Code Smells & Refactoring

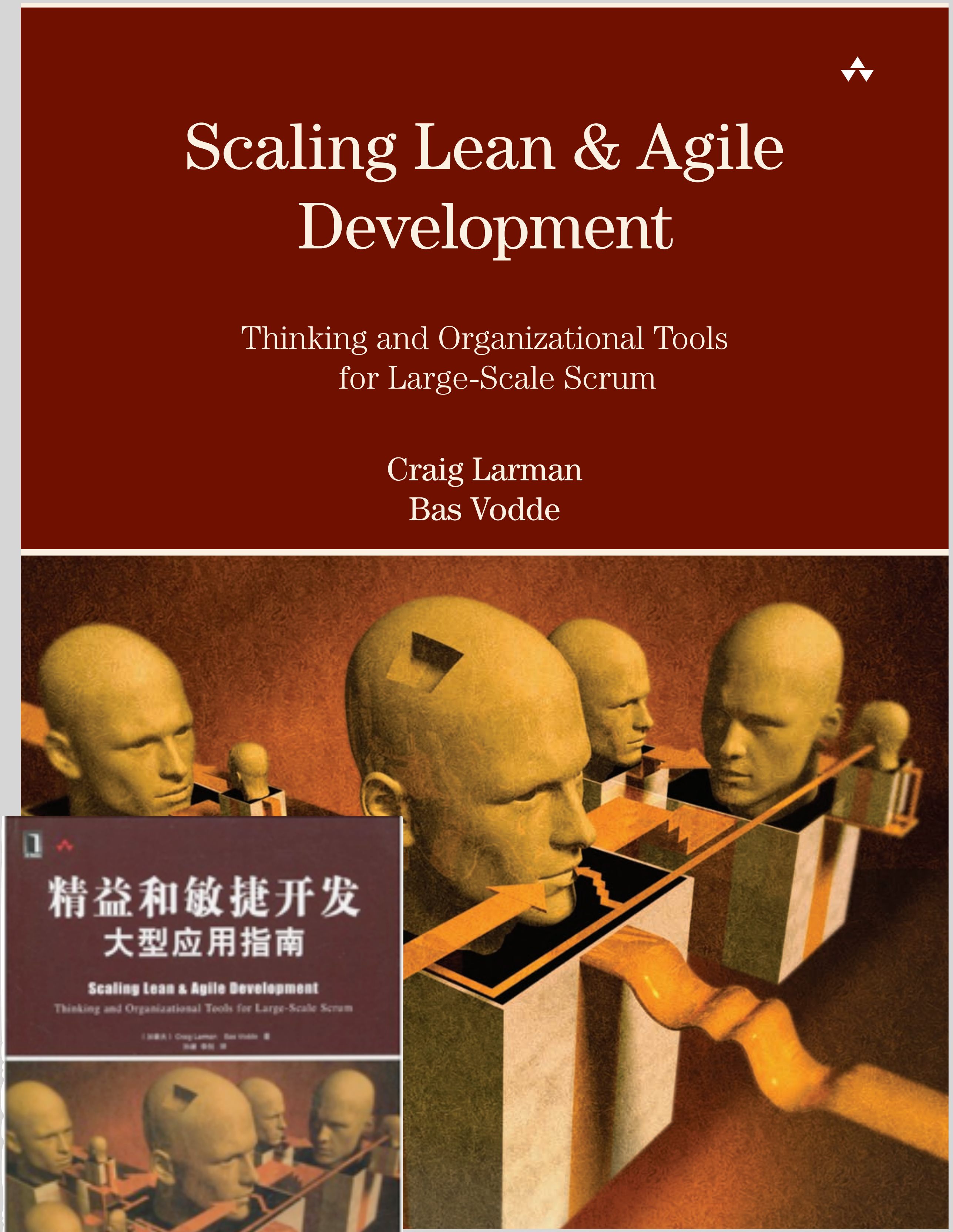
Who am I?

- Name: Bas Vodde
- Originally from Holland
- Lives in Singapore
  - Lived in China and Finland
- Works for Odd-e
- Agile coach, SW developer
- Led Agile transformation program in large company
- Experience with large embedded products

Who am I?

- Name: Bas Vodde
- O y fr lland
- Li Sin e
  - n and
- Wo
- Agi h leveloper
- Led ra atio prog la mpa
- Expe wit large embedded products

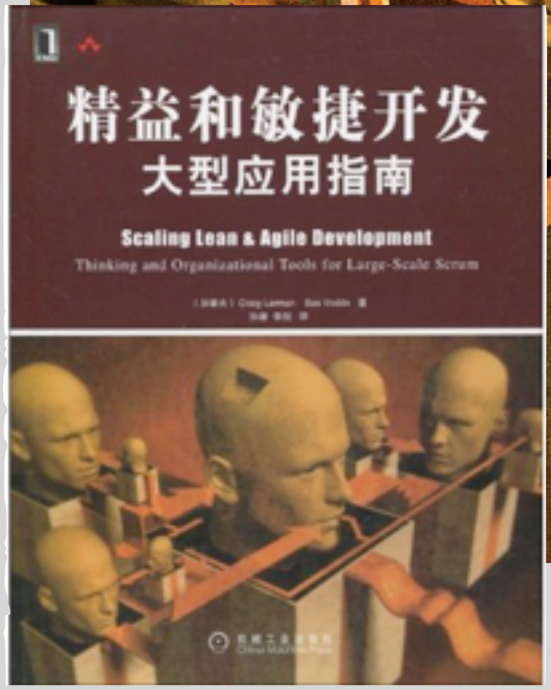
K.O.



Scaling Lean & Agile Development

Thinking and Organizational Tools for Large-Scale Scrum

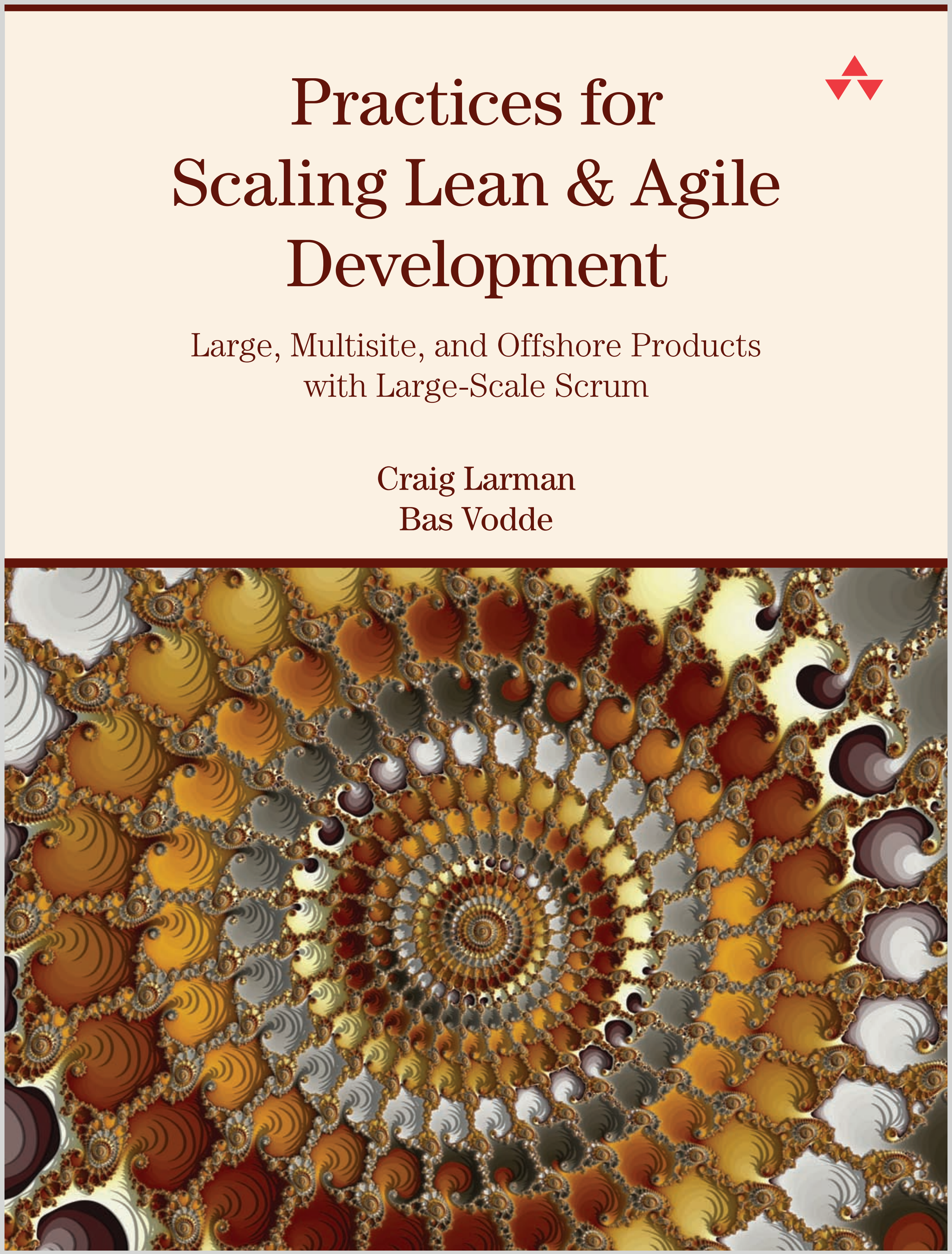
Craig Larman
Bas Vodde



精益和敏捷开发
大型应用指南

Scaling Lean & Agile Development
Thinking and Organizational Tools for Large-Scale Scrum

[illegible] Craig Larman Bas Vodde 著
[illegible] 译



Practices for Scaling Lean & Agile Development

Large, Multisite, and Offshore Products with Large-Scale Scrum

Craig Larman
Bas Vodde

Who am I?

- Name: Steven Mak
- Agile Coach at Odd-e
- Lives in Hong Kong
- Agile, TDD Coaching
- I love coding - Java, C/C++, PHP, Perl, and some weird ones
- I speak English, Cantonese, and Mandarin

Who am I?

- Name: Stanly Lau
- Originate & lives in Singapore
- Works for Odd-e
- Agile coach, SW developer
- Insurance, Mobile Safety & Education
- Java, .Net

Quick Intro

Refactoring

Structured code transformation to prepare the code for change

Key points:

- Doesn’t adjust functionality
- Small and disciplined
- Well defined
- Keeps code healthy

Code Smells

It smells!

It stinks!

A sign there is probably something wrong with your code

Example: Duplicate code

```
public DataSet navigateFileUp()
{
    int fileCount = this.db.getFileCount();
    int x = 0;
    if(fileCount <= 0) return null;
    x = (this.db.getCurrentFileIndex() + 1) % fileCount;
    if(this.db.getFileSize(x) == 0)//file still empty
        return this.db.setNewCurrentSet(x);
    return this.db.setCurrentSet(x, this.db.getCurrentCorrectAnsweredIndex(), 0);
}

public DataSet navigateFileDown()
{
    int fileCount = this.db.getFileCount();
    int x = 0;
    if(fileCount <= 0) return null;
    x = (this.db.getCurrentFileIndex() - 1 + fileCount) % fileCount;
    if(this.db.getFileSize(x) == 0)//file still empty
        return this.db.setNewCurrentSet(x);
    return this.db.setCurrentSet(x, this.db.getCurrentCorrectAnsweredIndex(), 0);
}
```

Example: Duplicate code

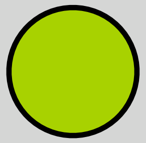
```
public DataSet navigateFileUp()
{
    int fileCount = this.db.getFileCount();
    int x = 0;
    if(fileCount <= 0) return null;
    x = (this.db.getCurrentFileIndex() + 1) % fileCount;
    if(this.db.getFileSize(x) == 0)//file still empty
        return this.db.setNewCurrentSet(x);
    return this.db.setCurrentSet(x, this.db.getCurrentCorrectAnsweredIndex(), 0);
}

public DataSet navigateFileDown()
{
    int fileCount = this.db.getFileCount();
    int x = 0;
    if(fileCount <= 0) return null;
    x = (this.db.getCurrentFileIndex() - 1 + fileCount) % fileCount;
    if(this.db.getFileSize(x) == 0)//file still empty
        return this.db.setNewCurrentSet(x);
    return this.db.setCurrentSet(x, this.db.getCurrentCorrectAnsweredIndex(), 0);
}
```

Only difference!

Refactoring visualized

Without refactoring:

Original program:



Making changes:

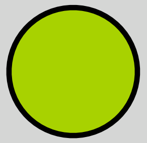
More changes:

Odd-e

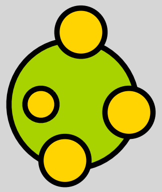
Refactoring visualized

Without refactoring:

Original program:



Making changes:

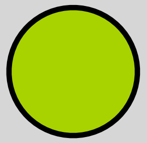


More changes:

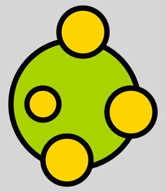
Refactoring visualized

Without refactoring:

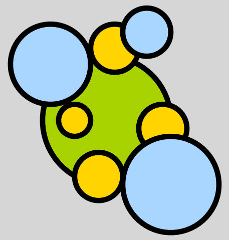
Original program:



Making changes:



More changes:



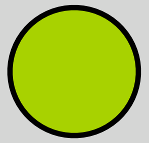
Cost of change increases rapidly!

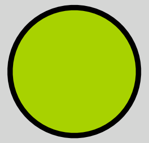
Refactoring visualized

Without refactoring:

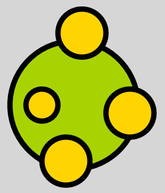
With refactoring:

Original program:

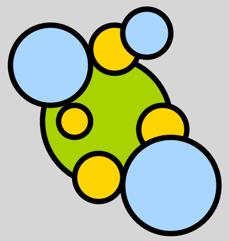




Making changes:



More changes:



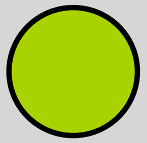
Cost of change increases rapidly!

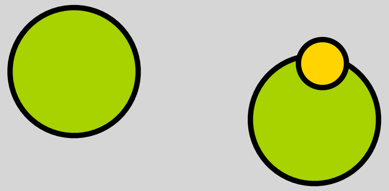
Refactoring visualized

Without refactoring:

With refactoring:

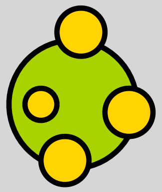
Original program:



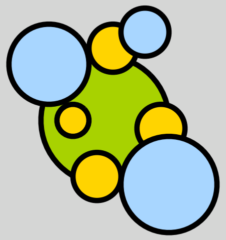


Small change

Making changes:



More changes:



Cost of change increases rapidly!

Refactoring visualized

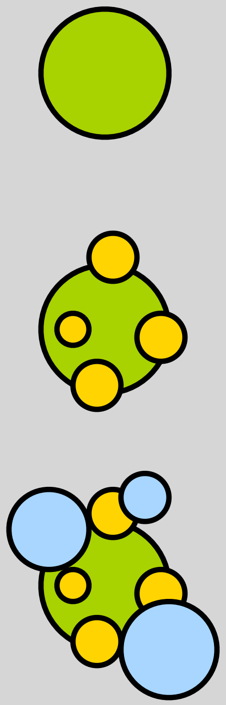
Without refactoring:

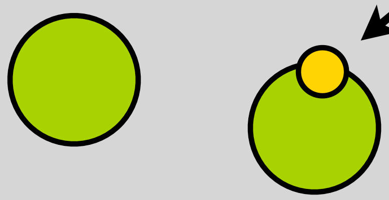
With refactoring:

Original program:

Making changes:

More changes:

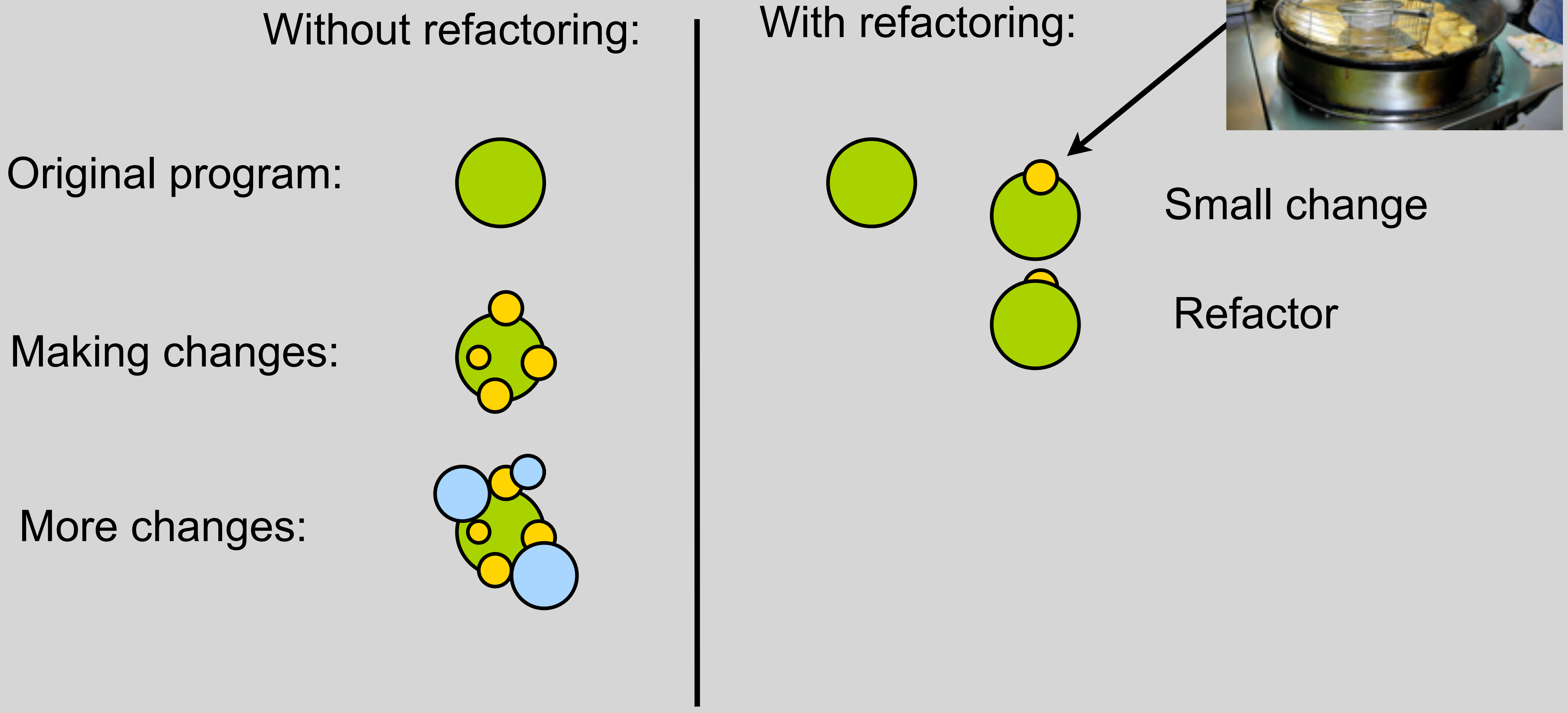




Small change

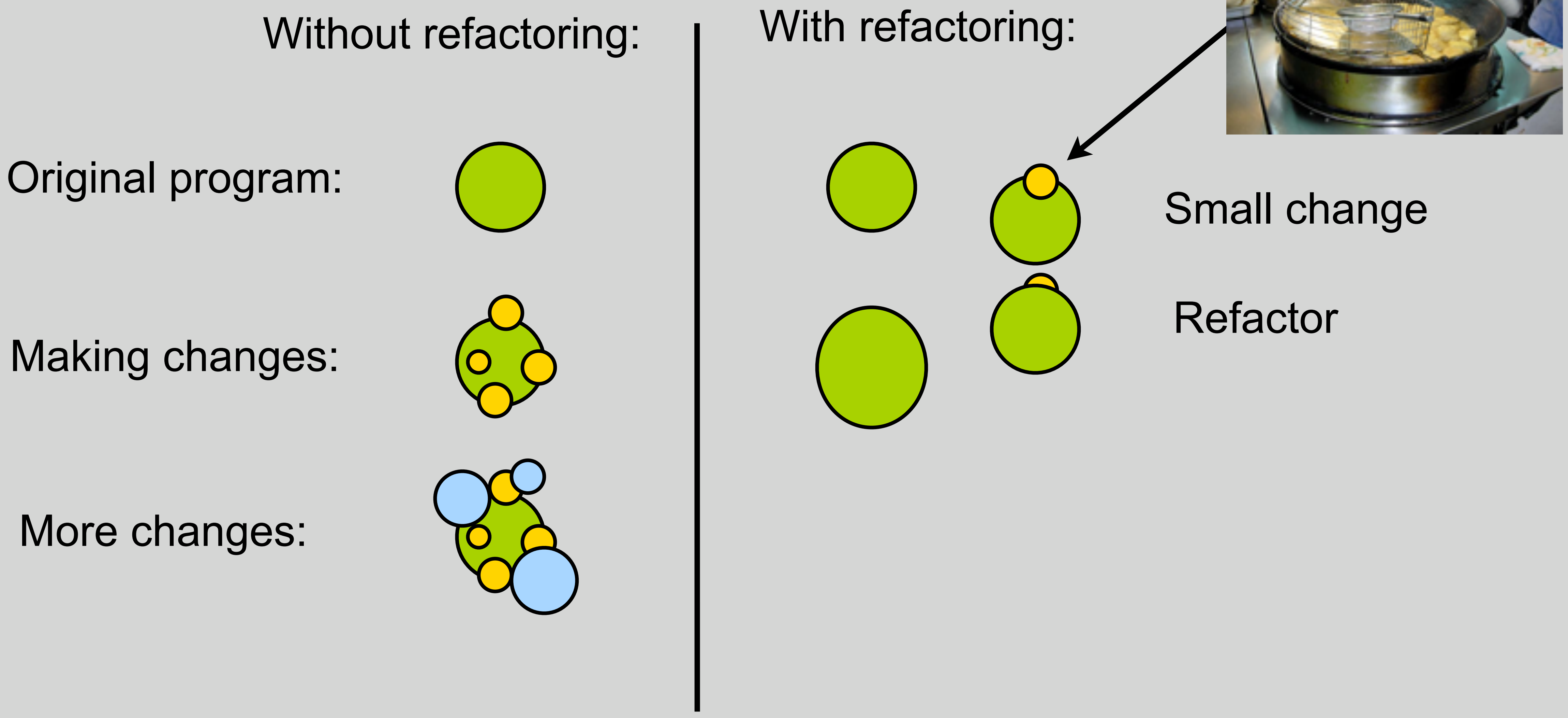
Cost of change increases rapidly!

