Lightning Talks

Olve Maudal – Technical debt is good!

Allan Kelly – Dialogue Sheet Retrospectives

Martin Winkler – There Is No Base

Didier Verna – Letting Go of Control

Tom Gilb – User Stories: Why They Might Be Too Light

Chris Oldwood – The Ups and Downs of Being an ACCU Member

Pete Goodliffe – Manyfestos

Peter Pilgrim – We Are Going To Live Forever!

Roger Orr – Errors Are Evil

Mark Dalgarno – Optimizing For Unhappiness

Lightning Talks

Olve Maudal – Technical debt is good!

Allan Kelly – Dialogue Sheet Retrospectives

Martin Winkler – There Is No Base

Didier Verna – Letting Go of Control

Tom Gilb – User Stories: Why They Might Be Too Light

Chris Oldwood – The Ups and Downs of Being an ACCU Member

Pete Goodliffe – Manyfestos

Peter Pilgrim – We Are Going To Live Forever!

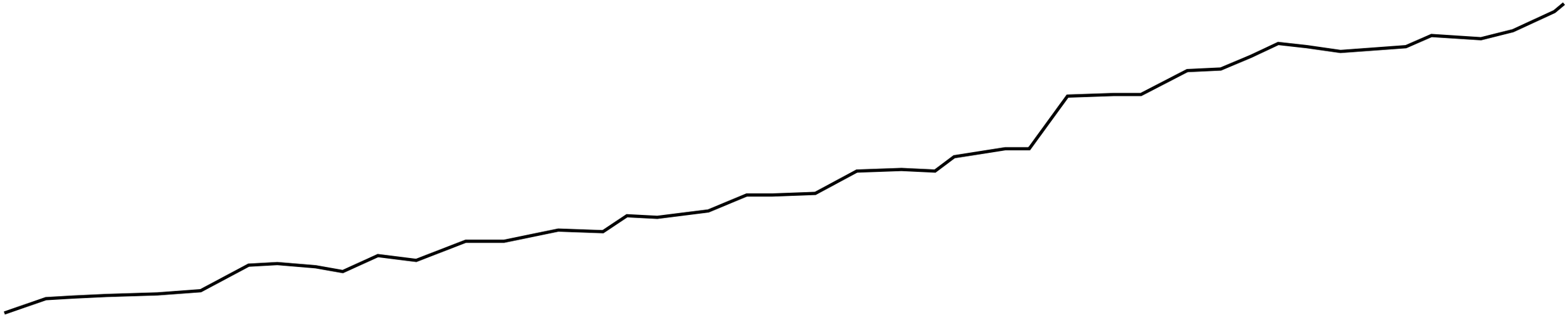
Roger Orr – Errors Are Evil

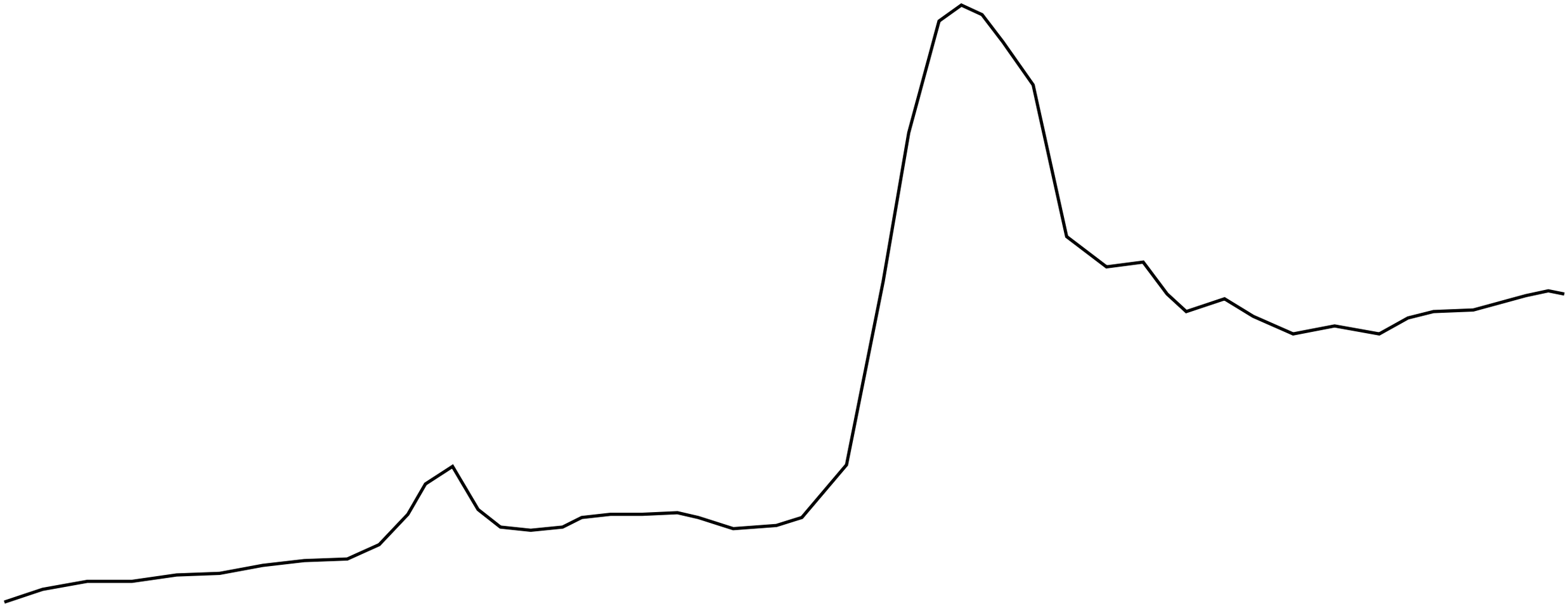
Mark Dalgarno – Optimizing For Unhappiness

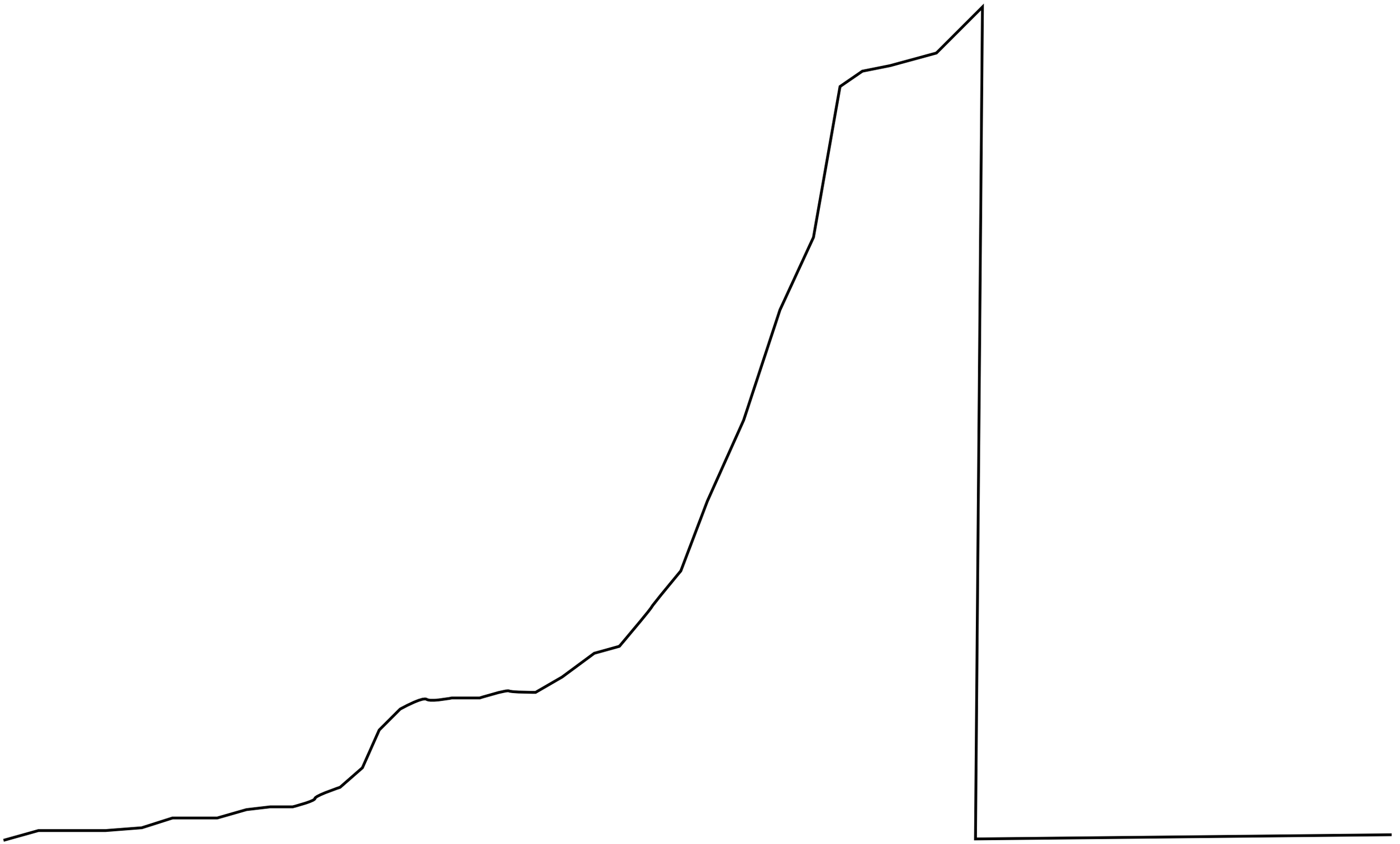
ACCU 2011

Technical debt is good!

A 5 minute lightning talk at the ACCU conference
Oxford, April 13-16 2011







Technical Debt Quadrant

Reckless	**Prudent**
"We don't have time for design"	*"We must ship now and deal with consequences"*
Deliberate	
Inadvertent	
"What's Layering?"	*"Now we know how we should have done it"*

http://martinfowler.com/bliki/TechnicalDebtQuadrant.html


VISA
MasterCard
VISA

Klæbu
Sparebank

"Let's upgrade this external library before the next release instead"

"Let's upgrade this external library before the next release instead"

INFORMED

"We do not have time to plan these activities"

"We do not have time to plan these activities"

IDIOTIC

"Hmm, this is not as elegant as I hoped for"

"Hmm, this is not as elegant as I hoped for"

"What is polymorphism?"

"What is polymorphism?"

