

emergent design



NEAL FORD software architect / meme wrangler

ThoughtWorks®

nford@thoughtworks.com
3003 Summit Boulevard, Atlanta, GA 30319
www.nealford.com
www.thoughtworks.com
blog: memeagora.blogspot.com
twitter: neal4d

Evolutionary architecture and emergent design: Investigating architecture and design

Discovering more-maintainable design and architecture

Level: Intermediate

[Neal Ford](#) (nford@thoughtworks.com), Software Architect / Meme Wrangler, ThoughtWorks Inc.

24 Feb 2009

Software architecture and design generate a lot of conversational heat but not much light. To start a new conversation about alternative ways to think about them, this article launches the [Evolutionary architecture and emergent design](#) series. Evolutionary architecture and emergent design are agile techniques for deferring important decisions until the last responsible moment. In this introductory installment, series author Neal Ford defines architecture and design and then identifies overarching concerns that will arise throughout the series.

Architecture and design in software have resisted firm definitions for a long time because software development as a discipline has not yet fully grasped all their intricacies and implications. But to create reasonable discourse about these topics, you have to start somewhere. This article series concerns evolutionary architecture and emergent design, so it makes sense to start the series with some definitions, considerations, and other ground-setting.

Defining architecture

Architecture in software is one of the most talked about yet least understood concepts that developers grapple with. At conferences, talks and birds-of-a-feather gatherings about architecture pack the house, but we still have only vague definitions for it. When we discuss architecture, we're really talking about several different but related concerns that generally fall into the broad categories of *application architecture* and *enterprise architecture*.

About this series

This [series](#) aims to provide a fresh perspective on the often-discussed but elusive concepts of software architecture and design. Through concrete examples, Neal Ford gives you a solid grounding in the agile practices of *evolutionary architecture* and *emergent design*. By deferring important architectural and design decisions until the last responsible moment, you can prevent unnecessary complexity from undermining your software projects.

www.ibm.com/developerworks/java/library/j-eaed1/index.html?S_TACT=105AGX02&S_CMP=EDU

bit.ly/nf-ead-all

agenda

what is software design

architecture vs. design

obfuscators

enablers

harvesting patterns



“There are known
unknowns.

That is to say
there are things
that we now know
we don't know.

But there are
also unknown
unknowns.

There are things we do not
know we don't know.”

There are things we do not
know we don't know."

big up-front
architecture
& design
fail
because
of unknown
unknowns



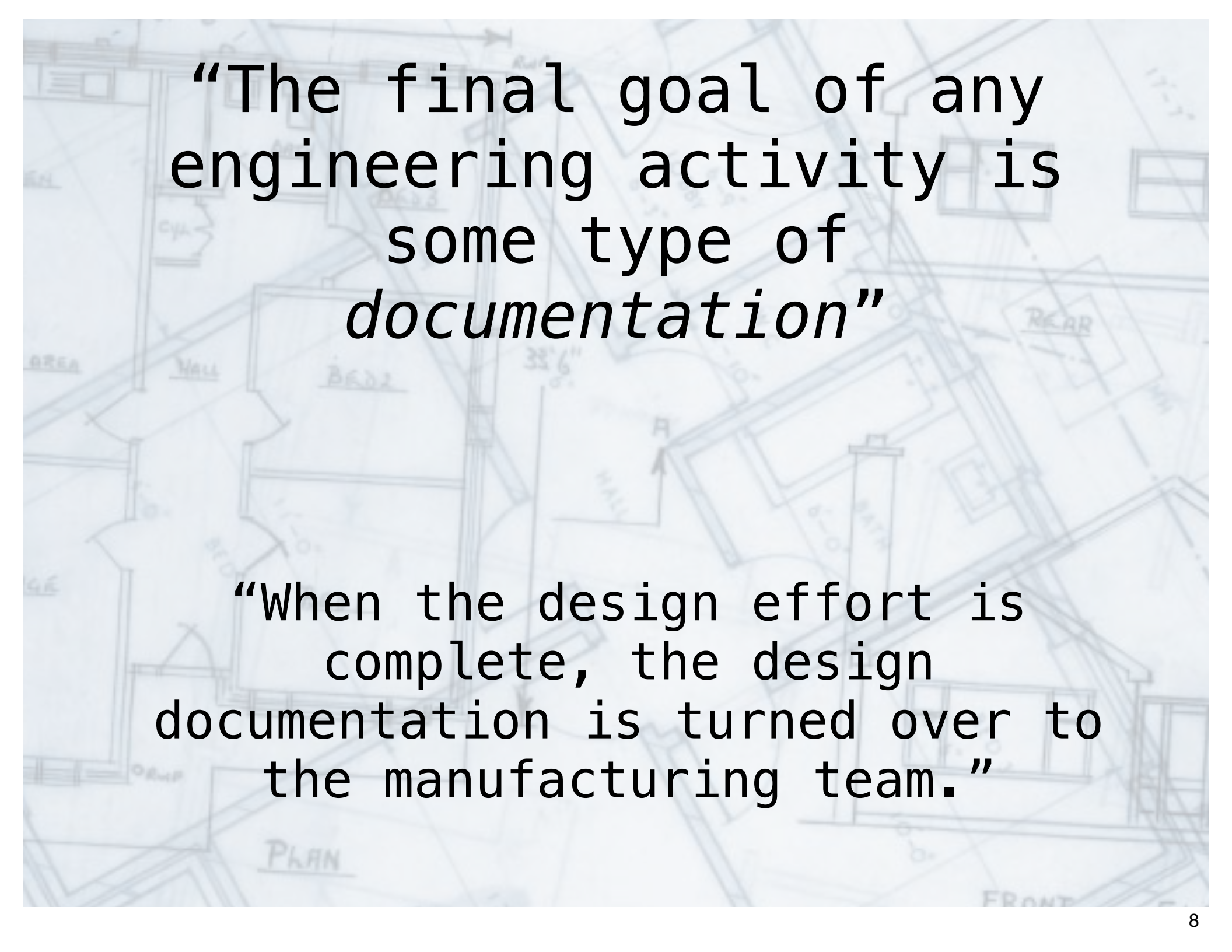
the future is
hard to predict!

“what is software design?”

Jack C. Reeves
fall 1992, c++ journal

http://www.developerdotstar.com/mag/articles/reeves_design.html

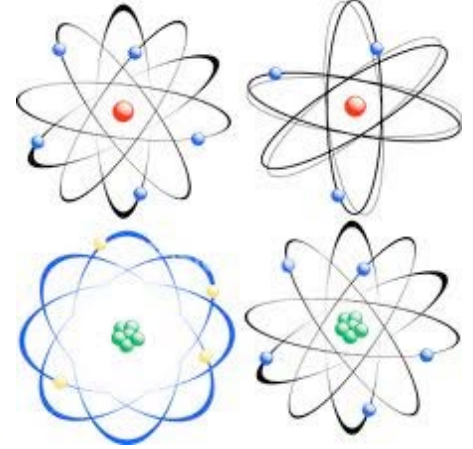
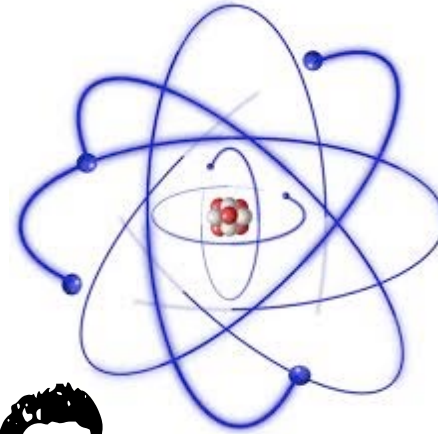
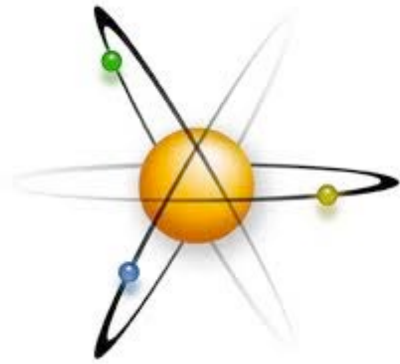
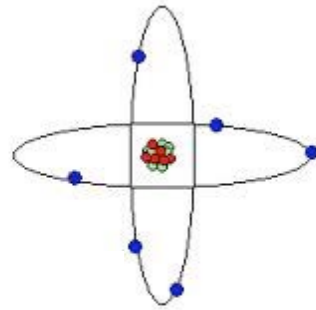
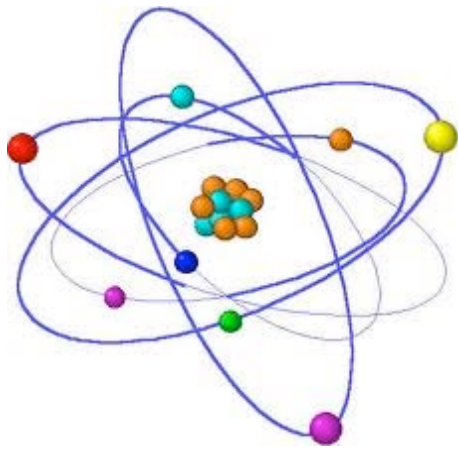


The background of the slide is a faded, light blue architectural drawing of a building floor plan. It shows various rooms, corridors, and structural elements. Labels like 'BED', 'HALL', 'BATH', 'REAR', and 'FRONT' are visible, along with dimensions such as '33'6"'.


“The final goal of any engineering activity is some type of *documentation*”

“When the design effort is complete, the design documentation is turned over to the manufacturing team.”

manufacturing for
physical things



VS



software
manufacturing

compilation

```

if (doubleListValue != null) {
    if (altSyntax()) {
        // the same logic as with findValue(String)
        // if value start with %{ and end with }, just cut it off!
        if (doubleListValue.startsWith("%{") && doubleListValue.endsWith("}")) {
            doubleListValue = doubleListValue.substring(2, doubleListValue.length() - 1);
        }
    }

    addParameter("doubleListValue", doubleListValue);
} else if (tmpDoubleList instanceof Map) {
    addParameter("doubleListValue", "value");
}

if (formName != null) {
    addParameter("formName", findString(formName));
} else {
    // ok, let's look it up
    Component form = findAncestor(Form.class);
    if (form != null) {
        addParameter("formName", form.getParameters().get("name"));
    }
}

Class valueClazz = getValueClassType();

if (valueClazz != null) {
    if (doubleValue != null) {
        addParameter("doubleNameValue", findValue(doubleValue, valueClazz));
    } else if (doubleName != null) {
        addParameter("doubleNameValue", findValue(doubleName.toString(), valueClazz));
    }
} else {
    if (doubleValue != null) {
        addParameter("doubleNameValue", findValue(doubleValue));
    } else if (doubleName != null) {
        addParameter("doubleNameValue", findValue(doubleName.toString()));
    }
}

```

design ==
complete source code

```

if (doubleListValue != null) {
    if (altSyntax()) {
        // the same logic as with findValue(String)
        // if value start with %{ and end with }, just cut it off!
        if (doubleListValue.startsWith("%{") && doubleListValue.endsWith("}")) {
            doubleListValue = doubleListValue.substring(2, doubleListValue.length() - 1);
        }
    }

    addParameter("doubleListValue", doubleListValue);
} else if (tmpDoubleList instanceof Map) {
    addParameter("doubleListValue", "value");
}

if (formName != null) {
    addParameter("formName", findString(formName));
} else {
    // ok, let's look it up
    Component form = findAncestor(Form.class);
    if (form != null) {
        addParameter("formName", form.getParameters().get("name"));
    }
}

Class valueClazz = getValueClassType();

if (valueClazz != null) {
    if (doubleValue != null) {
        addParameter("doubleNameValue", findValue(doubleValue, valueClazz));
    } else if (doubleName != null) {
        addParameter("doubleNameValue", findValue(doubleName.toString(), valueClazz));
    }
} else {
    if (doubleValue != null) {
        addParameter("doubleNameValue", findValue(doubleValue));
    } else if (doubleName != null) {
        addParameter("doubleNameValue", findValue(doubleName.toString()));
    }
}

```

software \$\$\$ traditional

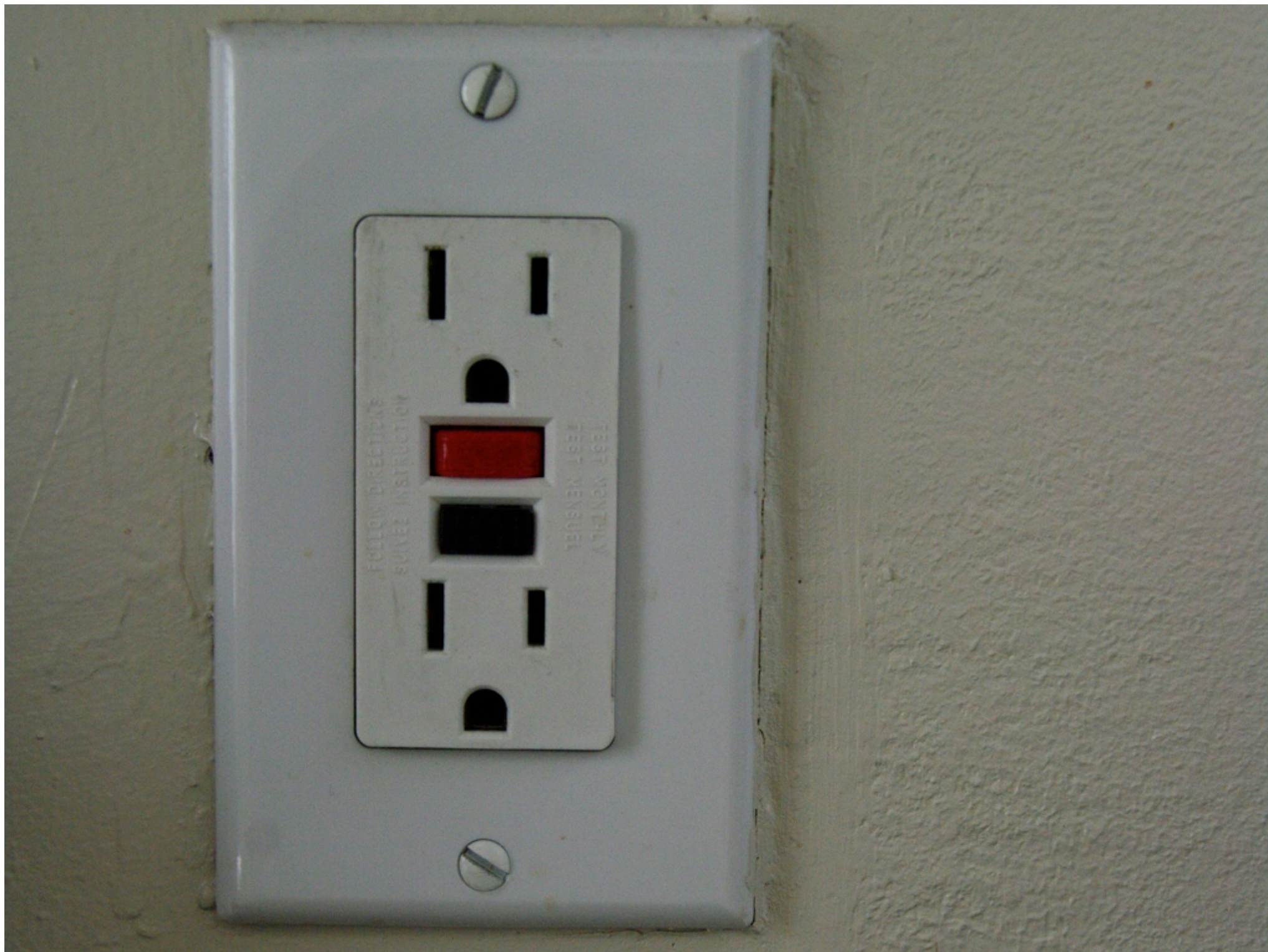
“Given that

software designs are
relatively easy to turn out

and essentially free to build,

an unsurprising revelation is
that software designs tend to be
incredibly large and complex.”