Refactoring visualized



Cost of change increases rapidly!

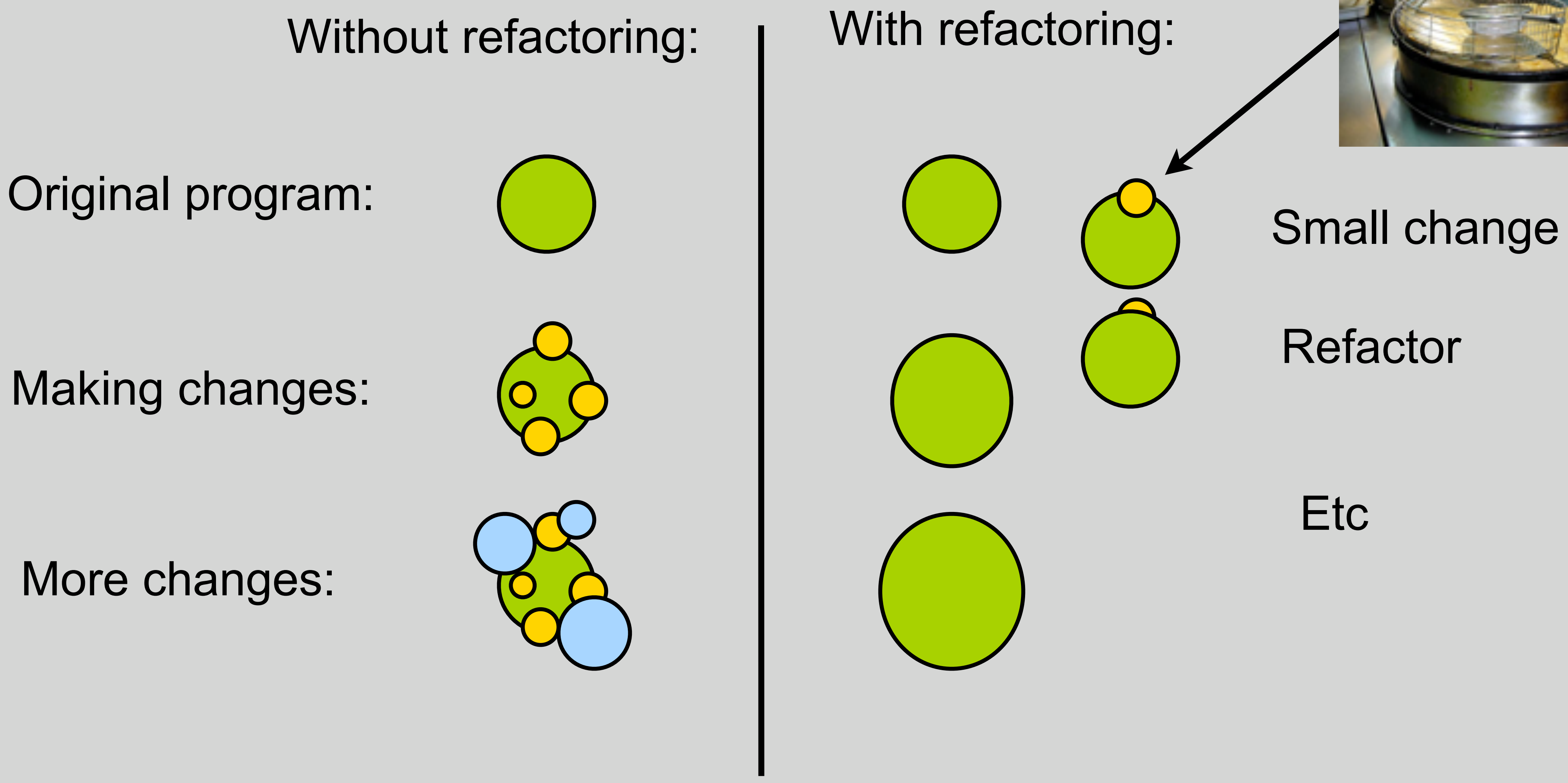
Refactoring visualized



Cost of change increases rapidly!

Odd-e

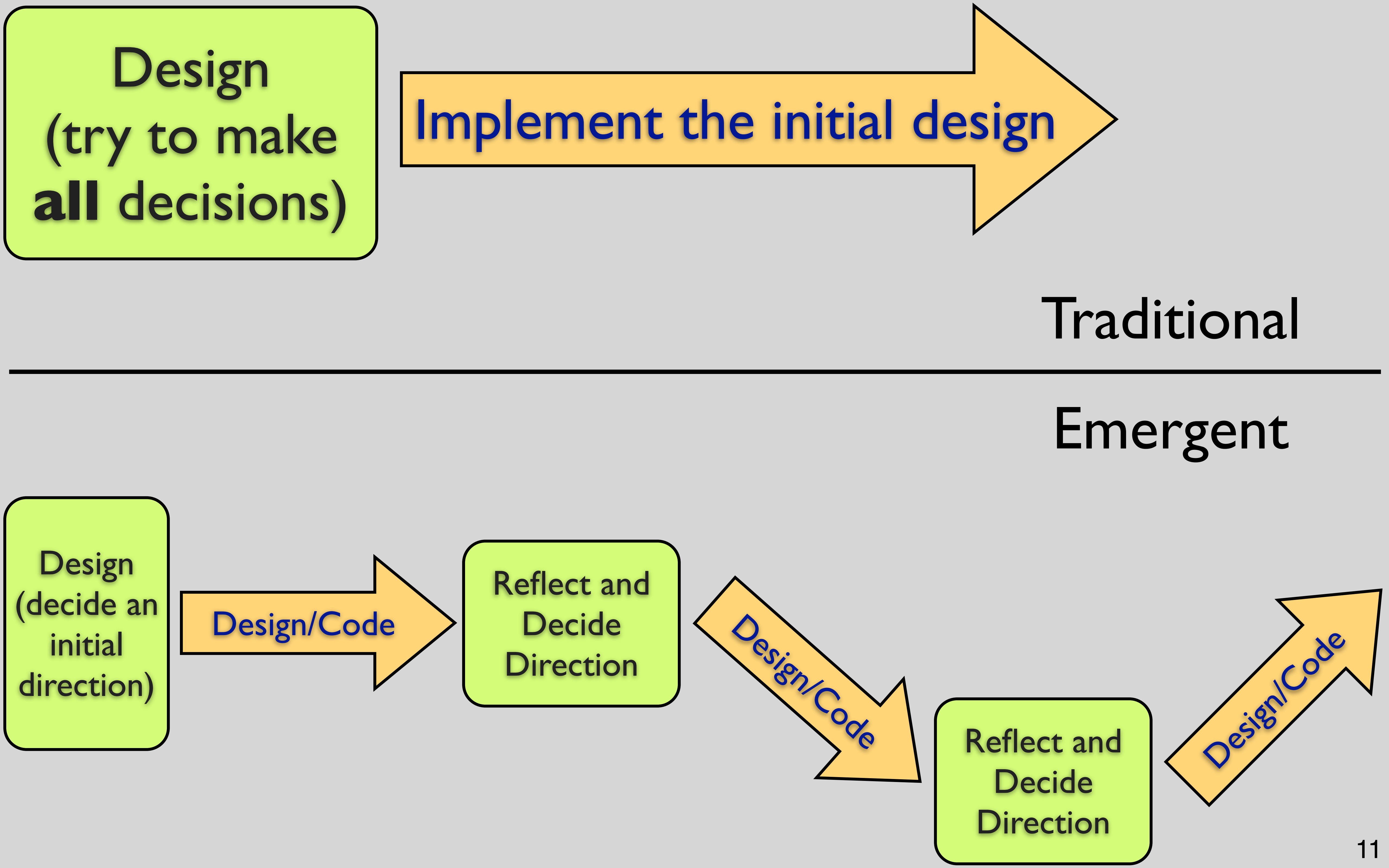
Refactoring visualized



Cost of change increases rapidly!

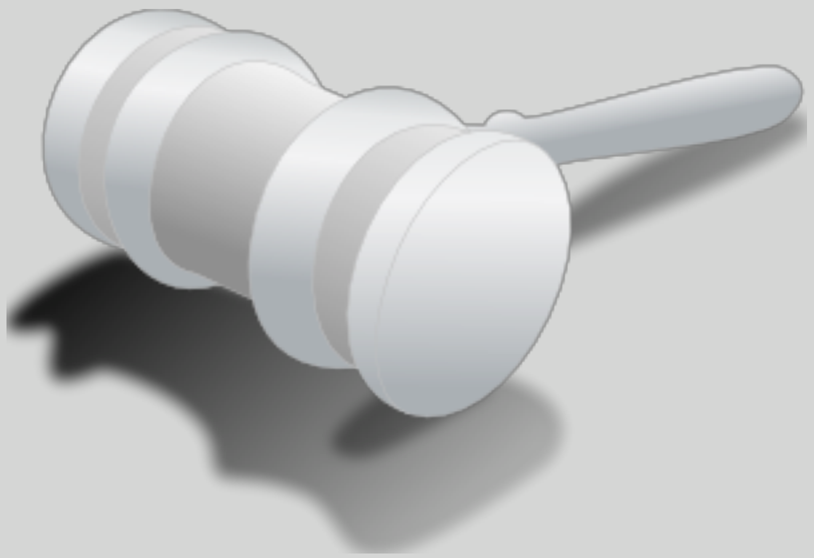
Cost of change does not increase

Traditional vs Emergent


Design (try to make **all** decisions)
Implement the initial design
Traditional
Emergent
Design (decide an initial direction)
Design/Code
Reflect and Decide Direction
Design/Code
Reflect and Decide Direction
Design/Code

Traditional vs Emergent

Traditional design is about decision

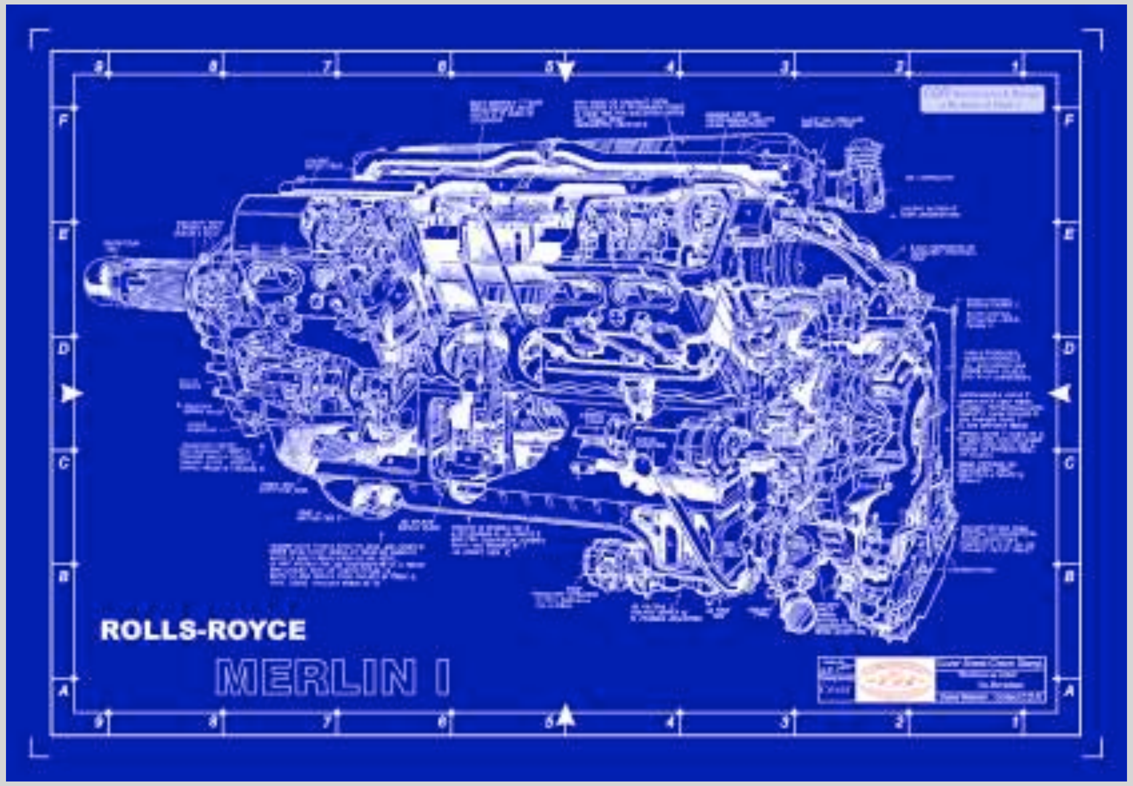


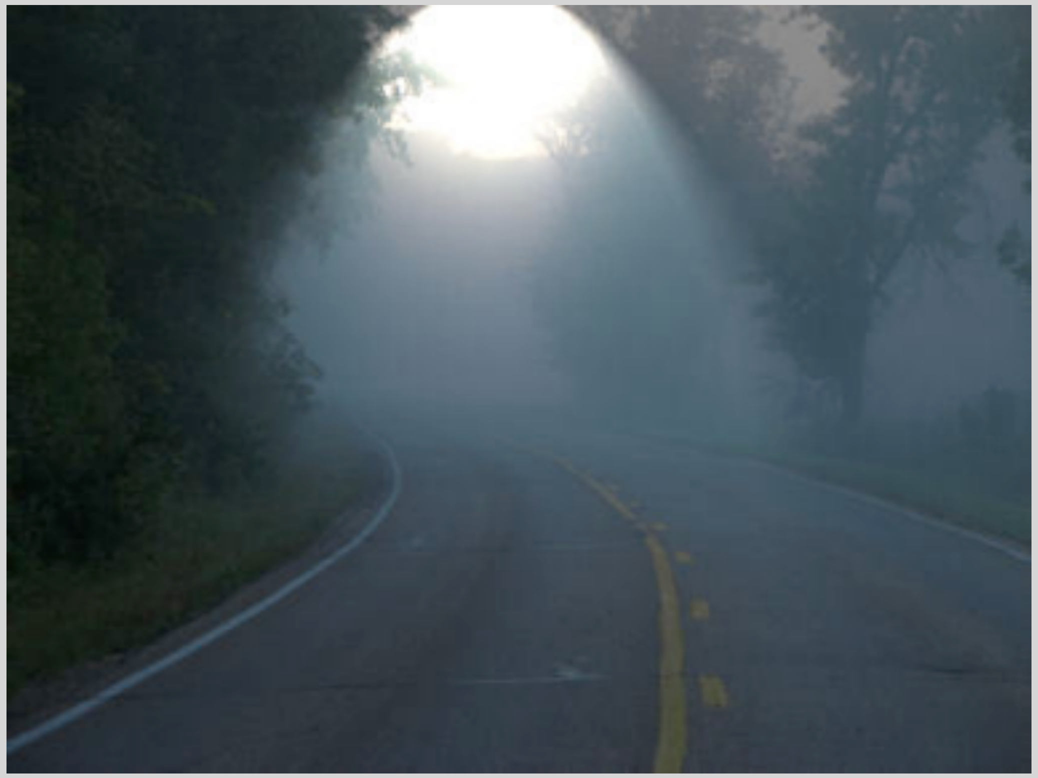
Decide the whole blueprint

Emergent design is about direction

Make sure we are going in the right direction

Traditional vs Emergent





Mindset:
We can think of most things beforehand. That will be most efficient

Mindset:
Premature speculation is probably wrong. Gradual discovery and learning leads to better solutions

Emergent Design

Let the design be the way it wants to be!
Code, listen you shall.

