Agile Record

The Magazine for Agile Developers and Agile Testers

Agile Adoption Stories – What Worked and What Didn't

May 2013
issue 14

www.agilerecord.com free digital version made in Germany ISSN 2191-1320

Gilb's Mythodology Column

The Green Week: Reducing Technical Debt by Engineering

by Tom & Kai Gilb

Our client Confirmit.com has used our Evo Agile Method [2] successfully since 2003 [1]. They have adapted it, from the beginning, to *their* environment, and continued to innovate and learn. Their business success has been attributed to their remarkable product quality improvement, and *that* improvement specifically to the Evo Agile method, by them, on their website, and share offerings prospectus. Evo differs from other agile methods, in that it focuses on multiple, quantified, software-and-system qualities.

This column will focus on an innovation, the Green Week, that Confirmit, led by their method champion Trond Johansen, made and reported in 2005; two years after adopting Evo.

When we started in 2003, Confirmit had an 8 year old web-based system; a 'legacy' product that had grown, as most do, to meet rapidly emerging market demands. By 2005 there were the usual difficulties in enhancing the product, a web-based opinion survey tool, serving markets worldwide, to meet new opportunities, quickly and safely.

We recommended in 2003 that they spend 4 days a week on *value delivery cycles* to their customer base, and one day a week 'refactoring'. Their development team at the time was 13 plus 3 testers.

The 4-day value delivery cycle aimed at something like 25 distinct quality improvements (for example Usability Intuitiveness) or performance capacity improvements. The stakeholders aimed at were users and Confirmit's future market. The refactoring was aimed at their *development team*, as stakeholders. The team that did the development initially, also did the maintenance of the system for years, until today.

Let me be explicit, the people who had to 'suffer' bug fixing and long term enhancement were actually in full control of the architecture and design of the entire system. Maintenance was not farmed out to people who just had to suffer it. Most of the staff were not merely programmers, they had formal education in real engineering.

Well, the one day of refactoring was not a great success, while the 4 days of value delivery cycles, to quantified quality and performance requirements was a big success. To my knowledge there is nothing even near-as-good of quantified results, reported for *any* other Agile Effort! If you know of one, AgileRecord (.com) would like to hear from you! One possible reason for lack of success was that the refactoring was one-day-a-week, and I suspect it was a Friday, where Norwegians want to sneak off early for a Cabin Weekend ('working off site'). But I really don't know.

They asked themselves, 'why should our customers get all the quality improvements?' What not, us hard working developers, get some systematic quality improvements too?

So they decided to spend one week a month, using Evo [2] 'engineering' 'ease of maintenance' and 'testability' into their organization and their product. In other words: 3 weeks being customer oriented, and 1 week a month being internally oriented. Of course, improvements in maintenance capability also improve their ability to respond to customers!

User Week 1	User Week 2	User Week 3	Developer Week 4
Select a Goal	Select a Goal	Select a Goal	Select a Goal
Brainstorm Designs	Brainstorm Designs	Brainstorm Designs	Brainstorm Designs
Estimate Design Impact/Cost	Estimate Design Impact/Cost	Estimate Design Impact/Cost	Estimate Design Impact/Cost
Pick best design	Pick best design	Pick best design	Pick best design
Implement design	Implement design	Implement design	Implement design
Test design	Test design	Test design	Test design
Update Progress to Goal	Update Progress to Goal	Update Progress to Goal	Update Progress to Goal

Figure 1. The weekly development cycles, with the Green Week.

The key idea here is that we start by *quantifying as requirements*, all Confirmit system (the software product, the service product, the technical organization) attributes, related to ease of maintaining the system, in the widest sense of 'maintaining' [3]

Here are the requirements they quantified as requirements initially:

Speed, Maintainability, Nunit Tests, Peer Tests, Test Director Tests, Robustness.Correctness, Robustness.Boundary Conditions, Resource Usage CPU, Maintainability DocCode, Synchronization Status.