Jack Reeves





$$\int_0^{\pi/2} \langle 3\cos t, 6t, 6\sin t \rangle \langle -\sin t, \cos t, 1 \rangle dt \quad \iint \langle 8a \sin^3 \phi \cos \phi, -8a \sin \phi \cos \phi, 1 \rangle$$

$$\int_0^{\pi/2} -3\cos t \sin t + 6t \cos t + 6 \sin t dt$$

$$\sqrt{4x^2 + 16 + 4} = 2\sqrt{x^2 + 5}$$

$$\nabla f = \langle 2x, -4, -2 \rangle$$

$$\int_0^1 \langle u, \frac{6\pi}{2}(1-t), 6 \rangle \langle 0, -1, t - \pi/2 t \rangle dt$$

$$\frac{y^2}{4} = 1 \int_0^1 3\pi t dt \quad \begin{matrix} 8a^3 \sin^3 \phi \sin \theta \cos \theta, \\ 48a^3 \sin^3 \phi \sin \theta \cos \theta, \\ a^2 \sin \phi \cos \phi \end{matrix}$$

$$\int_0^1 \langle 3t, 0, 6-6t \rangle \langle 1, -1, 0 \rangle dt$$

$$\int_0^3 \int_0^{\sqrt{x^2+5}} \frac{3x - 2\sqrt{x^2+5}}{-2} dy dx$$

$$= \int_0^3 \int_0^{\sqrt{x^2+5}} -\sqrt{x^2+5} dy dx$$

$$-\int_0^1 3t dt$$

$$\langle e^{2y+5z}, 6xe^{2y+5z}, 15xe^{2y+5z} \rangle$$

$$\iint_D = 5 + 5 dy dx$$

$$f = \frac{e^{2y+5z}}{2y+5z} + \iint r = \langle r \sin \theta, r \cos \theta, r \rangle$$

$$\int_0^4 \int_0^4 8+7 dy dx$$

$$\iiint_R \langle 8y, -8x, 1 \rangle \cdot \frac{1}{r^2} \cdot r dr d\theta dz \quad r \in [0, 1]$$

$$15 = 16$$

$$x^2 + y^2 + z^2$$

$$r \cos \theta \sin \theta \cdot r$$

```

@Test public void test_a_bunch_of_numbers() {
    Set<Integer> expected = new HashSet<Integer>(
        Arrays.asList(PERFECT_NUMS));
    for (int i = 2; i < 33550340; i++) {
        if (expected.contains(i))
            assertTrue(classifierFor(i).isPerfect());
        else
            assertFalse(classifierFor(i).isPerfect());
    }
}

```

testing = engineering rigor in software

```

@Test(expected = InvalidNumberException.class)
public void cannot_classify_negative_numbers() {
    new Classifier6(-20);
}

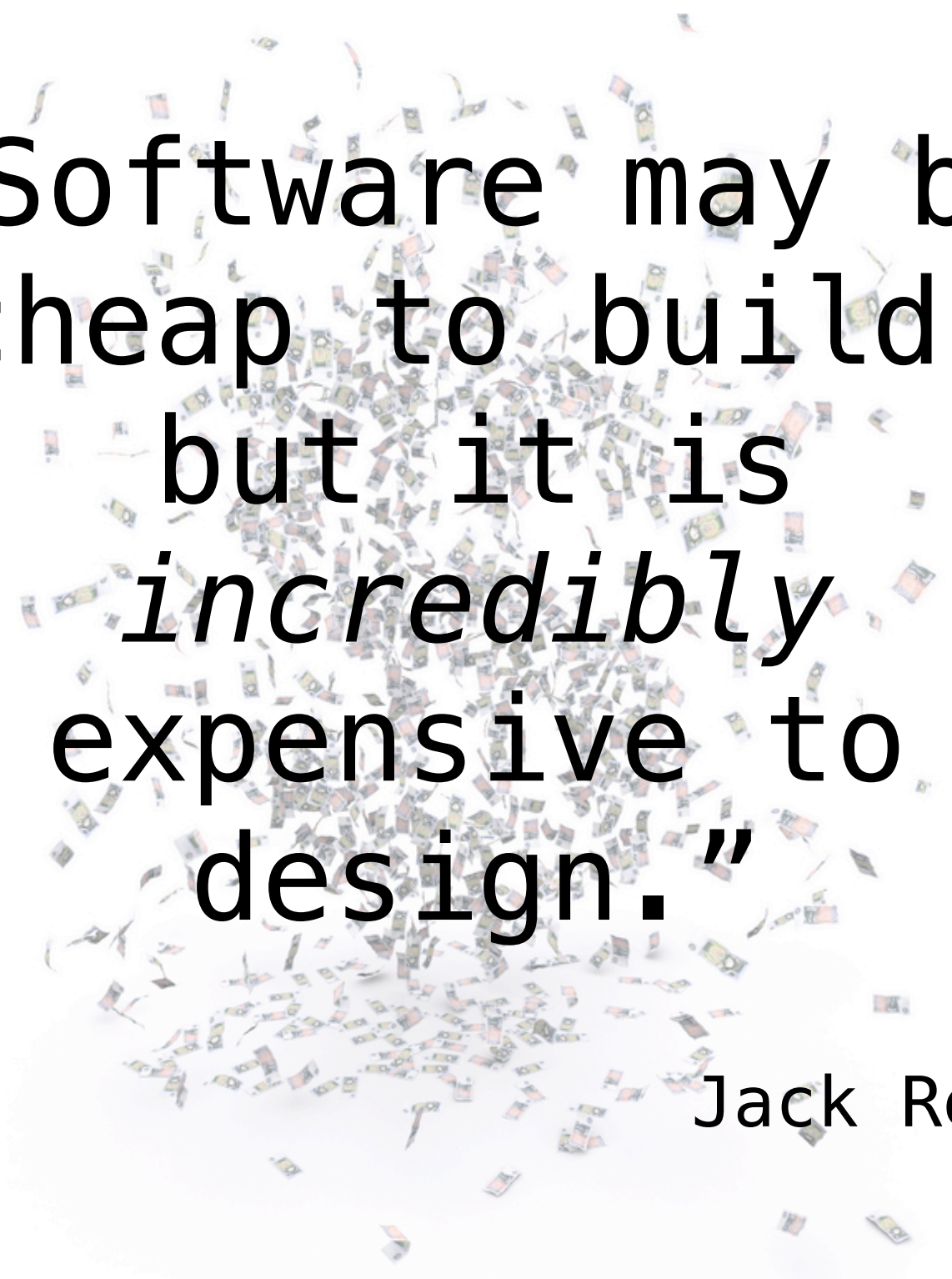
```

```

@Test public void sum() {
    Classifier6 c = new Classifier6(20);
    calculateFactors(c);
    int expected = 1 + 2 + 4 + 5 + 10 + 20;
    assertEquals(sumOfFactors(c), expected);
}

```





“Software may be
cheap to build,
but it is
incredibly
expensive to
design.”

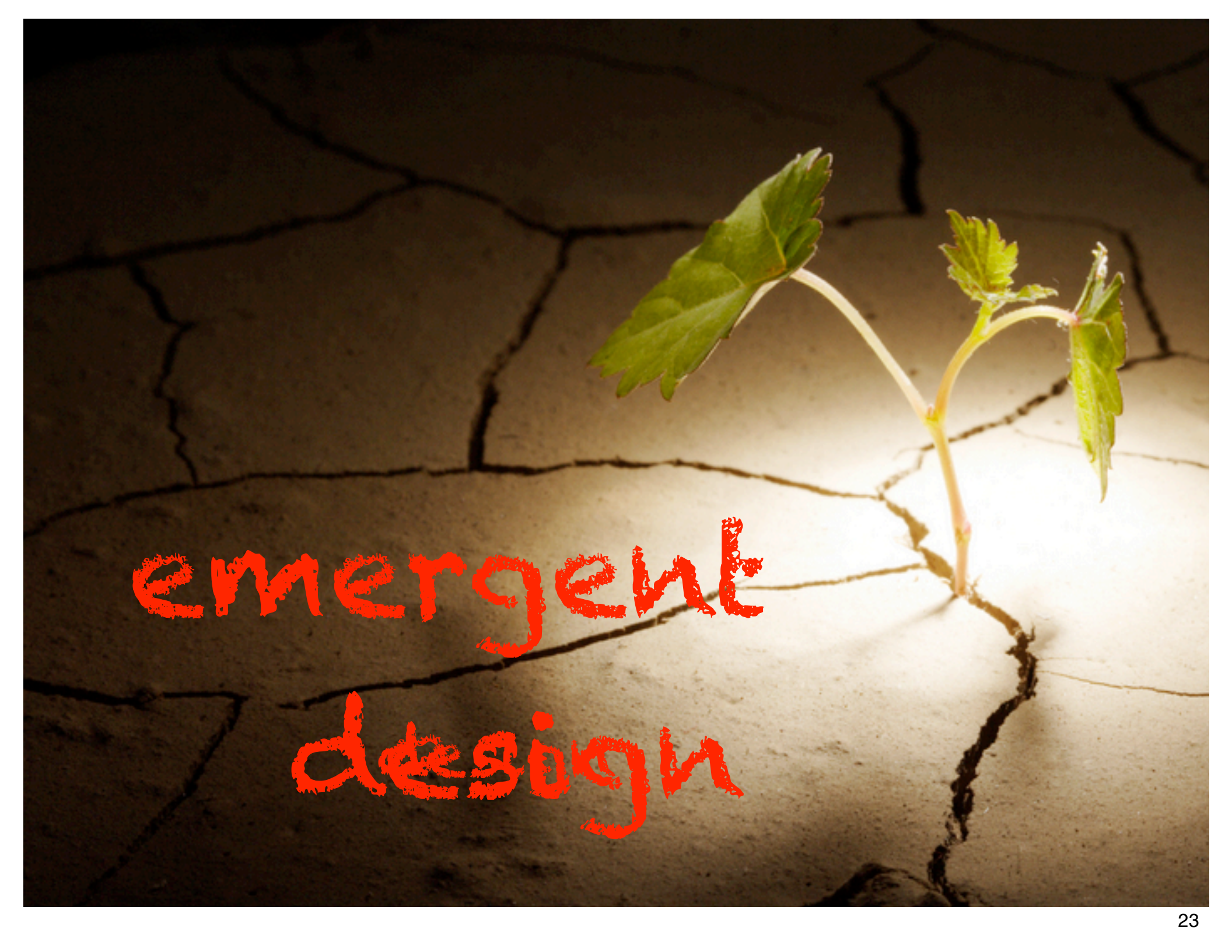
Jack Reeves

design

stuff that's
hard to
change later

architecture

(as little
of that
stuff as
possible)

A small green seedling with two leaves is growing out of a crack in dry, cracked earth. The scene is dramatically lit by a spotlight, creating a bright circular area around the plant and deep shadows elsewhere. The text "emergent design" is overlaid in a red, textured font.

emergent
design

Emergent, a.

[L. *emergens*, p. pr. of *emergere*.]