Steering the design

Steering the direction
of the design based on:

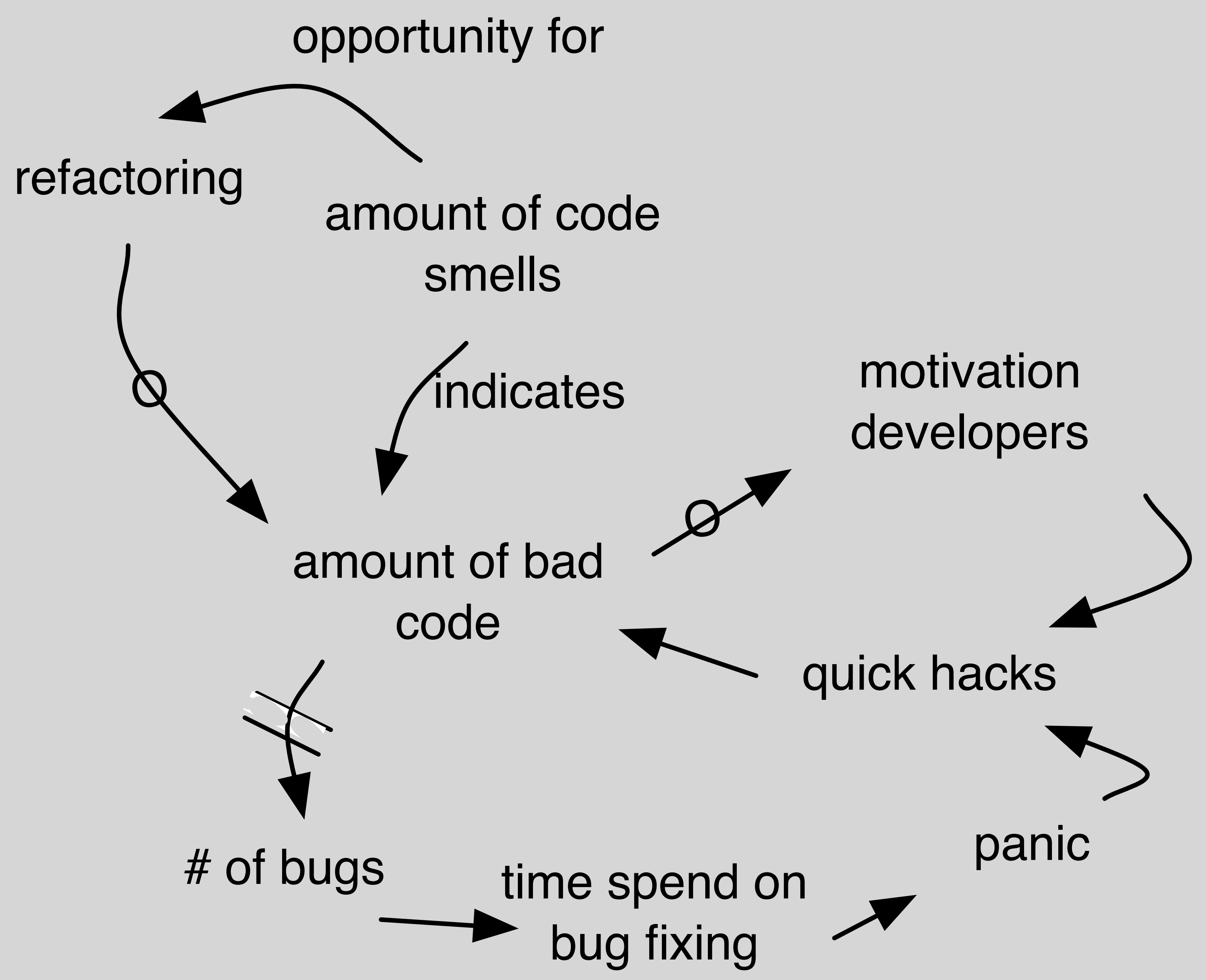
- Order of test-driving
- Principles
- Patterns

Odd-e

Buddi Refactoring

Refactor based on smells

Why? How?



opportunity for
refactoring
amount of code smells
indicates
O
motivation developers
amount of bad code
O
quick hacks
# of bugs
time spend on bug fixing
panic

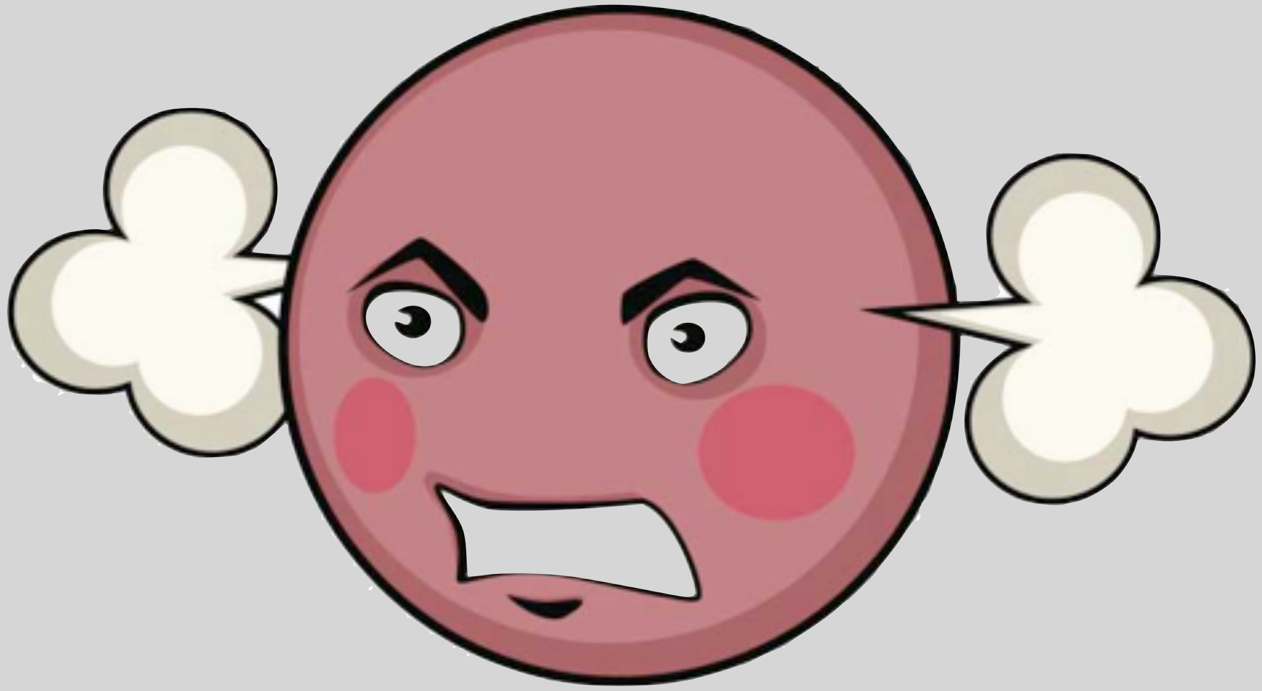
Refactoring

How do you know, when to refactor?

When your code smells!

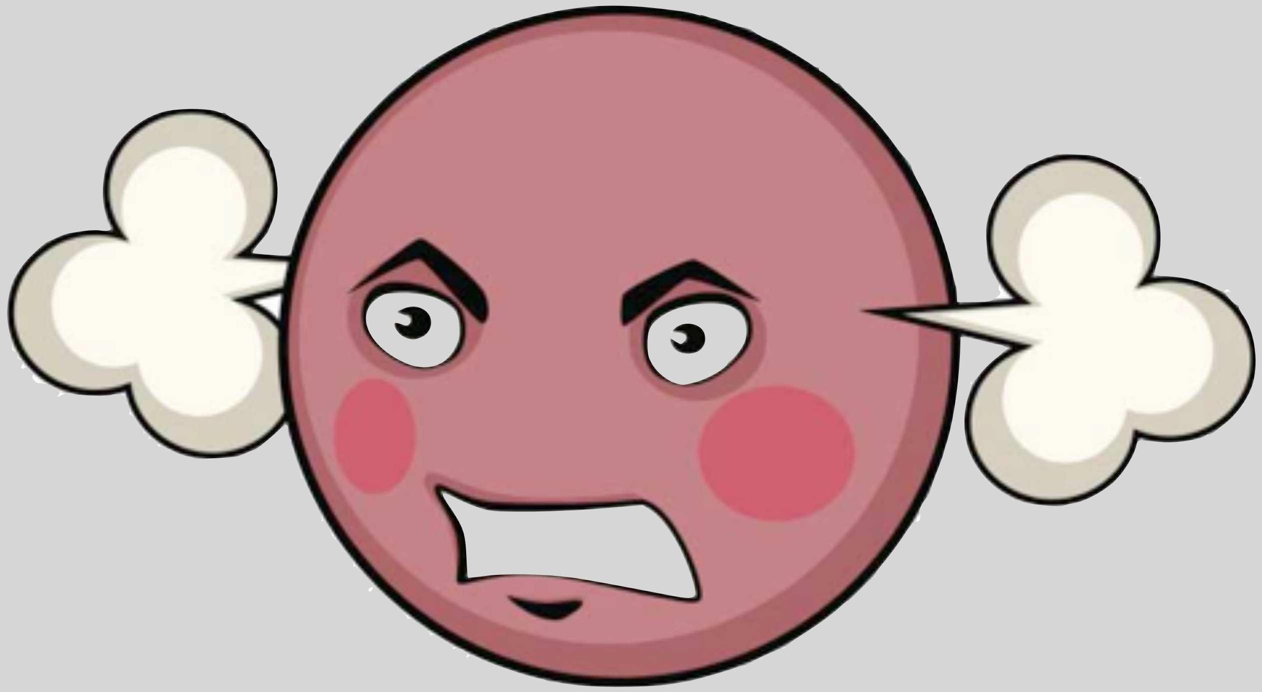
Good code an opinion?

Good code an opinion?



NO!

Good code an opinion?



NO!

Good code does **not** smell!

What about performance?

What about performance?

Not an excuse to write smelly code.

Messy - Debug Info

```
        public long getAmount(Date startDate, Date endDate){
            if (startDate.after(endDate))
                throw new RuntimeException("Start date cannot be before End Date!");

            Logger.getLogger(this.getClass().getName()).info("Starting to calculate the budgeted amount for " + getFullName() + " between " + star
    and " + endDate + ".");

            //If Start and End are in the same budget period
            if (getBudgetPeriodType().getStartOfBudgetPeriod(startDate).equals(
                        getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
    //              Logger.getLogger().info("Start Date and End Date are in the same period.");
                long amount = getAmount(startDate);
    //              Logger.getLogger().info("Amount = " + amount);
                long daysInPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
    //              Logger.getLogger().info("Days in Period = " + daysInPeriod);
                long daysBetween = DateUtil.getDaysBetween(startDate, endDate, true);
    //              Logger.getLogger().info("Days Between = " + daysBetween);

    //              Logger.getLogger().info("Returning " + (long) (((double) amount / (double) daysInPeriod) * daysBetween));
    //              Logger.getLogger().info("Finished calculating the budget amount.\n\n");
                return (long) (((double) amount / (double) daysInPeriod) * daysBetween);
            }

            //If the area between Start and End overlap at least two budget periods.
            if (getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).equals(
                        getBudgetPeriodType().getStartOfBudgetPeriod(endDate))
                        || getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).before(
                                getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
    //              Logger.getLogger().info("Start Date and End Date are in different budget periods.");
                long amountStartPeriod = getAmount(startDate);
    //              Logger.getLogger().info("Amount Start Period = " + amountStartPeriod);
                long daysInStartPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
    //              Logger.getLogger().info("Days in Start Period = " + daysInStartPeriod);
                long daysAfterStartDateInStartPeriod = DateUtil.getDaysBetween(startDate, getBudgetPeriodType().getEndOfBudgetPeriod(startDate),
    //              Logger.getLogger().info("Days After Start Date in Start Period = " + daysAfterStartDateInStartPeriod);
                double totalStartPeriod = (((double) amountStartPeriod / (double) daysInStartPeriod) * daysAfterStartDateInStartPeriod);
    //              Logger.getLogger().info("Total in Start Period = " + totalStartPeriod);

                double totalInMiddle = 0;
                for (String periodKey : getBudgetPeriods(
                            getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1),
                            getBudgetPeriodType().getBudgetPeriodOffset(endDate, -1))) {
                        totalInMiddle += getAmount(getPeriodDate(periodKey));
                        Logger.getLogger(this.getClass().getName()).info("Added " + getAmount(getPeriodDate(periodKey)) + " to total for one perio
    between; current value is " + totalInMiddle);
                }
    //              Logger.getLogger().info("Total in Middle = " + totalInMiddle);

                long amountEndPeriod = getAmount(endDate);
    //              Logger.getLogger().info("Amount End Period = " + amountEndPeriod);
```

Long Method

Large number of line of code
(anything over 5-10)

Too long!

```
    public long getAmount(Date startDate, Date endDate){
        if (startDate.after(endDate))
            throw new RuntimeException("Start date cannot be before End Date!");

        //If Start and End are in the same budget period
        if (getBudgetPeriodType().getStartOfBudgetPeriod(startDate).equals(
                getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
            long amount = getAmount(startDate);
            long daysInPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
            long daysBetween = DateUtil.getDaysBetween(startDate, endDate, true);

            return (long) (((double) amount / (double) daysInPeriod) * daysBetween);
        }

        //If the area between Start and End overlap at least two budget periods.
        if (getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).equals(
                getBudgetPeriodType().getStartOfBudgetPeriod(endDate))
                || getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).before(
                        getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
            long amountStartPeriod = getAmount(startDate);
            long daysInStartPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
            long daysAfterStartDateInStartPeriod = DateUtil.getDaysBetween(startDate, getBudgetPeriodType().getEndOfBudgetPeriod(startDate), true);
            double totalStartPeriod = (((double) amountStartPeriod / (double) daysInStartPeriod) * daysAfterStartDateInStartPeriod);

            double totalInMiddle = 0;
            for (String periodKey : getBudgetPeriods(
                    getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1),
                    getBudgetPeriodType().getBudgetPeriodOffset(endDate, -1))) {
                totalInMiddle += getAmount(getPeriodDate(periodKey));
            }

            long amountEndPeriod = getAmount(endDate);
            long daysInEndPeriod = getBudgetPeriodType().getDaysInPeriod(endDate);
            long daysBeforeEndDateInEndPeriod = DateUtil.getDaysBetween(getBudgetPeriodType().getStartOfBudgetPeriod(endDate), endDate, true);
            double totalEndPeriod = (long) (((double) amountEndPeriod / (double) daysInEndPeriod) * daysBeforeEndDateInEndPeriod);

            return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
        }

        throw new RuntimeException("You should not be here.  We have returned all legitimate numbers from getAmount(Date, Date) in BudgetCategoryImpl.
Please contact Wyatt Olson with details on how you got here (what steps did you perform in Buddi to get this error message).");
    }
```

Long method often indicates other smells

Safety first!

Check if the test quality

```java
public class BudgetCategoryTest {

    @Test
    public void testBudgetCategory(){
        try {
            BudgetCategoryType bct = ModelFactory.getBudgetCategoryType(BudgetCategoryTypes.BUDGET_CATEGORY_TYPE_MONTH);
            BudgetCategory bc = ModelFactory.createBudgetCategory("Test", bct, false);
            bc.setAmount(DateUtil.getDate(2007, Calendar.APRIL, 1), 100);
            bc.setAmount(DateUtil.getDate(2007, Calendar.MAY, 1), 200);
            bc.setAmount(DateUtil.getDate(2007, Calendar.JUNE, 1), 240);
            bc.setAmount(DateUtil.getDate(2007, Calendar.JULY, 1), 10);
            bc.setAmount(DateUtil.getDate(2007, Calendar.AUGUST, 1), 130);
            bc.setAmount(DateUtil.getDate(2007, Calendar.SEPTEMBER, 1), 13);
            bc.setAmount(DateUtil.getDate(2007, Calendar.OCTOBER, 1), 333);
            bc.setAmount(DateUtil.getDate(2007, Calendar.NOVEMBER, 1), 331);

            assertEquals((double) 100, bc.getAmount(DateUtil.getDate(2007, Calendar.APRIL, 1)), 1);
            assertEquals((double) 100, bc.getAmount(DateUtil.getDate(2007, Calendar.APRIL, 10)), 1);
            assertEquals((double) 100, bc.getAmount(DateUtil.getDate(2007, Calendar.APRIL, 28)), 1);

            assertEquals((double) 300, bc.getAmount(DateUtil.getDate(2007, Calendar.APRIL, 1), DateUtil.getDate(2007, Calendar.MAY, 31)), 1);
            assertEquals((double) 149, bc.getAmount(DateUtil.getDate(2007, Calendar.APRIL, 15), DateUtil.getDate(2007, Calendar.MAY, 15)), 1);

        }
        catch (Exception e){
            fail("Exception: " + e);
        }
    }

    @Test
    public void budgetPeriodWeekly() throws Exception {
        BudgetCategoryType bct = ModelFactory.getBudgetCategoryType(BudgetCategoryTypes.BUDGET_CATEGORY_TYPE_WEEK);
        BudgetCategory bc = ModelFactory.createBudgetCategory("Weekly Test", bct, false);

        bc.setAmount(DateUtil.getDate(2007, Calendar.SEPTEMBER, 10), 100);
        bc.setAmount(DateUtil.getDate(2007, Calendar.SEPTEMBER, 17), 100);
        bc.setAmount(DateUtil.getDate(2007, Calendar.SEPTEMBER, 24), 100);
        bc.setAmount(DateUtil.getDate(2007, Calendar.OCTOBER, 1), 100);
        bc.setAmount(DateUtil.getDate(2007, Calendar.OCTOBER, 8), 100);
        bc.setAmount(DateUtil.getDate(2007, Calendar.OCTOBER, 15), 100);
        bc.setAmount(DateUtil.getDate(2007, Calendar.OCTOBER, 22), 100);
        bc.setAmount(DateUtil.getDate(2007, Calendar.OCTOBER, 29), 100);
        bc.setAmount(DateUtil.getDate(2007, Calendar.NOVEMBER, 5), 100);
        bc.setAmount(DateUtil.getDate(2007, Calendar.NOVEMBER, 12), 100);

        assertEquals(100l,
                bc.getAmount(DateUtil.getDate(2007, Calendar.OCTOBER, 1)));
        assertEquals(14l,
                bc.getAmount(
                        DateUtil.getDate(2007, Calendar.OCTOBER, 1),
                        DateUtil.getDate(2007, Calendar.OCTOBER, 1)));
        assertEquals(28l,
                bc.getAmount(
                        DateUtil.getDate(2007, Calendar.OCTOBER, 1),
                        DateUtil.getDate(2007, Calendar.OCTOBER, 2)));
        assertEquals(100l,
                bc.getAmount(
                        bct.getStartOfBudgetPeriod(DateUtil.getDate(2007, Calendar.OCTOBER, 1)),
                        bct.getEndOfBudgetPeriod(DateUtil.getDate(2007, Calendar.OCTOBER, 1))));
        assertEquals(200l,
                bc.getAmount(
                        bct.getStartOfBudgetPeriod(DateUtil.getDate(2007, Calendar.OCTOBER, 1)),
                        bct.getEndOfBudgetPeriod(bct.getBudgetPeriodOffset(DateUtil.getDate(2007, Calendar.OCTOBER, 1), 1))));
        assertEquals(400l,
                bc.getAmount(
                        bct.getStartOfBudgetPeriod(DateUtil.getDate(2007, Calendar.OCTOBER, 1)),
                        bct.getEndOfBudgetPeriod(bct.getBudgetPeriodOffset(DateUtil.getDate(2007, Calendar.OCTOBER, 1), 3))));
        assertEquals(442l,
                bc.getAmount(
                        DateUtil.getDate(2007, Calendar.OCTOBER, 1),
```

Some test,

not very good ones.

Add tests

```
@Test (expected=RuntimeException.class)
public void testBeginDateIsLaterThanStartDateThrowsAnException() throws Exception {
      bc.getAmount(fourthOfJuly2011 , secondOfJuly2011);
}

@Test
public void getAmountOfARangeInsideABudgetPeriod() throws Exception {
      bc.setAmount(secondOfJuly2011, 31);
      assertEquals(3, bc.getAmount(secondOfJuly2011, fourthOfJuly2011));
}

@Test
public void getAmountOfARangeTwoNeigbouringBudgetPeriods() throws Exception {
      bc.setAmount(twentyJune2011, 300);
      bc.setAmount(secondOfJuly2011, 31);
      assertEquals(114, bc.getAmount(twentyJune2011, fourthOfJuly2011));
}

@Test
public void getAmountOfARangeWithManyBudgetPeriodsInBetween() throws Exception {
      bc.setAmount(tenthMarch2011, 31);
      bc.setAmount(twentyApril2011, 3000);
      bc.setAmount(twentyMay2011, 3100);
      bc.setAmount(twentyJune2011, 3000);
      bc.setAmount(secondOfJuly2011, 31);
      assertEquals(9100 + 2 + 22, bc.getAmount(tenthMarch2011, secondOfJuly2011));
}
```

Buckled? Ready?

