

# ***Software Patterns and Quality***

***“Can Patterns Help Quality?”***

**Joseph W. Yoder**

**The Refactory, Inc.**

**joe@refactory.com**

**<http://www.refactory.com>**

**<http://www.joeyoder.com>**

# Outline

- Software Patterns & Quality
- Quality without a Name (QWAN)
- Different Qualities of a System
- Design and Quality Tradeoffs
- Summary

# Quality

- ◆ **Quality Definition:** a peculiar and essential character or nature, an inherent feature or property, a degree of excellence or grade, a distinguishing attribute or characteristic



# Quality (Who's perspective)

Artist important/boring	Scientist true/false
Designer cool/uncool	Engineer good/bad

“The Four Winds of Making”...Gabriel

An architect might have perspectives from an artist to a design or an engineer!

Rich Gold "The Plenitude: Creativity, Innovation & Making Stuff (Simplicity: Design, Technology, Business, Life)"

Triggers and Practices – Richard Gabriel <http://www.dreamsongs.com>

# Quality Definitions

- ISO 9000 - "Degree to which a set of inherent characteristic fulfills requirements"
- (Noriaki Kano and others) - A two-dimensional model of quality: "must-be quality" and "attractive quality". The former is near to the "fitness for use" and the latter is what the customer would love, but has not yet thought about.
- (Joseph M. Juran) - "Fitness for use". Fitness is defined by the customer.
- (Philip B. Crosby in the 1980s) - "Conformance to requirements".
- (Gerald M. Weinberg) - "Value to some person"

# Quality by Attributes

**Quality** in everyday life and business, engineering and manufacturing can be seen as the *usefulness* of something. Usually describes certain attributes.

- ◆ For example you could describe quality in terms of performance, reliability (fault tolerance), safety, maintainability, reusability, etc...

Does quality on the inside mean quality on the outside?

# Non-functional Requirements

Accessibility

Compatibility

Efficiency

Effectiveness

Extensibility

Maintainability

Performance

Reliability

Safety

Scalability

Security

Stability

Supportability

Usability

Other terms for non-functional requirements are "constraints", "quality attributes", and "quality of service requirements"

Qualities are usually described by "ilities" as seen in non-functional requirements...but quality can also focus on how well the functional requirements are met (how to measure this?)

# Software Quality

From a software engineering perspective, quality can be seen as:

How well software is designed (*quality of design*)

How well the software conforms to that design (*quality of conformance*)

How excellent the implementation is (quality of the code ... )

*Quality of design* can be seen as how valid the design and requirements are in meeting the requirements or creating a the "right" product

*Quality of conformance* is focused more on implementation

A definition in Steve McConnell's *Code Complete* similarly divides software into two pieces: **internal** and **external quality characteristics**.

External quality characteristics are those parts of a product that face its users, where internal quality characteristics are those that do not.

A definition by Tom DeMarco says "a product's quality is a function of how much it changes the world for the better". But whose definition do we use for changing the world for the better? Quality can be seen as a value to some person. Is the person a developer, an end user, a manager, the CEO, the CFO?

# Software Quality

Why has quality been so elusive for software.

*Japan leads the world in manufacturing quality partly because of SQC.*

*Why has SQC and other manufacturing techniques failed with SW?*

How can we raise the quality of our software systems?

Is high quality software up to this point more of an art than a science?

Whose perspective on quality are we looking at:

- The Developer
- End User
- Manager
- CEO

Quality Costs...are we willing to pay for it and how do we maintain it?

Need to deliver quality software ontime and within budget!!!

# Are Patterns the Answer?

We hear a lot of hype about patterns.

Thousands of Patterns published!!!

Proven best known practices in Industry

Do Patterns help embed specific qualities into a software system?

“Patterns and Pattern Languages are ways to describe best practices, good designs, and capture experience in a way that it is possible for others to reuse this experience”

# What about Patterns?

Quite often you hear:

“A pattern is a proven solution to a problem in a context.”

Alexander writes:

“Each pattern is a three-part rule, which expresses a relation between a certain context, a problem, and a solution.”

Are these definitions the same?

