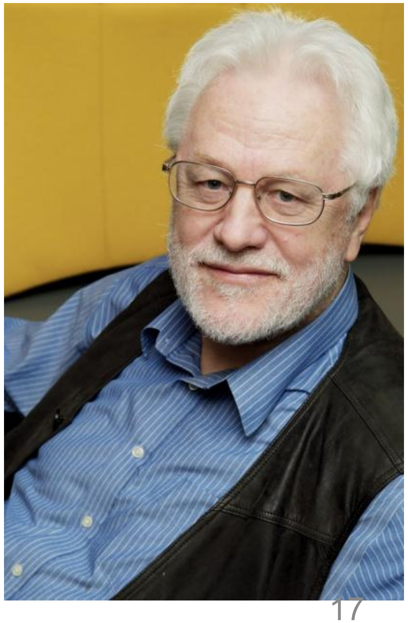
Chris Oldwood's Challenge: (11 April, ACCU)

"Can you quantify 'software robustness' ?"

15-Minute Lightning Talk
ACCU Bristol
© Friday 12 April 2013,
18:00 session
by **Tom Gilb**



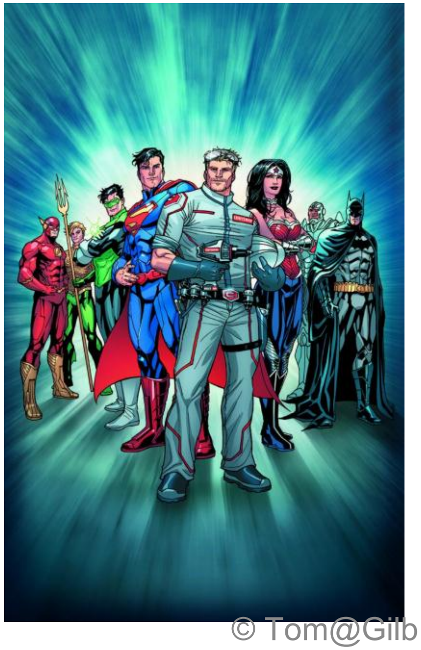
Are we engineering software yet, or still ‘softcrafting’?

‘Softcrafters’

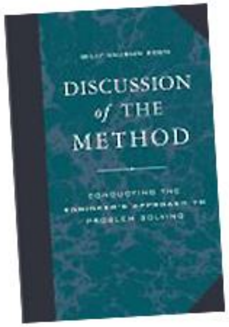
Alliance of Code Craftspeople United

A ‘Softcrafter’ is a person who practices the craft of programming software for computers

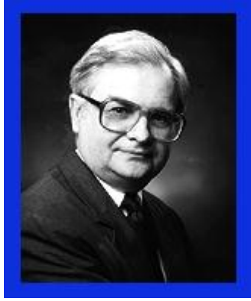
(Gilb, PoSEM, 1988)



- This type of person is better known as a ‘*programmer*’ (or even a ‘*developer*’)
- Sometimes they call themselves *software engineers*
  - without any engineering competence or qualifications
  - an <u>illegal</u> act, in some places (TX, CAN)
- This is rather like a good *carpenter*, calling himself a *structural engineer*, or an *architect*



Billy Koen’s Definition of ‘***Engineering***’



"Engineering is a risk-taking activity.

To control these risks, engineers have many heuristics:

1. **They make only small changes in what has worked in the past,**
2. **They try to arrange matters so that, if they are wrong, they can retreat, and**
3. **They feed back past results in order to improve future performance.”**

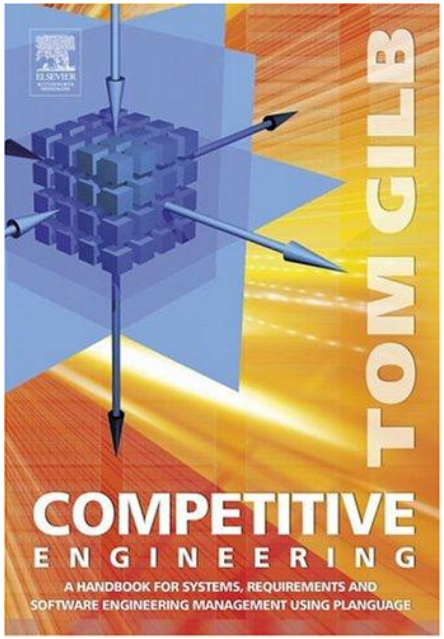
- **"Engineers cannot simply work their way down a list of steps, ... but ...**
- **they must circulate freely within the proposed plan ...”**

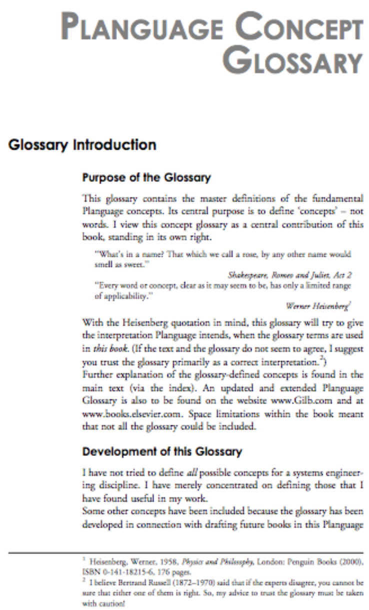
http://www.dce.kemu.edu.tw/~thinle/congtackyu_2008 Engineering_Method.pdf
http://online.engineering.illinois.edu/webcourses/seminars/ETC/notes/01-24-07.pdf

‘Engineering’ is

- an **evolutionary** process
- using *practical* **principles**
- in order to determine and identify the **means** to deliver
- the best-achievable balance of **Performance and Cost levels**
- for optimal **stakeholder** satisfaction
- in a complex, **risk-filled** environment

Source: Planguage Glossary in CE Book, 2005


ELSEVIER
TOM GILB
COMPETITIVE
ENGINEERING
A HANDBOOK FOR SYSTEMS, REQUIREMENTS AND SOFTWARE ENGINEERING MANAGEMENT USING PLANGUAGE



Planguage Concept Glossary

Glossary Introduction

Purpose of the Glossary

This glossary contains the master definitions of the fundamental Planguage concepts. Its central purpose is to define ‘concepts’ – not words. I view this concept glossary as a central contribution of this book, standing in its own right.

> “What’s in a name? That which we call a rose, by any other name would smell as sweet.”
>
> *Shakespeare, Romeo and Juliet, Act 2*
>
> “Every word or concept, clear as it may seem to be, has only a limited range of applicability.”
>
> *Werner Heisenberg*[1]

With the Heisenberg quotation in mind, this glossary will try to give the interpretation Planguage intends, when the glossary terms are used in *this book*. (If the text and the glossary do not seem to agree, I suggest you trust the glossary primarily as a correct interpretation.[2])

Further explanation of the glossary-defined concepts is found in the main text (via the index). An updated and extended Planguage Glossary is also to be found on the website www.Gilb.com and at www.books.elsevier.com. Space limitations within the book meant that not all the glossary could be included.

