

Big Ball of Mud: Patterns and Controls

(derived from the work of Brian Foote and Joseph Yoder, © 1999 Foote and Yoder)

Bar Biszick, CQA

Quality Assurance Systems Analyst
Enterprise Systems Department
SAFECO Insurance, Seattle WA

Presentation handout:
Quality Assurance Institute's
International IT Quality Conference
Tuesday, April 24, 2001

Big Ball of Mud: Characteristics and Controls

(derived from the work of Brian Foote and Joseph Yoder, © 1999 Foote and Yoder)

➤ **Big Ball of Mud Pattern** (“shantytown,” “spaghetti code”)

Pattern

Architecture/scope not apparent
Most dev work reactive, not proactive
Functions not defined, documented
Clearly articulate system scope
Convolutd flow
Redundant code
Feels like treading water

Controls

Engage the team in analysis
Sell the long term benefits
Derive and document a structure
Define and document functional areas
Map logic paths
Impose strict coding procedures
Submit every change to review
Quarantine and rewrite unstable parts
Balance controls against creativity

➤ **Throwaway Code Pattern** (“Quick Hack”, “Permanent Prototype” or “Killer Demo”)

Pattern

Quick and dirty solution
Never intended for release
Casual structure
Fast path to domain knowledge
Poor or non-existent documentation

Controls

Discover and manage root business cause
Systematically drive instability out
Clarify functions
Modularize code
Quarantine and rebuild unstable chunks
Requires constant inspection

➤ **Piecemeal Growth Pattern** (“Urban Sprawl,” “Iterative-Incremental Development”)

Pattern

Independent modularized system
Adapts well to change
Change effect generally limited to modules
Easily re-configurable
Tends toward over-complexity

Controls

Consider the whole as well as the part
Rigorously control growth
Add modules only compelled by need
Concentrate attention on interfaces
Re-factor relentlessly

➤ **Keep It Working Pattern** (“Vitality,” “Baby Steps,” “Daily Build,” “First, Do No Harm”)

Pattern

Grown, rather than built
Miniscule incremental changes
Useful where 24x7 required
Maintainability takes precedence
Roll back strategy essential

Controls

Clear Goals/Visible Milestones
Constantly prove the system works
Small term changes, long term vision.
Require daily builds
Require rigorous regression testing
Roll back at the first sign of instability

➤ **Shearing Layers Pattern** (Artifacts Collected to Domains That Change at Different Rates)

Pattern

Organizational domains clearly defined
Relationships between domains are mapped
A hierarchy of domains is managed
How fast domains can accommodate change is known

Controls

Consider friction between close domains
Consider cascading changes to domains
Consider the timing of change
Anticipate delayed response to change
Shift fast changing elements to user control

➤ **Sweeping It Under The Rug Pattern** (“Pretty Face,” “Hiding It Under The Rug”)

Pattern

Teams become secretive, territorial
Members go dark, resist reviews
Management shielded from issues
Easier to hide than address problems

Controls

Expose and manage fear
Encourage team project responsibility
Take a systematic approach to correction
Contain and localize problems
Build in cross-training and peer reviews

➤ **Reconstruction Pattern** (“Total Rewrite,” “Demolition”)

Pattern

Architecture can't accommodate change
Code is beyond repair/comprehension
Original team members are gone
Project support has declined
Team morale is low

Controls

Post mortem of original design
Justify rewrite in business terms
Sell the team's experience
Alternatively re-factor:
--Replace/update components