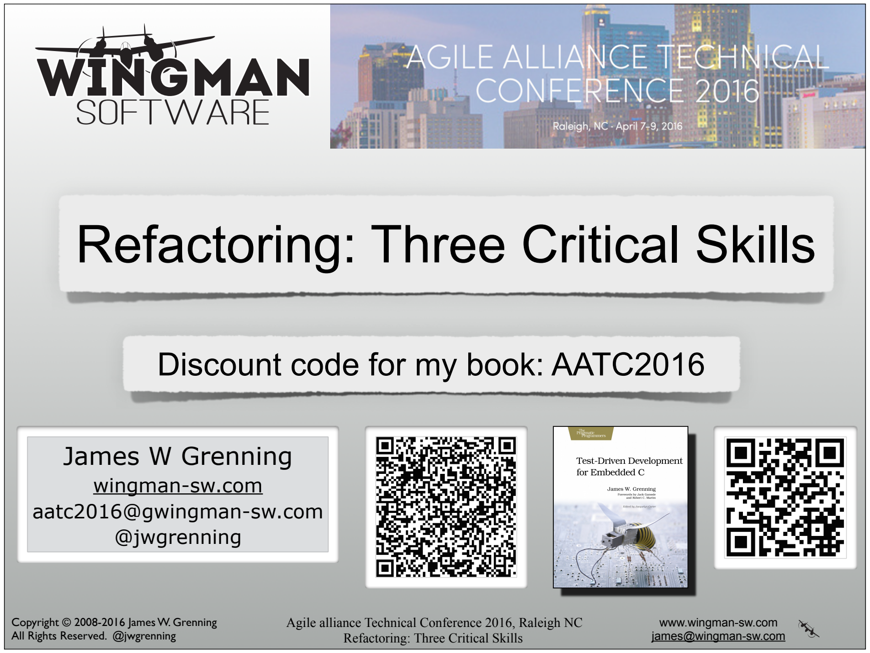


WINGMAN SOFTWARE

AGILE ALLIANCE TECHNICAL CONFERENCE 2016

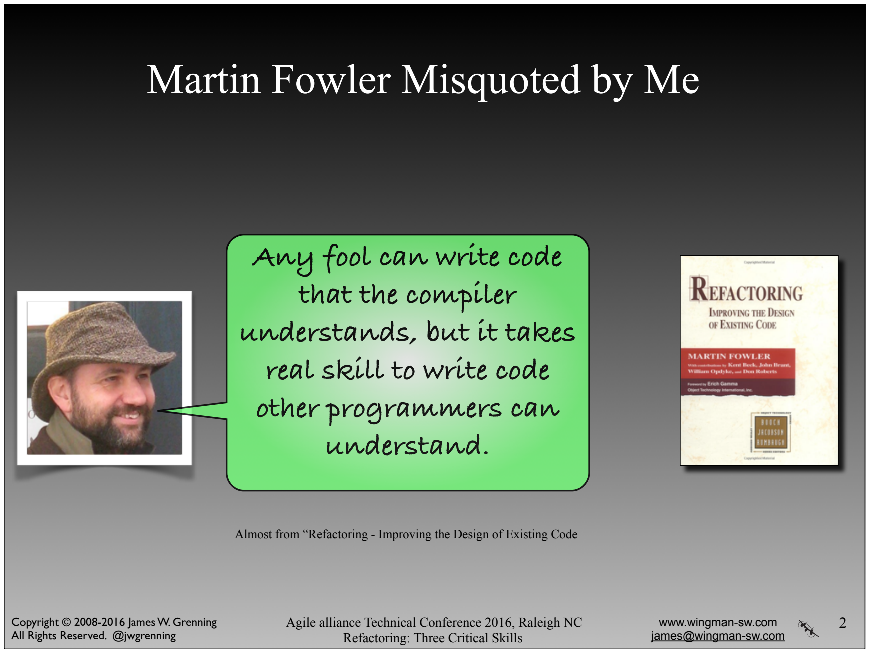
Raleigh, NC - April 7-9, 2016

Refactoring: Three Critical Skills

Discount code for my book: AATC2016

James W Grenning
wingman-sw.com
aatc2016@gwingman-sw.com
@jwgrenning

Test-Driven Development for Embedded C
James W. Grenning

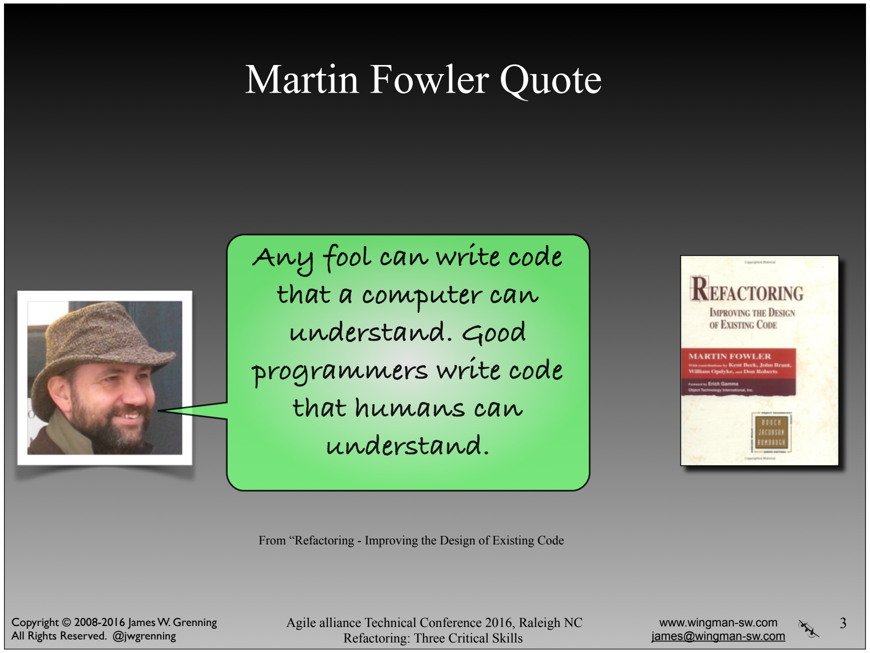


Martin Fowler Misquoted by Me

Any fool can write code that the compiler understands, but it takes real skill to write code other programmers can understand.