Development of this Glossary

I have not tried to define *all* possible concepts for a systems engineering discipline. I have merely concentrated on defining those that I have found useful in my work.

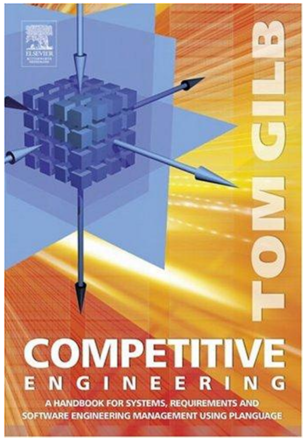
Some other concepts have been included because the glossary has been developed in connection with drafting future books in this Planguage

[1] Heisenberg, Werner, 1958, *Physics and Philosophy*, London: Penguin Books (2000), ISBN 0-141-18215-6, 176 pages.

[2] I believe Bertrand Russell (1872–1970) said that if the experts disagree, you cannot be sure that either one of them is right. So, my advice to trust the glossary must be taken with caution!

Planguage Concept Glossary as edited in Competitive Engineering book 2005
http://www.gilb.com/tiki-download_file.php?fileId=387
Full Glossary http://www.gilb.com/tiki-download_file.php?fileId=386

‘Engineering’ is


ELSEVIER
TOM GILB
COMPETITIVE
ENGINEERING
A HANDBOOK FOR SYSTEMS, REQUIREMENTS AND SOFTWARE ENGINEERING MANAGEMENT USING PLANGUAGE

- an evolutionary process

Robustness

(and other qualities)

s

d identify

- the best ievable balance of

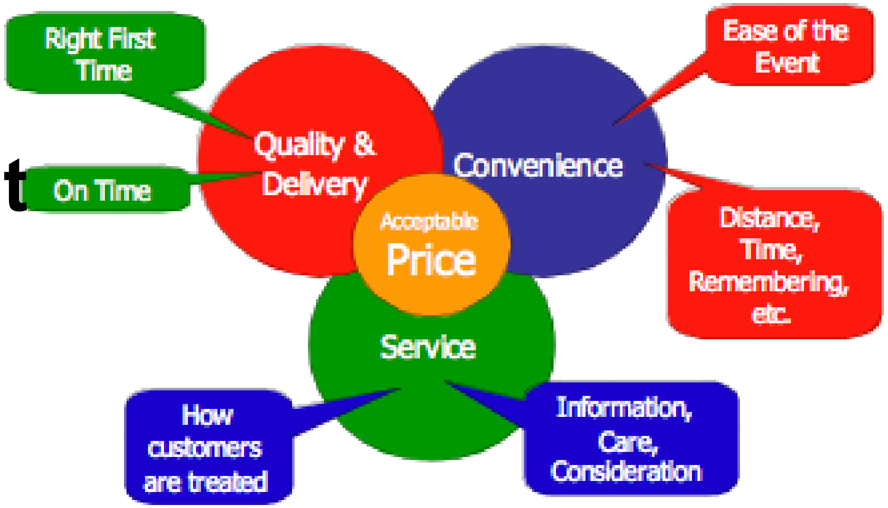
  Performance and Cost levels
- for optimal stakeholder satisfaction
- in a complex, risk-filled environment

Source: Planguage Glossary in CE Book

‘Software Engineering’ is (IMHO)

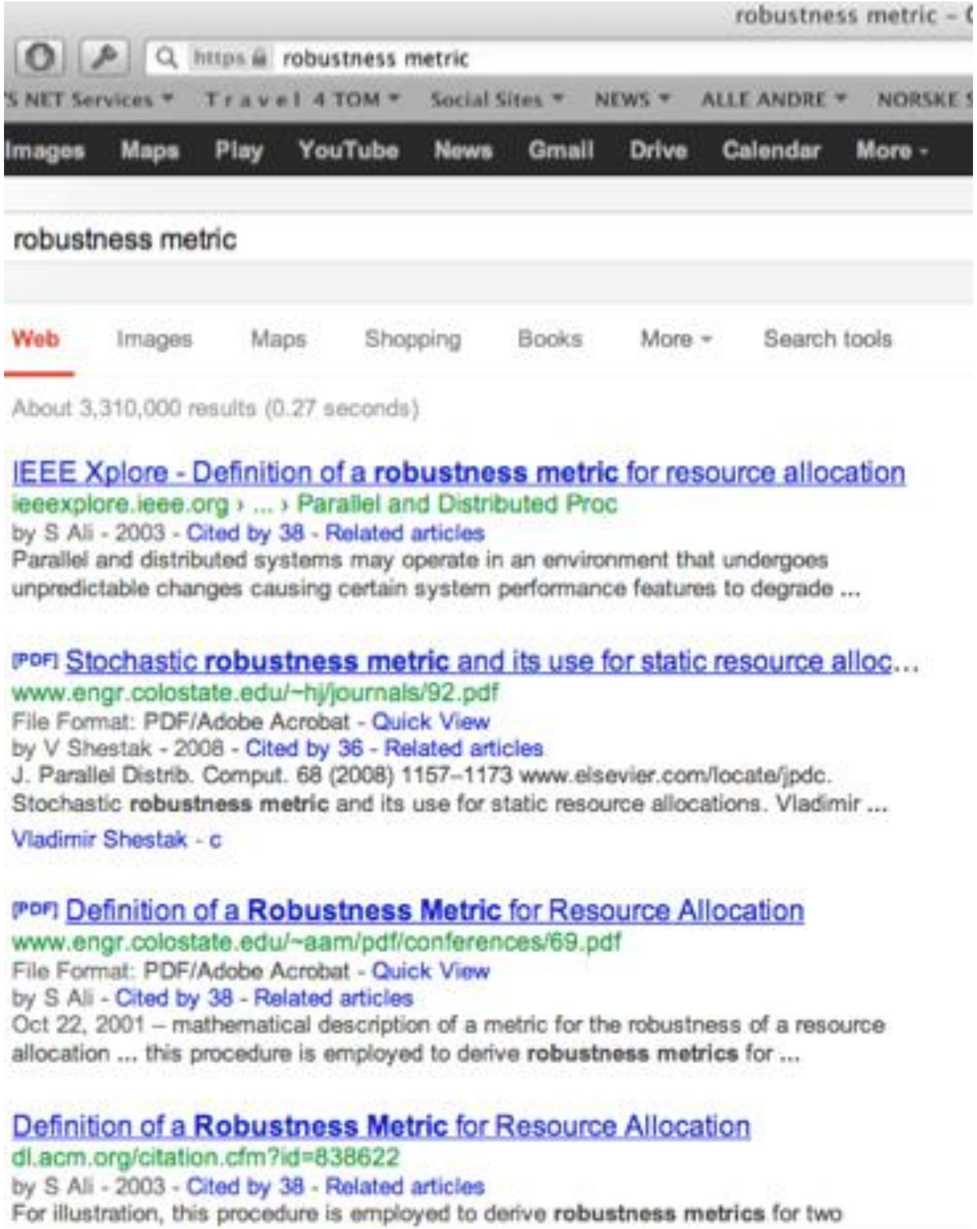
- the engineering discipline
- of enabling and motivating software systems
- to deliver a balanced set of values,
  - directly or indirectly,
- to a balanced set stakeholders,
- throughout their lifecycle*.