"Ah, now we understand how we should have done it"

"Ah, now we understand how we should have done it"

LEARNING

"Let's copy-paste this code, then fix just what we need"

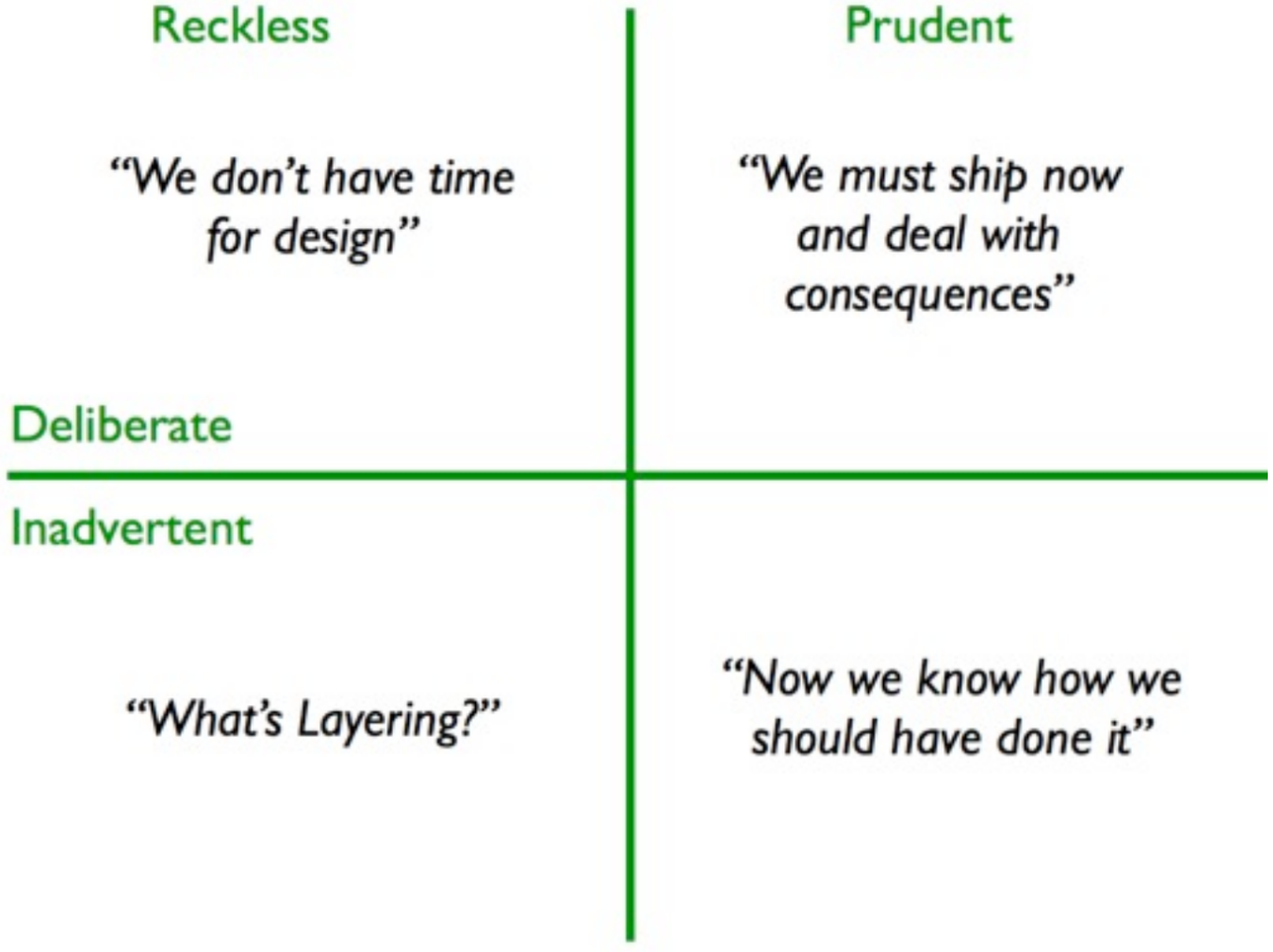
"Let's copy-paste this code, then fix just what we need"

"Let's copy-paste this code, then fix just what we need"

"Let us just ship the product, then deal with the consequeces"

"Let us just ship the product, then deal with the consequeces"

INTENTIONAL


Reckless
Prudent
"We don't have time for design"
"We must ship now and deal with consequences"
Deliberate
Inadvertent
"What's Layering?"
"Now we know how we should have done it"

INFORMED

	Reckless	Prudent
Deliberate	*"We don't have time for design"*	*"We must ship now and deal with consequences"*
Inadvertent	*"What's Layering?"*	*"Now we know how we should have done it"*

IDIOTIC

INFORMED

	Reckless	Prudent
Deliberate	*"We don't have time for design"*	*"We must ship now and deal with consequences"*
Inadvertent	*"What's Layering?"*	*"Now we know how we should have done it"*

IDIOTIC

INFORMED

Deliberate	Reckless	Prudent
	"We don't have time for design"	*"We must ship now and deal with consequences"*
Inadvertent	*"What's Layering?"*	*"Now we know how we should have done it"*

INEVITABLE

IDIOTIC

INFORMED

Reckless	Prudent
"We don't have time for design"	*"We must ship now and deal with consequences"*

Deliberate

Inadvertent

"What's Layering?"	*"Now we know how we should have done it"*

INAPT

INEVITABLE

IDIOTIC

INFORMED

Reckless	Prudent
"We don't have time for design"	*"We must ship now and deal with consequences"*

Deliberate

Inadvertent

"What's Layering?"	*"Now we know how we should have done it"*

INAPT

INEVITABLE

LEARNING

IDIOTIC

INFORMED

STUPID

	Reckless	Prudent
Deliberate	*"We don't have time for design"*	*"We must ship now and deal with consequences"*
Inadvertent	*"What's Layering?"*	*"Now we know how we should have done it"*

INAPT

INEVITABLE

LEARNING

IDIOTIC

STUPID

INFORMED

CONSIDERED

	Reckless	Prudent
Deliberate	*"We don't have time for design"*	*"We must ship now and deal with consequences"*
Inadvertent	*"What's Layering?"*	*"Now we know how we should have done it"*

INAPT

INEVITABLE

LEARNING

IDIOTIC

STUPID

INFORMED

CONSIDERED

INTENTIONAL

	Reckless	Prudent
Deliberate	*"We don't have time for design"*	*"We must ship now and deal with consequences"*
Inadvertent	*"What's Layering?"*	*"Now we know how we should have done it"*

INAPT

INEVITABLE

LEARNING

IDIOTIC

STUPID

INFORMED

CONSIDERED

INTENTIONAL

	Reckless	Prudent
Deliberate	*"We don't have time for design"*	*"We must ship now and deal with consequences"*
Inadvertent	*"What's Layering?"*	*"Now we know how we should have done it"*

INAPT

INEVITABLE

LEARNING

IDIOTIC

STUPID

INFORMED

CONSIDERED

INTENTIONAL

	Reckless	Prudent
Deliberate	*"We don't have time for design"*	*"We must ship now and deal with consequences"*
Inadvertent	*"What's Layering?"*	*"Now we know how we should have done it"*

INFORMED
WISE
INTENTIONAL

	Reckless	Prudent
Deliberate	*"We don't have time for design"*	*"We must ship now and deal with consequences"*
Inadvertent	*"What's Layering?"*	*"Now we know how we should have done it"*

Klæbu Sparebank

150 år

1858 – 2008

Klæbu Sparebank

The savvy developer treats technical debt just as the entrepreneur does financial debt. They use it. It speeds delivery, so long as it is properly managed.

(source: http://www.c2.com/cgi/wiki?TechnicalDebt)

Greed is good!

Technical debt is good!

Lightning Talks

Olve Maudal – Technical debt is good!

Allan Kelly – Dialogue Sheet Retrospectives

Martin Winkler – There Is No Base

Didier Verna – Letting Go of Control

Tom Gilb – User Stories: Why They Might Be Too Light

Chris Oldwood – The Ups and Downs of Being an ACCU Member

Pete Goodliffe – Manyfestos

Peter Pilgrim – We Are Going To Live Forever!

Roger Orr – Errors Are Evil

Mark Dalgarno – Optimizing For Unhappiness

Lightning Talks

Olve Maudal – Technical debt is good!

Allan Kelly – Dialogue Sheet Retrospectives

Martin Winkler – There Is No Base

Didier Verna – Letting Go of Control

Tom Gilb – User Stories: Why They Might Be Too Light

Chris Oldwood – The Ups and Downs of Being an ACCU Member

Pete Goodliffe – Manyfestos

Peter Pilgrim – We Are Going To Live Forever!

Roger Orr – Errors Are Evil

Mark Dalgarno – Optimizing For Unhappiness

Lightning Talks

Olve Maudal – Technical debt is good!

Allan Kelly – Dialogue Sheet Retrospectives

Martin Winkler – There Is No Base

Didier Verna – Letting Go of Control

Tom Gilb – User Stories: Why They Might Be Too Light

Chris Oldwood – The Ups and Downs of Being an ACCU Member

Pete Goodliffe – Manyfestos

Peter Pilgrim – We Are Going To Live Forever!