REFACTORING
IMPROVING THE DESIGN OF EXISTING CODE
MARTIN FOWLER

Almost from "Refactoring - Improving the Design of Existing Code

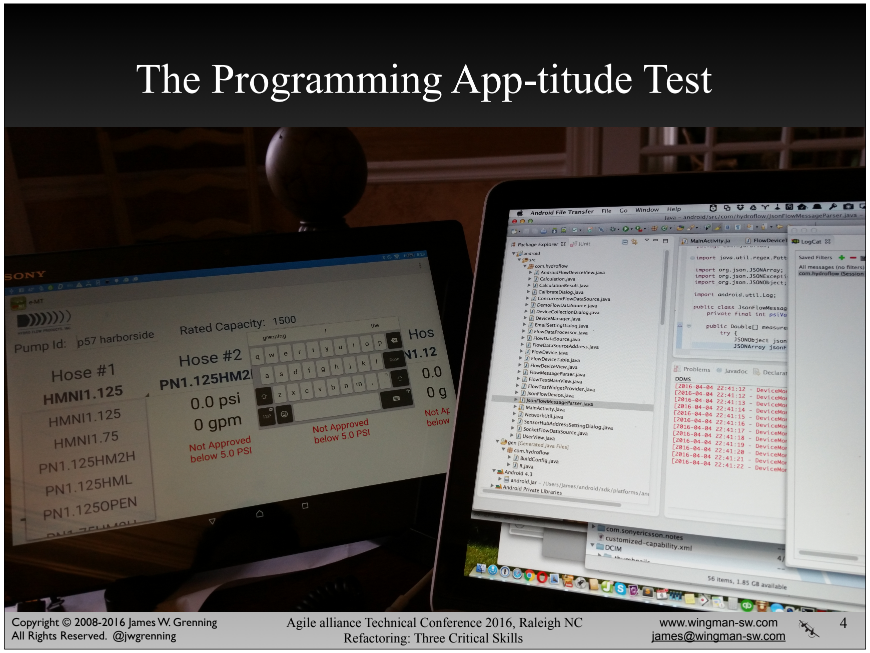


Martin Fowler Quote

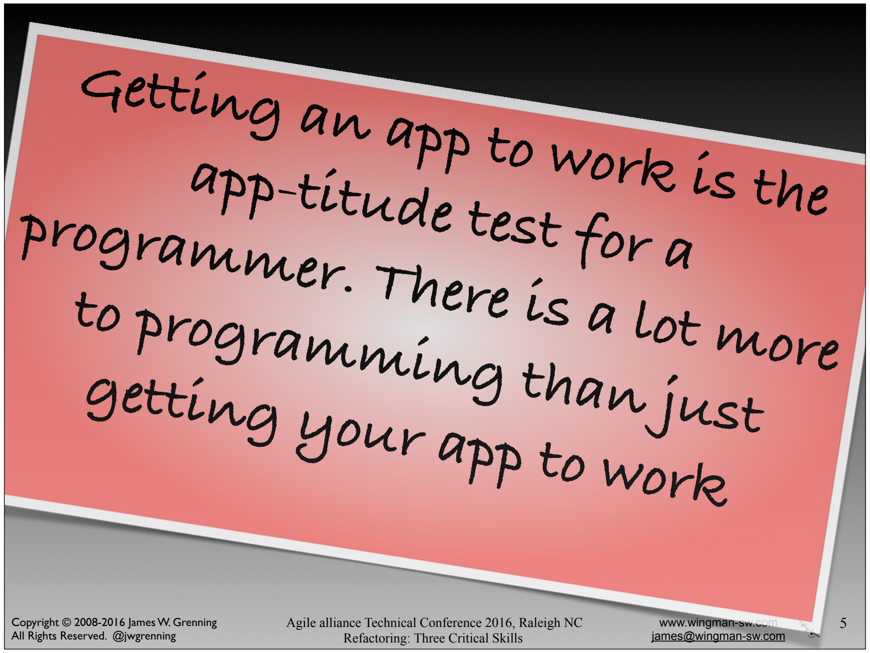
Any fool can write code that a computer can understand. Good programmers write code that humans can understand.

REFACTORING
IMPROVING THE DESIGN OF EXISTING CODE
MARTIN FOWLER

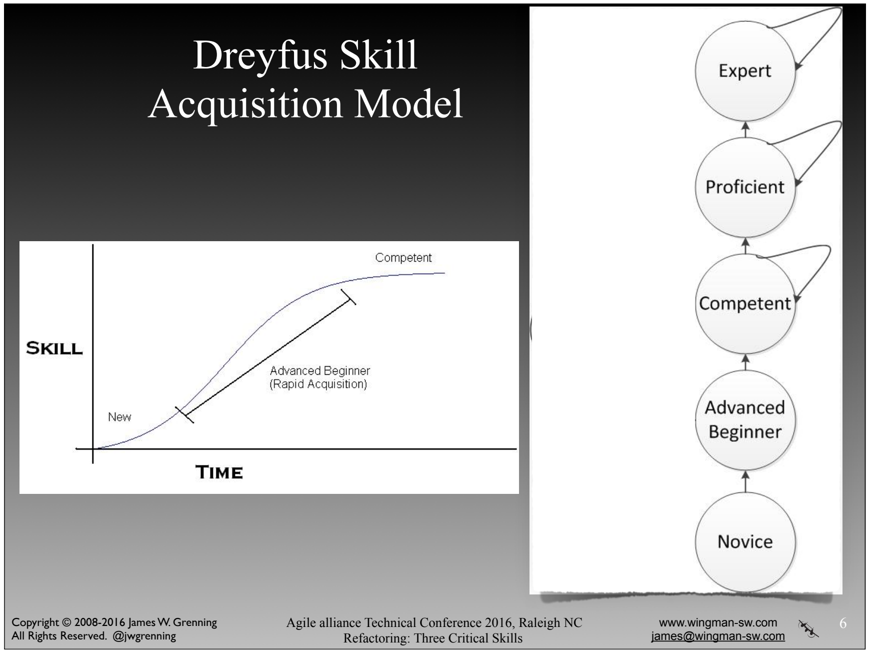
From "Refactoring - Improving the Design of Existing Code



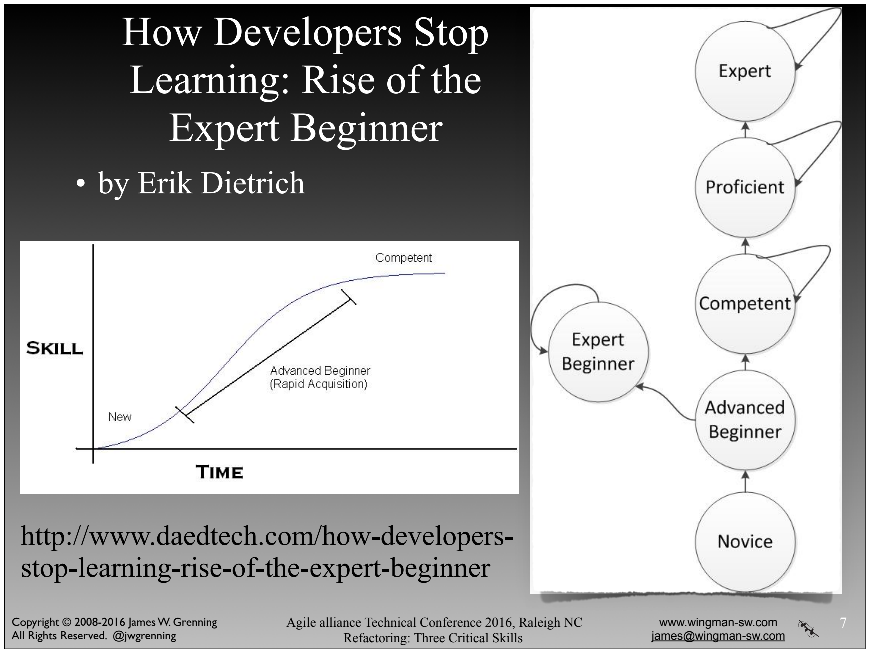
The Programming App-titude Test



Getting an app to work is the app-titude test for a programmer. There is a lot more to programming than just getting your app to work



