

A Question of Craftsmanship

Kevlin Henney

kevlin@curbralan.com

@KevlinHenney



Copyrighted Material

WILEY SERIES IN
SOFTWARE DESIGN PATTERNS

PATTERN-ORIENTED SOFTWARE ARCHITECTURE

A Pattern Language for
Distributed Computing

Volume 4



Frank Buschmann
Kevlin Henney
Douglas C. Schmidt

Copyrighted Material



Copyrighted Material

WILEY SERIES IN
SOFTWARE DESIGN PATTERNS

PATTERN-ORIENTED SOFTWARE ARCHITECTURE

On Patterns and Pattern Languages

Volume 5



Frank Buschmann
Kevlin Henney
Douglas C. Schmidt

Copyrighted Material

97



Collective Wisdom
from the Experts

97 Things Every Programmer Should Know

O'REILLY®

Edited by Kevlin Henney

Art. Craft. Engineering. Science. These are the swirling muses of design patterns. Art and science are stories; craft and engineering are actions.

Craft is midway between art and science; art and craft stand over against engineering and science. Art is the unique example, the first thing, the story as artifact condensing out of talent and desire. Craft is reliable production of quality. A craftsman might be disappointed but rarely fails. A work of craft is the product of a person and materials. Engineering is reliable and efficient production of things for the use and convenience of people. Science is a process of making a story that can be used for engineering.

Wayne Cool

foreword to
*Pattern-Oriented Software Architecture, Volume 5:
On Patterns and Pattern Languages*


```

/* grep: search for regexp in file */
int grep(char *regexp, FILE *f, char *name)
{
    int n, nmatch;
    char buf[BUFSIZ];

    nmatch = 0;
    while (fgets(buf, sizeof buf, f) != NULL)
        n = strlen(buf);
        if (n > 0 && buf[n-1] == '\n')
            buf[n-1] = '\0';
        if (match(regexp, buf)) {
            nmatch++;
            if (name != NULL)
                printf("%s:", name);
            printf("%s\n", buf);
        }
    }
    return nmatch;
}
    
```

```

/* matchhere: search for regexp at beginning of text */
int matchhere(char *regexp, char *text)
{
    if (regexp[0] == '\0')
        return 1;
    if (regexp[1] == '*')
        return matchstar(regexp[0], regexp+1, text);
    if (regexp[0] == '$' && regexp[1] == '\0')
        return *text == '\0';
    if (*text != '\0' && (regexp[0] == '.' || reg
        return matchhere(regexp+1, text+1);
    return 0;
}
    
```

```

/* match: search for regexp anywhere in text */
int match(char *regexp, char *text)
{
    if (regexp[0] == '^')
        return matchhere(regexp+1, text);
    do {
        /* must look even if string is empty */
        if (matchhere(regexp, text))
            return 1;
    } while (*text++ != '\0');
    return 0;
}
    
```

```

/* matchstar: search for c*regexp at beginning of text */
int matchstar(int c, char *regexp, char *text)
{
    do {
        /* a * matches zero or more instances */
        if (matchhere(regexp, text))
            return 1;
    } while (*text != '\0' && (*text++ == c || c == '.'));
    return 0;
}
    
```

LD E H	95	PET Z	200
LD D H	22 67	ADD L	133
CALL PNOVE	205 255 66	LD L A	111
END	217	LD A (HL)	126
RND A	167	RET	201

```

import re, collections

def words(text): return re.findall('[a-z]+', text.lower())

def train(features):
    model = collections.defaultdict(lambda: 1)
    for f in features:
        model[f] += 1
    return model

NWORDS = train(words(file('big.txt').read()))

alphabet = 'abcdefghijklmnopqrstuvwxyz'

def edits1(word):
    splits = [(word[:i], word[i:]) for i in range(len(word) + 1)]
    deletes = [a + b[1:] for a, b in splits if b]
    transposes = [a + b[1] + b[0] + b[2:] for a, b in splits if len(b)>1]
    replaces = [a + c + b[1:] for a, b in splits for c in alphabet if b]
    inserts = [a + c + b for a, b in splits for c in alphabet]
    return set(deletes + transposes + replaces + inserts)

def known_edits2(word):
    return set(e2 for e1 in edits1(word) for e2 in edits1(e1) if e2 in NWORDS)

def known(words):
    return set(w for w in words if w in NWORDS or
                (define (eval exp env)
                    (lookup-variable-value exp env)
                    (next-of-quotation exp))
                ((assignment? exp) (eval-assignment exp env))
                ((definition? exp) (eval-definition exp env))
                ((if? exp) (eval-if exp env))
                ((lambda? exp)
                 (make-procedure (lambda-parameters exp)
                                 (lambda-body exp)
                                 env))
                ((begin? exp)
                 (eval-sequence (begin-actions exp) env))
                ((cond? exp) (eval (cond->if exp) env))
                ((application? exp)
                 (apply (eval (operator exp) env)
                        (list-of-values (operands exp) env)))
                (else
                 (error "Unknown expression type - EVAL" exp))))
    
```

/^1?\$/|(11+?)\1+\$/

```
return max(candi
```

P67

?

