Coderetreats: Reflective Practice and the Game of Life

Associate Professor David Parsons

Massey University, Auckland

Research seminar presentation, University of Canterbury, Christchurch, May 2014

Coderetreats

- **A day spent with other software developers, addressing a single problem over and over again with different partners and with different design constraints**

My Research

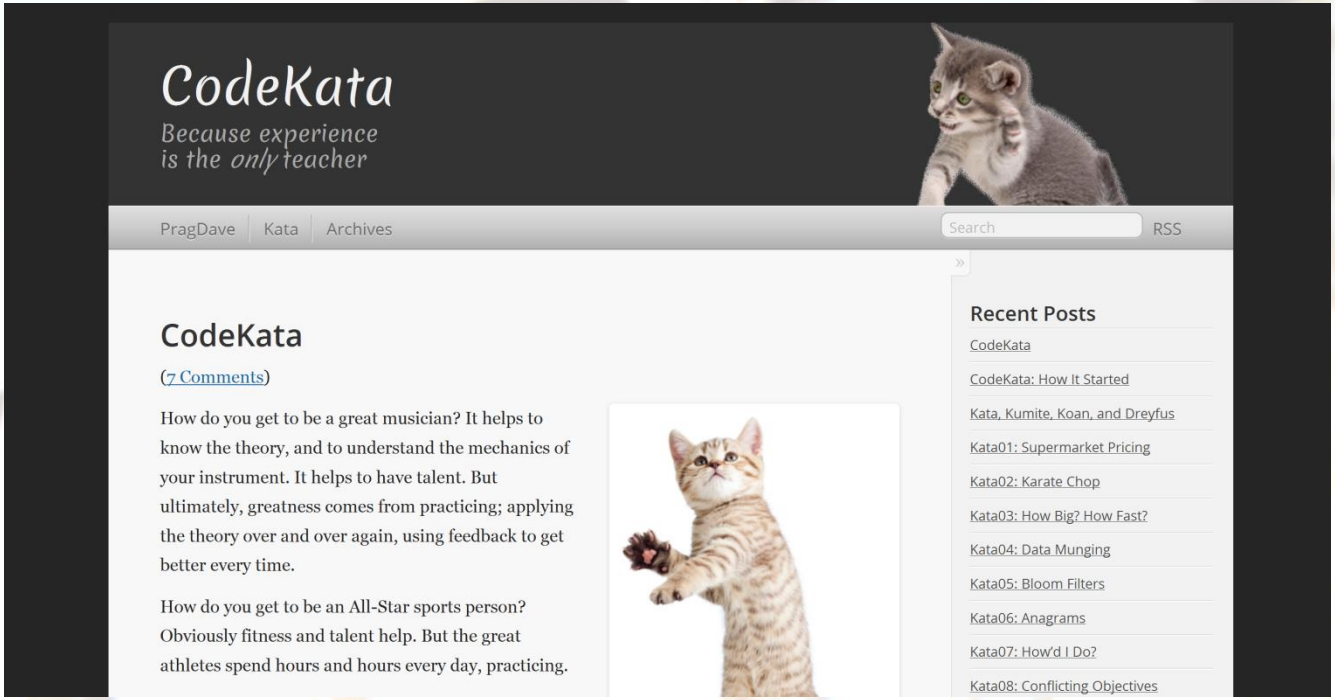
- **Run coderetreats**
- **Gather data:**
  - **Code**
  - **Survey responses**
- **Analyse:**
  - **Evidence for reflective practice**
  - **Evidence for skills development**
  - **Ideas for change**

Reflective Practice

- **How can software engineers improve their skills and techniques**
- **How do they rehearse and reflect on their craft?**
- **What are the ‘scales and studies’ of software?**

