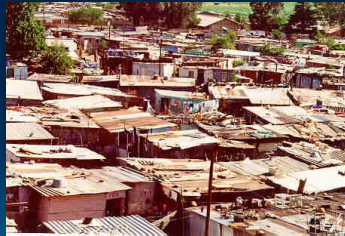


Big Balls of Mud

"Can we Avoid Them?"



Joseph W. Yoder -- www.refactory.com

Copyright 2011 Joseph W. Yoder & The Refactory, Inc.

Evolved from UIUC SAG

In the early 90's we were studying objects, frameworks, components, reusability, patterns, "good" architecture

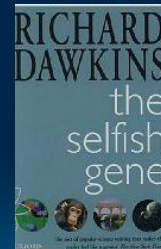
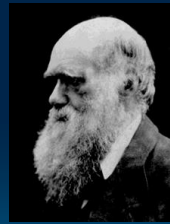


However, in our SAG group we often noticed that although we talk a good game, many successful systems do not have a good internal structure at all!

Escape From The Spaghetti Code Jungle

Selfish Class

Brian and I had just published a paper called Selfish Class which takes a *code's-eye view of software reuse and evolution*.



In contrast, our BBoM paper noted that in reality, a lot of code was hard to (re)-use.

Escape From The Spaghetti Code Jungle

Big Ball of Mud

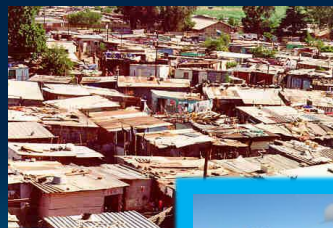
Alias: Shantytown, Spaghetti Code



A BIG BALL OF MUD is haphazardly structured, sprawling, sloppy, duct-tape and bailing wire, spaghetti code jungle.

The de-facto standard software architecture. Why is the gap between what we **preach** and what we **practice** so large?

We preach we want to build high quality systems but why are BBoMs so prevalent?



Escape From The Spaghetti Code Jungle

Why BBoM?

Why was this phenomenon so prevalent in our industry? We sure talk a good game.

We had seen where Lisp had failed, Smalltalk was starting to fail, Windows was winning. Why was this?

What is there about some systems that failed compared to systems that succeed, even when they seemed better?

Escape From The Spaghetti Code Jungle

Worse is Better

Idea resembles Gabriel's 1991
"Worse is Better"

Worse is Better is an argument to release early and then have the market help you design the final product. It is taken as the first published argument for open source, among other things.

Do BBoM systems have a Quality?

Escape From The Spaghetti Code Jungle

What exactly do we mean by "Big"?

Well, for teams I consider $> 10^2$ big
and for code I consider $> 10^5$ big

Teams can write good code. Smalltalk is an example. I've seen teams of things written by 10^1 or 10^2 be pretty good and definitely would not be considered to be a BBoM.

Escape From The Spaghetti Code Jungle

Mud == Anti-Pattern???

In one sense Mud could be seen as an anti-pattern.

Reasons Mud Happens:

Throwaway Code, Piecemeal Growth, Keep it Working.

Similar Forces that lead to BBoM and anti-patterns.

Difference is that with BBoMs many reasons why they happened and are even successful (and maybe even necessary given our current state of the art).

Anti-Patterns were almost the opposite when you looked at the book. These are counterproductive practices.

Escape From The Spaghetti Code Jungle

Legacy == Mud?



Escape From The Spaghetti Code Jungle

Legacy != Mud???

Does Legacy happen within months or a year after the first release?

Or is legacy after the second release?

What about Muddy code that is released on the first version? Is this a counterexample?

Is all Legacy Mud? Smalltalk???

Escape From The Spaghetti Code Jungle

Is Mud Normal?

Well, just read our paper....there are "normal" reasons why it happens. Maybe it is the best we can do right now.

If mud is such a bad thing, why do people keep making it?

Maybe if we accept it and teach it more then we can deal with it better and help prevent it from getting too bad.

Escape From The Spaghetti Code Jungle

Where Mud Comes From?