?

W?

RESTART?

CALL

CALL

CALL

JP Z

LD HL

```

(define (eval exp env)
    (lookup-variable-value exp env)
    (next-of-quotation exp))
((assignment? exp) (eval-assignment exp env))
((definition? exp) (eval-definition exp env))
((if? exp) (eval-if exp env))
((lambda? exp)
 (make-procedure (lambda-parameters exp)
                  (lambda-body exp)
                  env))
((begin? exp)
 (eval-sequence (begin-actions exp) env))
((cond? exp) (eval (cond->if exp) env))
((application? exp)
 (apply (eval (operator exp) env)
         (list-of-values (operands exp) env)))
(else
 (error "Unknown expression type - EVAL" exp)))
    
```



Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

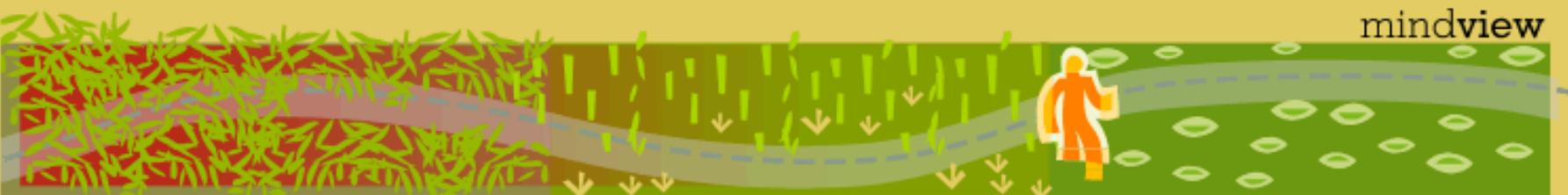
That is, while there is value in the items on the right, we value the items on the left more.

Anyway, the point I'm making is that we need to be very careful. Just as they're doing with management consultants, sooner or later our customers will figure out that the management practices of Agile don't deliver working software any more than they can grill cheese or power an electric bicycle.

The business of software is software. We don't make bean bags and we don't sell ice cream, even if that's what will solve the customer's problems. We make working software. And the *Manifesto for Agile Software Development* is a manifesto for doing it better.

If you don't know how to make software, then I'm afraid you've boarded the wrong train, my friend. This train is going to *Better Software*. The train for *Management Snake Oil* leaves from a different platform. You can't miss it. It's made of invisible gold and it runs on magic beans.

Jason Gorman, "We Emulate Management Consultants at Our Peril"
<http://parlezuml.com/blog/?postid=946>