Code Katas

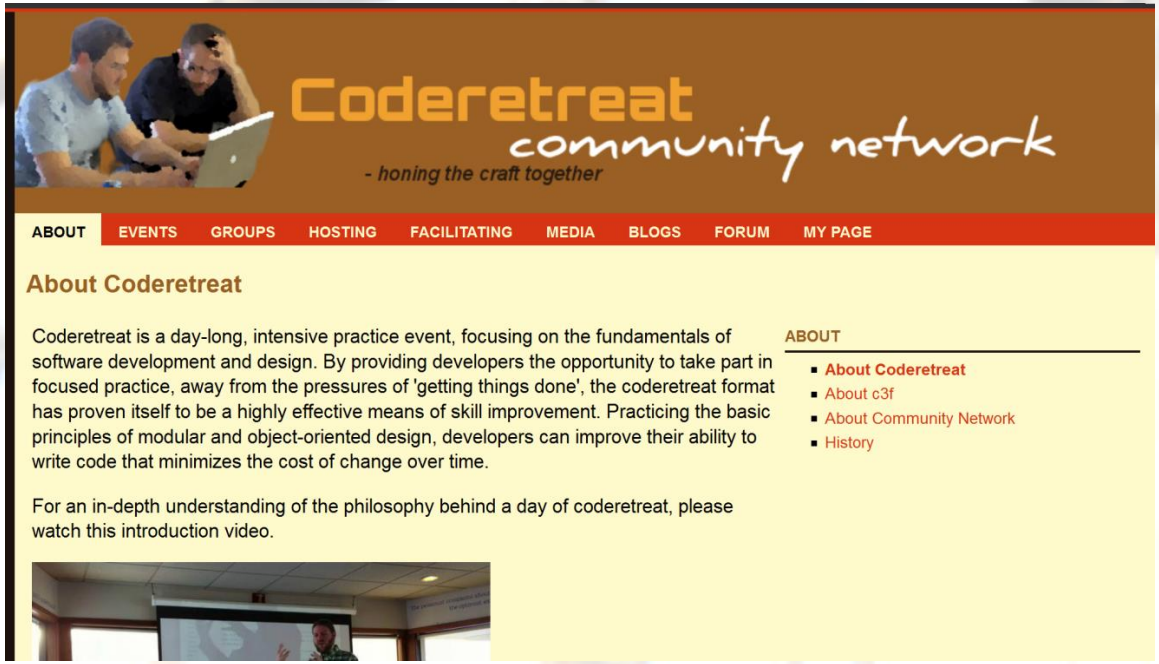
- **Dave Thomas (and a Kat?)**



http://codekata.com/

Coderetreat Web Site

- **Gary Bernhardt, Patrick Welsh, Nayan Hajratwala and Corey Haines**



http://coderetreat.org/

Design Fundamentals

- **Coderetreats are about '*Practicing the basic principles of modular and object-oriented design*'**
  - **Tests drive the code (TDD)**
  - **Duplication is removed (refactoring)**
  - **All the requirements are expressed (cohesion, self-documenting code)**
  - **Code contains no unnecessary features (YAGNI)**

Based on the XP simplicity rules:
http://c2.com/cgi/wiki?XpSimplicityRules

Coderetreat Structure

- **The kata is Conway's Game of Life**
- **Five or six 45 minute coding sessions**
- **Pair-programming and TDD**
- **After each session, code is deleted and partners swapped**
- **No expectation that a complete solution will be written in any session**
- **Different constraints can be applied**

What Do They Offer?

- **Changing problem settings**
- **Cycles of experience and reflection in action**
- **Reflective conversions with, and learning from, others**
- **Deliberate practice, not just doing what we are already good at**
- **Innovation and exploration**

Game of Life Simulation

- **Cells exist in a conceptually infinite two-dimensional grid**
- **A cell is something that has a binary state**
  - **life or death**

Initial States and Generations

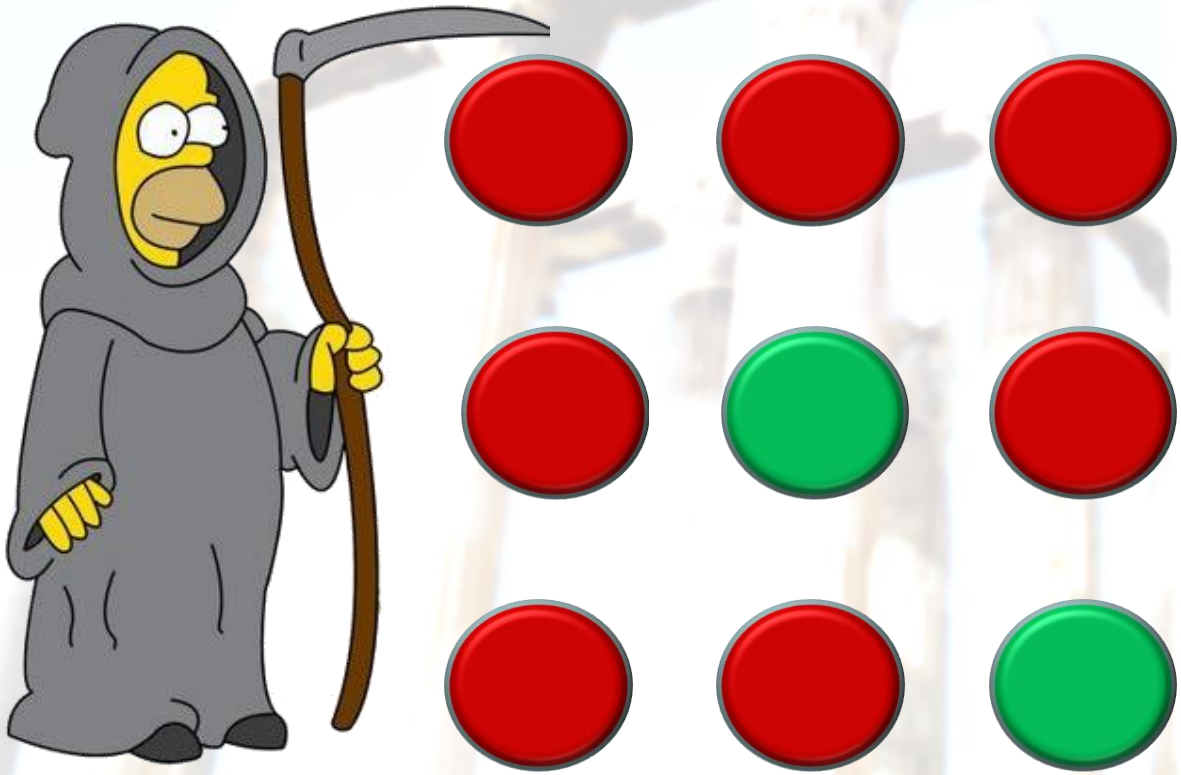
- **At the beginning of the simulation, each cell must be initialized as either alive or dead**
- **In each subsequent generation, it will either:**
  - **Remain in its current state**
    - **or**
  - **Transition to its other possible state**