1. Rising or emerging out of a fluid or anything that covers or conceals; issuing; coming to light.
[1913 Webster]

2. Suddenly appearing; arising unexpectedly; calling for prompt action; urgent.
[1913 Webster]

finding abstractions
& patterns

idiomatic patterns



technical

domain

validation

security

transactional data

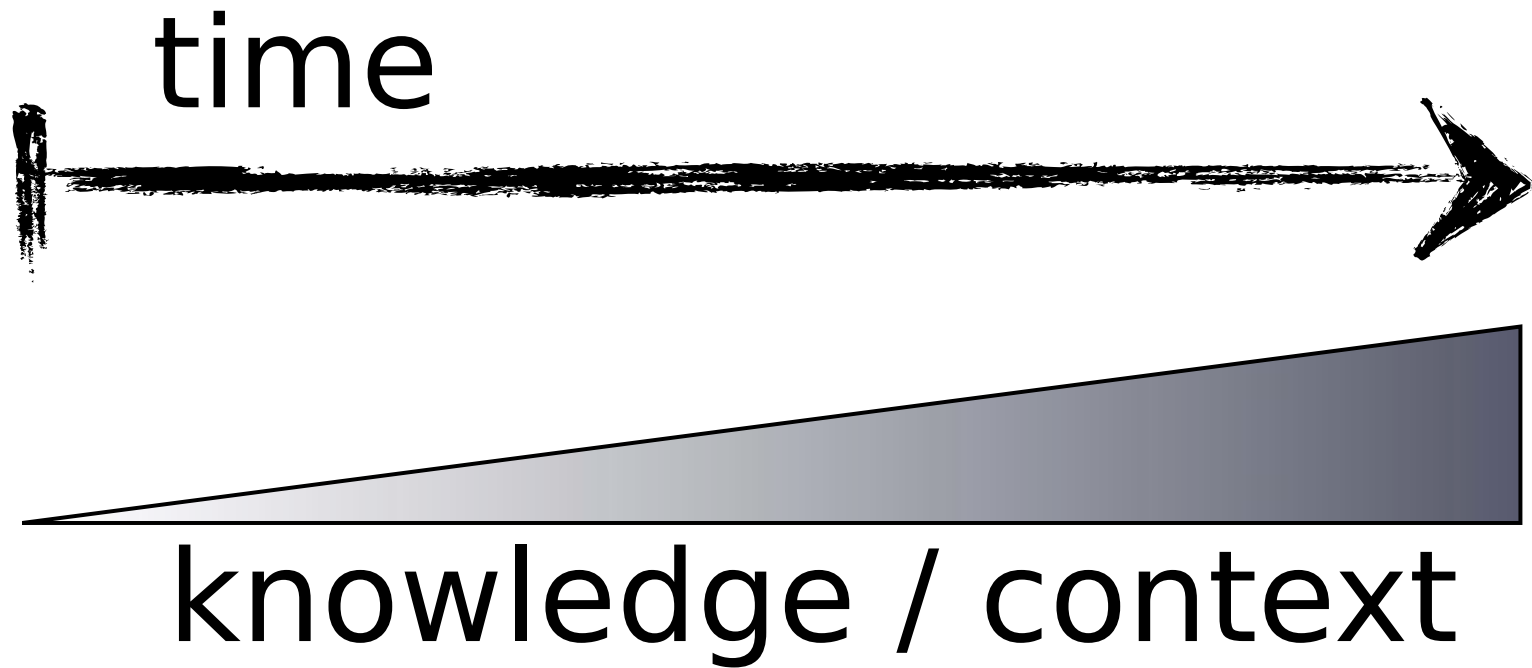
business rules

shared functionality

patterns describe effective abstractions

finding & harvesting
idiomatic patterns

last responsible
moment



the longer you can
wait, the better
the decision

Things that Obscure Emergent Design



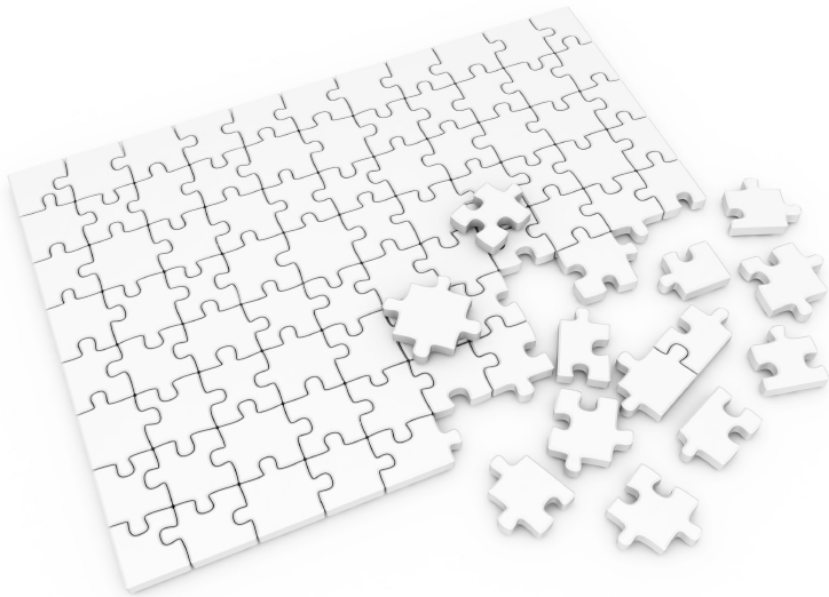
complexity



accidental complexity

*all the externally imposed
ways software becomes complex*

vs



essential
complexity
inherent complexity

examples

hunting
season

EJB /
Biztalk

field level
security

essential

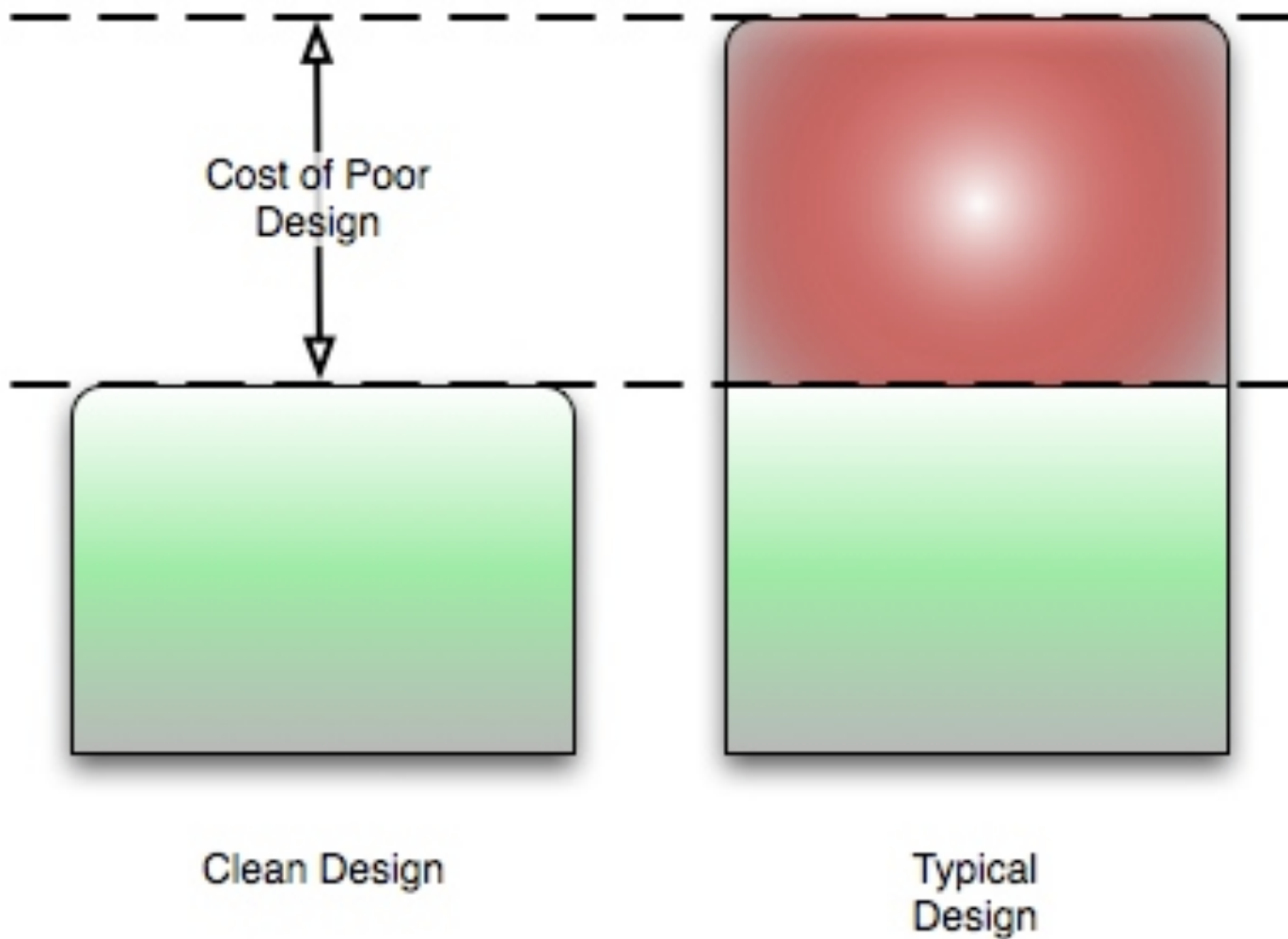
accidental





technical debt

technical debt



reckless

*"We don't have
time for design."*

prudent

*"We must ship
now & deal with
the consequences."*

deliberate

inadvertent

"What's layering?"

*"Now we know
how we should
have done it."*

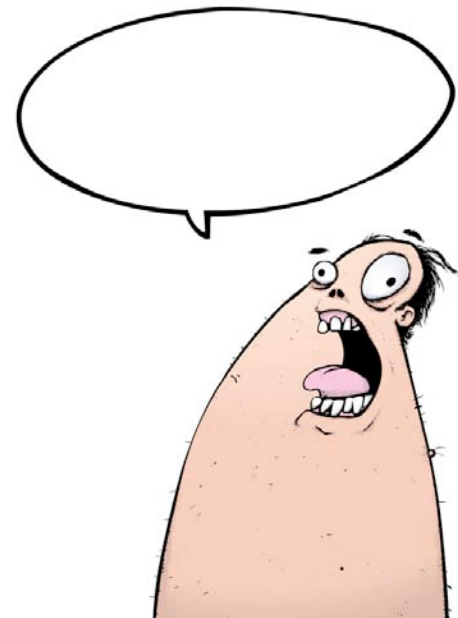
<http://martinfowler.com/bliki/TechnicalDebtQuadrant.html>

negotiating repayment

you must convince someone technical
debt exists...

...start a conversation about
repayment

demonstration



cyclomatic complexity

measures complexity of a method/function

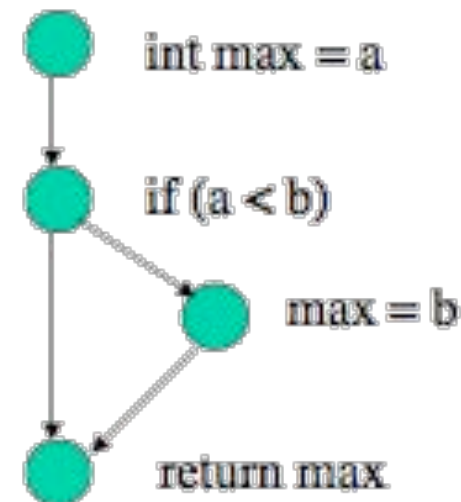
$$V(G) = e - n + 2$$

$V(G)$ = cyclomatic complexity of G

e = # edges

n = # of nodes

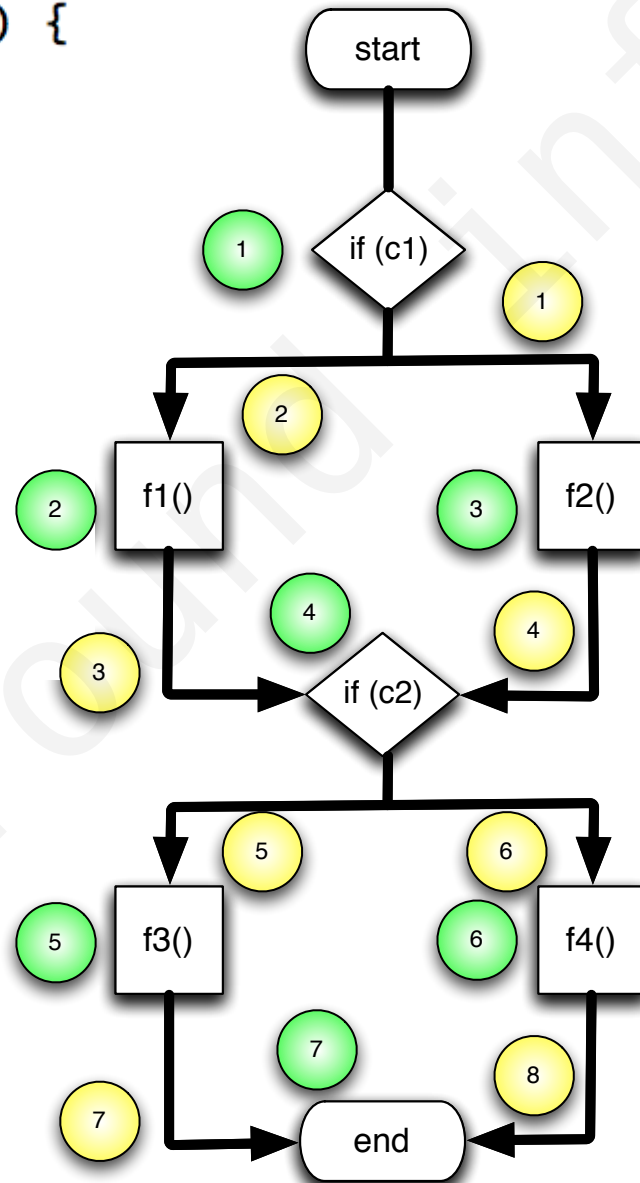
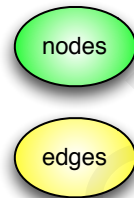
```
int max (int a, int b) {  
    int max = a;  
    if (a < b) {  
        max = b;  
    }  
    return max;  
}
```



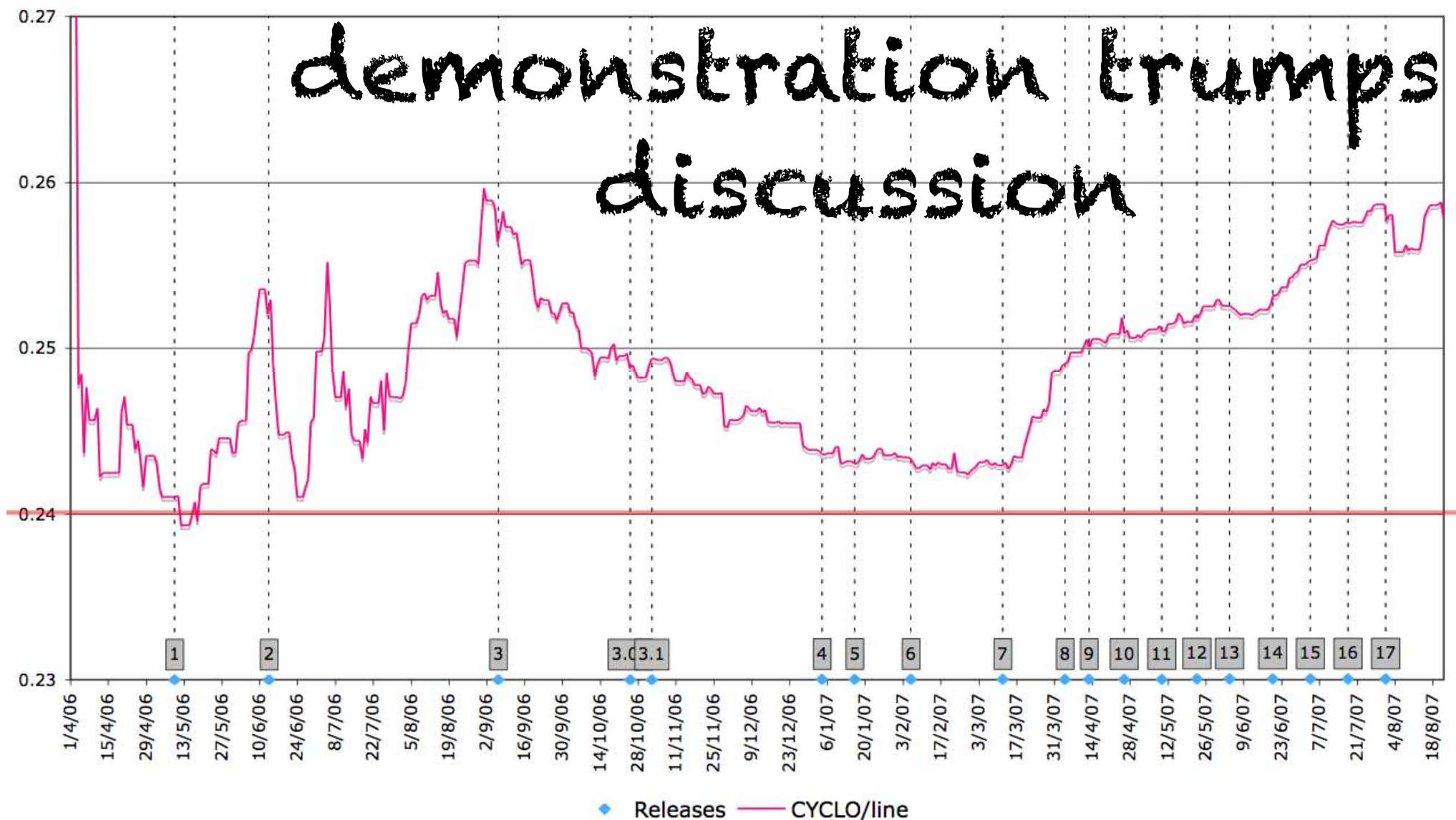
```

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

```



Operational Complexity (branching point density)



open source

nice visualizations for common metrics

sample instance (<http://nemo.sonarsource.org/>)

sonar

coverage duplications pmd **technical debt** cpd findbugs open source hotspots
quality cobertura drilldown timemachine action plans analysis maven dashboard checkstyle
unit tests continuous improvement clover coding rules **source code** plugins

Dashboard

Components
Violations drilldown
Time machine
Clouds
Design
Hotspots
Motion chart
Radiator
Timeline

Version 2.2.0-SNAPSHOT - Tue, 09 Feb 2010 17:17 - profile [Nemo rules with findbugs](#)

Lines of code

79,396 ▼

147,345 lines ▼

Classes

1,145

131 packages
7,323 methods ▼
+936 accessors

Comments

20.1%

19,988 lines ▼
30.7% docu. API
4,800 undocu. API
143 commented LOCs ▲

Duplications

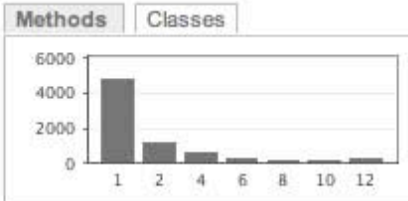
3.0%

4,432 lines
150 blocks
66 files

Complexity

2.7 / method**17.4** / class

19,908 cmpx ▼
38,241 statements ▼



Code coverage

26.2%

28.2% line coverage
21.7% branch coverage
1,126 tests
1:46 min ▲

Test success

99.8%

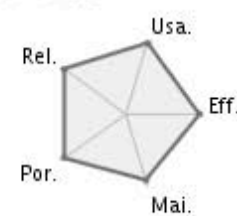
2 failures
0 errors

Events

All

2010-02-09	Version	2.2.0-SNAPSHOT
2009-09-28	Version	2.1.9-SNAPSHOT

Rules compliance

86.9%

Violations

3,833 ▲

Blocker	0
Critical	0
Major	3,460 ▲
Minor	0
Info	373

▲ Alerts : Unit test success (%) < 100.

