Code Craft

Nathaniel T. Schutta
@ntschutta

The Plan

- What is good code?
- Where do you start?
- Code Examples
- How do you fix it?
- How do you prevent it?

“Why do so many developers take the dark path... The path filled with complexity? Why is the simple path so evasive for so many?”

Ken Sipe

http://twitter.com/kensipe/status/6610942552

Which do you prefer?

Reading code?

Writing code?

“We should just throw it away and start over.”

“I’ll need to rewrite it.”

We need to read more code.

We need to talk about code.

We like this:

We get this:

http://www.flickr.com/photos/sgt_spanky/1513794 14/

Kevin has a camera

“Every programmer's horror: RT @BenAlabaster Great quote: "When I coded this, only God and I knew what I was doing, now only God knows."”

Kathy Sierra

http://twitter.com/KathySierra/status/6847734344

Twitter / Paul Warren: It seems I was a real geni ...

https://twitter.com/pawpoise/status/38010102002888704 Twitter, Inc.

html5 Gmail Saved Tabs Firebug Lite Dropbox GitHub pinboard read later

twitter

Login Join Twitter!

It seems I was a real genius 3 months ago when I wrote this code because I'm damned if I can understand it now

5:01 PM Feb 16th via Twitter for Mac
Retweeted by 4 people

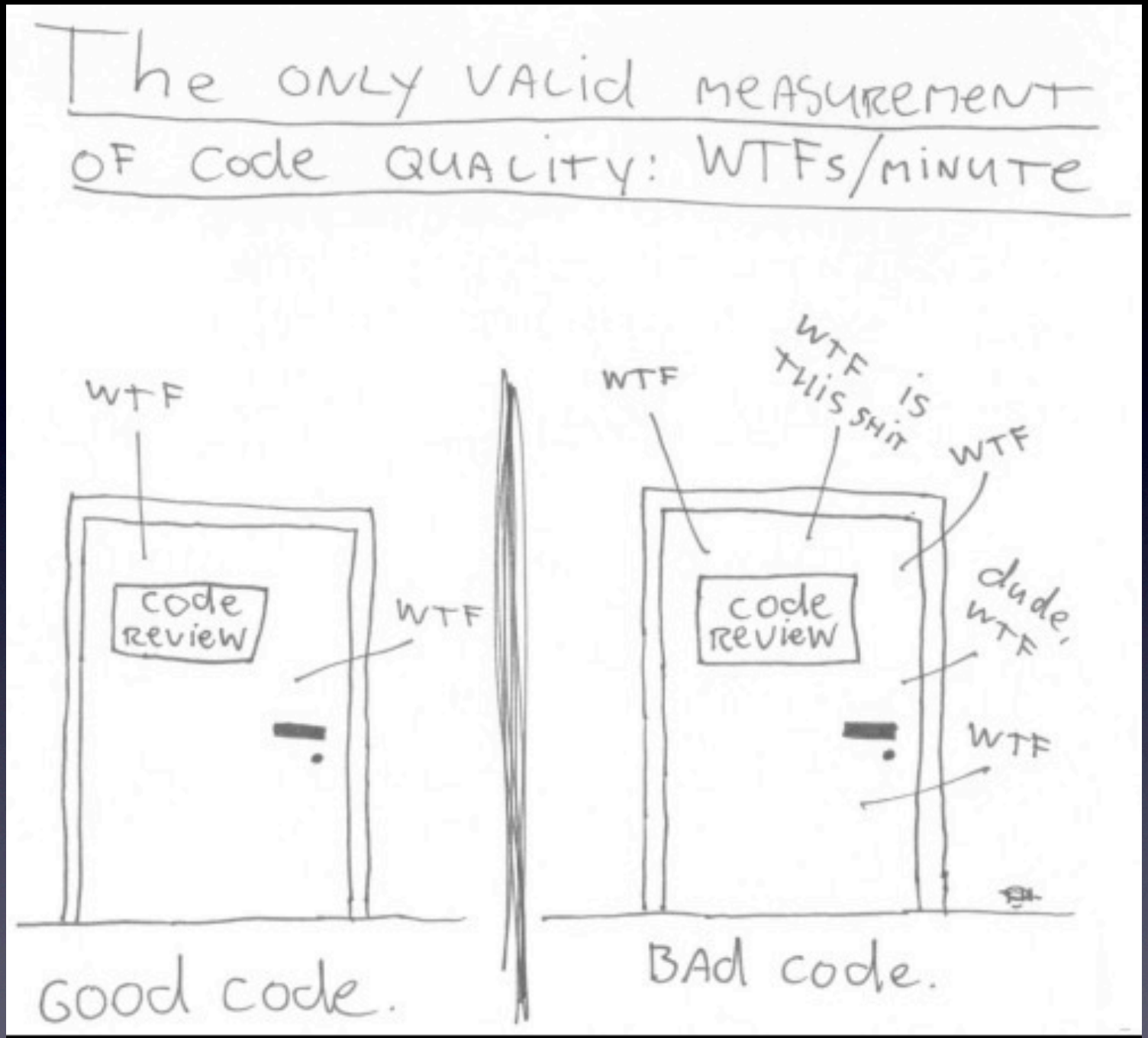
pawpoise
Paul Warren

© 2011 Twitter About Us Contact Blog Status Resources API Business Help Jobs Terms Privacy

https://twitter.com/pawpoise/status/38010102002888704

What is good code?