State Transitions

- **State transitions depend on the states of eight neighbouring cells**
- **There are four rules to these transitions**
  1. **Underpopulation**
  2. **Next generation**
  3. **Overcrowding**
  4. **Reproduction**

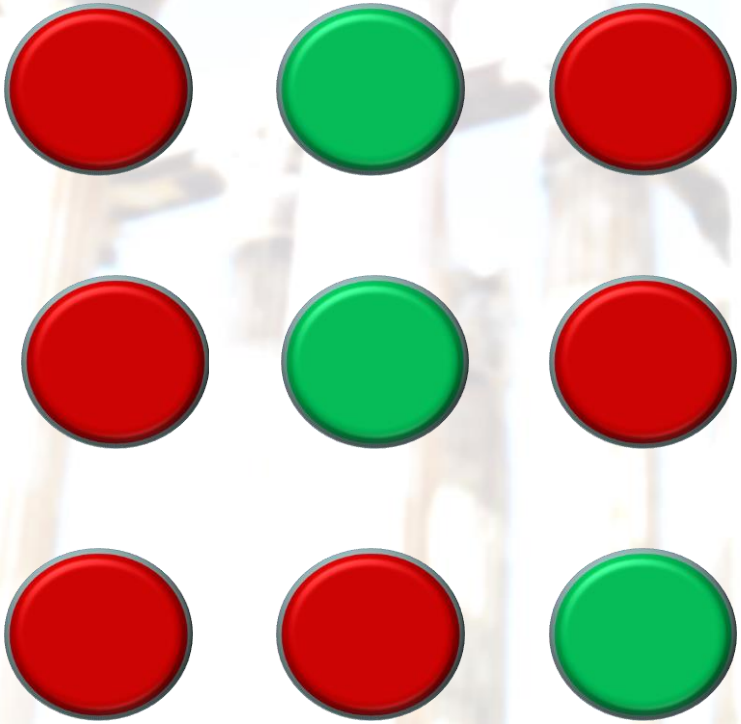
Underpopulation

- **A live cell with fewer than two live neighbours dies**



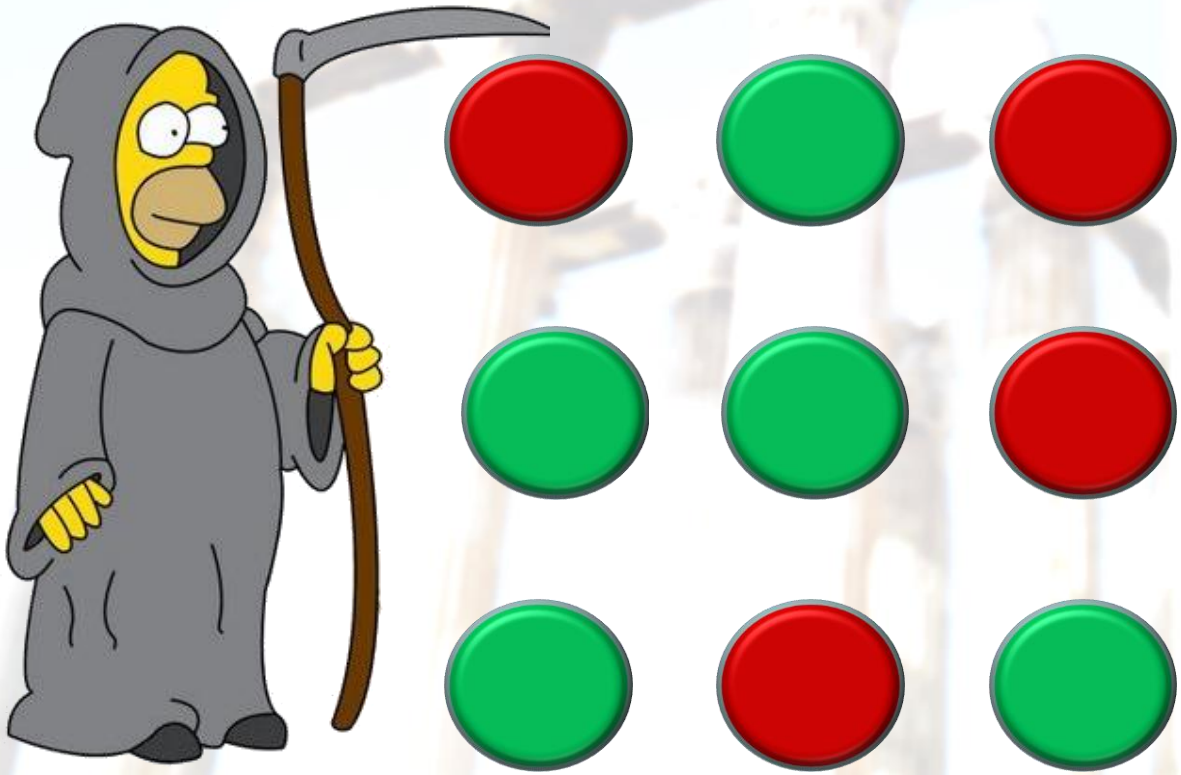