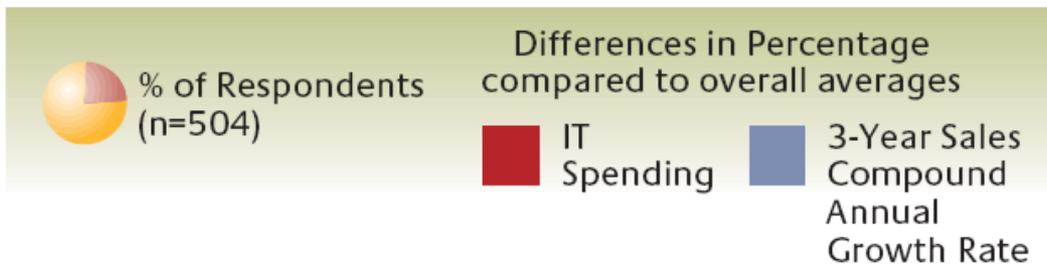
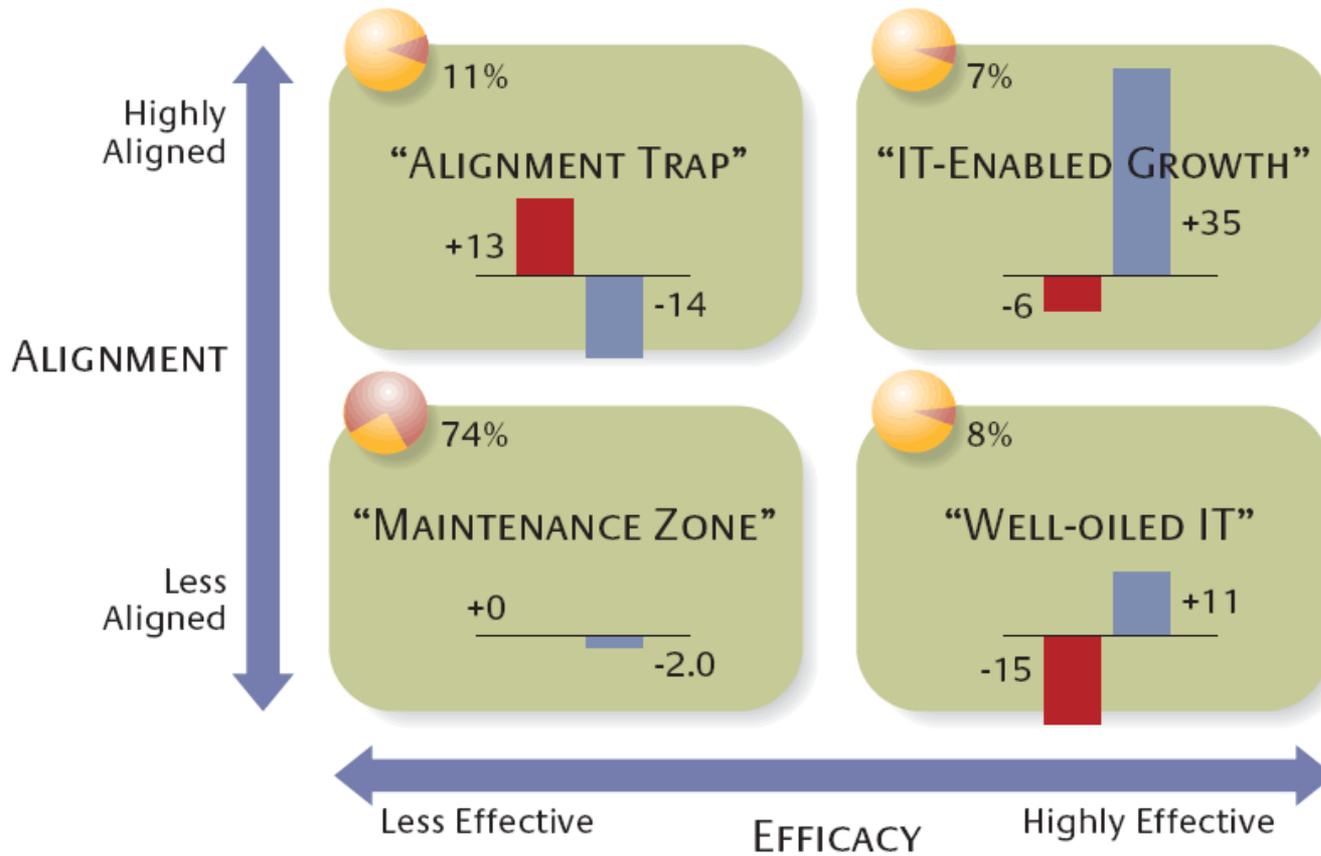
7-20-03 The Ideal Programmer

In January of this year, Scott Meyers (author of [Effective C++](#)), Bill Venners, Chuck Allison and myself convened a technical summit in Portland called "Writing Better Code." This was a small group that we invited from among people who had spent time thinking and writing about the issues around creation of code and projects. The list included:

- Scott Meyers
- Bruce Eckel
- Bill Venners
- Chuck Allison
- Joshua Bloch
- Alistair Cockburn
- Dave Thomas
- Andrew Hunt
- Pete McBreen
- Angelika Langer
- Chris Sells
- Kevlin Henney
- Randy Stafford
- Matt Gerrans

Links I Read

[Cafe Au Lait](#)
[Artima](#)
[Daily Python URL](#)
[Martin Fowler](#)
[Joel on Software](#)
[Paul Graham](#)
[Cringely](#)



Manifesto for Software Craftsmanship

Raising the bar.

As aspiring Software Craftsmen we are raising the bar of professional software development by practicing it and helping others learn the craft. Through this work we have come to value:

Not only working software,
but also **well-crafted software**

Not only responding to change,
but also **steadily adding value**

Not only individuals and interactions,
but also **a community of professionals**

Not only customer collaboration,
but also **productive partnerships**

That is, in pursuit of the items on the left we have found the items on the right to be indispensable.

