

# XP CONFERENCE INDIA 2016

Powered By Solutions*IQ*

19 - 20 August 2016  
Bangalore

[www.xpconference.in](http://www.xpconference.in)

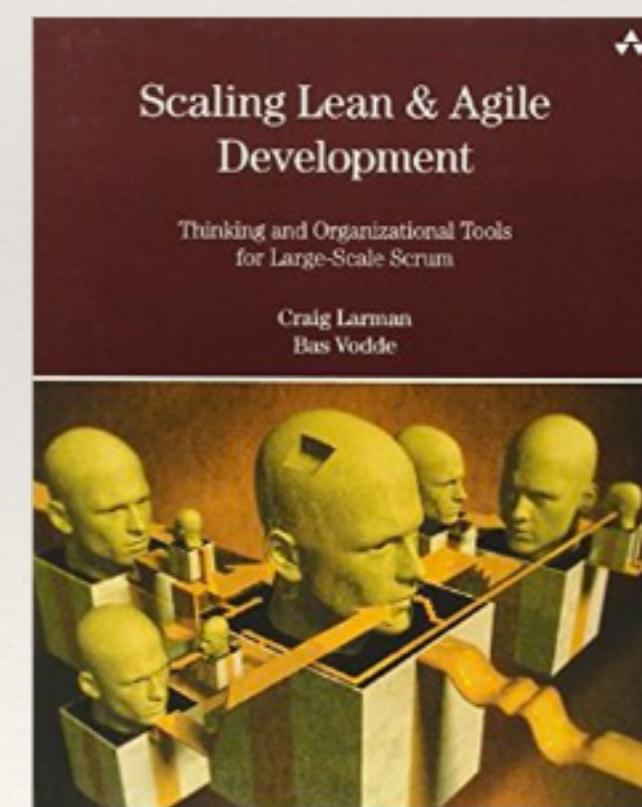
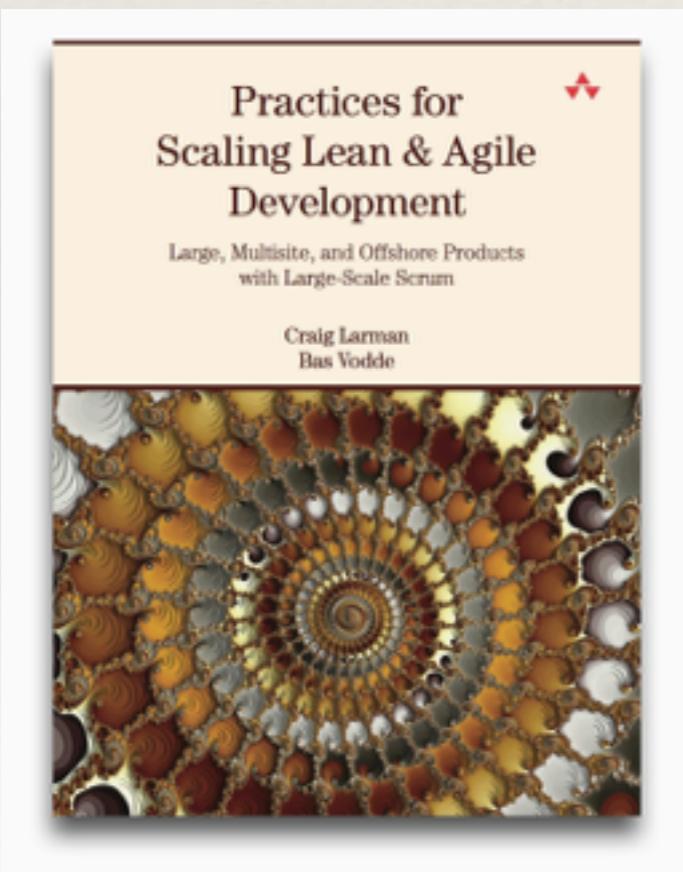


## REFACTORING FOR SOFTWARE DESIGN SMELLS

**Ganesh Samarthym**  
[ganesh@codeops.tech](mailto:ganesh@codeops.tech)

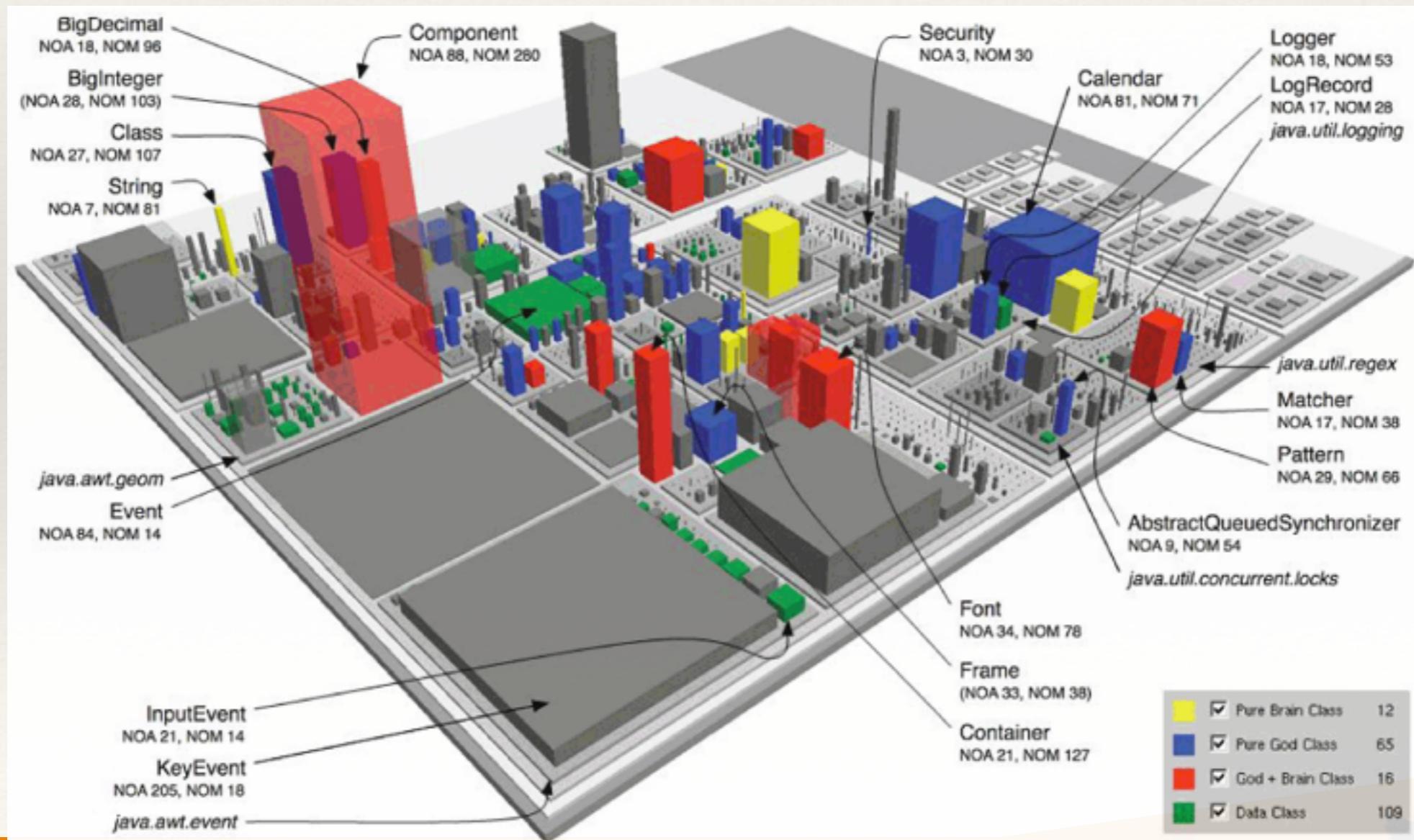
"The critical design tool for software development  
is a mind well educated in design principles"

*–Craig Larman*



# Warm-up question #1

What does this tool visualize?



# Warm-up question #2

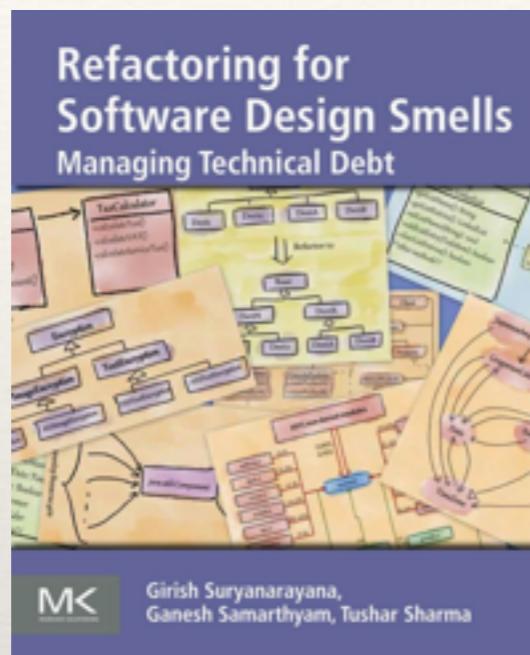


Who coined  
the term  
“code smell”?

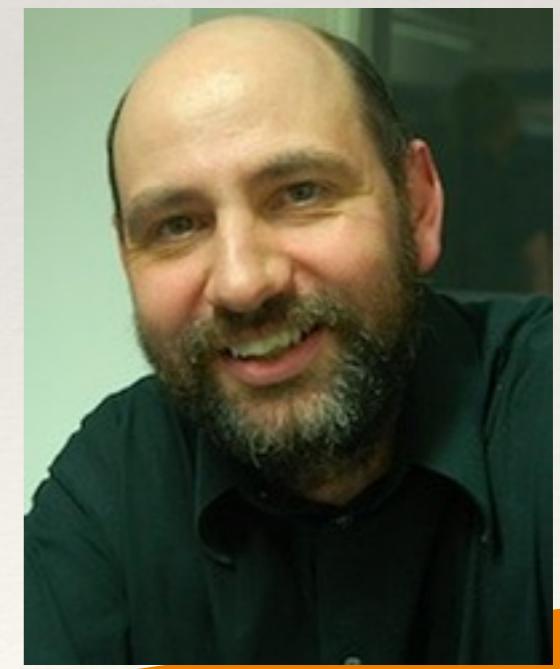
Hint: He also  
originated the  
terms TDD  
and XP



# Warm-up question #3

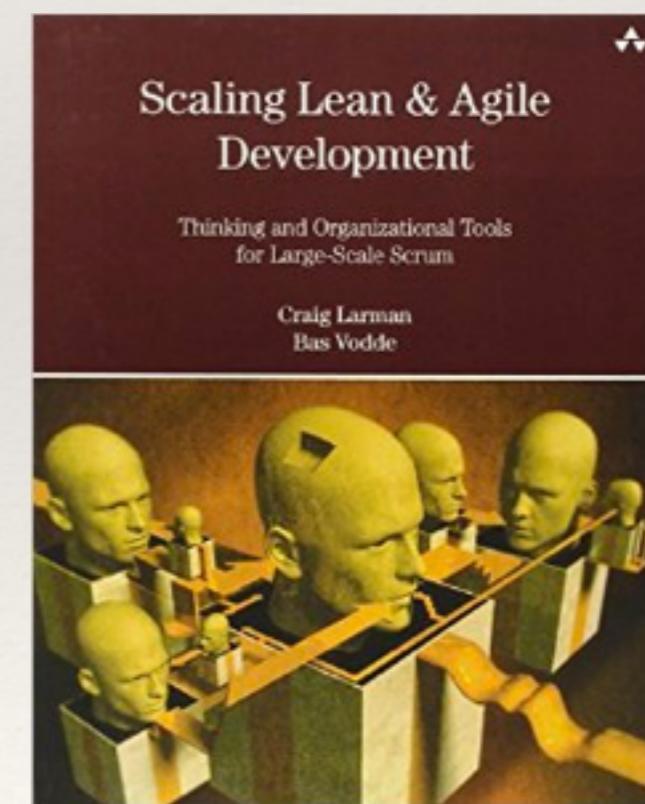
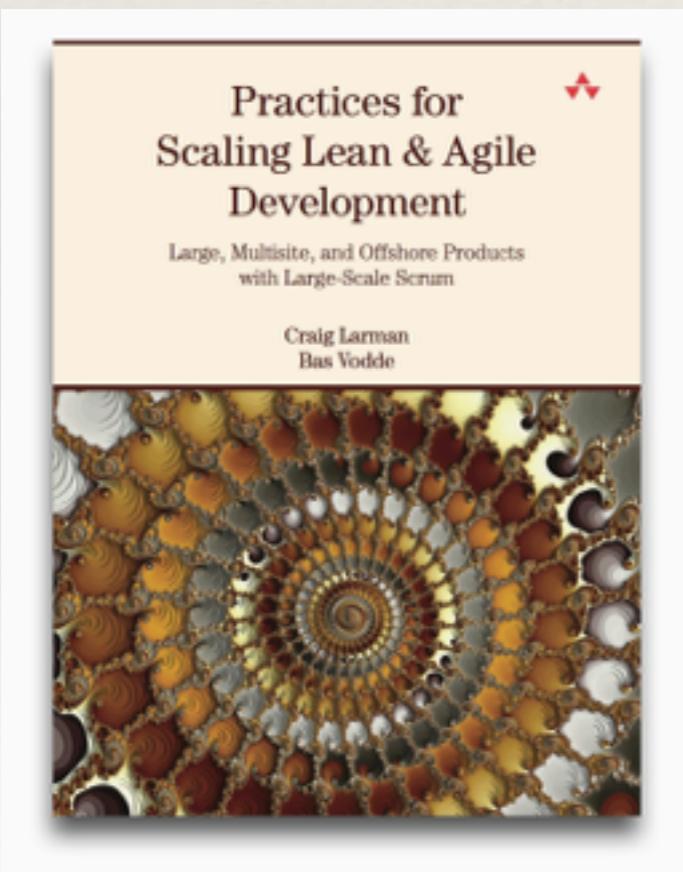


Who wrote the foreword for  
our book?



"The critical design tool for software development  
is a mind well educated in design principles"

*–Craig Larman*



# Why care about design quality and design principles?

<#>



#XPIndia2016

## Why care? Reason #1

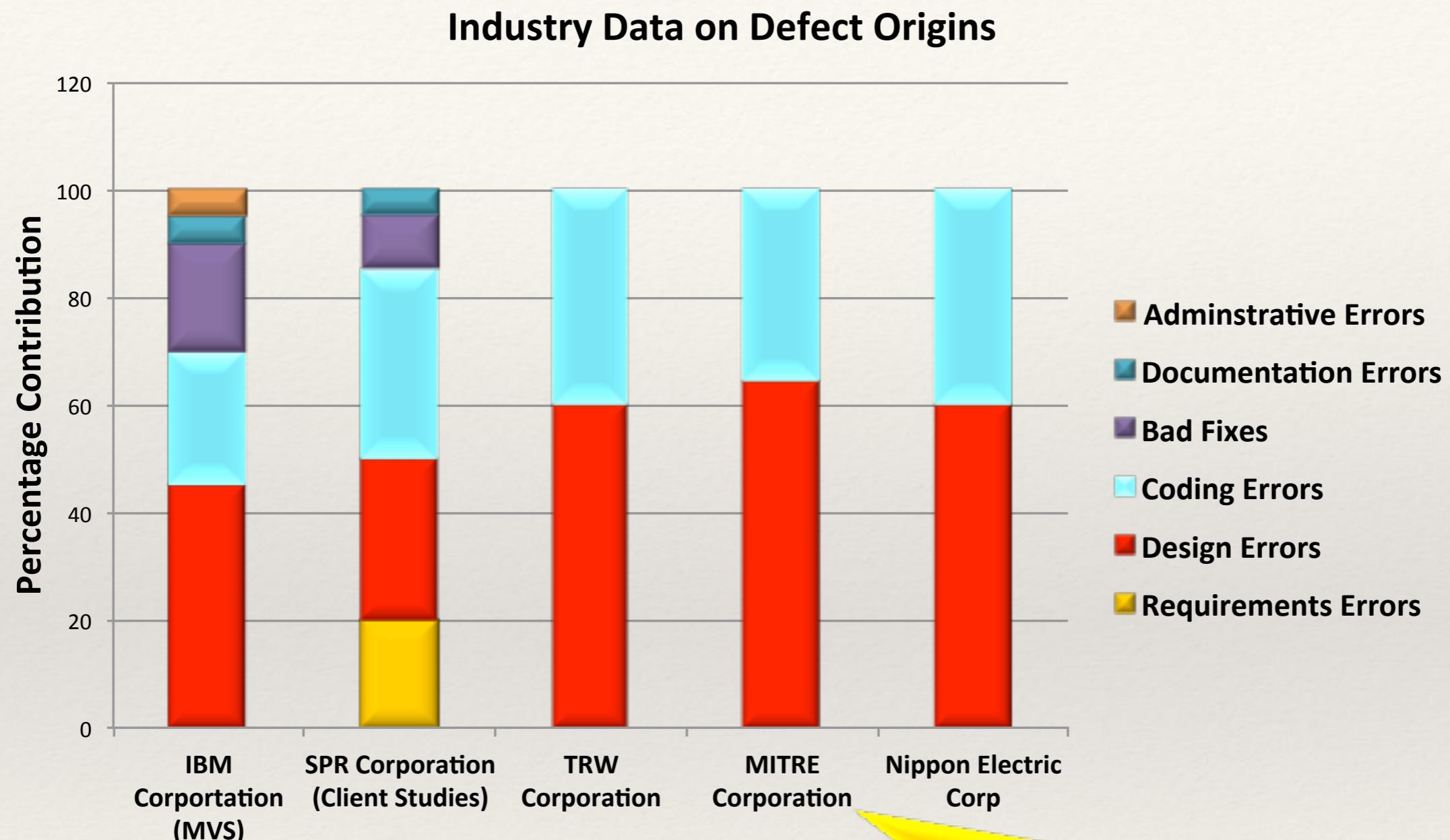
# Why care about design quality and design principles?



Poor software quality costs more than \$150 billion per year in U.S. and greater than \$500 billion per year worldwide

- Capers Jones

# Why care? Reason #1



Up to 64% of software defects can be traced back to errors in software design in enterprise software!