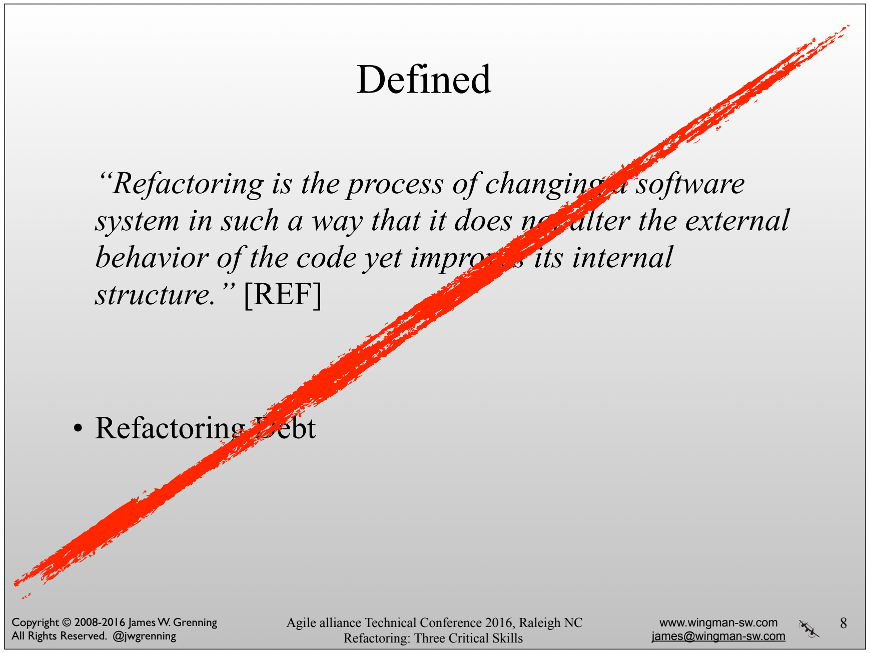
Dreyfus Skill Acquisition Model



How Developers Stop Learning: Rise of the Expert Beginner

- by Erik Dietrich

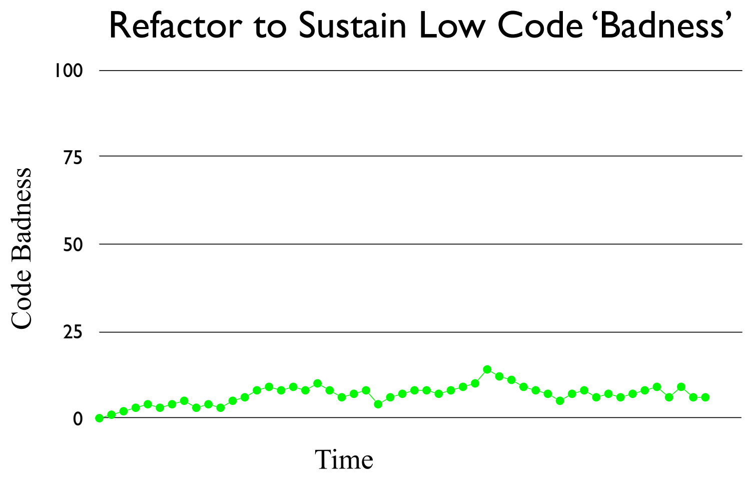
http://www.daedtech.com/how-developers-stop-learning-rise-of-the-expert-beginner

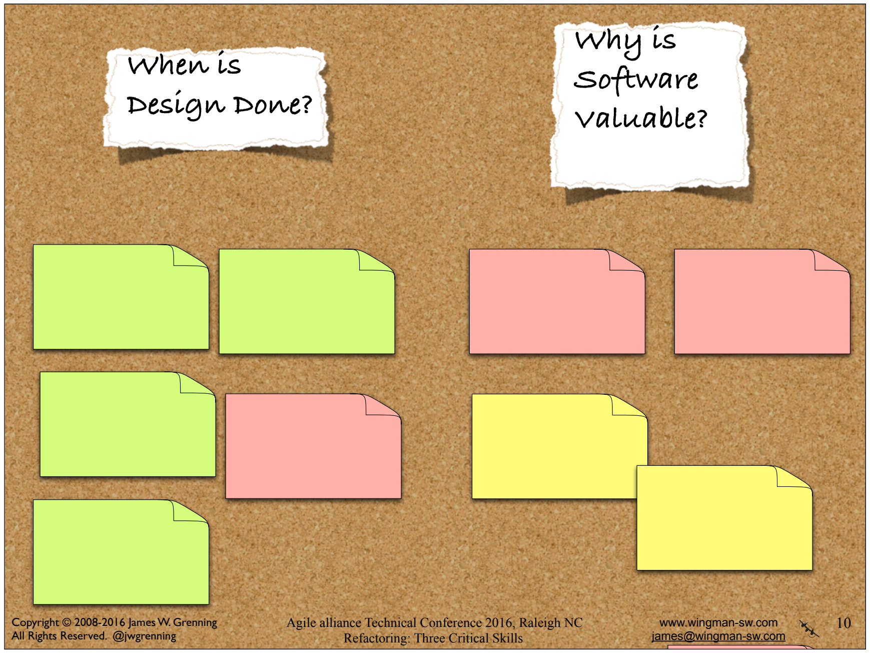


Defined

~~*"Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure."* [REF]~~

- ~~Refactoring Debt~~

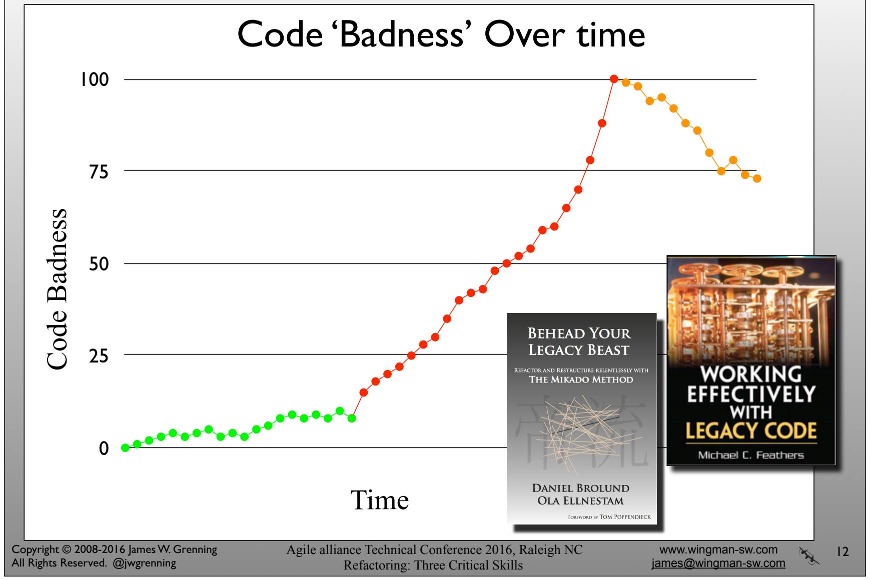




When is Design Done?

Why is Software Valuable?

The Two Values of Software



Defined Again

“Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure.” [REF]

- Refactoring Debt

Refactoring Goal

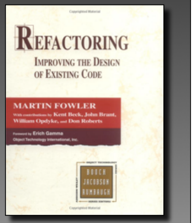
- “Clean code that works”
  - Ron Jeffries

How?

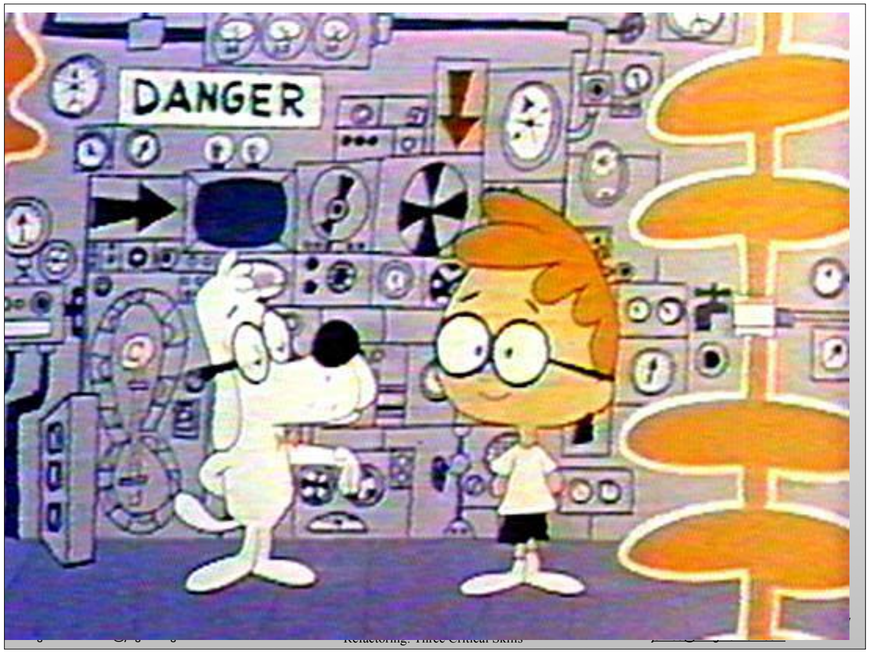
- Remove duplication
- Fix bad names
  - J.B. Rainsberger

