMessage());
        }
        return m;
    }

    public Comparator<DryEmployee> getComparatorFor(final String field) {
        return new Comparator<DryEmployee>() {
            public int compare(DryEmployee o1, DryEmployee o2) {
                Object field1, field2;
                Method method = getSelectionCriteriaMethod(field);
                try {
                    field1 = method.invoke(o1);
                    field2 = method.invoke(o2);
                } catch (Exception e) {
                    throw new RuntimeException(e);
                }
                return ((Comparable) field1).compareTo(field2);
            }
        };
    }
}
```

```
@Before public void setup() {
    _sorter = new EmployeeSorter();
    _list = new ArrayList<DryEmployee>();
    _list.add(new DryEmployee("Homer", 20000, 1975));
    _list.add(new DryEmployee("Smithers", 150000, 1980));
    _list.add(new DryEmployee("Lenny", 100000, 1982));
}

@Test public void name_comparisons() {
    _sorter.sort(_list, "name");
    assertThat(_list.get(0).getName(), is("Homer"));
    assertThat(_list.get(1).getName(), is("Lenny"));
    assertThat(_list.get(2).getName(), is("Smithers"));
}

@Test public void salary_comparisons() {
    _sorter.sort(_list, "salary");
    assertThat(_list.get(0).getSalary(), is(20000));
    assertThat(_list.get(1).getSalary(), is(100000));
    assertThat(_list.get(2).getSalary(), is(150000));
}

@Test public void hireYearComparison() {
    _sorter.sort(_list, "hireYear");
    assertThat(_list.get(0).getHireYear(), is(1975));
    assertThat(_list.get(1).getHireYear(), is(1980));
    assertThat(_list.get(2).getHireYear(), is(1982));
}
```

# refactoring tests

```
@Before public void setup() {
    _sorter = new EmployeeSorter();
    _list = new ArrayList<DryEmployee>();
    _list.add(new DryEmployee("Homer", 20000, 1975));
    _list.add(new DryEmployee("Smithers", 150000, 1980));
    _list.add(new DryEmployee("Lenny", 100000, 1982));
}

@Test public void name_comparisons() {
    _sorter.sort(_list, "name");
    assertThat(_list.get(0).getName(), is("Homer"));
    assertThat(_list.get(1).getName(), is("Lenny"));
    assertThat(_list.get(2).getName(), is("Smithers"));
}

@Test public void salary_comparisons() {
    _sorter.sort(_list, "salary");
    assertThat(_list.get(0).getSalary(), is(20000));
    assertThat(_list.get(1).getSalary(), is(100000));
    assertThat(_list.get(2).getSalary(), is(150000));
}

@Test public void hireYearComparison() {
    _sorter.sort(_list, "hireYear");
    assertThat(_list.get(0).getHireYear(), is(1975));
    assertThat(_list.get(1).getHireYear(), is(1980));
    assertThat(_list.get(2).getHireYear(), is(1982));
}
```

# generic tests

```
private HashMap<String, Object> expectations = new HashMap<String, Object>();

@Before public void setup() {
    _sorter = new EmployeeSorter();
    _list = new ArrayList<DryEmployee>();
    _list.add(new DryEmployee("Homer", 20000, 1975));
    _list.add(new DryEmployee("Smithers", 150000, 1980));
    _list.add(new DryEmployee("Lenny", 100000, 1982));
    expectations.put("name", new String[] {"Homer", "Lenny", "Smithers"});
    expectations.put("salary", new Integer[] {20000, 100000, 150000});
    expectations.put("hireYear", new Integer[] {1975, 1980, 1982});
}
```

```
private String[] FIELDS = new String[] {"name", "salary", "hireYear"};

@Test public void all_comparators() {
    for (String field : FIELDS) {
        _sorter.sort(_list, field);
        for (int j = 0; j < expectations.size(); j++) {
            Object[] o = (Object[]) expectations.get(field);
            try {
                Method m = DryEmployee.
                        class.
                        getDeclaredMethod("get" + methodNameFromString(field));
                for (int i = 0; i < o.length; i++)
                    assertThat(m.invoke(_list.get(i)), is(o[i]));
            } catch (Exception e) {
                fail();
            }
        }
    }
}

private static String methodNameFromString(String s) {
    return s.substring(0, 1).toUpperCase() + s.substring(1);
}
```

is this a good idea?

how far is *too* far?

reflection potion


don’t fear powerful
things

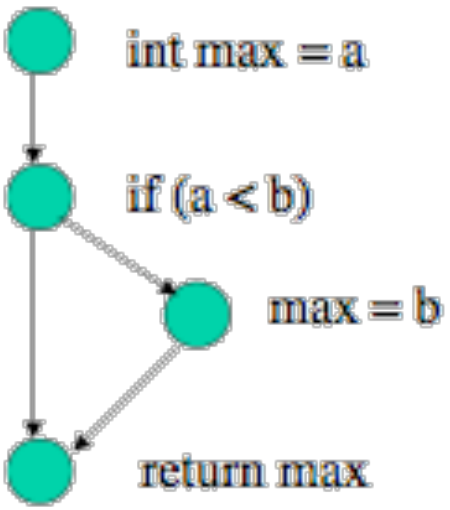
# choosing what to refactor

# cyclomatic complexity

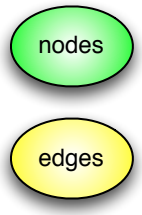
measures complexity of a function

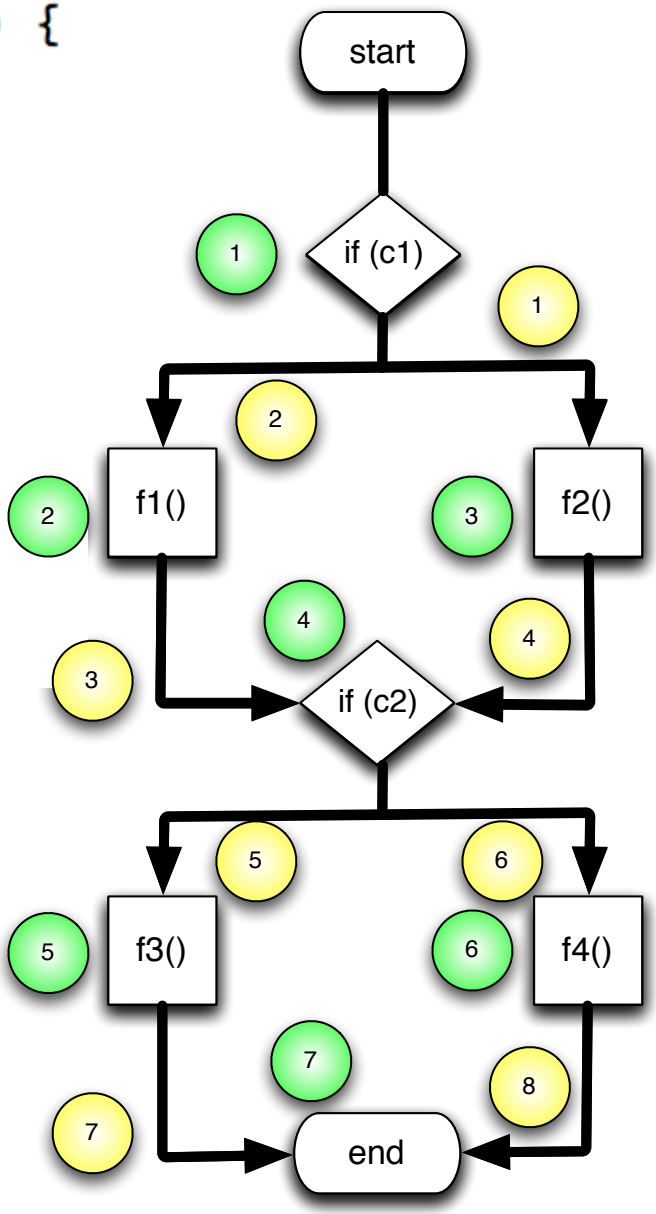
```
V(G)= e - n + 2
V(G) = cyclomatic complexity of G
e= # edges
n= # of nodes
```