Duplicate code

Two pieces of code that are conceptually the same

```java
//If Start and End are in the same budget period
if (getBudgetPeriodType().getStartOfBudgetPeriod(startDate).equals(
        getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
    long amount = getAmount(startDate);
    long daysInPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
    long daysBetween = DateUtil.getDaysBetween(startDate, endDate, true);

    return (long) (((double) amount / (double) daysInPeriod) * daysBetween);
}

//If the area between Start and End overlap at least two budget periods.
if (getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).equals(
        getBudgetPeriodType().getStartOfBudgetPeriod(endDate))
        || getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).before(
                getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
    long amountStartPeriod = getAmount(startDate);
    long daysInStartPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
    long daysAfterStartDateInStartPeriod = DateUtil.getDaysBetween(startDate,
        getBudgetPeriodType().getEndOfBudgetPeriod(startDate), true);
    double totalStartPeriod =
        (((double) amountStartPeriod / (double) daysInStartPeriod) * daysAfterStartDateInStartPeriod);

    double totalInMiddle = 0;
    for (String periodKey : getBudgetPeriods(
            getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1),
            getBudgetPeriodType().getBudgetPeriodOffset(endDate, -1))) {
        totalInMiddle += getAmount(getPeriodDate(periodKey));
    }

    long amountEndPeriod = getAmount(endDate);
    long daysInEndPeriod = getBudgetPeriodType().getDaysInPeriod(endDate);
    long daysBeforeEndDateInEndPeriod =
        DateUtil.getDaysBetween(getBudgetPeriodType().getStartOfBudgetPeriod(endDate), endDate, true);
    double totalEndPeriod =
        (long) (((double) amountEndPeriod / (double) daysInEndPeriod) * daysBeforeEndDateInEndPeriod);

    return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
}
```

```java
//If Start and End are in the same budget period
if (getBudgetPeriodType().getStartOfBudgetPeriod(startDate).equals(
        getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
    long amount = getAmount(startDate);
    long daysInPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
    long daysBetween = DateUtil.getDaysBetween(startDate, endDate, true);

    return (long) (((double) amount / (double) daysInPeriod) * daysBetween);
}

//If the area between Start and End overlap at least two budget periods.
if (getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).equals(
        getBudgetPeriodType().getStartOfBudgetPeriod(endDate))
        || getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).before(
                getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
    long amountStartPeriod = getAmount(startDate);
    long daysInStartPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
    long daysAfterStartDateInStartPeriod = DateUtil.getDaysBetween(startDate,
        getBudgetPeriodType().getEndOfBudgetPeriod(startDate), true);
    double totalStartPeriod =
        (((double) amountStartPeriod / (double) daysInStartPeriod) * daysAfterStartDateInStartPeriod);

    double totalInMiddle = 0;
    for (String periodKey : getBudgetPeriods(
            getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1),
            getBudgetPeriodType().getBudgetPeriodOffset(endDate, -1))) {
        totalInMiddle += getAmount(getPeriodDate(periodKey));
    }

    long amountEndPeriod = getAmount(endDate);
    long daysInEndPeriod = getBudgetPeriodType().getDaysInPeriod(endDate);
    long daysBeforeEndDateInEndPeriod =
        DateUtil.getDaysBetween(getBudgetPeriodType().getStartOfBudgetPeriod(endDate), endDate, true);
    double totalEndPeriod =
        (long) (((double) amountEndPeriod / (double) daysInEndPeriod) * daysBeforeEndDateInEndPeriod);

    return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
}
```

```
//If Start and End are in the same budget period
if (getBudgetPeriodType().getStartOfBudgetPeriod(startDate).equals(
        getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
    long amount = getAmount(startDate);
    long daysInPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
    long daysBetween = DateUtil.getDaysBetween(startDate, endDate, true);

    return (long) (((double) amount / (double) daysInPeriod) * daysBetween);
}

//If the area between Start and End overlap at least two budget periods.
if (getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).equals(
        getBudgetPeriodType().getStartOfBudgetPeriod(endDate))
        || getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).before(
                getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
    long amountStartPeriod = getAmount(startDate);
    long daysInStartPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
    long daysAfterStartDateInStartPeriod = DateUtil.getDaysBetween(startDate,
        getBudgetPeriodType().getEndOfBudgetPeriod(startDate), true);
    double totalStartPeriod =
        (((double) amountStartPeriod / (double) daysInStartPeriod) * daysAfterStartDateInStartPeriod);

    double totalInMiddle = 0;
    for (String periodKey : getBudgetPeriods(
            getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1),
            getBudgetPeriodType().getBudgetPeriodOffset(endDate, -1))) {
        totalInMiddle += getAmount(getPeriodDate(periodKey));
    }

    long amountEndPeriod = getAmount(endDate);
    long daysInEndPeriod = getBudgetPeriodType().getDaysInPeriod(endDate);
    long daysBeforeEndDateInEndPeriod =
        DateUtil.getDaysBetween(getBudgetPeriodType().getStartOfBudgetPeriod(endDate), endDate, true);
    double totalEndPeriod =
        (long) (((double) amountEndPeriod / (double) daysInEndPeriod) * daysBeforeEndDateInEndPeriod);

    return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
}
```

```
//If Start and End are in the same budget period
if (getBudgetPeriodType().getStartOfBudgetPeriod(startDate).equals(
        getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
    long amount = getAmount(startDate);
    long daysInPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
    long daysBetween = DateUtil.getDaysBetween(startDate, endDate, true);

    return (long) (((double) amount / (double) daysInPeriod) * daysBetween);
}

//If the area between Start and End overlap at least two budget periods.
if (getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).equals(
        getBudgetPeriodType().getStartOfBudgetPeriod(endDate))
        || getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).before(
                getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
    long amountStartPeriod = getAmount(startDate);
    long daysInStartPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
    long daysAfterStartDateInStartPeriod = DateUtil.getDaysBetween(startDate,
        getBudgetPeriodType().getEndOfBudgetPeriod(startDate), true);
    double totalStartPeriod =
        (((double) amountStartPeriod / (double) daysInStartPeriod) * daysAfterStartDateInStartPeriod);

    double totalInMiddle = 0;
    for (String periodKey : getBudgetPeriods(
            getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1),
            getBudgetPeriodType().getBudgetPeriodOffset(endDate, -1))) {
        totalInMiddle += getAmount(getPeriodDate(periodKey));
    }

    long amountEndPeriod = getAmount(endDate);
    long daysInEndPeriod = getBudgetPeriodType().getDaysInPeriod(endDate);
    long daysBeforeEndDateInEndPeriod =
        DateUtil.getDaysBetween(getBudgetPeriodType().getStartOfBudgetPeriod(endDate), endDate, true);
    double totalEndPeriod =
        (long) (((double) amountEndPeriod / (double) daysInEndPeriod) * daysBeforeEndDateInEndPeriod);

    return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
}
```

First make duplication obvious (same)

```
long amount = getAmount(startDate);
long daysInPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
long daysBetween = DateUtil.getDaysBetween(startDate, endDate, true);

return (long) (((double) amount / (double) daysInPeriod) * daysBetween);
```

Extract Local Variable (Alt-Shift-L)

```
long amount = getAmount(startDate);
long daysInPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
long daysBetween = DateUtil.getDaysBetween(startDate, endDate, true);
double amountInPeriod = (long) (((double) amount / (double) daysInPeriod) * daysBetween);

return (long) amountInPeriod;
```

```java
long amountStartPeriod = getAmount(startDate);
long daysInStartPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
long daysAfterStartDateInStartPeriod =
    DateUtil.getDaysBetween(startDate, getBudgetPeriodType().getEndOfBudgetPeriod(startDate), true);
double totalStartPeriod =
    (((double) amountStartPeriod / (double) daysInStartPeriod) * daysAfterStartDateInStartPeriod);
```

Extract Local Variable (Alt-Shift-L)

```java
Date endOfBudgetPeriod = getBudgetPeriodType().getEndOfBudgetPeriod(startDate);
long amountStartPeriod = getAmount(startDate);
long daysInStartPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
long daysAfterStartDateInStartPeriod = DateUtil.getDaysBetween(startDate, endOfBudgetPeriod, true);
double totalStartPeriod =
    (((double) amountStartPeriod / (double) daysInStartPeriod) * daysAfterStartDateInStartPeriod);
```

Remove useless parenthesis

```
double totalStartPeriod =
    (((double) amountStartPeriod / (double) daysInStartPeriod) * daysAfterStartDateInStartPeriod);
```

⇩

```
double totalStartPeriod =
    ((double) amountStartPeriod / (double) daysInStartPeriod) * daysAfterStartDateInStartPeriod;
```

```
double totalEndPeriod =
    (long) (((double) amountEndPeriod / (double) daysInEndPeriod) * daysBeforeEndDateInEndPeriod);
```

⇩

Casts without rounding?

```
double totalEndPeriod =
    ((double) amountEndPeriod / (double) daysInEndPeriod) * daysBeforeEndDateInEndPeriod;
```

Without this, Eclipse can't detect the duplication :(

```java
long amountEndPeriod = getAmount(endDate);
long daysInEndPeriod = getBudgetPeriodType().getDaysInPeriod(endDate);
long daysBeforeEndDateInEndPeriod =
    DateUtil.getDaysBetween(getBudgetPeriodType().getStartOfBudgetPeriod(endDate), endDate, true);
double totalEndPeriod =
    (long) ((double) amountEndPeriod / (double) daysInEndPeriod) * daysBeforeEndDateInEndPeriod;
```

Extract Local Variable (Alt-Shift-L)

```java
Date startOfBudgetPeriod = getBudgetPeriodType().getStartOfBudgetPeriod(endDate);
long amountEndPeriod = getAmount(startOfBudgetPeriod);
long daysInEndPeriod = getBudgetPeriodType().getDaysInPeriod(startOfBudgetPeriod);
long daysBeforeEndDateInEndPeriod =
    DateUtil.getDaysBetween(startOfBudgetPeriod, endDate, true);
double totalEndPeriod =
    (long) ((double) amountEndPeriod / (double) daysInEndPeriod) * daysBeforeEndDateInEndPeriod;
```

Use start instead of end of period.

```
long amountEndPeriod = getAmount(endDate);
long daysInEndPeriod = getBudgetPeriodType().getDaysInPeriod(endDate);
long daysBeforeEndDateInEndPeriod = DateUtil.getDaysBetween(startOfBudgetPeriod, endDate, true);
```

```
long amountEndPeriod = getAmount(startOfBudgetPeriod);
long daysInEndPeriod = getBudgetPeriodType().getDaysInPeriod(startOfBudgetPeriod);
long daysBeforeEndDateInEndPeriod = DateUtil.getDaysBetween(startOfBudgetPeriod, endDate, true);
```

Note: Parameter of getAmount and getDaysInPeriod is a Period, not a Date.

Therefore both start and end date of period will work!

Duplication is exactly the same now.

```
if (getBudgetPeriodType().getStartOfBudgetPeriod(startDate).equals(
            getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
      long amount = getAmount(startDate);
      long daysInPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
      long daysBetween = DateUtil.getDaysBetween(startDate, endDate, true);
      double amountInPeriod = ((double) amount / (double) daysInPeriod) * daysBetween;
      return (long) amountInPeriod;
}

//If the area between Start and End overlap at least two budget periods.
if (getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).equals(
            getBudgetPeriodType().getStartOfBudgetPeriod(endDate))
            || getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).before(
                        getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
      Date endOfBudgetPeriod = getBudgetPeriodType().getEndOfBudgetPeriod(startDate);
      long amountStartPeriod = getAmount(startDate);
      long daysInStartPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
      long daysAfterStartDateInStartPeriod = DateUtil.getDaysBetween(startDate, endOfBudgetPeriod, true);
      double totalStartPeriod = ((double) amountStartPeriod / (double) daysInStartPeriod) * daysAfterStartDateInStartPeriod;

      double totalInMiddle = 0;
      for (String periodKey : getBudgetPeriods(
                  getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1),
                  getBudgetPeriodType().getBudgetPeriodOffset(endDate, -1))) {
            totalInMiddle += getAmount(getPeriodDate(periodKey));
      }

      Date startOfBudgetPeriod = getBudgetPeriodType().getStartOfBudgetPeriod(endDate);
      long amountEndPeriod = getAmount(startOfBudgetPeriod);
      long daysInEndPeriod = getBudgetPeriodType().getDaysInPeriod(startOfBudgetPeriod);
      long daysBeforeEndDateInEndPeriod = DateUtil.getDaysBetween(startOfBudgetPeriod, endDate, true);
      double totalEndPeriod = ((double) amountEndPeriod / (double) daysInEndPeriod) * daysBeforeEndDateInEndPeriod;
      return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
}
```

Odd-e

Then extract duplication

Extract Method
Alt-Shift-M

```
//If Start and End are in the same budget period
if (getBudgetPeriodType().getStartOfBudgetPeriod(startDate).equals(
        getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
    double amountInPeriod = getAmountInPeriod(startDate, endDate);
    return (long) amountInPeriod;
}

//If the area between Start and End overlap at least two budget periods.
if (getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).equals(
        getBudgetPeriodType().getStartOfBudgetPeriod(endDate))
        || getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).before(
                getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
    Date endOfBudgetPeriod = getBudgetPeriodType().getEndOfBudgetPeriod(startDate);
    double totalStartPeriod = getAmountInPeriod(startDate, endOfBudgetPeriod);

    double totalInMiddle = 0;
    for (String periodKey : getBudgetPeriods(
            getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1),
            getBudgetPeriodType().getBudgetPeriodOffset(endDate, -1))) {
        totalInMiddle += getAmount(getPeriodDate(periodKey));
    }

    Date startOfBudgetPeriod = getBudgetPeriodType().getStartOfBudgetPeriod(endDate);
    double totalEndPeriod = getAmountInPeriod(startOfBudgetPeriod, endDate);
    return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
}
```

Good Principles

DRY
Don't Repeat Yourself

Once And Only Once

DO

DO NOT

DO NOT

COPY

DO NOT

COPY PASTE

Magic Numbers

A constant appears in the code

1 - 1

```
//If the area between Start and End overlap at least two budget periods.
if (getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).equals(
        getBudgetPeriodType().getStartOfBudgetPeriod(endDate))
        || getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).before(
                getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
    Date endOfBudgetPeriod = getBudgetPeriodType().getEndOfBudgetPeriod(startDate);
    double totalStartPeriod = getAmountInPeriod(startDate, endOfBudgetPeriod);

    double totalInMiddle = 0;
    for (String periodKey : getBudgetPeriods(
            getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1),
            getBudgetPeriodType().getBudgetPeriodOffset(endDate, -1))) {
        totalInMiddle += getAmount(getPeriodDate(periodKey));
    }

    Date startOfBudgetPeriod = getBudgetPeriodType().getStartOfBudgetPeriod(endDate);
    double totalEndPeriod = getAmountInPeriod(startOfBudgetPeriod, endDate);
    return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
}
```

Extract to method

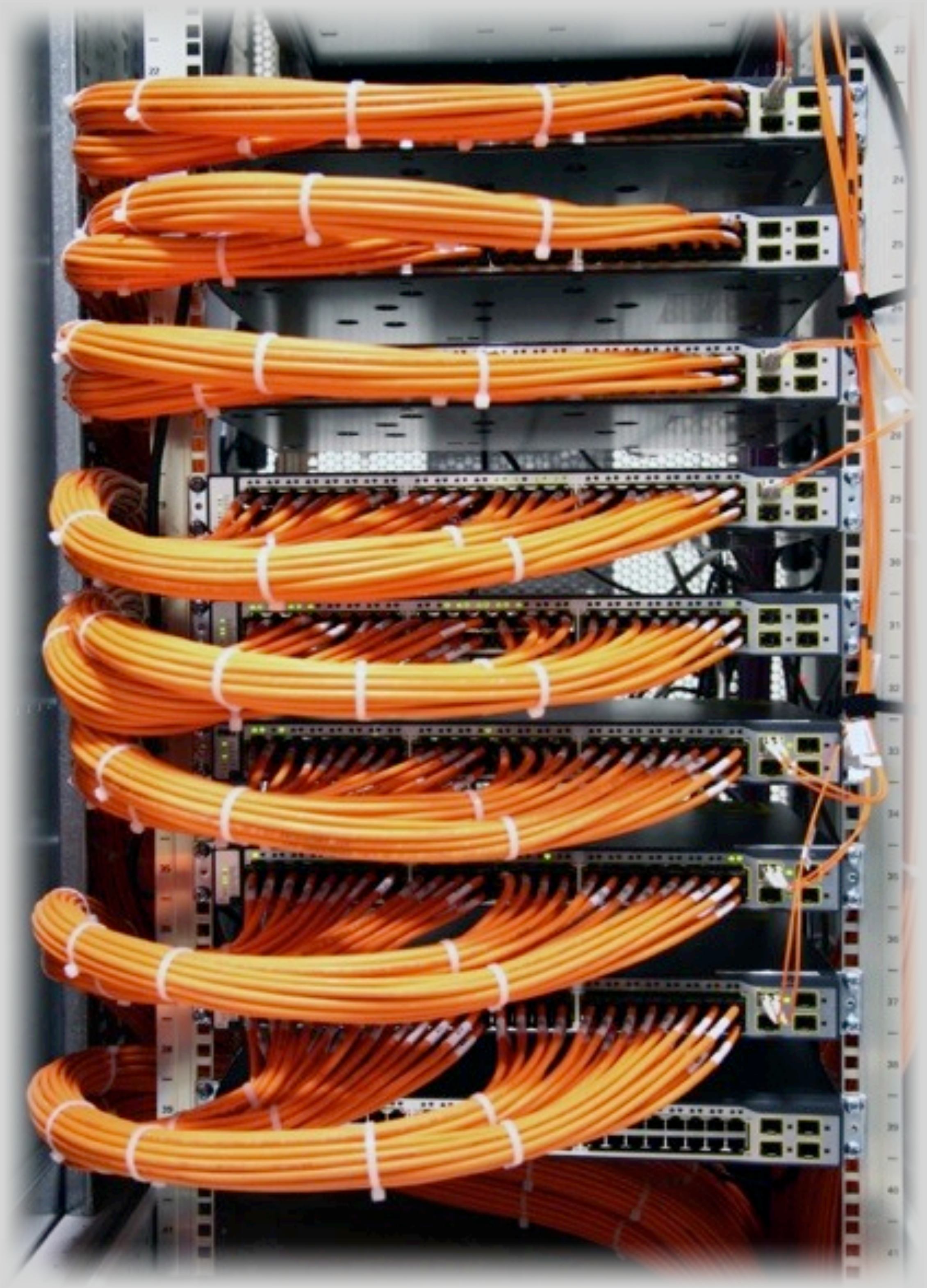
```
//If the area between Start and End overlap at least two budget periods.
if (getBudgetPeriodType().getStartOfNextBudgetPeriod(startDate).equals(
        getBudgetPeriodType().getStartOfBudgetPeriod(endDate))
        || getBudgetPeriodType().getStartOfNextBudgetPeriod(startDate).before(
                getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
    Date endOfBudgetPeriod = getBudgetPeriodType().getEndOfBudgetPeriod(startDate);
    double totalStartPeriod = getAmountInPeriod(startDate, endOfBudgetPeriod);

    double totalInMiddle = 0;
    for (String periodKey : getBudgetPeriods(
            getBudgetPeriodType().getStartOfNextBudgetPeriod(startDate),
            getBudgetPeriodType().getStartOfPreviousBudgetPeriod(startDate))) {
        totalInMiddle += getAmount(getPeriodDate(periodKey));
    }

    Date startOfBudgetPeriod = getBudgetPeriodType().getStartOfBudgetPeriod(endDate);
    double totalEndPeriod = getAmountInPeriod(startOfBudgetPeriod, endDate);
    return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
}
```

Additional cleanup

```
//If Start and End are in the same budget period
if (getBudgetPeriodType().getStartOfBudgetPeriod(startDate).equals(
        getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
    return (long) getAmountInPeriod(startDate, endDate);
}

//If the area between Start and End overlap at least two budget periods.
if (getBudgetPeriodType().getStartOfNextBudgetPeriod(startDate).equals(
        getBudgetPeriodType().getStartOfBudgetPeriod(endDate))
        || getBudgetPeriodType().getStartOfNextBudgetPeriod(startDate).before(
                getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
    double totalStartPeriod = getAmountInPeriod(startDate, getBudgetPeriodType().getEndOfBudgetPeriod(startDate));

    double totalInMiddle = 0;
    for (String periodKey : getBudgetPeriods(
            getBudgetPeriodType().getStartOfNextBudgetPeriod(startDate),
            getBudgetPeriodType().getStartOfPreviousBudgetPeriod(endDate))) {
        totalInMiddle += getAmount(getPeriodDate(periodKey));
    }

    double totalEndPeriod = getAmountInPeriod(getBudgetPeriodType().getStartOfBudgetPeriod(endDate), endDate);
    return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
}
```

Inline Temp - Alt-Shift-I

Data Clumps



A couple of items frequently appear together

```
public long getAmount(Date startDate, Date endDate){
    if (startDate.after(endDate))
        throw new RuntimeException("Start date cannot be before End Date!");

    //If Start and End are in the same budget period
    if (getBudgetPeriodType().getStartOfBudgetPeriod(startDate).equals(
            getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
        return (long) getAmountInPeriod(startDate, endDate);
    }

    //If the area between Start and End overlap at least two budget periods.
    if (getBudgetPeriodType().getStartOfNextBudgetPeriod(startDate).equals(
            getBudgetPeriodType().getStartOfBudgetPeriod(endDate))
            || getBudgetPeriodType().getStartOfNextBudgetPeriod(startDate).before(
                getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
        double totalStartPeriod = getAmountInPeriod(startDate,
                getBudgetPeriodType().getEndOfBudgetPeriod(startDate));

        double totalInMiddle = 0;
        for (String periodKey : getBudgetPeriods(
                getBudgetPeriodType().getStartOfNextBudgetPeriod(startDate),
                getBudgetPeriodType().getStartOfPreviousBudgetPeriod(endDate))) {
            totalInMiddle += getAmount(getPeriodDate(periodKey));
        }

        double totalEndPeriod = getAmountInPeriod(getBudgetPeriodType().getStartOfBudgetPeriod(endDate),
                endDate);
        return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
    }

    throw new RuntimeException("You should not be here.  We have returned all legitimate numbers from getAmount(Dat
BudgetCategoryImpl.  Please contact Wyatt Olson with details on how you got here (what steps did you perform in Buddi to
error message).");
}
```

Note: Also earlier we seen the smell that Date was used as Period

Add new abstraction

```
public long getAmount(Date startDate, Date endDate){
    if (startDate.after(endDate))
        throw new RuntimeException("Start date cannot be before End Date!");

    Period period = new Period(startDate, endDate);
```

```
public class Period {

    private final Date start;
    private final Date end;

    public Period(Date start, Date end) {
        this.start = start;
        this.end = end;
    }

}
```

Abstraction not yet used...

