Software Craftsmanship and Agile Code Games

Mike Clement
@mdclement
mike@softwareontheside.com
http://blog.softwareontheside.com
http://agilecodegames.com
“Often the true value of a thing isn’t the thing itself, but instead is the activity that created it.”

-Dave Thomas
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.
Principles of Agile Software Development

- Our highest priority is to **satisfy the customer** through early and continuous delivery of valuable software.
- Welcome **changing requirements**, even late in development.
- **Deliver** working software **frequently**.
- Business people and developers must **work together** daily throughout the project.
- Build projects around **motivated individuals**.
- The most efficient and effective method of conveying information is **face-to-face conversation**.
- **Working software** is the primary measure of progress.
- Agile processes promote **sustainable development**.
- Continuous attention to **technical excellence** and good design enhances agility.
- **Simplicity**--the art of maximizing the amount of work not done--is essential.
- The best architectures, requirements, and designs **emerge from self-organizing teams**.
- At regular intervals, the **team reflects** on how to become more effective, then adjusts accordingly.
Principles of Agile Software Development

- satisfy the customer
- changing requirements
- deliver frequently
- work together
- motivated individuals
- face-to-face conversation
- working software
- sustainable development
- technical excellence
- simplicity
- emerge from self-organizing teams
- team reflects
Principles of Agile Software Development

- satisfy the customer
- changing requirements
- **deliver frequently**
- work together
- motivated individuals
- face-to-face conversation
- **working software**
- sustainable development
- technical excellence
- simplicity
- emerge from self-organizing teams
- team reflects
Continuous attention to technical excellence and good design enhances agility.
Literature Roots (1999 and 2001)
Clean Code – August 2008
Manifesto for Software Craftsmanship

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.
Apprenticeship Patterns – October 2009
The Clean Coder – May 2011
Software Craftsmanship – 2013/2014
Quality Code – November 2013
Software Engineering?

Engineering is the application of scientific, economic, social, and practical knowledge in order to design, build, maintain, and improve structures, machines, devices, systems, materials and processes.

• Civil Engineers – do they drive rivets?
• Automotive Engineers – do they assemble cars?
• Computer Engineers – do they fabricate processors?
• Software Engineers – do they write code?
Craftsmanship: What’s different?

**Craftsmanship**

What's different?

Best for the business (even when you don't know it)

- Educate
  - Short views
  - Long views

Only way to go fast is to go well

- Boyscout Rule
- Hide sell (or get rid of it)

Better to be lucky than good... but if you aren't lucky you better be good

While the road going is rough, it's good enough to counter against
"Going slow today to go fast tomorrow"
“Going deliberately today to not be stuck tomorrow” – Me
“The only way to go fast, is to go well” – Uncle Bob
“Optimize for read time, not write time” – Me
Things that Software Craftsmen Value

• Growth mindset
• Adapting and changing
• Share over hoarding
• Safety to experiment
• Taking control of own destiny
• Inclusiveness
• Skill-centric over process-centric
• Situated Learning
Things that Software Craftsmen Value

• Growth mindset
• Adapting and changing
• Share over hoarding
• Safety to experiment
• Taking control of own destiny
• Inclusiveness
• Skill-centric over process-centric
• Situated learning
Situated learning is learning that takes place in the same context in which it is applied.
Getting Started in F#

Congratulations! You’ve decided to take your first step towards learning F#, a powerful language that enables you to write simple, concise code to solve complicated problems. In this course, you’ll learn the basics that you need to feel comfortable reading and writing F# programs.

Beginning F#
Getting Acquainted

Welcome to Try F# 3.0
For additional help: http://www.tryfsharp.org/Help
// About the Order of Evaluation
//
// Sometimes you'll need to be explicit about the order in which functions are evaluated. F# offers a couple mechanisms for doing this.

[<Koan(Sort = 4)>]
module "about the order of evaluation" =

[<Koan>]
let SometimesYouNeedParenthesisToGroupThings() =
  let add x y =
    x + y

  let result = add <| add 5 8 <| add 1 1

  AssertEquality result 15

  (* TRY IT: What happens if you remove the parenthesis?*)

[<Koan>]
let TheBackwardPipeOperatorCanAlsoHelpWithGrouping() =
Agile Code Games

How to learn without really trying
“Often the true value of a thing isn’t the thing itself, but instead is the activity that created it.”

-Dave Thomas
Code Katas

The scales and etudes of developing software
Part of a Guided Kata

FizzBuzz
Solo Randori

Coding up a problem, probably repeatedly (sometimes referred to as a kata)
Problem Description
Once upon a time there was a series of 5 books about a very English hero called Harry. (At least when this Kata was invented, there were only 5. Since then they have multiplied) Children all over the world thought he was fantastic, and, of course, so did the publisher. So in a gesture of immense generosity to mankind, (and to increase sales) they set up the following pricing model to take advantage of Harry's magical powers.