```
int max (int a, int b) {
    int max = a;
    if (a < b) {
        max = b;
    }
    return max;
}
```



```
public void doIt() {
    if (c1) {
        f1();
    } else {
        f2();
    }
    if (c2) {
        f3();
    } else {
        f4();
    }
}
```



# afferent coupling

∑ of how many classes use this class

incoming calls

determines what is the “hard, crunchy center” of your code base

measure with CKJM or other metrics tools

# ckjm

open source code metrics tool for java

simple to the extreme

run with command-line fu

several different output formats

includes ant & maven targets

# running ckjm

```
for i in lib/*.jar
do
        jar tf $i |
        sed -n "/\.class$/s,^,$i ,p"
done |
java -jar ~/bin/ckjm-1.8/build/ckjm-1.8.jar | ./ckjm2xml >metrics.xml
```

# ckjm2xml

```
#!/usr/bin/sed -f
1i\
<?xml version="1.0" ?>\
<ckjm>
s/^/<metric><classname>/
s/ /<\/classname><WMC>/
s/ /<\/WMC><DIT>/
s/ /<\/DIT><NOC>/
s/ /<\/NOC><CBO>/
s/ /<\/CBO><RFC>/
s/ /<\/RFC><LCOM>/
s/ /<\/LCOM><Ca>/
s/ /<\/Ca><NPM>/
s/$/<\/NPM><\/metric>/
$a\
</ckjm>
```

<?xml version="1.0" ?>
<ckjm>
<metric><classname>org.apache.struts2.dispatcher.mapper.DefaultActionMapper$2$3</classname><WMC>2</WMC><DIT>1</DIT><NOC>0</NOC><CBO>7</CBO><RFC>10</RFC><LCOM>0</LCOM><Ca>1</Ca><NPM>1</NPM></metric>
<metric><classname>org.apache.struts2.views.velocity.components.BeanDirective</classname><WMC>3</WMC><DIT>0</DIT><NOC>0</NOC><CBO>4</CBO><RFC>5</RFC><LCOM>3</LCOM><Ca>0</Ca><NPM>2</NPM></metric>
<metric><classname>org.apache.struts2.util.MergeIteratorFilter</classname><WMC>6</WMC><DIT>0</DIT><NOC>0</NOC><CBO>2</CBO><RFC>16</RFC><LCOM>0</LCOM><Ca>1</Ca><NPM>6</NPM></metric>
<metric><classname>org.apache.struts2.components.template.VelocityTemplateEngine</classname><WMC>5</WMC><DIT>0</DIT><NOC>0</NOC><CBO>12</CBO><RFC>32</RFC><LCOM>6</LCOM><Ca>0</Ca><NPM>3</NPM></metric>
<metric><classname>org.apache.struts2.views.jsp.ui.AnchorTag</classname><WMC>6</WMC><DIT>0</DIT><NOC>0</NOC><CBO>4</CBO><RFC>10</RFC><LCOM>13</LCOM><Ca>0</Ca><NPM>5</NPM></metric>
<metric><classname>org.apache.struts2.views.freemarker.StrutsBeanWrapper$FriendlyMapModel$1</classname><WMC>2</WMC><DIT>1</DIT><NOC>0</NOC><CBO>5</CBO><RFC>4</RFC><LCOM>1</LCOM><Ca>1</Ca><NPM>1</NPM></metric>
<metric><classname>org.apache.struts2.portlet.PortletRequestMap</classname><WMC>8</WMC><DIT>2</DIT><NOC>0</NOC><CBO>3</CBO><RFC>31</RFC><LCOM>0</LCOM><Ca>2</Ca><NPM>6</NPM></metric>
<metric><classname>org.apache.struts2.views.jsp.ui.TextFieldTag</classname><WMC>7</WMC><DIT>0</DIT><NOC>2</NOC><CBO>4</CBO><RFC>13</RFC><LCOM>11</LCOM><Ca>2</Ca><NPM>6</NPM></metric>
<metric><classname>org.apache.struts2.portlet.servlet.PortletServletConfig</classname><WMC>6</WMC><DIT>1</DIT><NOC>0</NOC><CBO>1</CBO><RFC>12</RFC><LCOM>0</LCOM><Ca>0</Ca><NPM>6</NPM></metric>
<metric><classname>org.apache.struts2.views.freemarker.tags.FileModel</classname><WMC>2</WMC><DIT>0</DIT><NOC>0</NOC><CBO>4</CBO><RFC>4</RFC><LCOM>1</LCOM><Ca>1</Ca><NPM>1</NPM></metric>
<metric><classname>org.apache.struts2.portlet.context.PortletActionContext</classname><WMC>17</WMC><DIT>1</DIT><NOC>0</NOC><CBO>3</CBO><RFC>22</RFC><LCOM>136</LCOM><Ca>7</Ca><NPM>16</NPM></metric>
<metric><classname>org.apache.struts2.dispatcher.VelocityResult</classname><WMC>10</WMC><DIT>0</DIT><NOC>0</NOC><CBO>13</CBO><RFC>40</RFC><LCOM>39</LCOM><Ca>0</Ca><NPM>5</NPM></metric>
<metric><classname>org.apache.struts2.components.TabbedPanel</classname><WMC>12</WMC><DIT>0</DIT><NOC>0</NOC><CBO>2</CBO><RFC>20</RFC><LCOM>54</LCOM><Ca>3</Ca><NPM>9</NPM></metric>
<metric><classname>org.apache.struts2.interceptor.ParameterAware</classname><WMC>1</WMC><DIT>1</DIT><NOC>0</NOC><CBO>0</CBO><RFC>1</RFC><LCOM>0</LCOM><Ca>1</Ca><NPM>1</NPM></metric>
<metric><classname>org.apache.struts2.views.velocity.components.AbstractDirective</classname><WMC>8</WMC><DIT>0</DIT><NOC>43</NOC><CBO>12</CBO><RFC>34</RFC><LCOM>28</LCOM><Ca>43</Ca><NPM>5</NPM></metric>
<metric><classname>org.apache.struts2.interceptor.CookieInterceptor</classname><WMC>7</WMC><DIT>0</DIT><NOC>0</NOC><CBO>9</CBO><RFC>30</RFC><LCOM>0</LCOM><Ca>0</Ca><NPM>4</NPM></metric>
<metric><classname>org.apache.struts2.interceptor.ProfilingActivationInterceptor</classname><WMC>5</WMC><DIT>0</DIT><NOC>0</NOC><CBO>4</CBO><RFC>14</RFC><LCOM>0</LCOM><Ca>0</Ca><NPM>5</NPM></metric>
<metric><classname>org.apache.struts2.components.table.WebTable$WebTableRowIterator</classname><WMC>5</WMC><DIT>1</DIT><NOC>0</NOC><CBO>1</CBO><RFC>11</RFC><LCOM>4</LCOM><Ca>1</Ca><NPM>3</NPM></metric>
<metric><classname>org.apache.struts2.interceptor.CookiesAware</classname><WMC>1</WMC><DIT>1</DIT><NOC>0</NOC><CBO>0</CBO><RFC>1</RFC><LCOM>0</LCOM><Ca>1</Ca><NPM>1</NPM></metric>
<metric><classname>org.apache.struts2.views.xslt.SimpleNodeList</classname><WMC>4</WMC><DIT>1</DIT><NOC>0</NOC><CBO>2</CBO><RFC>19</RFC><LCOM>0</LCOM><Ca>1</Ca><NPM>4</NPM></metric>
<metric><classname>org.apache.struts2.portlet.interceptor.PortletRequestAware</classname><WMC>1</WMC><DIT>1</DIT><NOC>0</NOC><CBO>0</CBO><RFC>1</RFC><LCOM>0</LCOM><Ca>1</Ca><NPM>1</NPM></metric>

...version="1.0"

| | .classname | WMC | DIT | NOC | CBO | RFC | LCOM | Ca | NPM |
|---|---|---|---|---|---|---|---|---|---|
| 1 | org.apache.struts2.dispatcher.mapper.DefaultActionMapper$2$3 | 2 | 1 | 0 | 7 | 10 | 0 | 1 | 1 |
| 2 | org.apache.struts2.views.velocity.components.BeanDirective | 3 | 0 | 0 | 4 | 5 | 3 | 0 | 2 |
| 3 | org.apache.struts2.util.MergeIteratorFilter | 6 | 0 | 0 | 2 | 16 | 0 | 1 | 6 |
| 4 | org.apache.struts2.components.template.VelocityTemplateEngine | 5 | 0 | 0 | 12 | 32 | 6 | 0 | 3 |
| 5 | org.apache.struts2.views.jsp.ui.AnchorTag | 6 | 0 | 0 | 4 | 10 | 13 | 0 | 5 |
| 6 | org.apache.struts2.views.freemarker.StrutsBeanWrapper$FriendlyMapModel$1 | 2 | 1 | 0 | 5 | 4 | 1 | 1 | 1 |
| 7 | org.apache.struts2.portlet.PortletRequestMap | 8 | 2 | 0 | 3 | 31 | 0 | 2 | 6 |
| 8 | org.apache.struts2.views.jsp.ui.TextFieldTag | 7 | 0 | 2 | 4 | 13 | 11 | 2 | 6 |
| 9 | org.apache.struts2.portlet.servlet.PortletServletConfig | 6 | 1 | 0 | 1 | 12 | 0 | 0 | 6 |
| 10 | org.apache.struts2.views.freemarker.tags.FileModel | 2 | 0 | 0 | 4 | 4 | 1 | 1 | 1 |
| 11 | org.apache.struts2.portlet.context.PortletActionContext | 17 | 1 | 0 | 3 | 22 | 136 | 7 | 16 |
| 12 | org.apache.struts2.dispatcher.VelocityResult | 10 | 0 | 0 | 13 | 40 | 39 | 0 | 5 |
| 13 | org.apache.struts2.components.TabbedPanel | 12 | 0 | 0 | 2 | 20 | 54 | 3 | 9 |
| 14 | org.apache.struts2.interceptor.ParameterAware | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 15 | org.apache.struts2.views.velocity.components.AbstractDirective | 8 | 0 | 43 | 12 | 34 | 28 | 43 | 5 |
| 16 | org.apache.struts2.interceptor.CookieInterceptor | 7 | 0 | 0 | 9 | 30 | 0 | 0 | 4 |
| 17 | org.apache.struts2.interceptor.ProfilingActivationInterceptor | 5 | 0 | 0 | 4 | 14 | 0 | 0 | 5 |
| 18 | org.apache.struts2.components.table.WebTable$WebTableRowIterator | 5 | 1 | 0 | 1 | 11 | 4 | 1 | 3 |

# struts 2.x

| classname | WMC | Ca |
|---|---|---|
| org.apache.struts2.components.Component | 28 | 177 |
| org.apache.struts2.views.freemarker.tags.TagModel | 7 | 47 |
| org.apache.struts2.views.velocity.components.AbstractDirective | 8 | 43 |
| org.apache.struts2.StrutsException | 7 | 23 |
| org.apache.struts2.components.UIBean | 53 | 22 |
| org.apache.struts2.dispatcher.mapper.ActionMapping | 13 | 20 |
| org.apache.struts2.views.jsp.ComponentTagSupport | 6 | 19 |
| org.apache.struts2.dispatcher.Dispatcher | 37 | 19 |
| org.apache.struts2.views.jsp.ui.AbstractUITag | 34 | 18 |
| org.apache.struts2.views.xslt.AdapterFactory | 9 | 16 |
| org.apache.struts2.views.xslt.AdapterNode | 10 | 15 |
| org.apache.struts2.ServletActionContext | 11 | 15 |
| org.apache.struts2.components.table.WebTable | 33 | 12 |
| org.apache.struts2.dispatcher.mapper.ActionMapper | 2 | 11 |
| org.apache.struts2.components.template.TemplateEngine | 2 | 10 |
| org.apache.struts2.components.template.Template | 7 | 10 |
| org.apache.struts2.dispatcher.StrutsResultSupport | 13 | 10 |
| org.apache.struts2.components.Form | 24 | 10 |
| org.apache.struts2.components.ListUIBean | 8 | 9 |
| org.apache.struts2.util.MakeIterator | 3 | 8 |
| org.apache.struts2.StrutsStatics | 0 | 7 |

# UIBean evaluateParams()