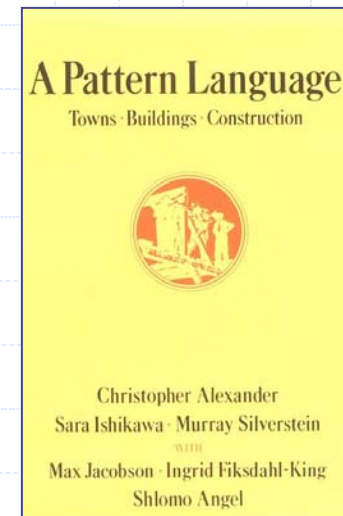
“Good Judgment comes from Experience comes from bad Judgment....Patterns come from Experience”

# Alexander's Pattern Definition

Each pattern describes a problem that occurs over and over again in our environment and then describes the core of the solution to that problem in such a way that you can use this solution a million times over without ever doing it the same way twice.

Alexander - building architect and author

- *The Timeless Way of Building*
- *A Pattern Language*



# What is a Pattern

It must be a solution to a problem in a context

You must be able to tell the problem solver what to do,  
how to solve the problem

It must be a mature, proven solution (Rules of 3)

It must contribute to human comfort or "*quality*" of life

It must be something you didn't invent yourself  
(Buschmann's Rule)

The solution should build on the insight of the problem solver,  
and can be implemented a million times without ever being  
the same twice

It cannot be formalized or automated  
(if it can, do that instead of writing a pattern)

It should have a dense set of interacting forces that are  
independent of the forces in other patterns

# What is a Pattern

Patterns can be thought of “Best Practices”  
Proven Solutions to Repeating Problems  
Embody Experiences of What Works...  
...and What Doesn't Work  
Captures or Describes Knowledge of Experts

Embody “Quality” Attributes for  
Solutions to specific Problems

# Alexander on Patterns

To seek the timeless way we must first know the quality without a name.

There is a central quality which is the root criterion of life and spirit in a man, a town, a building, or a wilderness. This "*quality*" is objective and precise, but it cannot be named.

I was no longer willing to start looking at any pattern unless it presented itself to me as having the capacity to connect up with some part of this quality [the quality without a name]. Unless a particular pattern actually was capable of generating the kind of life and spirit that we are now discussing, and that it had this quality itself, my tendency was to dismiss it, even though we explored many, many patterns.

# Quality Without a Name (QWAN)

- ◆ *From patterns perspective, Alexander looked at QWAN as “the quality” that imparts incommunicable beauty and immeasurable value to a structure. It encompasses all of the following:*

Universally recognizable aesthetic beauty and order  
Recursively nested centers of symmetry and balance  
Life and wholeness  
Resilience, adaptability, and durability  
Human comfort and satisfaction  
Emotional and cognitive resonance



The Samurai Hasekura

# Quality Without a Name (QWAN)

- ◆ So a crucial aspect of the QWAN is the effect it has upon the architecture's inhabitants and designers that makes them feel alive, whole, and comfortable. It is this kind of "habitability" that improves user comfort and quality of life (what TQM circles might refer to as "total customer satisfaction", and what Tom Peters means by "to thrill and delight" the customer in the pursuit of "Wow!").
- ◆ What makes that design cool, or what makes the system impressive.

# Problems with Quality

Here's the problem with Quality: too many people try to Quantify it. Quality is hard to quantify, or we'd call it Quantity. What makes a book good, a movie, a painting, ... should we measure the length of the Godfather II, and make all our movies that long, and achieve the same quality?

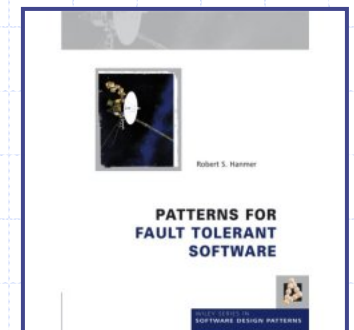
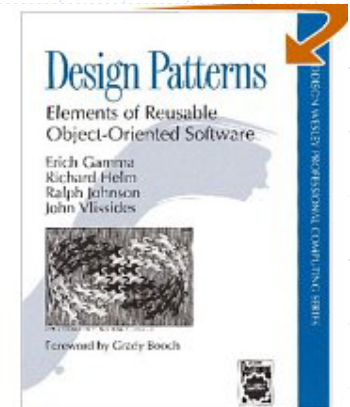
Hemingway was one of Literary Engineering's leading figure, so we should study his sentence and paragraph lengths, and ensure our sentences and paragraphs conform to these Hemmingway metrics. By doing so, would we make sure our literary artifacts are engineered with the same precision quality as Ernie achieved?

Metrics can help a little when it comes to quality but miss the point. QWAN can't really be quantify.