FONDAZIONE E MANIFESTO

DEL

FUTURISMO

(Pubblicato dal "FIGARO", di Parigi il 20 Febbraio 1909)

Avevamo vegliato tutta la notte — i miei amici ed io — sotto lampade di moschea dalle cupole di ottone traforato, stellate come le nostre anime, perchè come queste irradiate dal chiuso fulgore di un cuore elettrico. Avevamo lungamente calpestata su opulenti tappeti orientali la nostra atavica accidia, discutendo davanti ai confini estremi della logica e annerendo molta carta di frenetiche scritture.

Un immenso orgoglio gonfiava i nostri petti, poichè ci sentivamo soli, in quell'ora, ad esser desti e ritti, come fari superbi e come sentinelle avanzate, di fronte all'esercito delle stelle nemiche occhieggianti dai loro celesti accampamenti. Soli coi fuochisti che s'agitano davanti ai fornelli infernali delle grandi navi, soli coi neri fantasmi che frugano nelle pance arroventate delle locomotive lanciate a pazza corsa, soli cogli ubbriachi annaspanti, con un incerto batter d'ali, lungo i muri della città.

Sussultammo ad un tratto, all'udire il rumore formidabile degli enormi tramvai a due piani, che passano sobbalzando, risplendenti di luci multicolori, come i villaggi in festa che il Po straripato squassa e stradica d'improvviso, per trascinarli fino al mare, sulle cascate e attraverso i gorgi di un diluvio.

Poi, il silenzio divenne più cupo. Ma mentre ascoltavamo l'estenuato borbottio di preghiere del vecchio canale e lo scricchiar dell'ossa dei palazzi moribondi sulle loro barbe di umida verdura, noi udimmo subitamente ruggire sotto le nostre finestre gli automobili famelici.

— Andiamo, diss'io; andiamo, amici! Partiamo! Finalmente la mitologia e l'ideale mistico sono superati. Noi stiamo per assistere alla nascita del Centauro e presto vedremo volare i primi Angeli!... Bisognerà scuotere le porte della vita per provarne i cardini e i chiavistelli!... Partiamo! Ecco, sulla terra, la primissima aurora! Non v'è cosa che agguagli lo splendore della rossa spada del sole, che schermeggia per la prima volta nelle nostre tenebre millenarie!... —

We have been up all night, my friends and I, beneath mosque lamps whose brass cupolas are bright as our souls, because like them they were illuminated by the internal glow of electric hearts. And trampling underfoot our native sloth on opulent Persian carpets, we have been discussing right up to the limits of logic and scrawling the paper with demented writing.

Our hearts were filled with an immense pride at feeling ourselves standing quite alone, like lighthouses or like the sentinels in an outpost, facing the army of enemy stars encamped in their celestial bivouacs. Alone with the engineers in the infernal stokeholes of great ships, alone with the black spirits which rage in the belly of rogue locomotives, alone with the drunkards beating their wings against the walls.

1. We want to sing the love of danger, the habit of energy and rashness.
2. The essential elements of our poetry will be courage, audacity and revolt.
3. Literature has up to now magnified pensive immobility, ecstasy and slumber. We want to exalt movements of aggression, feverish sleeplessness, the double march, the perilous leap, the slap and the blow with the fist.
4. We declare that the splendour of the world has been enriched by a new beauty: the beauty of speed. A racing automobile with its bonnet adorned with great tubes like serpents with explosive breath ... a roaring motor car which seems to run on machine-gun fire, is more beautiful than the Victory of Samothrace.
5. We want to sing the man at the wheel, the ideal axis of which crosses the earth, itself hurled along its orbit.
6. The poet must spend himself with warmth, glamour and prodigality to increase the enthusiastic fervour of the primordial elements.
7. Beauty exists only in struggle. There is no masterpiece that has not an aggressive character. Poetry must be a violent assault on the forces of the unknown, to force them to bow before man.
8. We are on the extreme promontory of the centuries! What is the use of looking behind at the moment when we must open the mysterious shutters of the impossible? Time and Space died yesterday. We are already living in the absolute, since we have already created eternal, omnipresent speed.
9. We want to glorify war – the only cure for the world – militarism, patriotism, the destructive gesture of the anarchists, the beautiful ideas which kill, and contempt for woman.
10. We want to demolish museums and libraries, fight morality, feminism and all opportunist and utilitarian cowardice.
11. We will sing of the great crowds agitated by work, pleasure and revolt; the multi-coloured and polyphonic surf of revolutions in modern capitals: the nocturnal vibration of the arsenals and the workshops beneath their violent electric moons: the gluttonous railway stations devouring smoking serpents; factories suspended from the clouds by the thread of their smoke; bridges with the leap of gymnasts flung across the diabolic cutlery of sunny rivers: adventurous steamers sniffing the horizon; great-breasted locomotives, puffing on the rails like enormous steel horses with long tubes for bridle, and the gliding flight of aeroplanes whose propeller sounds like the flapping of a flag and the applause of enthusiastic crowds.

