# CS 635 Advanced Object-Oriented Design & Programming
Spring Semester, 2009
Doc 2 Big Ball of Mud
Jan 27, 2009

# References

Big Ball of Mud, http://www.laputan.org/mud/

# Reading

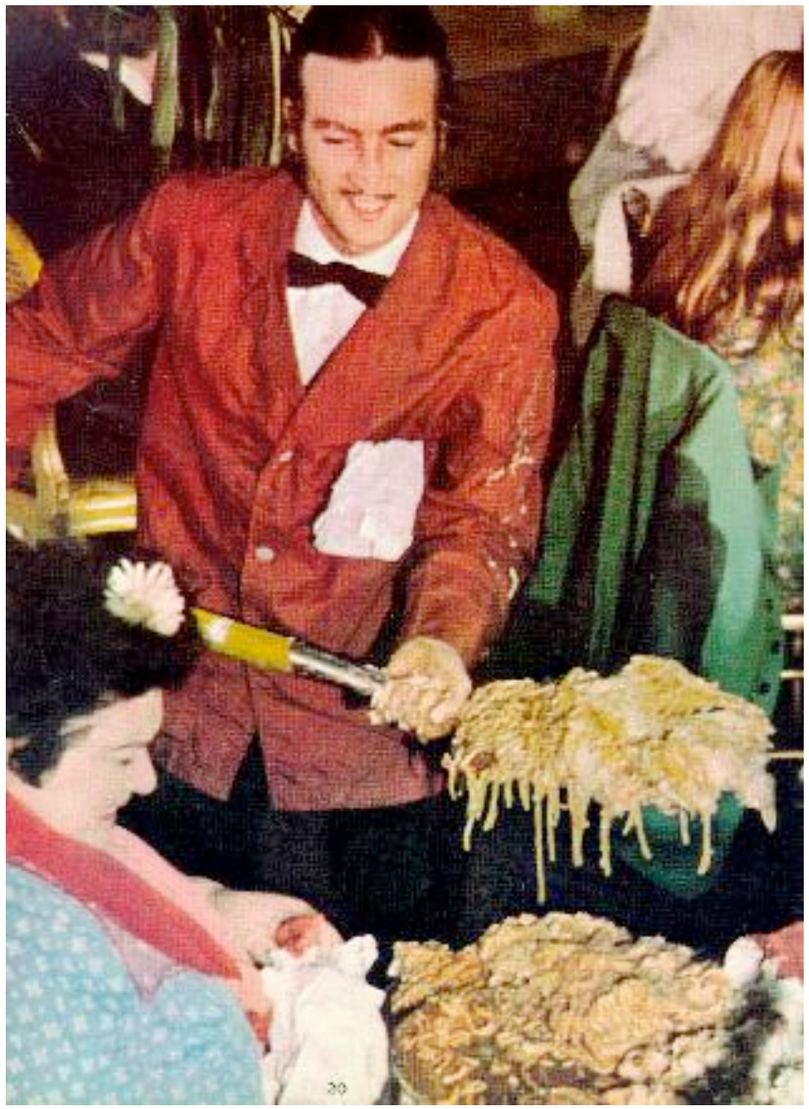
Jan 27 - Big Ball of Mud, http://www.laputan.org/mud/mud.html

Jan 29 - Refactoring, Chapters 1 & 2

Feb 3 - Refactoring, Chapters 3 & 4

Feb 5 - Iterators, Null Object Pattern, Visitor

# Who is this?



# What is a Big Ball of Mud?

# What Forces Lead to Big Ball of Mud

# Patterns

Big Ball of Mud
Throwaway Code
Piecemeal Growth
Keep it Working
Shearing Layers
Sweeping it Under the Rug
Reconstruction

# Big Ball of Mud

You need to deliver quality software on time, and under budget.

Therefore, focus first on features and functionality, then focus on architecture and performance.

# Problems

Variable and function names
- uninformative

Functions themselves may make extensive use of
- global variables,
- long lists of poorly defined parameters.

The function themselves are
- lengthy and convoluted,
- perform several unrelated tasks.

The programmer’s intent is next to impossible to discern.

We built the most complicated system that can possible work

# Three ways to deal with BIG BALLS OF MUD

# Extreme Programming Practices

Pair programming
Planning game
Test driven development
Customer part of development team
Continuous integration
Refactoring or design improvement
Small releases
Coding standards
Collective code ownership
Simple design
System metaphor
Sustainable pace

# Throwaway Code

You need an immediate fix for a small problem, or a quick prototype or proof of concept.

Therefore, produce, by any means available, simple, expedient, disposable code that adequately addresses just the problem at-hand.

Why do we need throwaway code?

What the main problem with throwaway code?

# Piecemeal Growth

Master plans are often rigid, misguided and out of date. Users' needs change with time.

Therefore, incrementally address forces that encourage change and growth.

Allow opportunities for growth to be exploited locally, as they occur.

Refactor unrelentingly.

# Keep it Working

Maintenance needs have accumulated, but an overhaul is unwise, since you might break the system.

Therefore, do what it takes to maintain the software and keep it going. Keep it working.

How do Piecemeal Growth and Keep it Working lead to a ball of mud?

How can use Piecemeal Growth and Keep it Working and avoid the ball of mud?

Is it advisable to use Piecemeal Growth and Keep it Working?

# Sweep it Under the Rug

Overgrown, tangled, haphazard spaghetti code is hard to comprehend, repair, or extend, and tends to grow even worse if it is not somehow brought under control.

Therefore, if you can’t easily make a mess go away, at least cordon it off.

This restricts the disorder to a fixed area, keeps it out of sight, and can set the stage for additional refactoring.

# Reconstruction

Your code has declined to the point where it is beyond repair, or even comprehension.

Therefore, throw it away and start over.

"Plan to throw one away, you will anyway"

Fred Brooks

# Problems with Starting Over

Cost

Time

Reintroduce bugs

Few features