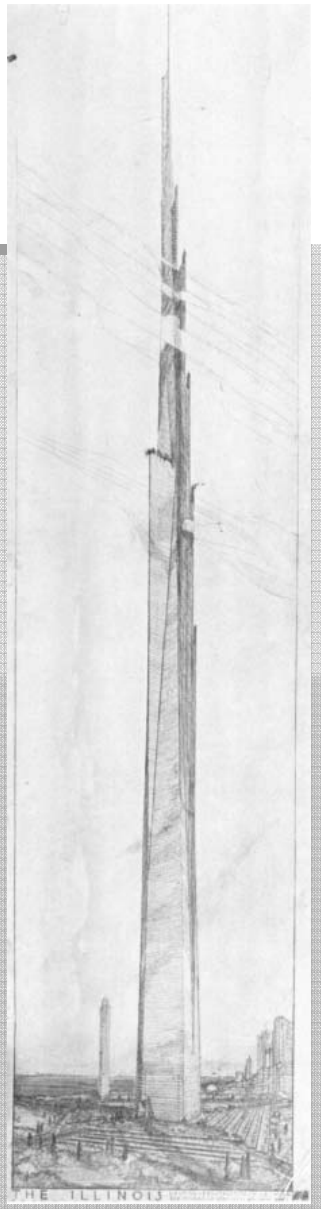
Big Balls of Mud in Agile Development - How to Avoid Them

Joseph W. Yoder

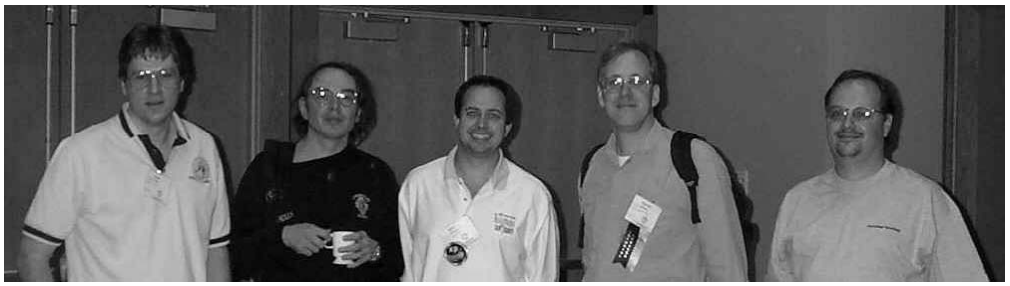
11 October 2009



Copyright 2009 Joseph W. Yoder & The Refactory, Inc.

Evolved from The UIUC SAG

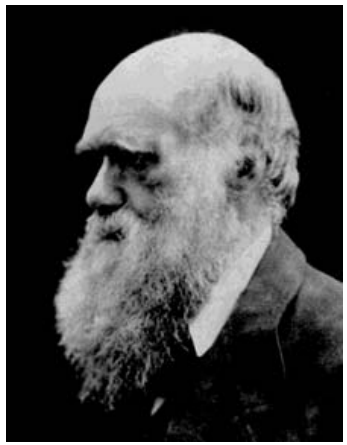
In the early 90's we were studying objects, frameworks, components, reusability, patterns, "good" architecture.

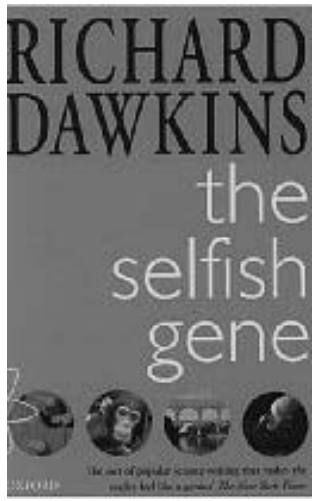


However, in our SAG group we often noticed that although we talk a good game, many successful systems do not have a good internal structure at all.

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.

Why BBoM?