Value:
Extending the concept



- thanks for Ian Sommerville and Frans Ver Schoor for inspiring this 2010 PL revision http://se9book.wordpress.com/2010/03/23/semat-and-the-definition-of-software-engineering/

Maybe some hope? Others are working on how to quantify "Robustness"



About 3,310,000 results (0.27 seconds)

IEEE Xplore - Definition of a **robustness metric** for resource allocation
ieeexplore.ieee.org › ... › Parallel and Distributed Proc
by S Ali - 2003 - Cited by 38 - Related articles
Parallel and distributed systems may operate in an environment that undergoes unpredictable changes causing certain system performance features to degrade ...

[PDF] Stochastic **robustness metric** and its use for static resource alloc...
www.engr.colostate.edu/~hj/journals/92.pdf
File Format: PDF/Adobe Acrobat - Quick View
by V Shestak - 2008 - Cited by 36 - Related articles
J. Parallel Distrib. Comput. 68 (2008) 1157–1173 www.elsevier.com/locate/jpdc. Stochastic **robustness metric** and its use for static resource allocations. Vladimir ...
Vladimir Shestak - c

[PDF] Definition of a **Robustness Metric** for Resource Allocation
www.engr.colostate.edu/~aam/pdf/conferences/69.pdf
File Format: PDF/Adobe Acrobat - Quick View
by S Ali - Cited by 38 - Related articles
Oct 22, 2001 – mathematical description of a metric for the robustness of a resource allocation ... this procedure is employed to derive **robustness metrics** for ...

Definition of a **Robustness Metric** for Resource Allocation
dl.acm.org/citation.cfm?id=838622
by S Ali - 2003 - Cited by 38 - Related articles
For illustration, this procedure is employed to derive **robustness metrics** for two

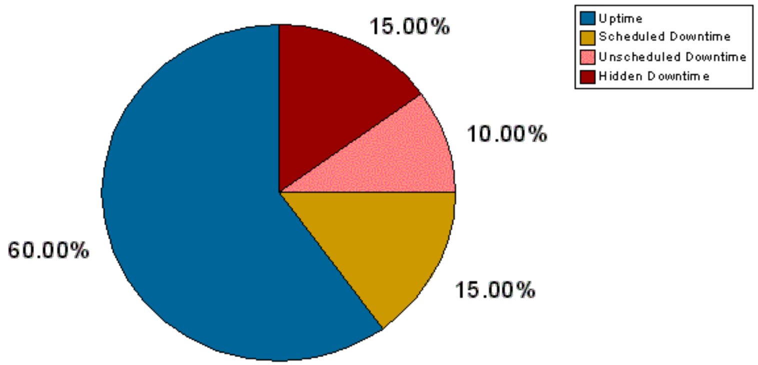
Rock Solid Robustness: “many splendored”

Type: Complex Product Quality Requirement.

Includes:

{Software Downtime,
Restore Speed,
Testability,
Fault Prevention Capability,
Fault Isolation Capability,
Fault Analysis Capability,
Hardware Debugging Capability}.

Software Downtime:



Type: Software Quality Requirement.
Version: 25 October 2007.
Part of: Rock Solid Robustness.
Ambition: To have minimal downtime due to software failures <- HFA 6.1.
Issue: Does this not imply that there is a system wide downtime requirement?

Scale: <Mean time between forced restarts for defined [Activity] for a defined [Intensity].>

Fail [Any Release or Evo Step, Activity = Recompute, Intensity = Peak Level]: 14 days <- HFA 6.1.1.

Goal [By 2008?, Activity = Data Acquisition, Intensity = Lowest level]: 300 days ??

Stretch: 600 days.

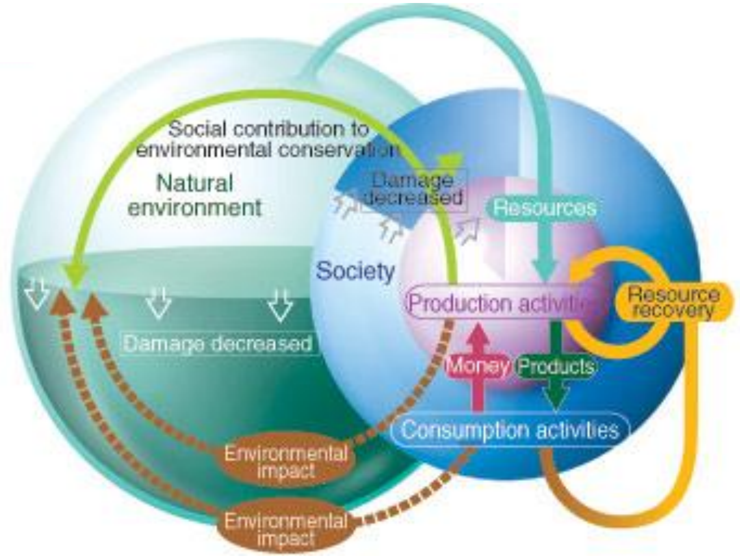
Restore Speed:

Type: Software Quality Requirement.

Version: 25 October 2007.

Part of: Rock Solid Robustness.

Ambition: Should an error occur (or the user otherwise desire tc do so), the system shall be able to restore the system to a previously saved state in less than 10 minutes. <-6.1.2 HFA.



Scale: Duration from Initiation of restore to complete and verified state of a defined [Previous: Default = Immediately Previous] saved state.

Initiation: defined as {Operator Initiation, System Initiation, ?}. Default = Any.

Goal [Initial and all subsequent released and Evo steps]: 1 minute?

Fail [Initial and all subsequent released and Evo steps]: 10 minutes. <- 6.1.2 HFA.

Catastrophe: 100 minutes.

Testability:

Type: Software Quality Requirement.

Part of: Rock Solid Robustness.

Initial Version: 20 Oct 2006.

Version: 25 October 2007.

Status: Demo draft.

Stakeholder: {Operator, Tester}.

Ambition: Rapid-duration automatic testing of <critical complex tests> with extreme operator setup and initiation.

Scale: The duration of a defined [Volume] of testing, or a defined [Type] by a defined [Skill Level] of system operator under defined [Operating Conditions].

Goal [All Customer Use, Volume = 1,000,000 data items,
Type = WireXXXX Vs DXX, Skill = First Time Novice,
Operating Conditions = Field, {Sea Or Desert}]: < 10 minutes.