Roger Orr – Errors Are Evil

Mark Dalgarno – Optimizing For Unhappiness

Letting Go of Control

Part 1/2

Didier Verna

Conclusion

- Our software is *out of control*
- This is only going to get *worse*
- We should be *afraid*
- We should be *ashamed*

The birth of a baby

A miracle of Nature

- Darwin: Evolution is far from perfection
- Up to 50% pregnancies lead to spontaneous abortion

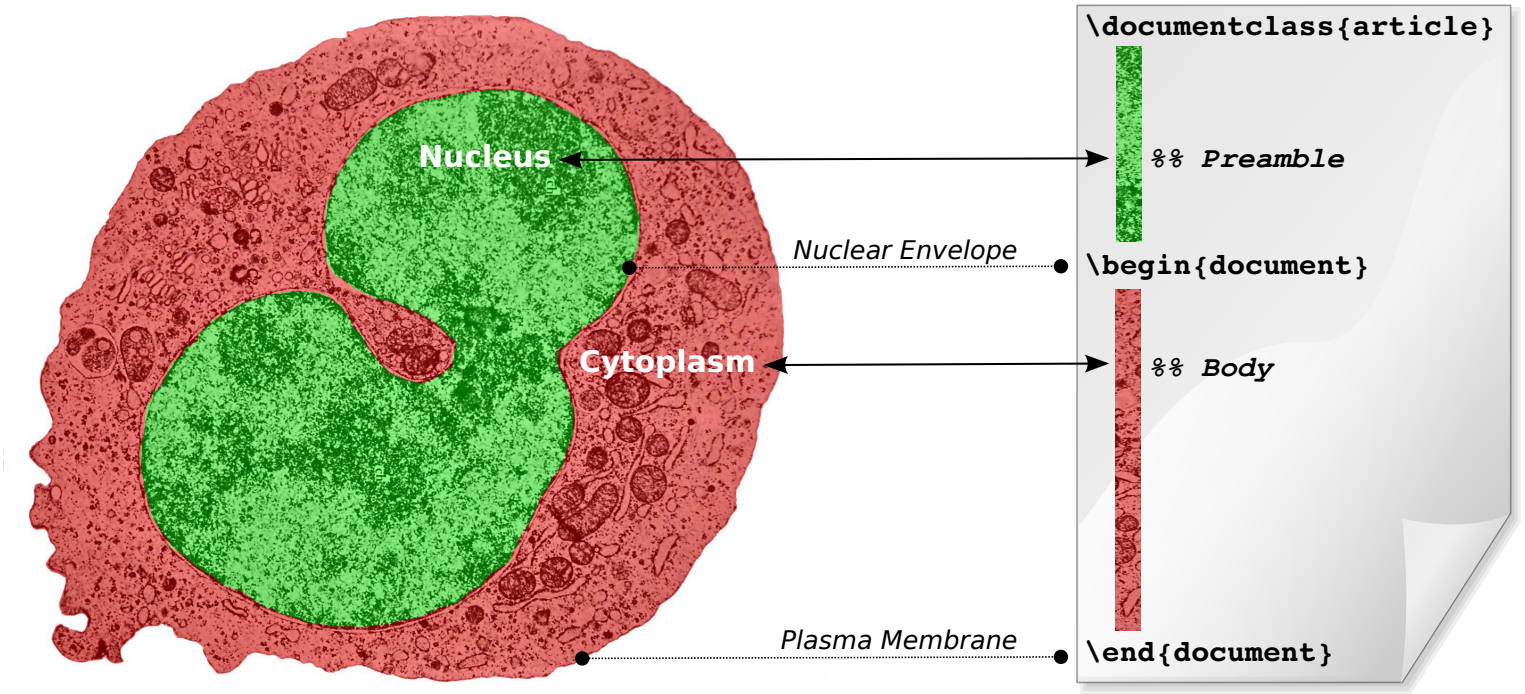
The birth of a baby DOCUMENT

A miracle of DON KNUTH

- When it doesn't work, you don't really know why
- When it does work, you *really* don't know why

The LaTeX biotope

And the viral propagation of styles



- LaTeX documents as eukaryote cells
- Styles as viral infection with new geneTeX material

Houston, we have a problem...

Classes, Styles, Conflicts: the biological realm of LaTeX. *In TUGBoat 31:2, proceedings of TUG 2010, the TeX Users Group Conference*, San Francisco, July 2010.

Houston, we have a problem...

Classes, Styles, Conflicts: the biological realm of LaTeX. *In TUGBoat 31:2, proceedings of TUG 2010, the TeX Users Group Conference*, San Francisco, July 2010.

- LaTeX is a mess
- Open Source software is a mess
- Proprietary software's gotta be a mess too

Intermediate conclusion

- Darwin / Jacob: Nature is a *tinkerer*
- Alon: the tinkerer as an engineer
- Verna: the engineer as a tinkerer

Intermediate conclusion

- Darwin / Jacob: Nature is a *tinkerer*
- Alon: the tinkerer as an engineer
- Verna: the engineer as a tinkerer

We should be ashamed!

End of Part 1

Stephanie Forrest:

"As programmers, we like to think of software as the product of our intelligent design, carefully crafted to meet well-specified goals. In reality, software evolves inadvertently through the actions of many individual programmers, often leading to unanticipated consequences. Large complex software systems are subject to constraints similar to those faced by evolving biological systems, and we have much to gain by viewing software through the lens of evolutionary biology."

Lightning Talks

Olve Maudal – Technical debt is good!

Allan Kelly – Dialogue Sheet Retrospectives

Martin Winkler – There Is No Base

Didier Verna – Letting Go of Control

Tom Gilb – User Stories: Why They Might Be Too Light

Chris Oldwood – The Ups and Downs of Being an ACCU Member

Pete Goodliffe – Manyfestos

Peter Pilgrim – We Are Going To Live Forever!

Roger Orr – Errors Are Evil

Mark Dalgarno – Optimizing For Unhappiness

User Stories: why they might be too light

by Tom @ Gilb . com

5 Minute Lightening Talk

ACCU Oxford

Thursday 14 April 2011, 18:00 session

Published Paper in AgileRecord.com

http://www.gilb.com/tiki-download_file.php?fileId=461

Gilb's Mythodology Column

User Stories: A Skeptical View

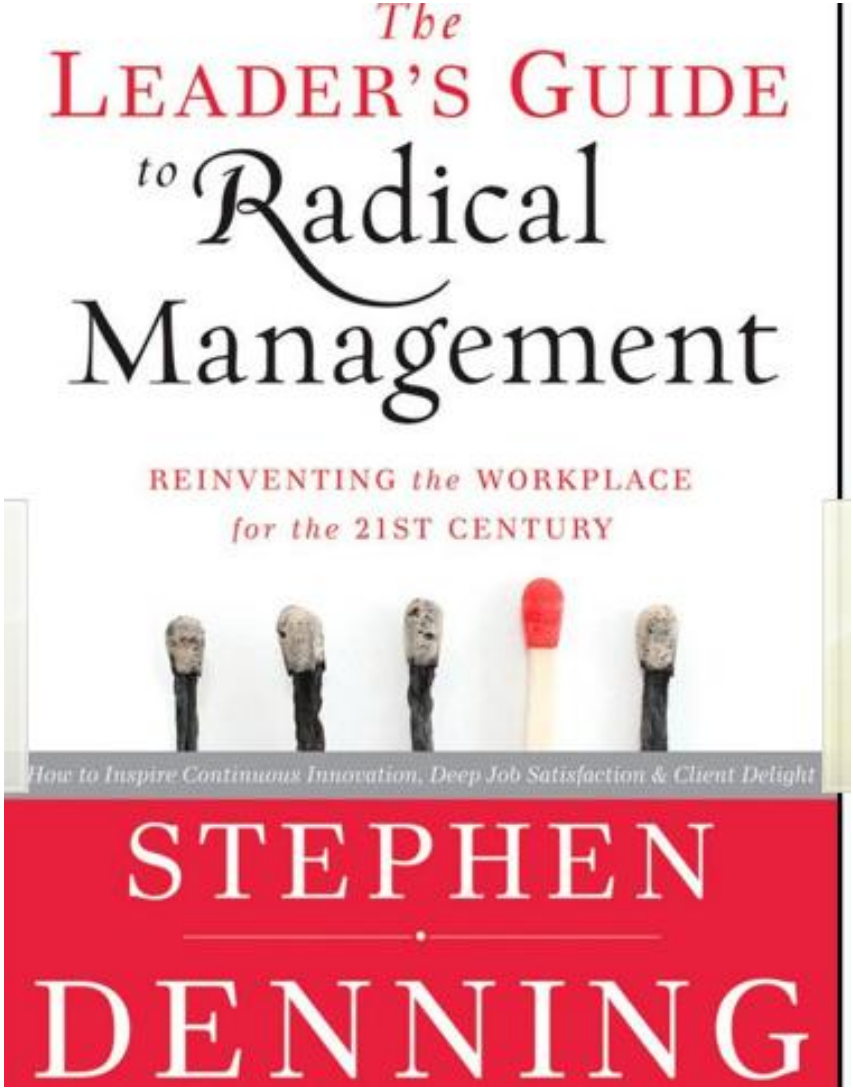
by Tom and Kai Gilb

The Skeptical View

We agree with the ideals of user stories, in the 'Myths' [1, Denning & Cohn] discussed below, but do not agree at all to Myth arguments given, that user stories are a good, sufficient or even of our product clearly superior to all competitive products at all times.

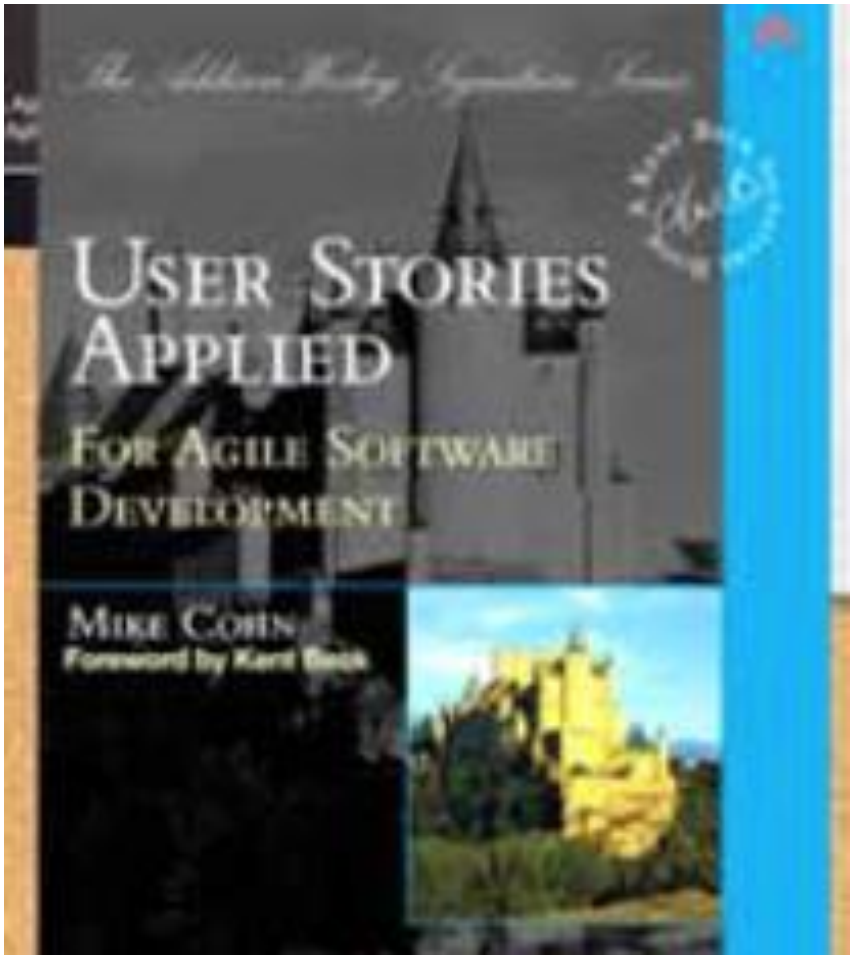
Scale: average seconds needed for defined [Users] to Correctly Complete defined [Tasks] defined [Help]