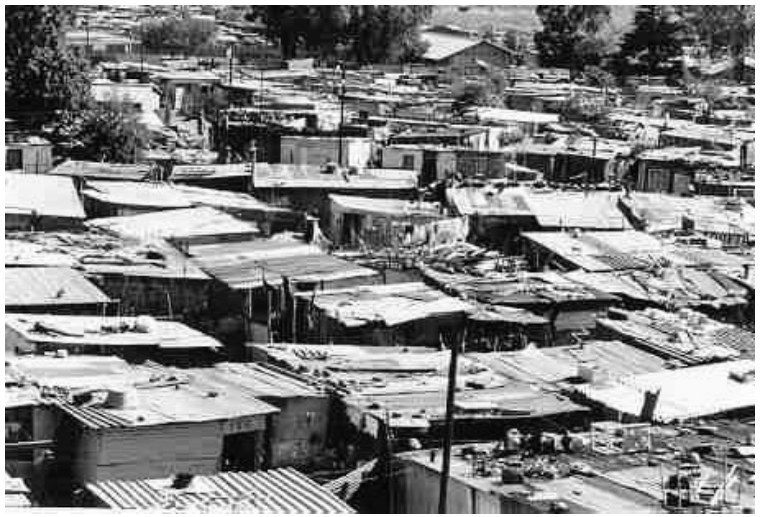
Why was this phenomenon so prevalent in our industry? We talk a good game.

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.

Big Ball of Mud

Alias: Shantytown, Spaghetti Code

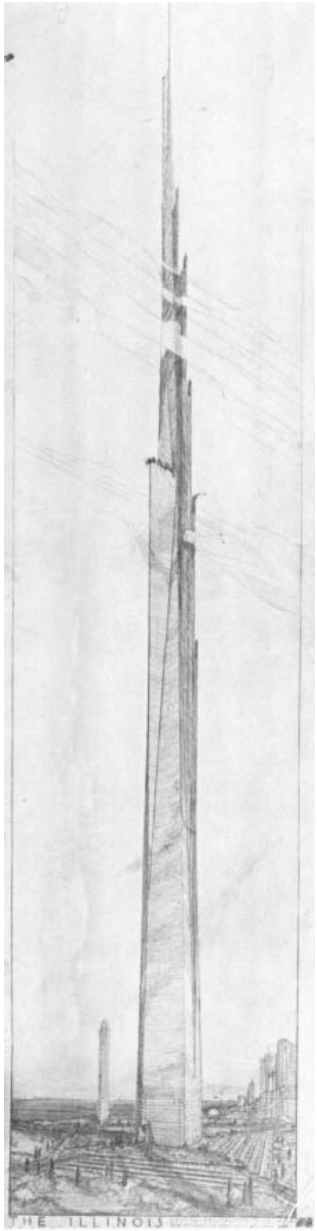
A BIG BALL OF MUD is haphazardly structured, sprawling, sloppy, duct-tape and bailing wire, spaghetti code jungle.



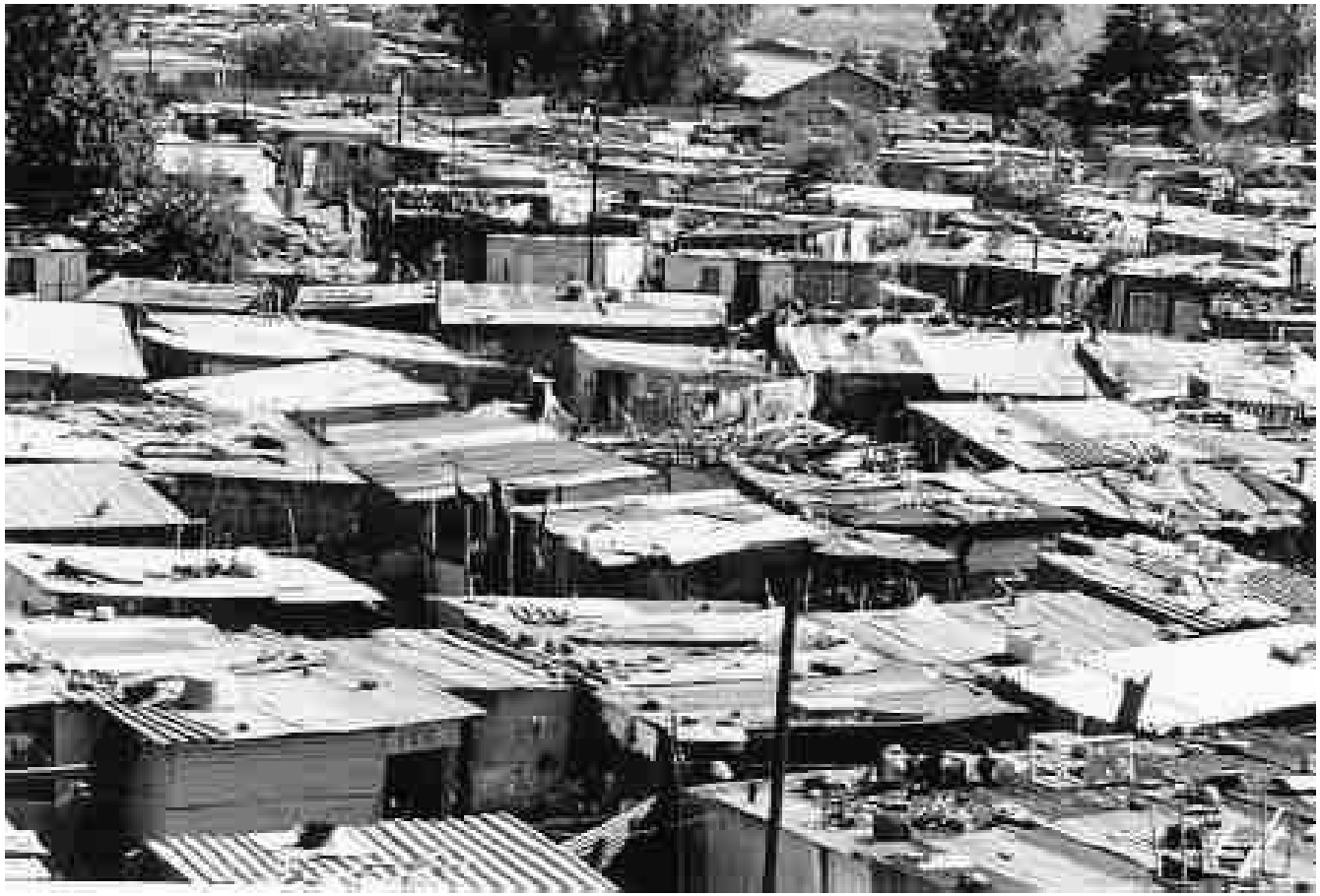
The de-facto standard software architecture. Why is the gap between what we **preach** and what we **practice** so large?