***Design Hypothesis**: Tool Simulators, Reverse Cracking Tool, Generation of simulated telemetry frames entirely in software, Application specific sophistication, for drilling – recorded mode simulation by playing back the dump file, Application test harness console <-6.2.1 HFA.*

Software Engineer

- **A software *engineer* is**
  - **–an engineer**
  - **– with a specialty in software**

- **They are characterized by the ability to assemble software components based on *quantified* attributes.**
- **This ability is aimed at the need to <u>meet multiple quantified requirement performance levels, within specified resource constraints,</u> and other constraint limitations.**
- **Consequently software engineers think in terms of**
  - ***measurable* system performance (including quality) characteristics,**
  - **and *costs for design*, implementation, decommissioning, adaptation, and operation.**
- **They know how to**
  - ***estimate* the multiple quantified attributes of a design component**
  - **and how to *measure* these attributes in the systems they engineer.**

Think of your ‘Future’ !

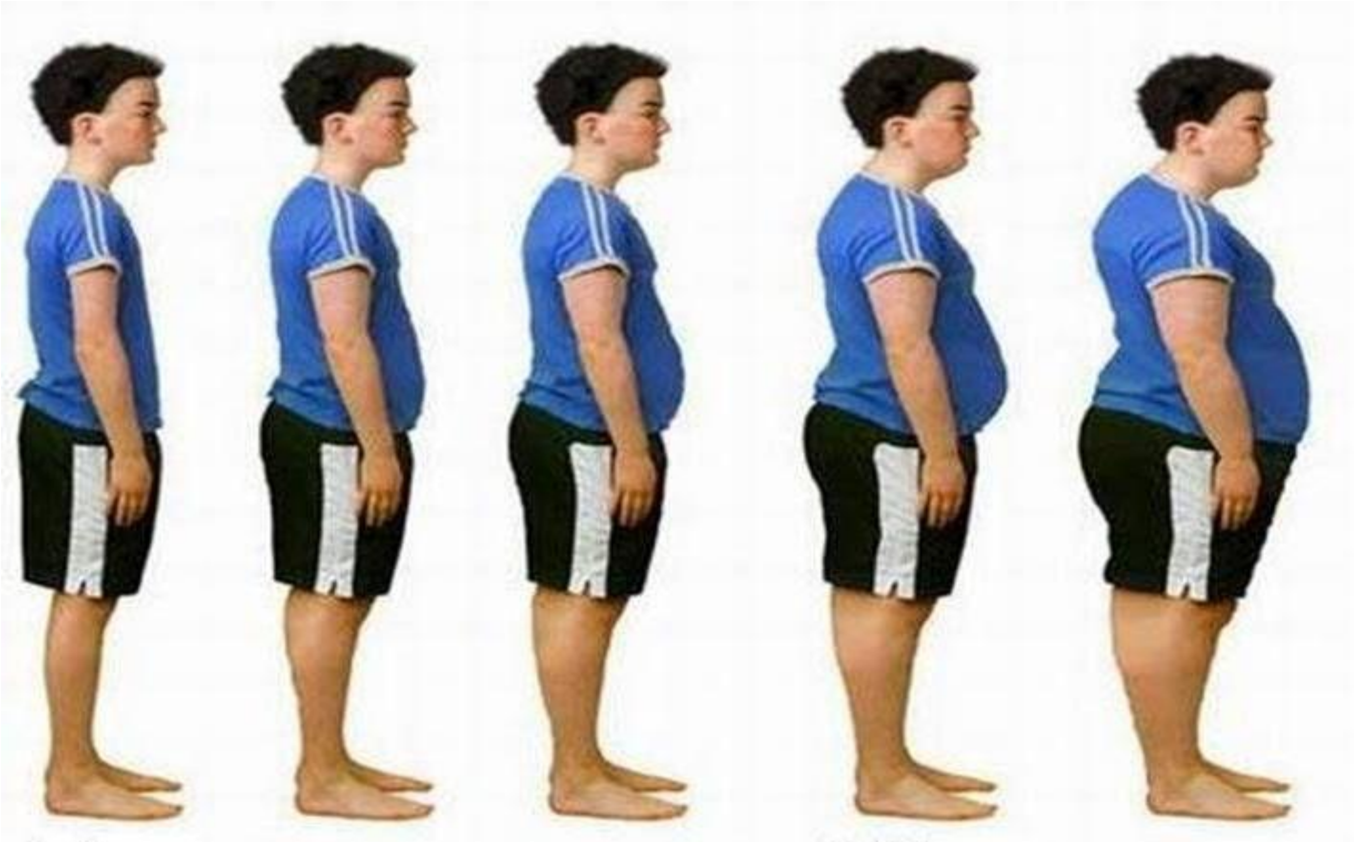
Hacker

Developer

Software Engineer

Software Architect

Project Manager

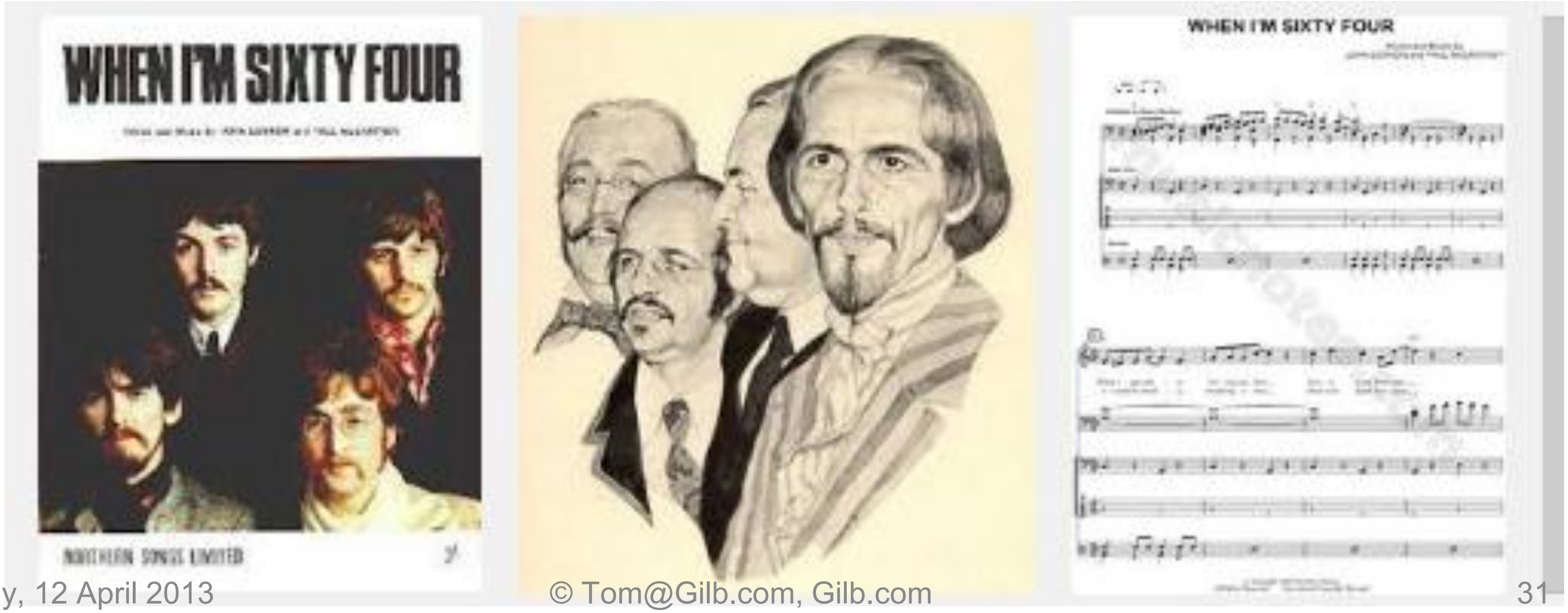


.

.

.

Would you prefer to be a Softcrafter, until you’re 64 or would you be able to advance to being a REAL ‘Software Engineer’?



http://www.gilb.com/dl171

Designing **Maintainability** in Software Engineering: a *Quantified* Approach.

Tom Gilb

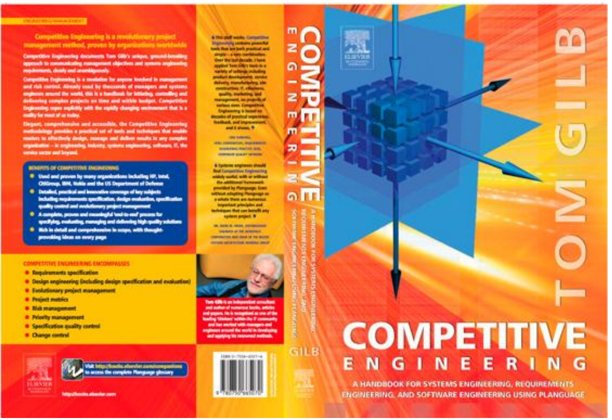
Result Planning Limited
Tom@Gilb.com

these slides at Gilb.com/downloads slides

For ACCU Oxford UK

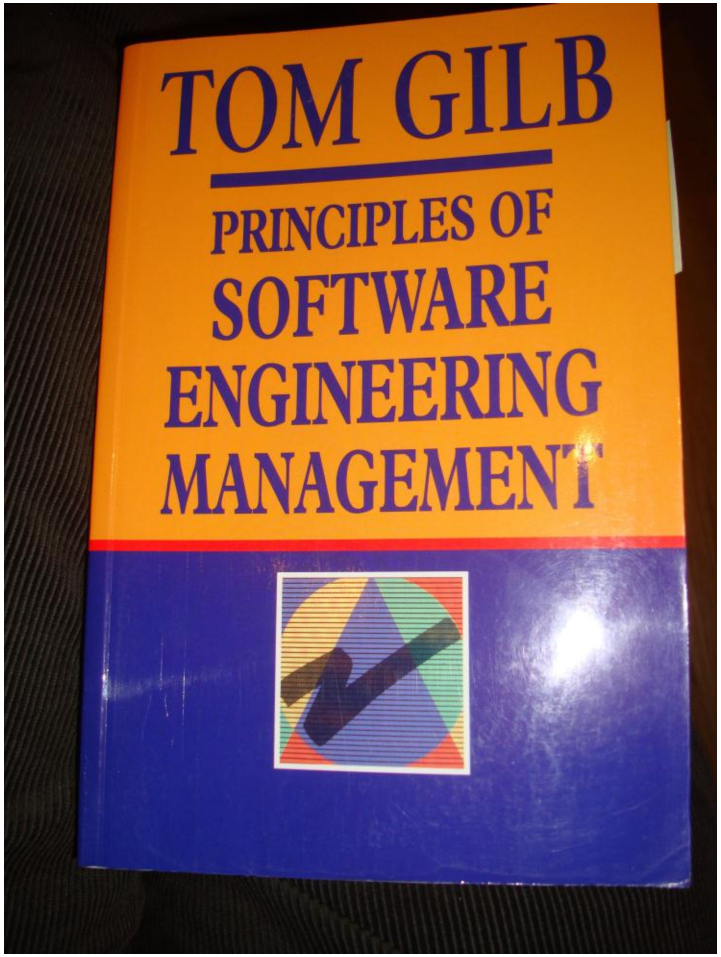
Friday 4th April 2008

1400 90 MInutes

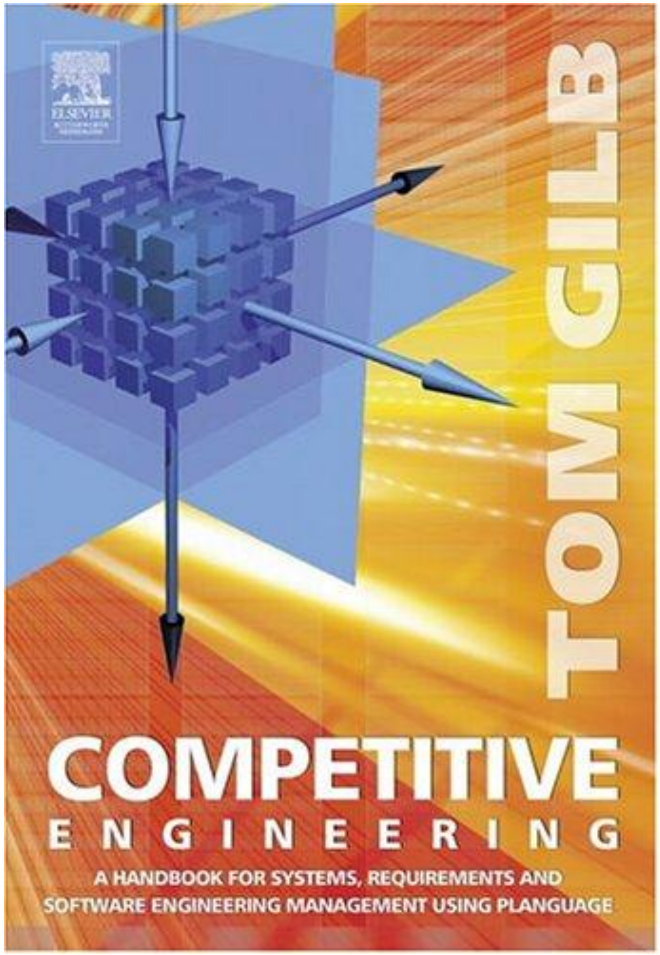

COMPETITIVE ENGINEERING
TOM GILB
A HANDBOOK FOR SYSTEMS ENGINEERING, REQUIREMENTS ENGINEERING, AND SOFTWARE ENGINEERING USING PLANGUAGE
GILB

End.