```
public void evaluateParams() {
    addParameter("templateDir", getTemplateDir());
    addParameter("theme", getTheme());

    String name = null;

    if (this.key != null) {

       if(this.name == null) {
            this.name = key;
        }

        if(this.label == null) {
            this.label = "%{getText('"+ key +"')}";
        }

    }

    if (this.name != null) {
        name = findString(this.name);
        addParameter("name", name);
    }

    if (label != null) {
        addParameter("label", findString(label));
    }
```

# identifying idiomatic patterns

lots of accidental complexity around **`evaluateParameters()`**

what is the purpose of this code?

what would make it cleaner?

move to a factory or strategy design pattern?

idiomatic patterns implemented via formal design patterns

# intersection of idiomatic & design patterns

# unit of work

```
public void addOrderFrom(ShoppingCart cart, String userName,
                         Order order) throws Exception {
    setupDataInfrastructure();
    try {
        add(order, userKeyBasedOn(userName));
        addLineItemsFrom(cart, order.getOrderKey());
        completeTransaction();
    } catch (Exception condition) {
        rollbackTransaction();
        throw condition;
    } finally {
        cleanUp();
    }
}
```

# command design pattern

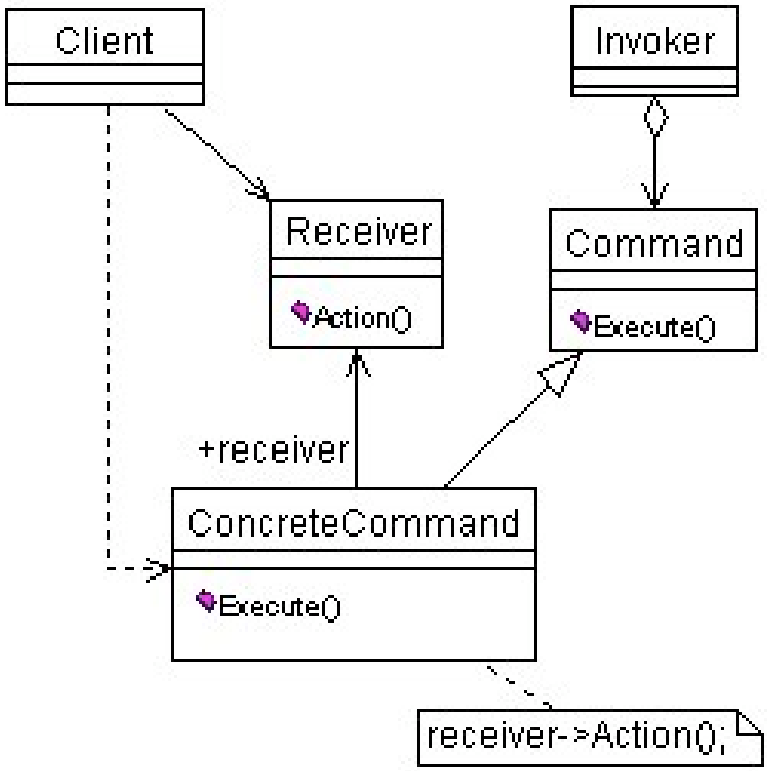
```
public void wrapInTransaction(Command c) throws Exception {
    setupDataInfrastructure();
    try {
        c.execute();
        completeTransaction();
    } catch (Exception condition) {
        rollbackTransaction();
        throw condition;
    } finally {
        cleanUp();
    }
}