\* <http://sqgne.org/presentations/2012-13/Jones-Sep-2012.pdf>



Portion of total effort spent repairing architecturally complex defects, though they account for only 8% of all defects.

```
252     document.getElementById('bigImageDesc').innerHTML = descriptions[page * 5 + currentImageIndex];
253 }
254
255 function updatePhotoDescription() {
256     if (descriptions.length > (page * 5) + (currentImageIndex + 1)) {
257         document.getElementById('bigImageDesc').innerHTML = descriptions[page * 5 + currentImageIndex + 1];
258     }
259 }
260
261 function updateAllImages() {
262     var i = 1;
263     while (i < 10) {
264         var elementId = 'foto' + i;
265         var elementIdBig = 'bigImage' + i;
```

“The problem with quick and dirty...is that dirty remains long after quick has been forgotten”

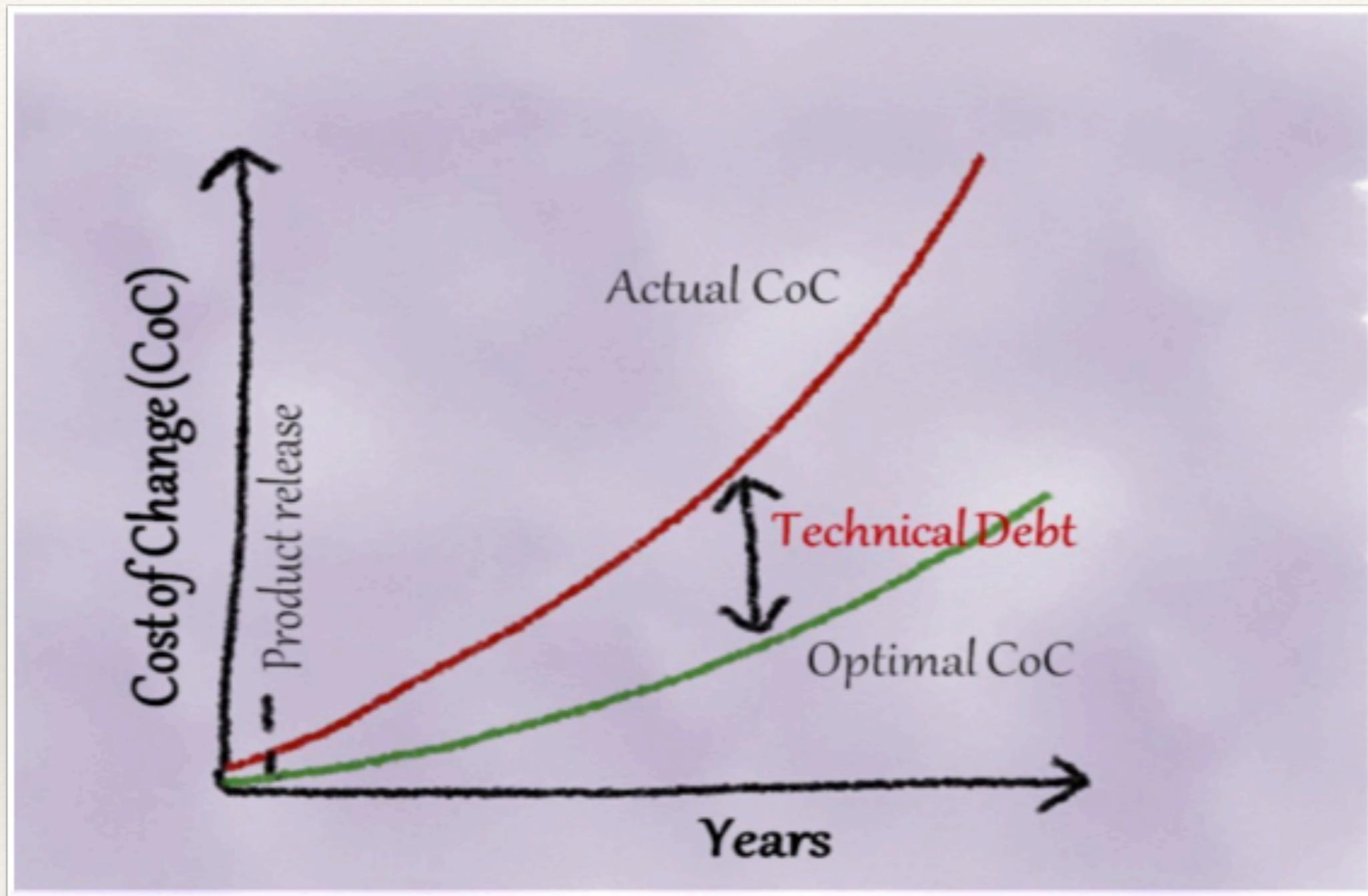
Steve C McConnell

# Why care? Reason #3



**“Technical debt is the debt that accrues when you knowingly or unknowingly make wrong or non-optimal design decisions”**

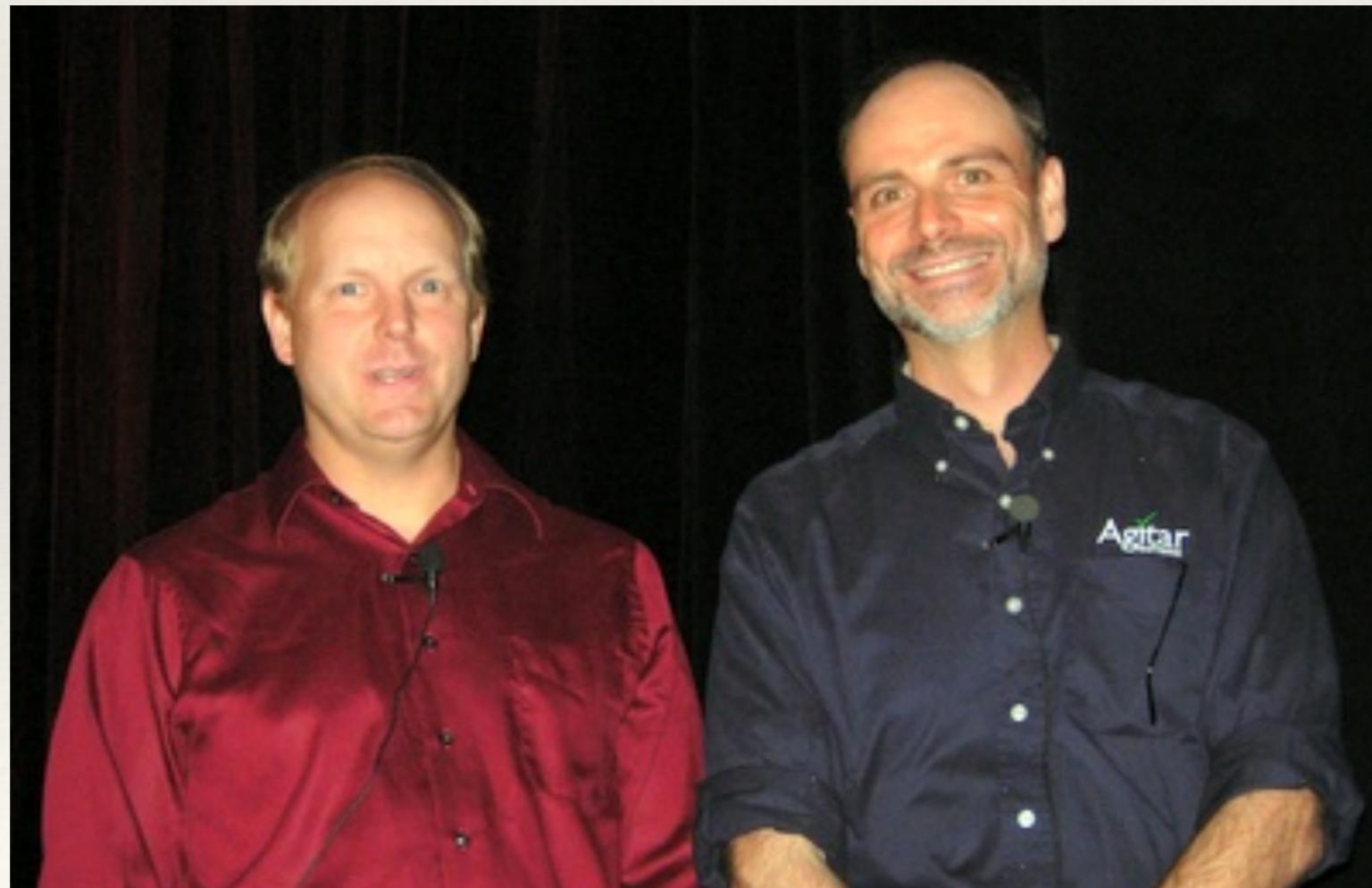
# Why care? Reason #3



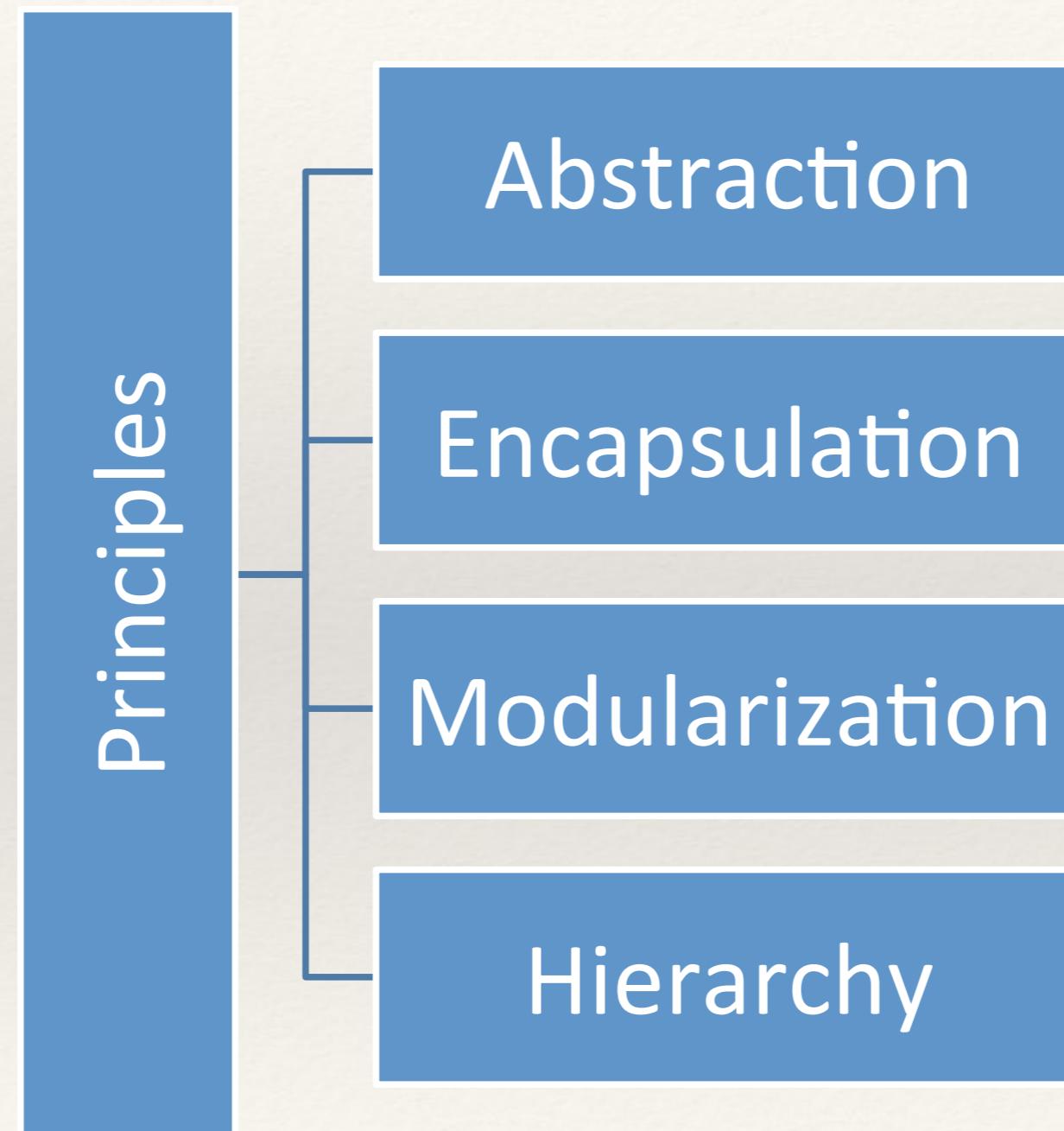
## Why care? Reason #2

"We thought we were just programming on an airplane"  
- Kent Beck

# JUnit



# Fundamental principles in software design



# Effective design: Java parallel streams

- ❖ Applies the principles of abstraction and encapsulation effectively to significantly simplify concurrent programming



# Parallel streams: example

```
import java.util.stream.LongStream;

class PrimeNumbers {
    private static boolean isPrime(long val) {
        for(long i = 2; i <= val/2; i++) {
            if((val % i) == 0) {
                return false;
            }
        }
        return true;
    }
    public static void main(String []args) {
        long num0fPrimes = LongStream.rangeClosed(2, 50_000)
            .parallel()
            .filter(PrimeNumbers::isPrime)
            .count();
        System.out.println(num0fPrimes);
    }
}
```



Prints 9592

```
long numOfPrimes = LongStream.rangeClosed(2, 100_000)
    .filter(PrimeNumbers::isPrime)
    .count();
System.out.println(numOfPrimes);
```

**2.510 seconds**



#XPIndia2016

Let's flip the switch by  
calling parallel() function



**Parallel code**

**Serial code**



#XPIndia2016

Prints 9592

```
long numOfPrimes = LongStream.rangeClosed(2, 100_000)
    .parallel()
    .filter(PrimeNumbers::isPrime)
    .count();

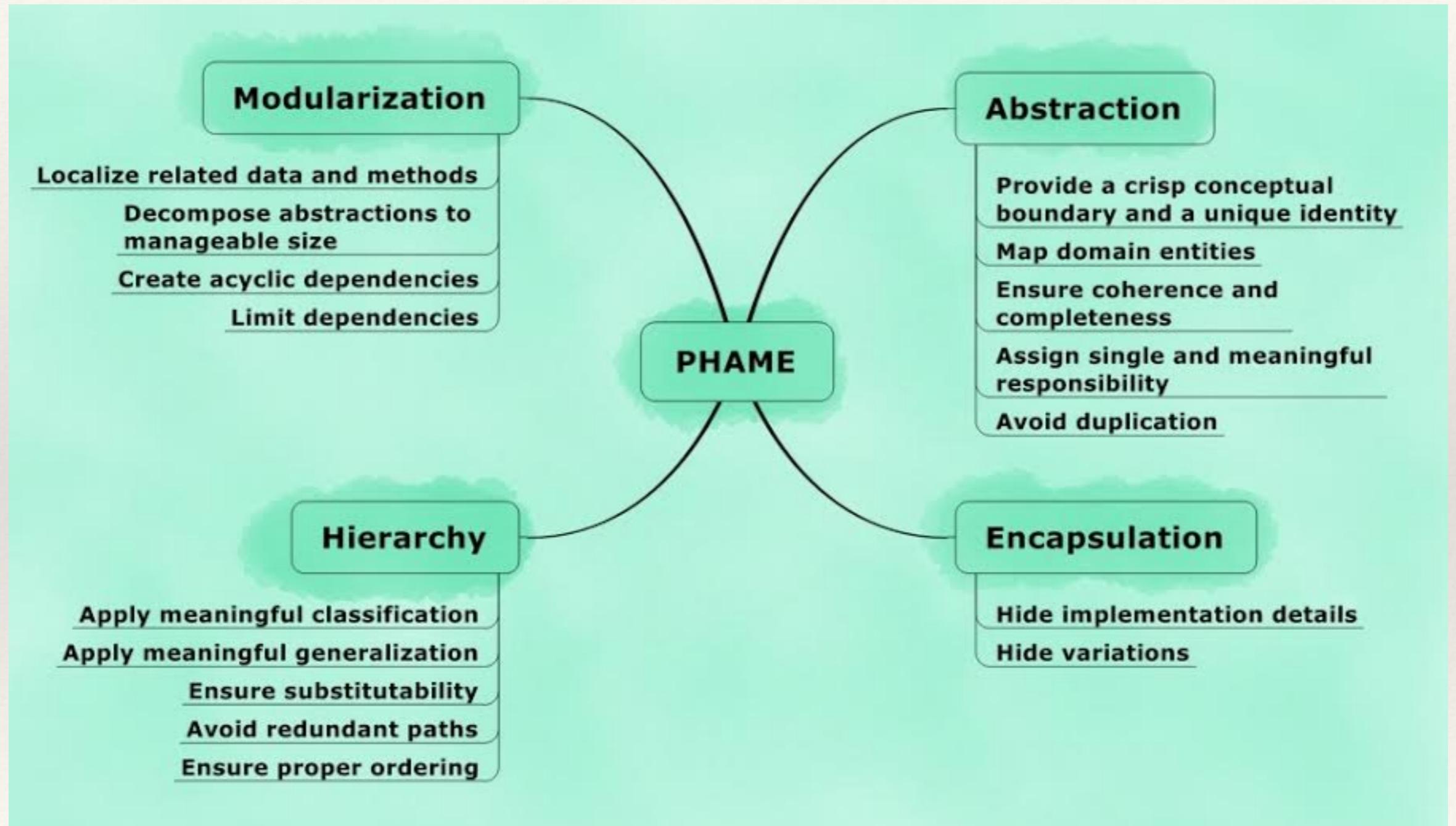
System.out.println(numOfPrimes);
```