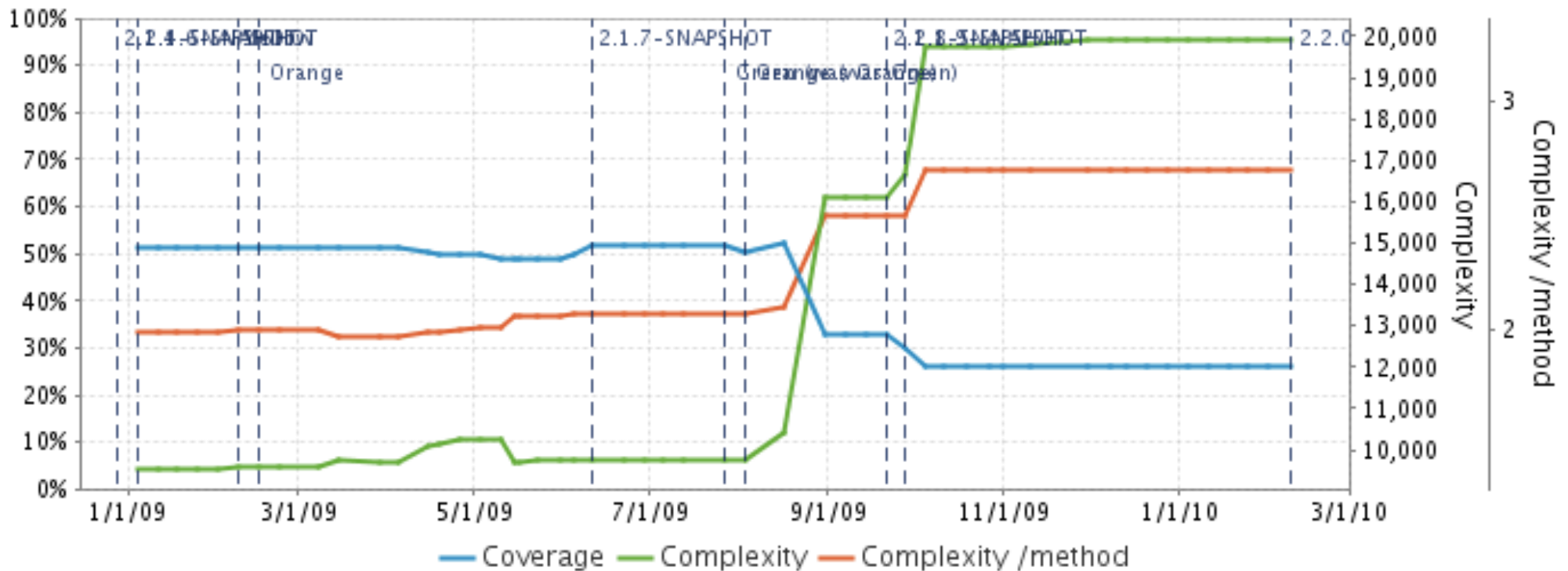
Technical Debt ⓘ

15.2%

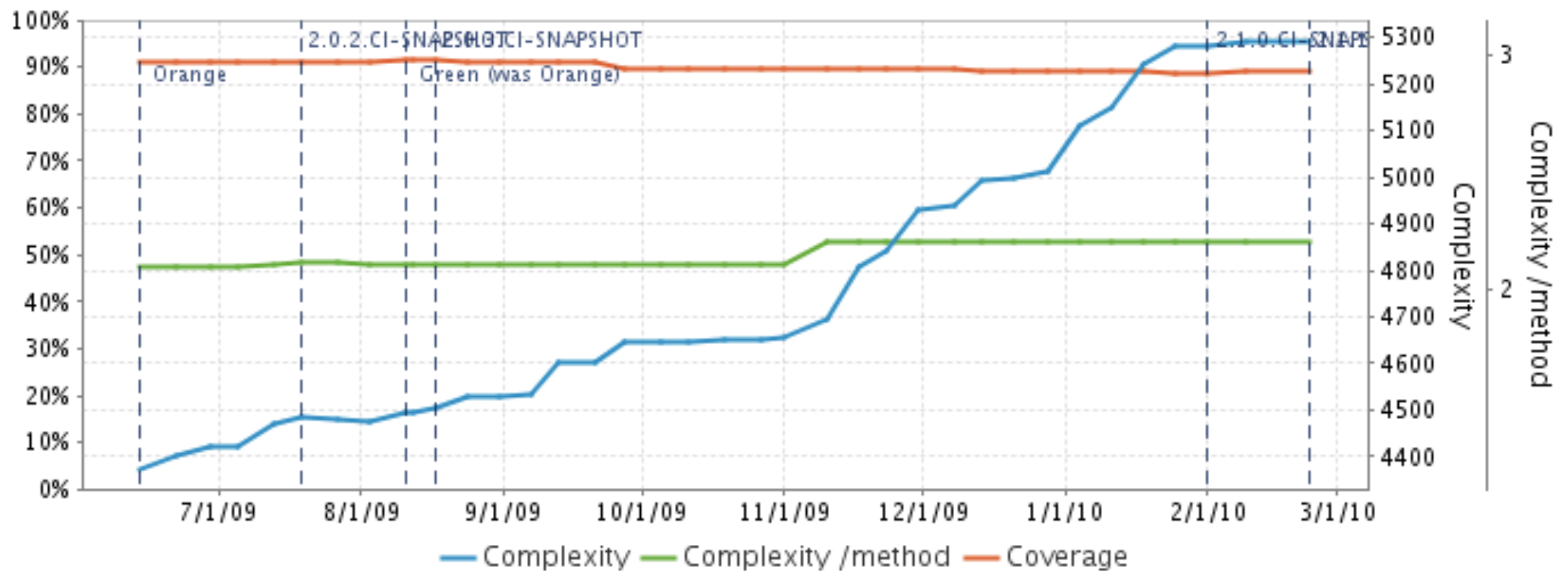
\$ 286,100 ▼
572 man days



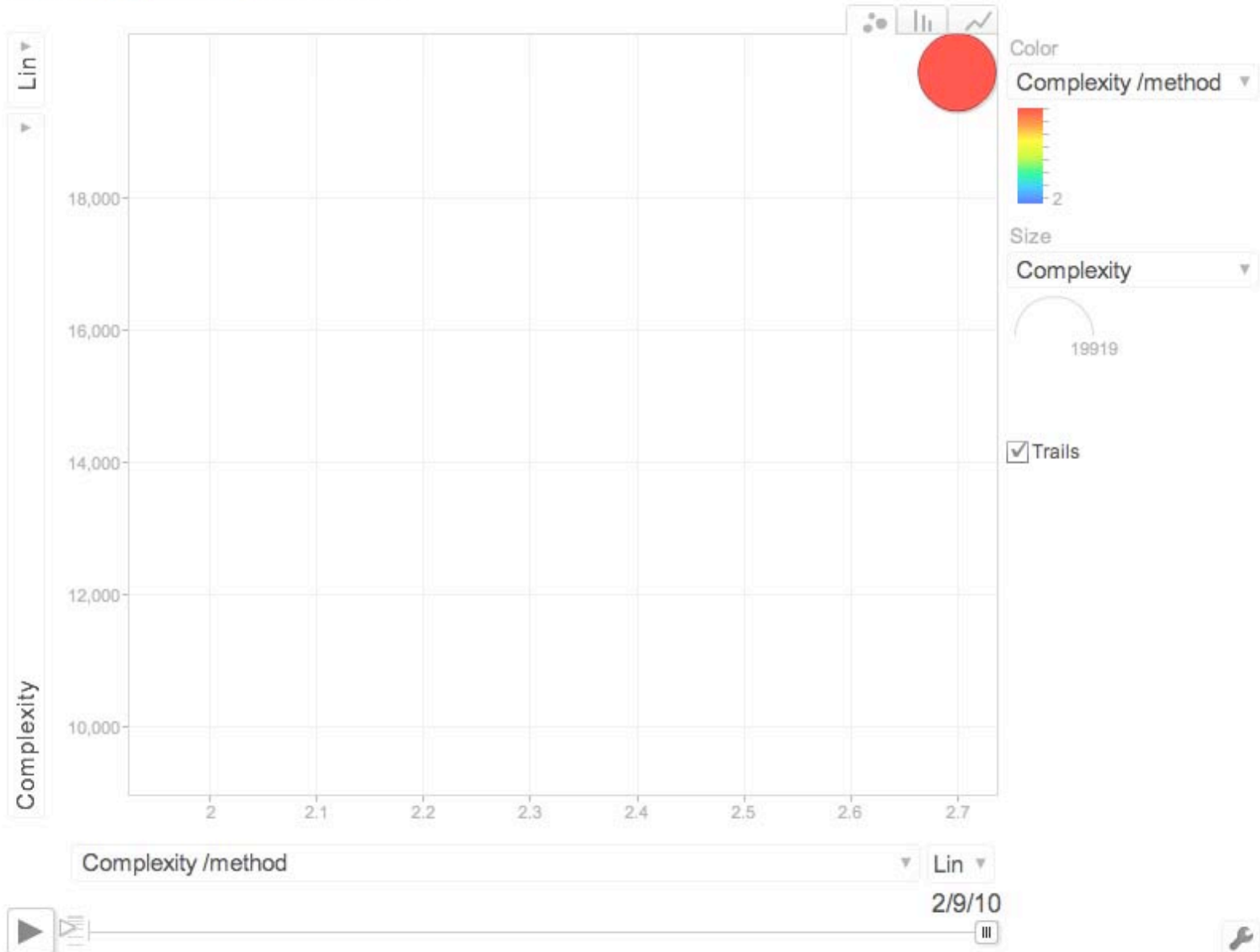
time machine (struts)



time machine (spring batch)



Period: Two years Components: ☐



A photograph of a large grid of Campbell's Soup cans, arranged in approximately 10 rows and 20 columns. The cans are of various flavors, including Tomato Soup, Chicken Noodle Soup, and others. The image is used as a background for the text.

rampant
genericness

genericness

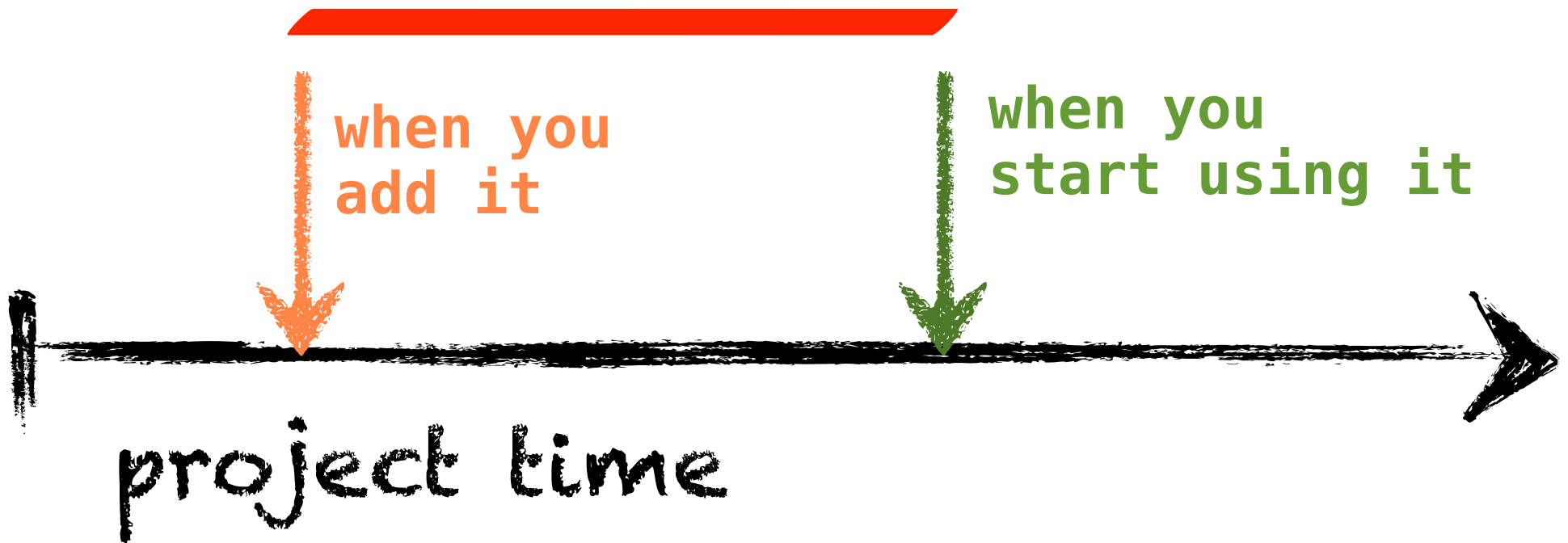
“if we build lots of layers
for extension, we can easily
build more onto it later”

increases software entropy

accidental complexity

generic obfuscation

technical debt



Emergent Design Accelerators



test driven *design*

more about design than testing

design will emerge from tests

better abstractions

less accidental complexity

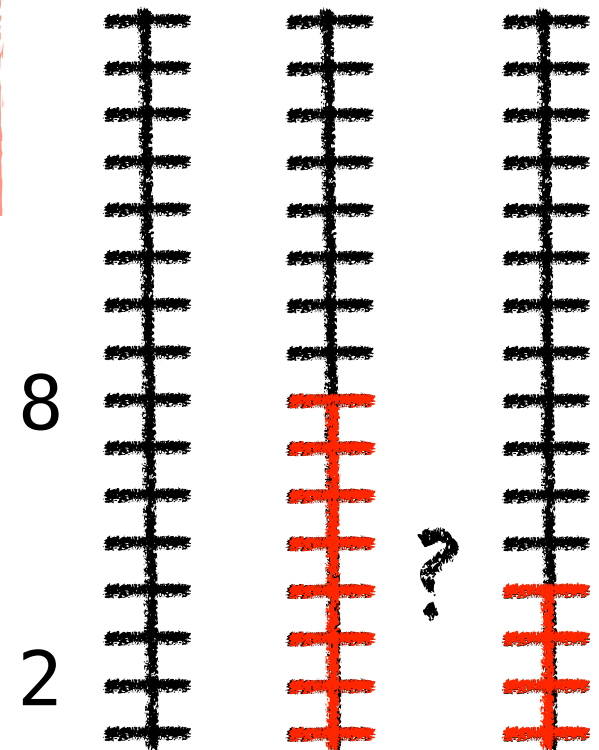
atomic understanding of intent

perfect number case study

\sum of the factors == number
(not including the number)

test-after, 1st pass

```
public class PerfectNumberFinder1 {  
    public static boolean isPerfect(int number) {  
        // get factors  
        List<Integer> factors = new ArrayList<Integer>();  
        factors.add(1);  
        factors.add(number);  
        for (int i = 2; i < number; i++)  
            if (number % i == 0)  
                factors.add(i);  
  
        // sum factors  
        int sum = 0;  
        for (int n : factors)  
            sum += n;  
  
        // decide if it's perfect  
        return sum - number == number;  
    }  
}
```



```

public class PerfectNumberFinder2 {
    public static boolean isPerfect(int number) {
        // get factors
        List<Integer> factors = new ArrayList<Integer>();
        factors.add(1);
        factors.add(number);
        for (int i = 2; i <= sqrt(number); i++)
            if (number % i == 0) {
                factors.add(i);
                factors.add(number / i);
            }

        // sum factors
        int sum = 0;
        for (int n : factors)
            sum += n;

        // decide if it's perfect
        return sum - number == number;
    }
}

```

← whole-number square roots

```

public class PerfectNumberFinder2 {
    public static boolean isPerfect(int number) {
        // get factors
        List<Integer> factors = new ArrayList<Integer>();
        factors.add(1);
        factors.add(number);
        for (int i = 2; i <= sqrt(number); i++)
            if (number % i == 0) {
                factors.add(i);
                // guard against whole-number square roots
                if (number / i != i)
                    factors.add(number / i);
            }

        // sum factors
        int sum = 0;
        for (int n : factors)
            sum += n;

        // decide if it's perfect
        return sum - number == number;
    }
}