Subjective!



http://www.osnews.com/story/19266/WTFs_m

“Comment seen in code:
// TODO: WTF”

Michael Feathers

https://twitter.com/mfeathers/status/9290086320

We all know
bad code right?

WE don’t write bad code.

That *other* guy.

Maybe we can agree on what bad code is.

Change Risk Analysis and Predictions.

http://www.crap4j.org/

Cyclomatic complexity vs. code coverage.

If it’s simple, coverage can be low.

Doesn’t have to be though.

If it’s complicated, better have high coverage.

There is a ceiling on CC.

It’s not 83!

Back to good code.

Certain practices.

Less is more.

Small codebase = good.

http://steve-yegge.blogspot.com/2007/12/codes-worst-enemy.html

Boilerplate = bad.

Even when its generated!

Short classes.

Few pages.

Short methods.

< 10 lines.

Do one thing.

Say what you do.

One level of indent.

No else statements.

Twitter / Michael Feathers: I think keyboards should g ...

https://twitter.com/mfeathers/status/292509590757376

html5 Gmail Saved Tabs Firebug Lite Dropbox GitHub pinboard read later

twitter

Login Join Twitter!

I think keyboards should give us little electric shocks whenever we try to nest a condition in a condition.

3:05 PM Nov 4th, 2010 via Echofon
Retweeted by 22 people

mfeathers
Michael Feathers

© 2011 Twitter About Us Contact Blog Status Resources API Business Help Jobs Terms Privacy

https://twitter.com/mfeathers/status/292509590757376

Descriptive names.

Fine line!

How Big Should a Function Be?

http://blog.objectmentor.com/articles/2007/10/29/how-big-should-a-function-be

“Object Calisthenics” by Jeff Bay

http://binstock.blogspot.com/2008/04/perfecting-oos-small-classes-and-short.html

9 Rules.

Simplify, simplify, simplify.

Remove duplication,
even small amounts.

What about terseness?

Portrait of a Noob.

http://steve-yegge.blogspot.com/2008/02/portrait-of-n00b.html

Composition over inheritance.

Reuse is a byproduct,
not a rationale.

http://news.ycombinator.com/item?id=1620244

High cohesion...

Low coupling.

SOLID.

http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod

Single Responsibility.

Open Closed.

Liskov Substitution.

Dependency Inversion.

Interface Segregation.

Uncle Bob vs. Joel on Software...

http://www.hanselminutes.com/default.aspx?showID=163

http://blog.stackoverflow.com/2009/01/podcast-38/

http://blog.objectmentor.com/articles/2009/01/31/quality-doesnt-matter-that-much-jeff-and-joel

http://hanselminutes.com/default.aspx?showID=168

http://blog.objectmentor.com/articles/2009/02/12/getting-a-solid-start

Measuring Open/Closed.

Code should grow by addition not mutation.

http://michaelfeathers.typepad.com/michael_feathers_blog/2011/01/measuring-the-closure-of-code.html

Look at file
commit histories...

Look at the files that are modified the most.

Turbulence.

http://programmingtour.blogspot.com/2011/03/turbulence-measuring-turbulent-nature.html

Churn report, etc.

Currently Ruby only.

Why?

SLAP.

http://www.ibm.com/developerworks/java/library/j-eaed4.html

Single level of abstraction.

Don’t bounce around.

Extract method...

Easier to read.

Testing!

A must, not a best practice.

Continuous Integration.

http://martinfowler.com/articles/continuousIntegration.html

CruiseControl, Go,
Hudson, Jenkins, Anthill.

Feedback crucial.

Respect it!

What about Continuous Deployment?

http://timothyfitz.wordpress.com/2009/02/10/continuous-deployment-at-imvu-doing-the-impossible-fifty-times-a-day/

Don’t go Dark.

http://www.codinghorror.com/blog/archives/001134.html

Don’t wait for perfect.

Another set of eyes.

Code reviews.

Code reviews are vital.

http://weblog.raganwald.com/2008/01/whatever-happened-to-code-reviews.html

Be a jerk.

http://www.relevancellc.com/2008/1/29/no-be-a-jerk

About the code!

Formal or informal.

15 minutes.

Try pairing.

Crucible.

http://www.atlassian.com/software/crucible/

Doesn’t have to be hard...

Where do you start?

Certain things...stand out.

Code smells.

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

Certain shapes.

Indents!

Use of old idioms.

Poorly named methods, variables, etc.

WTF?

What does that do?

Comments!

PLEASE
HANDLE WITH CARE
FRAGILE
THANK YOU

“We don’t go there...”

Low lottery numbers.

“Only Bob can do that.”

Code analysis.

Code coverage.

EMMA, Cobetura, Clover.