**1.235 seconds**

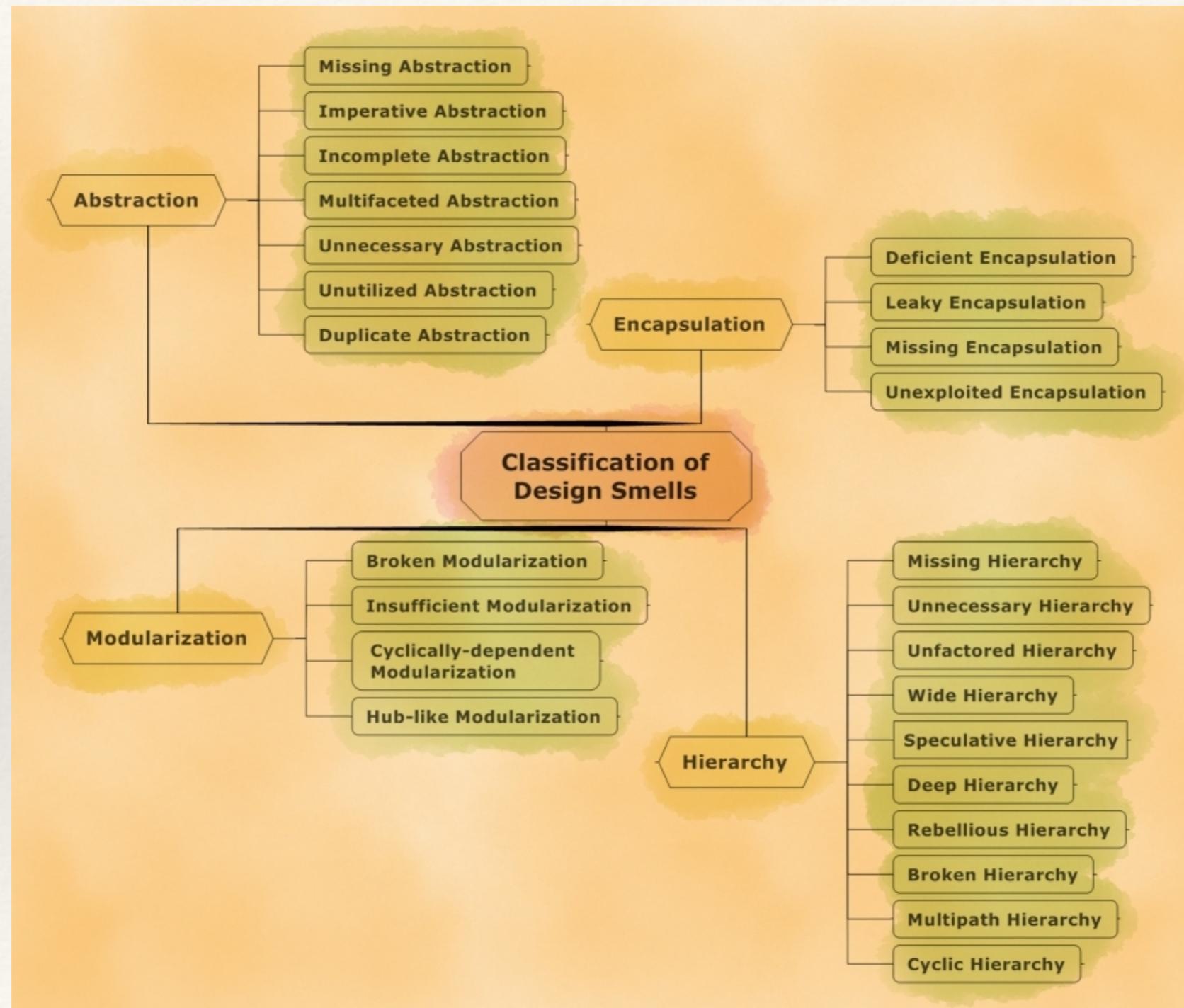


#XPIndia2016

# Proactive application: enabling techniques



# Reactive application: smells



# What are smells?

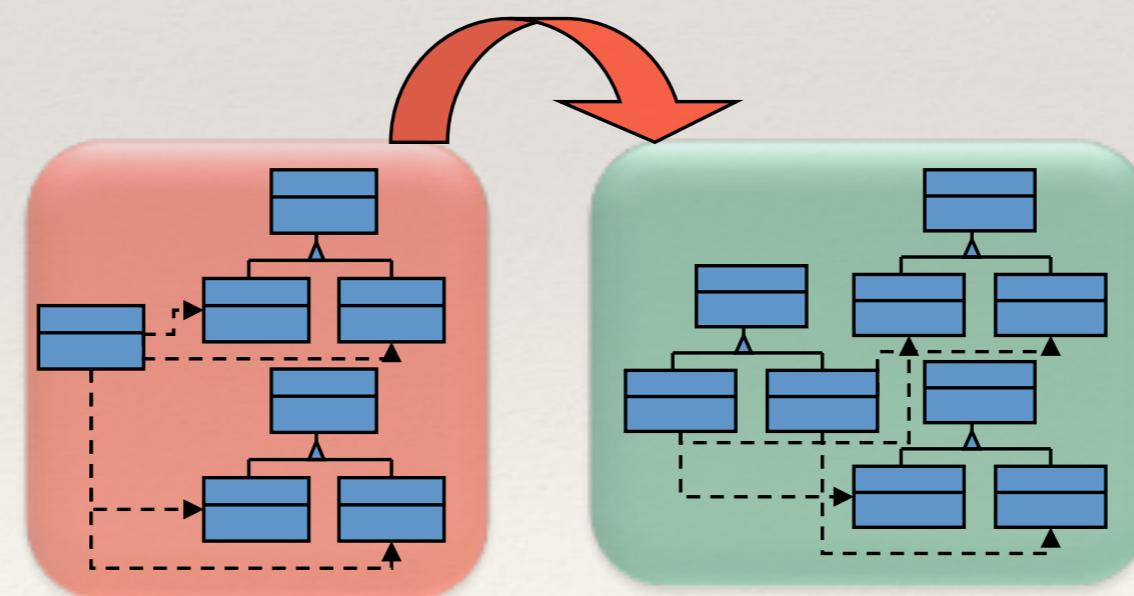
“Smells are certain structures  
in the code that suggest  
(sometimes they scream for)  
the possibility of refactoring.”



# What is refactoring?

**Refactoring (noun):** a *change* made to the *internal structure* of software to make it easier to *understand and cheaper to modify* without changing its observable behavior

**Refactor (verb):** to restructure software by applying a series of refactorings without changing its observable behavior



# Should this be “refactored”?

```
abstract class Printer {  
    private Integer portNumber = getPortNumber();  
    abstract Integer getPortNumber();  
    public static void main(String[]s) {  
        Printer p = new LPDPrinter();  
        System.out.println(p.portNumber);  
    }  
}  
  
class LPDPrinter extends Printer {  
    /* Line Printer Deamon port no is 515 */  
    private Integer defaultPortNumber = 515;  
    Integer getPortNumber() {  
        return defaultPortNumber;  
    }  
}
```



# Should this be “refactored”?

Does this have “long message chains” smell?

```
Calendar calendar = new Calendar.Builder()  
    .set(YEAR, 2003)  
    .set(MONTH, APRIL)  
    .set(DATE, 6)  
    .set(HOUR, 15)  
    .set(MINUTE, 45)  
    .set(SECOND, 22)  
    .setTimeZone(TimeZone.getDefault())  
    .build();  
System.out.println(calendar);
```

No - it's a feature  
(use of builder pattern)!



#XPIndia2016

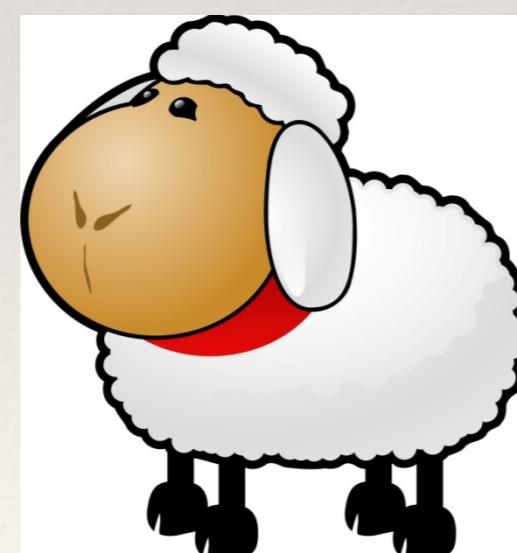
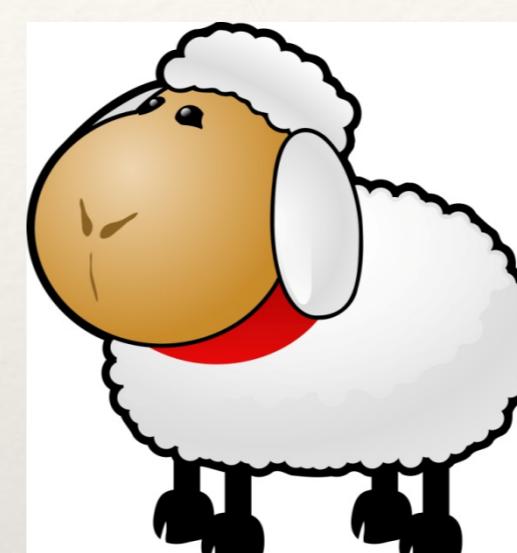
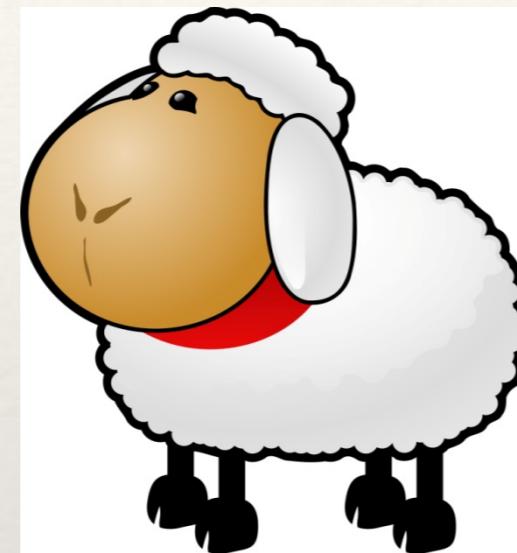
# Design smells: Example

“Lazy class”  
smell

```
public class FormattableFlags {  
    // Explicit instantiation of this class is prohibited.  
    private FormattableFlags() {}  
    /** Left-justifies the output. */  
    public static final int LEFT_JUSTIFY = 1<<0; // '-'  
    /** Converts the output to upper case */  
    public static final int UPPERCASE = 1<<1; // 'S'  
    /** Requires the output to use an alternate form. */  
    public static final int ALTERNATE = 1<<2; // '#'  
}
```



# Duplicated Code



# Duplicated Code

## Type 1

- **exactly identical** except for variations in whitespace, layout, and comments

## Type 2

- **syntactically identical** except for variation in symbol names, whitespace, layout, and comments

## Type 3

- identical except some **statements changed, added, or removed**

## Type 4

- when the fragments are **semantically identical** but implemented by syntactic variants

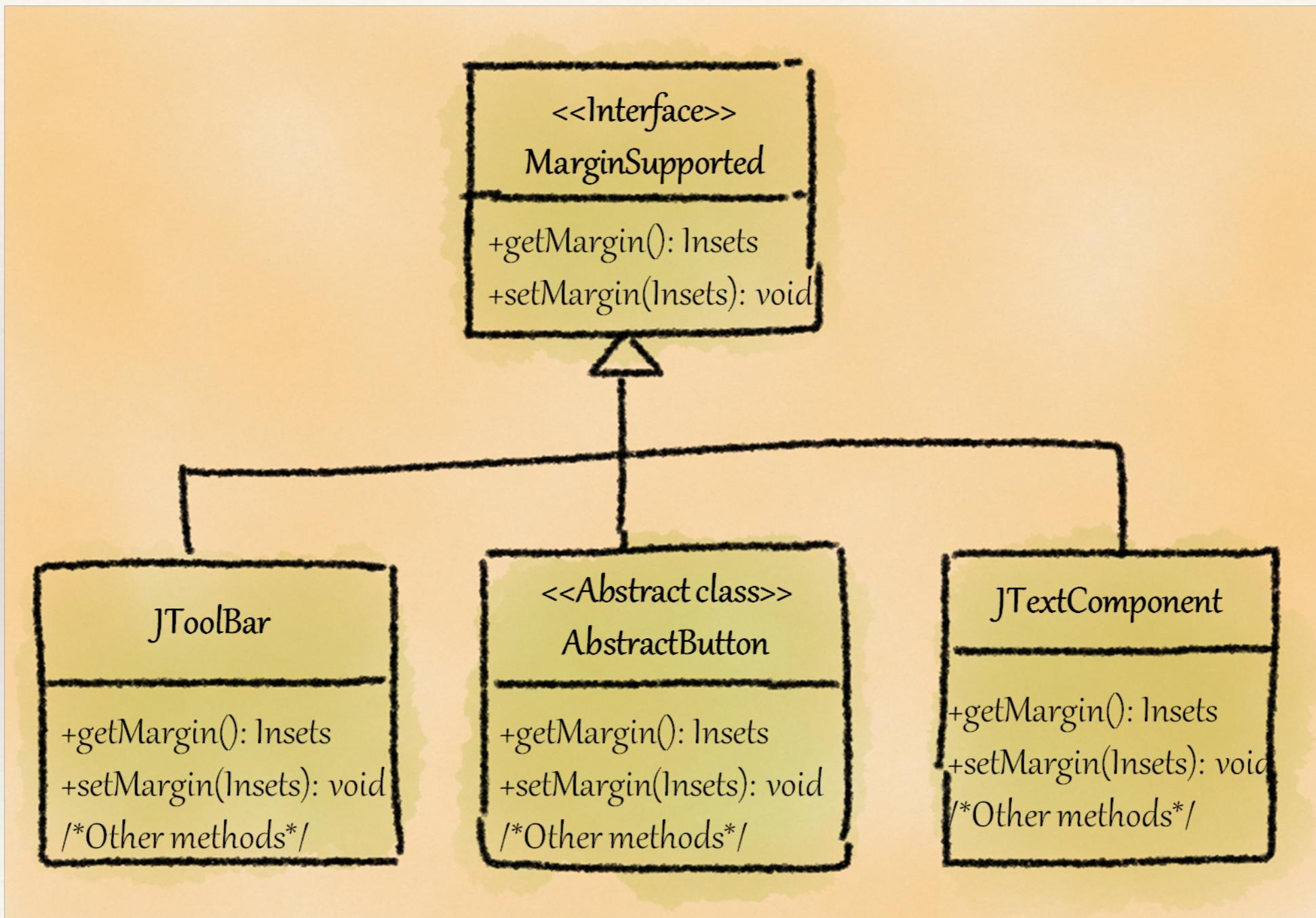


# What's that smell?

```
public Insets getBorderInsets(Component c, Insets insets){  
    Insets margin = null;  
    // Ideally we'd have an interface defined for classes which  
    // support margins (to avoid this hackery), but we've  
    // decided against it for simplicity  
    //  
    if (c instanceof AbstractButton) {  
        margin = ((AbstractButton)c).getMargin();  
    } else if (c instanceof JToolBar) {  
        margin = ((JToolBar)c).getMargin();  
    } else if (c instanceof JTextComponent) {  
        margin = ((JTextComponent)c).getMargin();  
    }  
    // rest of the code omitted ...
```



# Refactoring



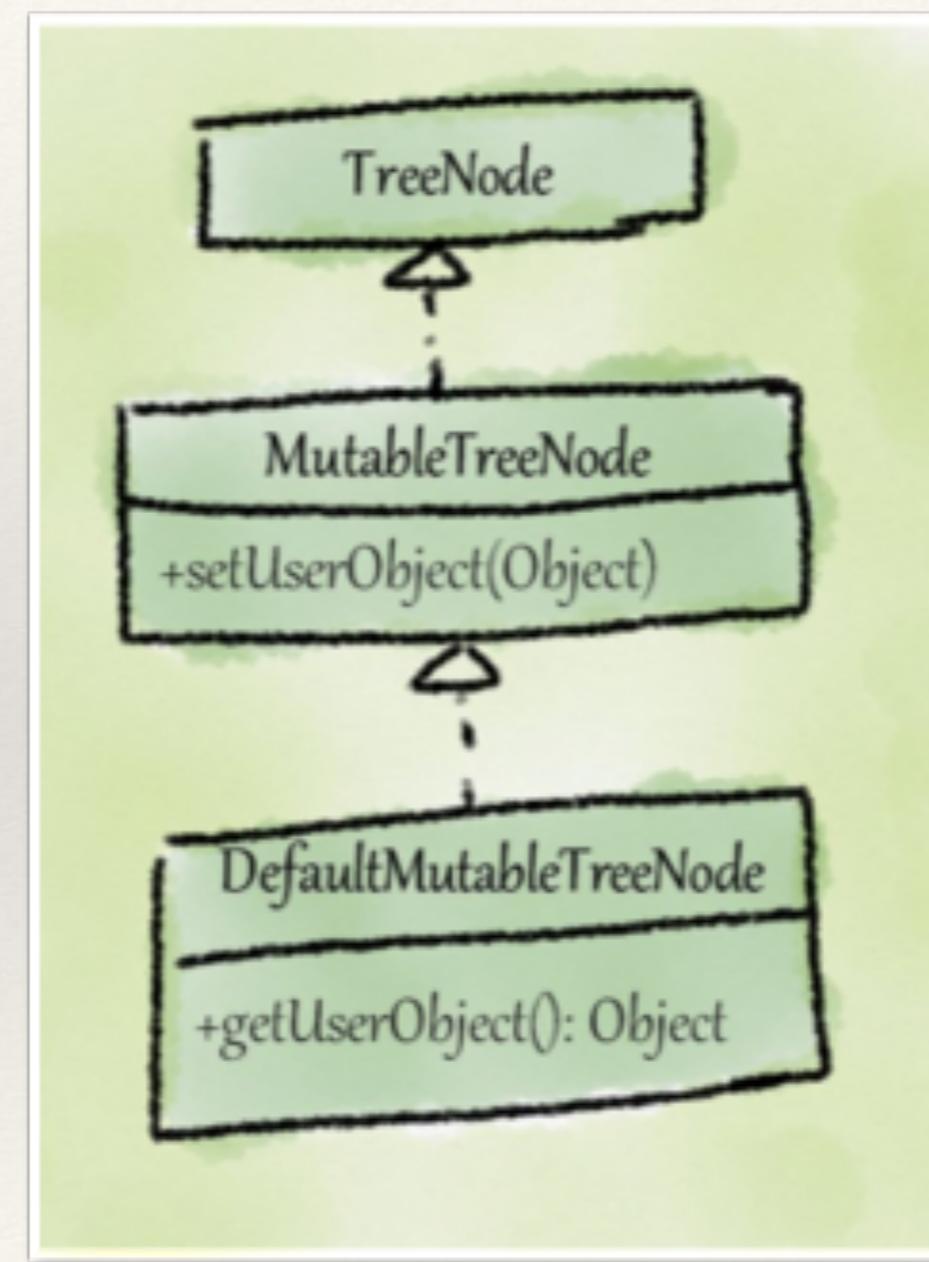
# Refactoring

```
if (c instanceof AbstractButton) {  
    margin = ((AbstractButton)c).getMargin();  
} else if (c instanceof JToolBar) {  
    margin = ((JToolBar)c).getMargin();  
} else if (c instanceof JTextComponent) {  
    margin = ((JTextComponent)c).getMargin();  
}
```