Next Generation

- **A live cell with two or three live neighbours lives on**



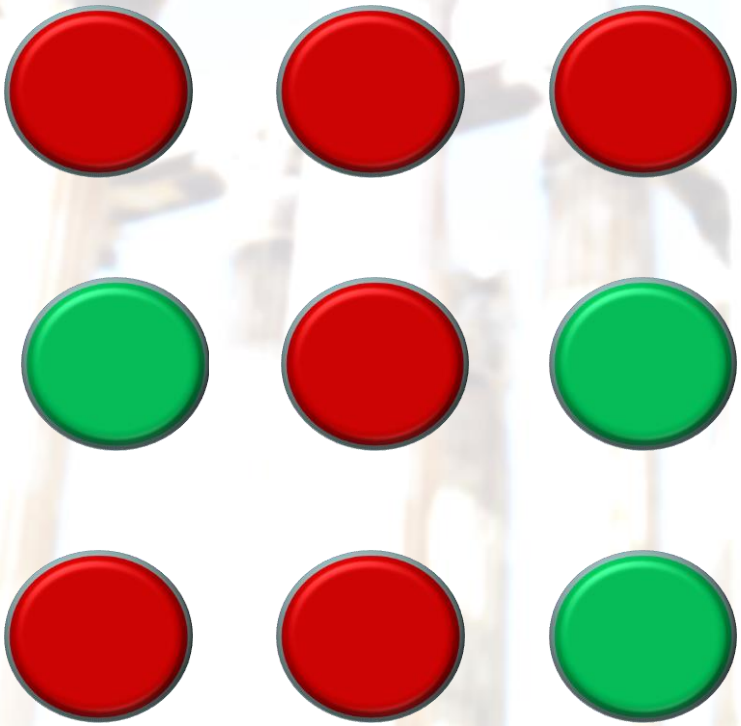
Overcrowding

- **A live cell with more than three live neighbours dies**



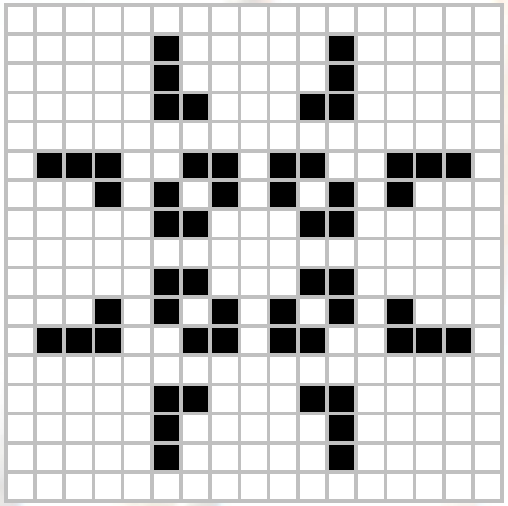
Reproduction

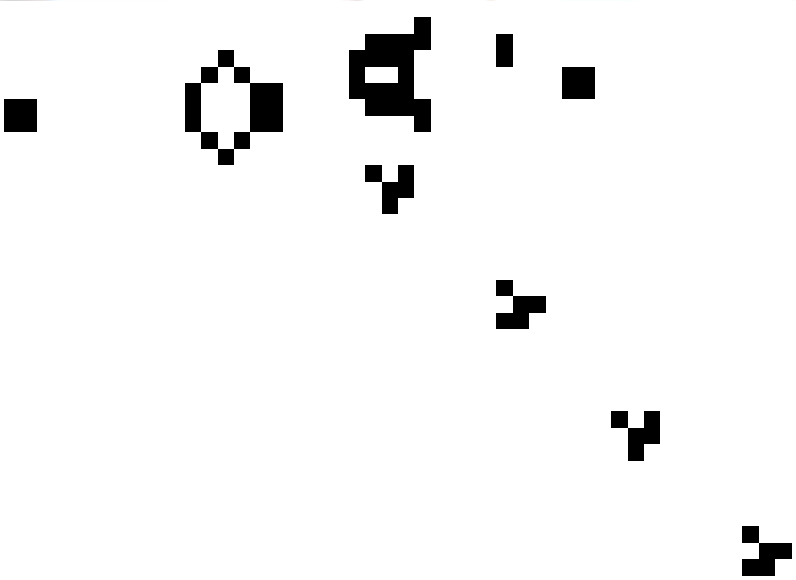
- **A dead cell with exactly three live neighbours becomes alive**