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

A Scene from a Nightmare



Big Ball of Mud



Worse is Better

Ideas 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?

Worse is Better (examples)

Betamax vs VHS Format

- Why did VHS win?
- Betamax was arguably a better format

Macintosh vs Windows

- Mac was easier to use
- Far superior in many ways

MS Word/Publisher vs FrameMaker

- Lot’s of people use word
- FrameMaker is better for books

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.

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.

Legacy == Mud?

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???

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.

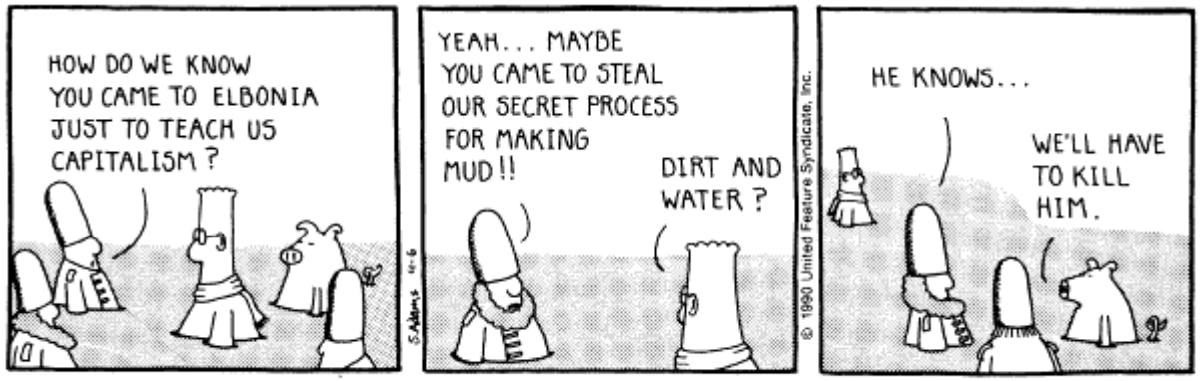
If mud is such a bad thing, why do people keep making it?

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

Question

So, why DO people build Big Balls of Mud?