......Or just a beginning for you?



1988
Ask me for Chapters on ‘Productivity’ or
“Perspectives on Evolutionary Delivery”


ELSEVIER
TOM GILB
COMPETITIVE ENGINEERING
A HANDBOOK FOR SYSTEMS, REQUIREMENTS AND SOFTWARE ENGINEERING MANAGEMENT USING PLANGUAGE

2005
Free Sample Chapters Gilb.com
Ask me for full digital copy free
(tom@gilb.com)

And now, if 5 minutes left

- As advertised yesterday
- As a possible option

- **User Stories Bashing**

- **Ok**
- **Comments on overgeneralisations about user stories**

The Addison-Wesley Signature Series
User Stories Applied
For Agile Software Development
Mike Cohn
Foreword by Kent Beck

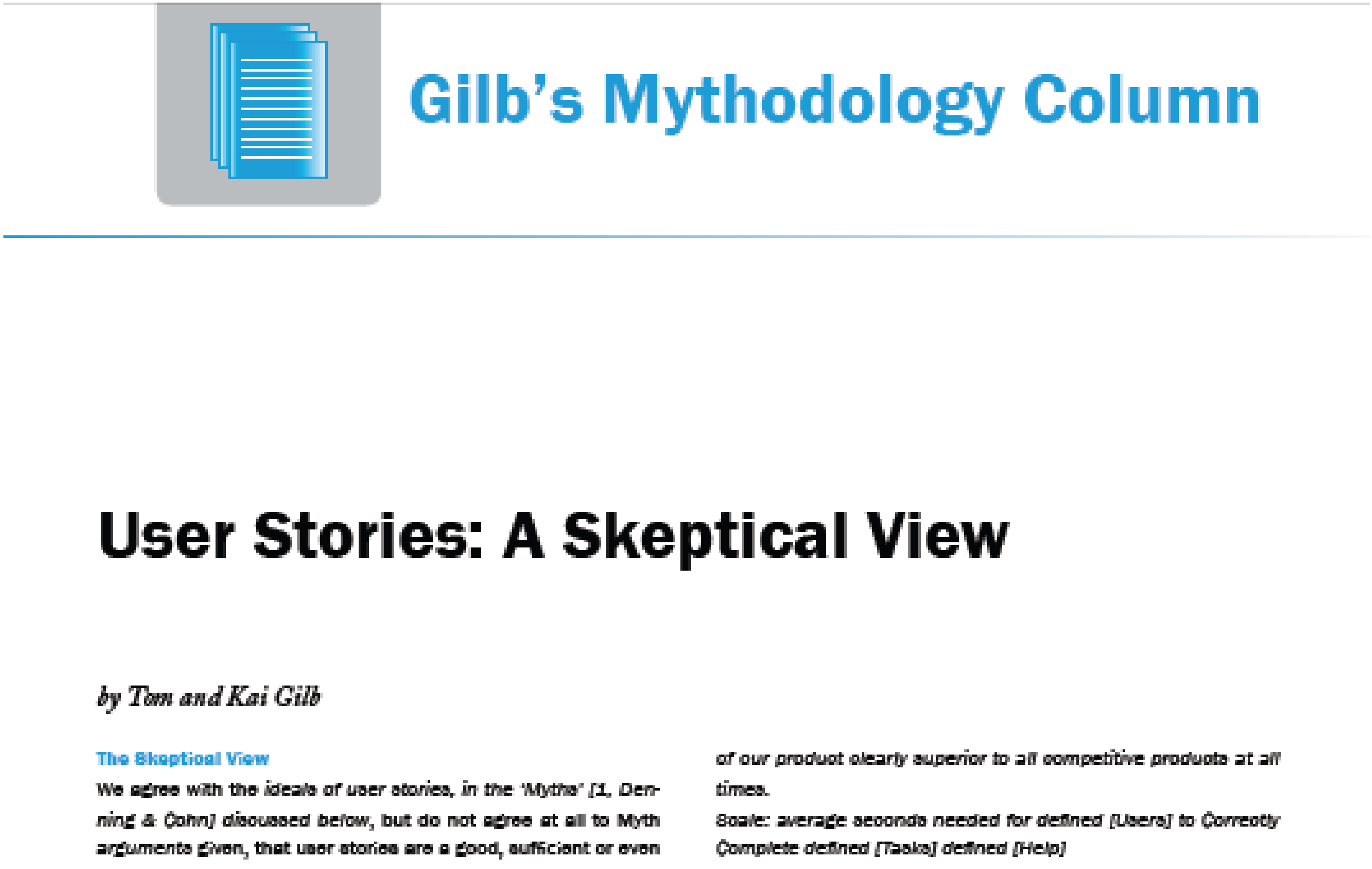
User Stories: why they might be too light for your complex purposes

by Tom @ Gilb . com

5 Minute Lightening Talk
ACCU Bristol
Friday 12 April 2013, 18:00 session
If time, inside my 15 minutes.
Otherwise this will be on Gilb.com/downloads slides