MANIFEST I OF „THE STYLE”, 1918.

1. There is an old and a new consciousness of time.

The old is connected with the individual.

The new is connected with the universal.

The struggle of the individual against the universal is revealing itself in the world-war as well as in the art of the present day.

2. The war is destroying the old world with its contents: individual domination in every state.

3. The new art has brought forward what the new consciousness of time contains: a balance between the universal and the individual.

4. The new consciousness is prepared to realise the internal life as well as the external life.

5. Traditions, dogmas and the domination of the individual are opposed to this realisation.

6. The founders of the new plastic art therefore call upon all, who believe in the reformation of art and culture, to annihilate these obstacles of development, as they have annihilated in the new plastic art (by abolishing natural form) that, which prevents the clear expression of art, the utmost consequence of all art notion.

7. The artists of to-day have been driven the whole world over by the same consciousness, and therefore have taken part from an intellectual point of view in this war against the domination of individual despotism. They therefore sympathize with all, who work for the formation of an international unity in Life, Art, Culture, either intellectually or materially.

8. The monthly editions of „The Style”, founded for that purpose, try to attain the new wisdom of life in an exact manner.

9. Co-operation is possible by:

I. Sending, with entire approval, name, address and profession to the editor of „The Style”.

II. Sending critical, philosophical, architectural, scientific, literary, musical articles or reproductions.

III. Translating articles in different languages or distributing thoughts published in „The Style”.

Signatures of the present collaborators:

THEO VAN DOESBURG, Painter.

ROBT. VAN 'T HOFF, Architect.

VILMOS HUSZAR, Painter.

ANTONY KOK, Poet.

PIET MONDRIAAN, Painter.

G. VANTONGERLOO, Sculptor.

JAN WILS, Architect.

40 CENTS

THE
**Communist
Manifesto**

382.14



KARL MARX & FREDERICK ENGELS

INTERNATIONAL



PUBLISHERS

Notes on Postmodern Programming

James Noble, Robert Biddle
Computer Science,
Victoria University of Wellington, New Zealand.
{robert,kjx}@mcs.vuw.ac.nz

March 23, 2002

0 Manifesto

The ultimate goal of all computer science is the program. The performance of programs was once the noblest function of computer science, and computer science was indispensable to great programs. Today, programming and computer science exist in complacent isolation, and can only be rescued by the conscious co-operation and collaboration of all programmers.

The universities were unable to produce this unity; and how indeed, should they have done so, since creativity cannot be taught? Designers, programmers and engineers must once again come to know and comprehend the composite character of a program, both as an entity and in terms of its various parts. Then their work will be filled with that true software spirit which, as "theory of computing", it has lost. Universities must return to programming. The worlds of the formal methods and algorithm analysis, consisting only of logic and mathematics, must become once again a world in which things are built. If the young person who rejoices in creative activity now begins his career as in the older days by learning to program, then the unproductive "scientist" will no longer be condemned to inadequate science, for their skills will be preserved for the programming in which they can achieve great things.

Designers, programmers, engineers, we must all return to programming! There is no essential difference between the computer scientist and the programmer. The computer scientist is an exalted programmer. By the grace of Heaven and in rare moments of inspiration which transcend the will, computer science may unconsciously blossom from the labour of the hand, but a base in programming is essential to every computer scientist. It is there that the original source of creativity lies.

Let us therefore create a new guild of programmers without the class-distinctions that raise an arrogant barrier between programmers and computer scientists! Let us desire, conceive, and create the new program of the future together. It will combine design, user-interfaces, and programming in a single form, and will one day rise towards the heavens from the hands of a million workers as the crystalline symbol of a new and coming faith.

Online Markets...

Networked markets are beginning to self-organize faster than the companies that have traditionally served them. Thanks to the web, markets are becoming better informed, smarter, and more demanding of qualities missing from most business organizations.