Where Mud Comes From



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

Software Decay

Robber Crops

Keep it Working, Piecemeal Growth, Throwaway Code

Copy ‘n’ Paste

The Age of Sampling

Big Bucket of Glue

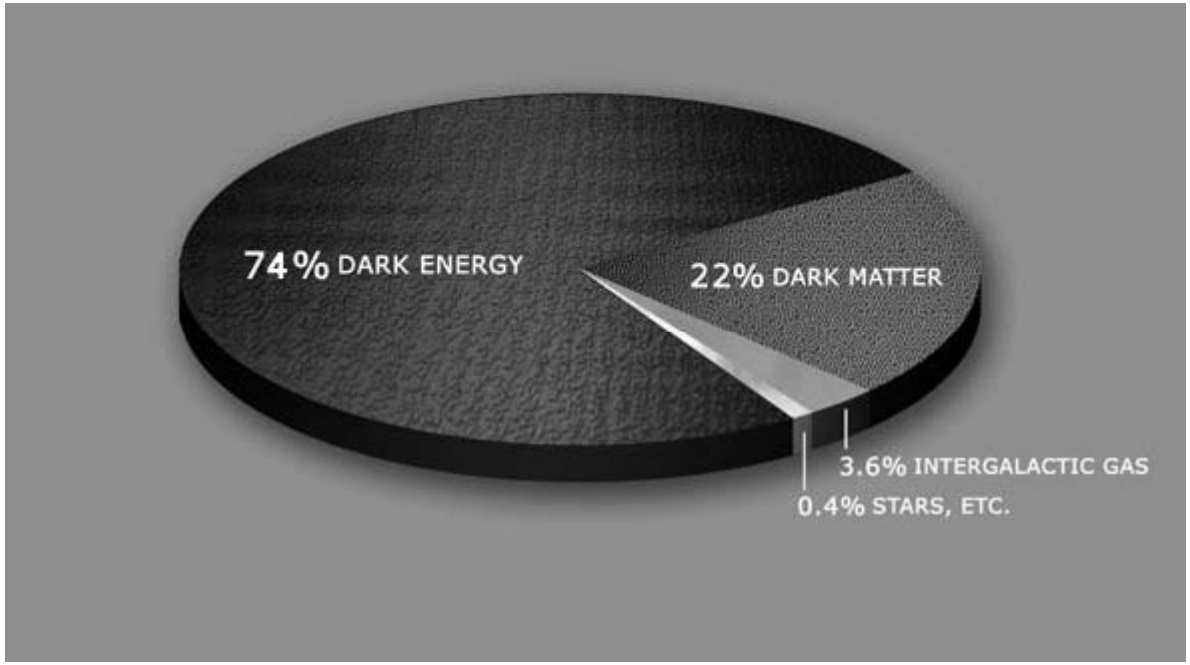
Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code by: Alan MacCormack, John Rusnak, Carliss Baldwin

Found a significant difference in structure between Linux and the first version of Mozilla, suggesting that Linux had a more modular architecture.

Yet also found that the re-design of Mozilla resulted in an architecture that was significantly more modular than that of its predecessor, and indeed, than that of Linux.

Different modes of organization are associated with designs that possess different structures. However, they also suggest that purposeful managerial actions can have a significant impact in adapting a design's structure.

The Mystery of Dark Matter



They Have a Name

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

Question

What Agile Practices help us avoid or cope with mud? Does Agile practices such as TDD really help minimize mud? What are we doing RIGHT?

What Agile Practices contribute to the problem? Encourage mud? So Is Mud really the best that Agile can do?

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!!!

Do we really need Agile Principles such as TDD to avoid mud?

After all, most of the early STers I know never practiced TDD or SCRUM yet we claim these methods helps create clean code and avoid mud more easily. Is this really true or are there other methods that work as well or better at minimizing mud? Maybe it is just good teams of people no matter what method they use?

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.