Use new abstraction

```
        //If Start and End are in the same budget period
        if (getBudgetPeriodType().getStartOfBudgetPeriod(period.getStartDate()).equals(
                getBudgetPeriodType().getStartOfBudgetPeriod(period.getEndDate()))){
            return (long) getAmountInPeriod(period.getStartDate(), period.getEndDate());
        }

        //If the area between Start and End overlap at least two budget periods.
        if (getBudgetPeriodType().getStartOfNextBudgetPeriod(period.getStartDate()).equals(
                getBudgetPeriodType().getStartOfBudgetPeriod(period.getEndDate()))
                || getBudgetPeriodType().getStartOfNextBudgetPeriod(period.getStartDate()).before(
                        getBudgetPeriodType().getStartOfBudgetPeriod(period.getEndDate()))){
            double totalStartPeriod = getAmountInPeriod(period.getStartDate(),
                    getBudgetPeriodType().getEndOfBudgetPeriod(period.getStartDate()));

            double totalInMiddle = 0;
            for (String periodKey : getBudgetPeriods(
                    getBudgetPeriodType().getStartOfNextBudgetPeriod(period.getStartDate()),
                    getBudgetPeriodType().getStartOfPreviousBudgetPeriod(period.getEndDate()))) {
                totalInMiddle += getAmount(getPeriodDate(periodKey));
            }

            double totalEndPeriod =
getAmountInPeriod(getBudgetPeriodType().getStartOfBudgetPeriod(period.getEndDate()),
                    period.getEndDate());
            return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
        }
```

Makes the code uglier....

Extract Method

Extract Method
Alt-Shift-M

```
    private long getAmount(Period period) {
        //If Start and End are in the same budget period
        if (getBudgetPeriodType().getStartOfBudgetPeriod(period.getStartDate()).equals(
                getBudgetPeriodType().getStartOfBudgetPeriod(period.getEndDate()))){
            return (long) getAmountInPeriod(period.getStartDate(), period.getEndDate());
        }

        //If the area between Start and End overlap at least two budget periods.
        if (getBudgetPeriodType().getStartOfNextBudgetPeriod(period.getStartDate()).equals(
                getBudgetPeriodType().getStartOfBudgetPeriod(period.getEndDate()))
                || getBudgetPeriodType().getStartOfNextBudgetPeriod(period.getStartDate()).before(
                        getBudgetPeriodType().getStartOfBudgetPeriod(period.getEndDate()))){
            double totalStartPeriod = getAmountInPeriod(period.getStartDate(),
                    getBudgetPeriodType().getEndOfBudgetPeriod(period.getStartDate()));

            double totalInMiddle = 0;
            for (String periodKey : getBudgetPeriods(
                    getBudgetPeriodType().getStartOfNextBudgetPeriod(period.getStartDate()),
                    getBudgetPeriodType().getStartOfPreviousBudgetPeriod(period.getEndDate()))) {
                totalInMiddle += getAmount(getPeriodDate(periodKey));
            }

            double totalEndPeriod = getAmountInPeriod(getBudgetPeriodType().getStartOfBudgetPeriod(period.getEndDate()
                    period.getEndDate());
            return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
        }

        throw new RuntimeException("You should not be here.  We have returned all legitimate numbers from
getAmount(Date, Date) in BudgetCategoryImpl.  Please contact Wyatt Olson with details on how you got here (what steps
did you perform in Buddi to get this error message).");
    }
```

And the original method

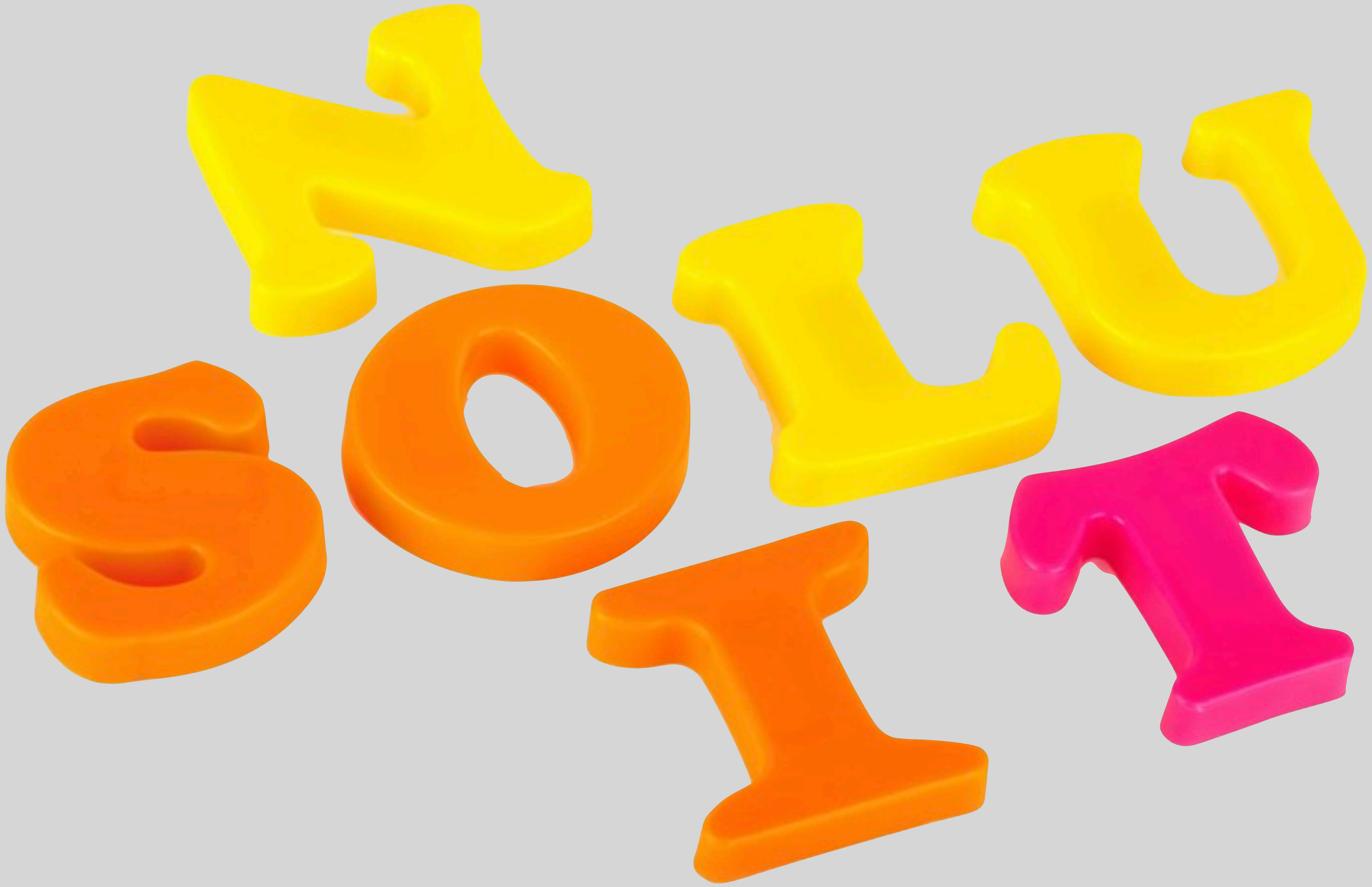
```java
public long getAmount(Date startDate, Date endDate){
    if (startDate.after(endDate))
        throw new RuntimeException("Start date cannot be before End Date!");

    return getAmount(new Period(startDate, endDate));
}
```

Requires no changes in the calling code.. yet

Data Clumps and other smells often suggest a common problem:

Missing domain objects!

Uncommunicative Name & Inconsistent Names



Ambiguous names

```
private double getAmountInPeriod(Date startDate, Date endDate) {
    long amount = getAmount(startDate);
    long daysInPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
    long daysBetween = DateUtil.getDaysBetween(startDate, endDate, true);
    double amountInPeriod = ((double) amount / (double) daysInPeriod) * daysBetween;
    return amountInPeriod;
}
```

The BudgetCategory class now contains:

3 methods called getAmount
1 method called getAmounts
1 method called getAmountInPeriod

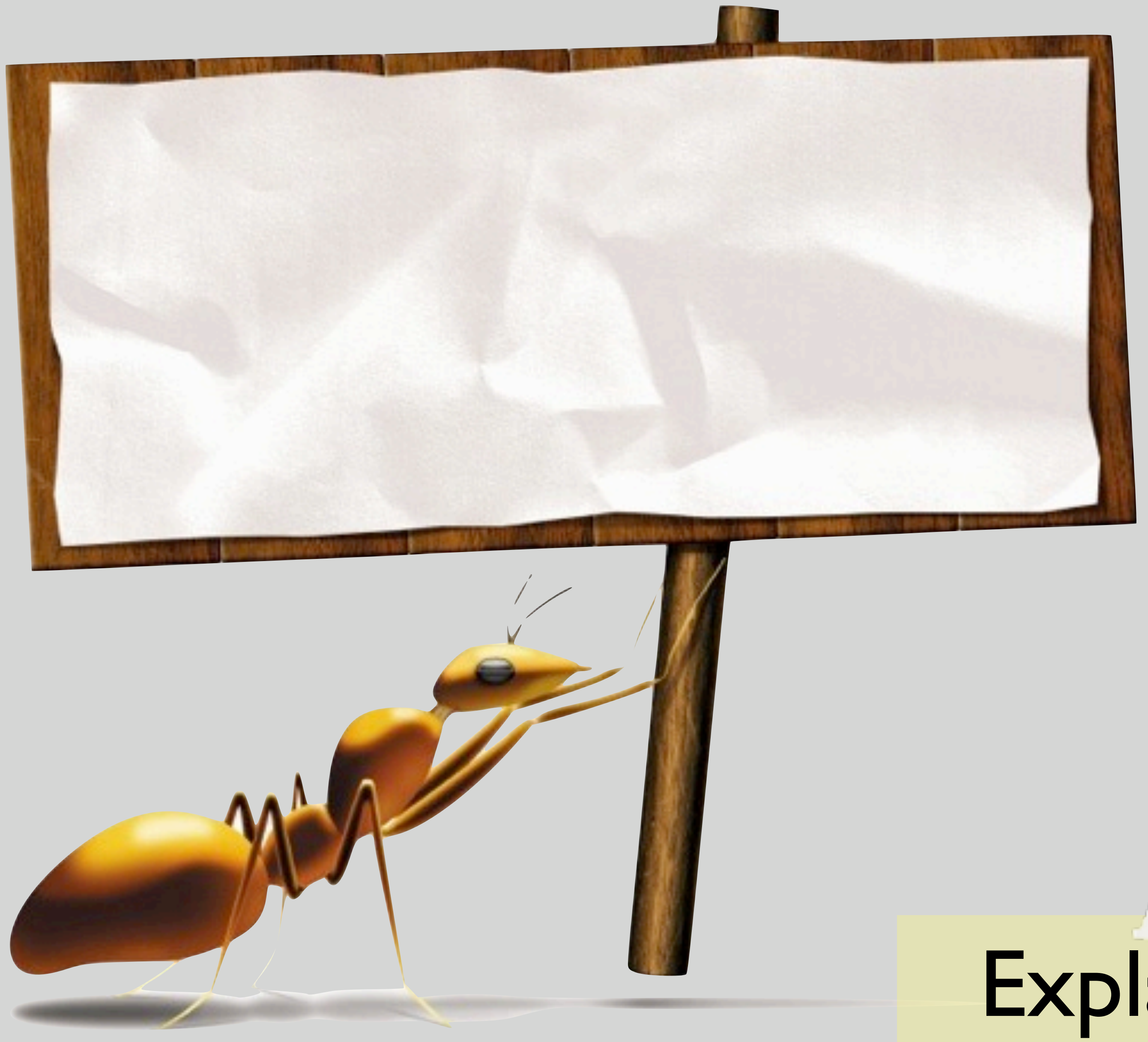
(and we caused 2 of these!)

Rename

```
public Map<String, Long> getAmounts()  ⟶  public Map<String, Long> getBudgetPeriods()

public void setAmounts(Map<String, Long> amounts)  ⟶  public void setBudgetPeriods(Map<String, Long> amounts)

public long getAmount(Date periodDate)  ⟶  public long getAmountFromBudgetPeriodContainingDate(Date periodDate)

public long getAmount(Date startDate, Date endDate) ⟶ public long getTotalAmountPeriod(Date startDate, Date endDate)

private long getAmount(Period period)  ⟶  private long getTotalAmountPeriod(Period period)

private double getAmount(Date startDate, Date endDate)

        ⟶  private double getAmountForPeriodWithinBudgetPeriodOfStartDate(Date startDate, Date endDate)
```

New names suggest new smells and domain object

Comments



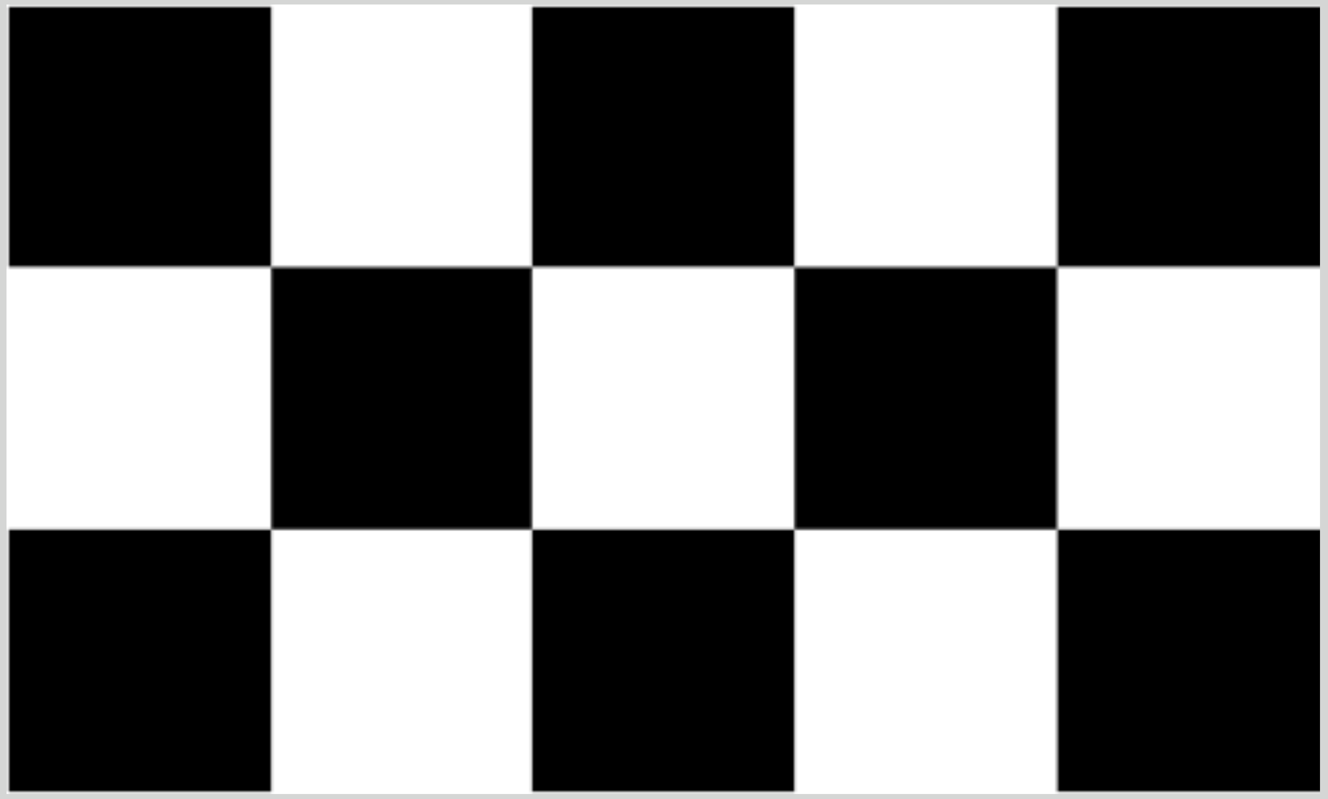
Explain how bad code works

```java
public long getAmount_step6(Period period){
    Date startDate = period.getStartDate();
    Date endDate = period.getEndDate();

    //If Start and End are in the same budget period
    if (getBudgetPeriodType().getStartOfBudgetPeriod(startDate).equals(
            getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
        return (long) getAmountInPeriod(startDate, endDate);
    }

    //If the area between Start and End overlap at least two budget periods.
    if (getBudgetPeriodType().getStartOfNextBudgetPeriod(startDate).equals(
            getBudgetPeriodType().getStartOfBudgetPeriod(endDate))
            || getBudgetPeriodType().getStartOfNextBudgetPeriod(startDate).before(
                    getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
        double totalStartPeriod = getAmountInPeriod(startDate, getBudgetPeriodType().getEndOfBudgetPeriod(startDate));

        double totalInMiddle = 0;
        for (String periodKey : getBudgetPeriods(
                getBudgetPeriodType().getStartOfNextBudgetPeriod(startDate),
                getBudgetPeriodType().getStartOfPreviousBudgetPeriod(endDate))) {
            totalInMiddle += getAmount(getPeriodDate(periodKey));
        }

        double totalEndPeriod = getAmountInPeriod(getBudgetPeriodType().getStartOfBudgetPeriod(endDate), endDate);
        return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
    }


    throw new RuntimeException("...");
}
```

Comments are deodorant

They hide other smells

Not all comments are bad.



Comments that describe code because it is unreadable by itself -> Bad

Comments that describe why the code works this way -> Sometimes ok!