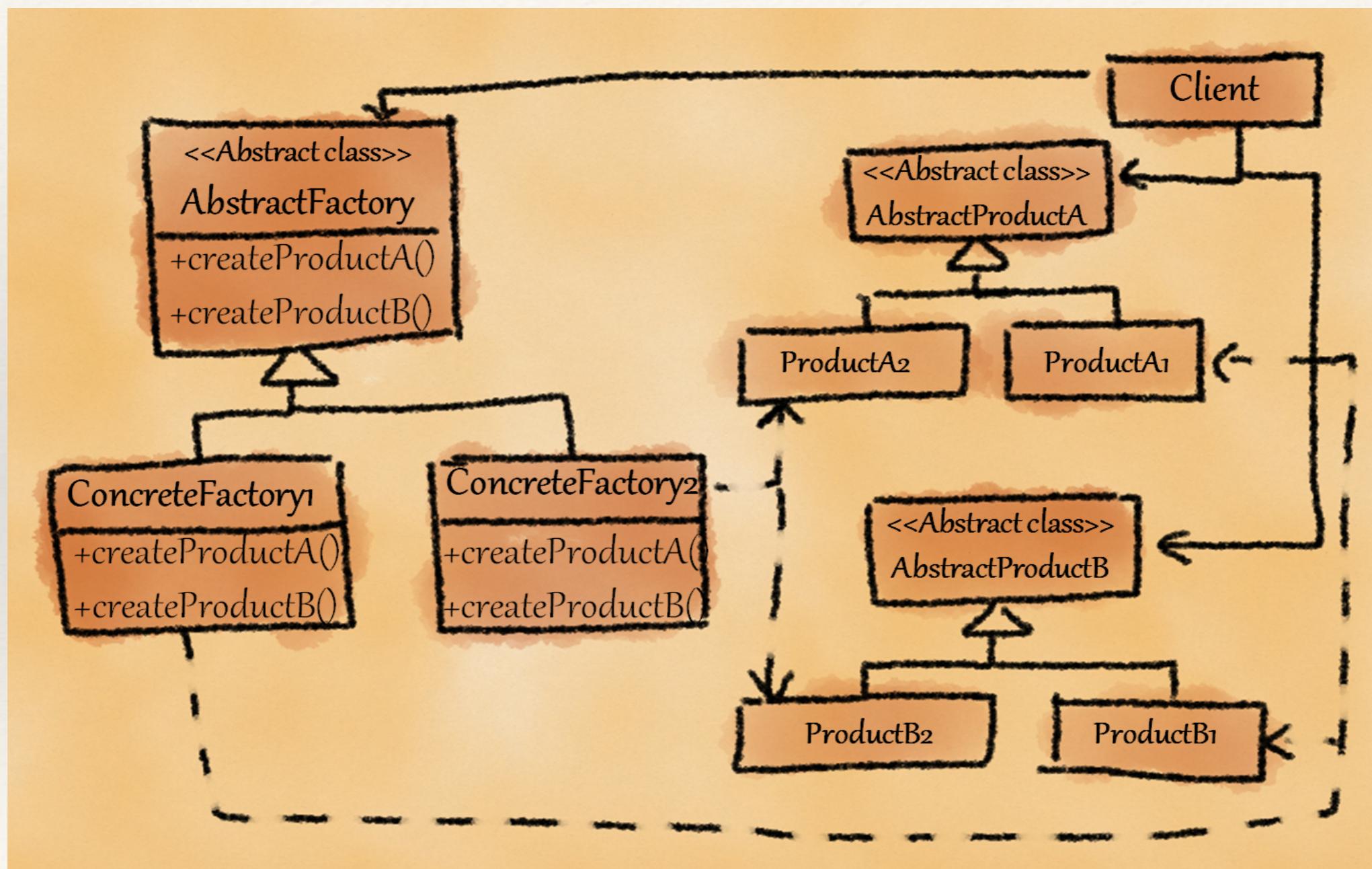


```
margin = c.getMargin();
```

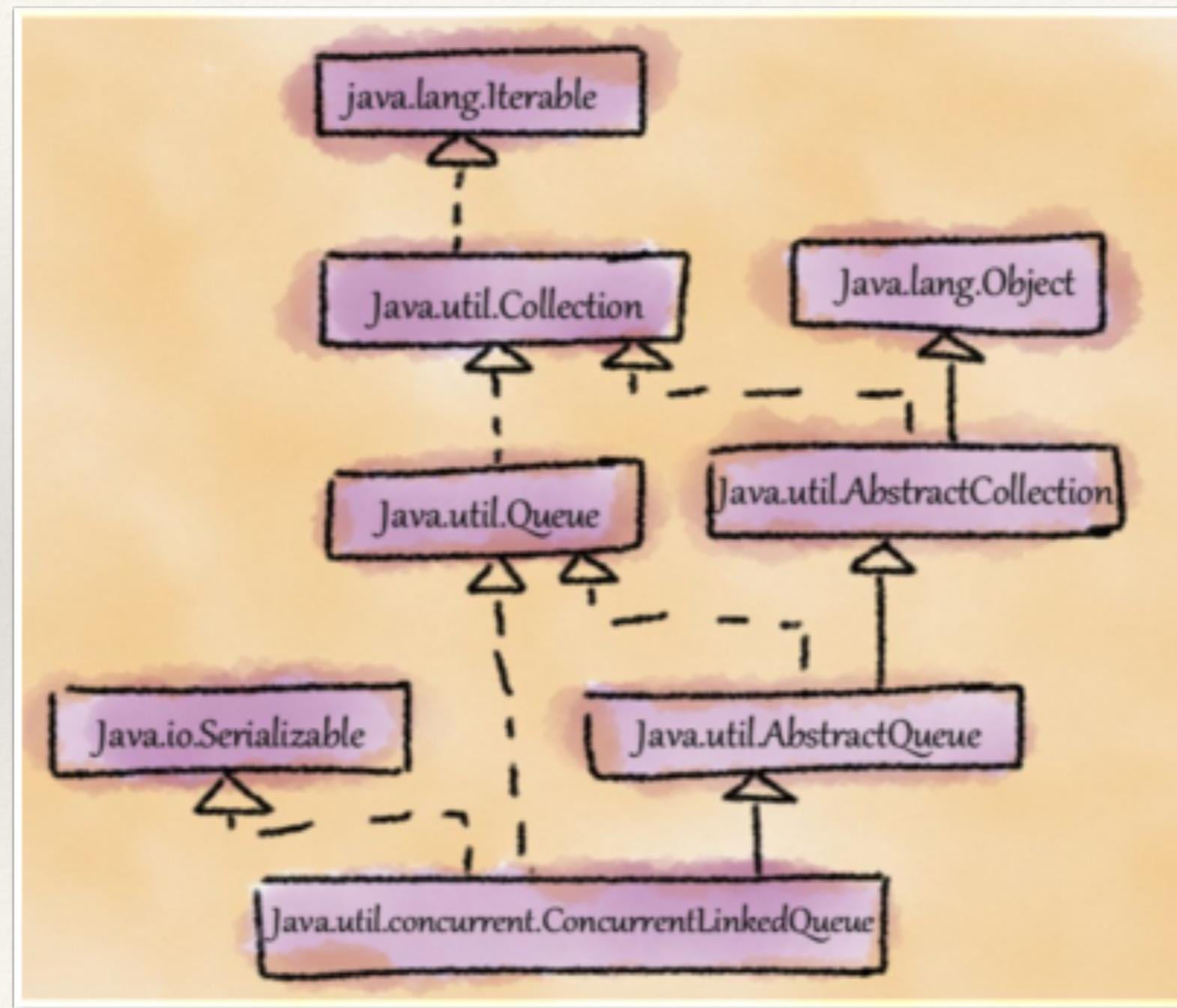
# Design smells: Example



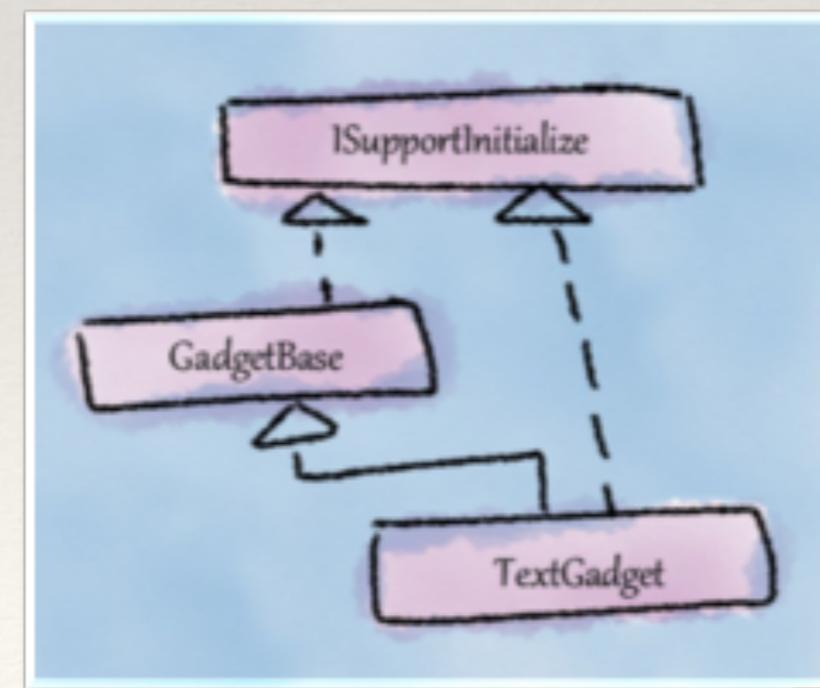
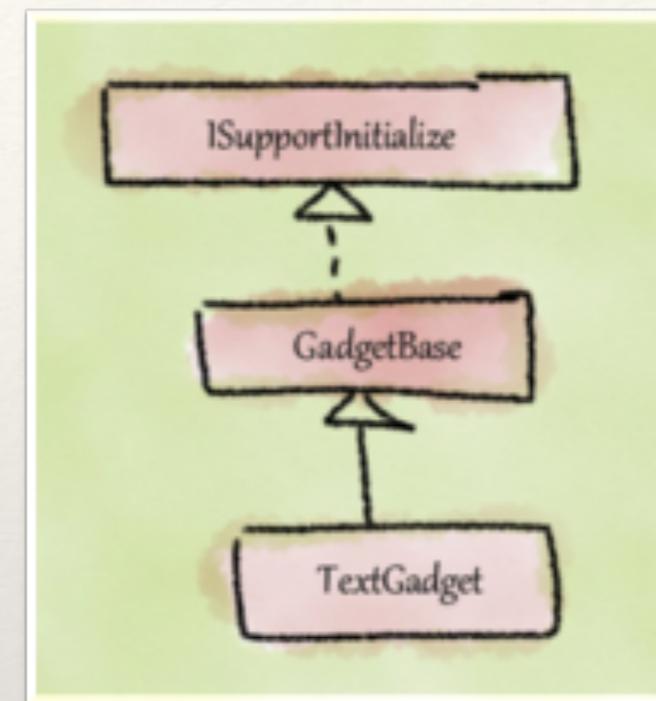
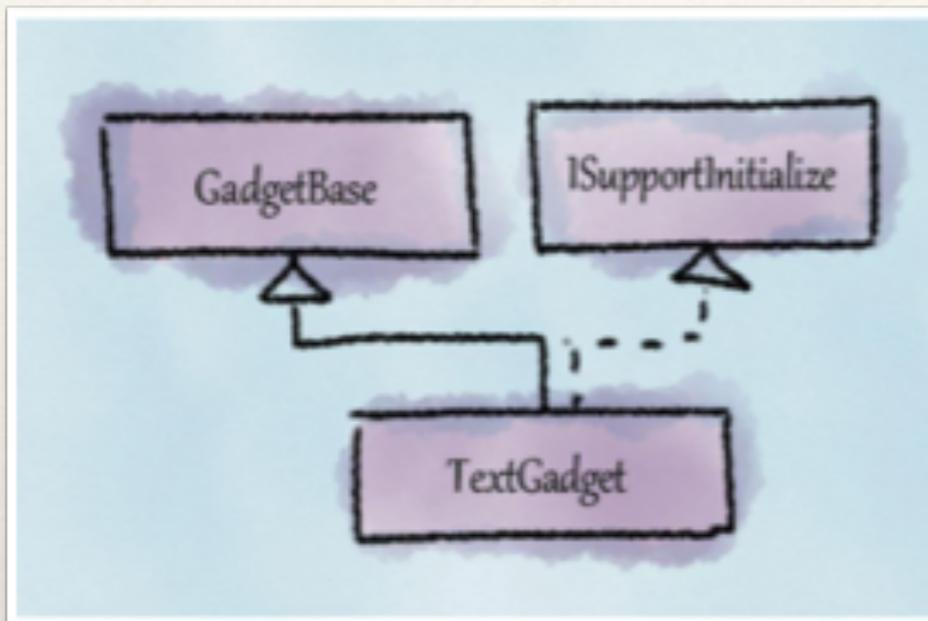
# Discussion example



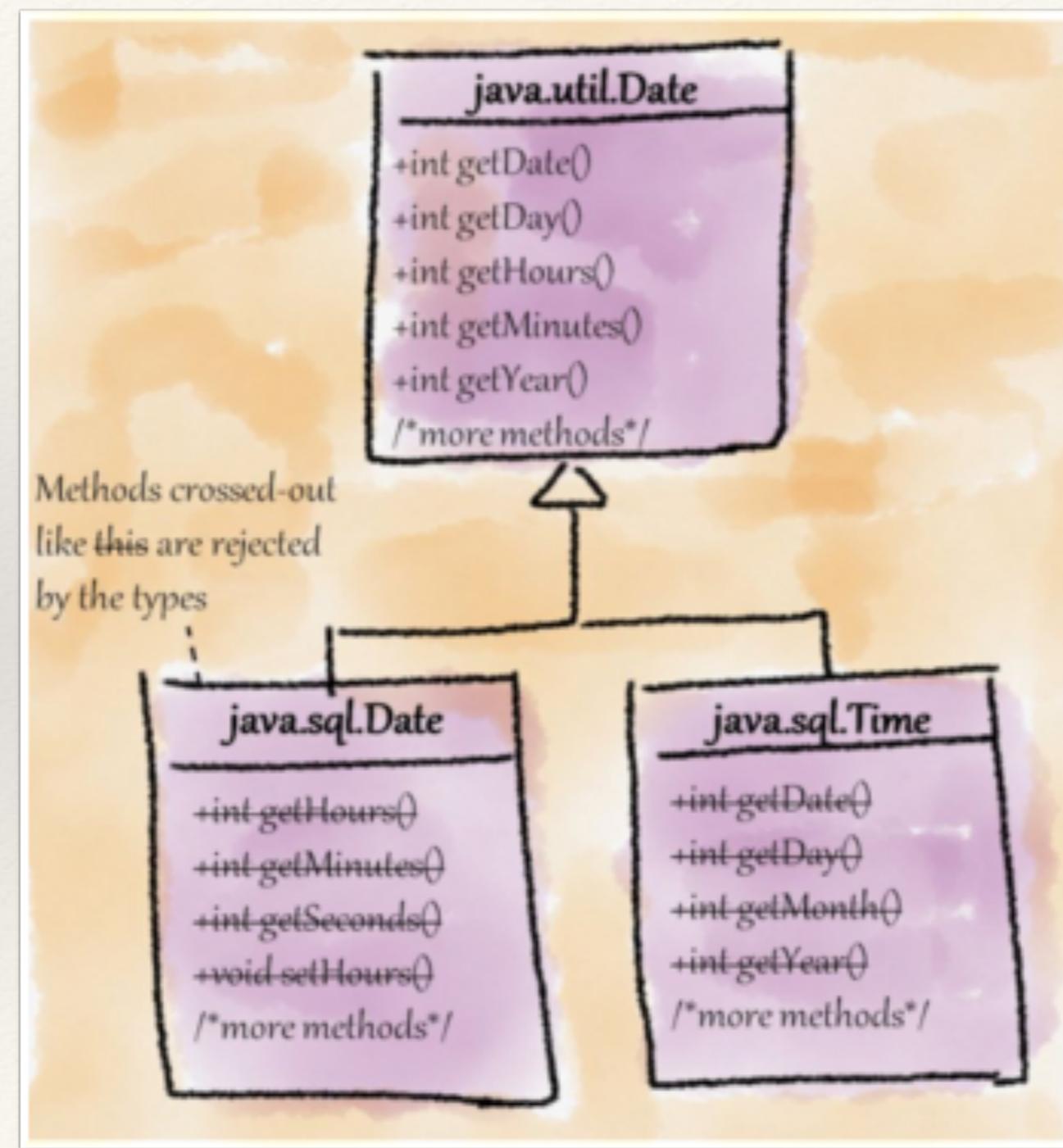
# Design smells: Example



# Discussion example



# Design smells: example #3



# Discussion example

```
// using java.util.Date  
Date today = new Date();  
System.out.println(today);
```

```
$ java DateUse  
Wed Dec 02 17:17:08 IST 2015
```

Why should we get the time and timezone details if I only want a date? Can I get rid of these parts?

No!