Odds are, if there's no tests...there are bugs.

Metrics are your friend.

Clover Coverage Report - checkstyle

Coverage timestamp: Wed Oct 14 2009 06:13:34 CDT

Overview Package File

FRAMES NO FRAMES SHOW HELP

Statistics for project Clover database Wed Oct 14 2009 06:13:22 CDT:

Stmts:	7,207	**LOC:**	44,401	**Total cmp:**	3,997	**Stmts/Method:**	4.12
Branches:	2,728	**NCLOC:**	21,217	**Cmp density:**	0.55	**Methods/Class:**	5.45
Methods:	1,748	**Files:**	275	**Avg method cmp:**	2.29	**Classes/Pkg:**	16.05
Classes:	321	**Packages:**	20				

0.2% of code in this project is excluded from these metrics. Remove Filter

Coverage

321 classes, 9,671 / 11,683 elements

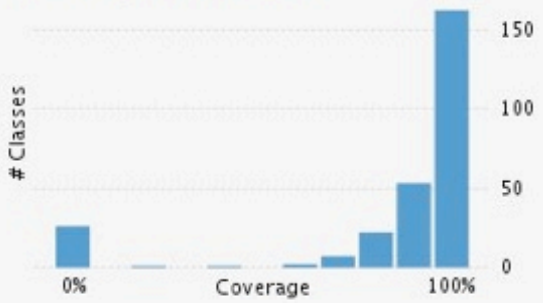
82.8%

Test Results

557 / 557 tests 7.71 secs

100%

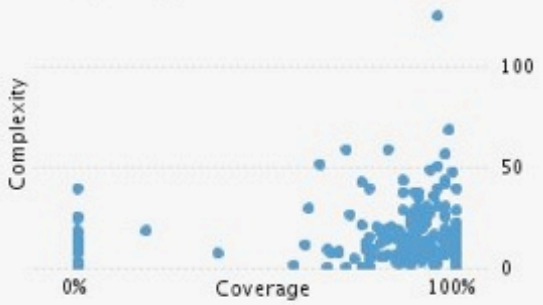
Class Coverage Distribution



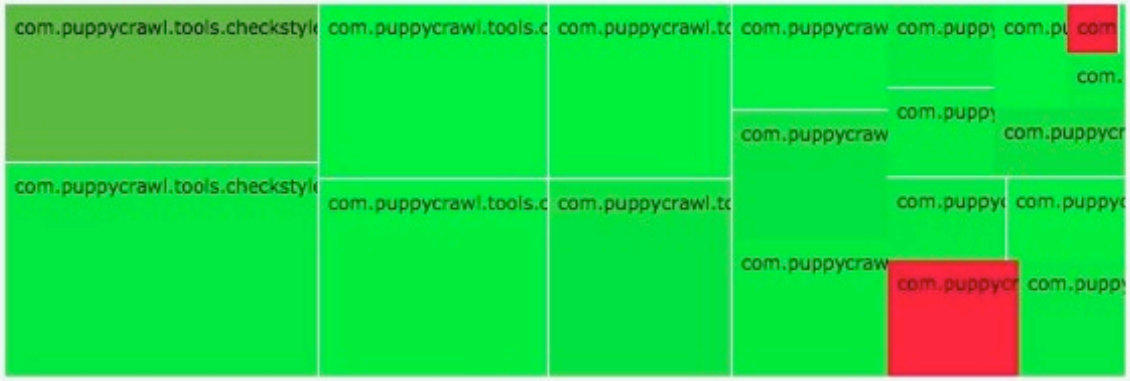
Top 20 Project Risks

Main ParameterAssignmentCheck ModifiedControlVariableCheck AutomaticBean.StrArrayConverter DesignForExtensionCheck SuppressElement PropertyCacheFile FileDrop.FileDropTargetListener JTreeTable.TreeTableCellEditor NPathComplexityCheck JTreeTable.ListToTreeSelectionModelWrapper ImportControlCheck CheckStyleTask.Formatter EqualsHashCodeCheck FileDrop TokenTypes LeftCurlyCheck ImportControlLoader SuppressionsLoader CheckDocsDoclet

Class Complexity



Coverage Tree Map


com.puppycrawl.tools.checkstyle
com.puppycrawl.tools.checkstyle
com.puppycrawl.tools.c
com.puppycrawl.tools.c
com.puppycrawl.tc
com.puppycrawl.tc
com.puppycraw
com.puppycraw
com.puppycraw
com.puppy
com.puppy
com.puppycr
com.puppycr
com.puppy
com.puppy
com.puppy
com.puppycr
com.puppy
com.puppy
com.puppy
com.puppycr
com.puppy
com.puppy
com.puppy

Clover Coverage Report - PMD-3.8

Coverage timestamp: Wed Oct 14 2009 06:14:43 CDT

Overview Package File

FRAMES NO FRAMES SHOW HELP

Statistics for project Clover database Wed Oct 14 2009 06:14:19 CDT:

Stmts:	24,051	**LOC:**	51,831	**Total cmp:**	10,890	**Stmts/Method:**	6.98
Branches:	8,968	**NCLOC:**	40,171	**Cmp density:**	0.45	**Methods/Class:**	6.41
Methods:	3,444	**Files:**	470	**Avg method cmp:**	3.16	**Classes/Pkg:**	12.2
Classes:	537	**Packages:**	44				

Coverage

537 classes, 19,621 / 36,463 elements

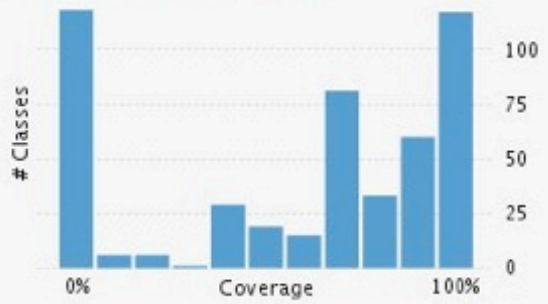
53.8%

Test Results

710 / 710 tests 5.39 secs

100%

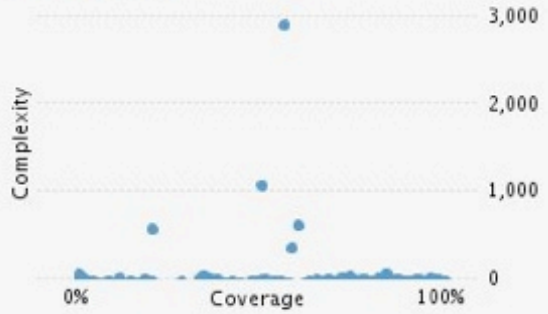
Class Coverage Distribution



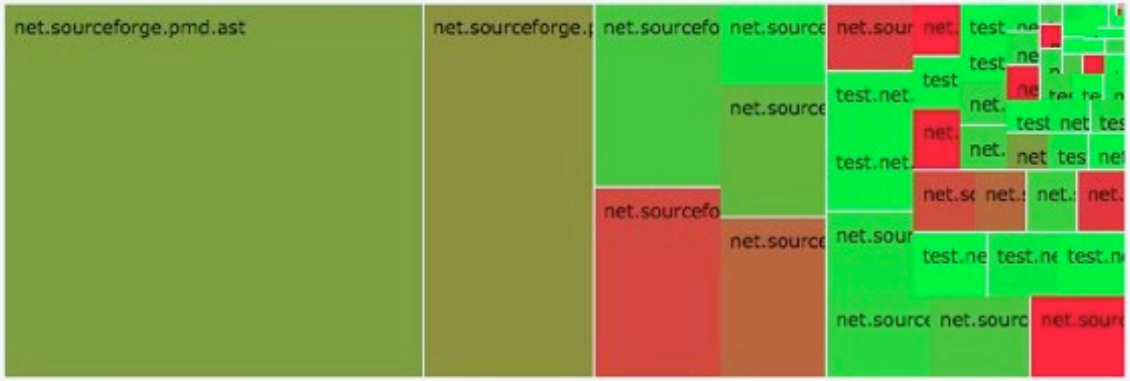
Top 20 Project Risks

Designer.XPathListener PHPTokenizer DaaRule PapariTextRenderer ParseException ParseException Designer.DFAListener Benchmark RubyTokenizer TextRenderer ReportHTMLPrintVisitor VBHTMLRenderer UselessAssignment StringConcatenationRule DFAPanel.DFACanvas CodeEditorTextPane TokenMgrError TokenMgrError TokenMgrError GUI.SaveListener

Class Complexity



Coverage Tree Map



Most Complex Packages

1. 57.1% net.sourceforge.pmd.ast (4554)
2. 51.4% net.sourceforge.pmd.jsp.ast (1745)
3. 20.6% net.sourceforge.pmd.cpd.cppast (736)
4. 75.3% net.sourceforge.pmd.rules (699)
5. 32.6% net.sourceforge.pmd.cpd (467)

Least Tested Methods

1. 0% JavaParser.AnnotationTypeMemberDeclaration() : void (47)
2. 0% CPPParserTokenManager.jjMoveStringLiteralDfa4_0(long,long,long,long) : int (29)
3. 0% JavaParser.CastLookahead() : void (23)
4. 0% JavaParserTokenManager.jjCanMove_1(int,int,int,long,long) : boolean (39)
5. 0% JavaParserTokenManager.jjCanMove_2(int,int,int,long,long) : boolean (39)
6. 0% CPPParserTokenManager.jjMoveStringLiteralDfa5_0(long,long,long,long) : int (25)
7. 0% JspParserTokenManager.jjMoveNfa_13(int,int) : int (12)
8. 0% VBHTMLRenderer.render(Report) : String (10)
9. 0% PMD.main(String[]) : void (12)
10. 0% DaaRule.execute(CurrentPath) : void (16)
11. 0% JspParser.generateParseException() : ParseException (12)
12. 0% DAAPathFinder.addNodeToTree() : void (10)

Most Complex Classes

What's your coverage?

Post it - visibility!