...People of Earth

The sky is open to the stars. Clouds roll over us night and day. Oceans rise and fall. Whatever you may have heard, this is our world, our place to be. Whatever you've been told, our flags fly free.

Our heart goes on forever. People of Earth, remember.

People in high tech take pride in their work. They are individuals who see the details of the things they produce in the light of the trials and triumphs they experience while creating products. In the courage of creation, they find a place to hang their individuality. Programmers and techno types appreciate elegant, spare code and the occasional well-turned architectural hack.

Rick Levine, Christopher Locke, Doc Searles and David Weinberger

The Cluetrain Manifesto

The Four Project Values

[Simplicity](#) + [Communication](#) + [Testing](#) = [Aggressiveness](#)

Simplicity

We strive always to "[do the simplest thing that could possibly work](#)". We're quite serious about this. We have found that with the right architecture and correctly-factored code, we can rapidly extend our software when the need arises. If we work on something "[we're gonna need](#)", we're not working on something we DO need.

Communication

We emphasize communication in every way. In our code, we all use the same [formatting conventions](#), naming conventions, and [coding standards](#). This means that all methods, no matter who wrote them, look familiar and are easy to understand when we come upon them.

We code so as to express intention, not algorithm. Our method names say what they accomplish, not how they accomplish it. At every level, we try to make our methods read like a human description of what is going on.

So that we can communicate design, we do it the same way, using CRC cards. Whenever we begin to work on some new objects, a few of us will sit down and do some cards. Whenever we make big changes, we'll get a larger group together and work out what is to be done. [But what about the formal documentation?](#)

Testing

We are fanatics for testing. We would like to have more lines of test than we do of actual code. Every class must have a corresponding unit test class. Each public method should have at least one unit test method. We presently have over 1400 [unit tests](#) in the system.

When we release code, all the unit tests must run at 100%. You can't release unless they do: if they don't, you fix the problem immediately.

We have [functional tests](#) as well, that test the system from end to end by paying one or more people and checking the results. There are hundreds of functional tests.

The result of all this testing is that we know the system works, and when we make errors we find them immediately.

Aggressiveness (Fearlessness)

The result of the other three values is that we can be aggressive, or fearless. We can change any part of the system to be better because we know we have a solid system of tests. We can try things and if we don't like how they work, we throw them away and try again. We know we won't break the system, which gives us the confidence to move forward rapidly.

Software Craftsmanship



*The New
Imperative*

Pete McBreen

Foreword by Dave Thomas



Craftsmanship has been used for centuries for the successful transmittal of skills and the development of communities of practice.

