

Big Ball of Mud

Ketul Mehta
Vishvesh Shah
Sayali Naval
Vishal Parekh

Features of Muddy Code:

- Lacks perceivable architecture
- Not desirable but common in practise
- Haphazardly constructed, Duct- tape, Sprawling Code, Spaghetti Code Jungle.
- Everything talks to everything else
- Extensive use of global variables
- Code Duplication
- Documentation

Muddy and Happy?



Your View?

Legacy == Mud?

OR

Legacy !=Mud?

Forces:

- Time
- Cost
- Skill and Experience
- Visibility
- Application Complexity

Patterns:

- Big Ball of Mud
- Throwaway Code
- Piecemeal Growth
- Keep it Working
- Shearing layers
- Sweeping it under the rug
- Reconstruction

Analogy between Shanty Town and Software Systems:

Alias:

1. Shanty Town
2. Spaghetti Code



Shanty Town:

- Slums built with inexpensive materials and simple tools
- Relatively unskilled labours
- Infrastructure is not a major concern
- Fulfills immediate requirement

Software Systems as compared to Shanty Town:

- Inadequate investment in tools and infrastructure
- Undercapitalized libraries and frameworks
- Individual portions of the system goes unchecked
- Short deadlines

Throw away code

- Throw away code can be referred as quick and dirty code that was intended to be used only once and then discarded.
- It is used when we need an immediate fix for a small problem or a quick prototype or a proof of concept.



Throw away code

- Time is frequently the decisive force that drives programmers to write the throw away code.
- Throw away code is often written as an alternative to reusing someone else's more complex code.
- It is nearly a universal practice.
- The real problem is when it is not thrown away.

Piecemeal Growth

- This is an incremental or iterative kind of software development.
- Before waterfall development , programmers used simple,casual relatively undisciplined “code and fix” approach to software development.
- Code and fix approach worked for small jobs but it did not scale well.

Piecemeal Growth

- Waterfall approach was able to flourish before because computer and business requirements changed at a more leisurely pace.
- But today's designers are confronted with a broad onslaught of the changing requirement.

Piecemeal Growth

- Master plans are often rigid, misguided and out of date. Users need change with time.
- We can't simply plan, we have to plan to be able to adapt.
- Incrementally addresses forces that encourage change and growth.
- Uncontrolled growth can ultimately be a malignant force. The result of neglecting to contain it can be a Big Ball Of Mud.

Keep It Working

Maintenance of a system:

- Taking a system down for a major overhaul.
- Bad Fix Injection.
- Might break the system.

Keep It Working

Work with a live system:

- Do what it takes to maintain the software and keep it going.
- Integrate new subsystems with the running system in such a way as to minimize disruption.



Keep It Working

Factors that ensure the system doesn't break:

- Baby steps.
- Daily/Weekly Build.
- A commitment to rigorous testing.

Shearing Layers

Concept:

- Different software artifacts change at different rates.
- Factor your system into different layers so that artifacts that change at similar rates are in same layer.

Shearing Layers

Different layers of software applications within organisations:

- Adaptability: Artifacts that evolve quickly provide a system with dynamism and flexibility.
- Stability: Slowly evolving objects are bulwarks against change.

Shearing Layers

Layers in software:

- Data: Changes most quickly.
- Code: Changes more slowly than data.

Shearing Layers

Layers in object-oriented languages:

- The abstract classes and components change more slowly.
- Some abstractions make their ways into the frameworks and libraries that constitute the system's infrastructure.

Shearing Layers

Software evolution:

- Enduring insights gravitate towards primary structural elements.
- Things in flux are spun out into the data.

Sweeping it under the rug

- If you can't make a mess go away, at least you can hide it.



Sweeping it under the rug

- How much chaos an individual can tolerate?
- Spaghetti code is hard to comprehend, repair, or extend.



Sweeping it under the rug

- Comprehensibility

It takes time and money.

- Morale

Minor modifications can lead to maintenance marathons.

Sweeping it under the rug

Therefore,

- If you can't easily make a mess go away, at least cordon it off.



Sweeping it under the rug

Solution

- Re-establishing the system's conceptual integrity
- Architectural rehabilitation
- Use of Facade

Reconstruction

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



Like a set of dominoes, the former home of the Atlanta Braves collapsed

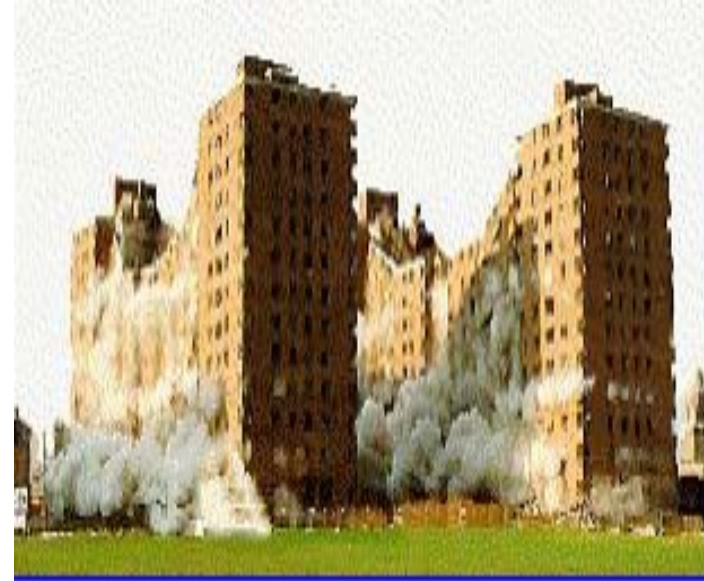
Reconstruction

- Obsolescence

Technically or economically outdated

- Change

Impossible to accommodate new demands



Reconstruction

- Cost

Traumatic for both who have worked on it and who have paid for it

- Organization

Demand considerable time and resources, requires high-level management support

Reconstruction

Advantages

- Re-establishes contact between architecture and implementation
- Issues are revisited
- Getting the experience needed to do the job

Conclusion

- Economics of the software world are such that the market moves so fast that long term architectural ambitions are foolhardy
- Casual architecture is natural during the early stages of a system's evolution
- People build BIG BALLS OF MUD because they work

Thank You!