DILBERT © United Feature Syndicate, Inc.
Redistribution in whole or in part prohibited.

People Write Code → People make Mud

Escape From The Spaghetti Code Jungle

Where Mud Comes From!

Software Tectonics

Reconstruction

- Major Upheaval
- Throw it away

Incremental Change

- Evolution
- Piecemeal Growth

Throwaway Code

Legacy Mush

Urban Sprawl

Slash and Burn Tactics

Merciless Deadlines

Sheer Neglect

Escape From The Spaghetti Code Jungle

Keep it Working, Piecemeal Growth, Throwaway Code



Escape From The Spaghetti Code Jungle

Copy 'n' Paste



Escape From The Spaghetti Code Jungle

The Age of Sampling



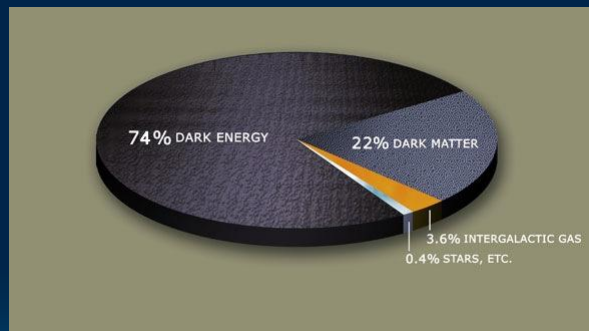
Escape From The Spaghetti Code Jungle

Big Bucket of Glue



Escape From The Spaghetti Code Jungle

The Mystery of Dark Matter



Accidental Complexity??? Maybe our current state of the art leads to Mud!

Escape From The Spaghetti Code Jungle

They Have a Name



Millionaires / Billionaires

Escape From The Spaghetti Code Jungle

Agile to the Rescue???

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

...From the Agile Manifesto

Escape From The Spaghetti Code Jungle

Can Agile Help?

Scrum, TDD, **Refactoring**, Regular
Feedback, **Testing**, More Eyes,

Good People!!!

Continuous attention to technical excellence!

Retrospectives!

Face-To-Face conversation.

Motivated individuals with the environment
and support they need.

Escape From The Spaghetti Code Jungle

Do Some Agile Principles Encourage mud?

Lack of Upfront Design?

Late changes to the requirements
of the system?

Continuously Evolving the Architecture?

Piecemeal Growth?

Focus on Process rather than Architecture?

Working code is the measure of success!

I'm sure there are more!!!

Escape From The Spaghetti Code Jungle

Quality

Quality Definition: a peculiar and essential character or nature, an inherent feature or property, a degree of excellence or grade, a distinguishing attribute or characteristic



Escape From The Spaghetti Code Jungle

Quality (Who's perspective)

Artist	Scientist
important/boring	true/false
Designer	Engineer
cool/uncool	good/bad

"The Four Winds of Making"...Gabriel

An architect has different perspectives than an artist, or a design, or an engineer...

Rich Gold "The Plenitude: Creativity, Innovation & Making Stuff (Simplicity: Design, Technology, Business, Life)"
Triggers and Practices – Richard Gabriel <http://www.dreamsongs.com>

Escape From The Spaghetti Code Jungle

Being Good Enough

- Quality of being good enough.
- Does it meet the minimum requirements
- Quality has many competing forces...are we designing a system for online orders or for controlling the space shuttle, they have different qualities, thus different patterns and solutions apply.
- Perfection is the enemy of **Good Enough!**
- Maybe Quality without a Number.

Escape From The Spaghetti Code Jungle

What is the Payoff?

The question that keeps getting asked is what value does the customer get from paying back this technical debt? What value does the customer get from simplifying this design? What value does the customer get from cleaning this code?

...

The answer is almost universally – none!!!

...Daniel Hinz comment on Brian Marick's Blog

Escape From The Spaghetti Code Jungle

Does Quality Code Matter?

Patterns about creating quality code that communicates well, is easy to understand, and is a pleasure to read. Book is about patterns of “Quality” code.