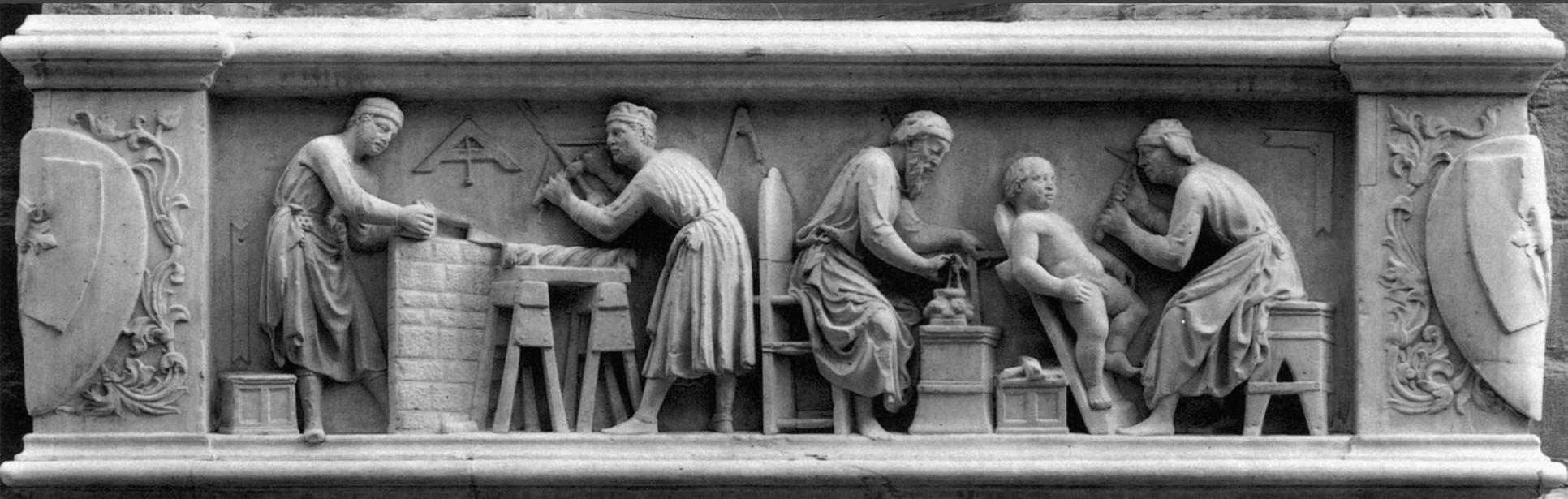
Pete McBreen, Software Craftsmanship

A guild is an association of craftsmen in a particular trade.

The earliest guilds were formed as confraternities of workers. They were organized in a manner something between a trade union, a cartel and a secret society. They often depended on grants of letters patent by an authority or monarch to enforce the flow of trade to their self-employed members, and to retain ownership of tools and the supply of materials. [...]

Two of the most outspoken critics of the guild system were Jean-Jacques Rousseau and Adam Smith, and all over Europe a tendency to oppose government control over trades in favour of laissez-faire free market systems was growing rapidly and making its way into the political and legal system. Karl Marx in his *Communist Manifesto* also criticized the guild system for its rigid gradation of social rank and the relation of oppressor/oppressed entailed by this system.

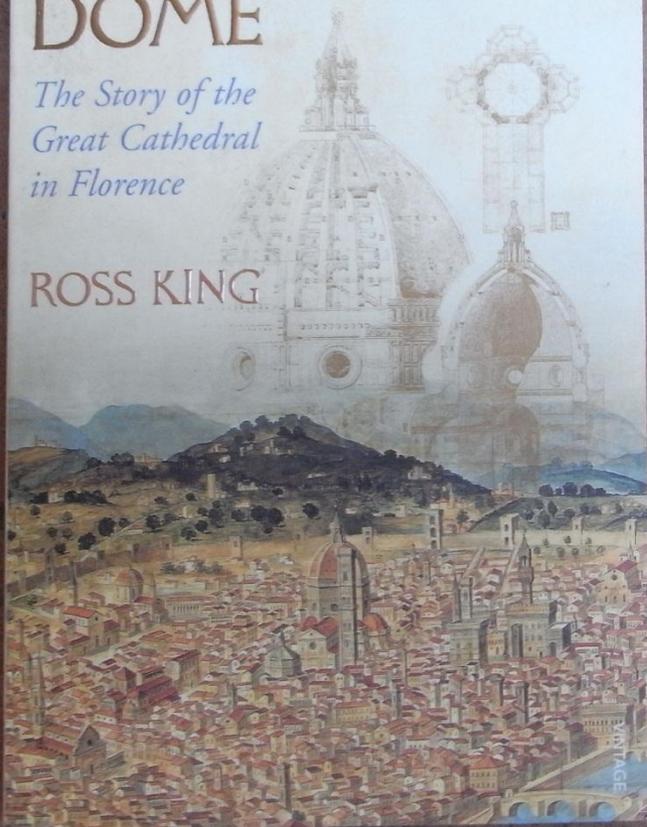
<http://en.wikipedia.org/wiki/Guild>



BRUNELLESCHI'S DOME

*The Story of the
Great Cathedral
in Florence*

ROSS KING



HERITAGE



Software development can only be considered immature because of how we use our experience, not because we lack experience.



9:50 AM Nov 1st, 2009 from TweetDeck

Delete



KevlinHenney
Kevlin Henney

We should be careful to get out of an experience only the wisdom that is in it — and stop there; lest we be like the cat that sits down on a hot stove lid. She will never sit on a hot stove lid again — and that is well; but also she will never sit down on a cold one anymore.

Mark Twain

Q: Why it is important to acknowledge and learn from your mistakes?

A: Acknowledging and learning from our mistakes will help to prevent making the same mistakes in the future and to improve productivity in the workplace.

http://wiki.answers.com/Q/Why_it_is_important_to_acknowledge_and_learn_from_your_mistakes

The assertion that we can learn something from every failure is often heard. This study by Earl Miller and his colleagues Mark Histed and Anitha Pasupathy of the Massachusetts Institute of Technology's Picower Institute for Learning and Memory tests that notion by looking at the learning process at the level of neurons. The study shows how brains learn more effectively from success than from failure. [...] Brain cells keep track of whether recent behaviours were successful or not. When a certain behaviour was successful, cells became more finely tuned to what the animal was learning. After a failure, there was little or no change in the brain – nor was there any improvement in behaviour.