Key is change over time.

100% is the target.

http://emma.sourceforge.net/

http://cobertura.sourceforge.net/

http://www.atlassian.com/software/clover/

Coverage isn’t everything.

Asserts are important ;)

Jester - change code, tests should fail.

http://jester.sourceforge.net/

PMD.

Scans code
looking for bugs.

Dead code, empty blocks, complication.

Copy/paste detector.

Simian.

Similarity analyzer.

Duplicate code finder.

http://www.redhillconsulting.com.au/products/simian/

FindBugs.

Based on idea of bug patterns.

Looks for common problems.

Works on the bytecode.

http://findbugs.sourceforge.net/

Listen to the compiler.

Don’t ignore warnings.

Search the code base.

Write tests!

Testing Will Challenge Your Conventions

http://blog.objectmentor.com/articles/2007/07/17/testing-will-challenge-your-conventions

Decipher once.

Best docu you can buy.

Tests = production code.

Treat them that way.

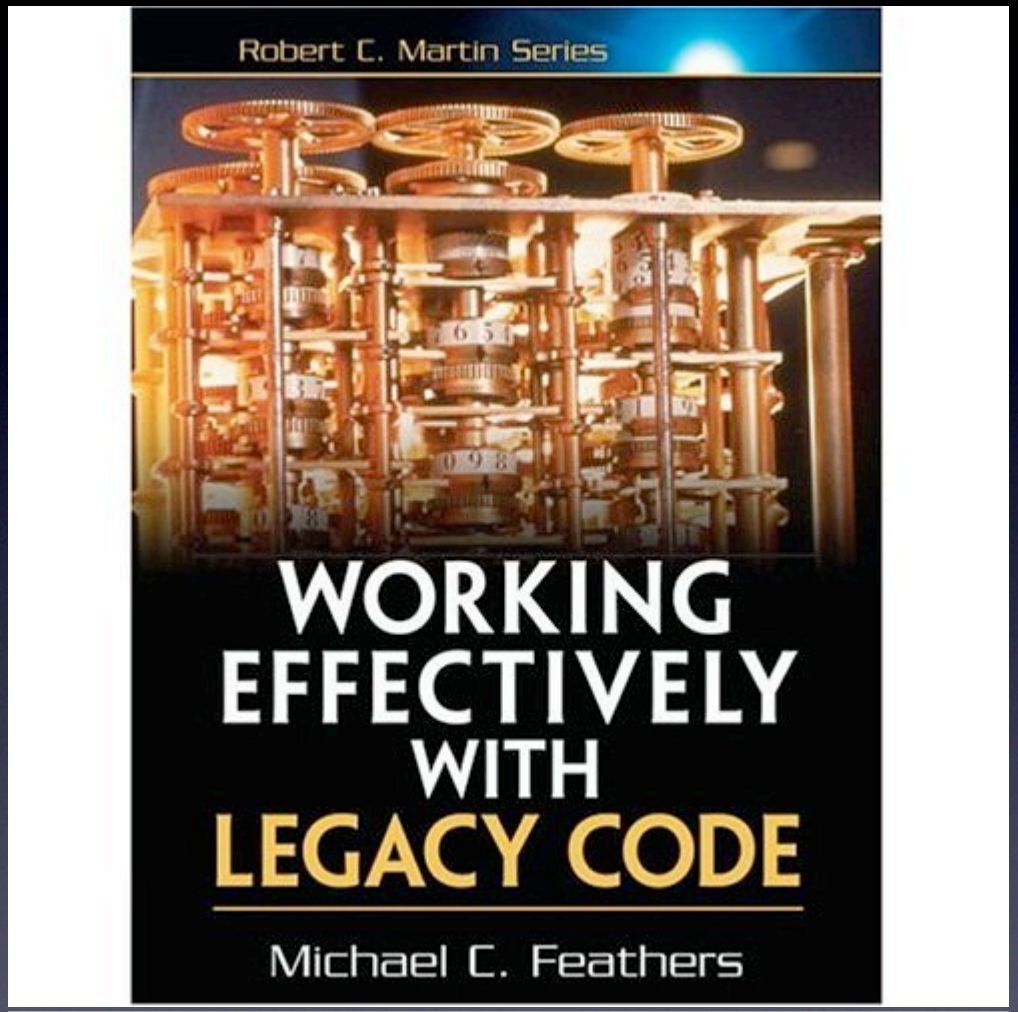
Look for injection points.

Override getters.

Look for seams.

Injection points.

Isolate!


Robert C. Martin Series
WORKING EFFECTIVELY WITH LEGACY CODE
Michael C. Feathers

Access modifiers are more of a suggestion.

Explore alternatives...

Groovy!

Break encapsulation.

Testability is worth it.

Code!