#XPIndia2016

# So what!

```
Date today = new Date();
System.out.println(today);
Date todayAgain = new Date();
System.out.println(todayAgain);
```

```
System.out.println(today.compareTo(todayAgain) == 0);
```

```
Thu Mar 17 13:21:55 IST 2016
Thu Mar 17 13:21:55 IST 2016
false
```

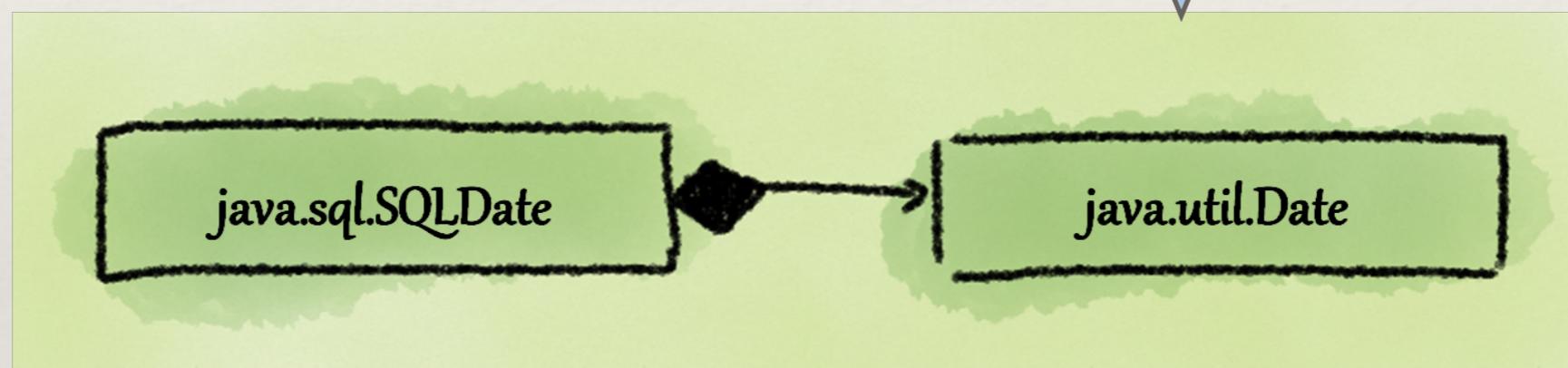
What is going  
on here?



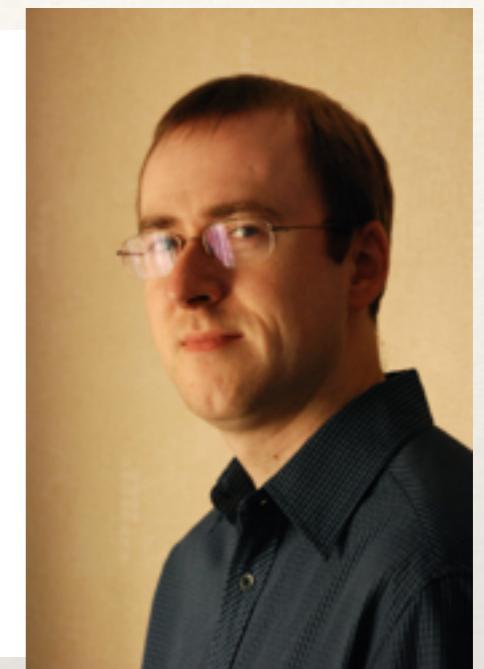
#XPIndia2016

# Refactoring for Date

Replace inheritance  
with delegation



# Joda API



Stephen Colebourne



#XPIndia2016

# java.time package!



**Date, Calendar, and TimeZone**

Java 8 replaces  
these types

# Refactored solution

```
LocalDate today = LocalDate.now();
System.out.println(today);
LocalDate todayAgain = LocalDate.now();
System.out.println(todayAgain);

System.out.println(today.compareTo(todayAgain) == 0);
```

2016-03-17  
2016-03-17  
**true**

Works fine  
now!



# Refactored example ...

```
LocalDate today = LocalDate.now();  
System.out.println(today);
```

```
LocalTime now = LocalTime.now();  
System.out.println(now);
```

```
ZoneId id = ZoneId.of("Asia/Tokyo");  
System.out.println(id);
```

```
LocalDateTime todayAndNow = LocalDateTime.now();  
System.out.println(todayAndNow);
```

```
ZonedDateTime todayAndNowInTokyo = ZonedDateTime.now(ZoneId.of("Asia/Tokyo"));  
System.out.println(todayAndNowInTokyo);
```

```
2016-03-17  
13:28:06.927  
Asia/Tokyo  
2016-03-17T13:28:06.928  
2016-03-17T16:58:06.929+09:00[Asia/Tokyo]
```



You can use only date, time, or even timezone, and combine them as needed!



# More classes in Date/Time API

## Instant

- Represents machine time starting from Unix epoch
- Typically used for timestamps

## Period

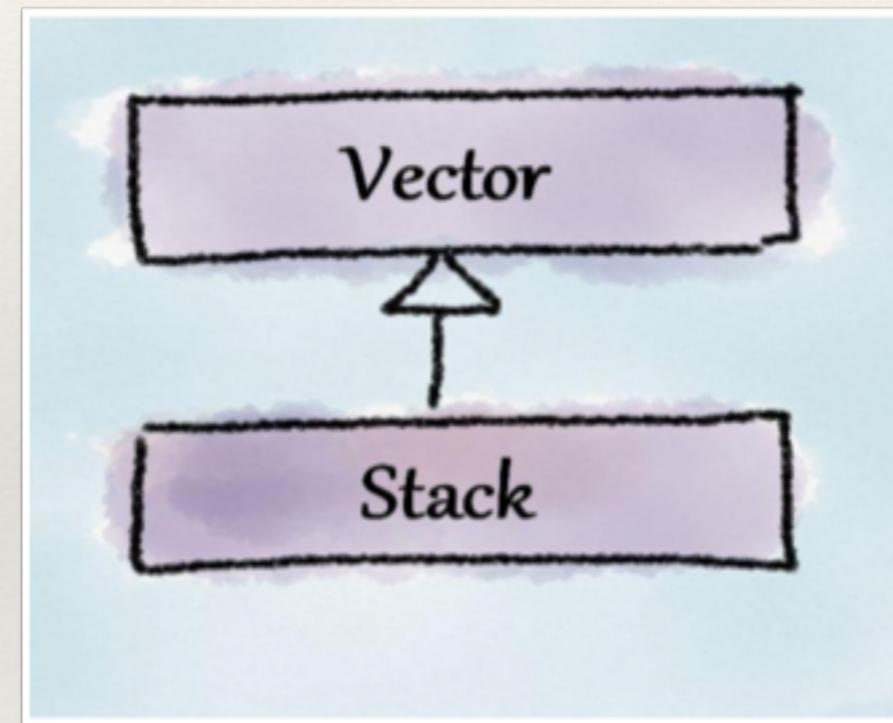
- Represents amount of time in terms of years, months and days
- Typically used for difference between two LocalDate objects

## Duration

- Represents amount of time in terms of hours, minutes, seconds, and fractions of seconds
- Typically used for difference between two LocalTime objects



# What's that smell?



# Liskov's Substitution Principle (LSP)



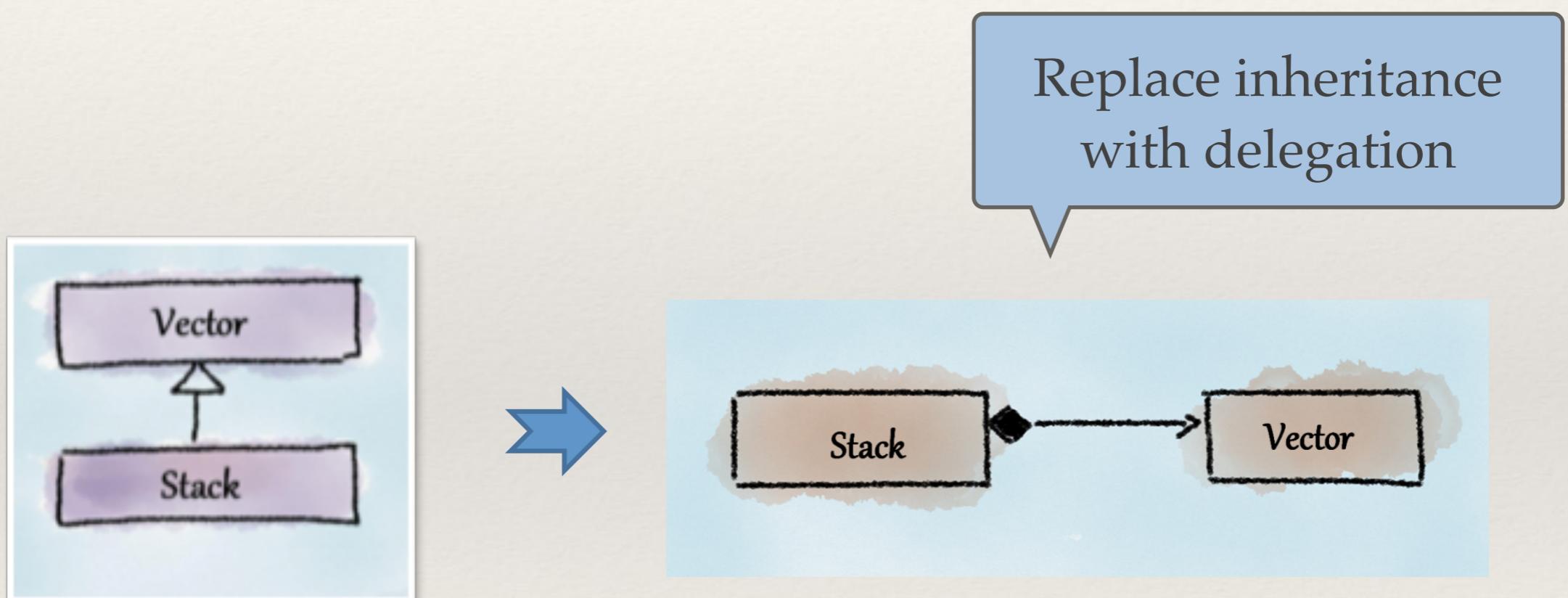
It should be possible to replace  
objects of supertype with  
objects of subtypes without  
altering the desired behavior of  
the program

Barbara Liskov



#XPIndia2016

# Refactoring



# What's that smell?

```
switch (transferType) {  
    case DataBuffer.TYPE_BYTE:  
        byte bdata[] = (byte[])inData;  
        pixel = bdata[0] & 0xff;  
        length = bdata.length;  
        break;  
  
    case DataBuffer.TYPE USHORT:  
        short sdata[] = (short[])inData;  
        pixel = sdata[0] & 0xffff;  
        length = sdata.length;  
        break;  
  
    case DataBuffer.TYPE_INT:  
        int idata[] = (int[])inData;  
        pixel = idata[0];  
        length = idata.length;  
        break;  
  
    default:  
        throw new UnsupportedOperationException("This method has not been "+ "implemented  
for transferType " + transferType);  
}
```

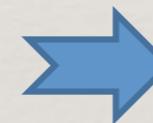
# Replace conditional with polymorphism

```
protected int transferType;
```



```
protected DataBuffer dataBuffer;
```

```
switch (transferType) {  
    case DataBuffer.TYPE_BYTE:  
        byte bdata[] = (byte[])inData;  
        pixel = bdata[0] & 0xff;  
        length = bdata.length;  
        break;  
    case DataBuffer.TYPE USHORT:  
        short sdata[] = (short[])inData;  
        pixel = sdata[0] & 0xffff;  
        length = sdata.length;  
        break;  
    case DataBuffer.TYPE_INT:  
        int idata[] = (int[])inData;  
        pixel = idata[0];  
        length = idata.length;  
        break;  
    default:  
        throw new UnsupportedOperationException("This method  
has not been "+ "implemented for transferType " +  
transferType);  
}
```



```
pixel = dataBuffer.getPixel();  
length = dataBuffer.getSize();
```



# Scenario

```
class Plus extends Expr {  
    private Expr left, right;  
    public Plus(Expr arg1, Expr arg2) {  
        left = arg1;  
        right = arg2;  
    }  
    public void genCode() {  
        left.genCode();  
        right.genCode();  
        if(t == Target.JVM) {  
            System.out.println("iadd");  
        }  
        else { // DOTNET  
            System.out.println("add");  
        }  
    }  
}
```

How to separate:

- a) code generation logic from node types?
- b) how to support different target types?



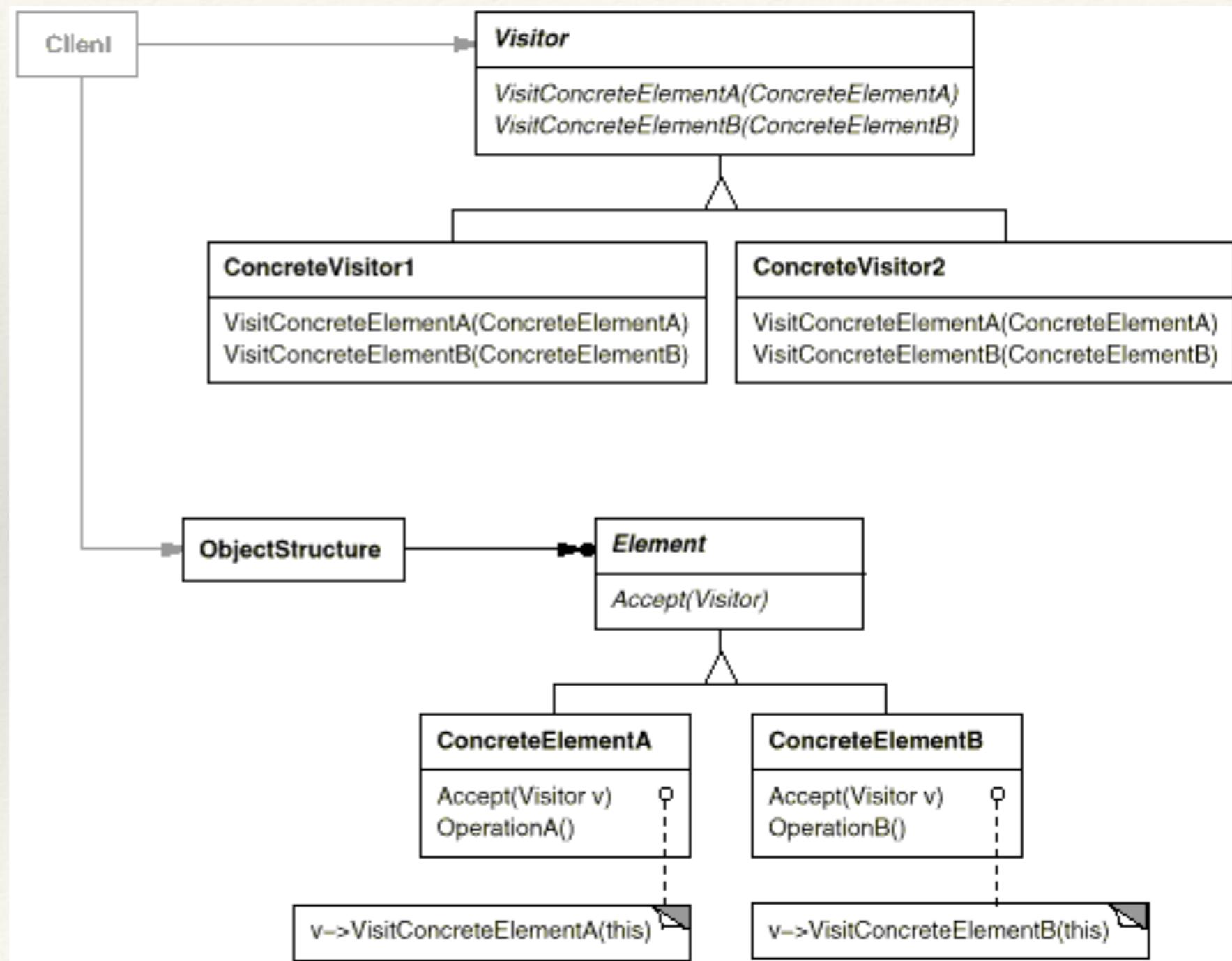
# A solution using Visitor pattern

```
class Plus extends Expr {  
    private Expr left, right;  
    public Plus(Expr arg1, Expr arg2) {  
        left = arg1;  
        right = arg2;  
    }  
    public Expr getLeft() {  
        return left;  
    }  
    public Expr getRight() {  
        return right;  
    }  
    public void accept(Visitor v) {  
        v.visit(this);  
    }  
}
```

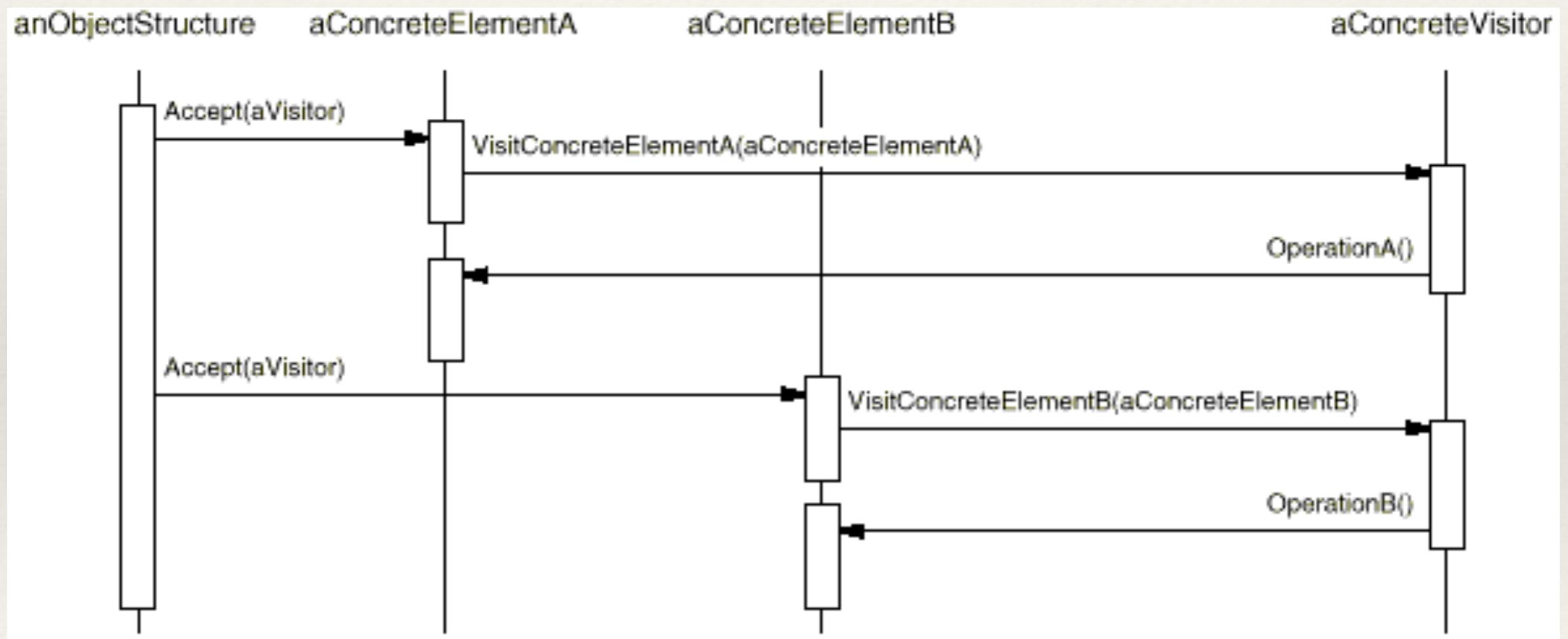
```
class DOTNETVisitor extends Visitor {  
    public void visit(Constant arg) {  
        System.out.println("ldarg " + arg.getVal());  
    }  
    public void visit(Plus plus) {  
        genCode(plus.getLeft());  
        genCode(plus.getRight());  
        System.out.println("add");  
    }  
    public void visit(Sub sub) {  
        genCode(sub.getLeft());  
        genCode(sub.getRight());  
        System.out.println("sub");  
    }  
    public void genCode(Expr expr) {  
        expr.accept(this);  
    }  
}
```



# Visitor pattern: structure



# Visitor pattern: call sequence



# Visitor pattern: Discussion

Represent an operation to be performed on the elements of an object structure.  
Visitor lets you define a new operation without changing the classes of the  
elements on which it operates



- ❖ Create two class hierarchies:
  - ❖ One for the elements being operated on
  - ❖ One for the visitors that define operations on the elements

# Refactoring with Lambdas

```
File[] files = new File(".").listFiles(  
    new FileFilter() {  
        public boolean accept(File f) { return f.isFile(); }  
    }  
);  
for(File file: files) {  
    System.out.println(file);  
}
```



```
Arrays.stream(new File("."))  
    .listFiles(file -> file.isFile())  
    .forEach(System.out::println);
```

# Refactoring APIs

```
public class Throwable {  
    // following method is available from Java 1.0 version.  
    // Prints the stack trace as a string to standard output  
    // for processing a stack trace,  
    // we need to write regular expressions  
    public void printStackTrace();  
    // other methods omitted  
}
```



# Refactoring: Practical concerns

“Fear of breaking  
working code”

Where do I have time  
for refactoring?