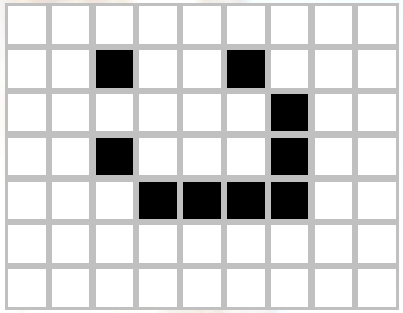


Conway's Game of Life

- **Some examples of generational patterns (from Wikipedia)**







Our Study

- **Designed to:**
  - **Explore how a coderetreat supports reflective practice**
  - **Gather empirical data on how code creation evolves in a coderetreat**
  - **Gather survey data on programmer practice and experience in a coderetreat**

Experimental Setting

- **36 final-year undergraduates taking a course in software architecture**
- **Participated in a coderetreat as part of their lab practice**
- **Coding in Visual Studio (C# or VB.NET)**
- **Standard 5 session coderetreat structure**
- **Common constraints applied in sessions 2-5**

Session Constraints

1. **No constraints**
2. **‘ping-pong’**
   - **roles are swapped within pairs**
3. **‘mute’**
   - **silent pairing**
4. **‘no arrays’**
   - **forces a new approach to existing design**
5. **‘new requirement (track generations)’**
   - **requires new design features**

Measuring Reflective Practice

- **From previous work:**
  - **Instances of events**
  - **Time spent on activities**
  - **Use of resources**
  - **Direct feedback (forum access and posts)**
  - **Relative time spent on activities**
  - **Recording with software tools**