public void addOrderFrom(final ShoppingCart cart, final String userName,
                         final Order order) throws Exception {
    wrapInTransaction(new Command() {
        public void execute() {
            add(order, userKeyBasedOn(userName));
            addLineItemsFrom(cart, order.getOrderKey());
        }
    });
}
```

# *unit of work*

common idiomatic pattern

implemented via the command design pattern

use aspects



# world's worst framework

every single project depends on it

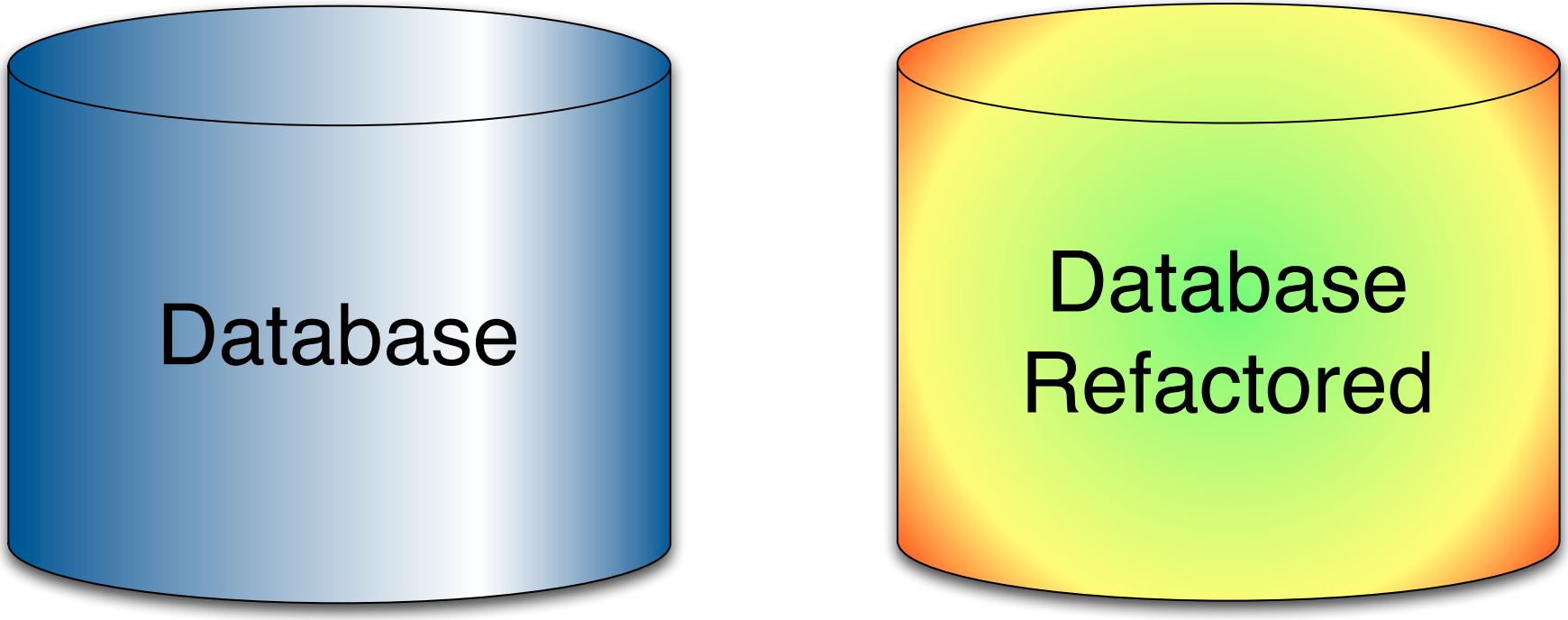
strange configuration

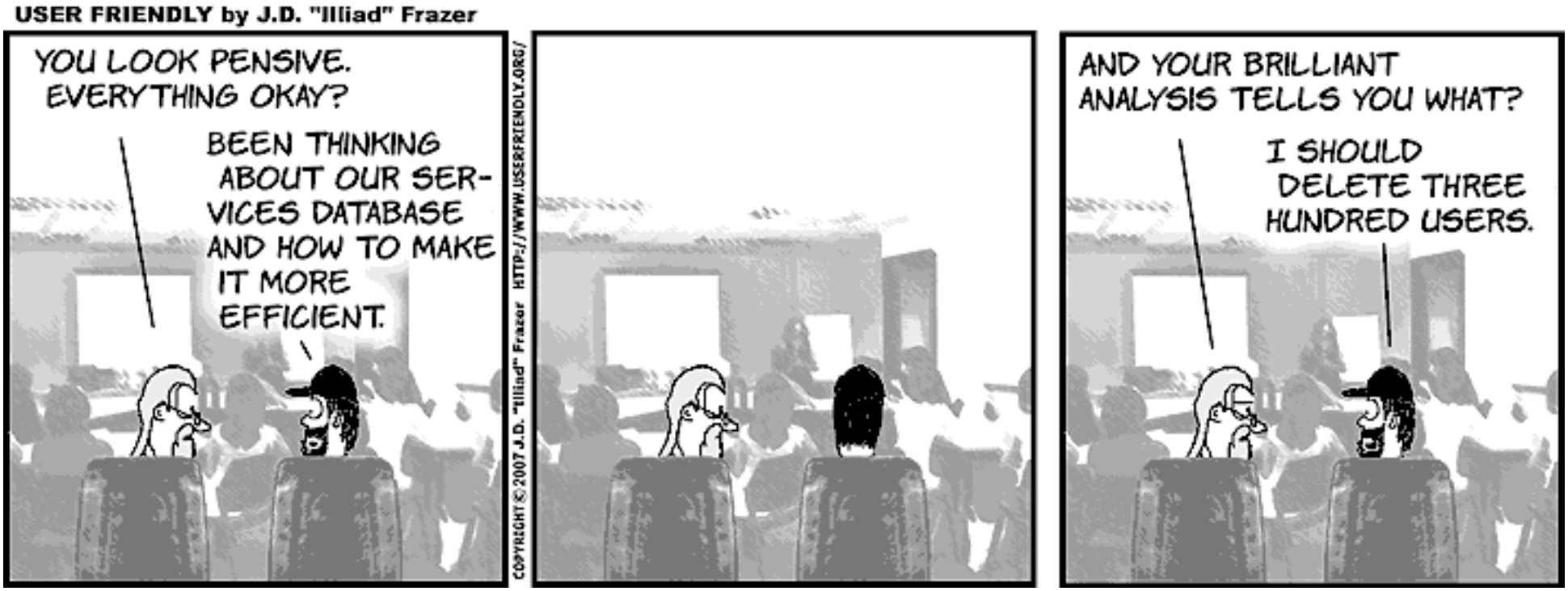
hard to test

no refactoring support

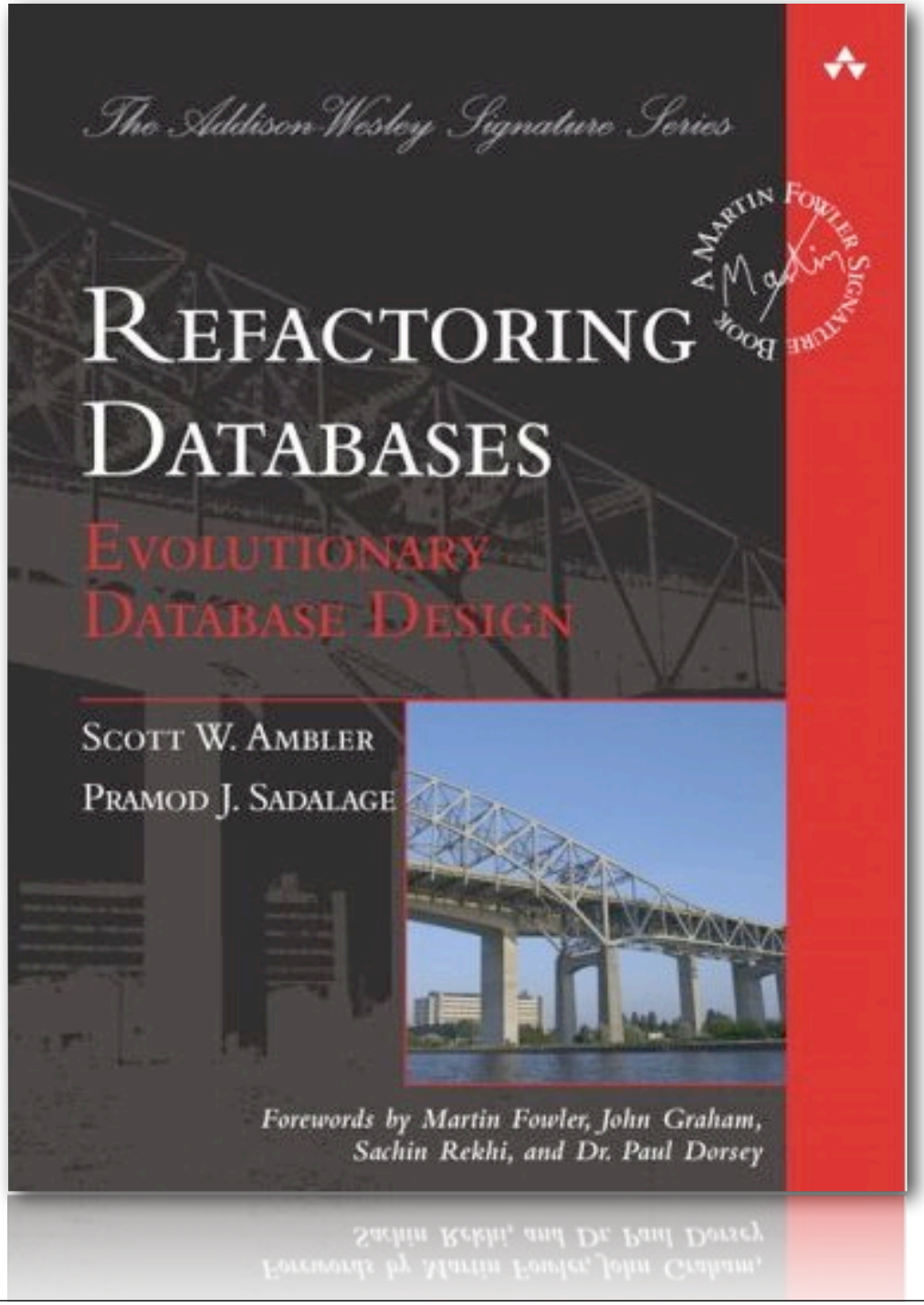
outlives every project

# database refactoring






http://ars.userfriendly.org/cartoons/?id=20090304


The Addison Wesley Signature Series
A Martin Fowler Signature Book
Refactoring Databases
Evolutionary Database Design
Scott W. Ambler
Pramod J. Sadalage
Forewords by Martin Fowler, John Graham,
Sachin Rekhi, and Dr. Paul Dorsey

# process

1. verify that a database refactoring is appropriate
2. choose the most appropriate database refactoring
3. deprecate the original database schema
4. test before, during, and after
5. modify the database schema

# process

6. migrate the source data

7. modify external access program(s)

8. run regression tests

9. version control your work

10. announce the refactoring

# dbDeploy

migrations for the rest of the world