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

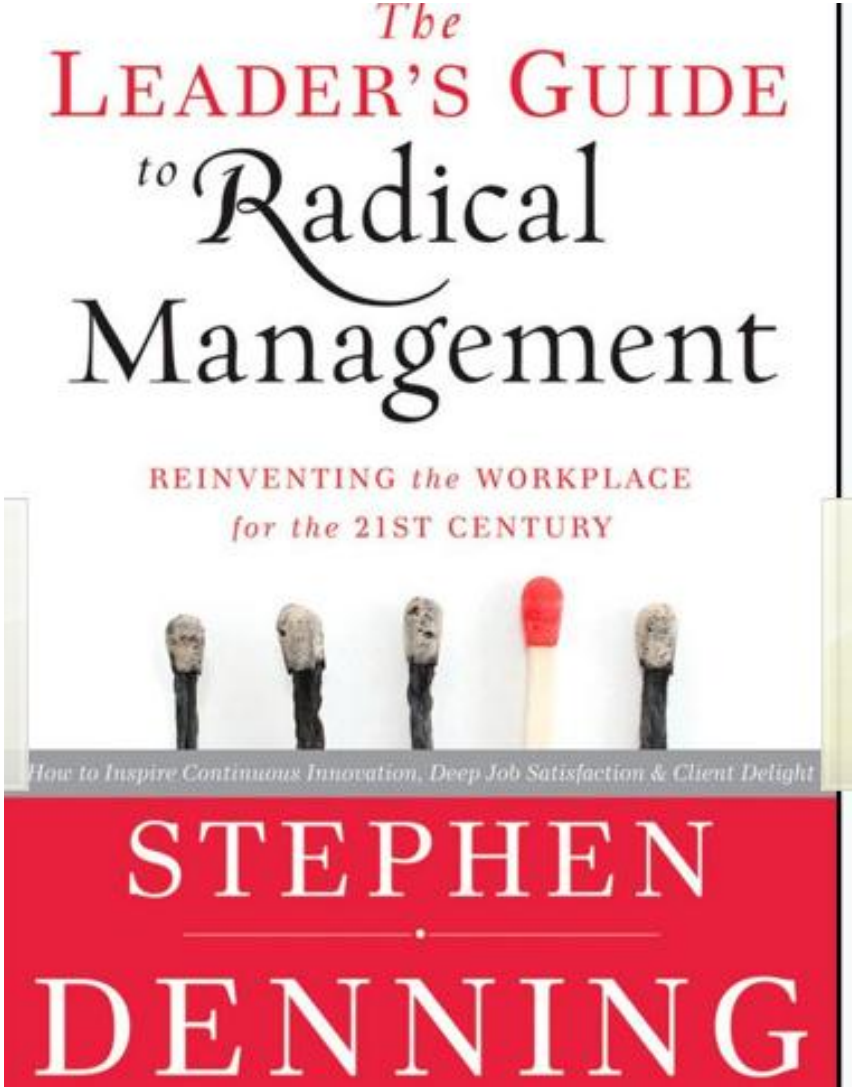
The Skeptical View

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

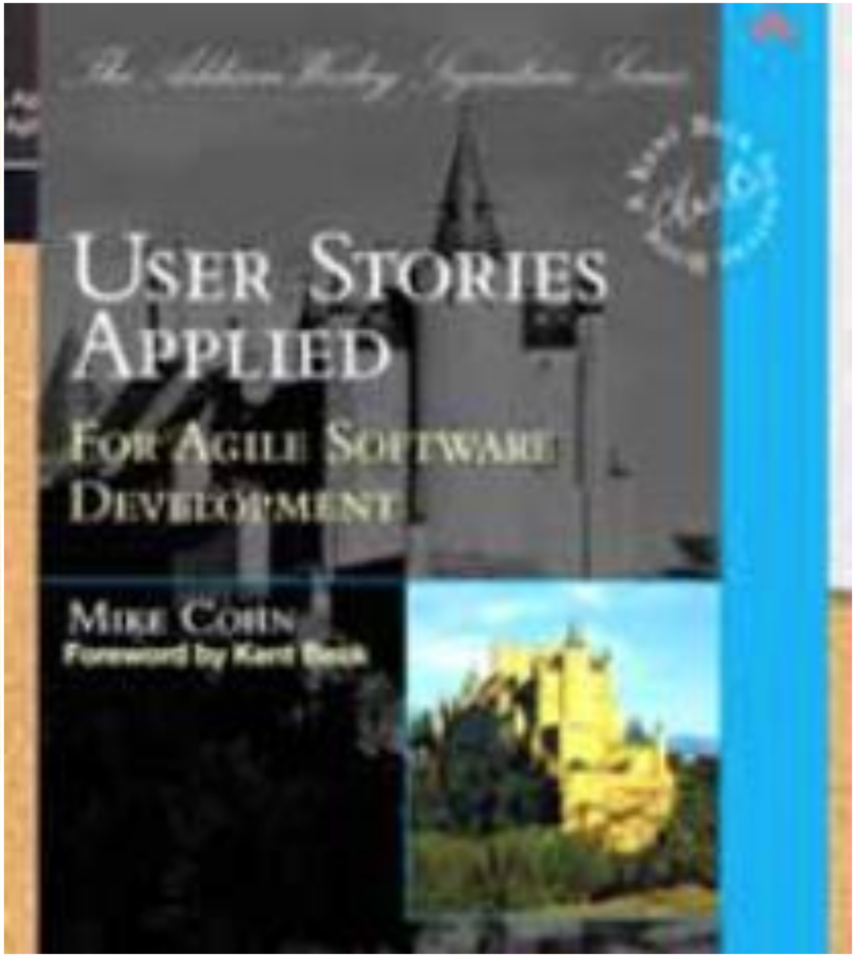

The Leader's Guide to Radical Management
REINVENTING the WORKPLACE for the 21ST CENTURY
How to Inspire Continuous Innovation, Deep Job Satisfaction & Client Delight
STEPHEN DENNING

http://stevedenning.typepad.com/

From Mike Cohns User Stories Work

Mike Cohn
mike@mountaingoatsoftware.com
www.mountaingoatsoftware.com
(720) 890-6110 (office)
(303) 810-2190 (mobile)

Agile Estimating and Planning
User Stories Applied
For Agile Software Development
Mike Cohn
Foreword by Kent Beck
MOUNTAIN GOAT SOFTWARE
© 2003–2008 Mountain Goat Software®

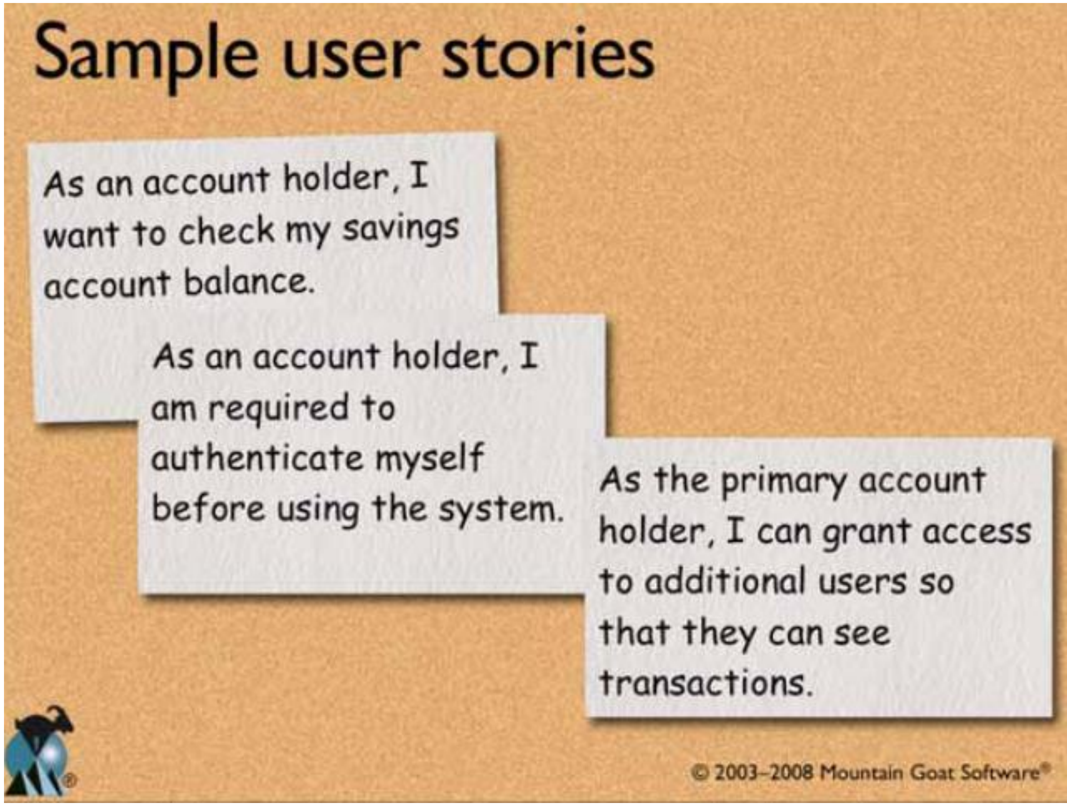


The Addison-Wesley Signature Series
User Stories Applied
For Agile Software Development
Mike Cohn
Foreword by Kent Beck

User Stories: Samples

Structure

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



Sample user stories

As an account holder, I want to check my savings account balance.

