

Big Ball of Mud Pattern

Brian Foote
Joseph Yoder

This paper examines the most frequently deployed architecture: the **BIG BALL OF MUD**.

... a casually, even haphazardly, structured system.

Why does it exist/persist ?

Some patterns involved include

BIG BALL OF MUD

THROWAWAY CODE

PIECEMEAL GROWTH

KEEP IT WORKING

SWEEPING IT UNDER THE RUG

RECONSTRUCTION

A number of forces can conspire to drive even the most architecturally conscientious organisations to produce BIG BALLS OF MUD. These “global” forces are at work in all the patterns presented.

Time: There may not be enough time to consider the long-term architectural implications of one’s design and implementation decisions.

Because of more mundane concerns such as cost, time-to-market, and programmer skill.

Architecture is often seen as a luxury or a frill, or the indulgent pursuit of lily-gilding compulsives who have no concern for the bottom line.

Architecture is often treated with neglect, and even disdain.

Architecture is a long-term concern.

The concerns above have to be addressed if a product is not to be stillborn in the marketplace, while the benefits of good architecture are realised later in the lifecycle, as frameworks mature, and reusable black-box components.

Experience: Even when one has the time and inclination to take architectural concerns into account, one’s experience, or lack thereof, with the domain can limit the degree of architectural sophistication that can be brought to a system, particularly early in its evolution.

Turnover can wreak havoc on an organisation's institutional memory, with the perhaps dubious consolation of bringing fresh blood aboard.

Skill: Programmers differ in their levels of skill, as well as in expertise, predisposition and temperament.

Complexity: One reason for a muddled architecture is that software often reflects the inherent complexity of the application domain.

Change: Architecture is a hypothesis about the future that holds that subsequent change will be confined to that part of the design space encompassed by that architecture.

Cost: Architecture is expensive, especially when a new domain is being explored.

1. BIG BALL OF MUD

alias

SHANTYTOWN

SPAGHETTI CODE

... built from common, inexpensive materials and simple tools. ... using relatively unskilled labour. and maintenance of this sort of housing can be quite labour intensive. There is little specialisation.

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

More often, the customer needs something working by tomorrow. Often, the people who control and manage the development process simply do not regard architecture as a pressing concern.

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

In other words, it's okay if the system looks at first like a BIG BALL OF MUD, at least until you know better.

2. THROWAWAY CODE

alias

QUICK HACK

KLEENEX CODE

DISPOSABLE CODE

SCRIPTING

KILLER DEMO

PERMANENT PROTOTYPE

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

Time

Quick-and-dirty coding is often rationalised as being a stopgap measure. All too often, time is never found for this follow up work. The code languishes, while the program flourishes.

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

3. PIECEMEAL GROWTH

alias

URBAN SPRAWL

ITERATIVE-INCREMENTAL DEVELOPMENT

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.

The biggest risk associated with PIECEMEAL GROWTH is that it will gradually erode the overall structure of the system, and inexorably turn it into a BIG BALL OF MUD.

4.KEEP IT WORKING

alias

VITALITY

BABY STEPS

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.

5. SWEEPING IT UNDER THE RUG

alias

POTEMKIN VILLAGE

HOUSECLEANING

PRETTY FACE

QUARANTINE

HIDING IT UNDER THE BED

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. [can lead to the use of various new patterns e.g. FAÇADE]

6. RECONSTRUCTION

also known as

TOTAL REWRITE

DEMOLITION

THROWAWAY THE FIRST ONE

START OVER

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

Therefore, throw it away it and start over.

CONCLUSION

In the end, software architecture is about how we distil experience into wisdom, and disseminate it. Still, we do not consider these patterns to be anti-patterns. There are good reasons that good programmers build BIG BALLS OF MUD

The key is to ensure that the system, its programmers, and, indeed the entire organisation, *learn* about the domain, and the architectural opportunities looming within it, as the system grows and matures.