“O. M. FSM. Just found this in code (not mine....reallly!):
If data.Count > 0 Then
Assert.IsNotNull(data)
#SMASHFACE #SMASHFACE”

Jason Bock

http://twitter.com/jasonbock/status/9199855735

```
if (debit_credit_cd == 'C') {
    amt = amt * -1;
} else {
    amt = amt * 1;
}
```

```
private static final Map<Class<? extends BaseEntity>, Class<?
            dataModelRegistry = new HashMap<Class<? extends
}
```

```java
private String getDescription(AbstractCacheFactory<?> factory, Object key) {
  try {
    if (factory instanceof CountryFactory) {
        return ((CountryFactory) factory).findByKey(key).getCountryDescription();
    } else if (factory instanceof OnDepositFactory) {
        return ((OnDepositFactory) factory).findByKey(key).getDescription();
    } else if (factory instanceof StateProvinceFactory) {
        return ((StateProvinceFactory) factory).findByKey(key).getDescription();
    } else if (factory instanceof BRFactory) {
        return ((BRFactory) factory).findByKey(key).getDescription();
    } else if (factory instanceof StrategyFactory) {
        return ((StrategyFactory) factory).findByKey(key).getDescription();
    } else if (factory instanceof VendorFactory) {
        return ((VendorFactory) factory).findByKey(key).getDescription();
    } else if (factory instanceof CodeFactory) {
        return ((CodeFactory) factory).findByKey(key).getDescription();
    } else if (factory instanceof PortfolioFactory) {
        return ((PortfolioFactory) factory).findByKey(key).getDescription();
    }
    return ViewKeysConstants.NON_BREAKING_SPACE;
  } catch (Exception e) {
      return ViewKeysConstants.NON_BREAKING_SPACE;
  }
}
```

```
if (className == foo) {
  do something
} else if (className == bar) {
  do something else
} else if {
  ....
(repeat 10-15 more times...)
```

```
try {
  ...
} catch (AppException ae) {
  log "caught exception" + ae
  throw (ae)
}
```

```
try {
    date = dateFormatter.parse(inDate);
} catch (ParseException e) {
    throw new ParseException(e.toString(), 0);
}
```

```
} catch (FooFileWriterException e) {
  System.out.println("catch is c");
  assertTrue(1==1);
}
```

```
@Test
public final void testConvert() {
  // fail("Not yet implemented"); // TODO
}
```

```
boolean response = true;
assertEquals(true, response);
```

```
private void removeTimestamp(SOAPMessageContext smc) {
try {
  // get the header
  SOAPHeader hdr = smc.getMessage().getSOAPPart().getEnvelope().getHeader();
  if(hdr != null)
  {
    // iterate through the header list
    for(Iterator headers = hdr.examineAllHeaderElements(); headers.hasNext();)
    {
      SOAPHeaderElement header = (SOAPHeaderElement)headers.next();
      Name parentElementName = header.getElementName();

      // if working with the security header
      if(parentElementName.getLocalName().equals("Security"))
      {
        // iterate through security nodes
        for(Iterator headerIter = header.getChildElements(); headerIter.hasNext();)
        {
          Object elementObj = headerIter.next();
          if(elementObj instanceof SOAPElement)
          {
            SOAPElement element = (SOAPElement)elementObj;
            if(element.getElementName().getLocalName().equals("Timestamp"))
            {
               SOAPElement securityElement = element.getParentElement();
               securityElement.removeChild(element);
            }
          }
        }
      }
    }
  }
}
```

```java
if (StringUtils.isBlank(fieldOne)) {

  ExceptionBean bean = new ExceptionBean();
  bean.setErrorCode(103);
  bean.setMessage(Constants.FIELD_ONE_NO_VALUE);
  listExceptionBean.getExceptionBean().add(bean);

  response = false;
}
if (StringUtils.isBlank(fieldTwo)) {

  ExceptionBean bean = new ExceptionBean();
  bean.setErrorCode(104);
  bean.setMessage(Constants.FIELD_TWO_NO_VALUE);
  listExceptionBean.getExceptionBean().add(bean);

  response = false;
}
if (StringUtils.isBlank(fieldThree)) {

  ExceptionBean bean = new ExceptionBean();
  bean.setErrorCode(105);
  bean.setMessage(PrismdrConstants.FIELD_THREE_NO_VALUE);
  listExceptionBean.getExceptionBean().add(bean);

  response = false;
}
```