But...Kent states, “...**this book is built on a fragile premise: that good code matters. I’ve seen too much ugly code make too much money to believe that quality of code is either necessary or sufficient for commercial success or widespread use. However I still believe quality of code matters.**”

Patterns assist with making code more bug free and easier to maintain and extend.

Escape From The Spaghetti Code Jungle

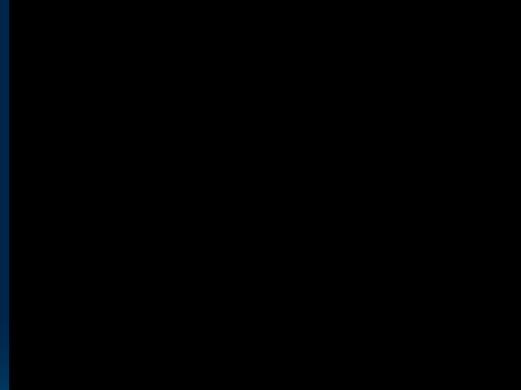
Some Answers to Mud!?!

Can we gentrify, rehabilitate, or make-over code helping clean up the mud?

Can **refactoring**, patterns, frameworks, components, agile, and objects help with mud?

Escape From The Spaghetti Code Jungle

Total Code Makeover



Can we just Refactor out of Mud?

Sweep the Mess Under the Rug?

Escape From The Spaghetti Code Jungle

Total Code Makeover

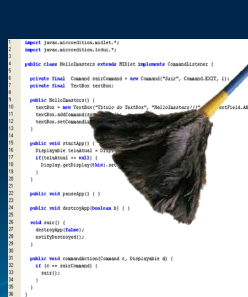
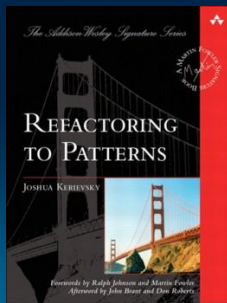


Escape From The Spaghetti Code Jungle

Code Make Over

Refactoring can help reverse some mud. The tradeoff is cost and time....maybe with technology

Refactoring to Better Design (Patterns)...



Escape From The Spaghetti Code Jungle

Refactorings

Behavior Preserving Program Transformations

- Rename Instance Variable
- Promote Method to Superclass
- Move Method to Component

Always done for a reason!!!

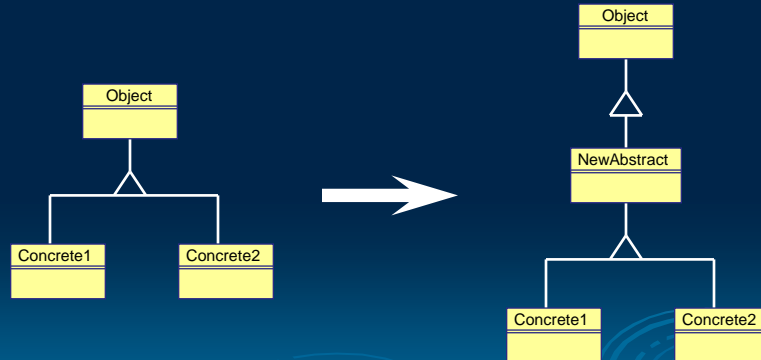
Refactoring is key and integral to most Agile processes!!!

Escape From The Spaghetti Code Jungle



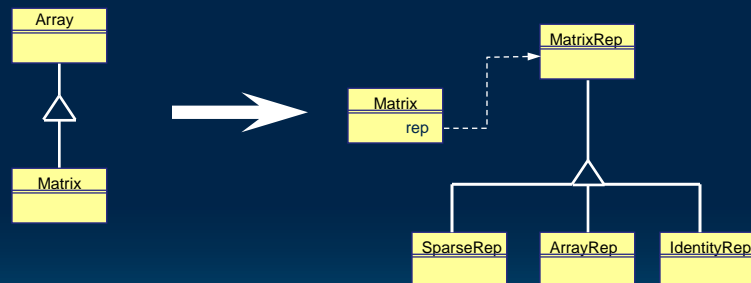
A Simple Refactoring

Create Empty Class



Adapted from Don Roberts, The Refactory, Inc.
Escape From The Spaghetti Code Jungle

A Complex Refactoring



Refactoring can be hard but there are a lot of small steps that lead to big gains in mud busting

Adapted from Don Roberts, The Refactory, Inc.