How to refactor code in legacy projects (no automated tests, difficulty in understanding, lack of motivation, ...)?

Is management buy-in necessary for refactoring?

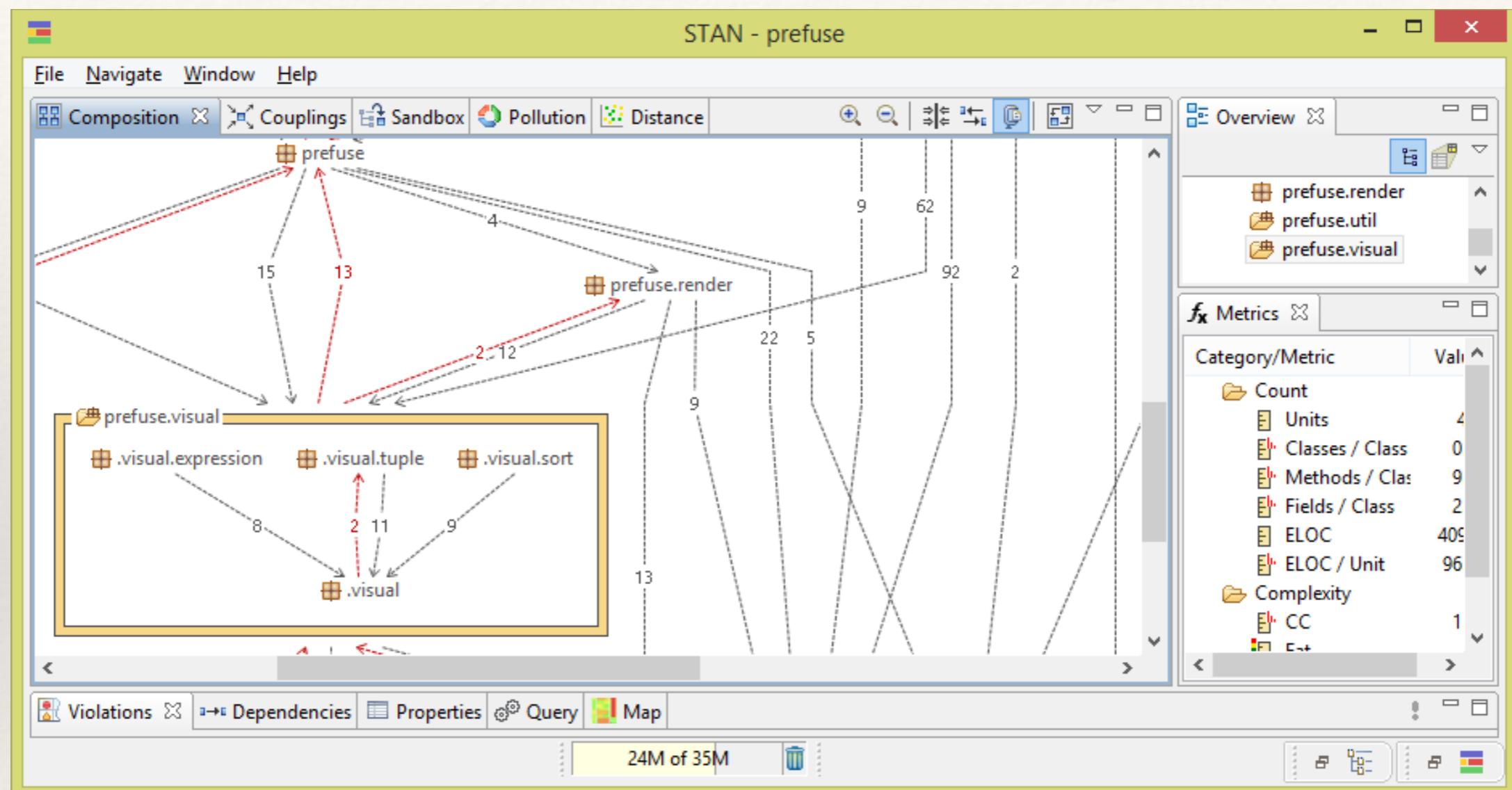


# Tool driven approach for design quality



#XPIndia2016

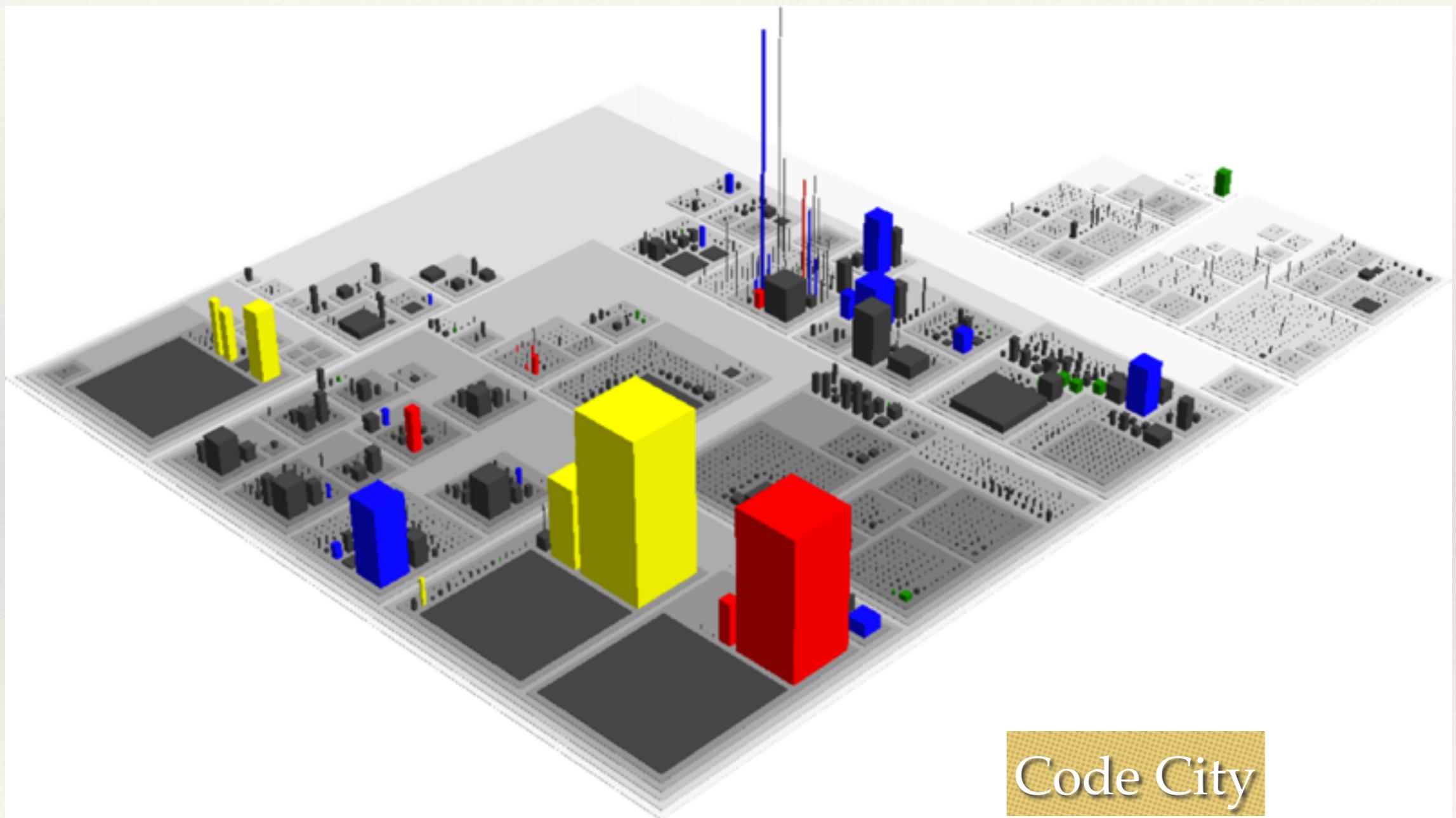
# Comprehension tools



STAN

<http://stan4j.com>

# Comprehension tools



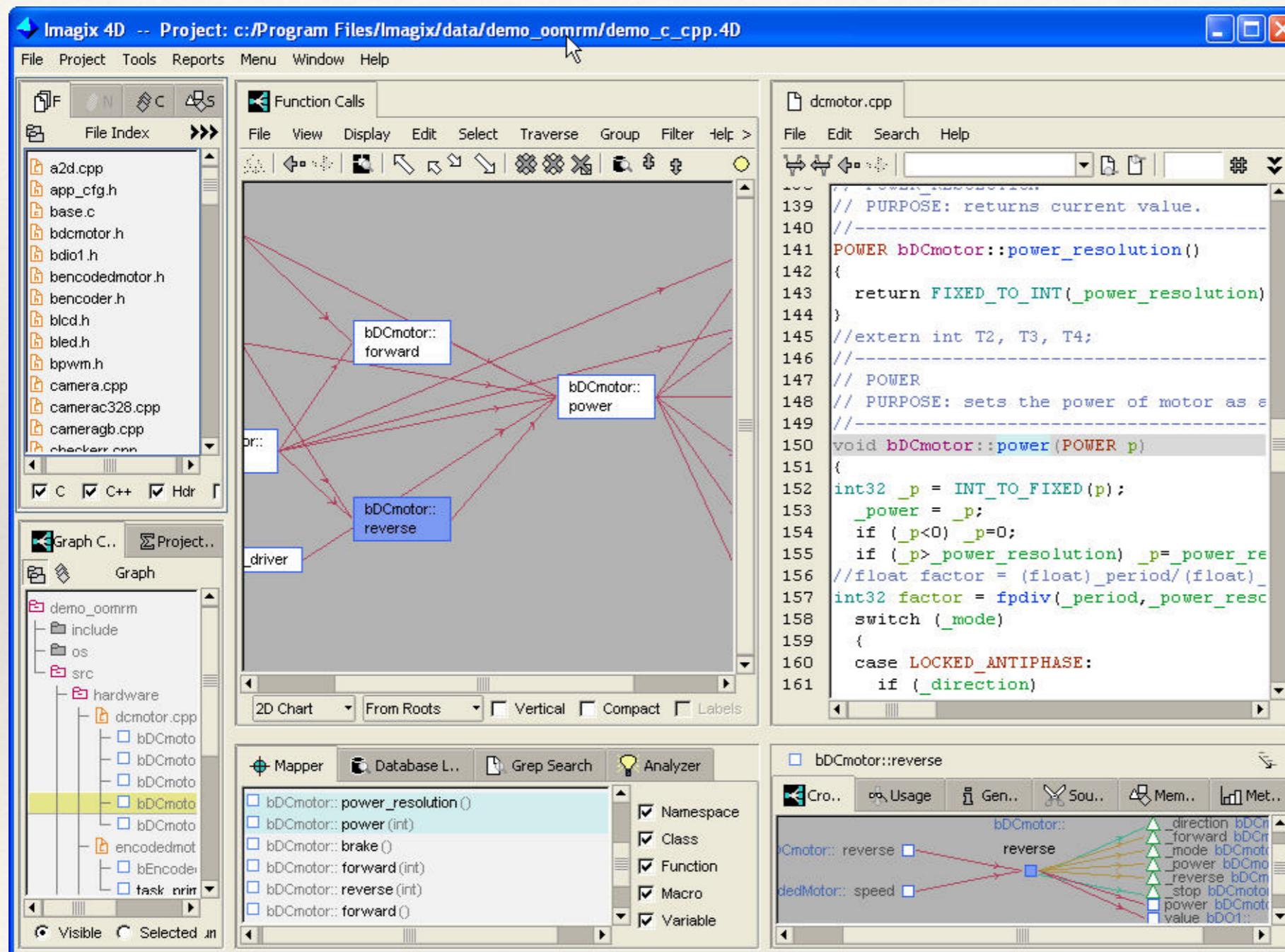
Code City

<http://www.inf.usi.ch/phd/wettel/codecity.html>



#XPIndia2016

# Comprehension tools



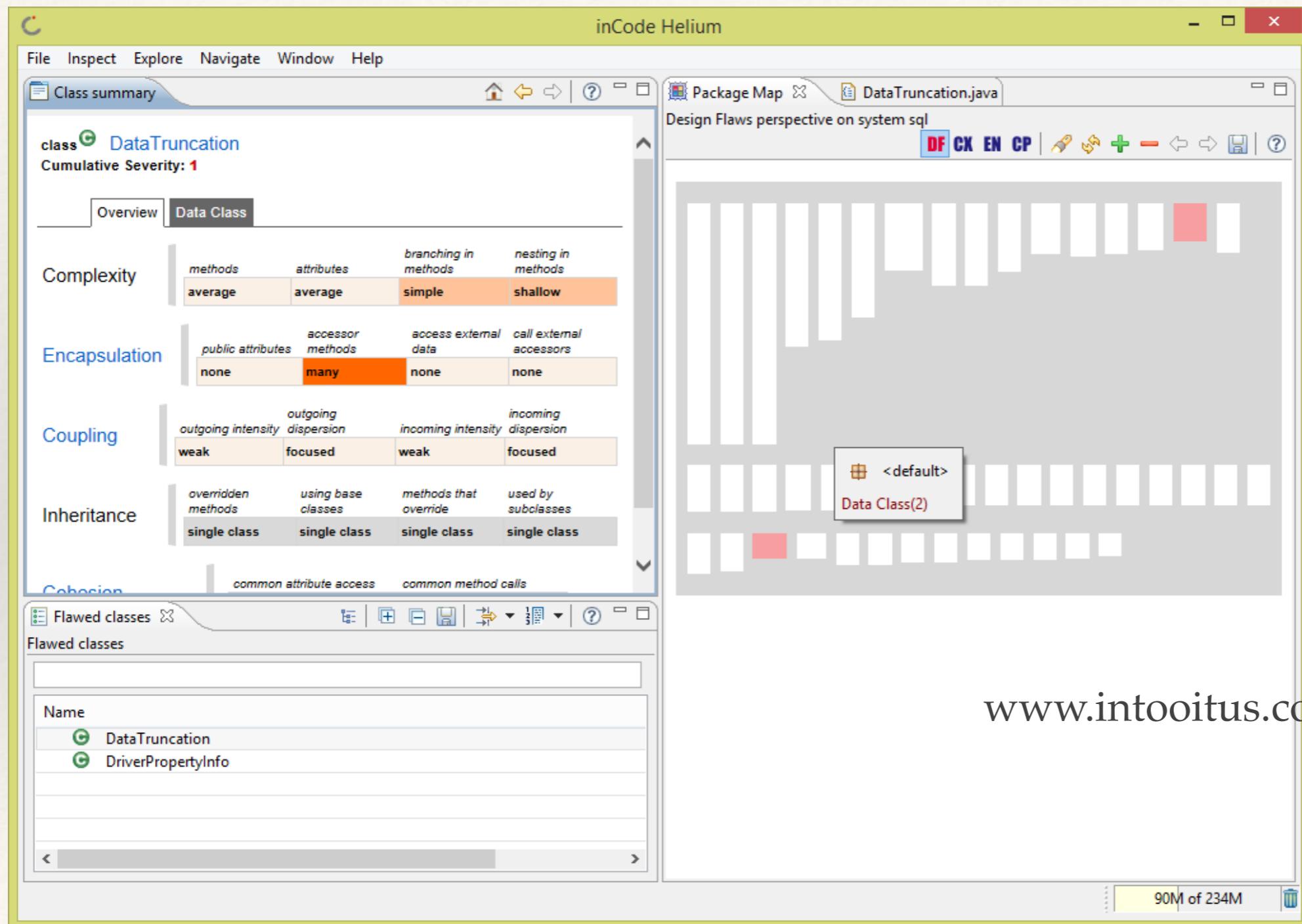
Imagix 4D

<http://www.imagix.com>



#XPIndia2016

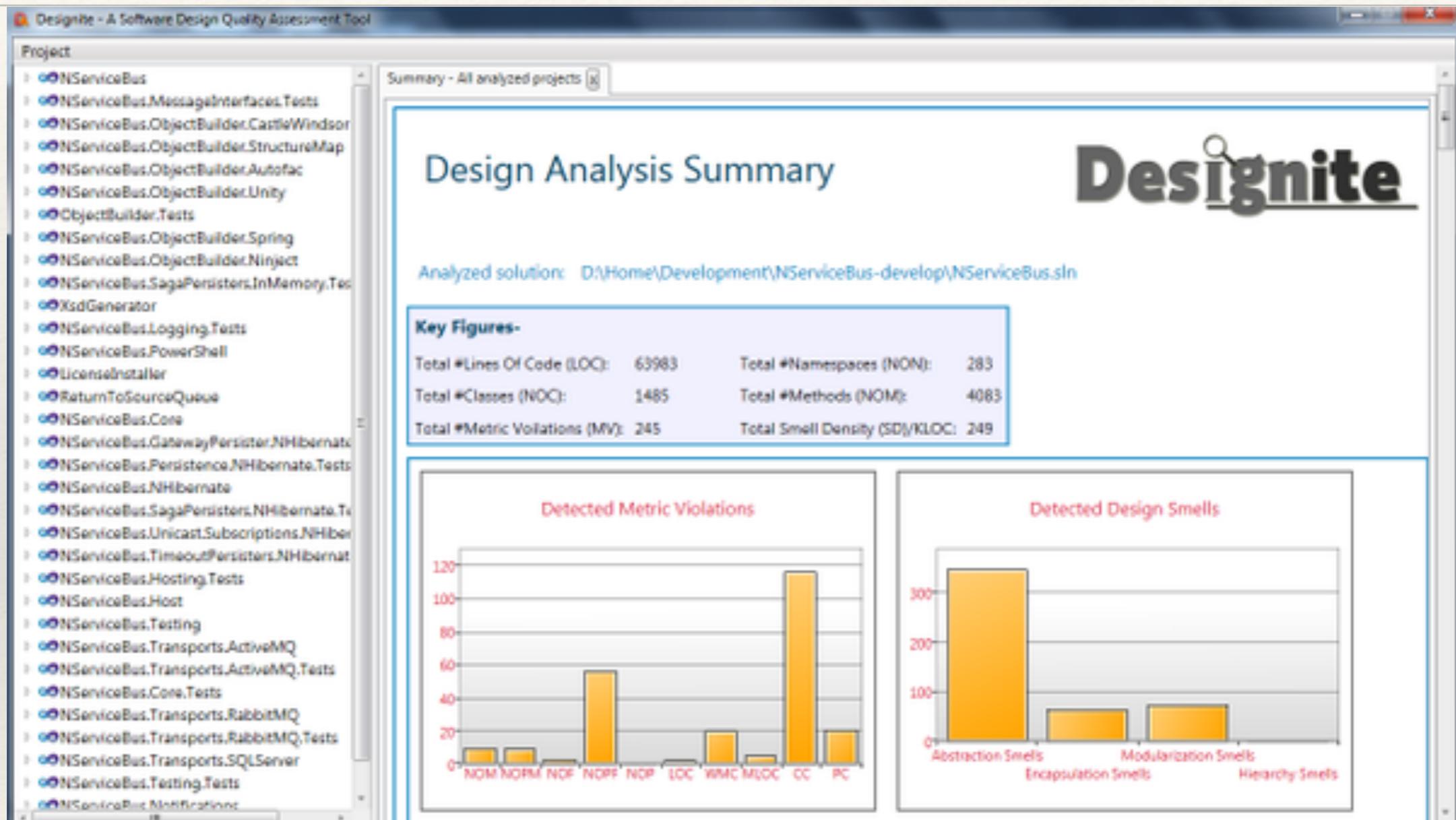
# Smell detection tools



Infusion

[www.intooitus.com/products/infusion](http://www.intooitus.com/products/infusion)

# Smell detection tools



Designite

[www.designite-tools.com](http://www.designite-tools.com)



#XPIndia2016

# Clone analysers

PMD Duplicate Code Detector

**File**

Root source directory: /usr/local/src/apache\_1.3.33/src/

Report duplicate chunks larger than: 75

Language: C++

Extension: .cpp

Also scan subdirectories?

Ignore literals and identifiers?

Go Cancel

Source	Matches	Lines
(2 separate files)		237
(2 separate files)		123
(2 separate files)		136
(2 separate files)		113
(2 separate files)		15
(2 separate files)		35
(2 separate files)		107
(2 separate files)		84
.../nametab.h		27
.../fcgi_config.c		68
(2 separate files)		29
.../os.c		39
.../uri_delims.h		6
.../http_main.c		42
.../ap_ebcdic.c		12
.../mod_include.c		20
(2 separate files)		36
.../mod_rewrite.c		28
.../ap_base64.c		5
.../ap_ebcdic.c		10
.../engine.c		19
.../proxy_ftp.c		23
.../mod_rewrite.c		20
(2 separate files)		42

```

Found a 36 line (173 tokens) duplication in the following files:
Starting at line 317 of /usr/local/src/apache_1.3.33/src/os/netware/mod_log_nw.c
Starting at line 320 of /usr/local/src/apache_1.3.33/src/modules/standard/mod_log_config.c
                                         ap_escape_logitem(r
                                         : "");
}
static const char *log_status(request_rec *r, char *a)
{
    return pfmt(r->pool, r->status);
}

static const char *clf_log_bytes_sent(request_rec *r, char *a)
{
    if (!r->sent_bodyct) {
        return "-";
    }
    else {
        long int bs;
        ap_bgetopt(r->connection->client, BO_BYTECT, &bs);
        return ap_psprintf(r->pool, "%ld", bs);
    }
}

static const char *log_bytes_sent(request_rec *r, char *a)
{
    if (!r->sent_header) {

```

PMD Copy Paste Detector (CPD)

<http://pmd.sourceforge.net/pmd-4.3.0/cpd.html>