```

```

public class Classifier6 {
    private Set<Integer> _factors;
    private int _number;

    public Classifier6(int number) {
        if (number < 1)
            throw new InvalidNumberException(
                "Can't classify negative numbers");
        _number = number;
        _factors = new HashSet<Integer>();
        _factors.add(1);
        _factors.add(_number);
    }

    private boolean isFactor(int factor) {
        return _number % factor == 0;
    }

    public Set<Integer> getFactors() {
        return _factors;
    }

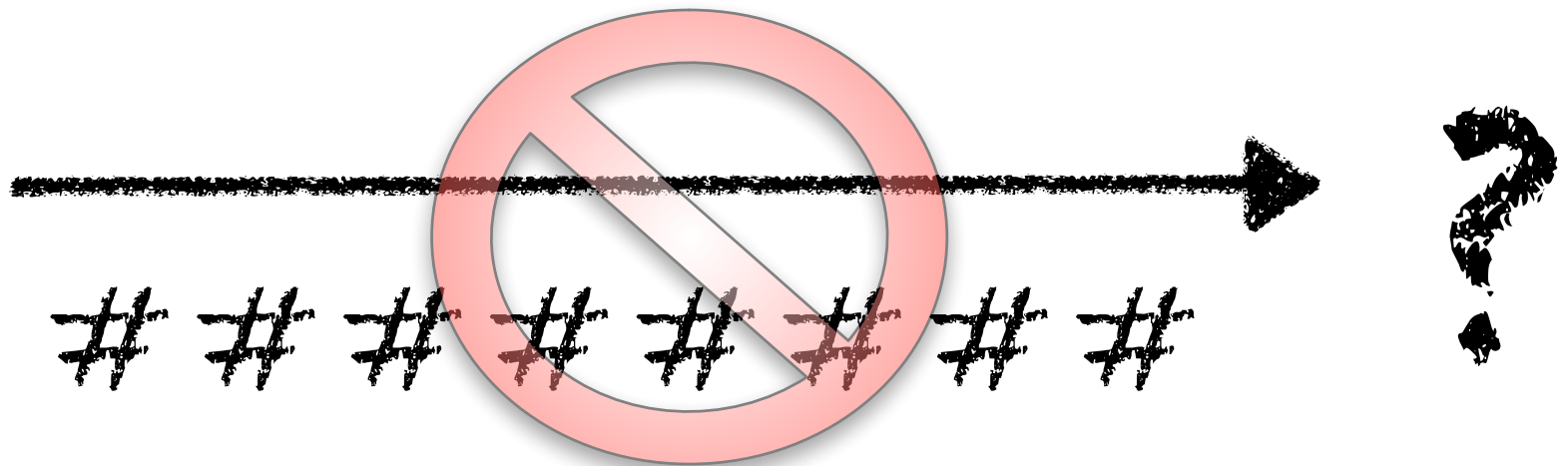
    private void calculateFactors() {
        for (int i = 2; i < sqrt(_number) + 1; i++)
            if (isFactor(i))
                addFactor(i);
    }

    private void addFactor(int factor) {
        _factors.add(factor);
        _factors.add(_number / factor);
    }

    private int sumOfFactors() {
        calculateFactors();
        int sum = 0;
        for (int i : _factors)
            sum += i;
        return sum;
    }

    public boolean isPerfect() {
        return sumOfFactors() - _number == _number;
    }
}

```

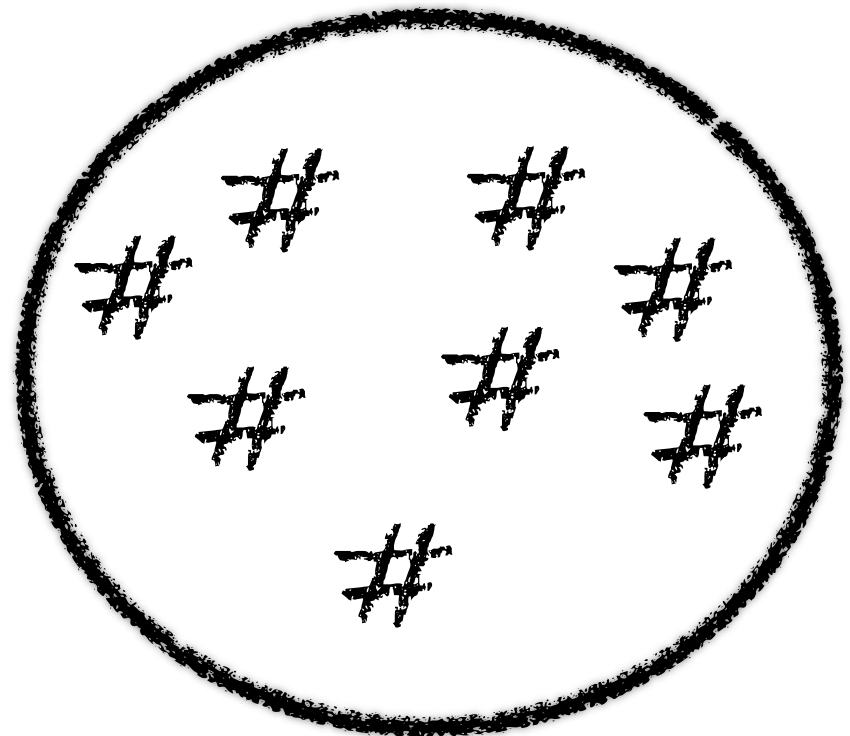


```
Done: 1 of 1 Failed: 1(0.035 s)

Output Statistics

java.lang.AssertionError:
Expected: is <[1, 2, 3, 6]>
got: <[1, 6, 2, 3]>

at org.junit.Assert.assertThat(Assert.java:502)
at org.junit.Assert.assertThat(Assert.java:492)
at com.nealford.conf.tdd.perfectnumbers.Classifier3Test
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMe
at sun.reflect.DelegatingMethodAccessorImpl.invoke(Dele
at org.junit.internal.runners.TestMethod.invoke(TestMet
at org.junit.internal.runners.MethodRoadie.runTestMetho
at org.junit.internal.runners.MethodRoadie$2.run(Method
```



test-after

```
for (int i = 2; i <= sqrt(number); i++)  
    if (number % i == 0) {  
        factors.add(i);  
        // account for whole-number square roots  
        if (number / i != i)  
            factors.add(number / i);  
    }
```

TDD

```
private void calculateFactors() {  
    for (int i = 2; i < sqrt(_number) + 1; i++)  
        if (isFactor(i))  
            addFactor(i);  
}  
  
private void addFactor(int factor) {  
    _factors.add(factor);  
    _factors.add(_number / factor);  
}
```

```

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

```

refactoring
towards
design

```

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

```

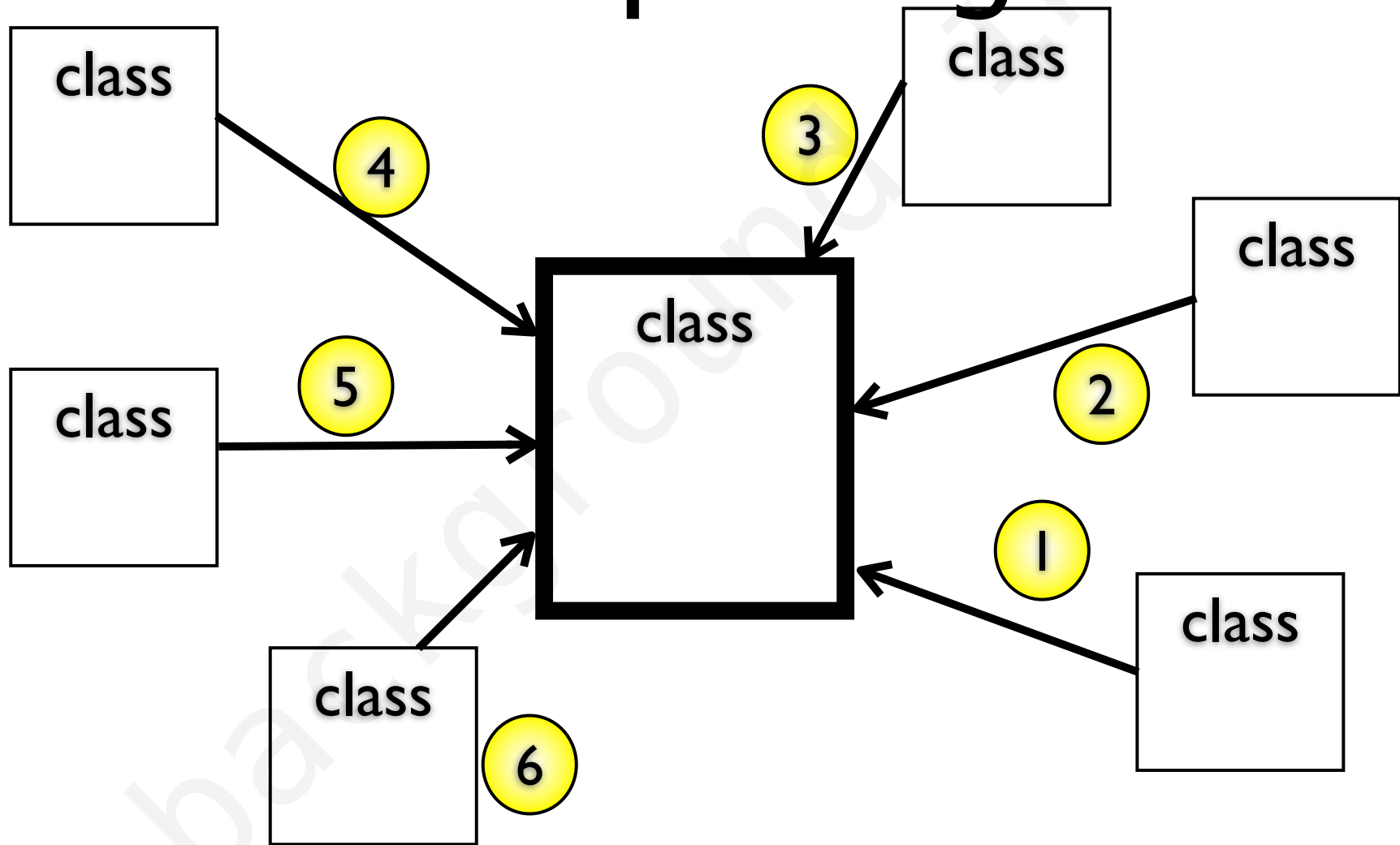
idiomatic “unit of work” pattern

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

see the *composed method* pattern
Smalltalk Best Practice Patterns Kent Beck

Refactoring to Harvest Idiomatic Patterns

afferent coupling



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

UIBean evaluateParams()

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

    String name = null;

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

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

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

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

evaluate.*Params ?

```
find . -name "*.java" | xargs grep -l "void evaluate.*Params"
```

```
./org/apache/struts2/components/AbstractRemoteCallUIBean.java  
./org/apache/struts2/components/Anchor.java  
./org/apache/struts2/components/Autocompleter.java  
./org/apache/struts2/components/Checkbox.java  
./org/apache/struts2/components/ComboBox.java  
./org/apache/struts2/components/DateTimePicker.java  
./org/apache/struts2/components/Div.java  
./org/apache/struts2/components/DoubleListUIBean.java  
./org/apache/struts2/components/DoubleSelect.java  
./org/apache/struts2/components/File.java  
./org/apache/struts2/components/Form.java  
./org/apache/struts2/components/FormButton.java  
./org/apache/struts2/components/Head.java  
./org/apache/struts2/components/InputTransferSelect.java
```

```
./org/apache/struts2/components/Label.java  
./org/apache/struts2/components/ListUIBean.java  
./org/apache/struts2/components/OptionTransferSelect.java  
./org/apache/struts2/components/Password.java  
./org/apache/struts2/components/Reset.java  
./org/apache/struts2/components/Select.java  
./org/apache/struts2/components/Submit.java  
./org/apache/struts2/components/TabbedPanel.java  
./org/apache/struts2/components/table/WebTable.java  
./org/apache/struts2/components/TextArea.java  
./org/apache/struts2/components/TextField.java  
./org/apache/struts2/components/Token.java  
./org/apache/struts2/components/Tree.java  
./org/apache/struts2/components/UIBean.java  
./org/apache/struts2/components/UpDownSelect.java
```

extract the
embedded
framework

Capturing Idiomatic Patterns



APIS



idiomatic “unit of work” pattern

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

Java

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

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

Groovy

```
public class OrderDbClosure {
    def wrapInTransaction(command) {
        setupDataInfrastructure()
        try {
            command()
            completeTransaction()
        } catch (RuntimeException ex) {
            rollbackTransaction()
            throw ex
        } finally {
            cleanUp()
        }
    }

    def addOrderFrom(cart, userName, order) {
        wrapInTransaction {
            add order, userKeyBasedOn(userName)
            addLineItemsFrom cart, order.orderKey
        }
    }
}
```

Annotations

```

public class Country {
    private List<Region> regions = new ArrayList<Region>();
    private String name;

    public Country(String name){
        this.name = name;
    }

    @MaxLength(length = 10)
    public String getName(){
        return name;
    }

    public void addRegion(Region region){
        regions.add(region);
    }

    public List<Region> getRegions(){
        return regions;
    }
}

```

```

@Retention(RetentionPolicy.RUNTIME)
public @interface MaxLength {
    int length() default 0;
}

```

```

public abstract class Validator {

    public void validate(Object obj) throws ValidationException {
        Class cls = obj.getClass();
        for(Method method : cls.getMethods()){
            if (method.isAnnotationPresent(getAnnotationType())){
                validateMethod(obj, method,
                    method.getAnnotation(getAnnotationType()));
            }
        }
    }

    protected abstract Class getAnnotationType();
    protected abstract void validateMethod(Object obj, Method method,
                                           Annotation annotation);
}

```

```

public class MaxLengthValidator extends Validator {

    protected void validateMethod(Object obj,
                                   Method method,
                                   Annotation annotation) {

        try {
            if (method.getName().startsWith("get")){
                MaxLength length = (MaxLength)annotation;
                String value = (String)method.invoke(obj, new Object[0]);
                if ((value != null) && (length.length() < value.length())){
                    String string = method.getName() +
                                    " is too long. Is " +
                                    value.length() +
                                    " but should be no longer than " + length.length();
                    throw new ValidationException(string);
                }
            }
        } catch (Exception e) {
            throw new ValidationException(e.getMessage());
        }
    }

    @Override
    protected Class getAnnotationType() {
        return MaxLength.class;
    }
}

```

```

public class Region {
    private String name = "";
    private Country country = null;

    public Region(String name, Country country) {
        this.name = name;
        this.country = country;
        this.country.addRegion(this);
    }

    public void setName(String name){
        this.name = name;
    }

    @Unique(scope = Country.class)
    public String getName(){
        return this.name;
    }

    public Country getCountry(){
        return country;
    }
}

```

```

@Retention(RetentionPolicy.RUNTIME)
public @interface Unique {
    Class scope() default Unique.class;
}

```

```

public class UniqueValidator extends Validator{

    @Override
    protected void validateMethod(Object obj, Method method, Annotation annotation) {
        Unique unique = (Unique) annotation;
        try {
            Method scopeMethod = obj.getClass().getMethod("get" + unique.scope().getSimpleName());
            Object scopeObj = scopeMethod.invoke(obj, new Object[0]);

            Method collectionMethod = scopeObj.getClass().getMethod("get" + obj.getClass().getSimpleName() + "s");
            List collection = (List)collectionMethod.invoke(scopeObj, new Object[0]);
            Object returnValue = method.invoke(obj, new Object[0]);
            for(Object otherObj: collection){
                Object otherReturnValue = otherObj.getClass().
                    getMethod(method.getName()).invoke(otherObj, new Object[0]);
                if (!otherObj.equals(obj) && otherReturnValue.equals(returnValue))
                    throw new ValidationException(method.getName() + " on " +
                        obj.getClass().getSimpleName() + " should be unique but is not since");
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
            throw new ValidationException(e.getMessage());
        }
    }

    @Override
    protected Class getAnnotationType() {
        return Unique.class;
    }
}

```

```

public class UniqueValidationTestCase extends TestCase {

    public void testNoExceptionWillBeThrownSinceThereAreNoDuplicateNames(){
        Country country = new Country("UK");

        Region region1 = new Region("South", country);
        Region region2 = new Region("North", country);

        UniqueValidator validator = new UniqueValidator();

        validator.validate(region1);
    }

    public void testWillFailIfTwoRegionsWithinACountryHaveTheSameName(){
        Country country = new Country("UK");

        Region region1 = new Region("North", country);
        Region region2 = new Region("North", country);

        UniqueValidator validator = new UniqueValidator();

        try{
            validator.validate(region1);
            fail();
        } catch (ValidationException ignored){
        }
    }
}

```

annotations add expressiveness

Sticky Annotations (Ruby)



Limiting testing

```
require 'test/unit'
class CalculatorTest < Test::Unit::TestCase

  def test_some_complex_calculation
    assert_equal 2, Calculator.new(4).complex_calculation
  end

end
```

conditional method definition

```
class CalculatorTest<Test::Unit::TestCase

  if ENV["BUILD"] == "ACCEPTANCE"

    def test_some_complex_calculation
      assert_equal 2, Calculator.new(4).complex_calculation
    end

  end

end
```

annotation

```
class CalculatorTest<Test::Unit::TestCase
  extend TestDirectives

  acceptance_only
  def test_some_complex_calculation
    assert_equal 2, Calculator.new(4).complex_calculation
  end
end
```

using hook methods

```
module TestDirectives

  def acceptance_only
    @acceptance_build = ENV["BUILD"] == "ACCEPTANCE"
  end

  def method_added(method_name)
    remove_method(method_name) unless @acceptance_build
    @acceptance_build = false
  end

end
```

cross-cutting concerns

```
class Approval
  extend Loggable

  logged
  def decline(approver, comment)
    #implementation
  end
end
```

```

module Loggable
  def logged
    @logged = true
  end

  def method_added(method_name)
    logged_method = @logged
    @logged = false

    if logged_method
      original_method = :"unlogged_#{method_name.to_s}"
      alias_method original_method, method_name

      define_method(method_name) do |*args|
        arg_string = args.collect{ |arg| arg.inspect + " " } unless args.empty?
        log_message = "called #{method_name}"
        log_message << " with #{arg_string}" if arg_string
        Logger.log log_message
        self.send(original_method, *args)
      end
    end
  end
end
end

```

```

public ActionForward edit(ActionMapping mapping, ActionForm form,
                        HttpServletRequest request,
                        HttpServletResponse response)
    throws Exception {
    PersonForm personForm = (PersonForm) form;
    if (personForm.getId() != null) {
        PersonManager mgr = (PersonManager) getBean("personManager");
        Person person = mgr.getPerson(personForm.getId());
        personForm = (PersonForm) convert(person);
        updateFormBean(mapping, request, personForm);
    }
    return mapping.findForward("edit");
}

```

Struts

expressiveness matters!

```

def edit
  @person = Person.find(params[:id])
end

```

Ruby on Rails

expressiveness matters

a lot!

if code == design...