Lazy Class

A class isn't doing much

```java
public class BudgetCategoryTypeMonthly extends BudgetCategoryType {

    public Date getStartOfBudgetPeriod(Date date) {
        return DateUtil.getStartOfMonth(date);
    }

    public Date getEndOfBudgetPeriod(Date date) {
        return DateUtil.getEndOfMonth(date);
    }

    public Date getBudgetPeriodOffset(Date date, int offset) {
        return getStartOfBudgetPeriod(DateUtil.addMonths(DateUtil.getStartOfMonth(date), 1 * offset));
    }

    public long getDaysInPeriod(Date date) {
        return DateUtil.getDaysInMonth(date);
    }

    public String getDateFormat() {
        return "MMM yyyy";
    }

    public String getName() {
        return BudgetCategoryTypes.BUDGET_CATEGORY_TYPE_MONTH.toString();
    }
}
```

Type in the classname also smells

Welcome BudgetPeriod!

```
private long getTotalAmountPeriod(Period period) {

    BudgetPeriod firstBudgetPeriod = createBudgetPeriodFromDate(period.getStartDate());

    //If Start and End are in the same budget period
    if (getBudgetPeriodType().getStartOfBudgetPeriod(period.getStartDate()).equals(
            getBudgetPeriodType().getStartOfBudgetPeriod(period.getEndDate()))){
        return (long) getAmountForPeriodWithinBudgetPeriodOfStartDate(period.getStartDate(), period.getEndDate());
    }
```

Abstraction suggested by:

- new method names
- lazy class
- comments

Use BudgetPeriod

```
BudgetPeriod firstBudgetPeriod = createBudgetPeriodFromDate(period.getStartDate());
BudgetPeriod lastBudgetPeriod = createBudgetPeriodFromDate(period.getEndDate());

if (firstBudgetPeriod.equals(lastBudgetPeriod)){
    return (long) getAmountForPeriodWithinBudgetPeriodOfStartDate(period.getStartDate(), period.getEndDate());
}
```

Now useless comment deleted

And first implementation

```
public class BudgetPeriod {

    private Period period;

    public BudgetPeriod(BudgetCategoryType type, Date date) {
        period = new Period (type.getStartOfBudgetPeriod(date), type.getEndOfBudgetPeriod(date));
    }

    @Override
    public boolean equals(Object object) {
        BudgetPeriod otherBudgetPeriod = (BudgetPeriod) object;
        return otherBudgetPeriod.period.equals(this.period);
    }

}
```

Yes, BudgetCategoryType is still there!
Removing will be gradual

Replace getBudgetPeriodType with BudgetPeriod

```
    private long getTotalAmountPeriod(Period period) {

        BudgetPeriod firstBudgetPeriod = createBudgetPeriodFromDate(period.getStartDate());
        BudgetPeriod lastBudgetPeriod = createBudgetPeriodFromDate(period.getEndDate());

        if (firstBudgetPeriod.equals(lastBudgetPeriod))
            return (long) getAmountForPeriodWithinBudgetPeriodOfStartDate(period.getStartDate(), period.getEndDate());

        //If the area between Start and End overlap at least two budget periods.
        if (firstBudgetPeriod.nextBudgetPeriod().getStartDate().equals(lastBudgetPeriod.getStartDate())
                || firstBudgetPeriod.nextBudgetPeriod().getStartDate().before(lastBudgetPeriod.getStartDate())){
            double totalStartPeriod = getAmountForPeriodWithinBudgetPeriodOfStartDate(period.getStartDate(),
                    firstBudgetPeriod.getEndDate());

            double totalInMiddle = 0;
            for (String periodKey : getBudgetPeriods(
                    firstBudgetPeriod.nextBudgetPeriod().getStartDate(),
                    lastBudgetPeriod.previousBudgetPeriod().getStartDate())) {
                totalInMiddle += getAmountFromBudgetPeriodContainingDate(getPeriodDate(periodKey));
            }

            double totalEndPeriod = getAmountForPeriodWithinBudgetPeriodOfStartDate(lastBudgetPeriod.getStartDate(),
period.getEndDate());
            return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
        }

        throw new RuntimeException("You should not be here.  We have returned all legitimate numbers from
getAmount(Date, Date) in BudgetCategoryImpl.  Please contact Wyatt Olson with details on how you got here (what steps
did you perform in Buddi to get this error message).");
    }
```

Added implementation

```
public BudgetPeriod nextBudgetPeriod() {
    return new BudgetPeriod (type, type.getBudgetPeriodOffset(period.getStartDate(), 1));
}

public Date getStartDate() {
    return period.getStartDate();
}

public Date getEndDate() {
    return period.getEndDate();
}

public BudgetPeriod previousBudgetPeriod() {
    return new BudgetPeriod (type, type.getBudgetPeriodOffset(period.getStartDate(), -1));
}
```

And removed nextBudgetPeriod which was added to BudgetCategoryType while removing magic numbers.
Functionality has moved in BudgetCategory

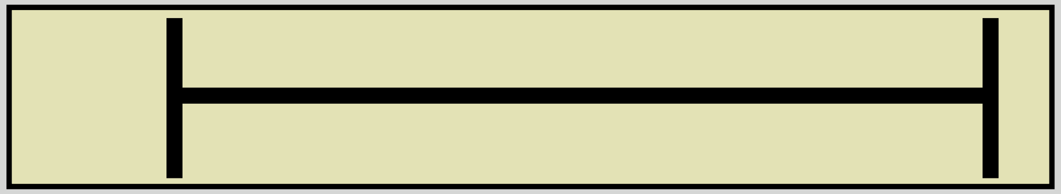
Dead code

Code that is not used or not useful

Periods and BudgetPeriods

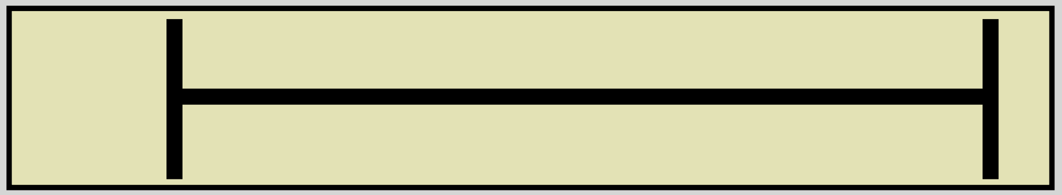
Periods and BudgetPeriods

First if-statement

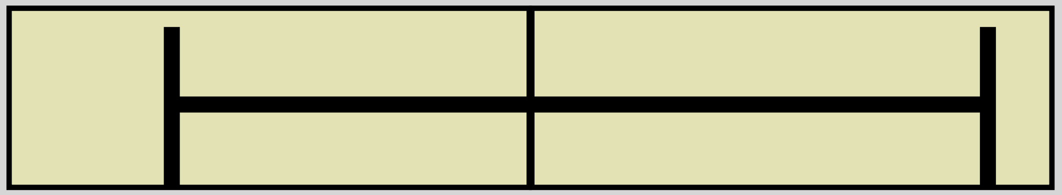


Periods and BudgetPeriods

First if-statement

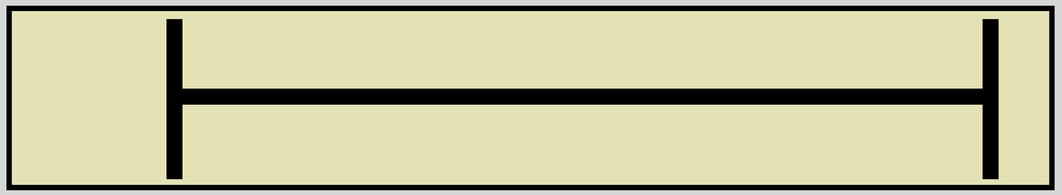


Second if-statement

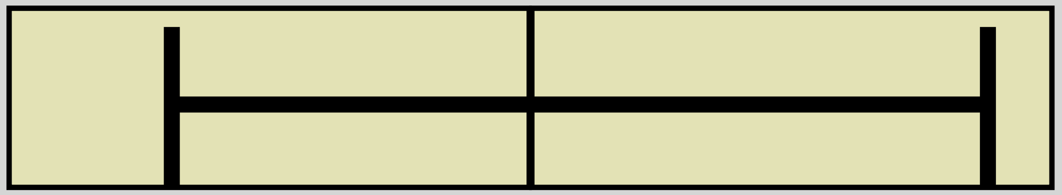


Periods and BudgetPeriods

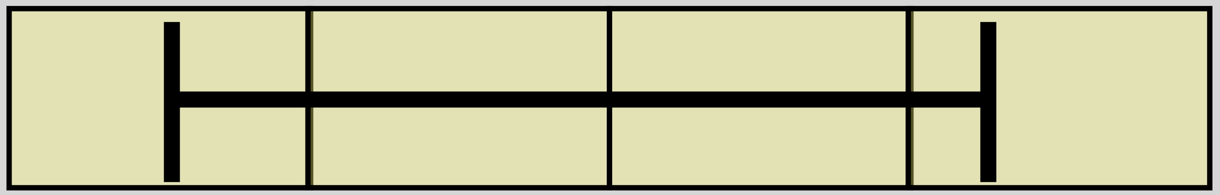
First if-statement



Second if-statement

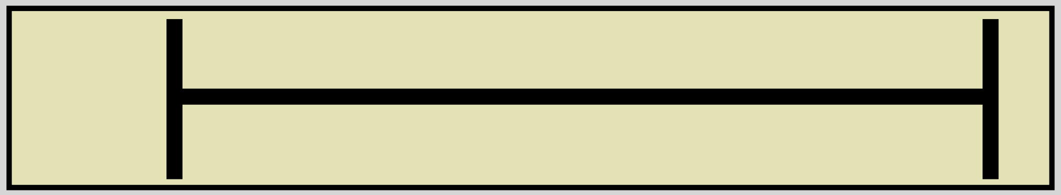


Second if-statement

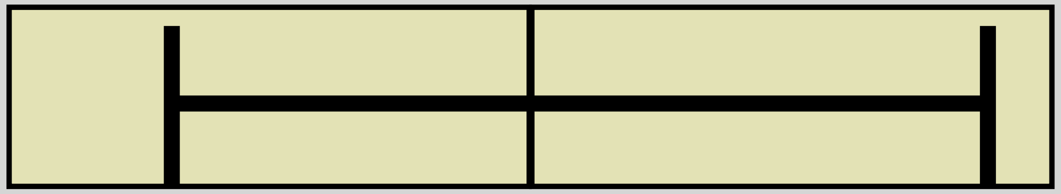


Periods and BudgetPeriods

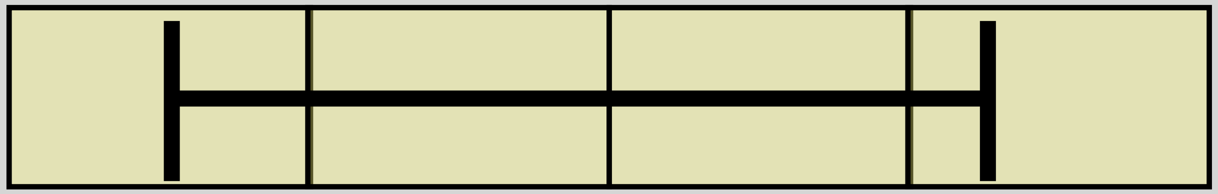
First if-statement



Second if-statement



Second if-statement



No other options!

```
private long getTotalAmountPeriod(Period period) {

    BudgetPeriod firstBudgetPeriod = createBudgetPeriodFromDate(period.getStartDate());
    BudgetPeriod lastBudgetPeriod = createBudgetPeriodFromDate(period.getEndDate());

    if (firstBudgetPeriod.equals(lastBudgetPeriod))
        return (long) getAmountForPeriodWithinBudgetPeriodOfStartDate(period.getStartDate(), period.getEndDate());

    //If the area between Start and End overlap at least two budget periods.
    if (firstBudgetPeriod.nextBudgetPeriod().getStartDate().equals(lastBudgetPeriod.getStartDate())
            || firstBudgetPeriod.nextBudgetPeriod().getStartDate().before(lastBudgetPeriod.getStartDate())){
        double totalStartPeriod = getAmountForPeriodWithinBudgetPeriodOfStartDate(period.getStartDate(),
                firstBudgetPeriod.getEndDate());

        double totalInMiddle = 0;
        for (String periodKey : getBudgetPeriods(
                firstBudgetPeriod.nextBudgetPeriod().getStartDate(),
                lastBudgetPeriod.previousBudgetPeriod().getStartDate())) {
            totalInMiddle += getAmountFromBudgetPeriodContainingDate(getPeriodDate(periodKey));
        }

        double totalEndPeriod = getAmountForPeriodWithinBudgetPeriodOfStartDate(lastBudgetPeriod.getStartDate(),
eriod.getEndDate());
        return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
    }

    throw new RuntimeException("You should not be here.  We have returned all legitimate numbers from
etAmount(Date, Date) in BudgetCategoryImpl.  Please contact Wyatt Olson with details on how you got here (what steps
id you perform in Buddi to get this error message).");
}
```

TRUE! It cannot possibly come here :)

Removed dead code

```
    private long getTotalAmountPeriod(Period period) {

        BudgetPeriod firstBudgetPeriod = createBudgetPeriodFromDate(period.getStartDate());
        BudgetPeriod lastBudgetPeriod = createBudgetPeriodFromDate(period.getEndDate());

        if (firstBudgetPeriod.equals(lastBudgetPeriod))
            return (long) getAmountForPeriodWithinBudgetPeriodOfStartDate(period.getStartDate(), period.getEndDate());

        double totalStartPeriod = getAmountForPeriodWithinBudgetPeriodOfStartDate(period.getStartDate(),
                firstBudgetPeriod.getEndDate());

        double totalInMiddle = 0;
        for (String periodKey : getBudgetPeriods(
                firstBudgetPeriod.nextBudgetPeriod().getStartDate(),
                lastBudgetPeriod.previousBudgetPeriod().getStartDate())) {
            totalInMiddle += getAmountFromBudgetPeriodContainingDate(getPeriodDate(periodKey));
        }

        double totalEndPeriod = getAmountForPeriodWithinBudgetPeriodOfStartDate(lastBudgetPeriod.getStartDate(),
period.getEndDate());
        return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
    }
```

Data Clump and Odd Name

```
private long getTotalAmountPeriod(Period period) {

    BudgetPeriod firstBudgetPeriod = createBudgetPeriodFromDate(period.getStartDate());
    BudgetPeriod lastBudgetPeriod = createBudgetPeriodFromDate(period.getEndDate());

    if (firstBudgetPeriod.equals(lastBudgetPeriod))
        return (long) getAmountForPeriodWithinBudgetPeriod(period, firstBudgetPeriod);

    double totalStartPeriod = getAmountForPeriodWithinBudgetPeriod(new Period (period.getStartDate(),
            firstBudgetPeriod.getEndDate()), firstBudgetPeriod);

    double totalInMiddle = 0;
    for (String periodKey : getBudgetPeriods(
            firstBudgetPeriod.nextBudgetPeriod().getStartDate(),
            lastBudgetPeriod.previousBudgetPeriod().getStartDate())) {
        totalInMiddle += getAmountFromBudgetPeriodContainingDate(getPeriodDate(periodKey));
    }

    double totalEndPeriod = getAmountForPeriodWithinBudgetPeriod(new Period(lastBudgetPeriod.getStartDate(),
            period.getEndDate()), lastBudgetPeriod);
    return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
}
```

Yes, still ugly...

Feature Envy

More interested in another class than its own

Requesting too much information from Periods

```java
private double getAmountForPeriodWithinBudgetPeriod(Period period, BudgetPeriod firstBudgetPeriod) {
    long amount = getAmountFromBudgetPeriodContainingDate(firstBudgetPeriod.getStartDate());
    long daysInPeriod = getBudgetPeriodType().getDaysInPeriod(firstBudgetPeriod.getStartDate());
    long daysBetween = DateUtil.getDaysBetween(period.getStartDate(), period.getEndDate(), true);
    double amountInPeriod = ((double) amount / (double) daysInPeriod) * daysBetween;
    return amountInPeriod;
}
```

Move functionality to Period and BudgetPeriod

```java
private double getAmountForPeriodWithinBudgetPeriod(Period period, BudgetPeriod firstBudgetPeriod) {
    long amount = getAmountFromBudgetPeriodContainingDate(firstBudgetPeriod.getStartDate());
    long daysInPeriod = firstBudgetPeriod.getAmountOfDays();
    long daysBetween = period.getAmountOfDays();
    double amountInPeriod = ((double) amount / (double) daysInPeriod) * daysBetween;
    return amountInPeriod;
}
```

Additional Cleanup

```
private double getAmountForPeriodWithinBudgetPeriod(Period period, BudgetPeriod firstBudgetPeriod) {
    long amount = getAmountFromBudgetPeriodContainingDate(firstBudgetPeriod.getStartDate());
    long daysInPeriod = firstBudgetPeriod.getAmountOfDays();
    long daysBetween = period.getAmountOfDays();
    double amountInPeriod = ((double) amount / (double) daysInPeriod) * daysBetween;
    return amountInPeriod;
}
```

Inline Temp - Alt-Shift-I

```
private double getAmountForPeriodWithinBudgetPeriod(Period period, BudgetPeriod firstBudgetPeriod) {
    long amount = getAmountFromBudgetPeriod(firstBudgetPeriod);
    long daysInPeriod = firstBudgetPeriod.getAmountOfDays();
    long daysBetween = period.getAmountOfDays();
    return ((double) amount / (double) daysInPeriod) * daysBetween;
}
```

Primitive Obsession

Use of primitives in higher-level abstraction methods

```
    double totalStartPeriod = getAmountForPeriodWithinBudgetPeriod(new Period (period.getStartDate(),
            firstBudgetPeriod.getEndDate()), firstBudgetPeriod);

    double totalInMiddle = 0;
    for (String periodKey : getBudgetPeriods(
            firstBudgetPeriod.nextBudgetPeriod().getStartDate(),
            lastBudgetPeriod.previousBudgetPeriod().getStartDate())) {
        totalInMiddle += getAmountFromBudgetPeriodContainingDate(getPeriodDate(periodKey));
    }

    double totalEndPeriod = getAmountForPeriodWithinBudgetPeriod(new Period(lastBudgetPeriod.getStartDate(),
            period.getEndDate()), lastBudgetPeriod);
    return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
}
```

```
double totalInMiddle = 0;
for (BudgetPeriod budgetPeriod :
        getBudgetPeriods(firstBudgetPeriod.nextBudgetPeriod(), lastBudgetPeriod.previousBudgetPeriod())) {
    totalInMiddle += getAmountFromBudgetPeriod(budgetPeriod);
}
```

And BudgetPeriods

```
public List<String> getBudgetPeriods(Date startDate, Date endDate){
    List<String> budgetPeriodKeys = new LinkedList<String>();

    Date temp = getBudgetPeriodType().getStartOfBudgetPeriod(startDate);

    while (temp.before(getBudgetPeriodType().getEndOfBudgetPeriod(endDate))){
        budgetPeriodKeys.add(getPeriodKey(temp));
        temp = getBudgetPeriodType().getBudgetPeriodOffset(temp, 1);
    }

    return budgetPeriodKeys;
}
```

```
private List<BudgetPeriod> getBudgetPeriods(BudgetPeriod firstBudgetPeriod, BudgetPeriod lastBudgetPeriod) {
    List<BudgetPeriod> budgetPeriodKeys = new LinkedList<BudgetPeriod>();

    BudgetPeriod currentBudgetPeriod = firstBudgetPeriod;

    while (currentBudgetPeriod.getStartDate().before(lastBudgetPeriod.getEndDate())){
        budgetPeriodKeys.add(currentBudgetPeriod);
        currentBudgetPeriod = currentBudgetPeriod.nextBudgetPeriod();
    }

    return budgetPeriodKeys;
}
```

