Software Development AntiPatterns

Mark Richards
Independent Consultant
Hands-on Software Architect / Published Author
Founder, DeveloperToArchitect.com
www.wmrichards.com

Author of *Software Architecture Fundamentals Video Series* (O’Reilly)
Author of Microservices Pitfalls and AntiPatterns (O’Reilly)
Author of Microservices vs. Service-Oriented Architecture (O’Reilly)
Author of *Enterprise Messaging Video Series* (O’Reilly)
Author of *Java Message Service 2nd Edition* (O’Reilly)
patterns
repeatable processes that produce positive results
anti-patterns

things we repeatedly do that produce negative consequences
what are the primary human traits that cause anti-patterns to occur?
haste
aggressive project deadlines and budgets drive lower acceptance levels for source code quality
apathy

general lack of care or concern about the problem or its outcome
ignorance

failure to seek a clear understanding of the problem or solution space
so many categories...

Economical
Organizational
Project Management
Analysis
Software Architecture
Software Development

Methodological
Testing
Requirements Management
Quality Assurance
Configuration Management
Enterprise Architecture
and too many anti-patterns!

Obligatory subcontracting
Funding me-too research
Repackaging as original
Analysis paralysis
Cash cow
Cost migration
Crisis mode
Design by committee
Escalation of commitment
Management by neglect
Management by numbers
Management by perkele
Management by wondering
Milk Monitor Promotion
Moral hazard
Mushroom management
Stovepipe
Vendor lock-in
Violin string organization
Puppet programming
Copy and paste programming
De-factoring
Golden hammer
Improbability factor
Low hanging fruit
Not built here
Premature optimization
Programming by permutation
Reinventing the square wheel
Reinventing the wheel
Silver bullet
Copper bullet
Tester Driven Development
Hostile testing
Meta-testing
Moving target
Re-coupling
Nurses-auditing-doctors
Turkish hat reform
Classpath hell
Dependency hell
DLL hell
Extension conflict
JAR hell
Magic Bullet
Chain Reaction
Ivory Tower
Buzzword-Driven Architecture
Death march
Groupthink
Smoke and mirrors
Software bloat
Bystander apathy
Napkin specification
Phony requirements
Retro-specification
Abstraction inversion
Ambiguous viewpoint
Big ball of mud
Blob
Gas factory
Input kludge
Interface bloat
Magic pushbutton
Race hazard
Railroaded solution
Re-coupling
Stovepipe system
Staralised schema
Anemic Domain Model
BaseBean
Call super
Circle-ellipse problem
Empty subclass failure
God object
Object cesspool
Object orgy
Poltergeists
Sequential coupling
Singletonitis
Yet Another Useless Layer
Yo-yo problem
Accidental complexity
Accumulate and fire
Action at a distance
Blind faith
Boat anchor
Bug magnet
Busy spin
Caching spin
Cargo cult programming
Checking type
Code momentum
Coding by exception
Error hiding
Expection handling
Hard code
Lava flow
Loop-switch sequence
Magic numbers
Magic strings
Monkey work
Packratting
Parallel protectionism
Ravioli code
Soft code
Spaghetti code
Wrapping wool in cotton

and many others...
Software Development AntiPatterns
software development anti-patterns

cargo cult programming

accidental complexity

lava flow
Cargo Cult Programming
AntiPattern
cargo cult programming anti-pattern

the practice of doing something without understanding the reasoning behind it
cargo cult programming anti-pattern

```java
if (year == 2016 || month.startsWith("M")) {
    System.out.println("true");
} else {
    System.out.println("false");
}
```
cargo cult programming anti-pattern

how would you parse this string into a String[] array?

String s = "BUY,AAPL,100,98.34";
cargo cult programming anti-pattern

String s = "BUY,AAPL,100,98.34";

List<String> array = new ArrayList<String>();
StringTokenizer st = new StringTokenizer(s, ",");
while (st.hasMoreTokens()) {
    array.add(st.nextToken());
}

String[] parts = new String[array.size()];
adarray.toArray(parts);
cargo cult programming anti-pattern

```java
class Account {
    private BigDecimal balance;
    private Date lastUpdatedDt;

    public BigDecimal getBalance() {
        return balance;
    }

    public void setBalance(BigDecimal balance) {
        this.balance = balance;
    }

    // other methods...
}
```
cargo cult programming anti-pattern

```java
public class Account {
    private BigDecimal balance;
    private Date lastUpdatedDt;
    ...

    public BigDecimal getBalance() {
        return (balance == null) ?
            new BigDecimal(0) : balance;
    }

    public void setBalance(BigDecimal balance) {
        this.balance = balance;
    }
    ...
}
```
cargo cult programming anti-pattern

```java
class Account {
    private BigDecimal balance;
    private long lastUpdatedDt;
    ...

    public Date getModifiedDate() {
        return new Date(lastUpdatedDt);
    }

    public void setModifiedDate(Date date) {
        this.lastUpdatedDt = date.getTime();
    }
    ...
}
```
"the rate of technology change will be so high that existing skills will be as outdated as quill and parchment. We will need to constantly re-skill ourselves just to keep a job." (1999)
cargo cult programming anti-pattern

we simply don't have time to read and learn everything there is to know about every technology!
cargo cult programming anti-pattern avoidance techniques take time to learn the language...

workerThread.yield();
workerThread.setDaemon(true);
doc.setHeader(stdHeader.intern());

know these packages and classes extremely well...

java.lang.*
java.io.*
java.math.*
java.text.*
java.util.*
cargo cult programming anti-pattern avoidance techniques
cargo cult programming anti-pattern
InfoQ
APM & Observability, NoOps, Effective Communication, Go for AWS Lambda, No JCP for Java EE, DynamoDB Streams, Vue.js, Resilience at Netflix Announcing QCon London day one keynote! Join Randy Shoup at QCon London and take away content...