open-source tool to help you manage database changes

version database schema changes along with code

builds delta scripts for you

# using dbDeploy

someone (dba) creates a sql baseline

developers create delta scripts for each change

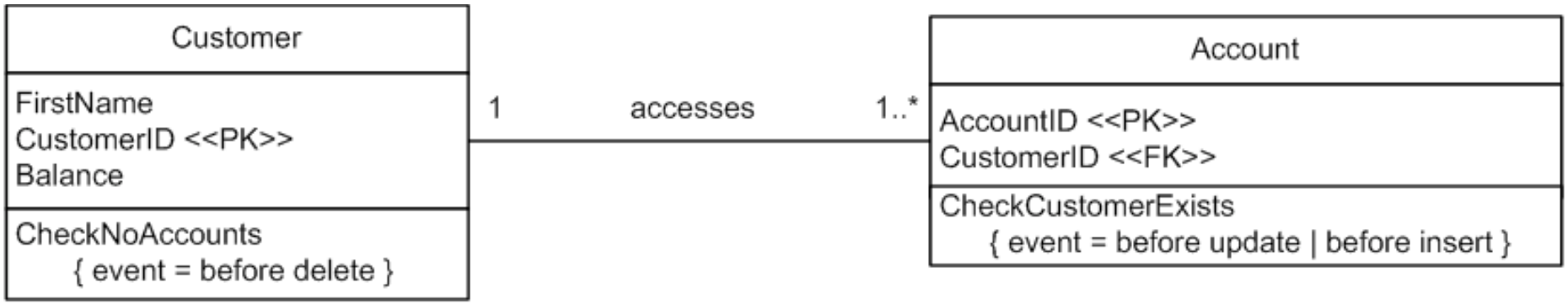
dbDeploy compares delta with patch table

generates change scripts

invoked from ant's sql task

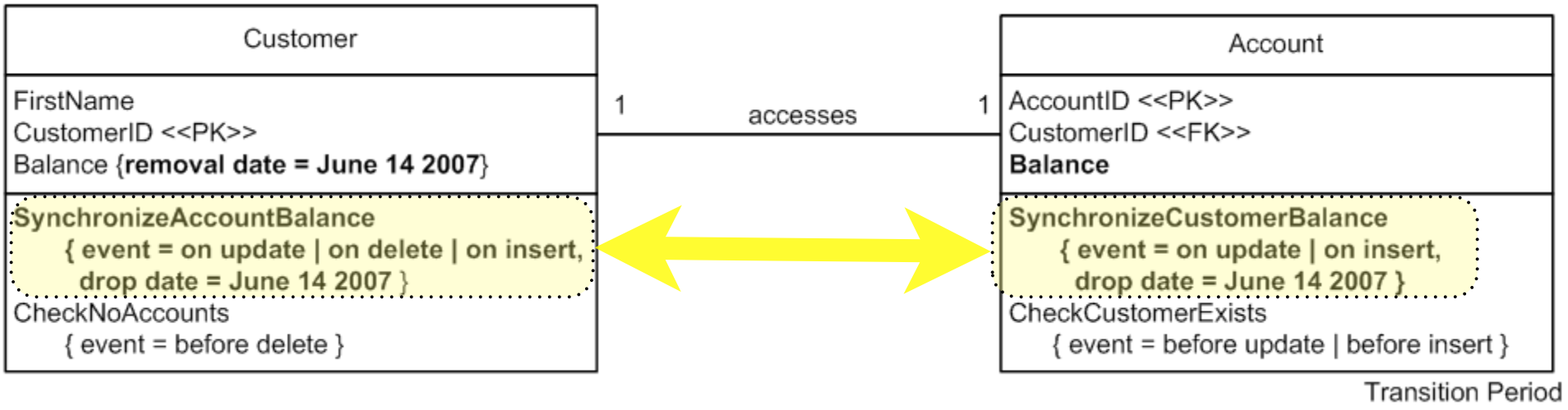
# example

move column refactoring

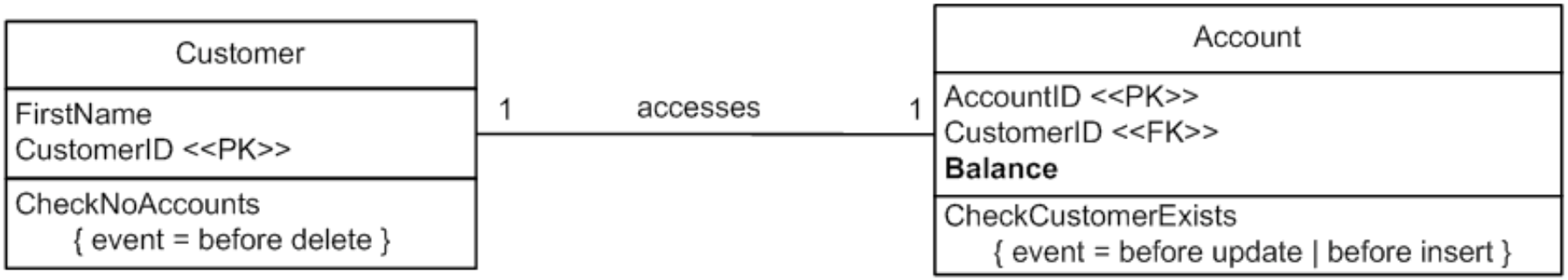



Original Schema

# transition period



# resulting schema




Resulting Schema

# strategies

smaller changes are easier to apply

uniquely identify individual refactorings

implement a large change by many smaller ones

prefer triggers over views or batch synchronization

choose a sufficient deprecation period

# strategies

encapsulate data access

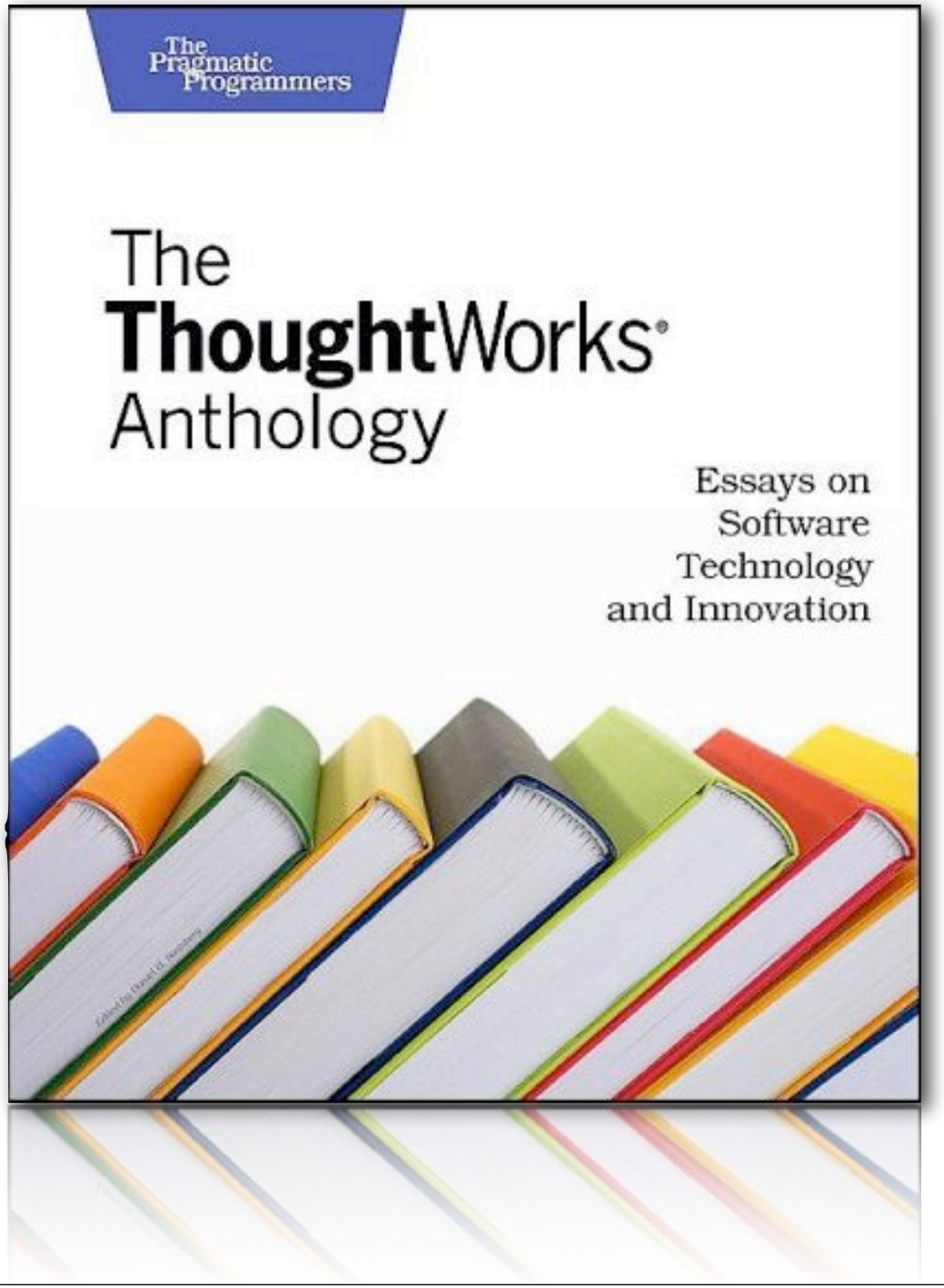
be able to easily set up a database environment

don’t duplicate SQL

put database assets under change control

beware of politics


build files


The Pragmatic Programmers
The ThoughtWorks® Anthology
Essays on Software Technology and Innovation

*When we...construct and maintain build files, all those code perfectionist ideals seem to disappear.*
► Paul Glover

Chapter 11

# Refactoring Ant Build Files

**by Julian Simpson, Build Architect**

## 11.1 Introduction

A bad software build system seems to actively resist change. The most innocent change can stop your colleagues from being able to work. Such a reputation won't encourage people to try to improve a build if it eventually gets the job done. This essay will show you how to help ease some of that pain and allow change to take place by describing some short refactorings to apply to your Ant build file. Each refactoring is expressed as a "before" and "after" code example (separated by an arrow to show the direction of transformation) with an explanation. You will be left with some concrete tools that you can apply to your Ant build to make it smaller, more legible, and easier to modify.

### What Is Refactoring? And What Is Ant?

*Refactoring* is the art of making small changes to a system to improve the readability, clarity, or ease of maintenance of a codebase. Refactoring doesn't change functionality; it's used to make the internals of a codebase easier to work on.

Ant is the build tool of choice for many Java-based software projects. It was written at a time when XML seemed to be the answer to any problem in software development. Hence, a project's dependencies are described in an XML file, typically called build.xml. This is the Java equivalent of a Makefile. Ant has been very successful as an open source project, but as projects grow, flex, and become more complicated, it generally becomes hard to maintain.

# *extract macrodef*

Take small blocks of ant code, and pull out into macrodef with appropriate name.