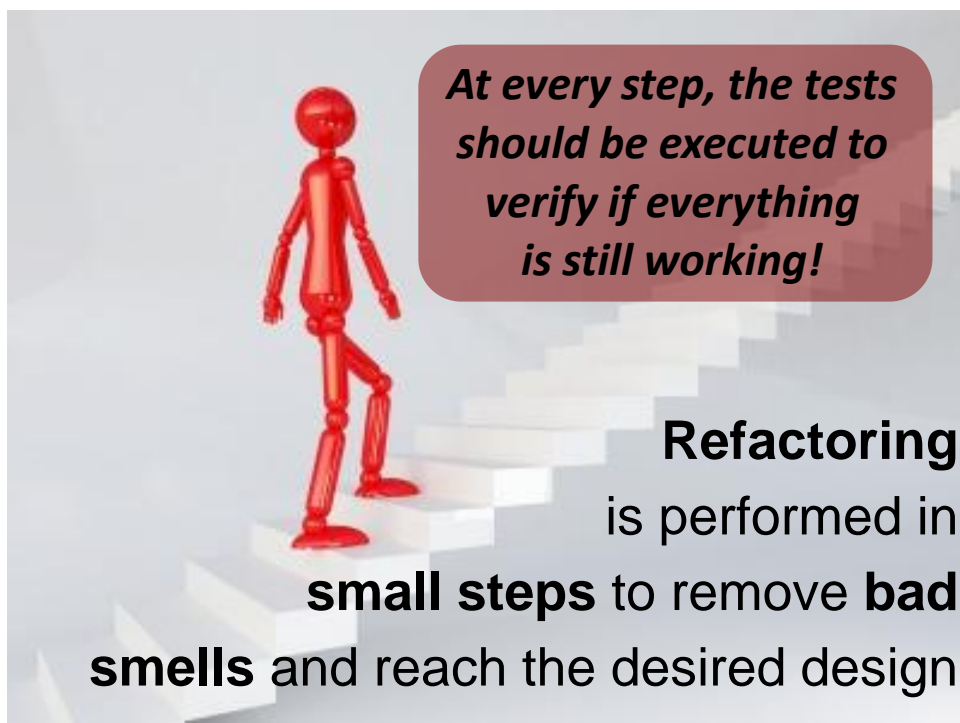
Escape From The Spaghetti Code Jungle

Catalogue of Refactorings

- Simpler Method Calls
- Composing Method
- Moving Features
- Organize Data
- Simplifying Conditionals
- Generalization

From Fowler's Book

Escape From The Spaghetti Code Jungle



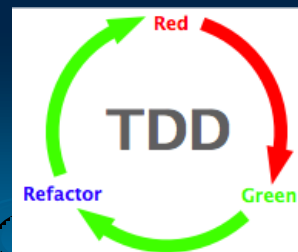
Testing



Escape From The Spaghetti Code Jungle

You Must Test

- When you find smelly code, you often apply refactorings to clean your code.
- Testing is a key principle for safe refactoring!



Escape From The Spaghetti Code Jungle

If we have a BBoM

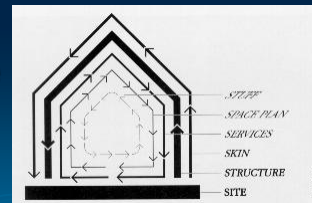
How can we even start?

How can we cordon off the mess?

Escape From The Spaghetti Code Jungle

Stuart Brand's Shearing Layers

- Buildings are a set of components that evolve in different timescales.
- Layers: site, structure, skin, services, space plan, stuff. Each layer has its own value, and speed of change (pace).
- Buildings adapt because faster layers (services) are not obstructed by slower ones (structure).