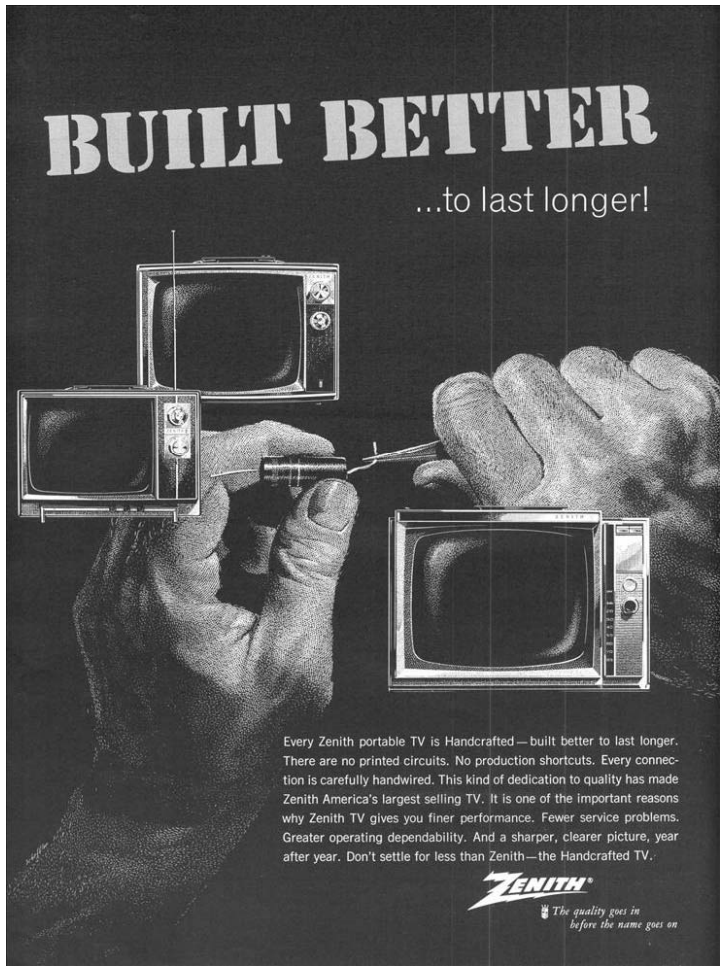
Question

Is Craftsmanship the Cure? Or maybe it is the problem? Is ascribing poor code to unhygienic habits, even malpractice, is enough, or naive; is mud inevitable?

Does quality matter? Quality on the outside vs. Quality on the inside? Does quality just get in the way? Is "Clean" the Answer? Or only part of the answer? Or a sideshow?

Shoddy Workmanship

The Quality Goes In



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

Quality (Who's perspective)

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

"The Four Winds of Making"…Gabriel

An architect might have perspectives from an artist to 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

Quality by Attributes

Quality in everyday life and business, engineering and manufacturing can be seen as the *usefulness* of something. Usually describes certain attributes.

- For example you could describe quality in terms of performance, reliability (fault tolerance), safety, maintainability, reusability, etc…

Does quality on the inside mean quality on the outside?

Non-functional Requirements

Accessibility	Reliability
Compatibility	Safety
Efficiency	Scalability
Effectiveness	Security
Extensibility	Stability
Maintainability	Supportability
Performance	Usability

Other terms for non-functional requirements are "constraints", "quality attributes“, and "quality of service requirements"

Qualities are usually described by “ilities” as seen in non-functional requirements…but quality can also focus on how well the functional requirements are met (how to measure this?)

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 to “being good enough”.

Brooklyn Bridge

The **Brooklyn Bridge,** 1883, one of the oldest suspension bridges in the United States, stretches over a mile from Brooklyn to Manhattan.

On completion, it was the largest suspension bridge in the world and the first steel-wire suspension bridge.

Brooklyn Bridge

Over engineered. Had 6 times what it needed which proved useful over time

What happens if we tried overdesign of our systems (a language for printing hello world) or the same line of code 6 times (is this 6 times more reliable?)

```
if (x != a) x = a; if (x != a) x = a; if (x != a) x = a;
if (x != a) x = a; if (x != a) x = a; if (x != a) x = a;
```

Redundant components can make our systems more reliable

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

Many Quality Patterns Written

Design Patterns

Patterns for Fault Tolerant Software

Performance Patterns

Small Memory Software Patterns

Analysis Patterns

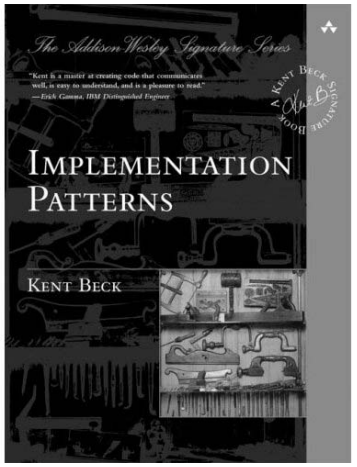
Security Patterns

Stability Patterns

Usability Patterns

Imitate or use proven quality techniques

Implementation Patterns



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.

Is Craftsmanship the Cure? Or is it the problem?

On the average, average companies get average people (law of averages).

Is ascribing poor code to unhygienic habits, even malpractice, is enough, or naive; is mud inevitable?

Does quality really matter?

Is "Clean" the Answer? Or only part of the answer? Or a sideshow? Who ever said code was supposed to be pretty?