I Say Martin Fowler Says

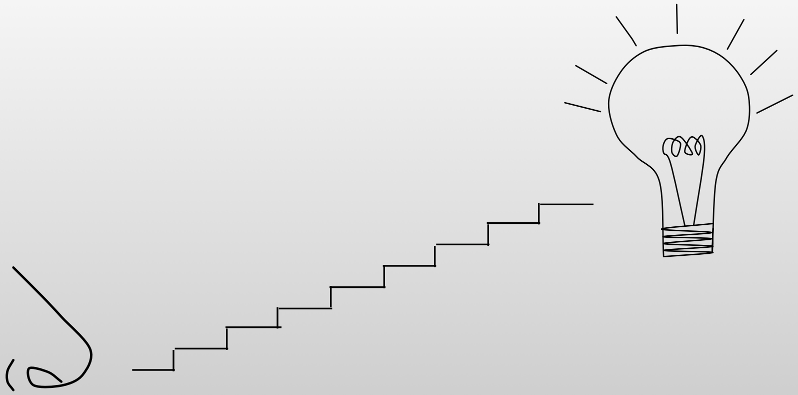
Any fool can write code that the compiler understands, but it takes real skill to write code other programmers can understand.



Almost from “Refactoring - Improving the Design of Existing Code



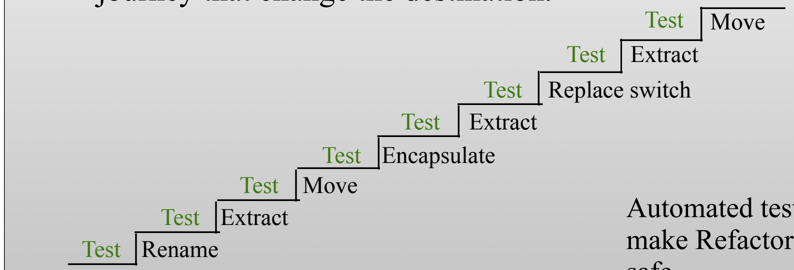
Three Critical Skills



Recognize what is wrong and fix it!

Refactorings

- Follow a course that keeps all test running as you move the design closer to the envisioned solution. Keep in mind, you are likely to see things on the journey that change the destination.

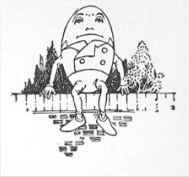


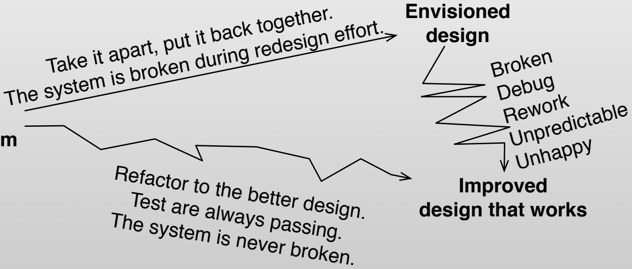
Automated tests make Refactoring safe.

Don’t Burn Bridges

It is easier to keep code running.
Than to fix it after you break it.

Refactoring vs.
Humpty Dumpty Redesign





Code with design problem

Take it apart, put it back together.
The system is broken during redesign effort.

Envisioned design

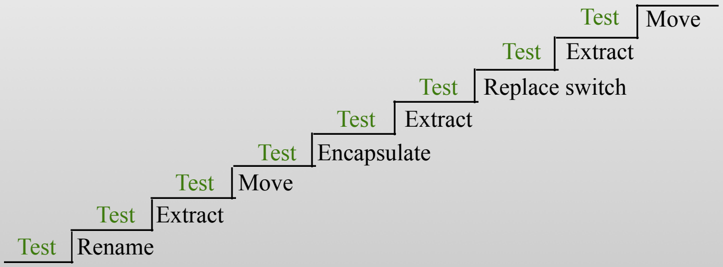
Broken
Debug
Rework
Unpredictable
Unhappy

Refactor to the better design.
Test are always passing.
The system is never broken.

Improved design that works

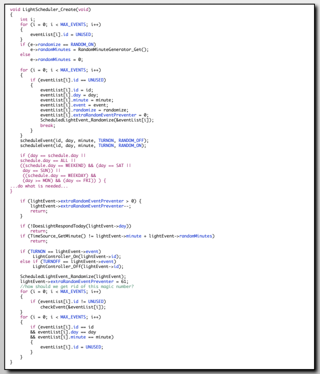
Humpty Dumpty from The Book of Knowledge,
p. 968, Vol. III, The Grolier Society, New York, 1911.

Step by Step Refactoring is a Skill to Master



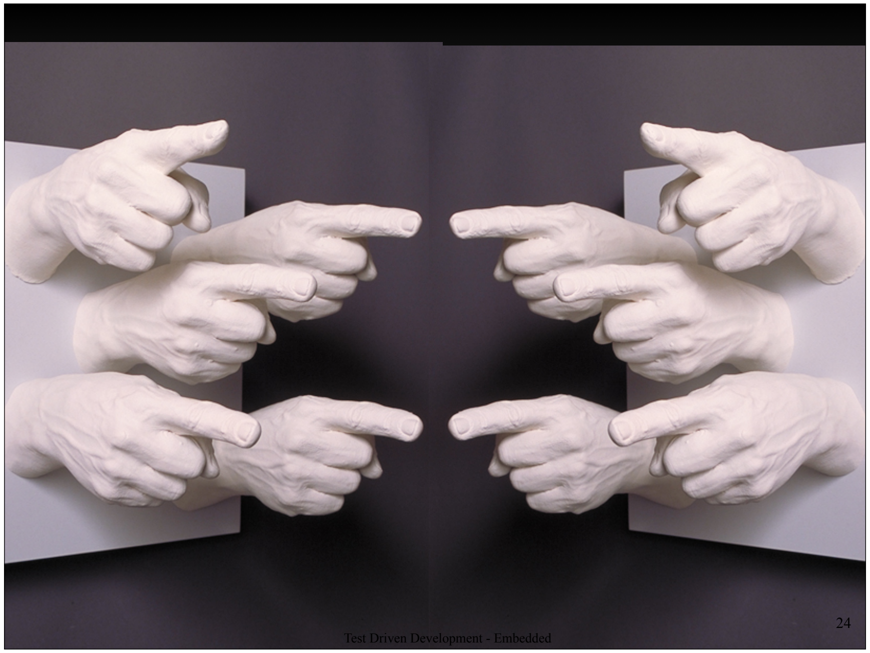
Test Rename
Test Extract
Test Move
Test Encapsulate
Test Extract
Test Replace switch
Test Extract
Test Move

