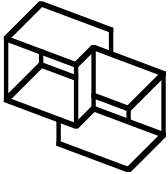


©2012 Software Craftsmanship Inc.

Applying Craftsmanship

What does 20 years of improvement look like?



Pete McBreen, Software Craftsmanship Inc.
pete@mcbreen.ab.ca

Software craftsmanship is necessary because most software is painful to use

With few exceptions, most software that people have to use as part of their job is abysmal

Corporate applications are in the dark ages and seem to be getting more baroque and unmaintainable

We need to find a better way to deliver working software that is a joy to use

Some progress has been made recently, but we still have a long way to go

©2012 Software Craftsmanship Inc. What does 20 years of improvement look like?

Page 2

The concept of Software Engineering is nearly 45 years old (1968 NATO conference)

It was born in an era of optimism

Man was going to walk on the moon

No matter what the problem there was going to be a *technological fix*

Science and engineering were transforming the world

It was only natural to try to apply engineering concepts to the problems of software

The originators even talked about *Software Manufacture*

©2012 Software Craftsmanship Inc. What does 20 years of improvement look like?

Page 3

From manufacturing it was a short step to software factories



But factories are not renowned as places where people actually think about what they are doing

Factories are places where unskilled laborers are supposed to do what they are told to do - i.e. follow the process

To date however this human wave approach to software development has not been successful

Cem Kaner has written about *Bad Software*

Cooper said *The Inmates are Running the Asylum*

Or as Casey warned we could *Set Phasers on Stun*

©2012 Software Craftsmanship Inc. What does 20 years of improvement look like?

Page 4

Standard software engineering practices seem to be designed to cause failure

Project managers do not understand technology

“Senior Developer” requires only 5 years experience

Project plans ignore variability

Waterfall is “ideal”

Testing comes last

People forget to think



©2012 Software Craftsmanship Inc. What does 20 years of improvement look like?

Page 5

Apologists insist that these problems are just because of faulty implementation

If reality still messes you up after 45 years then just maybe the original concept is wrong

We automate repeatable software development tasks because humans do not do repetition well

But what does the CMM aspire to? Repeatable, Defined and Optimizing processes!

How can we generate the requisite variety when optimizing the defined process has to go through the change control board?

The reality check has bounced!

©2012 Software Craftsmanship Inc. What does 20 years of improvement look like?

Page 6

Craftsmanship is necessary because software development is a high skill task

Talent and expertise are required to deliver software

Mechanical metaphors actively prevent us from delivering great software that users like to use

Craftsmanship is a way to connect passion with excellence

But enthusiasm needs to be guided by experience

Apprenticeship is a traditional way of gaining mastery in many complex fields

©2012 Software Craftsmanship Inc. What does 20 years of improvement look like?

Page 7

A key part of traditional craftsmanship is learning through an apprenticeship

A beginner progresses from Apprentice to Journeyman

Typically combines on the job experience with classroom training in specific techniques and skills

This works well for high skill trades that are relatively static

Unfortunately apprenticeship programs tend to limit innovation as it takes time to agree changes

Hard to adapt to new technology when it takes five years to agree to a curriculum change

©2012 Software Craftsmanship Inc. What does 20 years of improvement look like?

Page 8

We can still use the ideas from apprentice programs, we just have to be informal

The key idea is that trainees must contribute to real projects under close supervision

Not cheap labor, an investment in future employees

Best to start with maintenance - fixing the defects in existing software

Understanding the various failure modes is instructive, plus fixes get immediate feedback from users

Need tool changes to allow experienced developers to review the patches before they are committed

©2012 Software Craftsmanship Inc. What does 20 years of improvement look like?

Page 9

Too much software is developed without appropriate adult supervision

Alan Cooper hinted at a missing role in projects

Plenty of workers and managers, no *foremen*

Supervisors are a key part of most organizations

Hands on mentoring and coaching is nonexistent



©2012 Software Craftsmanship Inc. What does 20 years of improvement look like?

Page 10

We need to find a way to retain experience and expertise

We need to encourage corporations to create senior technical roles that remain active on projects

Skilled developers should not have to transition out of technical roles to advance their careers

Team leads need to take active responsibility for the quality of what their people produce

Team leads need to be very active in coaching their people to improve their skills in all areas

©2012 Software Craftsmanship Inc. What does 20 years of improvement look like?

Page 11

The sad fact is that many people on projects do not even know the basics of their day job

Small wonder then that projects are stressed

The Mythical Man-Month

Programmers at Work

Code Complete

The Pragmatic Programmer

Agile Software Development

Lessons learned in Software

Testing

About Face



©2012 Software Craftsmanship Inc. What does 20 years of improvement look like?

Page 12

Comp Sci and Software Engineering degrees do not seem to be effective

Not a popular sentiment, but how else do we explain how bad software is these days?

Part of the problem is that academia does not see that it has the responsibility to train software developers

Amusing as it may be, the Open Source community is a much better at training than most corporations

One key difference is that corporations do not have known individuals who are responsible for maintaining the code

©2012 Software Craftsmanship Inc. What does 20 years of improvement look like?

Page 13

So how can we as developers improve our skills?

The Pragmatic Programmers suggested that we all learn a new language every year

So in a sense we have Dave and Andy to thank for Rails

Reading books and code is also an effective strategy

But you have to read it with the intention of applying the new skills to a significant project

Working with others remains the best way to learn

©2012 Software Craftsmanship Inc. What does 20 years of improvement look like?