One copy of any of the five books costs 8 EUR. If, however, you buy two different books from the series, you get a 5% discount on those two books. If you buy 3 different books, you get a 10% discount. With 4 different books, you get a 20% discount. If you go the whole hog, and buy all 5, you get a huge 25% discount.

Note that if you buy, say, four books, of which 3 are different titles, you get a 10% discount on the 3 that form part of a set, but the fourth book still costs 8 EUR.

Potter mania is sweeping the country and parents of teenagers everywhere are queueing up with shopping baskets overflowing with Potter books. Your mission is to write a piece of code to calculate the price of any conceivable shopping basket, giving as big a discount as possible.

For example, how much does this basket of books cost?
2 copies of the first book
2 copies of the second book
2 copies of the third book
1 copy of the fourth book
1 copy of the fifth book
(answer: 51.20 EUR)

http://www.codingdojo.org/cgi-bin/index.pl?action=browse&id=KataPotter&revision=41
Wasa

Coding up a problem, with a pair
Group Randori

Ready... fight!
Quick Concepts

**TDD**
- Red
- Green
- Refactor

**Simple Design**
- Passes all tests
- Clear, expressive, consistent
- No duplication
- Minimal
Ways to get Green

• Fake it
• Obvious implementation
• Triangulation
Rules

• Pair Programming
• Use TDD
• Everyone should be following
• Pair should be explaining
• Audience gives suggestions only with when Green
• Today: Will switch pairs about every 2 minutes
Numbers to LCD

123456789
Translates to:

```
// _ _ _ _ _ _ _ _
// || || || || || ||
// || || || || || ||
```
Problem/Kata Lists

- http://codingdojo.org/
- http://codekata.com/
- http://katas.softwarecraftsmanship.org/
Koans
The Path to Enlightenment
Example Koans

- F# [https://github.com/ChrisMarinos/FSharpKoans](https://github.com/ChrisMarinos/FSharpKoans)
- Javascript [https://github.com/liammclennan/JavaScript-Koans](https://github.com/liammclennan/JavaScript-Koans)
- Javascript [https://github.com/mrdavidlaing/javascript-koans](https://github.com/mrdavidlaing/javascript-koans)
Coderetreat
A day like no other
Structure of Coderetreat

• Duration: 8:30am to 5 or 6pm (provide breakfast and lunch, not pizza)
• Problem: Conway's Game of Life
• Length of Session: 45 minutes (5 or 6 sessions usually)
• Pair-programming is necessary, as the knowledge transfer contained in that activity is essential to the practice
• Prefer using Test-Driven Development (TDD)
• After each session, pairs should be swapped
• After each session, code must be deleted, not put in a branch, not stashed, just deleted with no trace left
Activities/Constraints

• Explore the problem
• Ping pong – intro to pairing techniques
• No mouse
• Paper only
Activities/Constraints

• No naked primitives
• No conditionals
• No loops
• Only 2, 3, 4 lines per method
• Immutable
• Mute
• Evil pair/find the loophole
• Mute with find the loophole
• Tell, don’t ask
• TDD as if you meant it
Experience + Reflection = Learning
Closing Circle

• What, if anything, did you learn today?
• What, if anything, surprised you today?
• What, if anything, will you do differently in the future?
Mob Programming

Like Randori, but larger switching cycles, and a real team working on a real system throughout the day
One navigator, many pairs
One navigator, many mobs
Greater Sum Workshops

• 2 hours
• Opening Circle
  • Name
  • Programming Experience
  • Silly question (What’s your favorite... ?)
• Mobbing!
  • Possibly “One navigator, many mobs”
• Closing Circle
  • Borrowed from Coderetreat
Software Craftsmanship Atlanta and Utah Software Craftsmanship Format

Our meeting format differs from your usual technology user group. At a meeting we have:

• 0-3 Lightning talks (up to but not over 5 minutes each)
• ~30 min - Reading Discussion
• ~60 min - Interactive Craftsman-run Coding Exercise (based on the principles of a Coding Dojo)
“Often the true value of a *thing* isn’t the thing itself, but instead is the activity that created it.”

-Dave Thomas
Books

• The Pragmatic Programmer: From Journeyman to Master
• Clean Code
• Apprenticeship Patterns (free online)
• The Clean Coder
• Software Craftsmanship: Professionalism, Pragmatism, Pride
• Quality Code: Software Testing Principles, Practices, and Patterns
• The Coding Dojo Handbook
Resources

• http://agilemanifesto.org/
• http://manifesto.softwarecraftsmanship.org/
• http://coderetreat.org/
• http://mobprogramming.org/
• http://www.agilecodegames.com/
Mike Clement

- @mdclement
- mike@softwareontheside.com
- http://blog.softwareontheside.com
- http://agilecodegames.com
- https://github.com/mdclement
- http://www.greatersum.com
- Software Craftsmanship Atlanta – first Wed at 6pm