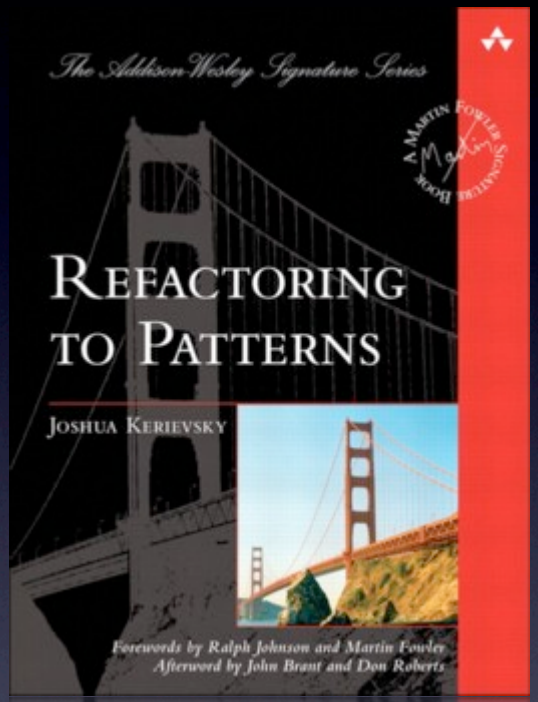
```
if (StringUtils.isNotBlank(foo)) {
  if (StringUtils.isBlank(f1) && StringUtils.isBlank(f2)
      && StringUtils.isBlank(f3) && StringUtils.isBlank(f4)
      && StringUtils.isBlank(f5) && StringUtils.isBlank(f6)
      && StringUtils.isBlank(f7) && StringUtils.isBlank(f8)
      && StringUtils.isBlank(f9) && StringUtils.isBlank(f10)
      && StringUtils.isBlank(f11) && StringUtils.isBlank(f12)
      && StringUtils.isBlank(f13) && StringUtils.isBlank(f14)
      && StringUtils.isBlank(f15)) {
      throw new FooException(Constants.INVALID_SITUATION_ONE,
            new FooExceptionBean());
  }
}

if (StringUtils.isNotBlank(bar)) {
  if (StringUtils.isBlank(f1) && StringUtils.isBlank(f2)
      && StringUtils.isBlank(f3) && StringUtils.isBlank(f4)
      && StringUtils.isBlank(f5) && StringUtils.isBlank(f6)
      && StringUtils.isBlank(f7) && StringUtils.isBlank(f8)
      && StringUtils.isBlank(f9) && StringUtils.isBlank(f10)
      && StringUtils.isBlank(f11) && StringUtils.isBlank(f12)
      && StringUtils.isBlank(f13) && StringUtils.isBlank(f14)
      && StringUtils.isBlank(f15)) {
      throw new FooException(Constants.INVALID_SITUATION_ONE,
              new FooExceptionBean());
  }
}
```

Fixing things.

Refactor.

Never going to get it right the first time.

Iterate, iterate, iterate.