—Stuart Brand, *How Buildings Learn*

Escape From The Spaghetti Code Jungle

Yoder and Foote's Software Shearing Layers

"Factor your system so that artifacts that change at similar rates are together."—Foote & Yoder, Ball of Mud, PLoPD4.



Escape From The Spaghetti Code Jungle

Sweep It Under the Rug



Cover it up to keep other areas clean
(Façade and other Wrapper Patterns)

Escape From The Spaghetti Code Jungle

Put a rug at the Front Door

Protect Important Components!

Keep other parts of the system clean.

Sometimes Glue code (Mediators) helps
keep others parts of the system cleaner.
(Anti-Corruption Layer -- Eric Evans)



Escape From The Spaghetti Code Jungle

Code Smells

A *code smell* is a **hint** that something has **gone wrong** somewhere in your code. Use the smell to **track** down the **problem**... Kent Beck

Bad Smells in Code was an essay
by KentBeck and MartinFowler,
published as Chapter 3 of:
**Refactoring Improving The
Design Of Existing Code.**

----Ward's Wiki

Have you ever
looked at a piece of
code that doesn't
smell very nice?



Escape From The Spaghetti Code Jungle

Ten Most Putrid List

- 1) Sloppy Layout,
- 2) Dead Code,
- 3) Lamé Names,
- 4) Commented Code,
- 5) Duplicated Code,
- 6) Feature Envy,
- 7) Inappropriate Intimacy,
- 8) Long Methods & Large Class,
- 9) Primitive Obsession & Long Parameter List,
- 10) Switch Statement & Conditional Complexity ...



Escape From The Spaghetti Code Jungle

Bad Formatting

```

1 public
2
3 class
4
5 SyntaxHighlighterTest {
    public
    static
    void
    main(String[] args) {
        System.out.println(
            "Nice highlighting!"
        );
    }
}

```

```

void foo(int x[], int y, int z){
    if (z > y + 1)
    {
        int a = x[y], b = y + 1, c = z;
        while (b < c)
        {
            if (x[b] <= a) b++; else {
                int d = x[b]; x[b] = x[--c];
                x[c] = d;
            }
        }
        int e = x[--b]; x[b] = x[y];
        x[y] = e; foo(x, y, b); bar(x, c, z);
    }
}

```

```

void bar(int i[], int j, int k)
{ return i[j] = int [k]}

```

Escape From The Spaghetti Code Jungle

Dead Code

```
void foo(int x[], int y, int z) {
    if (z > y + 1) {
        int a = x[y], b = y + 1, c = z;
        while (b < c) {
            if (x[b] <= a) b++; else {
                int d = x[b]; x[b] = x[--c];
                return;
                x[c] = d;
            }
            x[b] = a;
        }
        y = 5; // set y equal to 5
        int e = x[--b]; x[b] = x[y];
        x[y] = e; foo(x, y, b);
        /* used to use bar,
           might need it again
           bar(x, c, z); */
    }
}
```

```
void bar(int i[], int j, int k) {
    /* bar method returning nothing */
    if (j > k) {
        return k
        i[k] = i[j];
    }
    if (j == k) {
        return i[j] = int [k]
    }
}
```

```
213  /**
214   * Get additional attributes as a Map.
215   * @return Map A Map containing attribute name - value pairs.
216   */
217  public Map getAttributes() {
218      Map map = new HashMap(extraAttributes);
219      // Add property attributes using old names
220      /**
221       map.put(DEFINITIONS_CONFIG_PARAMETER_NAME, getDefinitionConfigFiles());
222       map.put(TILES_DETAILS_PARAMETER_NAME, Integer.toString(getDebugLevel()));
223       map.put(PARSER_DETAILS_PARAMETER_NAME, Integer.toString(getParserDebugLevel()));
224       map.put(PARSER_VALIDATE_PARAMETER_NAME, new Boolean(getParserValidate()));
225       /**
226       if ( ! "org.apache.struts.tiles.xmlDefinition.I18nFactorySet".equals(g
227       map.put(FACTORY_CLASSNAME_PARAMETER_NAME, getFactoryClassname()));
228       /**
229       return map;
230  }
```