Odd-e

And move it to BudgetPeriod (feature envy)

```
public List<BudgetPeriod> createBudgetPeriodListTill(BudgetPeriod lastBudgetPeriod) {
    List<BudgetPeriod> budgetPeriodKeys = new LinkedList<BudgetPeriod>();

    BudgetPeriod currentBudgetPeriod = this;

    while (currentBudgetPeriod.getStartDate().before(lastBudgetPeriod.getEndDate())){
        budgetPeriodKeys.add(currentBudgetPeriod);
        currentBudgetPeriod = currentBudgetPeriod.nextBudgetPeriod();
    }

    return budgetPeriodKeys;
}
```

And the call

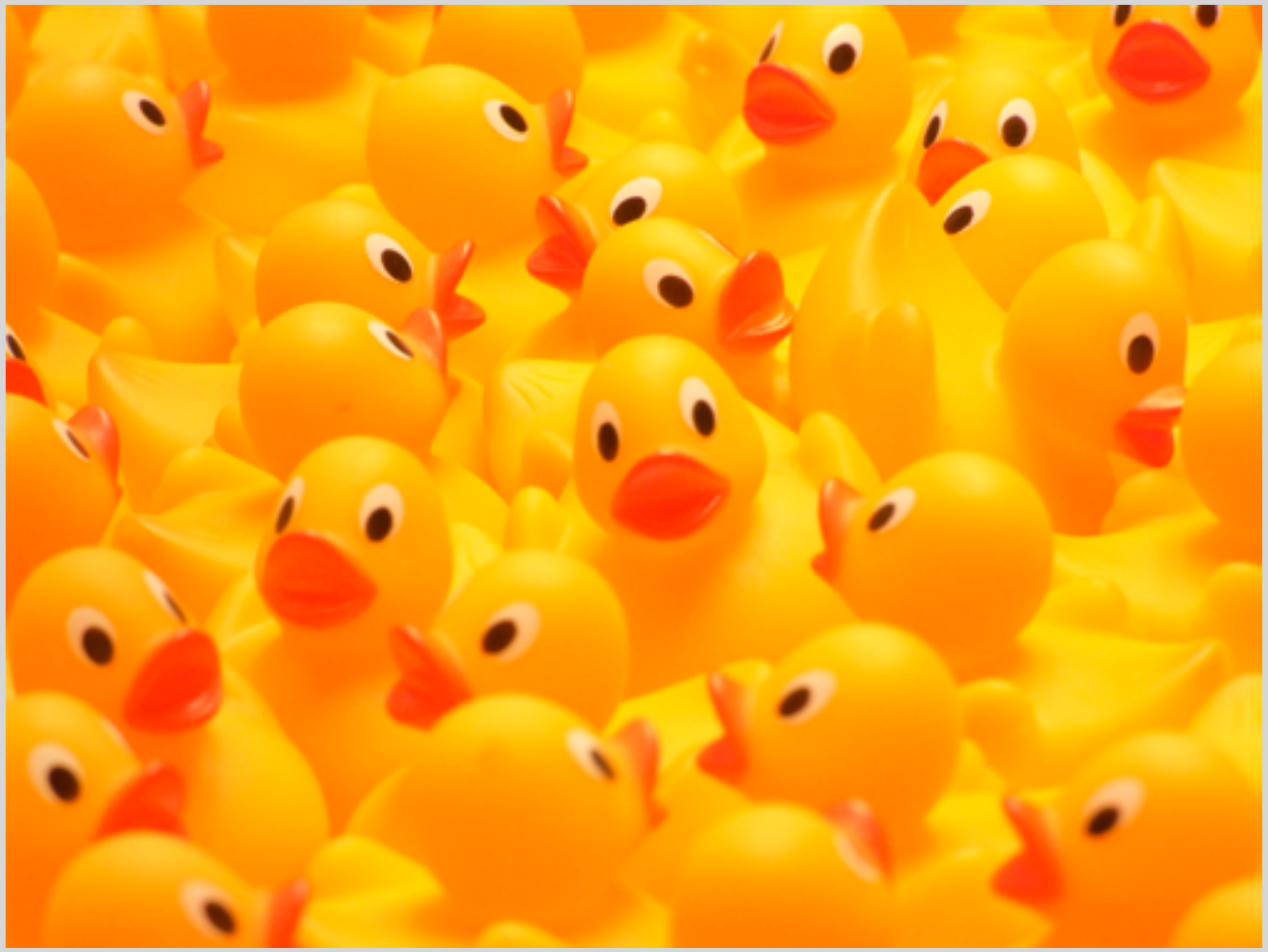
```
double totalInMiddle = 0;
for (BudgetPeriod budgetPeriod :
        firstBudgetPeriod.nextBudgetPeriod().createBudgetPeriodListTill(lastBudgetPeriod.previousBudgetPeriod()))

    totalInMiddle += getAmountFromBudgetPeriod(budgetPeriod);
}
```

Odd-e

Domain objects

Primitive Obsession is one of the most common and important smells.

It often suggests missing domain objects!



Abstraction Distraction

Use of different abstraction levels in the same code

```java
private long getTotalAmountPeriod(Period period) {

    BudgetPeriod firstBudgetPeriod = createBudgetPeriodFromDate(period.getStartDate());
    BudgetPeriod lastBudgetPeriod = createBudgetPeriodFromDate(period.getEndDate());

    if (firstBudgetPeriod.equals(lastBudgetPeriod))
        return (long) getAmountForPeriodWithinBudgetPeriod(period, firstBudgetPeriod);

    double totalStartPeriod = getAmountForPeriodWithinBudgetPeriod(new Period (period.getStartDate(),
            firstBudgetPeriod.getEndDate()), firstBudgetPeriod);

    double totalInMiddle = 0;
    for (BudgetPeriod budgetPeriod :
            firstBudgetPeriod.nextBudgetPeriod().createBudgetPeriodListTill(lastBudgetPeriod.previousBudgetPeriod())) {
        totalInMiddle += getAmountFromBudgetPeriod(budgetPeriod);
    }

    double totalEndPeriod = getAmountForPeriodWithinBudgetPeriod(new Period(lastBudgetPeriod.getStartDate(),
            period.getEndDate()), lastBudgetPeriod);
    return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
}
```

```java
BudgetPeriod firstBudgetPeriod = createFirstBudgetPeriod(period);
BudgetPeriod lastBudgetPeriod = createLastBudgetPeriod(period);
```

```
double totalStartPeriod = getAmountForPeriodWithinBudgetPeriod(new Period (period.getStartDate(),
        firstBudgetPeriod.getEndDate()), firstBudgetPeriod);
```

↓

```
double totalStartPeriod = getAmountForOverlappingDays(period, firstBudgetPeriod);
```

And new getAmount**For**OverlappingDays

```
private double getAmountForOverlappingDays(Period period, BudgetPeriod firstBudgetPeriod) {
    long amount = getAmountFromBudgetPeriod(firstBudgetPeriod);
    long daysInPeriod = firstBudgetPeriod.getAmountOfDays();
    long daysBetween = period.getAmountOfOverlappingDays(firstBudgetPeriod.getPeriod());
    return ((double) amount / (double) daysInPeriod) * daysBetween;
}
```

And new getAmount**Of**OverlappingDays in Period

```
public long getAmountOfOverlappingDays(Period period) {
    Date largestStartDate = (start.after(period.start)) ? start : period.start;
    Date smallestEndDate = (end.before(period.end)) ? end : period.end;

    if (smallestEndDate.before(largestStartDate))
        return 0;
    return new Period(largestStartDate, smallestEndDate).getAmountOfDays();
}
```

Use more generic version

```
if (firstBudgetPeriod.equals(lastBudgetPeriod))
    return (long) getAmountForPeriodWithinBudgetPeriod(period, firstBudgetPeriod);

double totalStartPeriod = getAmountForOverlappingDays(period, firstBudgetPeriod);

double totalInMiddle = 0;
for (BudgetPeriod budgetPeriod :
        firstBudgetPeriod.nextBudgetPeriod().createBudgetPeriodListTill(lastBudgetPeriod.previousBudgetPeriod()))

    totalInMiddle += getAmountFromBudgetPeriod(budgetPeriod);
}

double totalEndPeriod = getAmountForPeriodWithinBudgetPeriod(new Period(lastBudgetPeriod.getStartDate(),
        period.getEndDate()), lastBudgetPeriod);
```

And delete getAmountForPeriodWithinBudgetPeriod

```
if (firstBudgetPeriod.equals(lastBudgetPeriod))
    return (long) getAmountForOverlappingDays(period, firstBudgetPeriod);

double totalStartPeriod = getAmountForOverlappingDays(period, firstBudgetPeriod);

double totalInMiddle = 0;
for (BudgetPeriod budgetPeriod :
        firstBudgetPeriod.nextBudgetPeriod().createBudgetPeriodListTill(lastBudgetPeriod.previousBudgetPeriod())) {
    totalInMiddle += getAmountForOverlappingDays(period, budgetPeriod);
}

double totalEndPeriod = getAmountForOverlappingDays(period, lastBudgetPeriod);
```

Hmm...

```java
private long getTotalAmountPeriod(Period period) {

    BudgetPeriod firstBudgetPeriod = createFirstBudgetPeriod(period);
    BudgetPeriod lastBudgetPeriod = createLastBudgetPeriod(period);

    if (firstBudgetPeriod.equals(lastBudgetPeriod))
        return (long) getAmountForOverlappingDays(period, firstBudgetPeriod);

    double totalStartPeriod = getAmountForOverlappingDays(period, firstBudgetPeriod);

    double totalInMiddle = 0;
    for (BudgetPeriod budgetPeriod :
            firstBudgetPeriod.nextBudgetPeriod().createBudgetPeriodListTill(lastBudgetPeriod.previousBudgetPeriod())) {
        totalInMiddle += getAmountForOverlappingDays(period, budgetPeriod);
    }

    double totalEndPeriod = getAmountForOverlappingDays(period, lastBudgetPeriod);
    return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
}
```

Last refactoring made the first and last budgetPeriod the same, so we can delete the checks

Final version (?)

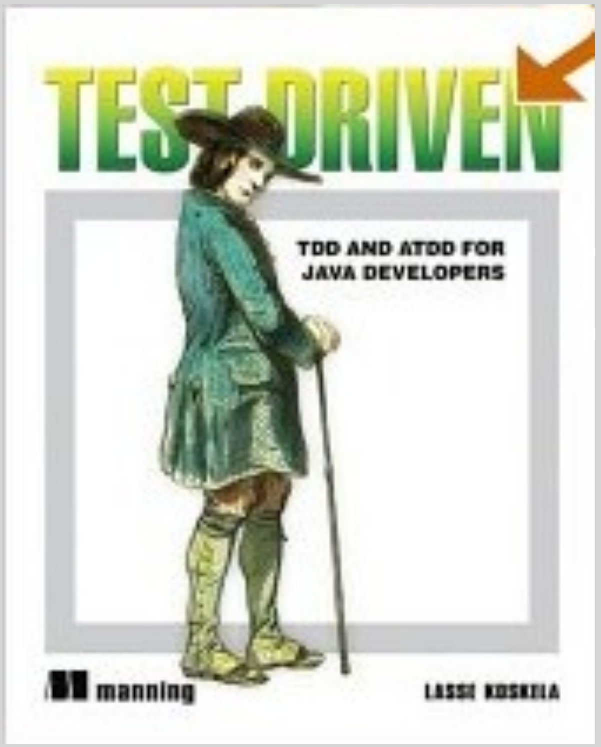
```
private long getTotalAmountPeriod(Period period) {

    BudgetPeriod firstBudgetPeriod = createFirstBudgetPeriod(period);
    BudgetPeriod lastBudgetPeriod = createLastBudgetPeriod(period);

    double total = 0;
    for (BudgetPeriod budgetPeriod :  firstBudgetPeriod.createBudgetPeriodListTill(lastBudgetPeriod)) {
        total += getAmountForOverlappingDays(period, budgetPeriod);
    }

    return (long) total;
}
```

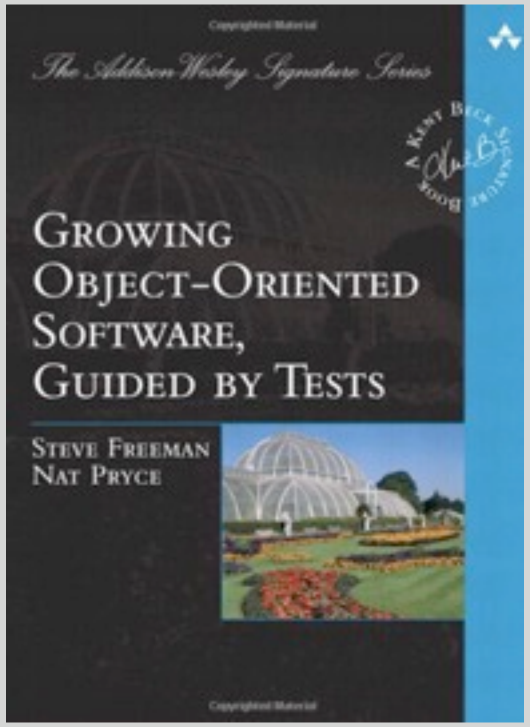
Probably not. Possible future directions:

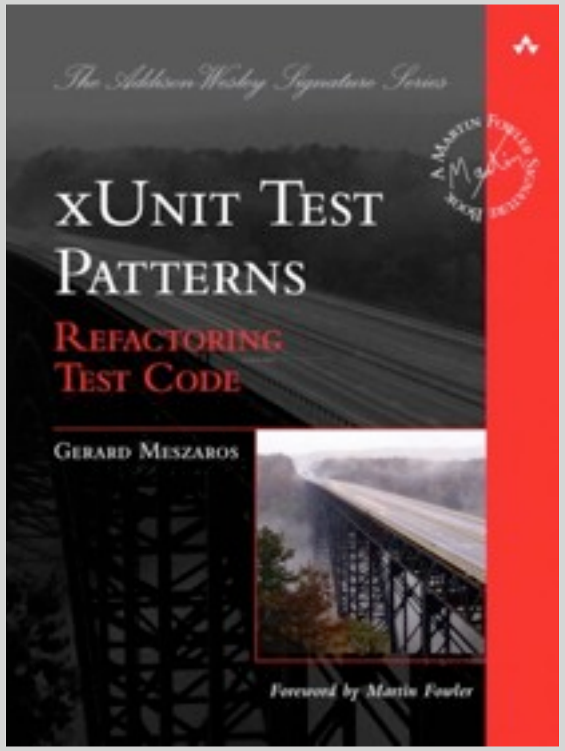
- Introduce Budget class (not exists!!)
- Introduce Money class (not exists!!!!!)
- Remove the BudgetCategoryType
- Much more primitive obsessions

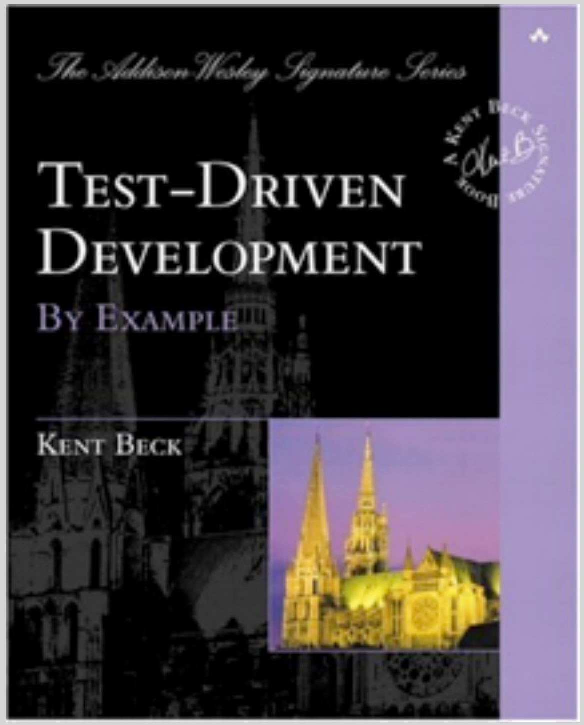
References

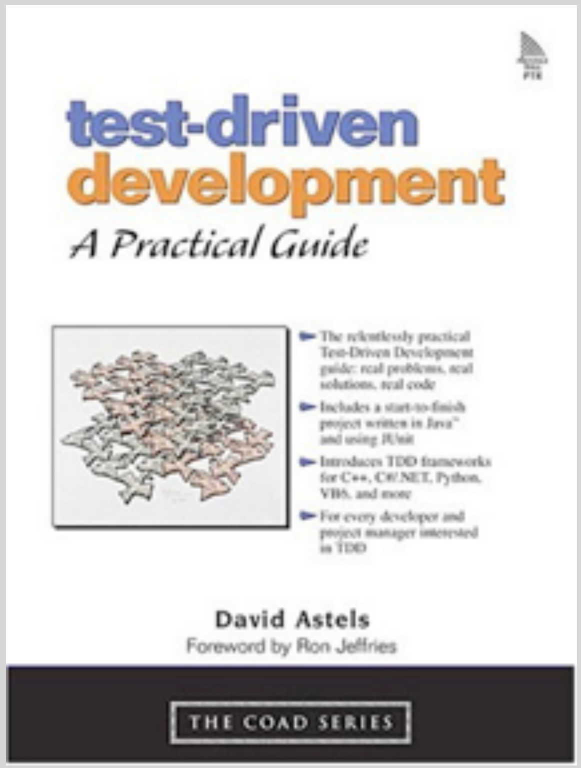
Test-Driven Development



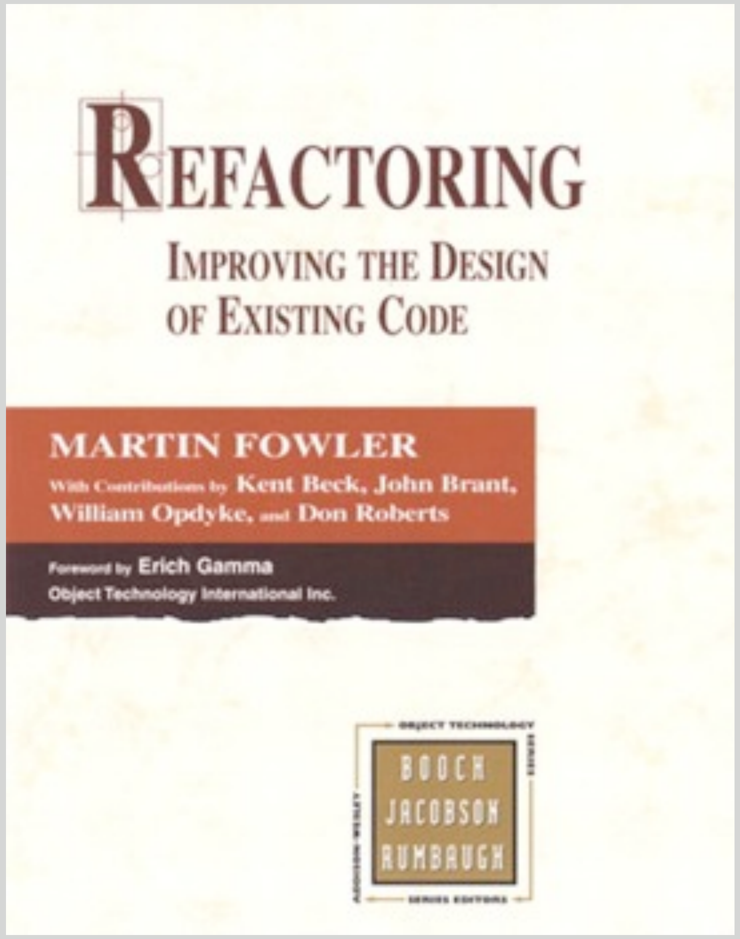

The Addison-Wesley Signature Series
Growing Object-Oriented Software, Guided by Tests
Steve Freeman
Nat Pryce
A Kent Beck Signature Book