Being clean or cleaning up certain parts of the mud might help in the short run. However, as we've heard before, Lipstick on a Pig and you still have a pig, or dressing it up, still doesn't get rid of the mess inside.

But maybe we don't need to clean it all up. Is this even completely possible for something of any size. I claim that at times you will always have some muddy code. There are counter forces to trying to make something too perfect. Perfection is the enemy of Good and sometimes Good enough is all we need. No normal end-user looks into the middle of something and says, that is it...we got beauty.

Amish Furniture

Question

When do we gentrify, rehabilitate, or make-over code, and when must be demolish and/or replace it? (In other words) Is mud reversible?

How much have patterns, frameworks, components, even objects helped with mud?

Total Code Makeover

Code Make Over?

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

Joel on Software says rewriting it is the one thing you should never do? Yet, we say we should do it…sometimes.

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?

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

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

Testing

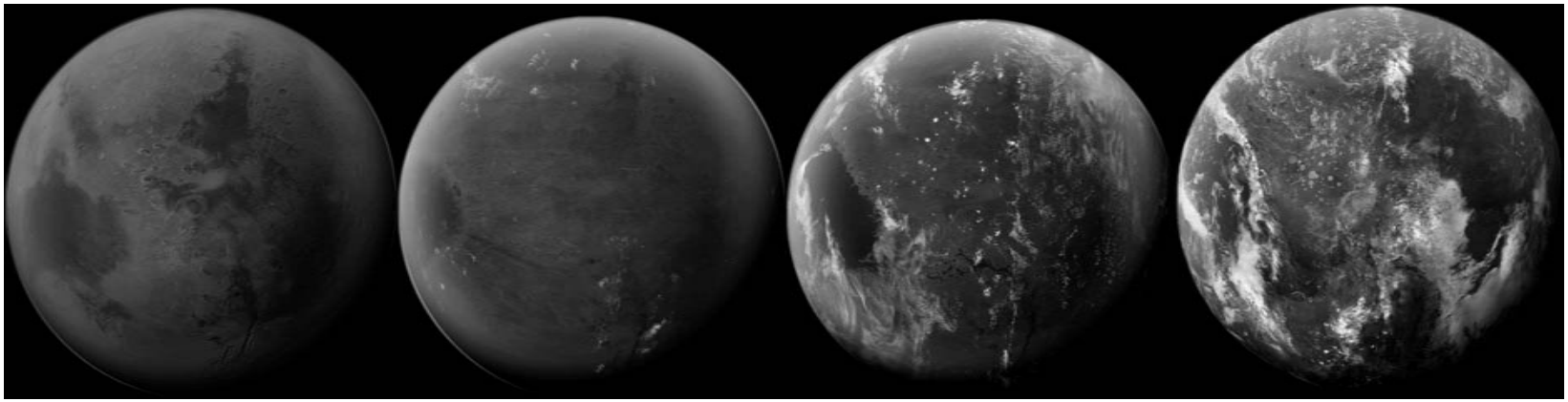

SYLVANIA
ELECTRONIC TUBE
SYLVANIA
6DR7


POLAM
PL504
4319
MADE IN POLAND


SELF-SERVICE
TUBE TESTER
Buy top quality guaranteed tubes HERE!

Commandos and Infantry

Terraforming Code



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!!!

Silver Buckshot

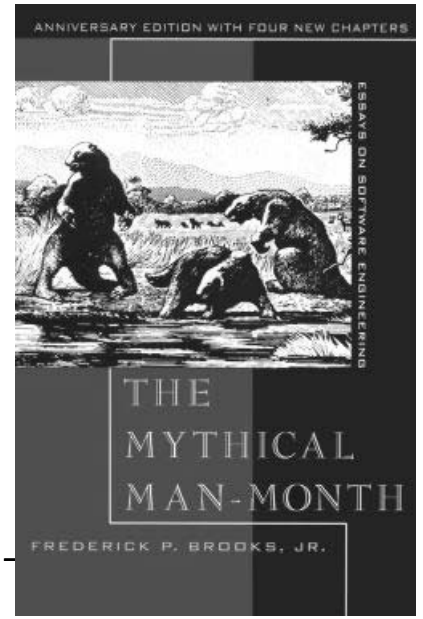
There are no silver bullets

…Fred Brooks

But maybe some silver buckshot

…promising attacks

Good Design
Frameworks
Patterns
Architecture
Process/Organization
Tools
Good People ***



So Maybe Agile Can 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.

Maybe Mud is why we have Agile….

It Takes a Village

That’s All