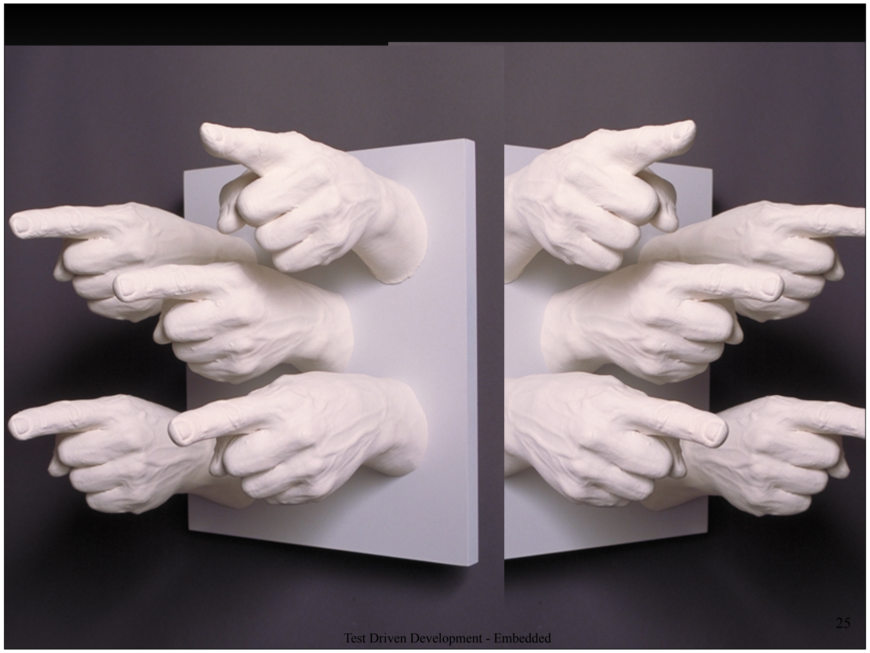
This Code Stinks



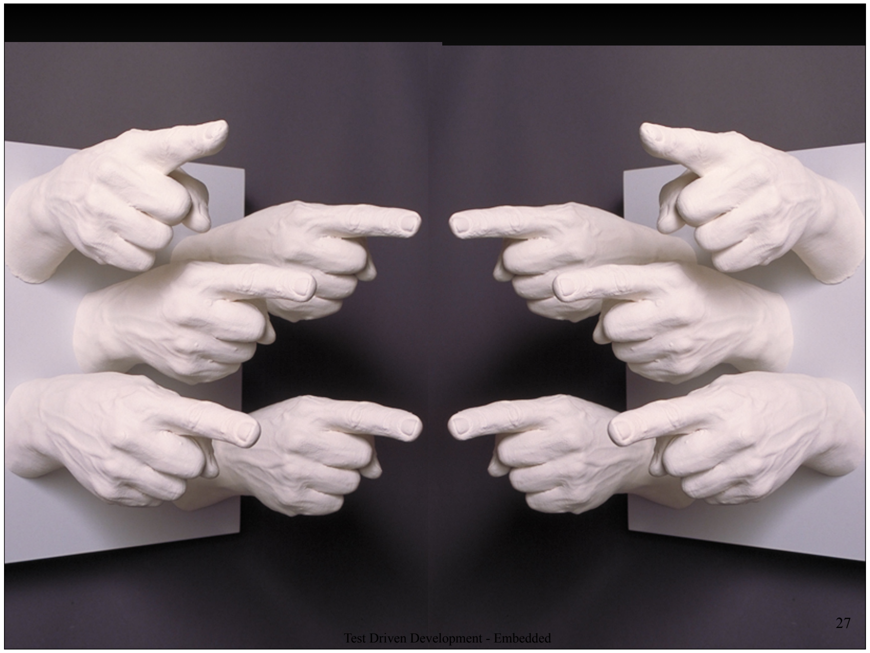
```
void LightScheduler_Create(void)
{
    int i;
    for (i = 0; i < MAX_EVENTS; i++)
    {
        eventList[i].id = UNUSED;
    }
    if (e->randomize == RANDOM_ON)
        e->randomMinutes = RandomMinuteGenerator_Get();
    else
        e->randomMinutes = 0;

    for (i = 0; i < MAX_EVENTS; i++)
    {
        if (eventList[i].id == UNUSED)
        {
            eventList[i].id = id;
            eventList[i].day = day;
            eventList[i].minute = minute;
            eventList[i].event = event;
            eventList[i].randomize = randomize;
            eventList[i].extraRandomEventPreventer = 0;
            ScheduledLightEvent_Randomize(&eventList[i]);
            break;
        }
    }
    scheduleEvent(id, day, minute, TURNON, RANDOM_OFF);
    scheduleEvent(id, day, minute, TURNON, RANDOM_ON);

    if (day == schedule.day ||
    schedule.day == ALL ||
    ((schedule.day == WEEKEND) && (day == SAT ||
     day == SUN)) ||
     ((schedule.day == WEEKDAY) &&
     (day >= MON) && (day <= FRI)) ) {
...do what is needed...
}

    if (lightEvent->extraRandomEventPreventer > 0) {
        lightEvent->extraRandomEventPreventer--;
        return;
    }

    if (!DoesLightRespondToday(lightEvent->day))
        return;
    if (TimeSource_GetMinute() != lightEvent->minute + lightEvent->randomMinutes)
        return;

    if (TURNON == lightEvent->event)
         LightController_On(lightEvent->id);
    else if (TURNOFF == lightEvent->event)
         LightController_Off(lightEvent->id);

    ScheduledLightEvent_Randomize(lightEvent);
    lightEvent->extraRandomEventPreventer = 61;
    //how should we get rid of this magic number?
    for (i = 0; i < MAX_EVENTS; i++)
    {
        if (eventList[i].id != UNUSED)
            checkEvent(&eventList[i]);
    }
    for (i = 0; i < MAX_EVENTS; i++)
    {
        if (eventList[i].id == id
        && eventList[i].day == day
        && eventList[i].minute == minute)
        {
            eventList[i].id = UNUSED;
        }
    }
}
```

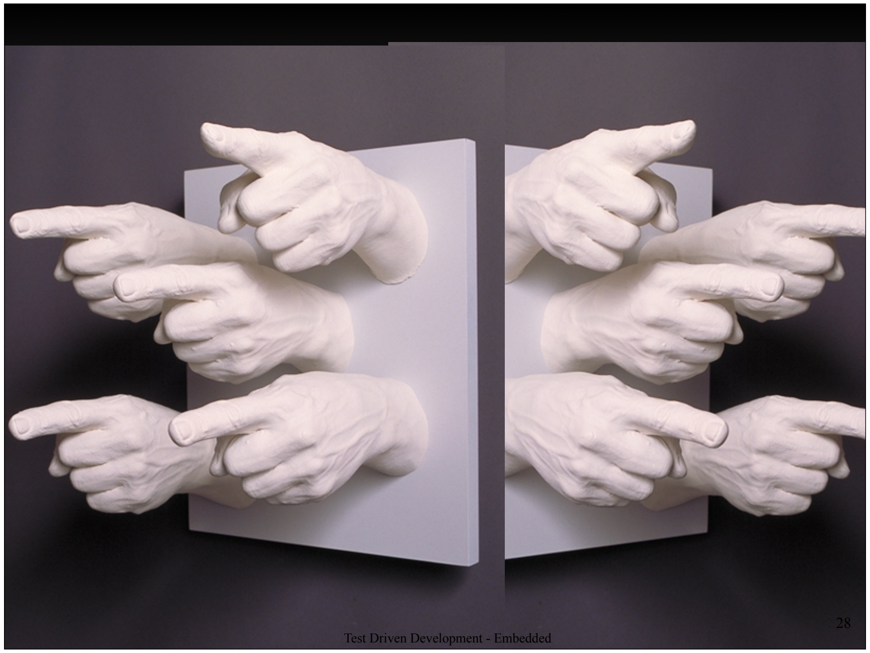
Is not a good nose.





Repeat After Me





Why Did You Do It?

"We don't have time for refactoring, there's still too much left to do."