# Metric tools

Understand

<https://scitools.com>

Project Metrics Browser

Project metrics for: C:\projects\C++\pixie.udb

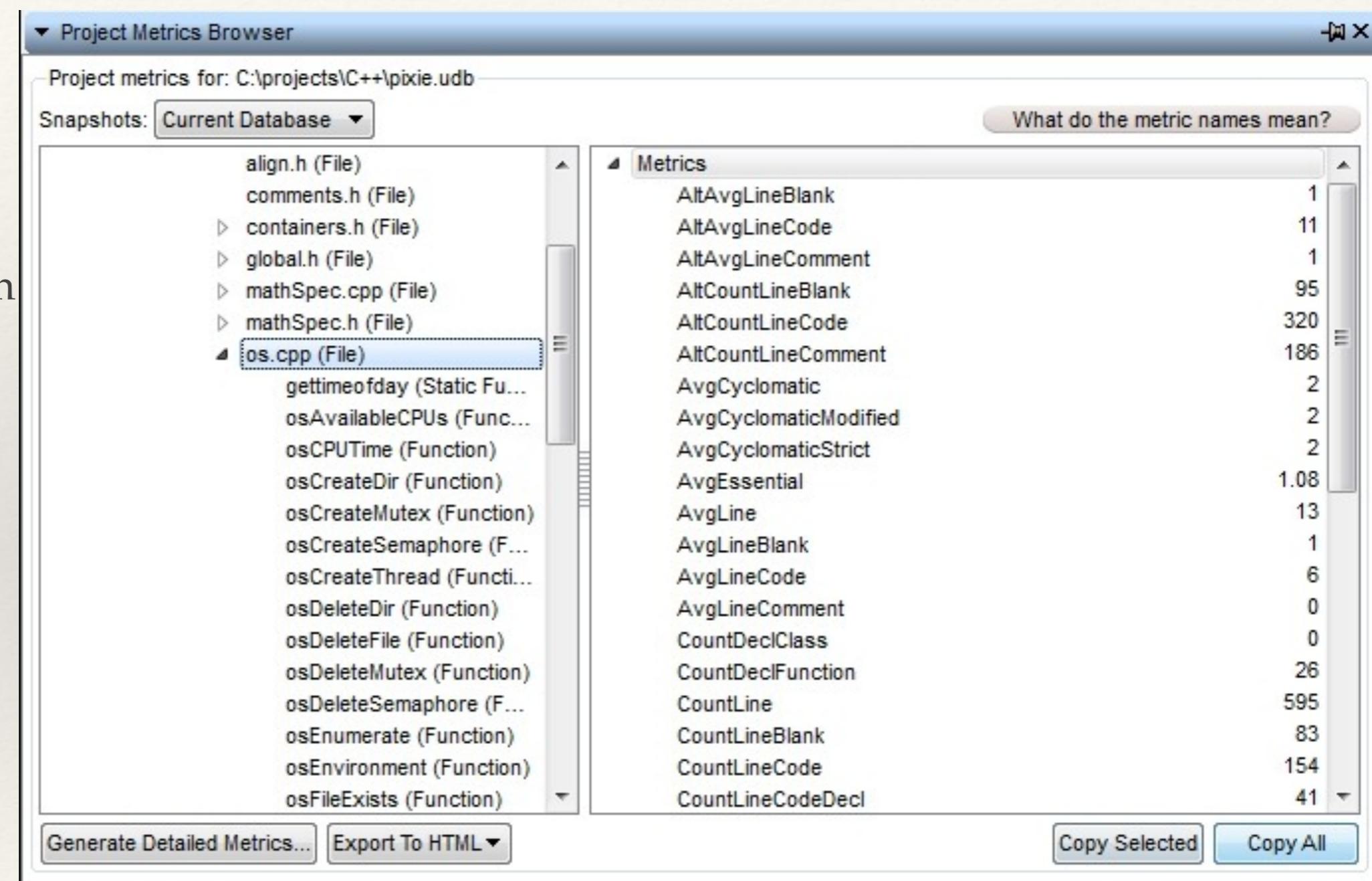
Snapshots: Current Database ▾

What do the metric names mean?

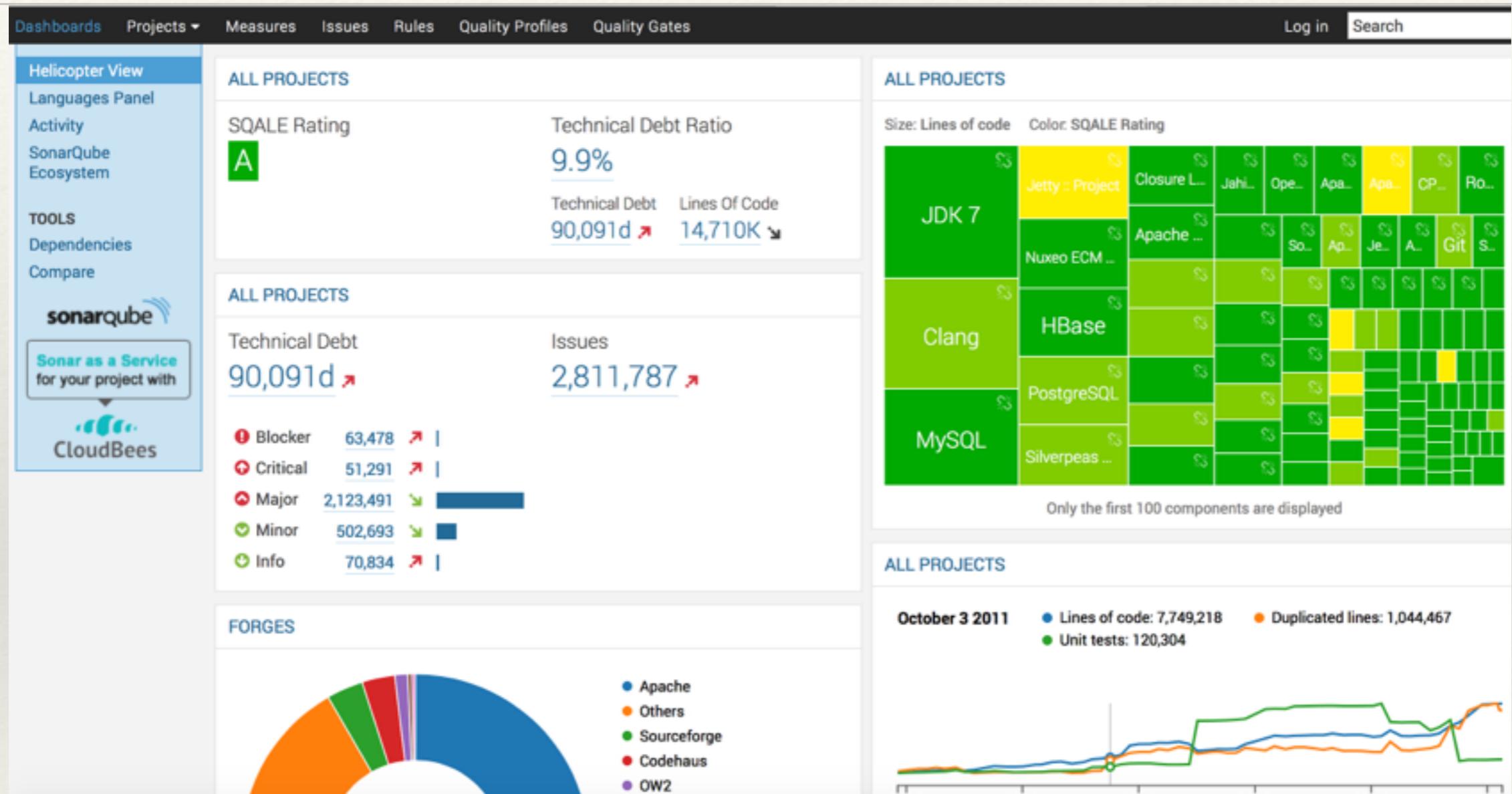
Metrics	
AltAvgLineBlank	1
AltAvgLineCode	11
AltAvgLineComment	1
AltCountLineBlank	95
AltCountLineCode	320
AltCountLineComment	186
AvgCyclomatic	2
AvgCyclomaticModified	2
AvgCyclomaticStrict	2
AvgEssential	1.08
AvgLine	13
AvgLineBlank	1
AvgLineCode	6
AvgLineComment	0
CountDeclClass	0
CountDeclFunction	26
CountLine	595
CountLineBlank	83
CountLineCode	154
CountLineCodeDecl	41

align.h (File)  
comments.h (File)  
containers.h (File)  
global.h (File)  
mathSpec.cpp (File)  
mathSpec.h (File)  
os.cpp (File)  
    gettimeofday (Static Fu...  
    osAvailableCPUs (Func...  
    osCPUTime (Function)  
    osCreateDir (Function)  
    osCreateMutex (Function)  
    osCreateSemaphore (F...  
    osCreateThread (Functi...  
    osDeleteDir (Function)  
    osDeleteFile (Function)  
    osDeleteMutex (Function)  
    osDeleteSemaphore (F...  
    osEnumerate (Function)  
    osEnvironment (Function)  
    osFileExists (Function)

Generate Detailed Metrics... Export To HTML ▾ Copy Selected Copy All



# Technical debt quantification/visualization tools



Sonarqube

<http://www.sonarqube.org>

# Refactoring tools

## Contents



Code Analysis



Navigation and Search



Coding Assistance



Refactorings



Project Level Features



Code Generation



Code Templates



Code Cleanup



Unit Testing



Internationalization

## ReSharper Features

*The ultimate Agile tool is ReSharper. It is the one thing for .NET developers that removes fear of change. Refactoring is just so darn easy that change isn't scary.*

Jaco Pretorius  
ThoughtWorks

### Code quality analysis

On-the-fly [code quality analysis](#) in C#, VB.NET, XAML, ASP.NET, ASP.NET MVC, JavaScript, TypeScript, CSS, HTML, and XML. ReSharper will tell you right away if your code contains errors or can be improved.

### Eliminate errors and code smells

Instant fixes to [eliminate errors and code smells](#). Not only does ReSharper warn you when there's a problem in your code but it provides quick-fixes to solve them automatically.

### Instantly traverse your entire solution

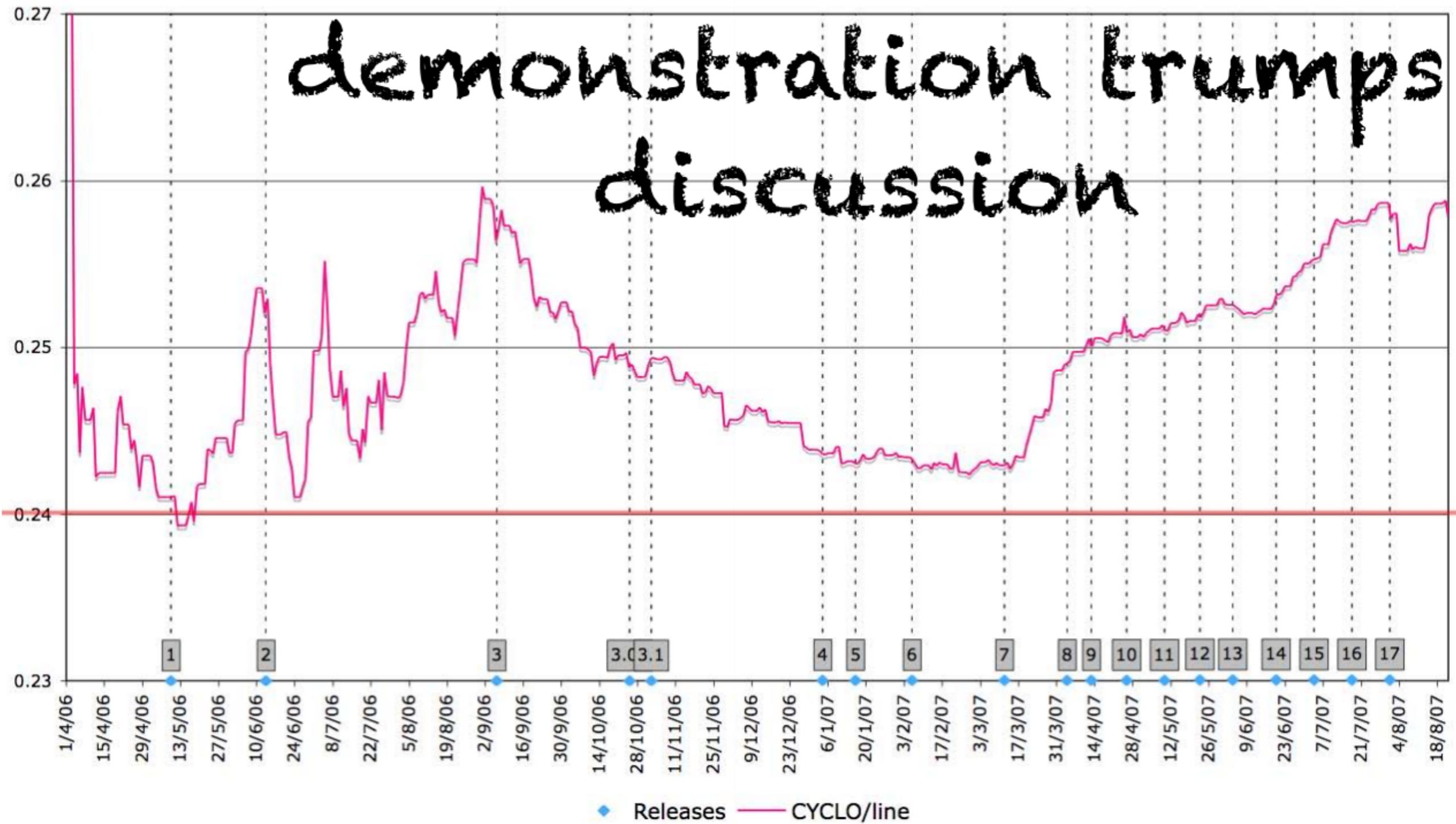
Navigation features to instantly [traverse your entire solution](#). You can jump to any file, type, or member in your code base in no time, or navigate from a specific symbol to its usages, base and derived symbols, or implementations.