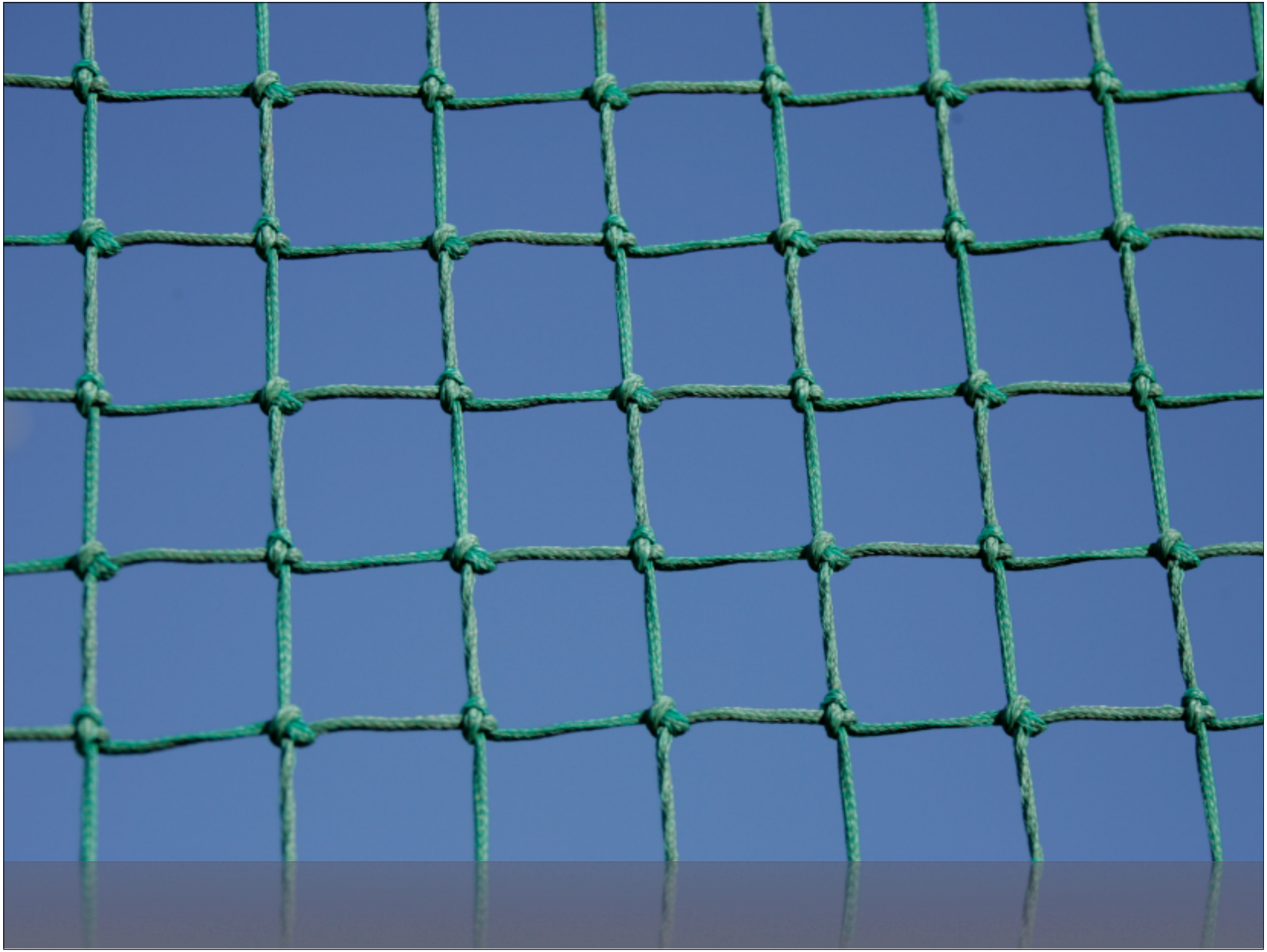


http://www.industriallogic.com/xp/refactoring/

Use your tools.

You’ve got tests right?

Sometimes...



Stressful.

Takes knowledge...

or courage.

Start small.

Crawl before you run.

Establish some success.

Technical debt.

Pay it down over time.

Tools help.

CI - vital.

Rapid feedback.

Miss the mark?
Fail the build.

JUnit, TestNG, whatever works for you!

Change Risk Analysis and Predictions.

http://www.crap4j.org/

FindBugs.

http://findbugs.sourceforge.net/

PMD.

http://pmd.sourceforge.net/

CPD.

Compiler settings.

Respect it!

How about a new language?

Shift the culture.

Culture kills.

All the best practices...

don’t mean much
when ignored.

“Don't turn off the fire alarm.
Deal with the fire.”

David Bock

Educate.

Brown bags.

Conferences.

Book clubs.

Whatever it takes.

Be critical of code,
not people.

Expectations matter.

We will not ship shit.

http://www.artima.com/weblogs/viewpost.jsp?thread=7588

Shame works.

	Build Failures	Take Down the Build BOX!
Ben		
Mark	1	1
Jason		
Cindy		
Alissa		
Ryan		

People can be sensitive.

If you can’t critique...

Terrible programmers.

http://blog.kickin-the-darkness.com/2007/09/confessions-of-terrible-programmer.html

You should improve over time...

"ON MY HONOR
I WILL DO MY BEST..."
BOY SCOUTS
OF AMERICA
3¢
UNITED STATES POSTAGE

“Don't put it off. Always leave any code file you touch a little cleaner than you found it. It'll make you a happier developer.”

Ehren Murdick

https://twitter.com/rubyprogrammer/status/9287115231

Not sure what to do?

Complexity kills.

Who needs hackers?

http://www.nytimes.com/2007/09/12/technology/techspecial/12threat.html

“Anyone can make the simple complicated. Creativity is making the complicated simple.”

Charles Mingus

Software is full of complexity.

Two flavors.

Essential complexity.

What we do is hard!

Domains are challenging.

Accidental complexity.

Ways we make it harder.

“Noisy” technologies.

http://memeagora.blogspot.com/2007/09/ruby-matters-meta-programming-synthesis.html

http://weblog.raganwald.com/2007/07/abbreviation-accidental-complexity-and.html

Minimize accidental complexity.

Duh.

Essence vs. ceremony.

http://www.relevancellc.com/2008/1/17/slides-posted-ending-legacy-code

Beware the rewrite.

Remember Netscape?

http://www.joelonsoftware.com/articles/fog0000000069.html

Lots of knowledge in legacy code...

Step carefully.

Sometimes...use the delete key.

Prevention.

We don’t work alone.

People come and go.

Onboarding.

One time setup...

How do I?

How do we pass knowledge?

How do we
share our values?

Tough problem.

Knowledge Exchange.

Many teams have nothing.

Or, out of date docu.

Many systems
are too formal.

Require “admins”
and approval.

Low ceremony trumps formal systems.

Wikis are ideal.

SharePoint works well too.

Guides are good.

Keep them up to date!

Version control!

Ask new people to update.

Developer handbook.

Setup IDE, workspace.

Printers, servers, SCM.

Code standards.

Team contacts, calendar.

Distribution lists, standups.

Whatever adds value.

Glossary.

Simple and very valuable.

Projects are full of jargon.

“Part of the benefit of being "into" something is having an insider lexicon.”

Kathy Sierra
Creating Passionate Users

http://headrush.typepad.com/creating_passionate_users/2006/11/why_web_20_is_m.html

That’s great...when you know it.

Many people won’t ask.

Just stay in the dark.

Let’s do better.

Various forms.

Flip charts, Excel, plain text.

Whatever works best for your team.

Keep it up to date!

Versioned.

Actively pass on knowledge.

Consider using videos, screencasts, podcasts.

Video time capsules.

http://codetojoy.blogspot.com/2011/01/software-archeology-and-video-time.html

Standups, pairing.

Coffee run.

Values matter.

Post them.

“New Rule.”

Questions?!?

Thanks!

Please complete your surveys.