...you want the most expressive
medium you can find

frequently meta-language nature

push for expressiveness

abstraction styles

imperative

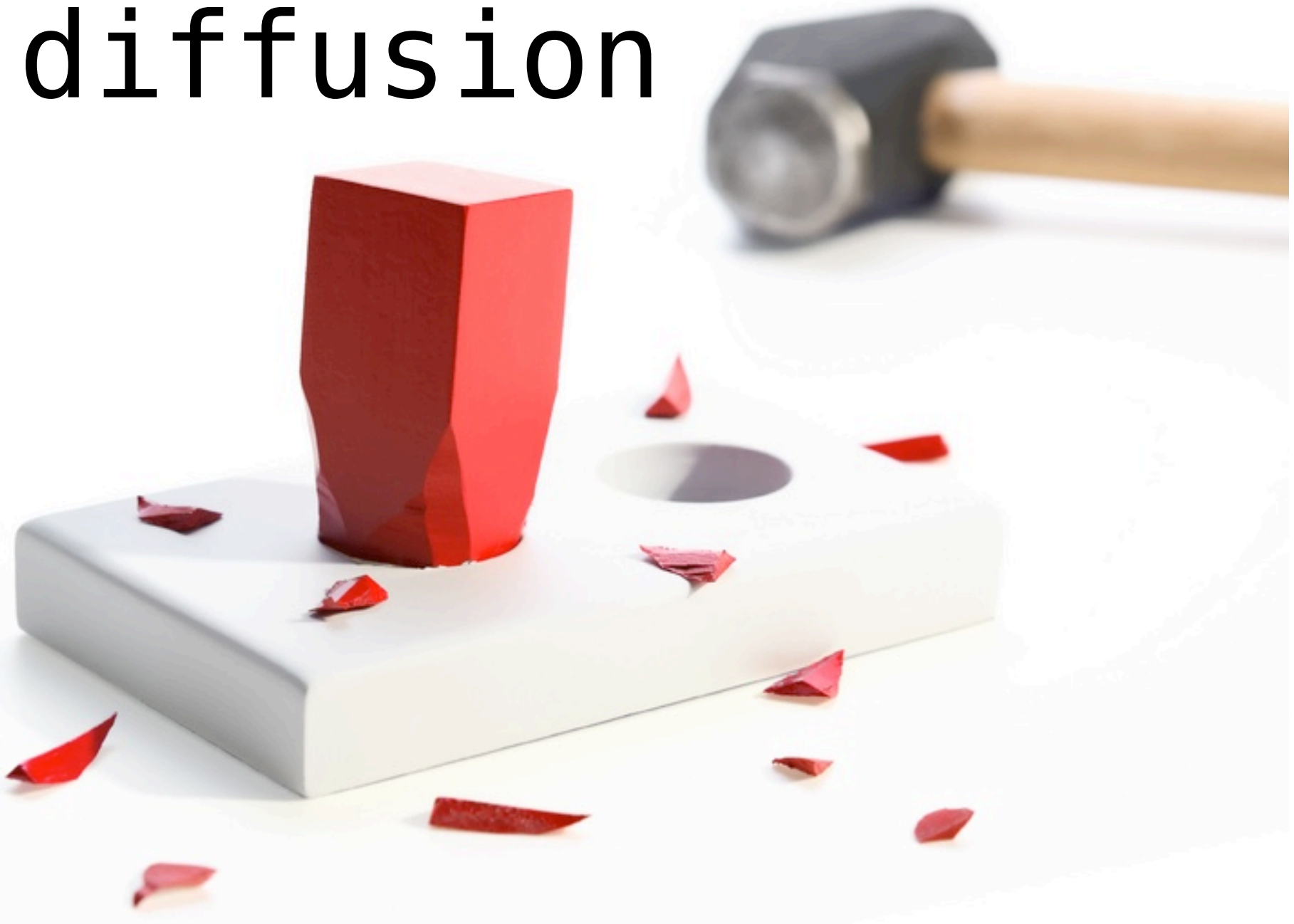
structured / modular

object-oriented

functional

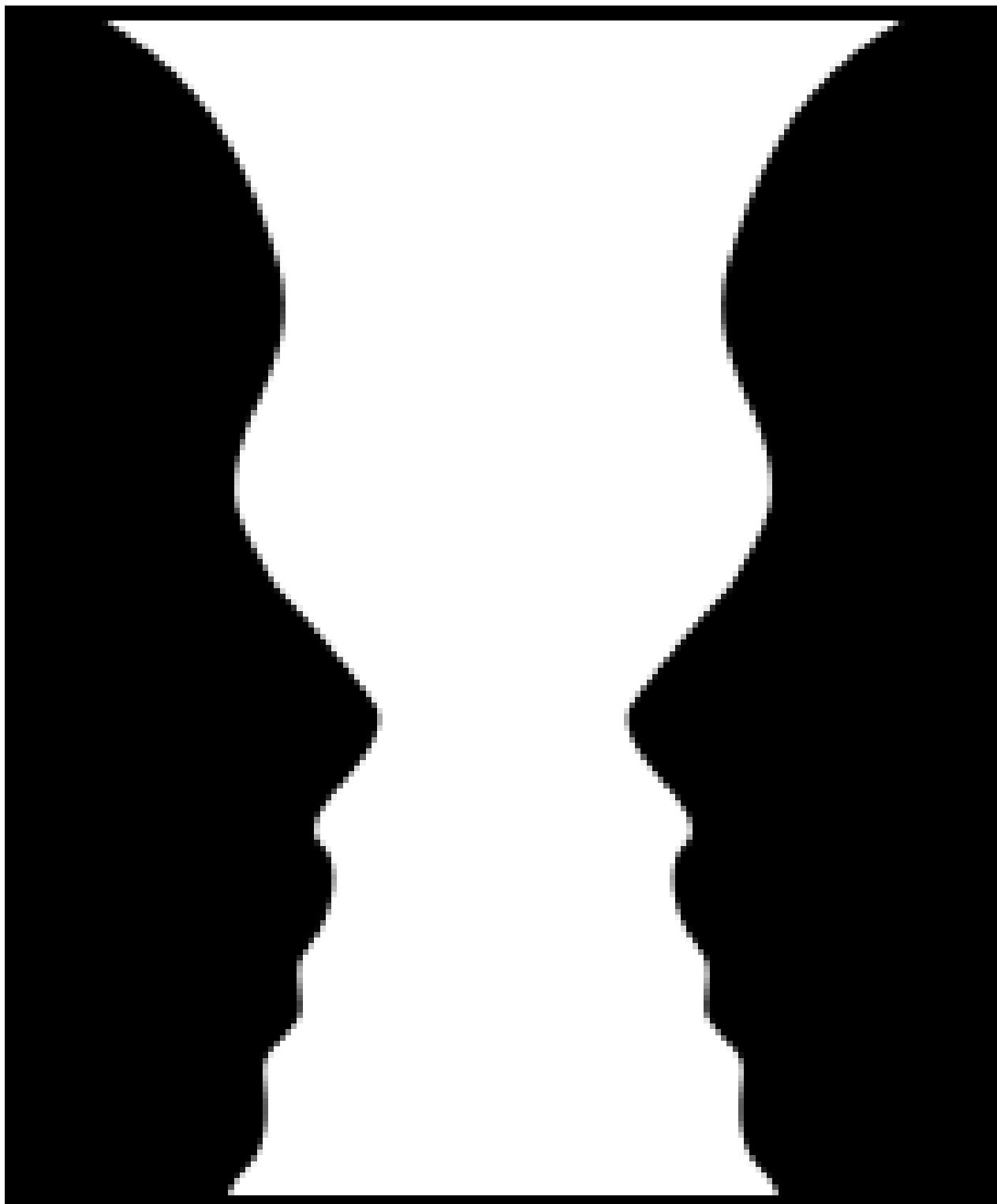
anti-objects

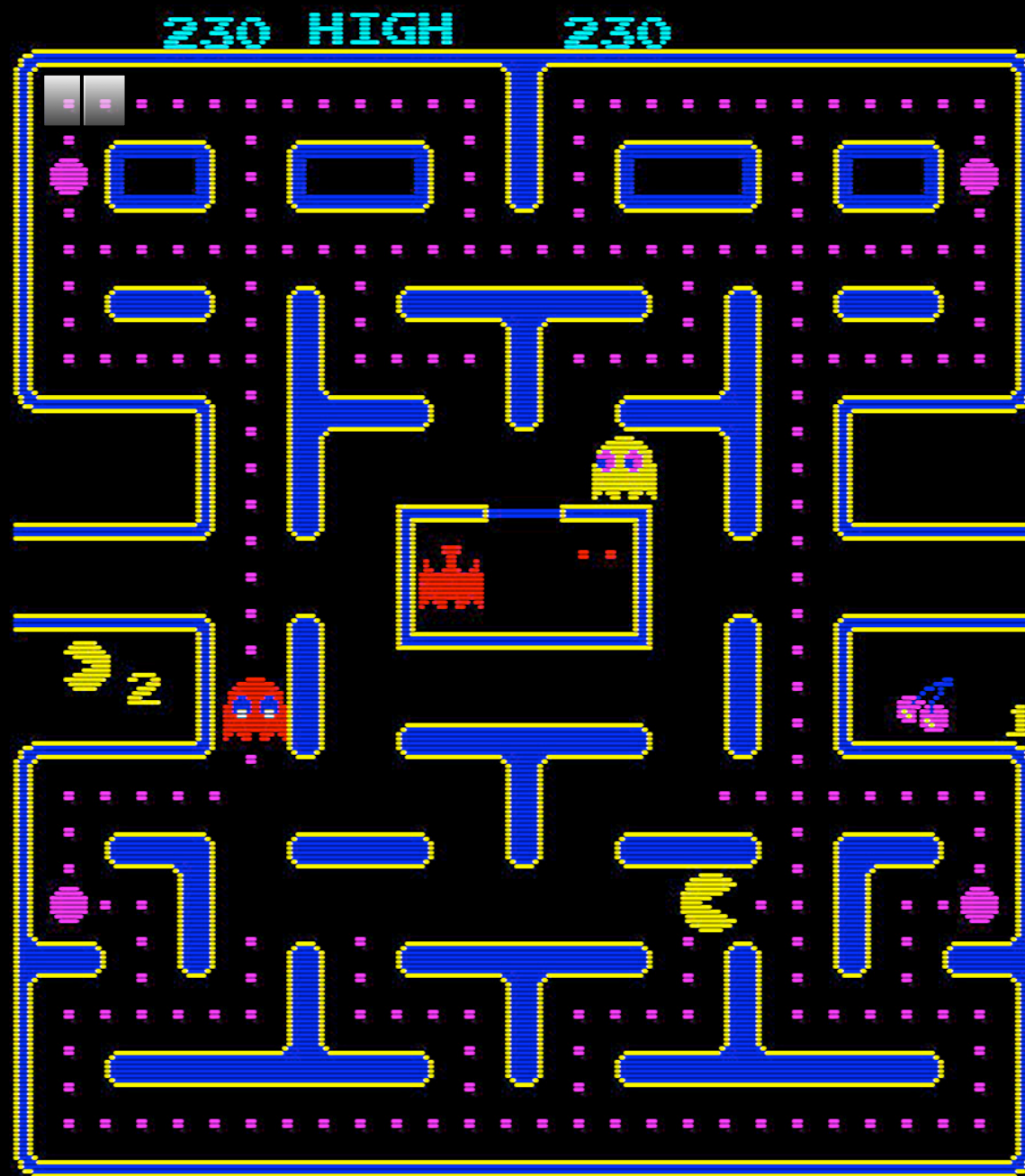
collaborative diffusion



“The metaphor of objects can go too far by making us try to create objects that are too much inspired by the real world. ”

“...an antiobject is a kind of object that appears to essentially do the opposite of what we generally think the object should be doing.”

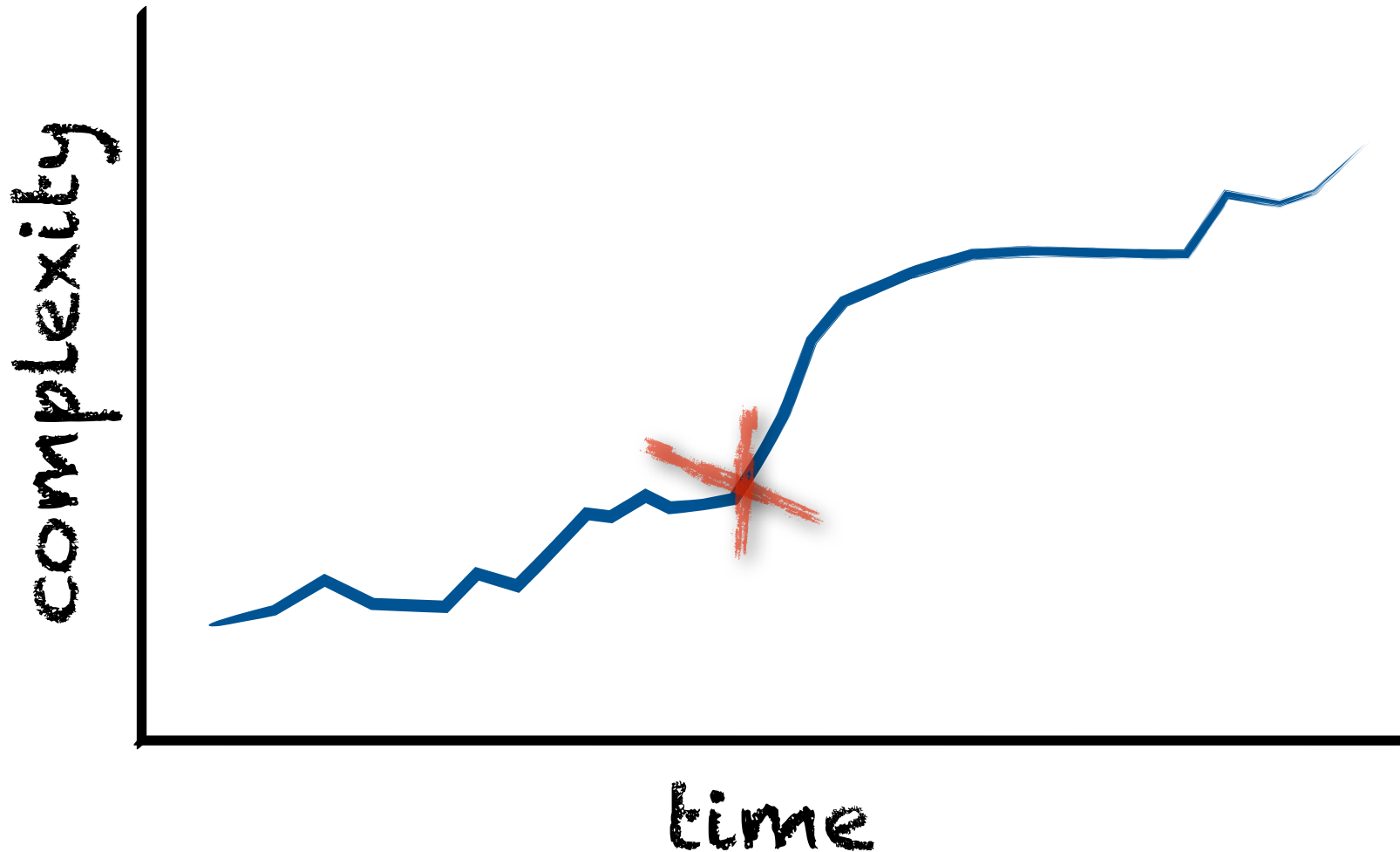




finding & harvesting
idiomatic patterns

last responsible
moment

last responsible moment



A top-down view of a rusty, brown metal bucket filled with numerous metal spikes. The spikes are of various lengths and are arranged haphazardly. Some spikes have markings on their heads, such as "5 OH" and "40". The bucket is placed on a surface of grey and blue gravel. A portion of another similar bucket is visible in the bottom left corner.