### Safely change your code base

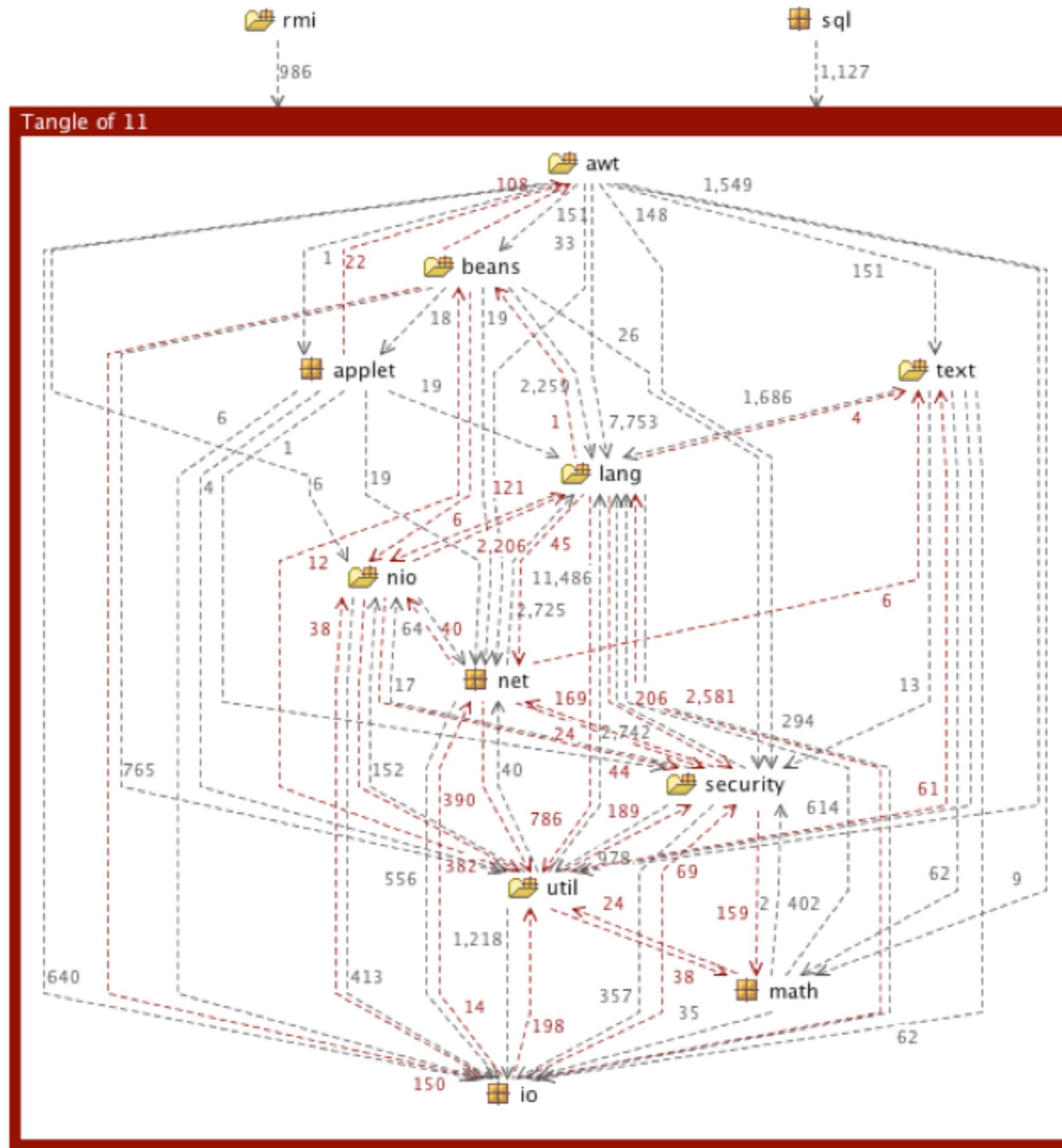
Enjoy solution-wide refactorings to safely change your code base. Whether you need to [revitalize legacy code](#) or put your project structure in order, you can lean on ReSharper.

ReSharper

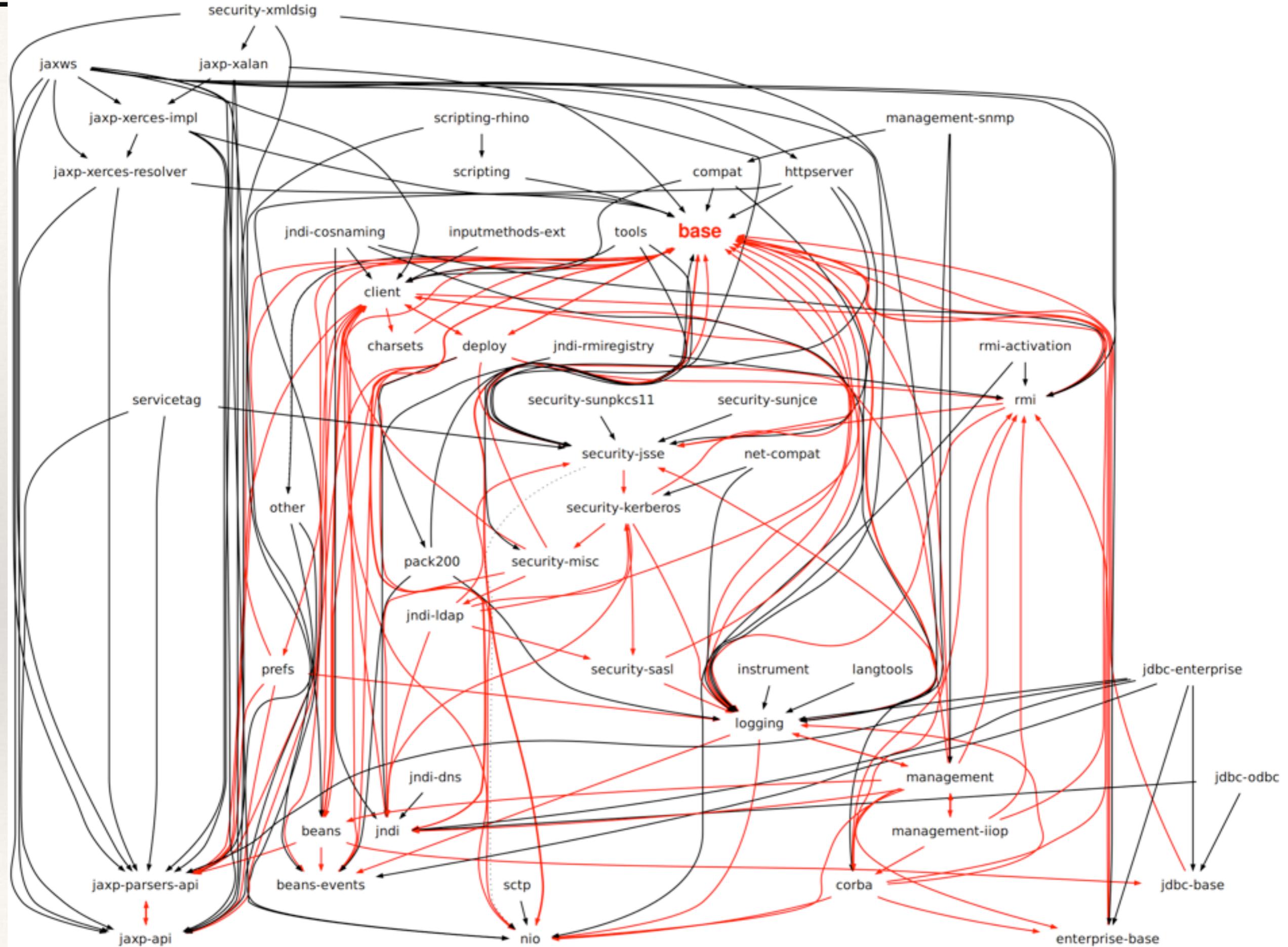
## Operational Complexity (branching point density)

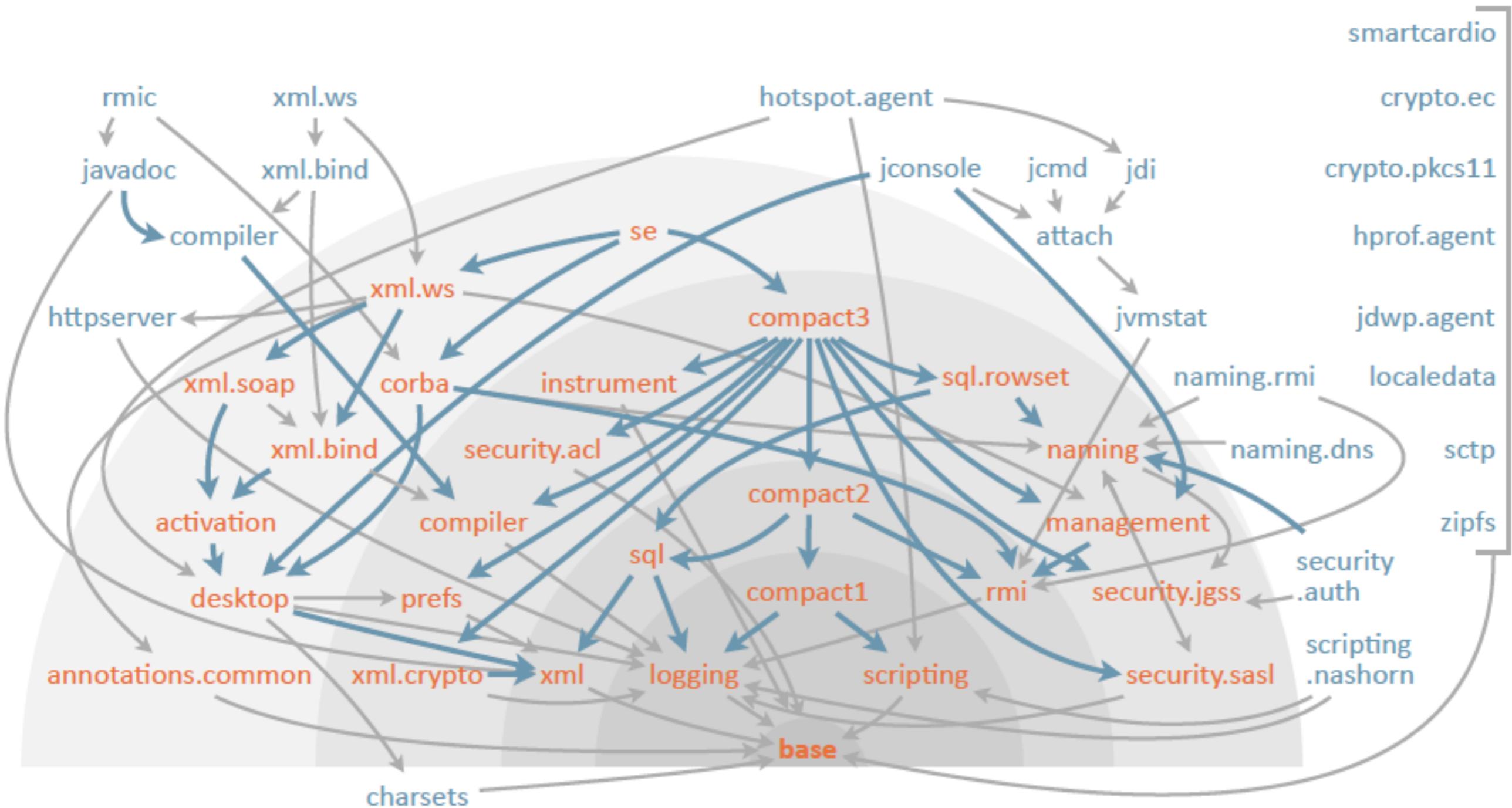


Source: Neal Ford, Emergent Design: [https://dl.dropbox.com/u/6806810/Emergent\\_Design%28Neal\\_Ford%29.pdf](https://dl.dropbox.com/u/6806810/Emergent_Design%28Neal_Ford%29.pdf)



# Tangles in JDK







# 1st International Workshop on Refactoring

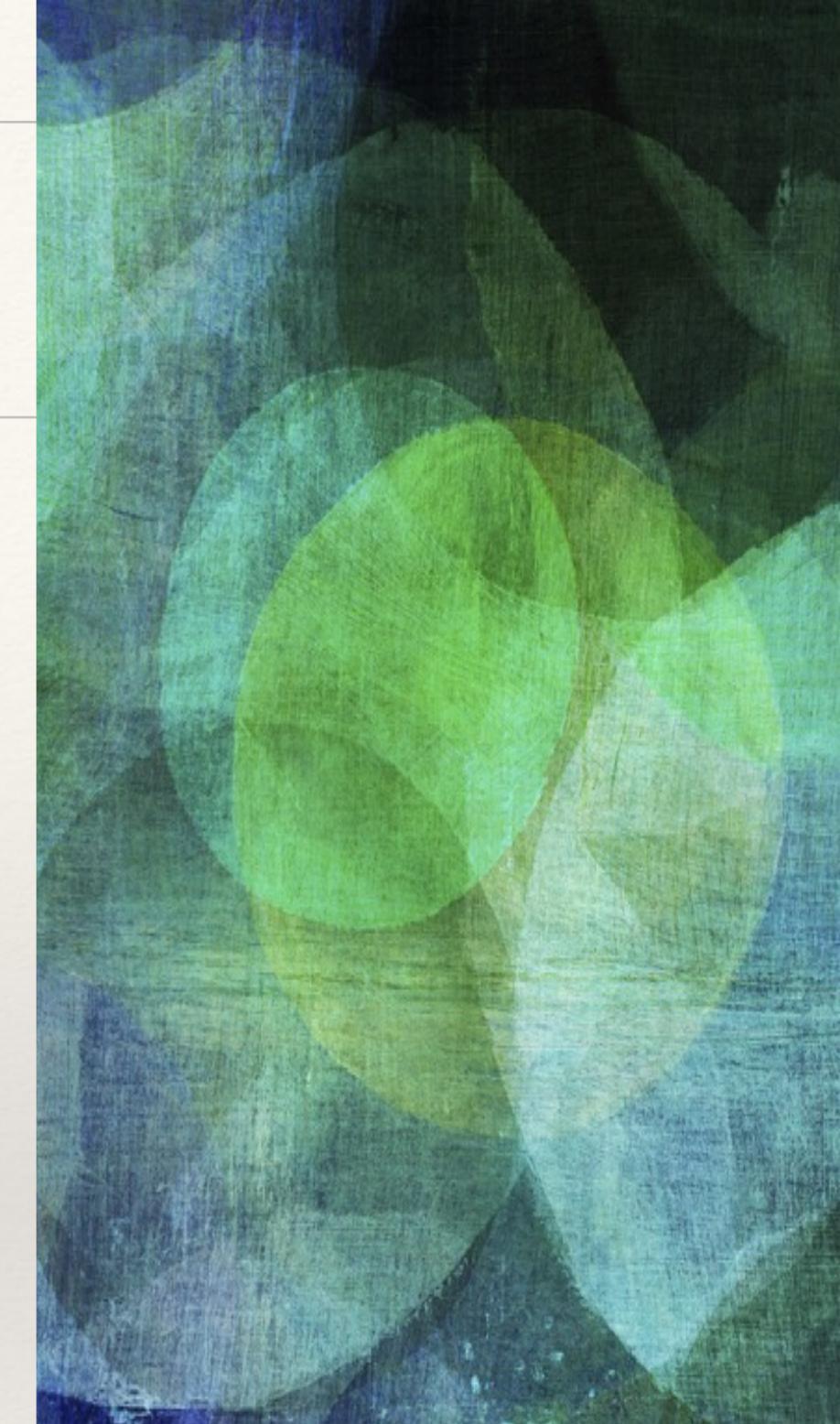
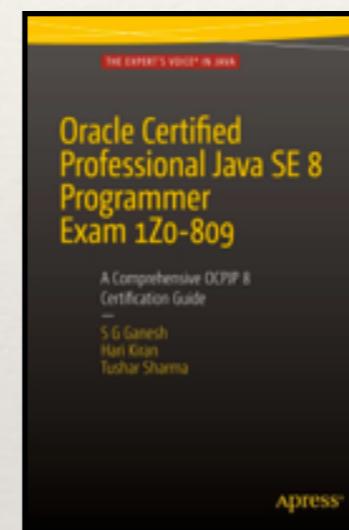
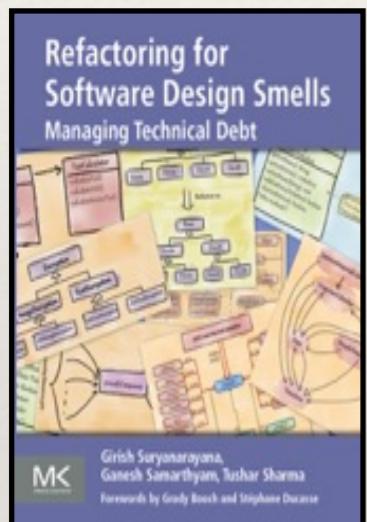
IWoR 2016

September 4, 2016, Singapore, Singapore

<http://www.softrefactoring.com/>

Refactoring for Software Architecture Smells

Ganesh Samarthyam, Tushar Sharma and Girish Suryanarayana



[ganesh@codeops.tech](mailto:ganesh@codeops.tech)

[www.codeops.tech](http://www.codeops.tech)

+91 98801 64463

[@GSamarthyam](https://twitter.com/GSamarthyam)

[slideshare.net/sgganesh](https://www.slideshare.net/sgganesh)

[bit.ly/ganeshsg](https://bit.ly/ganeshsg)