Original Claims



http://stevedenning.typepad.com/

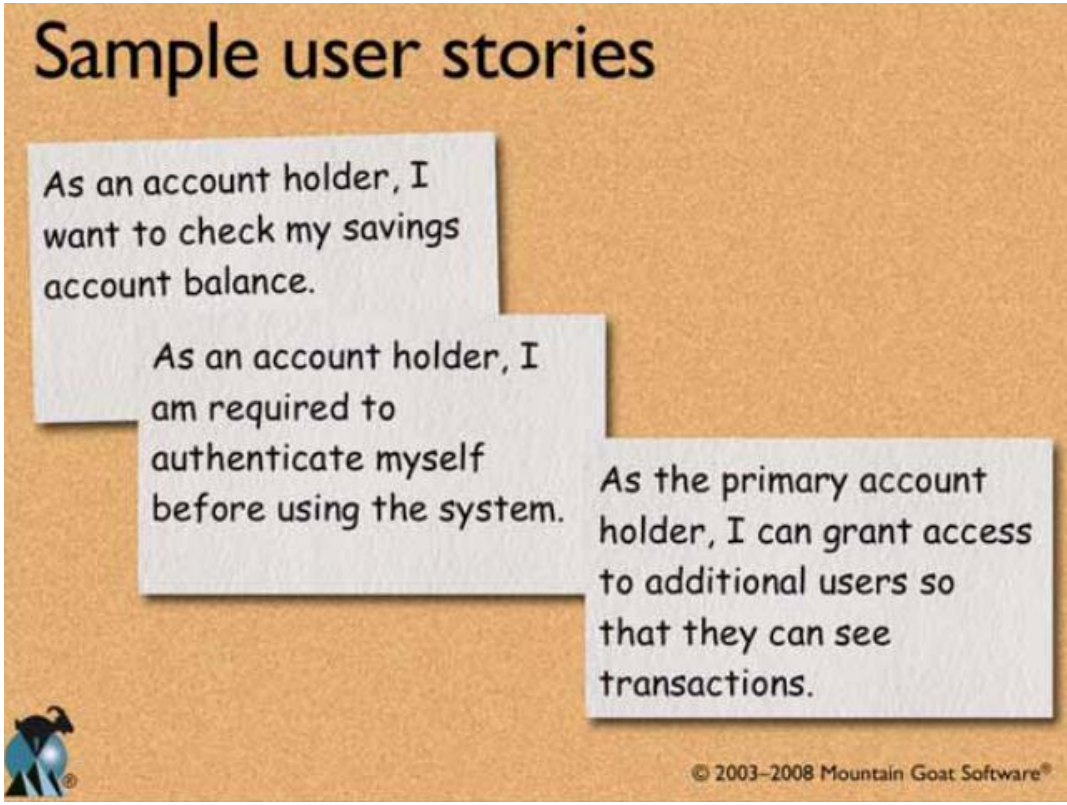
From Mike Cohns User Stories Work



User Stories: Samples

Structure

- **Stakeholder A**
- **Needs X**
- **Because Y**



My General Assertion

- **User Stories are good enough for small scale and non-critical projects**
- **But, they are not adequate for non-trivial projects**
- **The claims (myths in slides ahead) are not true when we scale up**

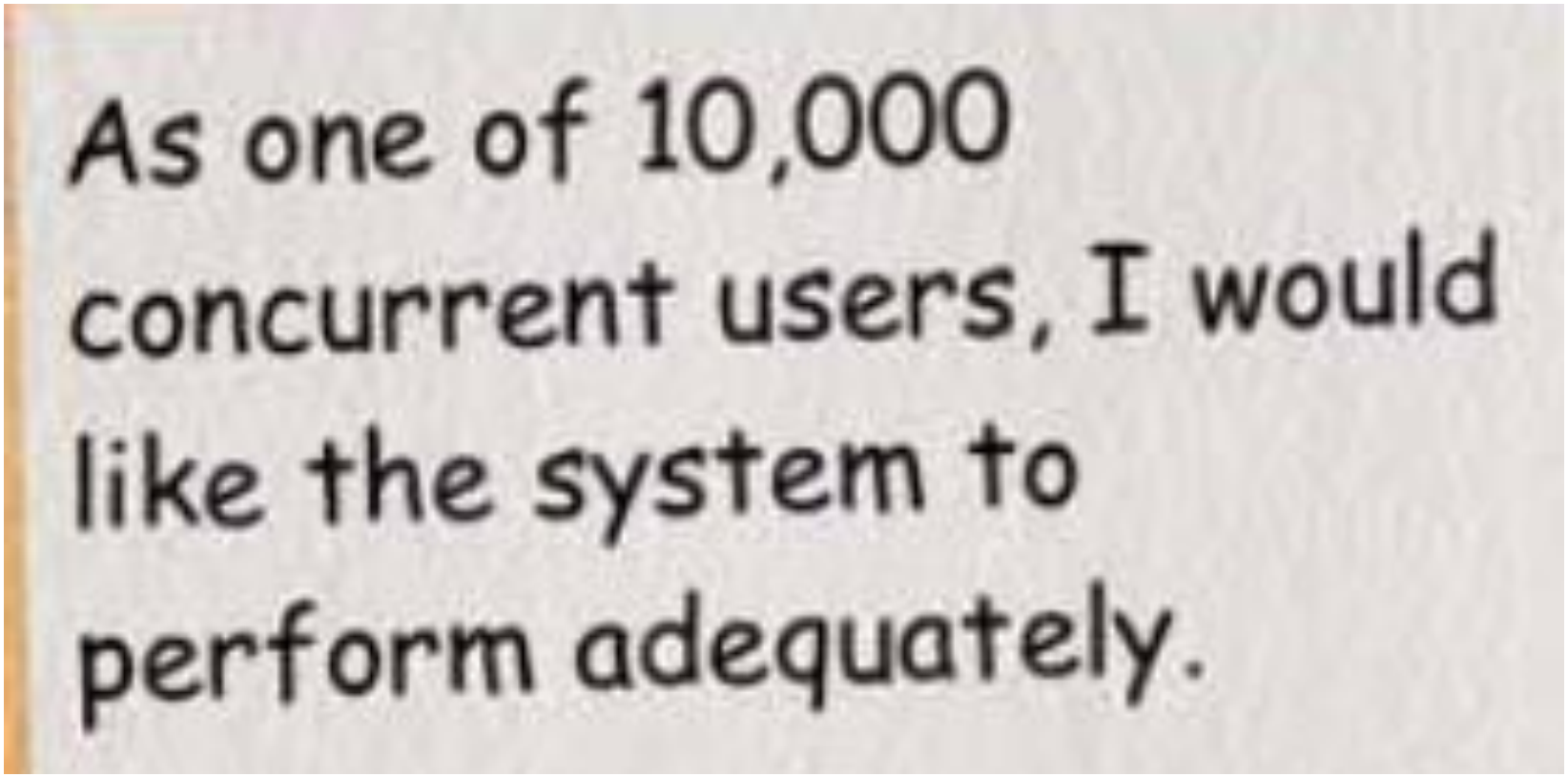
Myth 1:
User stories and the conversations provoked by them comprise *verbal communication*, which is clearer than written communication.

- **Verbal communication is not clearer than written communication**
- ***Dialogue***
  - **to clear up *'bad written user stories'***
  - **does not prove that there are no superior written formats**

- I, as a user, want clearer interfaces to save time

- Usability:
  - Scale: Time for defined Users to Successfully complete defined Tasks
  - Goal [Users = Novices, Tasks = Inquiry] 20 Seconds.
  - Successfully: defined as: correct, no need to correct it later.

Myth 2:
"User stories represent a common language. They are intelligible to both users and developers."



- What does 'perform' mean ?
- What does 'adequately' mean?
- What does it mean under higher or lower loads?

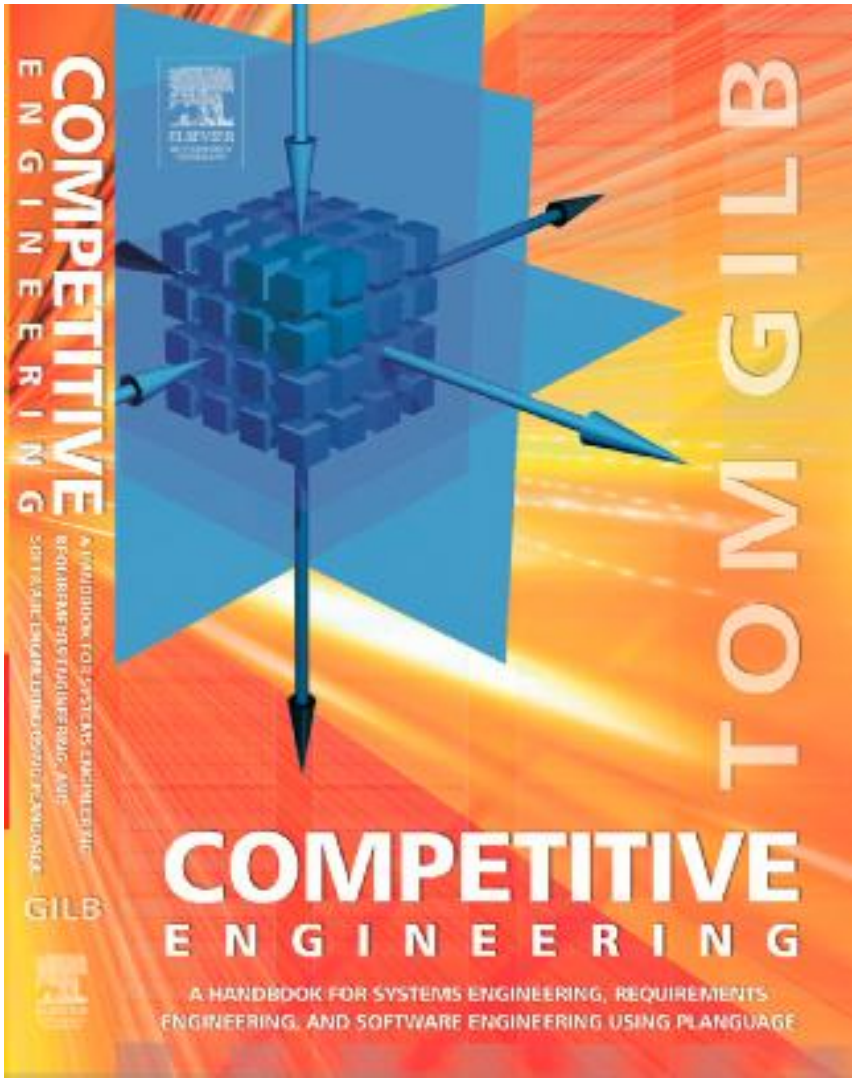
Myth 3:
“User stories are the *right size* for planning and prioritizing.”