How about: "Quality Without A Number...(QWANum)"

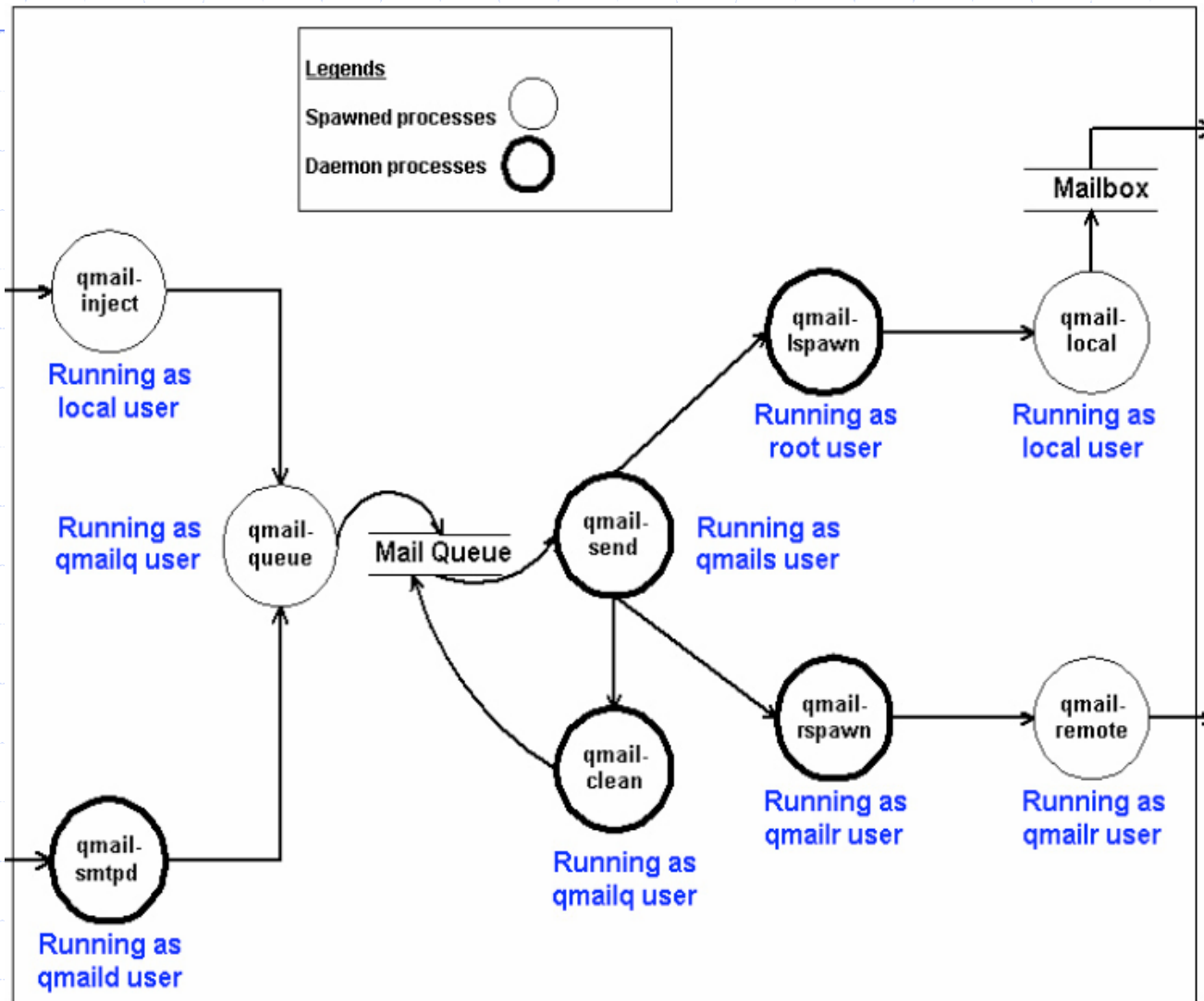
# Patterns and Quality

- ◆ In a sense, patterns are all about quality
- ◆ Design Patterns were about giving software the “quality” of being more reusable and maintainable...  
Quality of a good OO Design
- ◆ Fault Tolerance Patterns are about proven practices that help ensure build systems with the “quality” of being able to handle faults



# The Security Architecture of qmail

*"an example of quality"*



# QMail Patterns

Hafiz, Johnson, Afandi – PLoP '04

## Security Pattern: Compartmentalization [VM02]

### Problem

A security failure in one part of a system allows another part of the system to be exploited.

### Solution

Put each part in a separate security domain. Even when the security of one part is compromised, the other parts remain secure.