Page 14

Applying Craftsmanship requires an attitude change to consider *Software As Capital*

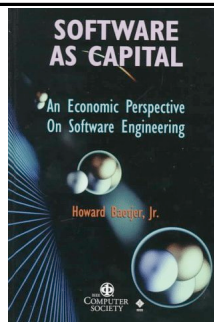
Too much effort goes into creation of software

Existing software gets little investment

Replacement not repair is the main focus

Broken software is good - we get to write more

Need to think of software as *embodied knowledge*



©2012 Software Craftsmanship Inc. What does 20 years of improvement look like?

Page 15

We need to think of software as embodied knowledge and act accordingly

Multi-generational teams are a good starting point

A team of twenty somethings has great energy, but old developers bring a breadth of experience to the task

Handoff to a maintenance team never works well

The team that created the application need to stay together to maintain and create future releases

Documentation is useful, but it has to be written by the experts, for the experts

Anything else drowns the team in paperwork

©2012 Software Craftsmanship Inc. What does 20 years of improvement look like?

Page 16

Too many companies are looking for instant answers to problems

Consultancies promote the quick fix idea

Limiting the opportunity for innovative thinking

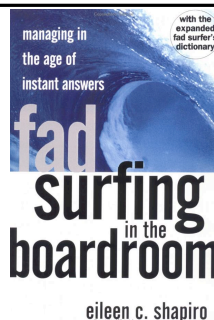
Often instant answers turn out to be a mistake

CASE tools

Outsourcing

Extreme Programming

Requirements traceability



©2012 Software Craftsmanship Inc. What does 20 years of improvement look like?

Page 17

Companies need to create senior technical roles with the authority to make decisions

Managers without technical skills should be transitioned out of IT roles

Uninformed managers are easy prey for sales promises

Many places have a high turnover of technical staff

Technically competent line managers provide visible evidence that development is a possible career path

©2012 Software Craftsmanship Inc. What does 20 years of improvement look like?

Page 18

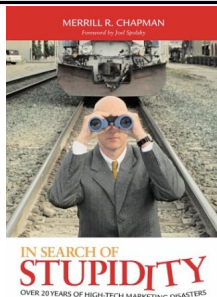
Most projects are badly mismanaged in corporations

Few projects allow time to think about the issues

Small wonder that many systems are half-baked

Project requirements are rarely questioned

Many projects are good responses to the wrong problem



©2012 Software Craftsmanship Inc. What does 20 years of improvement look like?

Page 19

We need project managers to report to the project lead

The person who is responsible for the success of the project needs to lead the project

Admin staff can handle tracking the project schedule

The lead needs to focus on the big picture items

Is the entire team working well?

Are we solving the right problems?

Do all our stakeholders understand what is happening?

Is the quality of the design appropriate?

Are the right contracts and budgets in place?

©2012 Software Craftsmanship Inc. What does 20 years of improvement look like?

Page 20

Software development can still learn lots from engineering design practices

Tom Gilb popularized incremental development in mid 1980's

Target was a release after every 2% of budget

In doing so we learned how to evolve designs

Real engineers are always improving their designs



©2012 Software Craftsmanship Inc. What does 20 years of improvement look like?

Page 21

Long lived software that has been improved over the years is typically quite good

Complete rewrites are necessary sometimes

But most times it is better to evolve the existing code

Gradual evolution over many releases works well

Continual improvement is what matters to users



©2012 Software Craftsmanship Inc. What does 20 years of improvement look like?

Page 22

Quality will improve as we recognize the need for individual accountability

Signing our work is a good first step

Hunt and Thomas raised this as an issue in 2000

The Agile Alliance recognized this as well

People over Process

Talent and expertise make a big difference



©2012 Software Craftsmanship Inc. What does 20 years of improvement look like?

Page 23

Longevity is not enough, developers need a design aesthetic as well

It is not just about code

Look and feel matter, users need pleasure

Improve usability as well as functionality

Both should evolve together over time



©2012 Software Craftsmanship Inc. What does 20 years of improvement look like?

Page 24

Improvement comes from spending a lot of time with your users

Many ideas that look neat fail in the real world

We learn when the rubber meets the road

Initial failure is OK as long as we learn from it

Need longevity in order to learn

Better decisions are made by practitioners



©2012 Software Craftmanship Inc. What does 20 years of improvement look like?

Page 25

Working with users is a central part of applying craftsmanship

The old days of expecting users to be thankful for whatever we give them are long gone

We need to learn how to fully engage our users in the development process

Even in a corporate context we need to think about delighting our users

This requires we learn how to *create what the user wants* rather than *build what we know how to build*

©2012 Software Craftmanship Inc. What does 20 years of improvement look like?

Page 26

Prediction is very difficult – especially if it is about the future *Niels Bohr*

Having deep domain knowledge means you have to ask your users

You are not your user

Especially when you know what your user needs

Remember that craftsmanship comes about through humility

We can always learn more from our users



©2012 Software Craftmanship Inc. What does 20 years of improvement look like?

Page 27

Software is meant to be pleasant to use, if it isn't we are doing something wrong

You need to work in QA to know how bad software is

There are some bright spots but users have a raw deal

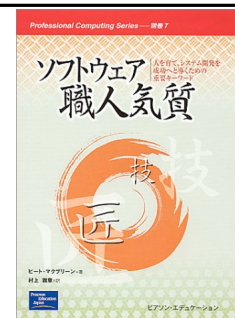
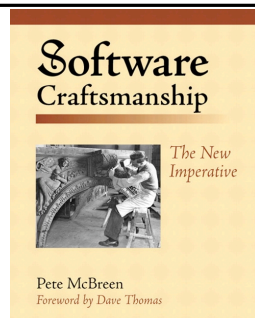
We need to commit to improving our users day job



©2012 Software Craftmanship Inc. What does 20 years of improvement look like?

Page 28

Software development is meant to be fun, if it isn't the process is wrong



©2012 Software Craftmanship Inc. What does 20 years of improvement look like?

Page 29