Anti-Pattern Patterns

Best Practices to Avoid The Worst

DEVELPRENEUR
BUILDING BETTER DEVELOPERS

When you start examining a large number of anti-patterns, several patterns emerge.

– Observations From The Anti-Pattern Season

Best Practices

- An Anti-Pattern Review
- Some Common Anti-Patterns
- Communication
- Design/Planning
- Cutting Corners

What is an Anti-Pattern?

An anti-pattern is a common response to a recurring problem that is usually ineffective and risks being highly counterproductive. The term, coined in 1995 by Andrew Koenig,was inspired by a book, Design Patterns, which highlights a number of design patterns in software development that its authors considered to be highly reliable and effective.

Anti-Pattern Types

- Organizational - Bleeding Edge
- Project Management - Death March
- Software Design - Gold Plating
- OOP - God Object
- Programming - Spaghetti Code
- Methodological - Golden Hammer
- Configuration Management - Dependency Hell

DEVELPRENEUR
BUILDING BETTER DEVELOPERS

Common Anti-Patterns

- Analysis paralysis: A project stalled in the analysis phase, unable to achieve support for any of the potential plans of approach
- Bicycle shed: Giving disproportionate weight to trivial issues
- Bleeding edge: Operating with cutting-edge technologies that are still untested or unstable leading to cost overruns, under-performance or delayed delivery
- Bystander apathy: The phenomenon in which people are less likely to or do not offer help to a person in need when others are present
- Cash cow: A profitable legacy product that often leads to complacency about new products
- Design by committee: The result of having many contributors to a design, but no unifying vision
- Escalation of commitment: Failing to revoke a decision when it proves wrong

Common Anti-Patterns

- Abstraction inversion: Not exposing implemented functionality required by callers of a function/method/constructor, so that the calling code awkwardly re-implements the same functionality in terms of those calls
- Ambiguous viewpoint: Presenting a model (usually Object-oriented analysis and design (OOAD)) without specifying its viewpoint
- Big ball of mud: A system with no recognizable structure
- Database-as-IPC: Using a database as the message queue for routine interprocess communication where a much more lightweight mechanism would be suitable
- Gold plating: Continuing to work on a task or project well past the point at which extra effort is not adding value
- Inner-platform effect: A system so customizable as to become a poor replica of the software development platform
- Input kludge: Failing to specify and implement the handling of possibly invalid input

Communication

- Cross Teams
- Vertical information and sharing
- Within the team
- Plans, Goals, Visions, and Progress
- Open Channels

Design/Planning

- Big Picture design/SDLC
- No need to rush to implementation
- Address changes as needed
- Small scale design and planning
- Review overall approach and design

Cutting Corners

- Documentation
- Quick Fixes
- Skipping SDLC steps in part or entirely
- Best practices like code reviews
- Deadline over quality

Final Thoughts

- A few steps can help us avoid most anti-patterns
- Not Rocket Science
- Learn from mistakes
- Follow best practices for a reason
- Questions? Comments?


Date
Date

What We Learned

- There are a lot of anti-patterns to learn from
- Standards and consistency are key
- Communicate
- Design
- Follow Through

Thank You!

I appreciate your time and would love to discuss any of this further. You can send questions, comments and suggestions through any of these methods.

- info@develpreneur.com
- https://develpreneur.com/contact-us
- @develpreneur
- https://www.facebook.com/Develpreneur

Our goal is making every developer better.

DEVELPRENEUR
BUILDING BETTER DEVELOPERS