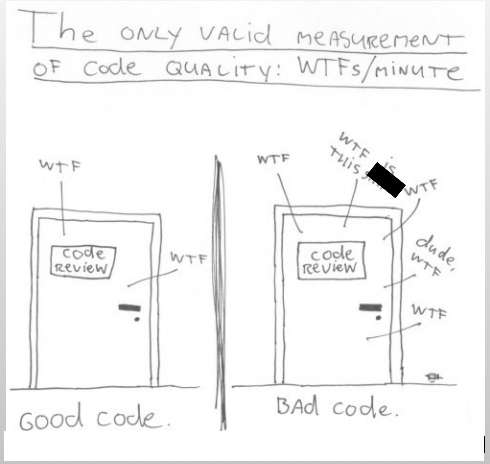
Reply
Retweet
Favorite
More
Are you too busy to improve?
No thanks!
We are too busy
Created by Håkan Forss @hakanforss http://hakanforss.wordpress.com

Refactoring Goal

- "Clean code that works"
  - Ron Jeffries

What's That?!

NIH vs. SOLID



Smells Found in Object Oriented Code [REF]

- Long method
- Large class
- Feature envy
- Duplicate code
- Inappropriate intimacy
- Refused bequest
- Lazy class
- Comments
- Contrived Complexity
- Long parameter list
- Primitive obsession

Smells Found in C Code

- Duplicate code
- Bad Pasta
- Long function
- Cryptic names
- Abstraction Distraction
- Switch case disgrace
- Bewildering Boolean
- Nefarious Nesting
- Long parameter list
- Willy-nilly initialization
- Global free-for-all
- Primitive obsession
- Comments
- Dead code

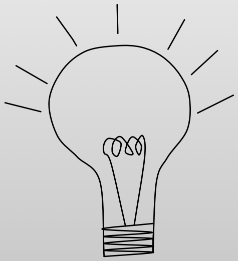
What is Well Commented Code?

Ottinger's Comment Rules

- Comment Rule: Comments are for things that cannot be expressed in code.
- Comment Redundancy Rule: Comments that restate the code must be deleted.
- Comment Single Truth Rule: If the comment says what the code could say, then the code must change to make the comment redundant.

Envisioning

- Once a design problem is identified, you must envision a better solution
- Look to apply design principles
  - – SOLID
  - – Single Responsibility
  - – Substitutability
  - – DRY - Don't Repeat Yourself
    - –from The Pragmatic Programmer
  - – Hexagonal Architecture
  - – Principle of Least Knowledge.
  - – Separation of Concerns (SoC).
  - – Domain Driven Design
  - – Rules of Simple Design



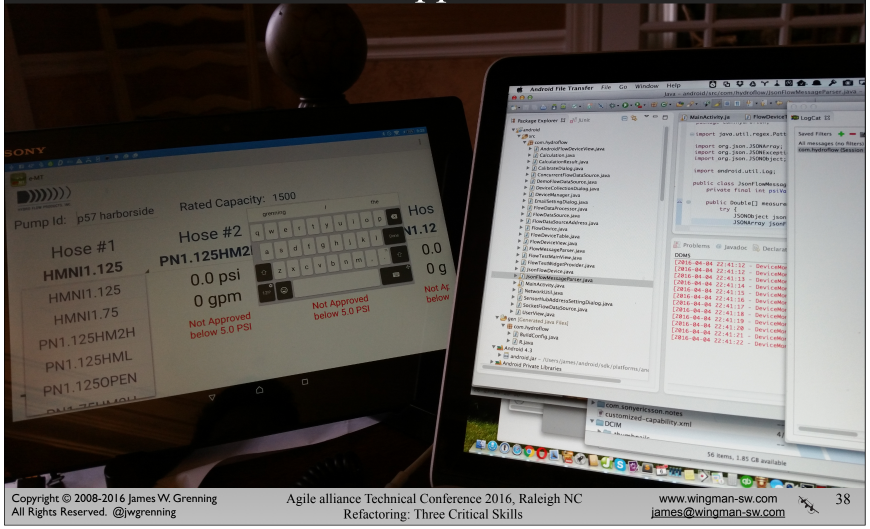
Rules of Simple Design

In Priority Order!

1. Passes all tests
2. No duplication
3. Expresses intent
4. Fewest classes and methods (no extra stuff)

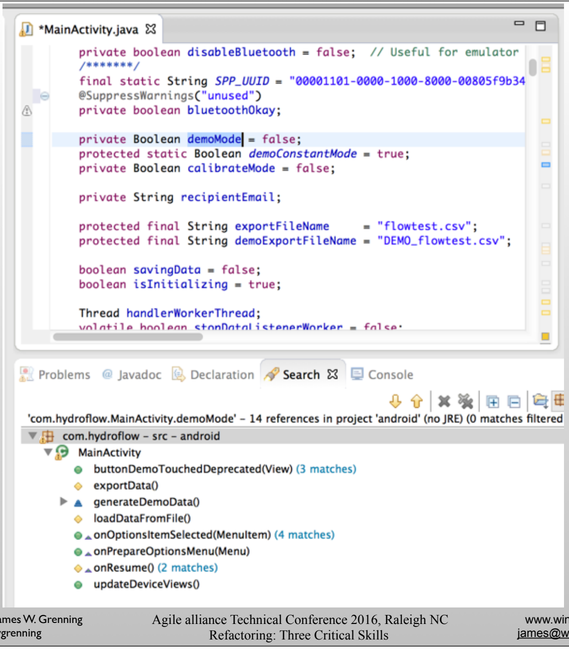
Kent Beck [XP, TDD]

It Works!
Passed the App-titude Test



Large Class with Muddled Responsibilities Requires Shotgun Surgery

https://commons.wikimedia.org/wiki/User:DiverDave



```
    private boolean disableBluetooth = false;  // Useful for emulator
    /*******/
    final static String SPP_UUID = "00001101-0000-1000-8000-00805f9b34
    @SuppressWarnings("unused")
    private boolean bluetoothOkay;

    private Boolean demoMode = false;
    protected static Boolean demoConstantMode = true;
    private Boolean calibrateMode = false;

    private String recipientEmail;

    protected final String exportFileName     = "flowtest.csv";
    protected final String demoExportFileName = "DEMO_flowtest.csv";

    boolean savingData = false;
    boolean isInitializing = true;

    Thread handlerWorkerThread;
    volatile boolean stopDataListenerWorker = false;
```

```
'com.hydroflow.MainActivity.demoMode' - 14 references in project 'android' (no JRE) (0 matches filtered
▼ com.hydroflow - src - android
  ▼ MainActivity
      buttonDemoTouchedDeprecated(View) (3 matches)
      exportData()
    ▶ generateDemoData()
      loadDataFromFile()
      onOptionsItemSelected(MenuItem) (4 matches)
      onPrepareOptionsMenu(Menu)
      onResume() (2 matches)
      updateDeviceViews()
```

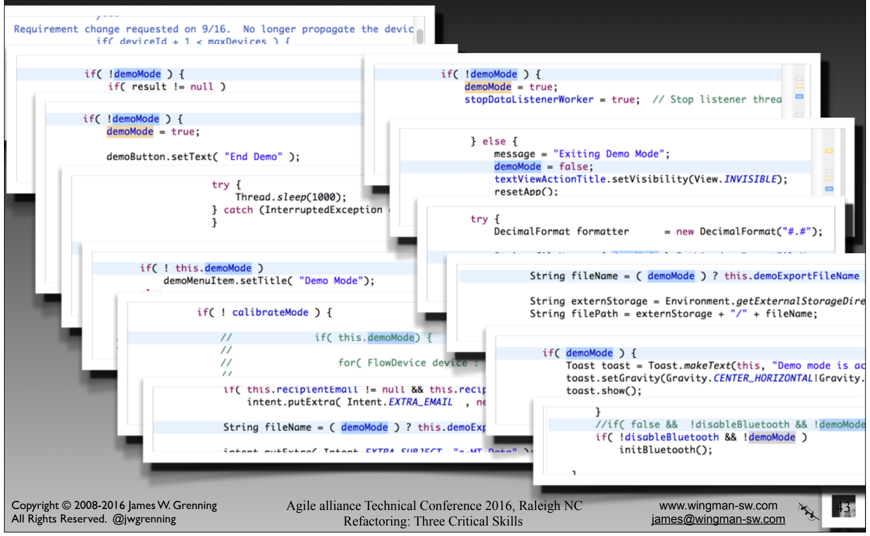
Add WiFi Data Source

- Bluetooth and Demo Data is already supported.
- It should be pretty east, right?