Myth 4:
User stories are *ideal for iterative development*, which is the nature of most software development.

Myth 5:
“User stories help *establish priorities* that make sense to both users and developers.”

Myth 6:
“The process enables *transparency*.
Everyone understands
why.”

References



- Ask me for free digital copy
  - Tom@Gilb.com
- Download Related Papers and Slides and 2 Chapters at
- www.Gilb.com
- (Downloads tab)

Lightning Talks

ACCU 2011

Olve Maudal – Technical debt is good!

Allan Kelly – Dialogue Sheet Retrospectives

Martin Winkler – There Is No Base

Didier Verna – Letting Go of Control

Tom Gilb – User Stories: Why They Might Be Too Light

Chris Oldwood – The Ups and Downs of Being an ACCU Member

Pete Goodliffe – Manyfestos

Peter Pilgrim – We Are Going To Live Forever!

Roger Orr – Errors Are Evil

Mark Dalgarno – Optimizing For Unhappiness

Lightning Talks

Olve Maudal – Technical debt is good!

Allan Kelly – Dialogue Sheet Retrospectives

Martin Winkler – There Is No Base

Didier Verna – Letting Go of Control

Tom Gilb – User Stories: Why They Might Be Too Light

Chris Oldwood – The Ups and Downs of Being an ACCU Member

Pete Goodliffe – Manyfestos

Peter Pilgrim – We Are Going To Live Forever!

Roger Orr – Errors Are Evil

Mark Dalgarno – Optimizing For Unhappiness

MANY FESTOS

PETE GOODLIFFE ★ ACCU 2011, Oxford, UK

Confusion of goals and perfection of means seems, in my opinion, to characterise our age.

Albert Einstein

it’s an epidemic!

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:

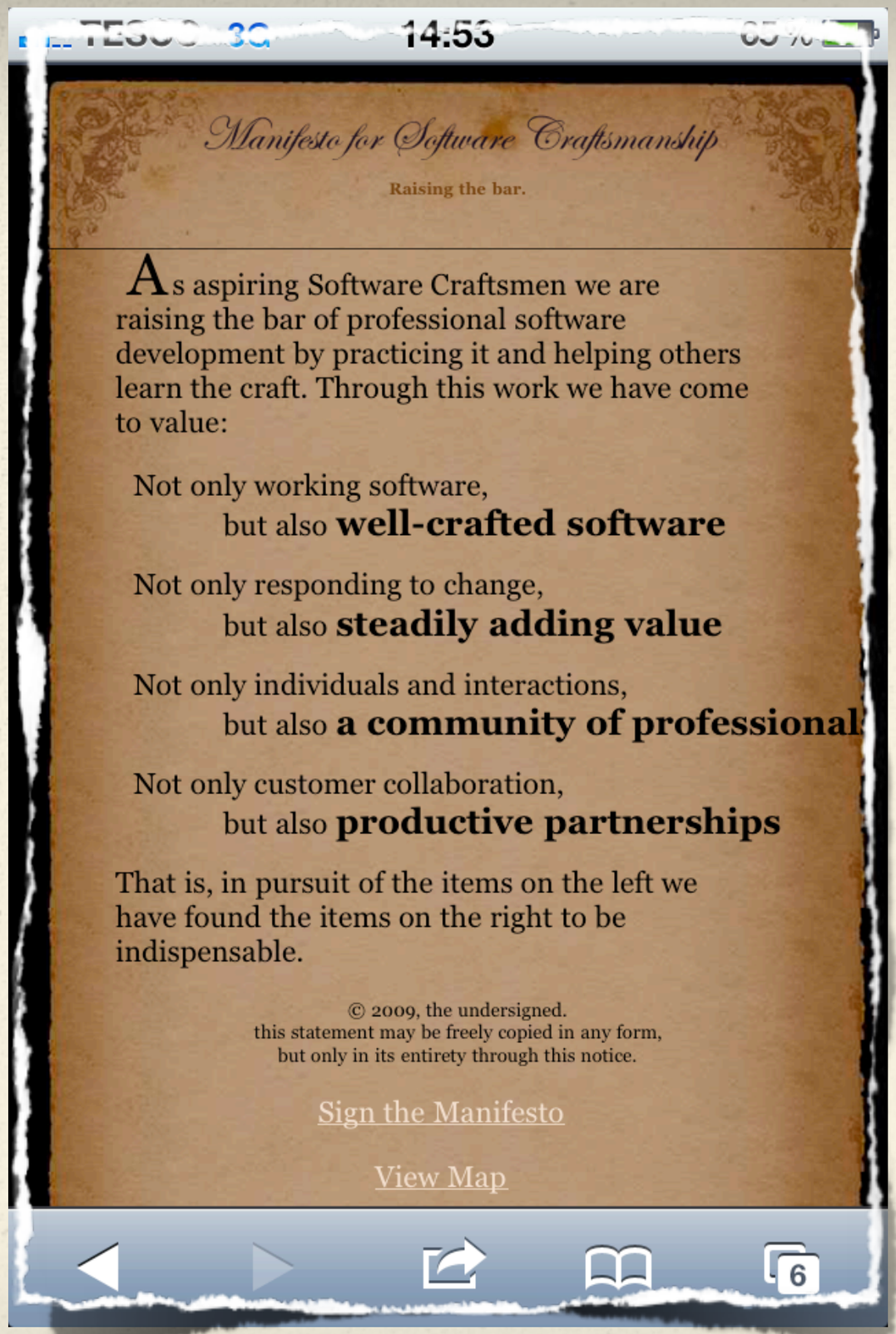
Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

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

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

Twelve Principles of Agile Software

Become a Signatory
View Signatories


14:53
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 professional
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.
© 2009, the undersigned.
this statement may be freely copied in any form,
but only in its entirety through this notice.
Sign the Manifesto
View Map
6

Refactoring Manifesto

Because the world needs better code

1. Make your products live longer!

   Refactoring means taking the opportunity to keep your product alive. Don't ditch it, stitch it! Don't end it, mend it! Refactoring is not a needless cost. It is anti-needless complexity that prevents change.

2. Design should be simple so that it is easy to refactor.

   Product designers: Make your products easy to change. Write clean, understandable code. Consumers: Buy products that are continuously refactored, or else find out why the developers didn't do that. Be critical and inquisitive.

3. Refactoring is not rewriting.

   Rewriting is throwing away the broken bit. This is NOT the kind of refactoring that we're talking about.

4. What doesn't kill it makes it stronger.

   Every time we refactor code, we add to its potential, its history, its soul and its inherent beauty.

5. Refactoring is a creative challenge.

   Refactoring is good for the imagination. Using new techniques, tools and materials ushers in possibility rather than dead ends.

6. Refactoring survives fashion.

   Refactoring is not about styling or trends. There are no due-dates on continuously refactored code.

7. To refactor is to discover.

   As you refactor objects, you'll learn amazing things about how they actually work. Or don't work.

SOA Manifesto

Service orientation is a paradigm that frames what you do. Service-oriented architecture (SOA) is a type of architecture that results from applying service orientation.

We have been applying service orientation to help organizations consistently deliver sustainable business value, with increased agility and cost effectiveness, in line with changing business needs.

Through our work we have come to prioritize:

Business value over technical strategy

Strategic goals over project-specific benefits

Intrinsic interoperability over custom integration

Shared services over specific-purpose implementations

Flexibility over optimization

Evolutionary refinement over pursuit of initial perfection

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

Guiding Principles

We follow these principles:

Respect the social and power structure of the organization.

Recognize that SOA ultimately demands change on many levels.

The scope of SOA adoption can vary. Keep efforts manageable and within meaningful boundaries.

Products and standards alone will neither give you SOA nor apply the service orientation paradigm for you.

SOA can be realized through a variety of technologies and standards.

GNU Operating System

The GNU Manifesto

The GNU Manifesto (which appears below) was written by Richard Stallman at the beginning of the GNU Project, to ask for participation and support. For the first few years, it was updated in minor ways to account for developments, but now it seems best to leave it unchanged as most people have seen it.

Since that time, we have learned about certain common misunderstandings that different wording could help avoid. Footnotes added since 1993 help clarify these points.

For up-to-date information about the available GNU software, please see the information available on our web server, in particular our list of software. For how to contribute, see http://www.gnu.org/help.

What's GNU? Gnu's Not Unix!

GNU, which stands for Gnu's Not Unix, is the name for the complete Unix-compatible software system which I am writing so that I can give it away free to everyone who can use it.(1) Several other volunteers are helping me. Contributions of time, money, programs and equipment are greatly needed.

So far we have an Emacs text editor with Lisp for writing editor commands, a source level debugger, a yacc-compatible parser generator, a linker, and around 35 utilities. A shell (command interpreter) is nearly completed. A new portable optimizing C compiler has compiled itself and may be released this year. An initial kernel exists but many more features are needed to emulate Unix. When the kernel and compiler are finished, it will be possible to distribute a GNU system suitable for program development. We will use TeX as our text formatter, but an nroff is being worked on. We will use the free, portable X Window System as well. After this we will add a portable Common Lisp, an Empire game, a spreadsheet, and hundreds of other things, plus online documentation. We hope to supply, eventually, everything useful that normally comes with a Unix system, and more.

GNU will be able to run Unix programs, but will not be identical to Unix. We will make all improvements that are convenient, based on our experience with other operating systems. In particular, we plan to have longer file names, file version numbers, a crashproof file system, file name completion perhaps, terminal-independent display support, and perhaps eventually a Lisp-based window system through which several Lisp programs and ordinary Unix programs can share a screen. Both C and Lisp will be available as system programming languages. We will try to support UUCP, MIT Chaosnet, and Internet protocols for communication.