## Performance Pattern: Small Processes

### Problem

A program memory processes can be limited by the memory used by the processes. If the processes grow unbounded, then there is a potential DoS scenario. How can a program with many processes be made safe from resource exhaustion?

### Solution

Make the processes small. Each process should perform one task. This will ensure that processes allocate limited memory.

## Security Pattern: Distributed Responsibility

### Problem

A security failure in a compartment can change any data in that compartment. A compartment has both an interface that is at risk of a security failure and data that needs to be secure.

### Solution

Partition responsibility across compartments such that compartments that are likely to fail do not have data that needs to be secure. Assign responsibilities in such a way that several of them need to fail in order for the system as a whole to fail.

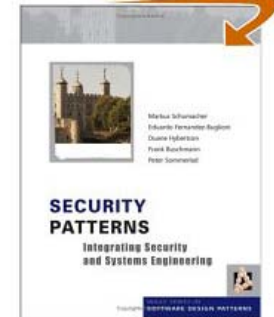
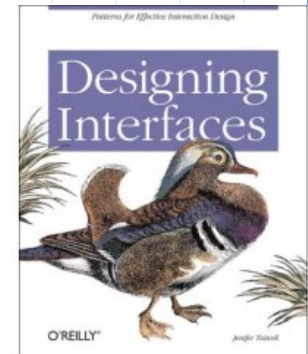
This is called Distributed Delegation by Varyard and Ward. [VW01]

# QMail Patterns and Quality

- ◆ Compartmentalize of Small Light Weight Processes for QMail done for Security, also was good for reliability, and has a good side affect of performance (some qualities are "generated qualities")
- ◆ Moral of Story -- Patterns sometimes can be applied for one quality and can benefit other qualities

# Design is about Tradeoffs

- ◆ Usability and Security often have orthogonal qualities
  - **Designing Interfaces: Patterns for Effective Interaction Design**
  - **Security Patterns: Integrating Security and Systems Engineering**



- ◆ Performance vs Small Memory



# When Dealing with Quality

Can you take quality too far?

# Big Ball of Mud

Alias: Shantytown, Spaghetti Code



A BIG BALL OF MUD is haphazardly structured, sprawling, sloppy, duct-tape and bailing wire, spaghetti code jungle.

The de-facto standard software architecture. Why is the gap between what we **preach** and what we **practice** so large?



We preach we want to build high quality systems but why are BBoMs so prevalent?

# Worse is Better

Ideas resembles Gabriel's 1991  
"Worse is Better"

Worse is Better is an argument to release early and then have the market help you design the final product. It is taken as the first published argument for open source, among other things.

Do BBoM systems have a Quality?

# Worse is Better (examples)

## Betamax vs VHS Format

- Why did VHS win?
- Betamax was arguably a better format

## Macintosh vs Windows

- Mac was easier to use
- Far superior in many ways

## MS Word/Publisher vs FrameMaker

- Lot's of people use word
- FrameMaker is better for books

# Being Good Enough

- Quality of being good enough
- Does it meet the minimum requirements
- Quality has many competing forces...are we designing a system for online orders or for controlling the space shuttle, they have different qualities, thus different patterns and solutions apply

# Brooklyn Bridge

- ◆ The **Brooklyn Bridge**, 1883, one of the oldest suspension bridges in the United States, stretches over a mile from Brooklyn to Manhattan.
- ◆ On completion, it was the largest suspension bridge in the world and the first steel-wire suspension bridge.



# Brooklyn Bridge

- ◆ Designed by John Augustus Roebling
- ◆ His son, Washington, succeeded him, but was stricken with caisson disease (decompression sickness, commonly known as "the bends").



# Brooklyn Bridge

- ◆ The occurrence of the bends caused Washington to halt construction of the Manhattan side of the tower 30 feet (10 m) short of bedrock. Today, the Manhattan tower rests only on sand.
- ◆ This is a big Change of the Design! But it has been proven that the quality was good enough. Cost of human life was too high.