MainActivity.java Metrics

- 1934 Lines of code
- 30 references to names containing ‘bluetooth’
- 18 references to names containing ‘demoMode’

How Many Details Can we Keep



```
Requirement change requested on 9/16.  No longer propagate the devic
        if( deviceId + 1 < maxDevices ) {
```

```
        if( !demoMode ) {
            if( result != null )
```

```
        if( !demoMode ) {
            demoMode = true;

            demoButton.setText( "End Demo" );
```

```
                    try {
                        Thread.sleep(1000);
                    } catch (InterruptedException
                    }
```

```
        if( ! this.demoMode )
            demoMenuItem.setTitle( "Demo Mode");
```

```
            if( ! calibrateMode ) {

    //                  if( this.demoMode) {
    //
    //                      for( FlowDevice device :
```

```
            if( this.recipientEmail != null && this.recip
                intent.putExtra( Intent.EXTRA_EMAIL   , ne

            String fileName = ( demoMode ) ? this.demoExp
```

```
        if( !demoMode ) {
            demoMode = true;
            stopDataListenerWorker = true;  // Stop listener threa
```

```
        } else {
            message = "Exiting Demo Mode";
            demoMode = false;
            textViewActionTitle.setVisibility(View.INVISIBLE);
            resetApp();
```

```
        try {
            DecimalFormat formatter      = new DecimalFormat("#.#");
```

```
            String fileName = ( demoMode ) ? this.demoExportFileName

            String externStorage = Environment.getExternalStorageDire
            String filePath = externStorage + "/" + fileName;
```

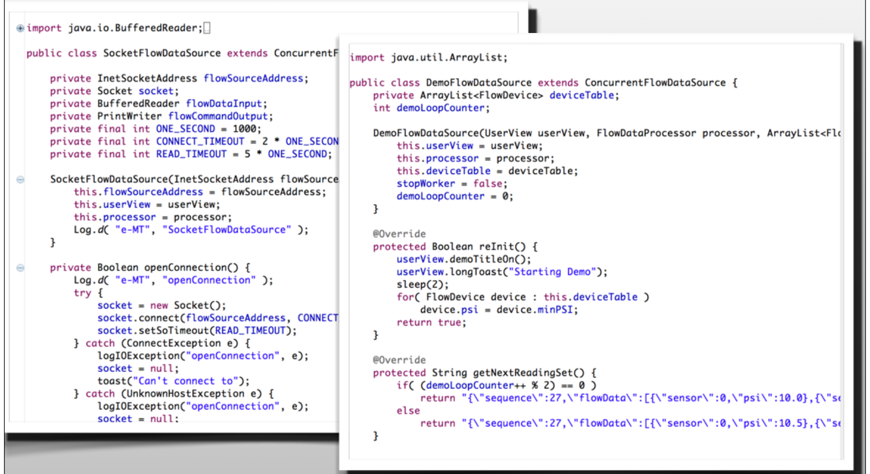
```
        if( demoMode ) {
            Toast toast = Toast.makeText(this, "Demo mode is ac
            toast.setGravity(Gravity.CENTER_HORIZONTAL|Gravity.
            toast.show();
```

```
            }
            //if( false &&  !disableBluetooth && !demoMode
            if( !disableBluetooth && !demoMode )
                initBluetooth();
```

Rely on Cause and Effect

- Small changes
- Tested each step of the way

Separation of Concerns

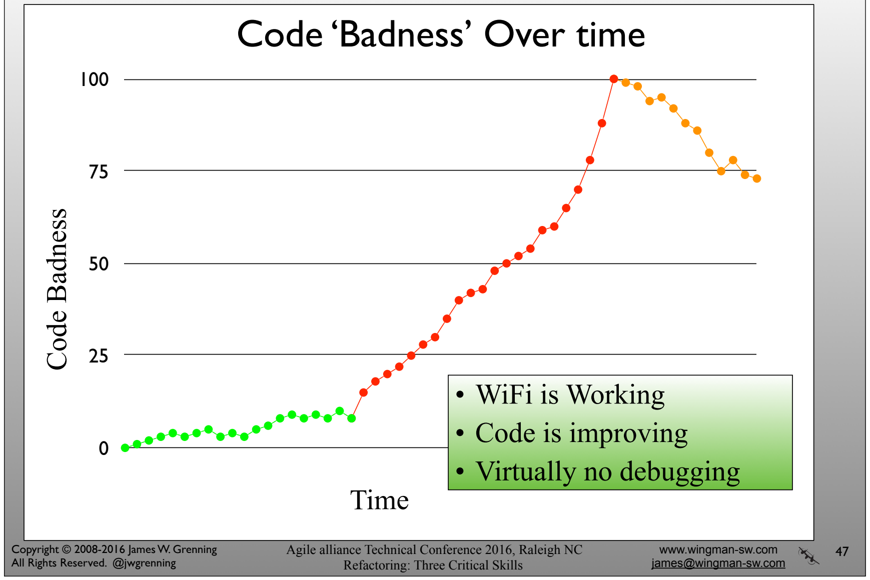


```
import java.io.BufferedReader;

public class SocketFlowDataSource extends ConcurrentF

    private InetSocketAddress flowSourceAddress;
    private Socket socket;
    private BufferedReader flowDataInput;
    private PrintWriter flowCommandOutput;
    private final int ONE_SECOND = 1000;
    private final int CONNECT_TIMEOUT = 2 * ONE_SECON
    private final int READ_TIMEOUT = 5 * ONE_SECOND;

    SocketFlowDataSource(InetSocketAddress flowSource
        this.flowSourceAddress = flowSourceAddress;
        this.userView = userView;
        this.processor = processor;
        Log.d( "e-MT", "SocketFlowDataSource" );
    }

    private Boolean openConnection() {
        Log.d( "e-MT", "openConnection" );
        try {
            socket = new Socket();
            socket.connect(flowSourceAddress, CONNECT
            socket.setSoTimeout(READ_TIMEOUT);
        } catch (ConnectException e) {
            logIOException("openConnection", e);
            socket = null;
            toast("Can't connect to");
        } catch (UnknownHostException e) {
            logIOException("openConnection", e);
            socket = null;
```

```
import java.util.ArrayList;

public class DemoFlowDataSource extends ConcurrentFlowDataSource {
    private ArrayList<FlowDevice> deviceTable;
    int demoLoopCounter;

    DemoFlowDataSource(UserView userView, FlowDataProcessor processor, ArrayList<Flo
        this.userView = userView;
        this.processor = processor;
        this.deviceTable = deviceTable;
        stopWorker = false;
        demoLoopCounter = 0;
    }

    @Override
    protected Boolean reInit() {
        userView.demoTitleOn();
        userView.longToast("Starting Demo");
        sleep(2);
        for( FlowDevice device : this.deviceTable )
            device.psi = device.minPSI;
        return true;
    }

    @Override
    protected String getNextReadingSet() {
        if( (demoLoopCounter++ % 2) == 0 )
            return "{\"sequence\":27,\"flowData\":[{\"sensor\":0,\"psi\":10.0},{\"se
        else
            return "{\"sequence\":27,\"flowData\":[{\"sensor\":0,\"psi\":10.5},{\"se
    }
```

Improving MainActivity.java Metrics

- 1213 Lines of code (down from 1934)
- 8 references to names containing 'socket'
  - Was 30 references to names containing 'bluetooth'
- 9 references to names containing 'demoMode'
  - Was 18 references to names containing 'demoMode'



Code 'Badness' Over time

- WiFi is Working
- Code is improving
- Virtually no debugging

Don't
Repeat
Yourself

- Is this about code duplication?
- It's about idea representation duplication!