spikes are your
friends

OVE.com - Buy

You are logged in as demodealer | >> Logout English

ove.com >> BUY >> SELL >> MY OVE >> SERVICES & TOOLS >> Help

Buy

>> Basic Search >> Advanced Search

Type: All Passenger Vehicles
 Make: All
 Model: All
 Trim: All
 Years: All - 2010
 VIN:
 Seller:
☐ Vehicles with Condition Reports

>> Facilitation Location >> Vehicle Location

☐ All Locations
 United States
☐ All United States Locations
☐ AR - Central Arkansas Auto Auction (15)
☐ AZ - Manheim Arizona (205)
☐ AZ - Manheim Phoenix (250)
☐ AZ - Manheim Tucson (114)
☐ AZ - ADESA Phoenix (0)
☐ AZ - DAA Southwest (0)

>> Sellers by Type >> Sellers A to Z

Expand All Collapse All Select All Unselect All

+ Captive Finance (Credit Cars)
 + Dealer
 + Factory
 + Fleet/Lease
 + Rental

>>Search

QUICK LINKS

[Make OVE your homepage](#)
[Newly Listed!](#)
[Expiring Soon!](#)
[Fuel Efficient](#) (4-cylinders)
[Hybrids / Alternative Fuel](#)
[In-Service Rentals](#)
[Specialty](#)
[Salvage](#)

Announcement:
 Chrysler Financial has suspended dealer floor plan accounts. Please contact your preferred Facilitation Service Provider to arrange for alternate payment terms.

ONLINE EVENT SALES

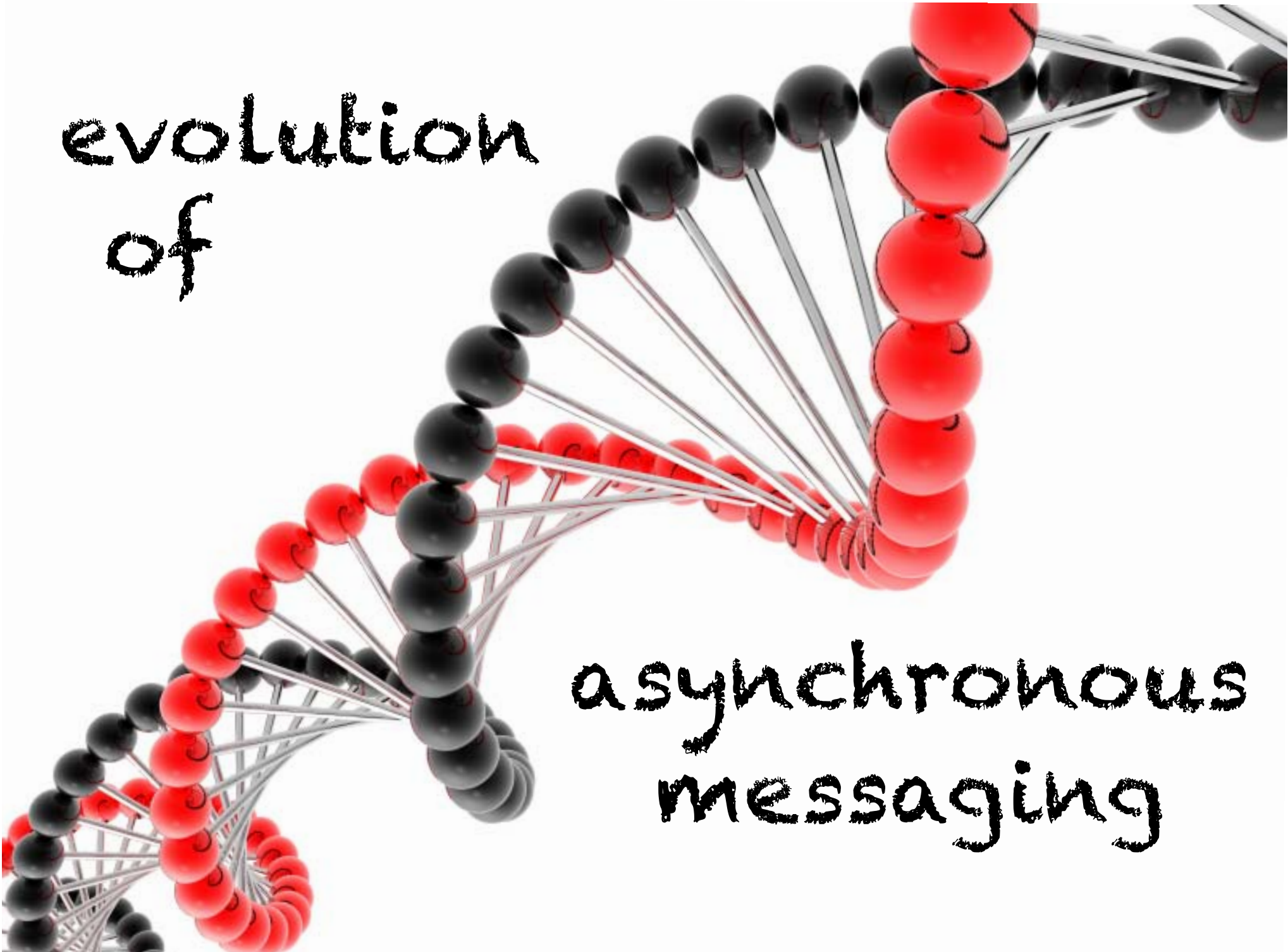
BROWSE FOR VEHICLES - Hide

Vehicle Type	Makes
Passenger Vehicles	AM General (6) Acura (202) Adventure (1) Airstream (3) Alfa (5) Alfa Romeo (1)
	Flagstaff (1) Fleetwood (12) Fontaine (3) Ford (4367) Forest River (15) Formula (1)
	Monaco (2) Monon (1) Monterey (1) Nash (2) Nautique (1) New Holland (1)

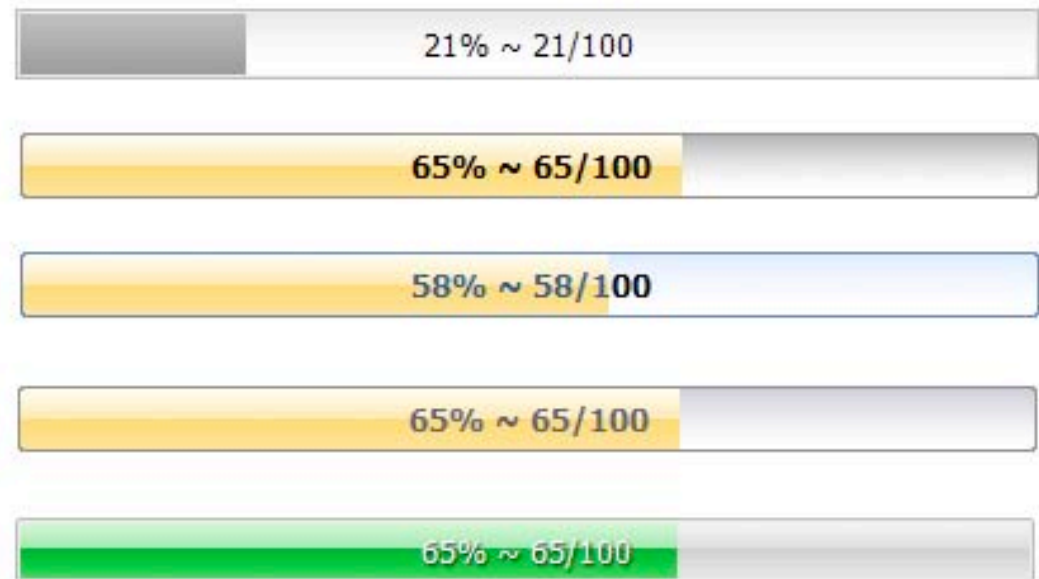
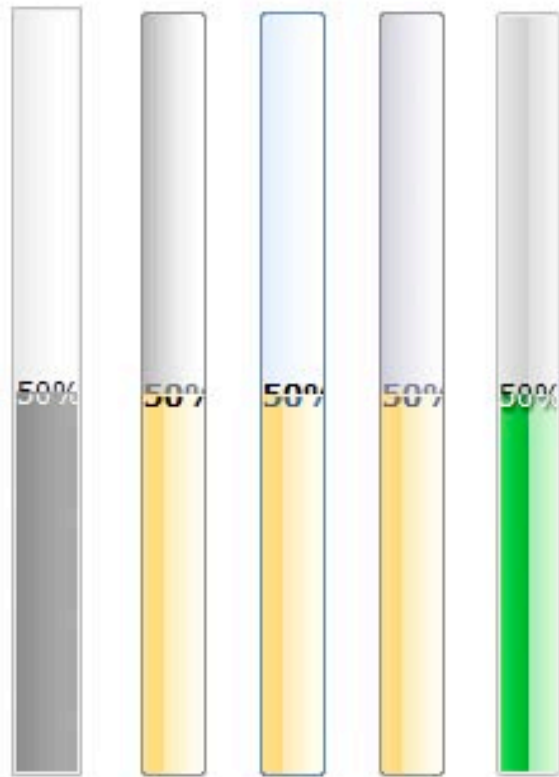
case study

evolution
of

asynchronous
messaging

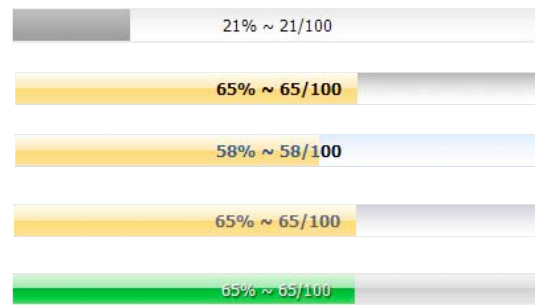
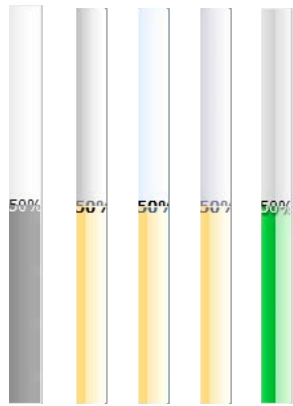