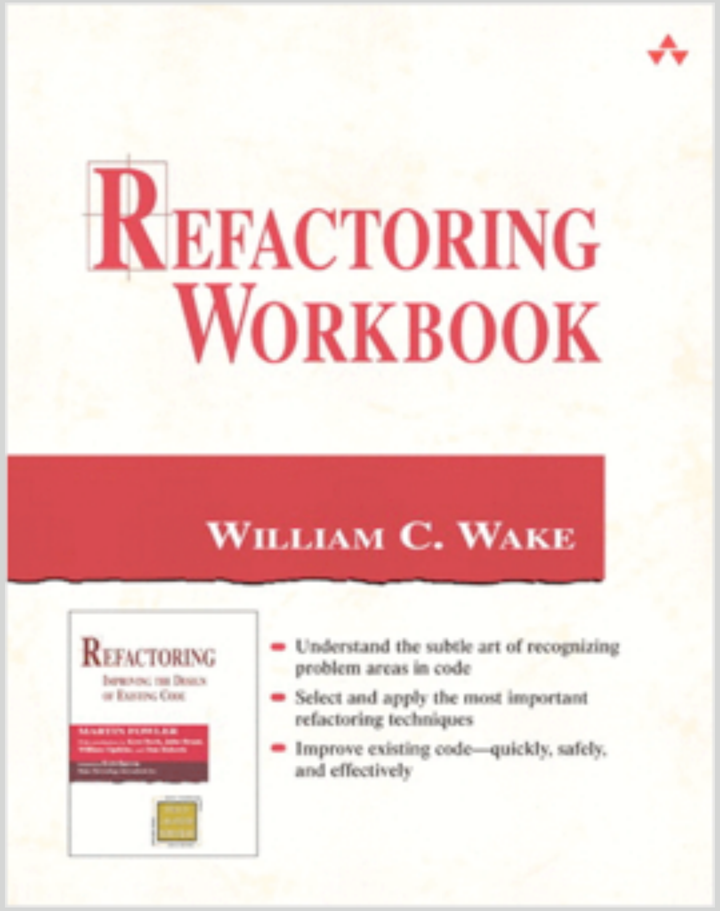

The Addison-Wesley Signature Series
xUnit Test Patterns
Refactoring Test Code
Gerard Meszaros
A Martin Fowler Signature Book
Foreword by Martin Fowler

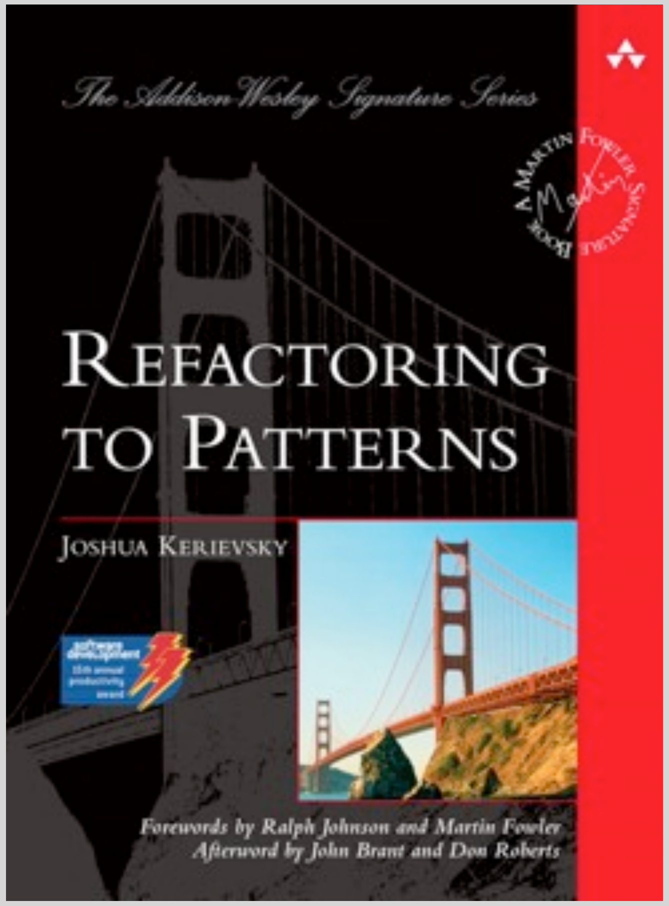

The Addison-Wesley Signature Series
Test-Driven Development
By Example
Kent Beck
A Kent Beck Signature Book



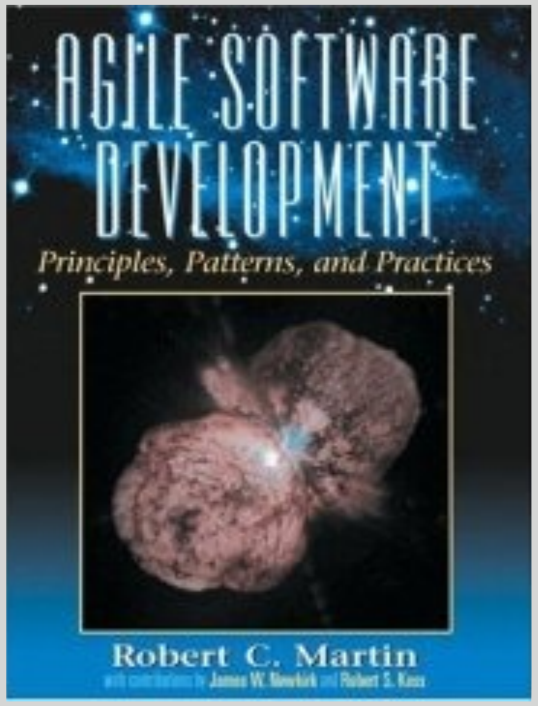
Refactoring

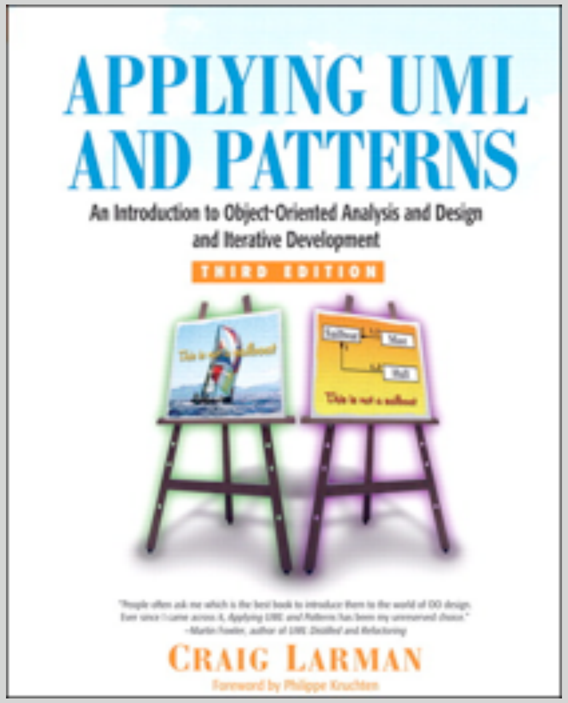

REFACTORING
IMPROVING THE DESIGN OF EXISTING CODE
MARTIN FOWLER
With Contributions by Kent Beck, John Brant, William Opdyke, and Don Roberts
Foreword by Erich Gamma
Object Technology International Inc.
BOOCH JACOBSON RUMBAUGH

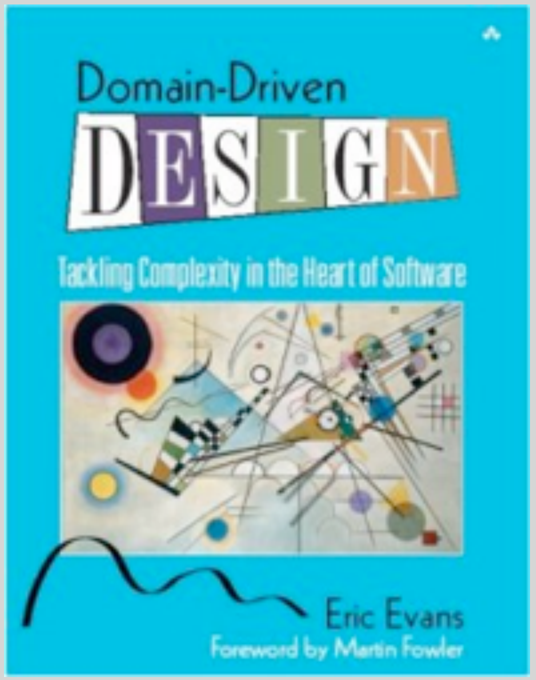

REFACTORING WORKBOOK
WILLIAM C. WAKE
Understand the subtle art of recognizing problem areas in code
Select and apply the most important refactoring techniques
Improve existing code—quickly, safely, and effectively

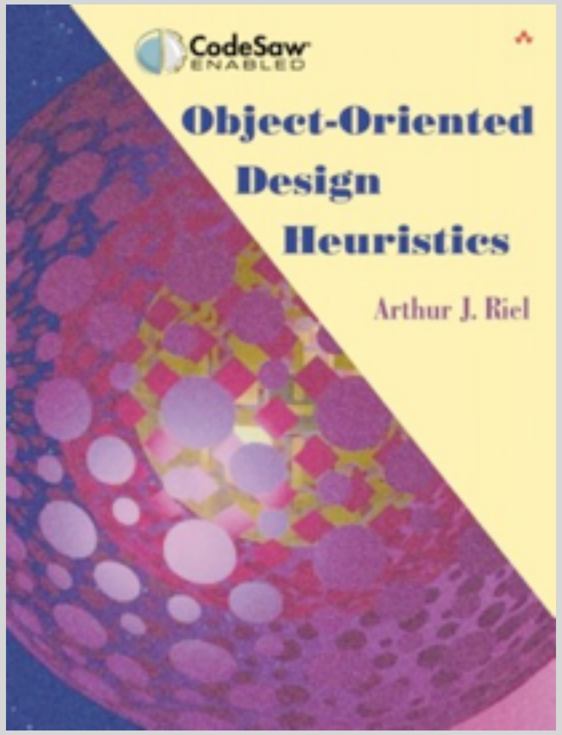

The Addison-Wesley Signature Series
A MARTIN FOWLER SIGNATURE BOOK
REFACTORING TO PATTERNS
JOSHUA KERIEVSKY
Forewords by Ralph Johnson and Martin Fowler
Afterword by John Brant and Don Roberts

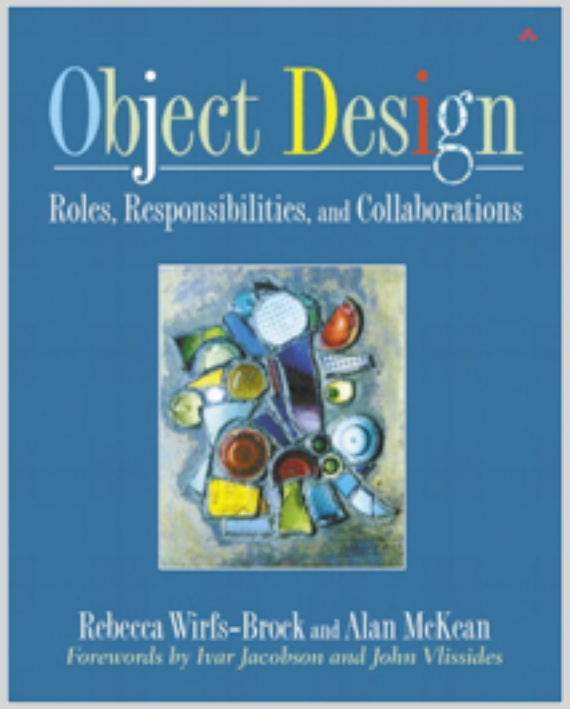
Object-Oriented Design


AGILE SOFTWARE DEVELOPMENT
Principles, Patterns, and Practices
Robert C. Martin

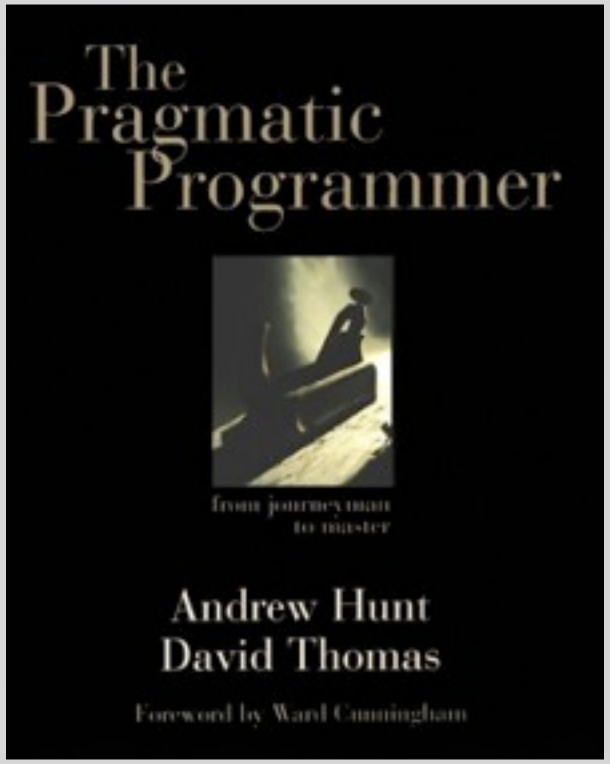

APPLYING UML AND PATTERNS
An Introduction to Object-Oriented Analysis and Design and Iterative Development
THIRD EDITION
CRAIG LARMAN

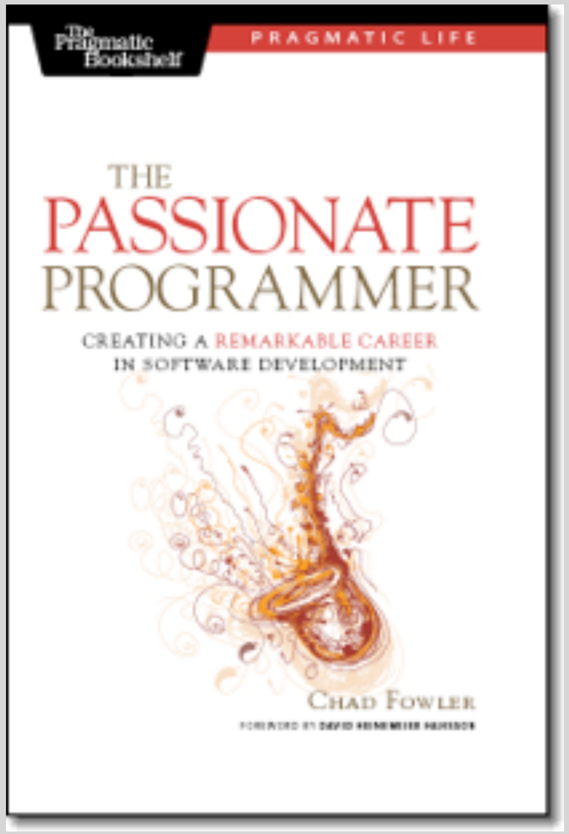

Domain-Driven DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler

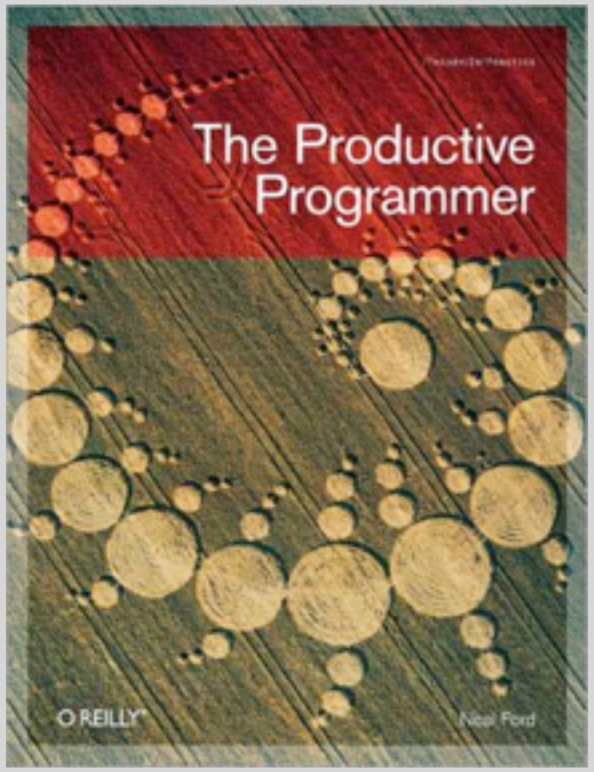


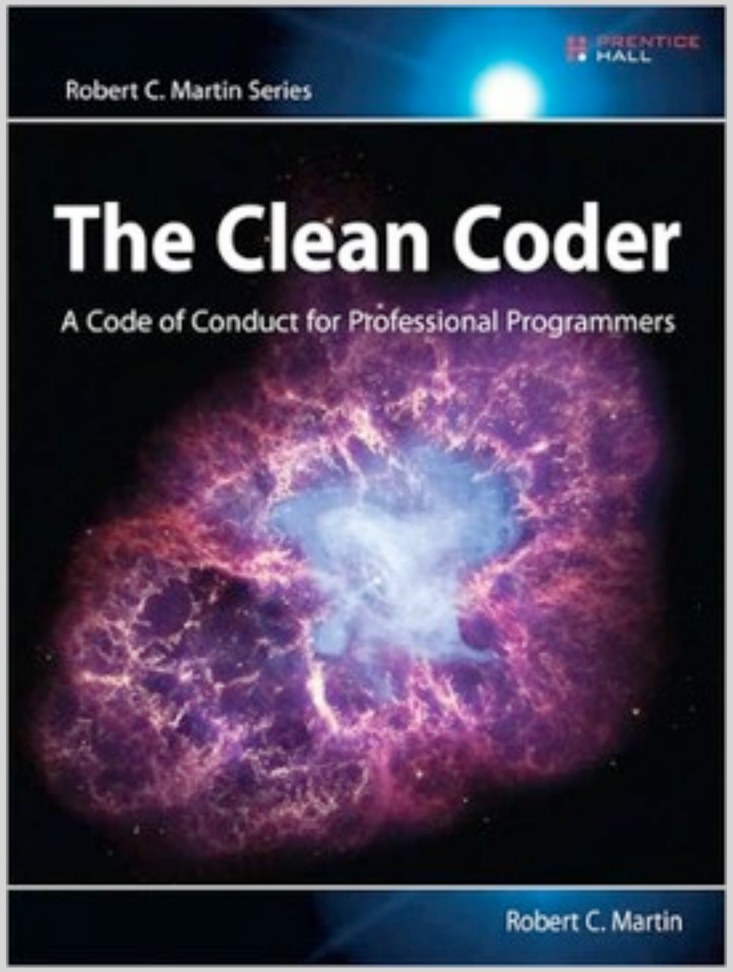

Object Design
Roles, Responsibilities, and Collaborations
Rebecca Wirfs-Brock and Alan McKean
Forewords by Ivar Jacobson and John Vlissides

Attitude!


The Pragmatic Programmer
from journeyman to master
Andrew Hunt
David Thomas
Foreword by Ward Cunningham


The Pragmatic Bookshelf
PRAGMATIC LIFE
THE PASSIONATE PROGRAMMER
CREATING A REMARKABLE CAREER IN SOFTWARE DEVELOPMENT
CHAD FOWLER


The Productive Programmer
O'REILLY
Neal Ford


Robert C. Martin Series
PRENTICE HALL
The Clean Coder
A Code of Conduct for Professional Programmers
Robert C. Martin

Software that got harmed during this presentation

- **Buddi** http://sourceforge.net/projects/buddi/

We mean no harm to the open source software movement, these open source projects, nor their authors. We are happy that they've contributed their source code to the software development community at large and that we have the opportunity to learn from others.

Odd-e Scrum Developer

Required to qualify for CSD:

- 3-day technical practices
- 1 additional day technical practice or CSM/CSPO
- 1-day Scrum Intro or CSM/CSPO

More info at:

http://www.scrumalliance.org//pages/certified_scrum_developer

Thank you

Steven Mak
steven@odd-e.com

Stanly Lau
stanly@odd-e.com