<http://www.asfct.org/documents/journal/2009-11/Vol1-2-9.pdf>

It has become commonplace to suggest that failure is good for entrepreneurs. In this view, failure that comes early in a founder's career can teach them important lessons about doing business and harden them up for the next start-up attempt. [...]

In the UK, the evidence is that novices are neither more nor less likely to have a business that either grows or survives than experienced founders. In Germany, where much more extensive statistical work has been undertaken, it is clear that those whose business had failed had worse-performing businesses if they restarted than did novices. [...]

In short, the assumption that entrepreneurs use the lessons of their own experience to improve their chances of creating a series of profitable businesses is not borne out by the evidence. Success in business remains, as in life, something of a lottery.

David Storey, "Lessons that are wasted on entrepreneurs"

Anti-patterns don't provide a resolution of forces as patterns do, and they are dangerous as teaching tools: good pedagogy builds on positive examples that students can remember, rather than negative examples. Anti-patterns might be good diagnostic tools to understand system problems.

James O Coplien, Software Patterns

Wise men profit more from fools than fools from wise men; for the wise men shun the mistakes of fools, but fools do not imitate the successes of the wise.

Cato the Elder

A capsule definition of engineering, independent of any discipline, as you're likely to find: the set of practices and techniques that have been determined to work reliably through experience.

Glenn Vanderburg

<http://confreaks.net/videos/282-lsrc2010-real-software-engineering>

Structural engineering is the science and art of designing and making, with economy and elegance, buildings, bridges, frameworks, and other similar structures so that they can safely resist the forces to which they may be subjected.

The Institution of Structural Engineers

Software engineering is the science and art of designing and making, with economy and elegance, applications, bridges, frameworks, and other similar structures so that they can safely resist the forces to which they may be subjected.

With Economy and Elegance

Software Engineering Reclaimed

Kevlin Henney

kevin@curbralan.com

Understanding Engineering

- The mistake has not been in treating software development as a kind of engineering...
 - ◆ But in assuming that it was a form of physical engineering
 - ◆ But in assuming that *engineering* was synonymous with *plan driven*
 - ◆ But in assuming a caricature of other engineering disciplines to compare against and mimic
 - ◆ But in assuming that there was no art, craft or sense of aesthetics in engineering

Understanding Family

- Software engineering is a kind of informational engineering, not a kind of physical engineering
 - ◆ Therefore, software engineering is a cousin, not a sibling (and certainly not a clone), of the other frequently cited forms of engineering – cousins are still family, but they have less in common
- This means that much of what has passed for *software engineering* until now is pastiche
 - ◆ Choose the wrong map, and you will end up lost and heading in the wrong direction

Emerging Discipline

- Many known solutions and solution approaches form a stable base to build on
 - ◆ The classic topics of computer science
 - ◆ Using commoditised tools and libraries
 - ◆ Patterns for capturing and communicating software architecture and development experience
- Clearer thinking about how people work
 - ◆ Empirical and lean development processes
 - ◆ Focusing on interconnected systems, not just parts

Emerging Professionalism

- A clearer sense of responsibility can be found in many approaches advocated today
 - ◆ Programmers are responsible for the quality of their code, which includes testing, clean code, reasoned and reasonable choice of implementation, appropriate runtime characteristics, etc.
- It is easier to form developer communities that communicate norms, practices and experiences
 - ◆ From discussion groups to Open Source projects
 - ◆ From informal group meetings to conferences

Shared Understanding?

- The challenge is to make sure that what is already known and won is known and won
 - ◆ Software development is a diverse and fast moving field, and there is already a lot to know — and not everyone wants to know
 - ◆ Many previous attempts at capturing and communicating expertise have been too rigid and imposed in their approach
 - ◆ A mixture of advocacy, osmosis, openness, respect and leading by example is needed

TYPING IS NOT THE BOTTLENECK



Graphic by Sebastian Hermida
<http://sbastn.com/2009/06/typing-is-not-the-bottleneck/>



The newest computer can merely compound, at speed, the oldest problem in the relations between human beings, and in the end the communicator will be confronted with the old problem, of what to say and how to say it.

Edward R Murrow

**Programming is difficult
business. It should never
be undertaken in ignorance.**

Douglas Crockford
JavaScript: The Good Parts

Software craftsmanship's not the "next big thing". It's an attempt to articulate what the "thing" always was



10:22 AM Aug 6th via Power Twitter

Retweeted by 27 people

 Reply  Retweet



jasongorman