GNU is aimed initially at machines in the 68000/16000 class with virtual memory, because they are

Software Testing Manifesto

www.softwaretesting...

Google

EuroSTAR 2008

SOFTWARE TESTING MANIFESTO

Manifesto | Post-it Notes | Most Voted | News | Contributors | Pictures | Suggestions

Manifesto

During EuroSTAR 2008 three workshops were held with the purpose of starting a discussion about software testing. To create a statement about the future of software testing to the test community worldwide; a Manifesto that gives the major guidelines for our journey into the future.

On this website the post-it notes are shown which were created during those workshops. The statements on these post-it notes are the first step towards the making of the Software Testing Manifesto. Each statement can be voted up or down by clicking on the green or red arrow shown besides each post-it. Furthermore every statement can be changed or discussed by leaving comments on that specific post-it note.

The final goal is that we will be left with the top ten most voted manifesto statements. These statements are the basis for the manifesto and will be transferred in to the final Software Testing Manifesto which will be published on this website. Change the way we look at the future of software testing. A future which also will be a big part of your own future.

CATEGORIES
- Competence
- EuroStar Favorites
- News
- People
- Process

RECENT POSTS
- Become Test-infected
- Be Inspirational
- Early Removal
- New Technology
- Not Predictable

Search

Webmastered by Marco van der Spek

Library Software Manifesto

By Roy Tennant - Posted on November 12th, 2007

Tagged: Integrated Library Systems

This is offered in an attempt to rationalize the relationship between libraries and library systems vendors, which is presently unhealthy. I encourage comments directly on this post (see below) or emailed to me directly.

Consumer Rights

- **I have a right to know what exists *now* and what is potential future functionality.** — Marketing materials may tout a new product or a new version of a product, but I have a right to know what I will receive if I buy the product today.
- **I have a right to use what I buy.** — For example, it should not cost extra to create another index of my data.
- **I have a right to the API if I've bought the product.** — An application program interface (API) is simply a structured way for one application to communicate with another. In other words, the ability of a software program to send a structured query to another application and receive a structured response. Using the API for a product I've bought should not incur an additional charge.
- **I have a right to complete and accurate documentation.**
- **I have a right to my data.** — This includes the ability to bring forward not just my records, but also usage data (for example, how many times a book was checked out), since such information will be increasingly important for relevance

TESCO 3G 14:57 63%

Open Cloud Manifesto.org

www.opencloudmanif… Google

list of supporters

Check out the latest version of the Cloud Computing Uses Cases White Paper - V4. It includes updates on SLAs!! Link

about

faqs

blogs, wikis, and more

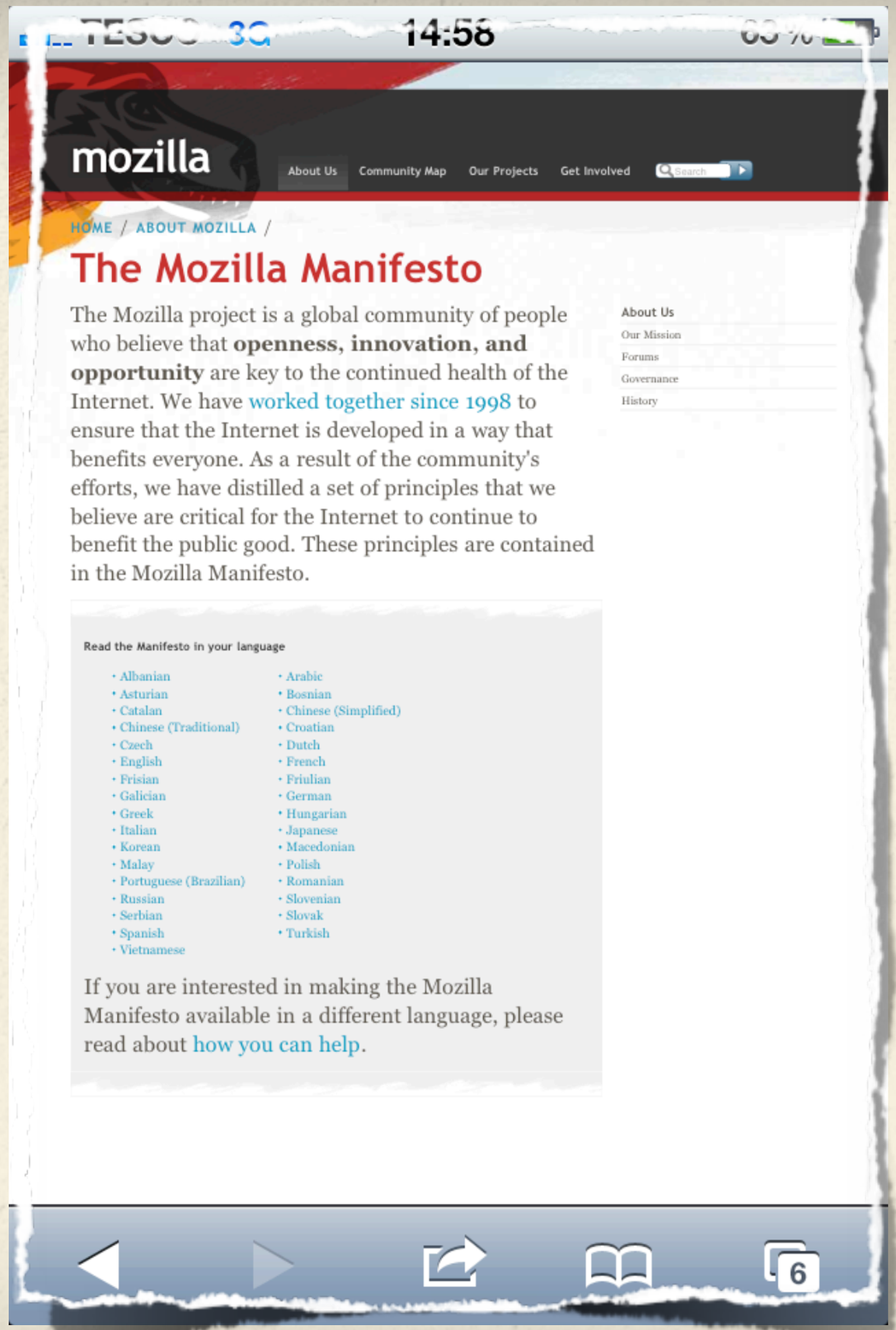
open cloud manifesto

dedicated to the belief that the cloud should be open

view the manifesto!

sign up to be added to the supporters list

6


14:58
mozilla
About Us
Community Map
Our Projects
Get Involved
Search
HOME / ABOUT MOZILLA /
The Mozilla Manifesto
The Mozilla project is a global community of people who believe that openness, innovation, and opportunity are key to the continued health of the Internet. We have worked together since 1998 to ensure that the Internet is developed in a way that benefits everyone. As a result of the community's efforts, we have distilled a set of principles that we believe are critical for the Internet to continue to benefit the public good. These principles are contained in the Mozilla Manifesto.
About Us
Our Mission
Forums
Governance
History
Read the Manifesto in your language
• Albanian
• Arabic
• Asturian
• Bosnian
• Catalan
• Chinese (Simplified)
• Chinese (Traditional)
• Croatian
• Czech
• Dutch
• English
• French
• Frisian
• Friulian
• Galician
• German
• Greek
• Hungarian
• Italian
• Japanese
• Korean
• Macedonian
• Malay
• Polish
• Portuguese (Brazilian)
• Romanian
• Russian
• Slovenian
• Serbian
• Slovak
• Spanish
• Turkish
• Vietnamese
If you are interested in making the Mozilla Manifesto available in a different language, please read about how you can help.
6

TESCO 3G 14:59

if you only have time for one clue this year, this is the one to get...

we are not seats or eyeballs or end users or consumers.
we are human beings - and our reach exceeds your grasp.
deal with it.

the cluetrain manifesto

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.

1. **Markets a**
2. **Markets c demograp**
3. **Conversat human. Th voice.**
4. **Whether c perspectiv humorous typically o**
5. **People rec the sound**
6. **The Intern among hur possible in**
7. **Hyperlinks**
8. **In both *int* *intra*netwo speaking to**

TESCO 3G 15:00

Alistair.Cockburn.us | User Manifesto

alistair.cockburn.us/U…

Google

Alistair Cockburn

User Manifesto

2010.01.07. SLC, UT.

…e Salt lake city roundtable today, we channeled Jeff Patton, …dly up in Minneapolis with some others trying to work out a …ner/user-centric extension to the Agile Manifesto. We decide… …e would come up with in about 20 minutes playing the role … …ying to get our wishes expressed.

…t have time to settle on the grammar, so there are three … …y the same words available. I don't think it matters a lot … sentiments are the same. You can vote on your prefe… …your view.

…duct teams to deliver products …herely "work" or fit a checklist, but **solve my pro…** …erely fit my needs at a moment, but **evolve as …**

…rk the way designers think I should work, but …rely see as tools, but **see through to the …** …ely use, but I **feel a joy in using.**

…ms to deliver products …**lems,** not merely that work or tha… …**eeds evolve,** not merely fit my …

TESCO 3G 15:01

experience design manifesto

people feel before things

Beta Version

Looking for a better life

The ultimate aim of all creative activity is to bring happiness to people's lives.

Happiness is an emotion that comes in result of positive experiences and affects human beings.

Experiences can happen in the past, present or future.

Past experiences must offer good memories as a way of re-living it in the present and desiring it for the future. Present experiences must connect to the mind in its whole, bringing focus to the action being performed and getting the mind to a flow state. Happiness comes from the experience itself and not from the result of it.

Future experiences must bring desire to the present, creating a present that is better because of the latent potential of a future experience.

Experiences can be based in real life or in fantasy, but reality and fantasy must empower one another in a way that the whole is better than the absolute sum of the separate experiences.

Experiences that foster happiness should have the following qualities:

- Make people feel confident of themselves.
- Make people feel they can do something

about me
know

my.blog
thoughts

why
understand

I have always thought that most design… pay little attention to the subjective side … what's known as User Experience. Ever… knows it's there, but they don't strive to … understand it, probably because it's har… measure and to bridge a concept to a m… tangible solution.

Being a true fan of this constant challenge … understanding the intangible, I thought … bringing together ideas to share with … everyone that thinks, reads, writes or talk… about this. That is the reason I have creat… a blog at first time.

Later, inspired by some talks, I had this id… of writing a tiny manifesto: something that … could point the way in which I believe design … should be done. Since we are dealing with … the way people feel, it is important we use … any knowledge we have in a good way.