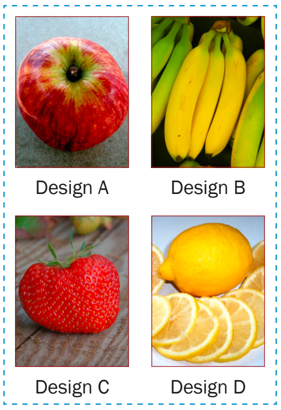
These are defined by Confirmit by a Scale of measure, a Past level on that scale, a Tolerable Level, and a Goal Level.

For example (made up by paper author for illustration):

```
Bug Maintainability:
Scale: Mean Hours to totally fix test and release a defined
[Priority] Bug
Past [2013, Priority = Critical]  1 hour
Goal [Release 23.0, Priority = Critical] 15 minutes.
```

As with the 2003 Confirmit Evo process, the engineers *themselves* are *completely empowered* to find designs they believe will move them towards their quantified goals. Trond calls this "Empowered Creativity". A short, about 30 minutes, design meeting starts the week's process for each chosen Goal they want to work on, and each design idea must be coupled with an estimate of how much progress towards the goal, the design will contribute. [4, Impact Estimation] The team will agree to adopt the designs with the best estimated potential effect on *their* goals, that can be done in the weekly cycle.

Design Suggestions



Impacts to Cost Evaluation

	A	B	C	D
Goal	30 %	10 %	−10 %	80 %
Cost	10	50	1	20
G/C	3:1	1:5	?	4:1

Figure 2. The team selects one of their suggested designs based on expected impact on a chosen maintainability goal, and its cost.

They get numeric feedback at the end of the week's cycle, on progress towards their Goal levels, and this is their starting point the next month of their 'Green Week', as they call it.

So, there you have it. Reduction of 'Technical Debt', Technical Debt expressed *quantitatively* and *multi-dimensionally*, by means of real software *engineering, not mere 'softcrafting'*.

Do you want to write an article for the next **Agile Record**?

Agile Record

If you would like to participate in the next issue, please complete the following steps:

1. Download the Microsoft Word template from our website and complete it, including a short biography of the author(s).
2. Submit your finished article to our online system, after which our editorial board will rate it and our editor José Díaz will accept or reject it.
3. If your article is accepted, we will contact you and ask you to send the figures and pictures to be included in the article in the highest resolution you can provide (72 DPI for screenshots, 300 DPI minimum for all other image files), as well as the photo(s) of the author(s) to **editorial@agilerecord.com**.
4. Download the consent form (PDF) from our website and sign it, scan it and send it to **editorial@agilerecord.com**. If an article was written by several authors, all of them have to sign the consent form individually.
5. After your article has been reviewed for spelling and grammar and has been returned to you, you accept or reject the changes.

Note:

- Please take care of copyrights and registered trademarks.
- We do not accept sales pitches advertising a product or company.
- Your article must not have been published before.
- There will be no remuneration.

Agile Record schedule

Issue	Deadline for Articles	Topic
No. 15 (Aug 2013)	July 1, 2013	"Distributed Team Management"
No. 16 (Nov 2013)	October 1, 2013	"Refactoring"
No. 17 (Feb 2014)	January 1, 2014	"Security Testing in an Agile Environment"

www.agilerecord.com
For any questions please email us at **editorial@agilerecord.com**.

Agile methods will hopefully mature towards Agile *Engineering*, as they attack larger and more complex-and-critical problems.

Several prominent Agile Gurus such as Jeff Sutherland (tweets, Agile Alliance), Mike Cohn (blog), Agile Alliance, have agreed in public statements that we need to add an 'engineering' dimension like this to Agile and Scrum, as mainly taught and practiced today. One client Deutsche Bank (source: Paul Fields, 2012) has recommended Evo as a protective value management envelope for all Agile processes in the bank.