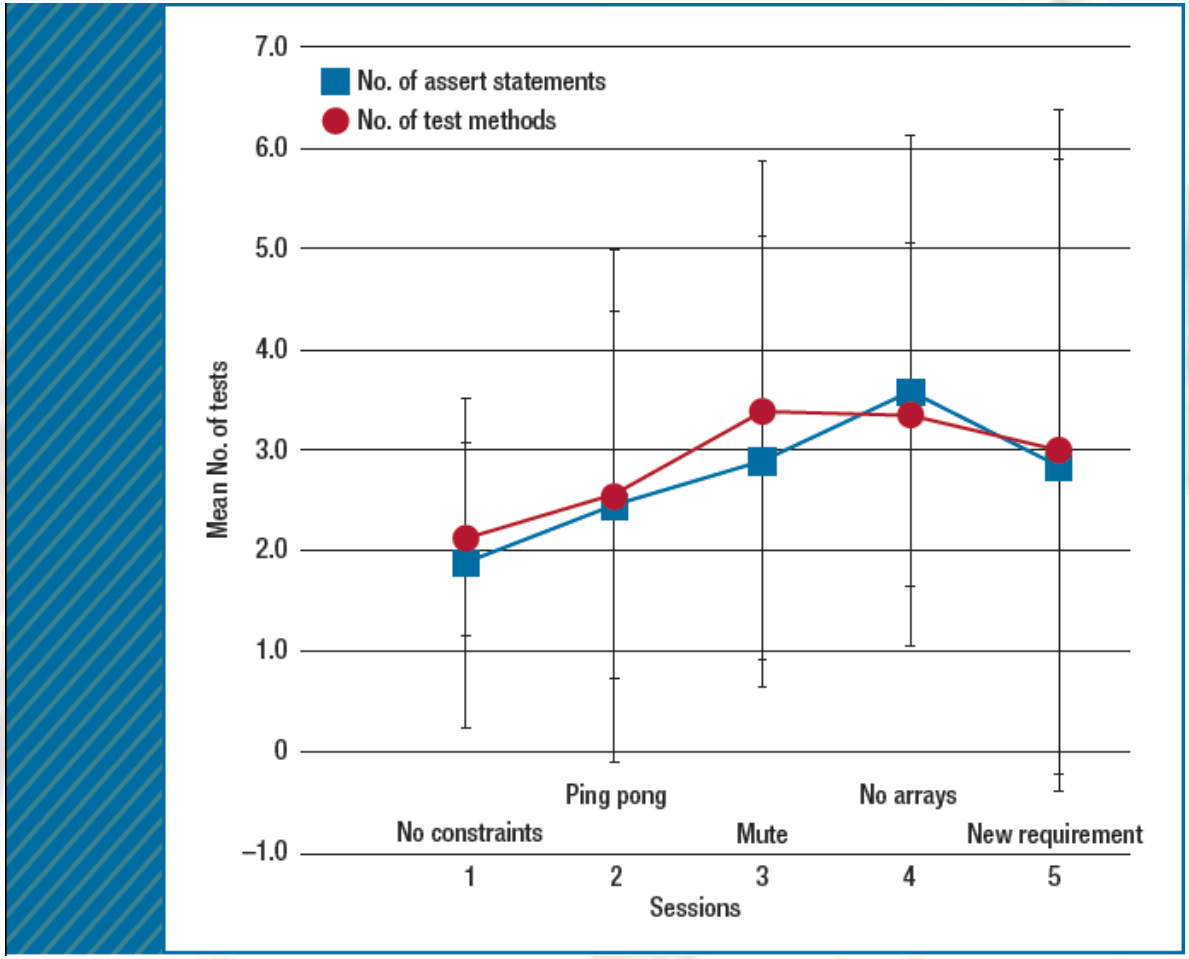
Metrics and Reflections

- **In our study we decided to:**
  - **Store and analyse code (software tools)**
  - **Count tests and assertions (instances of events)**
  - **Measure test quality (use of resources)**
  - **Compare test frequency (relative time)**
  - **Surveys (direct feedback)**

Code Analysis

- **Quantitative analysis**
  - **Test case code analysis over the five coding sessions**
  - **Average number of test methods**
  - **Average number of significant assertions**
- **Qualitative analysis**
  - **Compared test case code from first and final sessions**
  - **Used the Arrange, Act, Assert (3A) pattern as the quality benchmark**

Quantitative Results



Qualitative Results

- **Significant improvement in test quality**

First Session	Last Session
Only half the pairs wrote unit tests that had any utility	All but two of the pairs produced 3A-compliant test suites
Of those, only half wrote ‘quality’ (3A compliant) tests	Of those, two-thirds complied with best practices
About a third of all pairs had only stubs or meaningless test code	Around one-fifth still resorted to multiple assertions per unit test
Some could not even create a test project	Only a few included some inconclusive tests

Session Surveys

- ***What was the first thing that you tested in this session?***
- ***Why did you choose this particular test?***
- **We asked the same questions after each of the 5 sessions**

Session Survey Results

- **Test Driven Design adapts to session constraints**

TABLE 1

The "first test" created by pairs in each session.*

Session	Constraint	Constructor	Cell life	Counting neighbors	Game rules	Data structure*	Cell age*
1	None	5	5	3	0	-	-
2	Ping-pong	2	5	1	1	-	-
3	Mute	3	5	1	2	-	-
4	No arrays	0	0	0	0	3	-
5	Changing requirements (generation count)	3	3	0	0	0	6

** A dash in this column indicates that this test wasn't applicable to a specific session constraint.*

Final Survey

- **Designed to find out about self-reflection in the coderetreat process**
- **Some quantitative structured questions (e.g. Likert scale)**
- **Some qualitative free text responses**

Standard Questions

- **Asked at the end of all coderetreats**
  - **What, if anything, did you learn today?**
  - **What, if anything, surprised you today?**
  - **What, if anything, will you do differently in the future?**

What, if anything, did you learn today?

- ***'More about the TDD process, when to write tests, how to problem solve - especially under surprise constraints'***
- ***'I learnt a better understanding of starting off programming. I struggle with finding where to begin'***
- ***'How to write tests at all stages of code'***

What, if anything, surprised you today?

- ***'You asked us to write the program without arrays :('***
- ***'The no talking challenge was a surprise. I was expecting coding difficulties rather than communication difficulties'***
- ***'How many different ways you can do the same thing'***
- ***'Finally understanding something this semester'***

What, if anything, will you do differently in the future?