Of course, when you write something and … name it "manifesto", the first reaction is to … think of it as something supposed to be … and judge it's content. That's great, becau… the clash of different visions can be very …

TESCO 3G 15:02

Hacker Manifesto - Wikipedia, the free encycl…

en.wikipedia.org/wiki/…

Google

Log in / create account

Article Discussion Read Edit View history Search

WIKIPEDIA The Free Encyclopedia

Main page Contents Featured content Current events Random article Donate to Wikipedia

Interaction Help About Wikipedia Community portal Recent changes Contact Wikipedia

Toolbox Print/export

Languages Afrikaans Azərbaycanca Česky Deutsch Français Italiano Português Română Русский Slovenčina Svenska Tiếng Việt Українська

Hacker Manifesto

From Wikipedia, the free encyclopedia

This article needs additional citations for verification. Please help improve this article by adding reliable references. Unsourced material may be challenged and removed. (January 2011)

The Conscience of a Hacker (also known as ***The Hacker Manifesto***) is a small essay written January 8, 1986 by a computer security hacker who went by the handle (or pseudonym) of The Mentor (born Loyd Blankenship). It was written after the author's arrest, and first published in the underground hacker ezine *Phrack* in Volume One, Issue 7, Phile 3 of 10. It can be found on many websites, as well as on t-shirts and in films.[1]

It is considered a cornerstone of hacker culture, and it gives some insight into the psychology of early hackers. It is said to have shaped the hacker community's view of itself and its motivations. *The Manifesto* states that hackers choose to hack because it is a way for them to learn, and because they are often frustrated and bored by the limitations of standard society. It also expresses the satori of a hacker realizing his potential in the realm of computers.

The *Manifesto* acts as a guideline to hackers across the globe, especially those new to the field. It serves as an ethical foundation for hacking, and asserts that there is a point to hacking that supersedes selfish desires to exploit or harm other people, and that technology should be used to expand our horizons and try to keep the world free. [edit]

In popular culture

The article is quoted in the 1995 movie *Hackers*, although in the movie it is being read from an issue of the hacker magazine 2600, not the historically accurate *Phrack*. Mentor receives attribution in the credits of the movie.

It is also reproduced inside the CD case of the computer game *Uplink*.

The Mentor gave a reading of *The Hacker Manifesto* and offered additional insight at H2K2.

It is also an item in the game Culpa Innata.

"A Hacker Manifesto" is also the name of a book written by media studies professor McKenzie Wark.

A poster that can be ordered from the Cafepress website appears in the movie The Social Network on the wall of Mark Zuckerberg's dorm room. [edit]

See also

- Phrack
- Timeline of computer security hacker history

References

1. ^ Thomas, Douglas (2003). *Hacker Culture*. University of Minnesota Press. pp. xxiv. ISBN 9780816633463.

External links

- The original article *The Conscience of a Hacker* on phrack.org archive
- The original article *The Conscience of a Hacker* on TextFiles.com Usenet Archive
- FIDOin interview with The Mentor (July 31, 2000)

Computer security hacking
History
Phreaking
Cryptovirology
Hacker ethic
Hacker Manifesto, Black hat, Grey hat, White hat, Black Hat Briefings, DEF CON
Cybercrime
Computer crime, Crimeware
List of convicted computer criminals
Script kiddie
Hacking tools
Vulnerability
Exploit
Payload
Malware
Rootkit, Backdoor
Trojan horse, Virus, Worm
Spyware, Botnet, Keystroke logging
Antivirus software, Firewall, HIDS
Computer security
Computer insecurity
Application security
Network security

6

politics

I have the solution

Manifesto<PET_SUBJECT>

cheek.insert(tongue)

MANIFESTO#1

we believe in
a fixed set of immutable ideals

over
tailoring our approach to each specific situation.

MANIFESTO#2

we believe in
concentrating on and discussing only the things that interest us

over
the bigger problem.

MANIFESTO#3

we believe in
our own opinion

over
the opinions and experiences of others.

MANIFESTO#4

we believe in
arbitrary black-and-white mandates

over
real-world scenarios with complex issues
and delicate resolutions.

MANIFESTO#5

we believe that
when our approach is hard to follow

then
it only shows how much more
important it is.

MANIFESTO#6

we believe in
crafting an arbitrary set of commandments

over
the realisation that it's just never
that simple.

MANIFESTO#7

we believe in
trying to establish a movement to promote
our view
over
something that will be genuinely useful.

MANIFESTO#8

we believe that

we are better developers than those who don't agree with us

because

they don't agree with us.

MANIFESTO

that is,

we believe we're doing the right thing.

And if you don't, you're wrong.

And if you don't do what we do, you're doing it wrong.

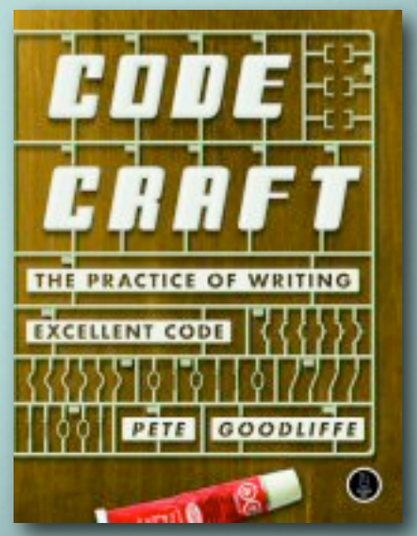
(thankyou)

Q & A

Pete Goodliffe @petegoodliffe pete@goodliffe.net

KEEP CALM AND CARRY ON

@petegoodliffe
pete@goodliffe.net
goodliffe.blogspot.com
www.goodliffe.net



BUMPH DULL, but important

THIS DOCUMENT WAS CREATED BY PETE GOODLIFFE

IT IS COPYRIGHT //

>>

>> ALL THOUGHTS ARE OWNED

>> PHOTOS AND IMAGES ARE MOSTLY SOURCED FROM THE WEB

THANK YOU FOR READING // I HOPE IT WAS USEFUL

Version 1.0 2011-04-14

Lightning Talks

ACCU 2011

Olve Maudal – Technical debt is good!

Allan Kelly – Dialogue Sheet Retrospectives

Martin Winkler – There Is No Base

Didier Verna – Letting Go of Control

Tom Gilb – User Stories: Why They Might Be Too Light

Chris Oldwood – The Ups and Downs of Being an ACCU Member

Pete Goodliffe – Manyfestos

Peter Pilgrim – We Are Going To Live Forever!