Escape F

Fix the Layout and Remove Useless Items

Format the Code Consistently

Agree on a standard format

Set the tools for consistent formatting

Run the tools over the code base

Remove Unreachable Code

Delete useless comments

Delete commented out code

Remove code that can't be reached,

Escape From The Spaghetti Code Jungle

Lame Names

```
void foo(int x[], int y, int z)
{
    if (z > y + 1)
    {
        int a = x[y], b = y + 1, c = z;
        while (b < c)
        {
            if (x[b] <= a) b++; else {
                int d = x[b]; x[b] = x[--c];
                x[c] = d;
            }
        }
        int e = x[--b]; x[b] = x[y];
        x[y] = e; foo(x, y, b);
        foo(x, c, z);
    }
}

void quicksort(int array[], int begin, int end) {
    if (end > begin + 1) {
        int pivot = array[begin],
        l = begin + 1, r = end;
        while (l < r) {
            if (array[l] <= pivot)
                l++;
            else
                swap(&array[l], &array[--r]);
        }
        swap(&array[--l], &array[begin]);
        sort(array, begin, l);
        sort(array, r, end);
    }
}
```

<http://dreamsongs.com/Files/BetterScienceThroughArt.pdf>

Escape From The Spaghetti Code Jungle



Fixing Names

Names should *mean something*.

Standards improve communication

- know and follow them.

Standard protocols

object ToString(), Equals()

ArrayList Contains(), Add(), AddRange()

Remove(), Count, RemoveAt(),

HashTable Keys, ContainsKey(),

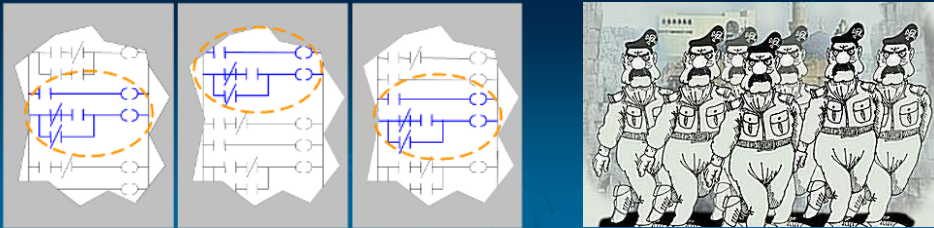
ContainsValue()

Standard naming conventions

Escape From The Spaghetti Code Jungle

Duplicate Code

- Do everything exactly once
- Duplicate code makes the system harder to understand and maintain
 - Any change must be duplicated
 - The maintainer has to change every copy



Escape From The Spaghetti Code Jungle

Fixing Duplicate Code

- Do everything exactly once!!!

DRY Principle

- Fixing Code Duplication
 - Move identical methods up to superclass
 - Move methods into common components
 - Break up Large Methods



**Do not
duplicate!**



REUSE

Escape From The Spaghetti Code Jungle

Inappropriate Intimacy

When classes depend on other's implementation details ...

Tightly coupled classes - you can't change one without changing the other.

Boundaries between classes are not well defined.



Escape From The Spaghetti Code Jungle

Refactoring Addresses Some Key Leverage Points

Refactoring is a technique that works with Brooks' "promising attacks" (from "No Silver Bullet"):

- buy rather than build: restructuring interfaces to support commercial SW
- grow don't build software: software growth involves restructuring
- requirements refinements and rapid prototyping: refactoring supports such design exploration, and adapting to changing customer needs
- support great designers: a tool in a designer's tool chest

Escape From The Spaghetti Code Jungle

Can tools Help?

What is the role of tools in draining these swamps?

What kinds of tools and practices might forestall software entropy; is mud preventable?



Tools can help, but too often too much is put on tools as the solution to all our problems.