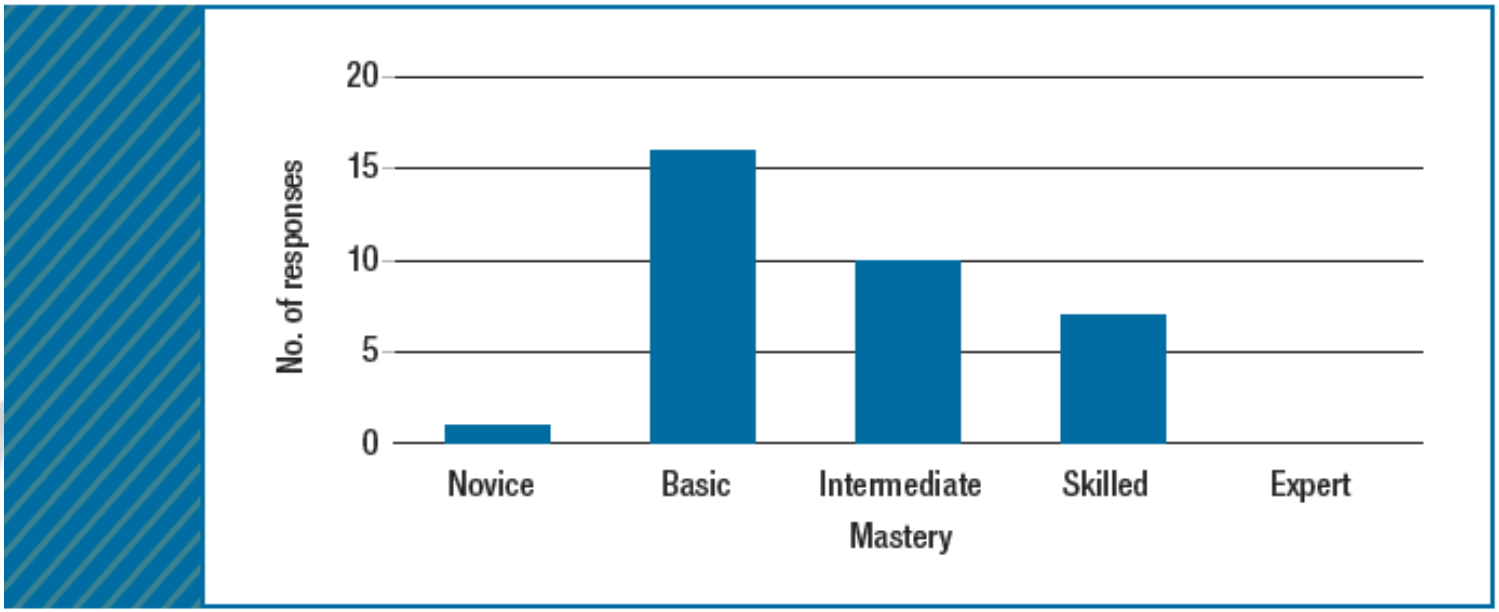
- ***'Study more before the retreat'***
- ***'Test when there is only skeletal/low amounts of code'***
- ***'Look more closely to the problem and modularise'***
- ***'I would sleep more the night before – truth'***

Zimmerman's Criteria

- **Questions measuring self-judgment and self-reaction**
  - **Mastery**
  - **Previous performance**
  - **Normative criteria**
  - **Collaborative criteria**
  - **Alignment of poor results with processes that can be controlled**

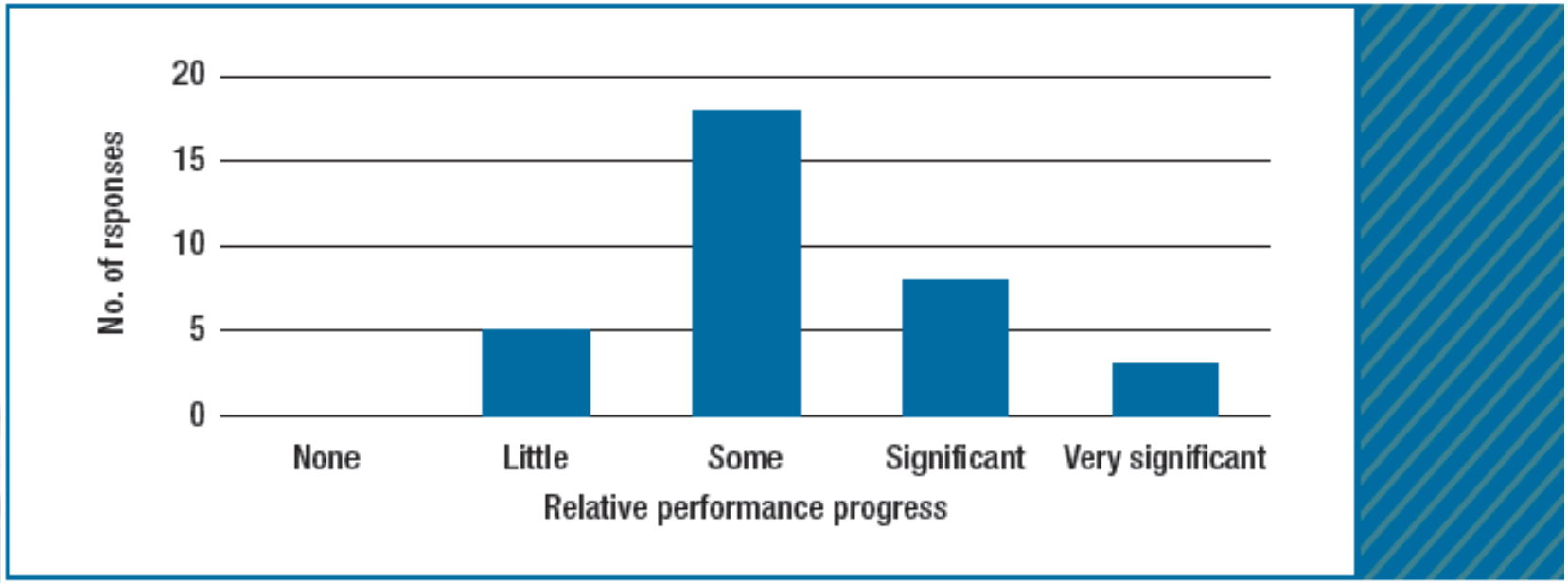
Mastery

- **Only one still felt they were a novice**



Previous Performance

- **All made progress, even the novice**



Normative and Collaborative

- **A realistic assessment of performance compared with peers**
- **All were comfortable in pair programming**
- **Only six more comfortable in the role of driver**
  - **May reflect lack of previous experience**