But we have, at least, started *the journey. Technology transfer takes 15 to 25 years. It took that long from my first 'software iteration 'Evo' publications in the Seventies, Eighties and onward [5], to get to the Agile Manifesto spark. Move over programmers, here come the engineers!*

References

[1] Confirmit Case
http://www.gilb.com/tiki-download_file.php?fileId=32
From Waterfall to Evolutionary Development (Evo): How we rapidly created faster, more user-friendly, and more productive software products for a competitive multi-national market. INCOSE 2005, T. Gilb and Trond Johansen.

[2] Evo method.
http://www.gilb.com/dl487
The Evo 'Standard' Process Description
Chapter 10 of Competitive Engineering: Evolutionary Project Management:
http://www.gilb.com/tiki-download_file.php?fileId=77

[3] Quantifying Maintainability Quality Requirements.
Gilb ACCU Conference, Oxford UK, Lecture Slides on "Designing Maintainability"
http://www.gilb.com/tiki-download_file.php?fileId=171
Designing Maintainability in Software Engineering: a Quantified Approach (INCOSE 2008)
http://www.gilb.com/tiki-download_file.php?fileId=138
Chapter 5 CE Book: Scales of Measure:
http://www.gilb.com/tiki-download_file.php?fileId=26

[4] Impact Estimation:
Impact Estimation Tutorial slides 1 Day tutorial at INCOSE Rome 2012
http://www.gilb.com/dl553

[5] Early published suggestions that we do Iterative and Incremental Software: Spreading ideas, like Agile Engineering 'takes time'

- Sutherland, Cohn, Beck, Highsmith, Cunningham, Martin citations, + 1985 ACM Paper.
- *http://www.gilb.com/dl573*
- *http://www.amazon.co.uk/Principles-Software-Engineering-Management-Gilb/dp/0201192462*
  - 1988 Book, deeply presenting Evolutionary development.
  - This book also contains considerable detail on multidimensional quantified agile engineering, but this was not picked up by the people who picked up the incremental ideas from the same book. Things take time. ■

> about the authors

Tom Gilb and Kai Gilb

Tom Gilb and Kai Gilb have, together with many professional friends and clients, personally developed the Agile methods they teach. The methods have been developed over five decades of practice all over the world in both small companies and projects, as well as in the largest companies and projects. Their website ***www.gilb.com/downloads*** *offers free papers, slides, and cases about Agile and other subjects. There are many organisations, and individuals, who use some or all of their methods. IBM and HP were two early corporate-wide adopters (1980, 1988). Recently (2012) over 15,000 engineers at Intel have voluntarily adopted the Planguage requirements specification methods; in addition to practicing to a lesser extent Evo, Spec QC and other Gilb methods. Many other multinationals are in various phases of adopting and practicing the Gilb methods. Many smaller companies also use the methods.*

Tom Gilb

Tom is the author of nine published books, and hundreds of papers on Agile and related subjects. His latest book 'Competitive Engineering' (CE) is a detailed handbook on the standards for the 'Evo' (Evolutionary) Agile Method, and also for Agile Spec QC. The CE book also, uniquely in the Agile community, defines an Agile Planning Language, called 'Planguage' for Quality Value Delivery Management. His 1988 book, Principles of Software Engineering Management (now in 20th Printing) is the publicly acknowledged source of inspiration from leaders in the Agile community (Beck, Highsmith, and many more), regarding iterative and incremental development methods. Research (Larman, Southampton University) has determined that Tom was the earliest published source campaigning for Agile methods (Evo) for IT and Software. His first 20-sprint agile (Evo) incremental value delivery project was done in 1960, in Oslo. Tom has guest lectured at universities all over UK, Europe, China, India, USA, Korea – and has been a keynote speaker at dozens of technical conferences internationally.

Kai Gilb

Kai Gilb has partnered with Tom in developing these ideas, holding courses and practicing them with clients since 1992. He coaches managers and product owners, writes papers, develops the courses, and is writing his own book, 'Evo – Evolutionary Project Management & Product Development.' Tom & Kai work well as a team; they approach the art of teaching their common methods somewhat differently. Consequently the students benefit from two different styles.