# Brooklyn Bridge

- ◆ Over engineered. Had 6 times what it needed which proved useful over time
- ◆ What happens if we tried overdesign of our systems (a language for printing hello world) or the same line of code 6 times (is this 6 times more reliable?)
  - if (x != a) x = a; if (x != a) x = a; if (x != a) x = a;  
if (x != a) x = a; if (x != a) x = a; if (x != a) x = a;
- ◆ Redundant components can make our systems more reliable



# Many Quality Patterns Written

Design Patterns

Patterns for Fault Tolerant Software

Performance Patterns

Small Memory Software Patterns

Analysis Patterns

Security Patterns

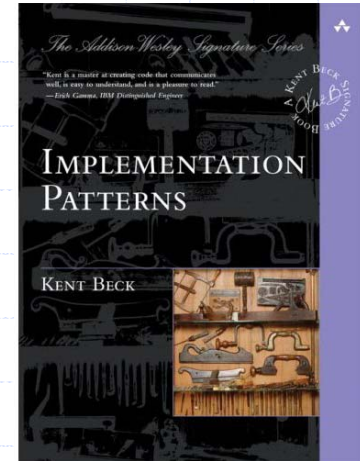
Stability Patterns

Usability Patterns

*Imitate or use proven quality techniques*

# Implementation Patterns

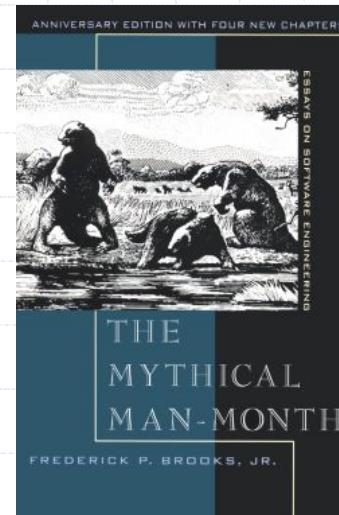
- Patterns about creating quality code that communicates well, is easy to understand, and is a pleasure to read. Book is about patterns of "Quality" code.
- But...Kent states, "...this book is built on a fragile premise: that good code matters. I've seen too much ugly code make too much money to believe that quality of code is either necessary or sufficient for commercial success or widespread use. However I still believe *quality* of code matters."
- Patterns assist with making code more bug free and easier to maintain and extend.



# Silver Buckshot

- ◆ There are no silver bullets  
...Fred Brooks
- ◆ But maybe some silver buckshot  
...promising attacks

Good Design  
Frameworks  
Patterns  
Architecture  
Process/Organization  
Tools  
Good People



# Draining the Swamp

You can escape from the  
*"Spaghetti Code Jungle"*

Indeed you can transform the landscape  
The key is not some magic bullet, but a  
long-term commitment to **architecture**,  
and to cultivating and refining *"quality"*  
**artifacts** for your domain (Refactoring)!

Patterns of the best practices can help!!!

# Themes

Theme 1: Quality is not easy to Quantify

Theme 2: To get quality, imitate quality  
(Patterns do that)

Patterns are about diversity, heterogeneity,  
variety, choice, not about nail it

They are about an amalgam of different proven  
ideas, not an unachievable ideal

Theme 3: Perfection is the enemy of flexibility, a  
false idol, a mirage, overkill, etc

# Summary

Perfect is the enemy of the adequate, and SQC glories perfection, and turns up its nose at the merely adequate

Studying and imitating quality is better than trying to quantify it and thereby eliminate "anti-quality" statistically.

Patterns say go for the good stuff, do what we know works and avoid what we know doesn't work.

This is the Quality of Patterns, focus on good principles rather than trying to count defects and eliminate them, that's beside the point

# Patterns can Lead to Quality of Life 😊



# Hillside Group Vision

- ◆ The Hillside Group is founded on the observation that software creation is one of the most difficult human endeavors, requiring the creation of novelty under pressure, without the benefits of a long tradition to fall back on...The world of software development is a mixture of concerns ranging from correctness and execution efficiency to the beauty and elegance of the architecture, design, and internal structure of systems to the overall aesthetics, usability, and humanity of systems and all the way to the organization of development and the manner of software production... Primary focus is on concerns about the qualities of our software systems including the process that produced it.

# Hillside Group Mission

The mission of the Hillside Group is to improve the quality of life of everyone who uses, builds, and encounters software systems-users, developers, managers, owners, educators, students, and society as a whole...The Hillside Group promotes the use of patterns and pattern languages to record, analyze, and improve software and its development, and supports any new practices that help achieve its mission.

# PLOP® Conferences

*www.hillside.net*



MensorePLOP

vikingPLOP

KOALAPLOP

EuroPLOP

Chili  
PLOP

SugarLoafPLOP

# Where to Find More Information and Acknowledgements

- <http://hillside.net>
- <http://www.refactory.com>
- <http://www.joeyoder.com>
- <http://www.dreamsongs.com>

Thanks to Ralph Johnson, Richard Gabriel,  
and Brian Foote for feedback on this talk.

# That's All