Refactoring Tools, Testing Tools, XUnit, Lint Tools, Code Critics, ...

Escape From The Spaghetti Code Jungle

Many Quality Patterns Written

Design Patterns

Patterns for Fault Tolerant Software

Performance Patterns

Small Memory Software Patterns

Analysis Patterns

Security Patterns

Stability Patterns

Usability Patterns

.....

Imitate or use proven quality techniques

<http://www.hillside.net>

Escape From The Spaghetti Code Jungle

Draining the Swamp

You can escape from the
"Spaghetti Code Jungle"

Indeed you can transform the landscape.

The key is not some magic bullet, but a
 long-term commitment to **architecture**,
 and to cultivating and refining *"quality"*
artifacts for your domain (**Refactoring**)!

Patterns of the best practices can help!!!

Escape From The Spaghetti Code Jungle

Silver Buckshot

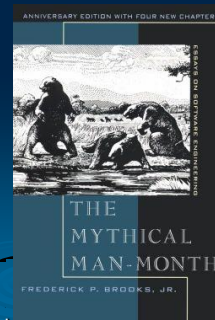
There are no silver bullets

...Fred Brooks

But maybe some silver buckshot

...promising attacks

Good Design
 Frameworks
Patterns
 Architecture
 Process/Organization
 Tools and Support
Refactoring
Good People ***



Escape From The Spaghetti Code Jungle

Mud is Here...

*It isn't always bad!
It can be contained!
It can be cleaned up!*



Our code can be more habitable!

Escape From The Spaghetti Code Jungle

So There is Some Hope!!!

Testing (TDD), **Refactoring**, Regular
Feedback, Patterns, More Eyes, ...

Good People!!!

Continuous attention to technical excellence!

Retrospectives!

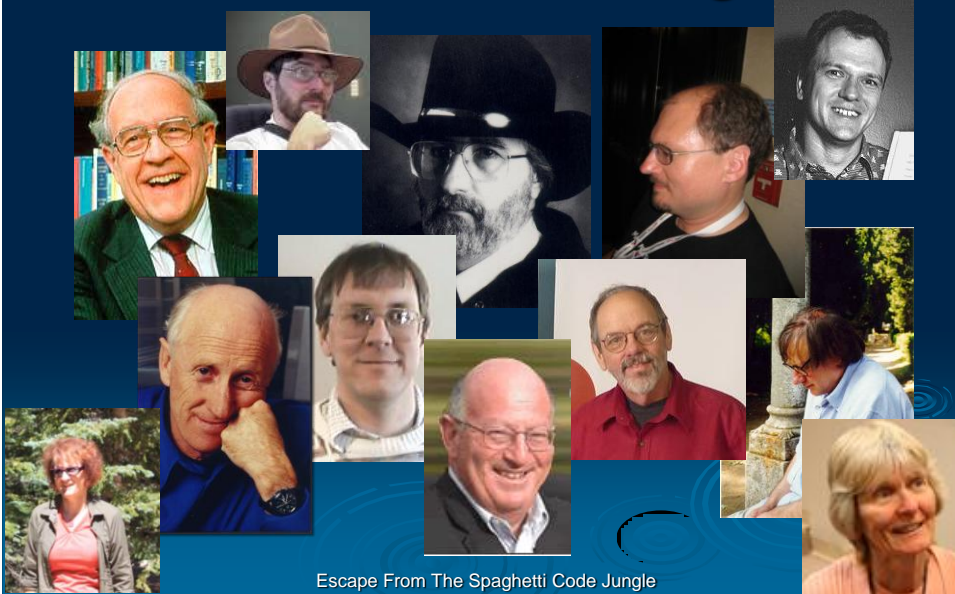
Face-To-Face conversation.

Motivated individuals with the *environment*
and *support* they need.

But, Maybe Mud is why we have Agile...

Escape From The Spaghetti Code Jungle

It Takes a Village



Escape From The Spaghetti Code Jungle

Dziękuję!!!



joe@refactory.com
Twitter: @metayoda

Escape From The Spaghetti Code Jungle