```
<target name="build_and_war_foo.war" >
    <javac srcdir="src/foo" destdir="classes/foo" />
    <copy todir="${classes.dir}" >
        <filterset>
            <filter token="ENV" value="${environment}" />
        </filterset>
        <fileset dir="config" />
    </copy>
    <war destfile="foo.war" >
        <fileset dir="${classes.dir}" />
    </war>
    <move todir="archives" file="foo.war" />
</target>
```

# `macrodef`

container task

wraps sequential or parallel tasks

invokable from anywhere in the build file

with attributes

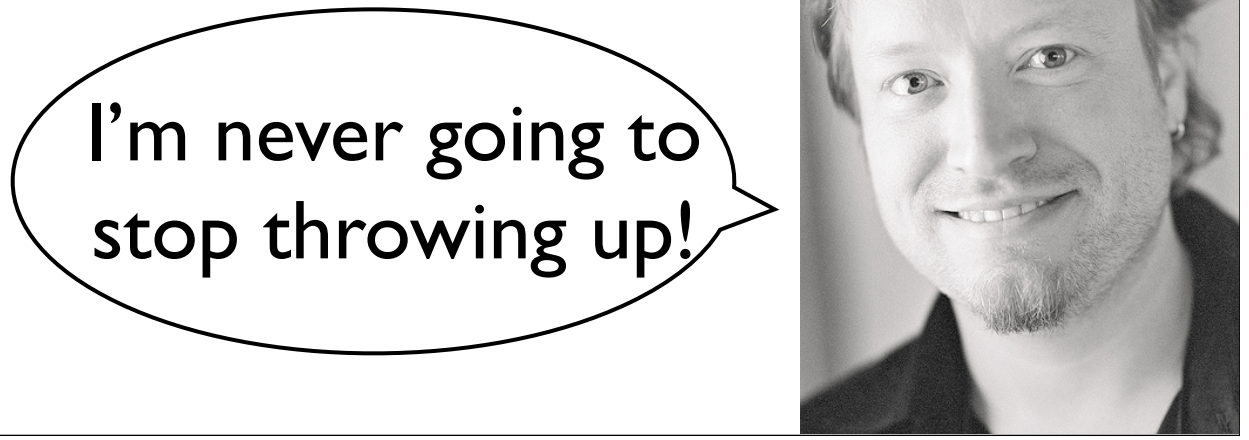
attributes can have default values