InfoQ
Kubernetes & Docker Special Report
January 2018 Kubernetes &Docker Special Report Sponsored by Latest Content, Top Viewed Content, News, Top Articles, and up to date on all the new content and news related to Kubernetes and Docker on InfoQ. We are also maintaining a portal page...

InfoQ
Immutability in Finance, Deep Image Priors, ML at Quora, Kubernetes Service Mesh, Intel's Quantum Processor, JPA 2.2, Fast...

InfoQ
Streaming Architecture, Rapid Prototyping, TensorFlow and Kubernetes, Spectre and Meltdown, Rust, Go, Genetic Algorithms, Intel SGX, Scaling Slack, Chaos Engineering CRDTs and the Quest for Distributed Consistency @QConLondon 2018...

InfoQ
The key software trends in 2017 and what that tells us about 2018
2017 saw the continued rapid pace of software innovation and developments. We witnessed a dramatic growth in machine learning and saw the continued adoption of "cloud native" architectures (including Microservices) as well as growing interest in chaos engineering...

InfoQ
2017 Highlights, Monitoring Modern Architectures with Data Science, Rust, Go, Spring Boot Microservices, Targeting GPUs with C#, HTML 5.2, Chaos Engineering, Autism Inclusion How do we select speakers at QCon.ai? QCon.ai has no call for papers...

InfoQ
The Software Architects' Newsletter - December 2017
A monthly overview of things you need to know as an architect or aspiring architect. The Software Architects' Newsletter Desk sends our fifth monthly architects' newsletter with you, which we will send on the last Friday of every month. InfoQ strives to facilitate...
cargo cult programming anti-pattern
cargo cult programming anti-pattern
Accidental Complexity
AntiPattern
accidental complexity anti-pattern

introducing non-essential complexity into a problem
accidental complexity anti-pattern
accidental complexity anti-pattern
essential complexity
"we have a hard problem"

accidental complexity
"we have made a problem hard"

accidental complexity anti-pattern
accidental complexity anti-pattern

“developers are drawn to complexity like moths to a flame - frequently with the same result”

- Neal Ford
accidental complexity anti-pattern
accidental complexity anti-pattern
avoidance techniques

paired-programming
tool-based reviews

over-the-shoulder code reviews
All of these parentheses and semicolons totally ruin the feng shui of the code.

Sally learns an important lesson about inviting the right people to her code reviews.
accidental complexity anti-pattern

source validation tool

read source files

validation report

- check header standards
- check interceptors
- check contract standards

- check sql calls
- check audit writes
- check other stuff...
accidental complexity anti-pattern

registry

Map<String, String> pluginRegistry = new HashMap<String, String>();
static {
    pluginRegistry.put(HEADER, "ValidatorHeaderPlugin");
    pluginRegistry.put(SQL, "ValidatorSqlPlugin");
    pluginRegistry.put(INTERCEPT, "ValidatorInterceptPlugin");
    pluginRegistry.put(AUDIT, "ValidatorAuditPlugin");
}
accidental complexity anti-pattern
plug-in contracts

```java
public interface ValidatorPlugin {
    public ValidatorOutputData execute(ValidatorInputData data);
}

public class ValidatorInputData {
    public String moduleName;
    public List<String> moduleContents;
}

public class ValidatorOutputData {
    public String validationResults;
}
```
accidental complexity anti-pattern

core system

```java
private String executeChecks(String moduleName) throws Exception {
    for (Map.Entry<String, String> entry : pluginRegistry.entrySet()) {
        Class<?> c = Class.forName(PLUGIN_PKG + entry.getValue());
        Constructor<?> con = c.getConstructor();
        ValidatorPlugin plugin = (ValidatorPlugin)con.newInstance();
        outputData = plugin.execute(inputData);
        System.out.println(outputData.validationResults);
    }
}
```
Lava Flow AntiPattern
lava flow anti-pattern

dead code solidifies into hardened globules within fluid code, resulting in unnecessary maintenance
public void placeOrder(Order order) {
    insertOrder(order);
    updateInventory(order);
    //checkOverstockDiscount(order);
    //calculateStateTax(order);
    processPayment(order);
}

lava flow anti-pattern

dead code

public void placeOrder(Order order) {
    insertOrder(order);
    updateInventory(order);
    //checkOverstockDiscount(order);
    //calculateStateTax(order);
    processPayment(order);
}
lava flow anti-pattern

dead code

dead class or interface

dead function or method

dead control statements

dead parameter or return value

dead variable
lava flow anti-pattern

dead artifacts

old reports

converted reports
lava flow anti-pattern
avoidance techniques

leverage version control
Software Development AntiPatterns

Mark Richards
Independent Consultant
Hands-on Software Architect / Published Author
Founder, DeveloperToArchitect.com
www.wmrichards.com

Author of Software Architecture Fundamentals Video Series (O’Reilly)
Author of Microservices Pitfalls and AntiPatterns (O’Reilly)
Author of Microservices vs. Service-Oriented Architecture (O’Reilly)
Author of Enterprise Messaging Video Series (O’Reilly)
Author of Java Message Service 2nd Edition (O’Reilly)
GREAT INDIAN DEVELOPER SUMMIT 2019
Conference: April 23-26, Bangalore

Register early and get the best discounts!

www.developersummit.com @greatindiandev bit.ly/gidslinkedin
facebook.com/gids19 bit.ly/saltmarchyoutube
flickr.com/photos/saltmarch/