Ordering Causes of Difficulty

- **Aligning processes and results**
- **Lack of skills and experience did not appear at the top of the list**
- **Working with others appeared at the bottom**

Order	Difficulty…
1	In using the test framework
2	In using the programming language
3	Due to participant's lack of skills and/or experience
4	Caused by the various constraints imposed in the sessions (for example, not talking)
5	In understanding the Game of Life problem
6	In working with others

Skills Development

- **The core skill in a coderetreat is writing tests**
- **The students' confidence in writing tests had improved from an average of 4.66 ($\sigma = 2.12$) to 5.64 ($\sigma = 1.92$), on a scale of 0 to 10**

Other Comments

- ***'It's a good activity as it prepares the students'***
- ***'I was pleasantly surprised by how useful it was'***
- ***'It was really helpful and I enjoyed it a lot'***
- ***'The more we did it the easier it got and getting different ideas was good'***

Conclusion

- **The coderetreat revealed positive outcomes:**
  - **Evidence of self-reflection**
  - **Collaborative effort**
  - **Improved conceptual understanding**
  - **Progress toward mastery**

Next Steps

- **Assessing professional practitioners' coderetreats**
- **Proposing new ways to enhance the value of coderetreats**

Global Day of Coderetreat 2013

GDCR Surveys

Developed with Jim Hurne (IBM) global coordinator

Responses

- **Final survey: 443**
  - **On a sample of about 2,200, enough for statistical significance with a 5% margin of error and 95% confidence**
- **Session surveys**
  - **From 118 (session 1) down to 25 (session 5)**
  - **Can be qualitatively coded**
    - **Currently coding in NVivo**

Work in Progress

- **Demographics**
  - **Representative**
- **Mastery**
  - **Improves with experience, up to a point…**
- **Progress**
  - **Less for novices**
- **Difficulty**
  - **Constraints for most, but languages for the most experienced**

Possible Futures

- **Change first session to provide structure**
- **Directly address the ‘simple design’ concepts**
- **Explore legacy coderetreats**

Reference

- **Parsons, D., Mathrani, A., Susnjak, T. & Leist, A. (2014). Coderetreats: Reflective Practice and the Game of Life, *IEEE Software, 31*(4), 58-64.**

- **First academic publication on coderetreats**

Coderetreats: Reflective Practice and the Game of Life

David Parsons, Anuradha Mathrani, Teo Susnjak, and Arno Leist, Massey University

// *Coderetreats—events where software developers spend a day in an informal yet intellectually challenging environment to practice their craft—encourage reflective practice by addressing a single programming problem with multiple coding partners and design constraints. An experiment with a group of final-year undergraduates studying software architecture reveals that coderetreats provide a context within which multiple aspects of self-reflection and motivation can be developed.* //

SOFTWARE PRACTITIONERS WORK in an industry that's constantly innovative and challenging, where the need to learn continuously and reflect on practice is an essential prerequisite to the creation of high-quality software. However, in the day-to-day activities of their profession, the freedom to explore new aspects of theory and practice can be restricted by the demands of producing software artifacts for clients. Developers need to have some way of honing their craft in practice sessions that are free from interruptions and pressures, where mistakes can be made safely and the same task can be attempted multiple times to gain feedback and learn how to improve the solution.

One such approach is the coderetreat, as described on the coderetreat.org website. Coderetreats are based on the use of a code kata, taken from the kata concept in martial arts, where the same action is performed repeatedly in an effort to improve. A code kata, then, is a simple programming problem that can be solved in many different ways, helping the developer weigh different design and implementation options (see http://codekata.com). Coderetreats build on this idea further by integrating techniques such as pair programming and test-driven development (TDD) within a particular set of practices and constraints.

A coderetreat follows the apprenticeship pattern of a "breakable toy,"[1] in which the programming problem itself doesn't play a central role. The focus of the activity is reflection on the fundamentals of simple modular design, where tests drive the code, duplication is removed, all the requirements are expressed, and code contains no unnecessary features.[2]

In a coderetreat, a group of developers gathers together, typically on a weekend, for a day of writing code for its own sake, to reflect on their craft and learn from each other. The typical structure of a coderetreat is as follows:

- The code kata to be addressed is John Conway's Game of Life.[3]
- There are five or six coding sessions, each lasting 45 minutes.
- Pair-programming and TDD will be used.
- After each session, all code must be deleted and partners should be swapped.

0740-7459/14/$31.00 © 2014 IEEE