progress bars & async upload

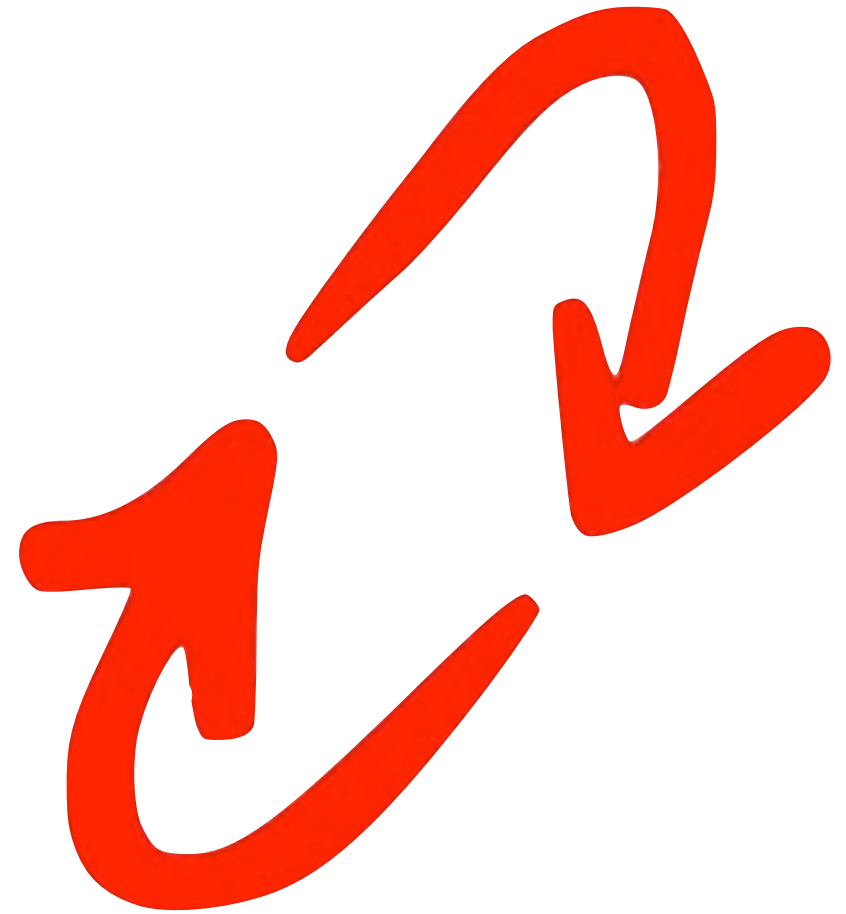


backgroundrb

<http://backgroundrb.rubyforge.org/>

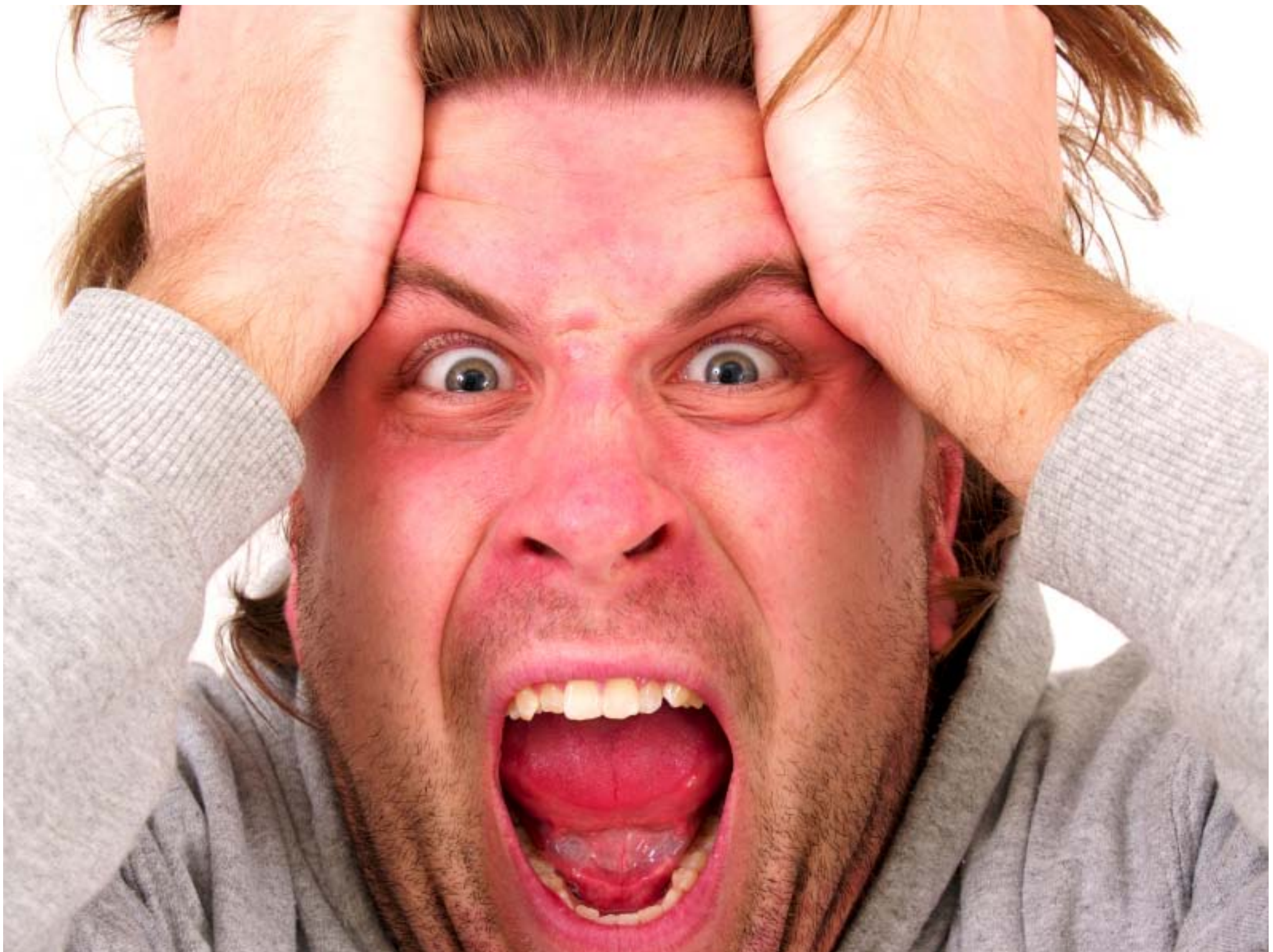


progress bars



continually run

timed events



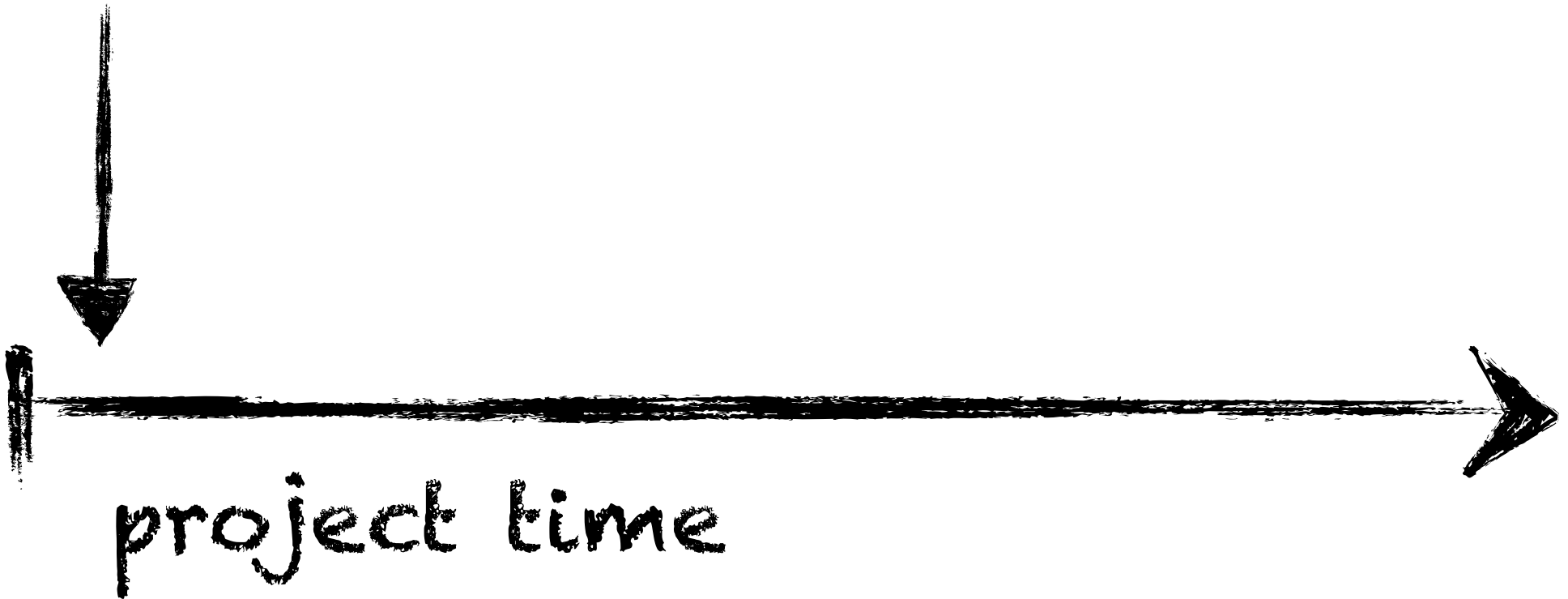
A black and white photograph of a massive flock of starlings in flight. The birds are densely packed, forming a large, dark, V-shaped pattern against a lighter sky. Below the birds, a body of water is visible, reflecting the light. The overall scene is dynamic and visually striking.

(Starling)

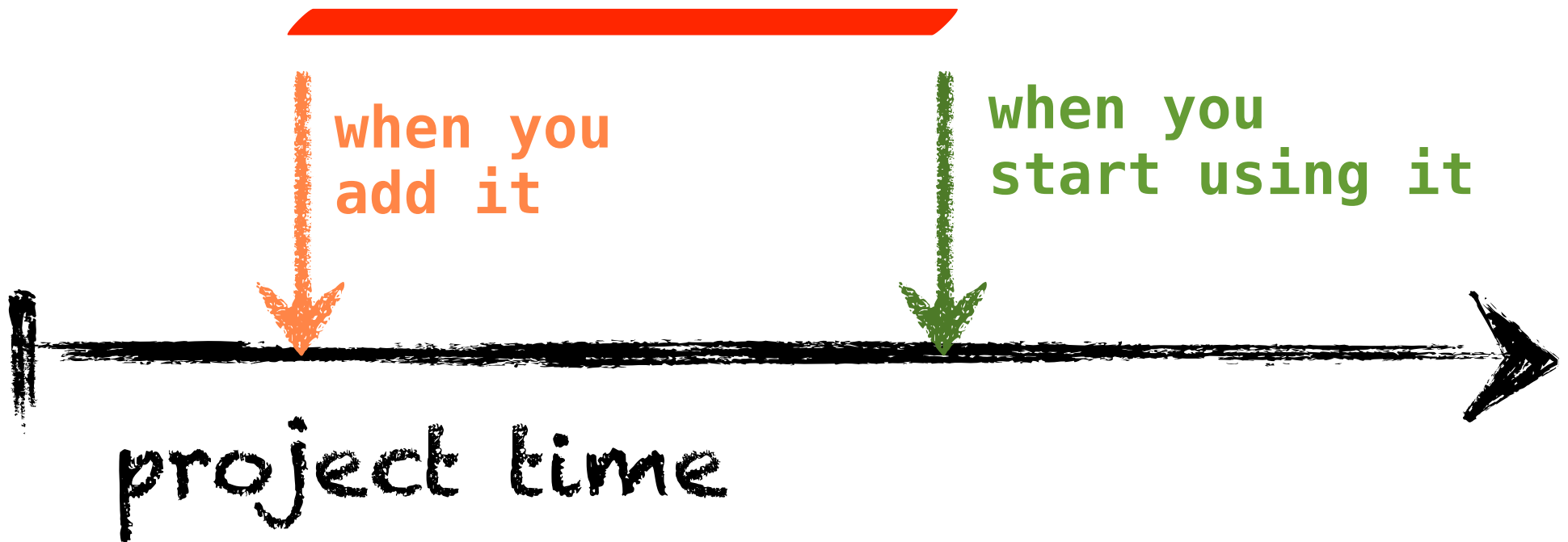
switch to a real
messaging queue

don't know what we don't know

“buy the fanciest one we
can” (just in case)



technical debt

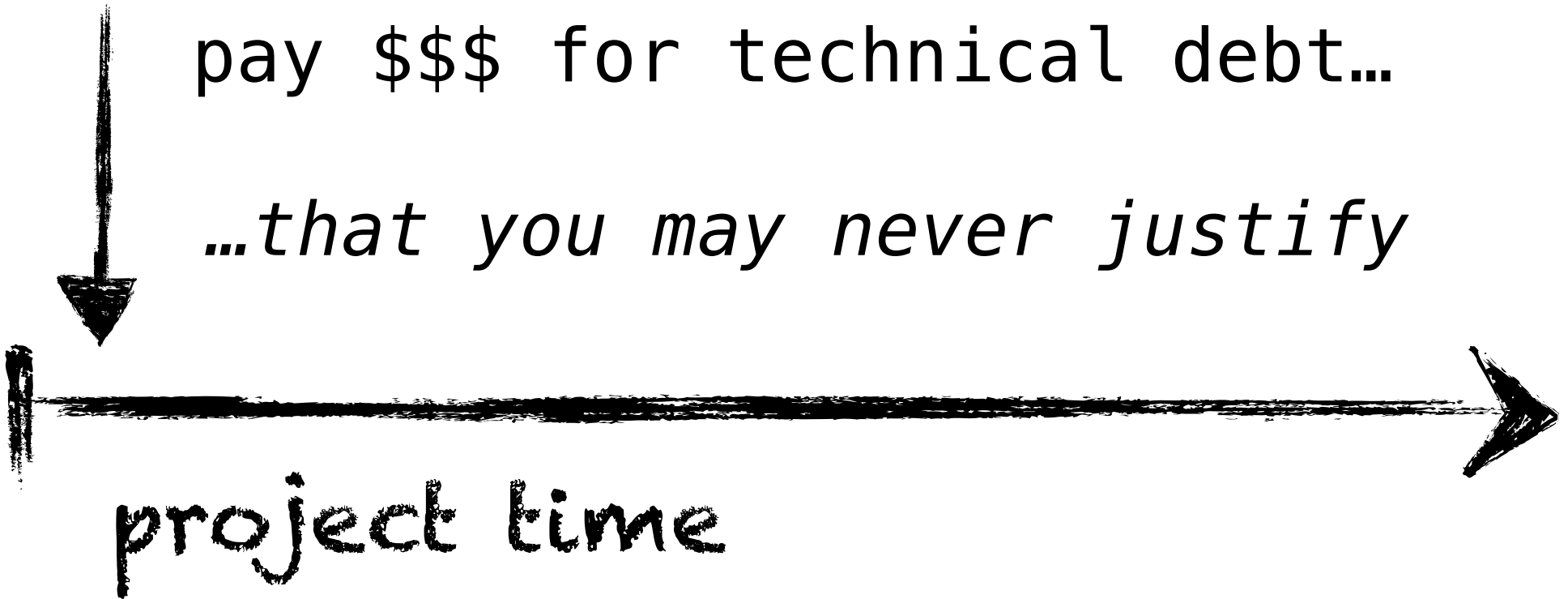


don't know what we don't know

“buy the fanciest one we
can” (just in case)

pay \$\$\$ for technical debt...

...that you may never justify



messaging
infrastructure

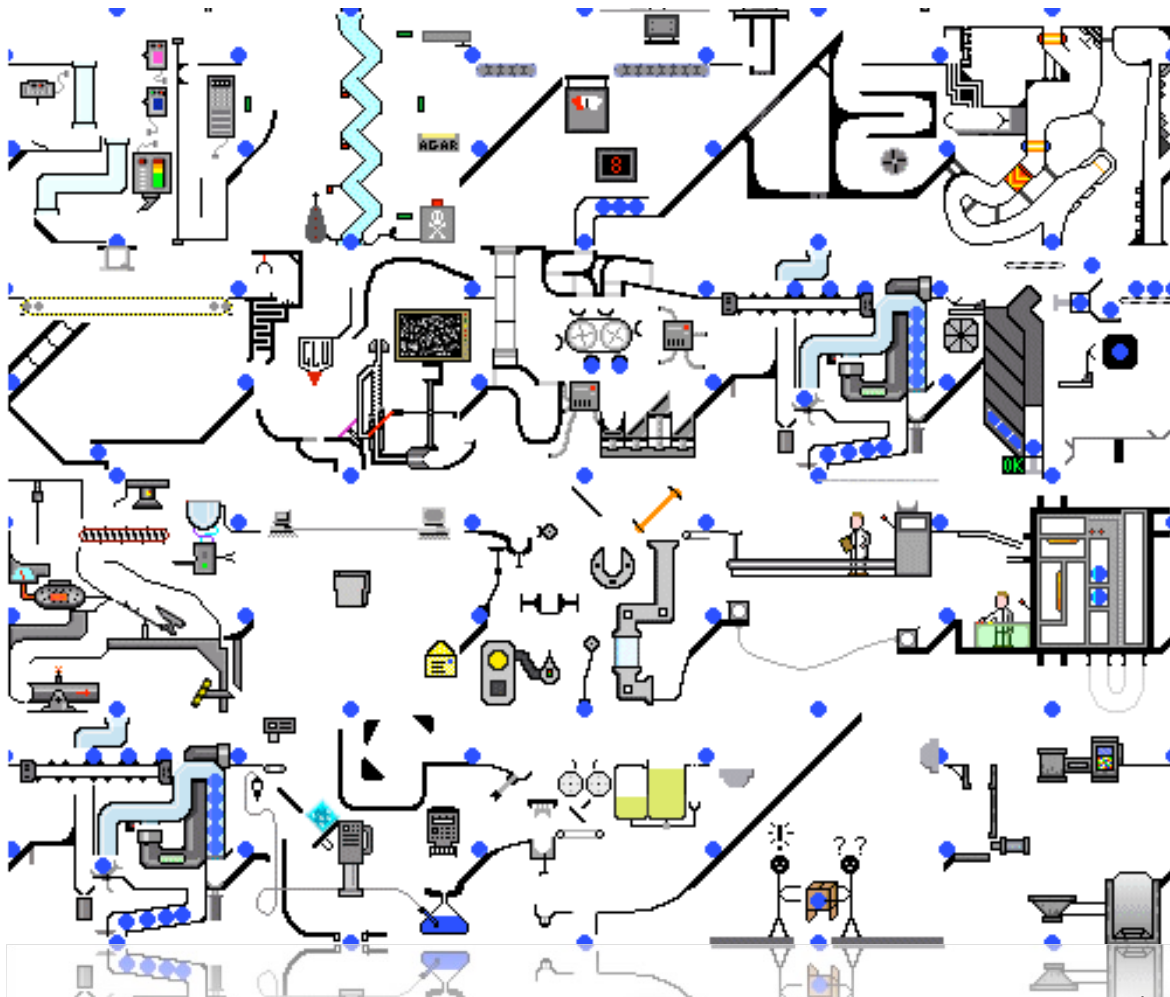
convert architectural
elements to design

A small green seedling with two leaves is growing out of a crack in dry, cracked earth. The seedling is positioned in the center of the frame, with its stem and leaves clearly visible. The earth is a light brown color and is covered in a network of dark, irregular cracks. The lighting is soft, highlighting the texture of the soil and the vibrant green of the plant.

Summary

evolution &
emergence require
good engineering
practices





trying to predict the
future leads to over-
engineering



prefer pro/re-active
to
predictive

?'S

please fill out the session evaluations



This work is licensed under the Creative Commons Attribution-Share Alike 3.0 License.

<http://creativecommons.org/licenses/by-sa/3.0/us/>

NEAL FORD software architect / meme wrangler

ThoughtWorks®

nford@thoughtworks.com
2002 Summit Boulevard, Atlanta, GA 30319
www.nealford.com
www.thoughtworks.com
blog: memeagora.blogspot.com
twitter: neal4d