C LightScheduler TEST Before Refactoring

```
TEST(LightScheduler, no_lights_controlled_when_its_not_the_scheduled_time)
{
    LightScheduler_AddTurnOn(3, EVERYDAY, 1200);
    FakeTimeService_SetDay(SUNDAY);
    FakeTimeService_SetMinute(1199);
    LightScheduler_Wakeup();
    LONGS_EQUAL(NO_LIGHT_ID, LightControllerSpy_GetLastId());
    LONGS_EQUAL(NO_LIGHT_STATE, LightControllerSpy_GetLastState());
}

TEST(LightScheduler, light_turns_on_at_the_scueduled_time_for_everyday)
{
    LightScheduler_AddTurnOn(3, EVERYDAY, 1200);
    FakeTimeService_SetDay(SUNDAY);
    FakeTimeService_SetMinute(1200);
    LightScheduler_Wakeup();
    LONGS_EQUAL(3, LightControllerSpy_GetLastId());
    LONGS_EQUAL(LIGHT_ON, LightControllerSpy_GetLastState());
}
```

C LightScheduler TEST with Code Duplication Removed

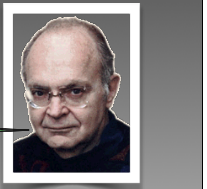
```
TEST(LightScheduler, no_lights_controlled_when_its_not_the_scheduled_time)
{
    WhatTheHeckDoICallThisFunciton(3, EVERYDAY, 1200,
                                   SUNDAY, 1199,
                                   NO_LIGHT_ID,NO_LIGHT_STATE);
}

TEST(LightScheduler, light_turns_on_at_the_scueduled_time_for_everyday)
{
    WhatTheHeckDoICallThisFunciton(3, EVERYDAY, 1200,
                                   SUNDAY, 1199,
                                   3,LIGHT_ON);
}
```

C LightScheduler TEST with 'Idea' Duplication Removed

```
TEST(LightScheduler, no_lights_controlled_when_its_not_the_scheduled_time)
{
    LightScheduler_AddTurnOn(3, EVERYDAY, 1200);
    TransitionClockTo(SUNDAY, 1199);
    LIGHTS_ARE_UNCHANGED;
}

TEST(LightScheduler, light_turns_on_at_the_scueduled_time_for_everyday)
{
    LightScheduler_AddTurnOn(3, EVERYDAY, 1200);
    TransitionClockTo(SUNDAY, 1200);
    THEN_LIGHT_IS_ON(3);
}
```

Donald Knuth Says

Let us change our traditional attitude to the construction of programs. Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.



Named from the Writer's Perspective

```
void SetTimeAndWakeup(int day, int minute)
{
    FakeTimeService_SetDay(day);
    FakeTimeService_SetMinute(minute);
    LightScheduler_Wakeup();
}

TEST(LightScheduler, light_turns_on_at_the_scueduled_time_for_everyday)
{
    LightScheduler_AddTurnOn(3, EVERYDAY, 1200);
    SetTimeAndWakeup(SUNDAY, 1200);
    THEN_LIGHT_IS_ON(3);
}
```

Named from the Reader's Perspective

```
void ClockChangesTo(int day, int minute)
{
    FakeTimeService_SetDay(day);
    FakeTimeService_SetMinute(minute);
    LightScheduler_Wakeup();
}

TEST(LightScheduler, light_turns_on_at_the_scueduled_time_for_everyday)
{
    LightScheduler_AddTurnOn(3, EVERYDAY, 1200);
    ClockChangesTo(SUNDAY, 1200);
    THEN_LIGHT_IS_ON(3);
}
```

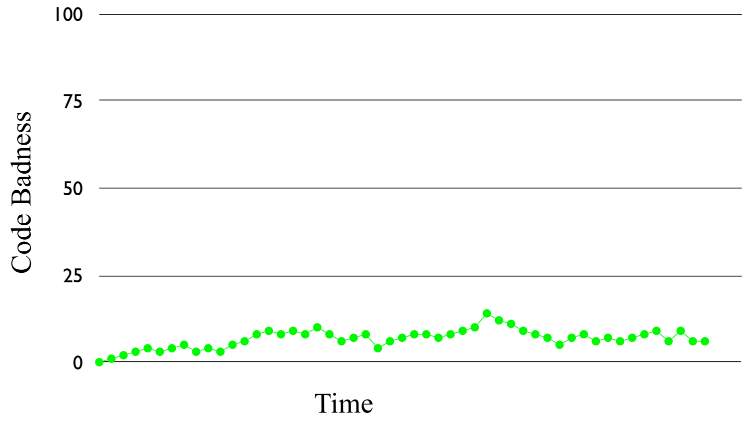
TDD - the Code Rot Radar

- As code becomes opaque, it resists tests.
- Lack of testability provides an early warning of impending design problems.
- Keeping the code clean, before it gets too messy, keeps the cost of refactoring low.

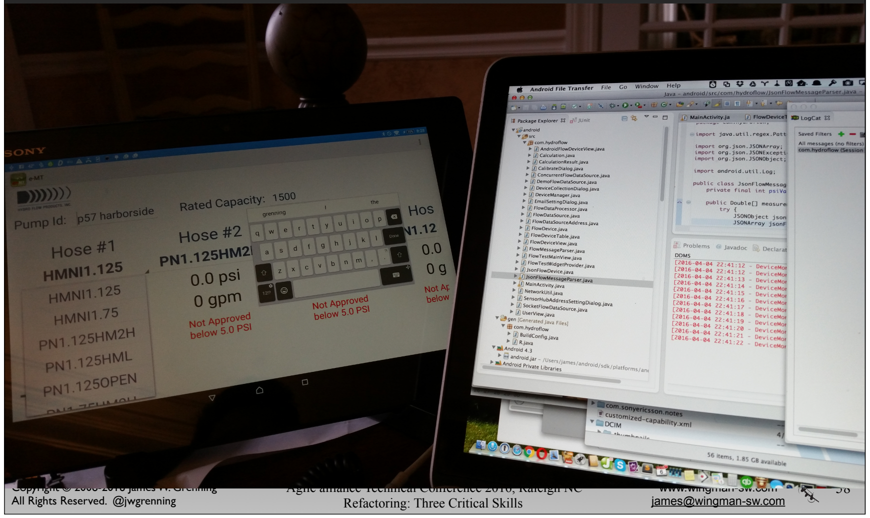
I guess the title of this talk is wrong There is a Forth Skill…

- Unit testing
- Test-Driven development

Refactor to Sustain Low Code 'Badness'



Programmers!
Get Past the App-titude Test



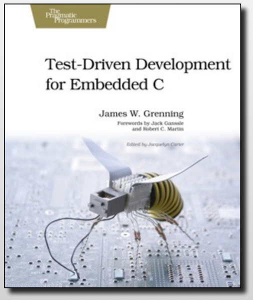
The Three Skills of Refactoring

Talk to me on Twitter
@jwgrenning

Find my book at
http://wingman-sw.com/tddec

Find me on linkedin.com
http://www.linkedin.com/in/jwgrenning
Please remind me how we met.

http://facebook.com/wingman.sw
http://www.wingman-sw.com
http://www.wingman-sw.com/blog



The Pragmatic Programmers

Test-Driven Development for Embedded C

James W. Grenning

Forewords by Jack Ganssle and Robert C. Martin

Edited by Jacquelyn Carter