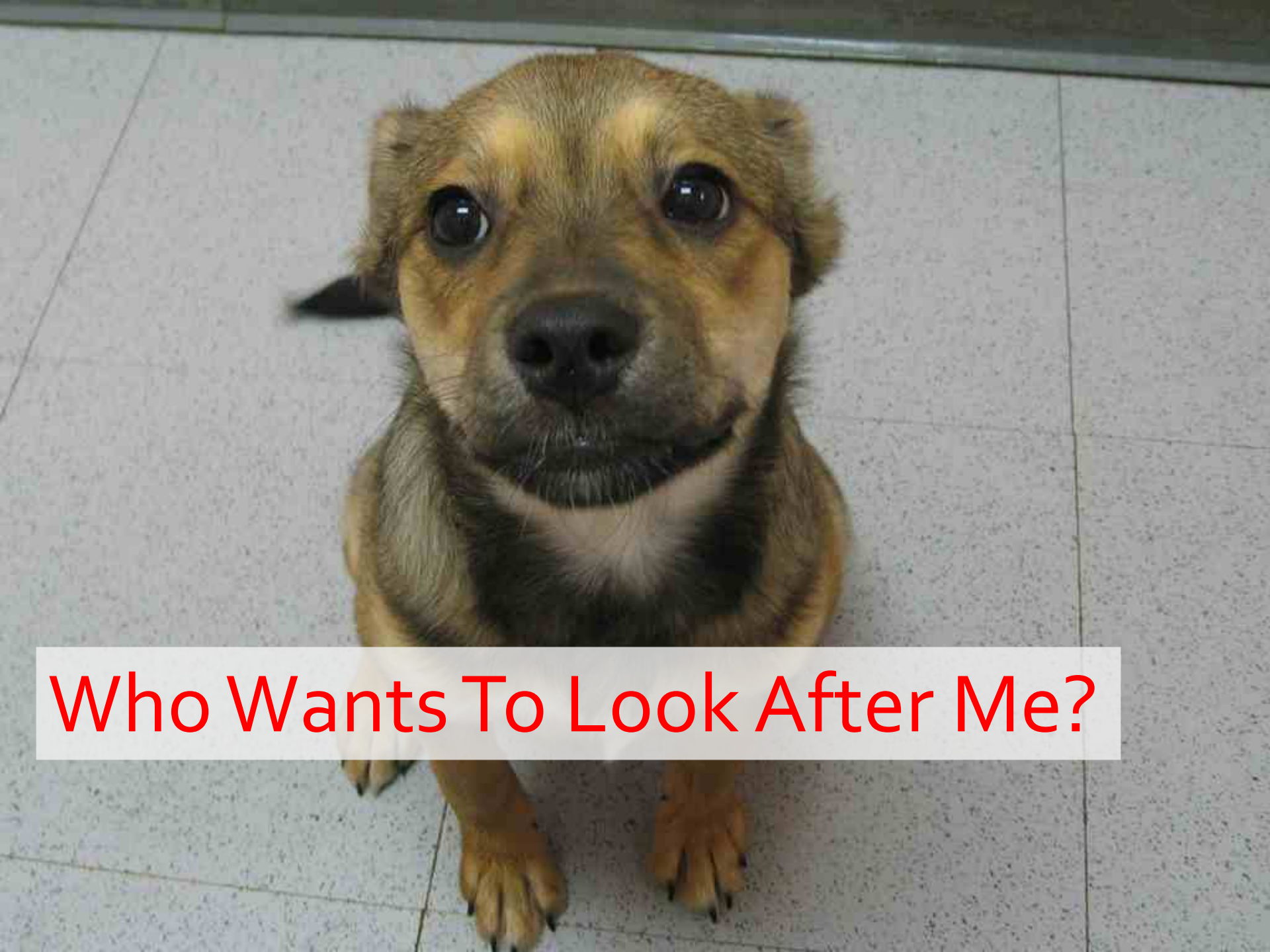
Roger Orr – Errors Are Evil

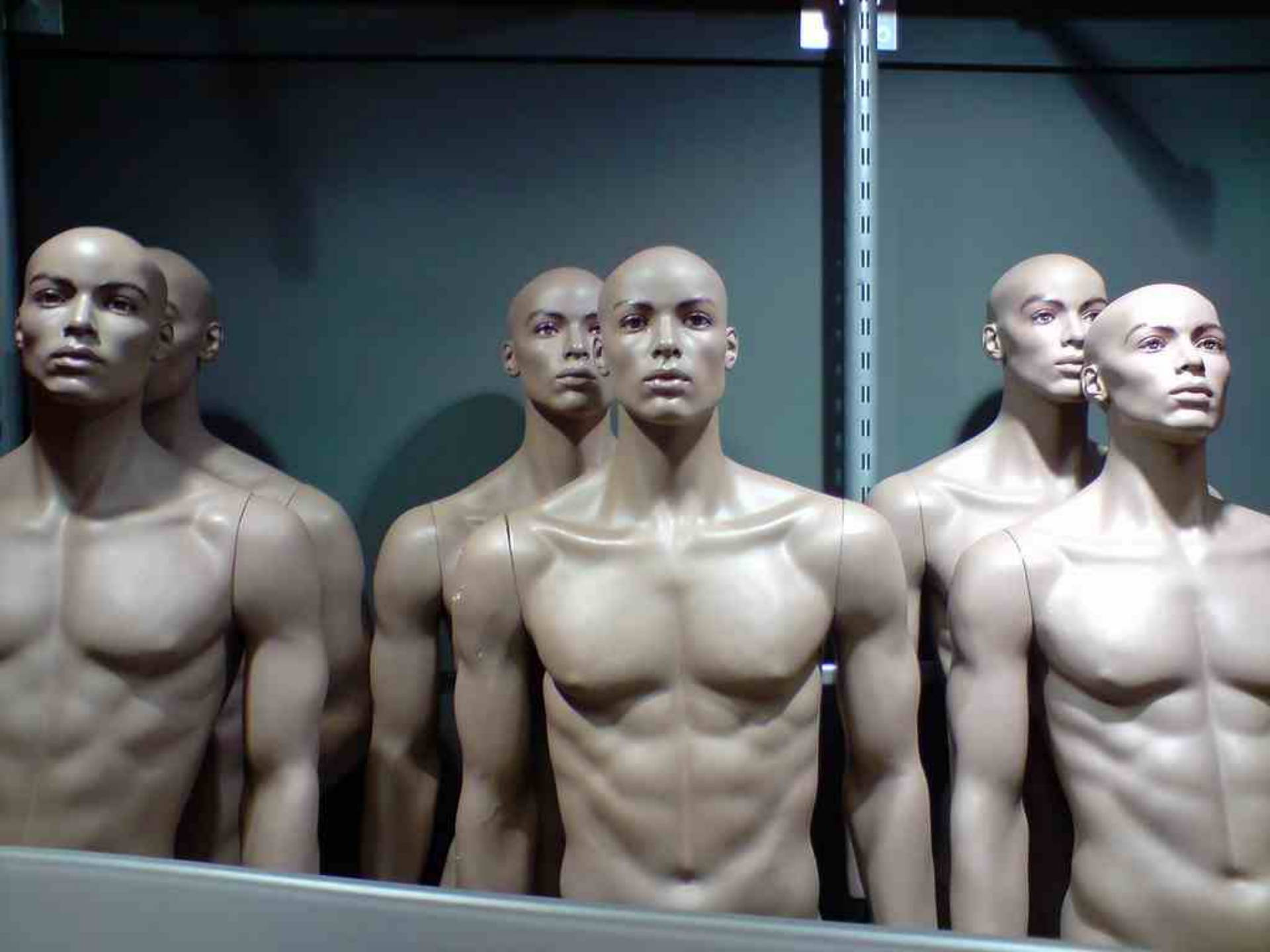
Mark Dalgarno – Optimizing For Unhappiness

Peter Pilgrim

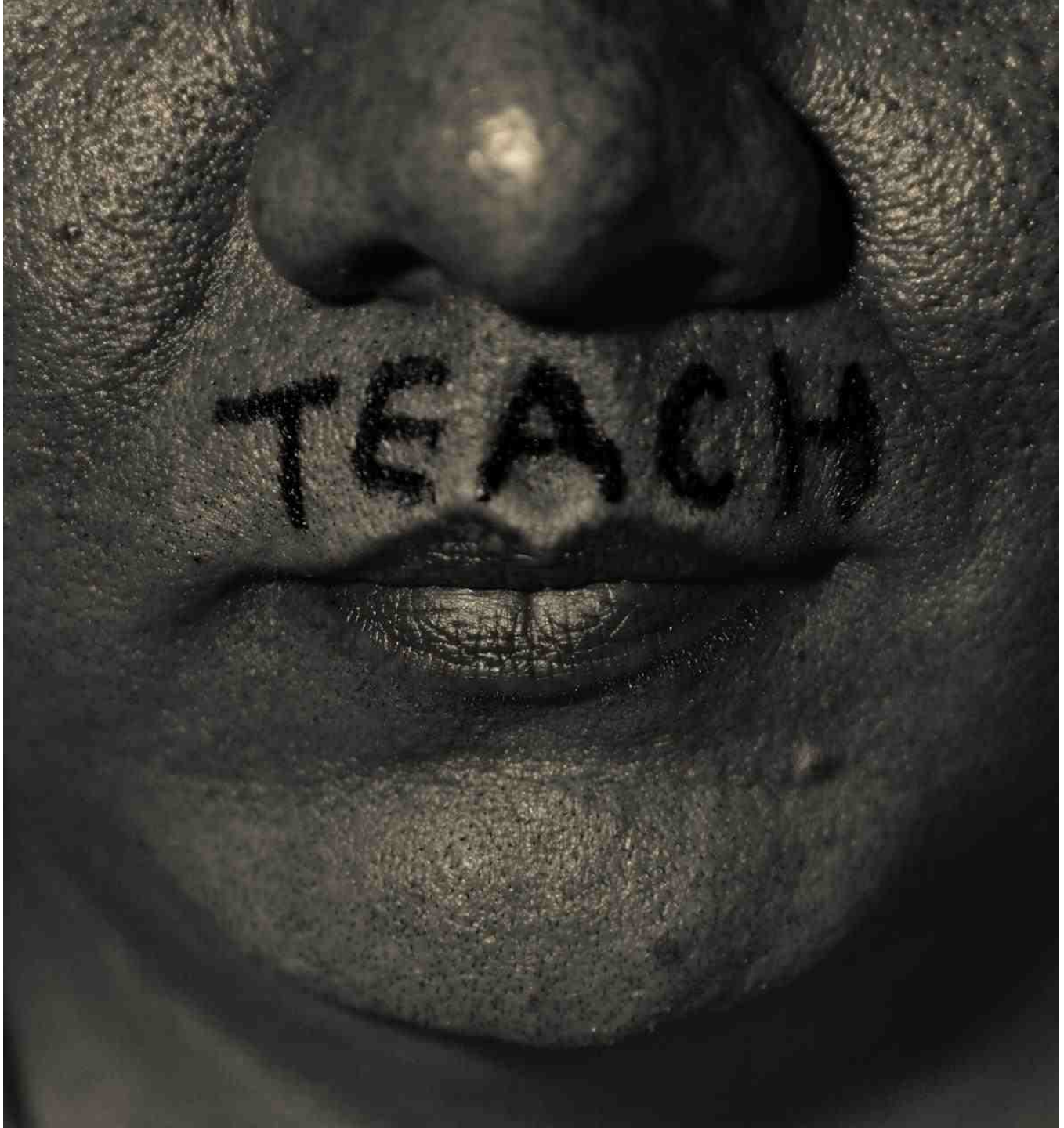
Java Champion, Scala Enthusiast

Are We Going To Live Forever?


Who Wants To Look After Me?




CAPTAIN


TEACH

Creative Commons Attributions

- Bullcitydogs a volunteer at APS of Durham http://www.flickr.com/photos/bullcitydogs/3236816175/sizes/l/
- Justageek http://www.flickr.com/photos/justageek/2300396608/
- Otsebmi http://www.flickr.com/photos/otsebmi/2848983818/
- Woicik from the Boogie down bronx http://www.flickr.com/photos/woicik/4358333258/

Lightning Talks

Olve Maudal – Technical debt is good!

Allan Kelly – Dialogue Sheet Retrospectives

Martin Winkler – There Is No Base

Didier Verna – Letting Go of Control

Tom Gilb – User Stories: Why They Might Be Too Light

Chris Oldwood – The Ups and Downs of Being an ACCU Member

Pete Goodliffe – Manyfestos

Peter Pilgrim – We Are Going To Live Forever!

Roger Orr – Errors Are Evil

Mark Dalgarno – Optimizing For Unhappiness

Lightning Talks

Olve Maudal – Technical debt is good!

Allan Kelly – Dialogue Sheet Retrospectives

Martin Winkler – There Is No Base

Didier Verna – Letting Go of Control

Tom Gilb – User Stories: Why They Might Be Too Light

Chris Oldwood – The Ups and Downs of Being an ACCU Member

Pete Goodliffe – Manyfestos

Peter Pilgrim – We Are Going To Live Forever!

Roger Orr – Errors Are Evil

Mark Dalgarno – Optimizing For Unhappiness

Optimizing for Unhappiness
- Tips for cost-conscious managers

Mark Dalgarno
@MarkDalgarno

Tip #1

You can never have enough CCTV cameras

Make sure your staff aren't wasting their time by monitoring them closely

Are they loitering in so-called *water-cooler chats*?

Are they taking too long in the toilet?

Tip #2

Fire your cleaners

Having cleaners is like saying you're happy with untidiness.

If your staff have to do their own cleaning, they'll naturally keep the place tidy

Tip #3

Require everyone to wear a suit to work

It will make your office look smarter after you've fired your cleaners.

The well-dressed developer is the productive developer.

Tip #4

Give Middle-Managers Preferential Treatment

Managers are your enforcers in the battle for cost-reduction, give them special treatment and they'll work twice as hard to enforce your rules.

Tip #5

Be vigilant!

Subversive organisations such as ACCU are saying there is a better way

Watch out for any of the following terms being used by your staff

Danger Words!!!

- Conference / Training
- Agile Software Development / Scrum / XP
- Self-organizing teams
- Technical Debt
- Clean code
- Test-Driven Development
- Steve Freeman / Nat Pryce

Lightning Talks

ACCU 2011

See you tomorrow

same time, same place