```
<macrodef name="build_code" >
    <attribute name="component" />
    <sequential>
        <javac srcdir="src/@{component}" destdir="classes/@{component}" />
        <copy todir="${classes.dir}" >
            <filterset>
                <filter token="ENV" value="${environment}" />
            </filterset>
            <fileset dir="config" />
        </copy>
    </sequential>
</macrodef>
<macrodef name="make_war" >
    <attribute name="component" />
    <sequential>
        <war destfile="@{component}.war" >
            <fileset dir="${classes.dir}" />
        </war>
        <move todir="archives" file="@{component}.war" />
    </sequential>
</macrodef>
<target name="foo.war" >
    <build_code component="foo" />
    <make_war component="foo" />
</target>
```

# *introduce declaration*

use ant's built-in declarative logic to replace if conditions, which can be hard to debug.

```
<target name="deploy" >
    <if>
        <equals arg1="${j2ee.server}" arg2="was" />
        <then>
            <antcall target="was_deploy" />
        </then>
        <else>
            <antcall target="weblogic_deploy" />
        </else>
    </if>
</target>
```



```
<property name="j2ee.server" value="was" />
<import file="${j2ee.server}.build.xml" />
<!-- there is now a an appropriate target named
deploy depending on the version of the app server -->
```

# *move target to wrapper build file*

pull continuous integration targets out of the developer build file; provide some indirection.

# separating concerns

mixed concerns obstruct refactoring

logically separate build and continuous integration logic

before:

```
<target name="build" >
    <!-- developer build-->
</target>
<target name="functest" >
    <!-- functional tests-->
</target>
<target name="cruise" depends="update,build,tag" />
<target name="functional_cruise" depends="update,build,functest,tag" />
```

```
<target name="build" >
    <!-- developer build-->
</target>
<target name="functest" >
    <!-- functional tests-->
</target>
```

*developer concerns*

```
<project name="cruise" default="tag" >
    <target name="tag" depends="build" >
        <!-- code to tag the files you have checked out -->
    </target>
    <target name="build" depends="update" >
        <ant buildfile="build.xml" />
    </target>
    <target name="update" >
        <!-- code to update from your scm system-->
    </target>
</project>
<!-- END ccbuild -->
<project default="update" basedir="."
    xmlns:my="antlib:com.thoughtworks.monkeybook" >
    <target name="update" depends="build" >
        <my:svn_up/>
    </target>
</project>
<!-- END antlibccbuild -->
```

*build concerns*

# *enforce internal target*

make the first character of internal target names a hyphen so they cannot be called from the command line

# Ant encapsulation

no scoping for ant tasks...

...so make you own via conventions

```
<target name="-init" >
    <mkdir dir="build" />
</target>
```

# lots of bad habits

naming

treating build files like bash

scoping

methods

*reuse*


*everything* in your
infrastructure matters

# ? ' s

please fill out the session evaluations
samples at `github.com/nealford`



**NEAL FORD** software architect / meme wrangler

**ThoughtWorks**

**nford@thoughtworks.com**
**3003 Summit Boulevard, Atlanta, GA 30319**
www.nealford.com
www.thoughtworks.com
memeagora.blogspot.com