As an account holder, I am required to authenticate myself before using the system.

As the primary account holder, I can grant access to additional users so that they can see transactions.

© 2003–2008 Mountain Goat Software®

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

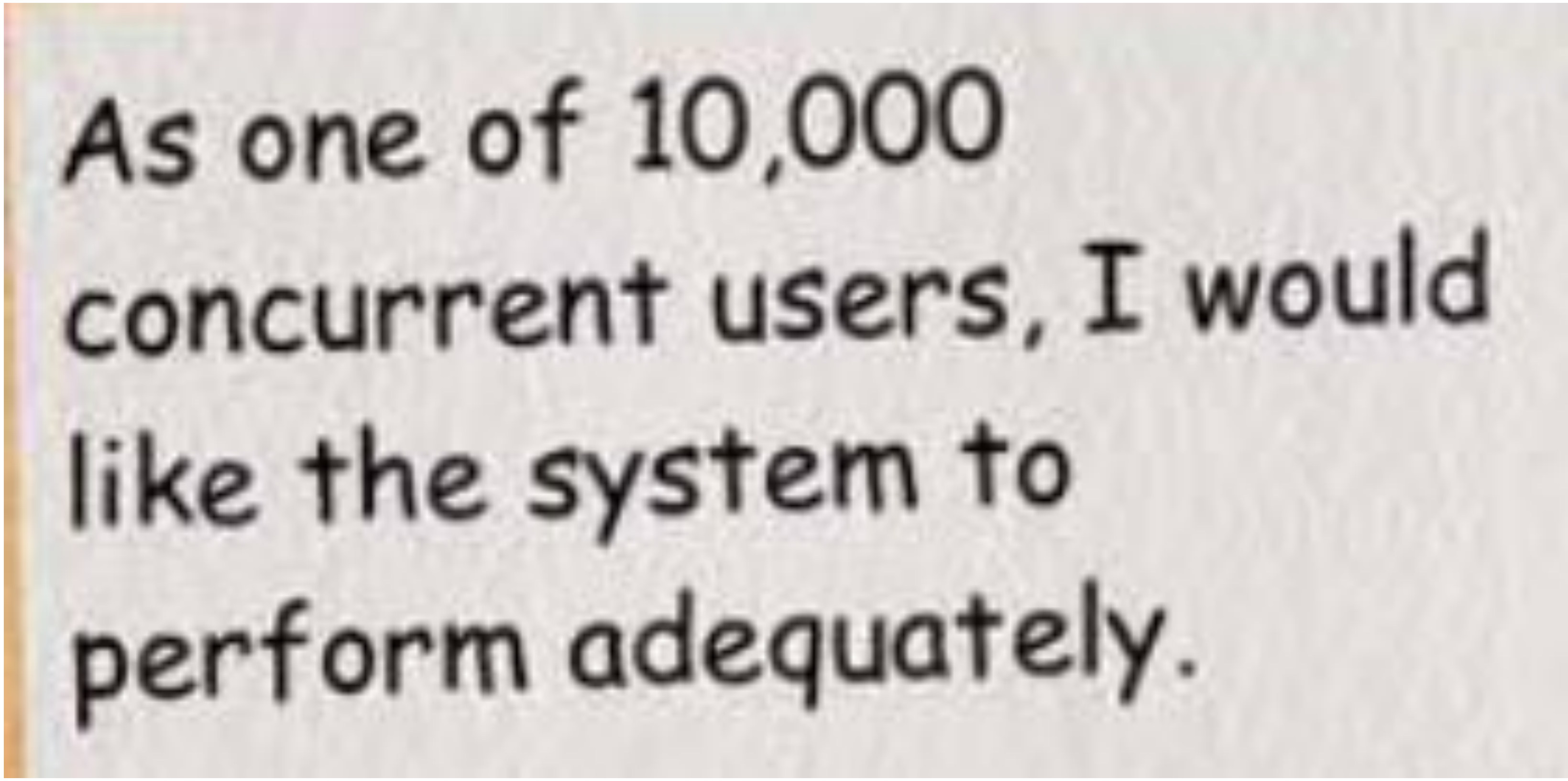
Myth 1: User stories and the conversations provoked by them comprise *verbal communication*, which is clearer than written communication.

- **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.”



As one of 10,000 concurrent users, I would like the system to perform adequately.

- 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.”

- **Right Size** [Requirement]: defined as:
- **The size that is sufficient for all requirements purposes,**
- **without any ‘In project’ supplements,**
- **at a cost that is lower than**
- **the costs of dealing with defects in the statement later.**

- Assertion
- User Stories are *rarely* detailed enough and clear enough to do intelligent planning (for example *estimation*)
- Or intelligent (dynamic) Prioritization

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

- User stories are a disaster for iterative development
- ⍰ because you cannot understand their incremental and final consequences;
- ⍰ you cannot measure evolutionary *value* delivery progress toward such objectives.   ⍰

- The nature of software development should not be to ‘write use cases’, stories, and functions,
- ⍰ as some seem to believe.
- ⍰ The Agile ideal is to **deliver incremental *value* to *stakeholders***.

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

- **Ambiguous unintelligible written stories are a logically bad basis for determining the priority of that story for anyone**.
- › Here is my idea of ‘priority’.
- › A potential increment will be prioritized
- based on ‘stakeholder value for costs’, with ‘respect to risk’.
- › Ambiguous written stories **do not admit numeric evaluation of value** for defined stakeholders, or of all cost aspects, or of all risk aspects.
- › Also a well-defined requirement can be evaluated for potential value to stakeholders,
  - it **cannot** be evaluated for cost.
  - The cost resides entirely in the **design**,
  - and the design is in principle not chosen yet!
  -
- Consequently you cannot choose best value for money with user stories alone.

- Try the story:
- ***“We want the most intuitive system possible”***
- What is the cost?
- You cannot have any useful idea of cost,
  - because the requirement is so vague that you cannot even understand it fully,
  - let alone *choose* a best design at all; and you cannot *cost* a design that is not
  - *chosen*. It is illogical

In addition, *until you know the specific design*,

- you *cannot understand the risk of deviation* from your objectives and costs,
- so you cannot prioritize iterations with regard to risk either.

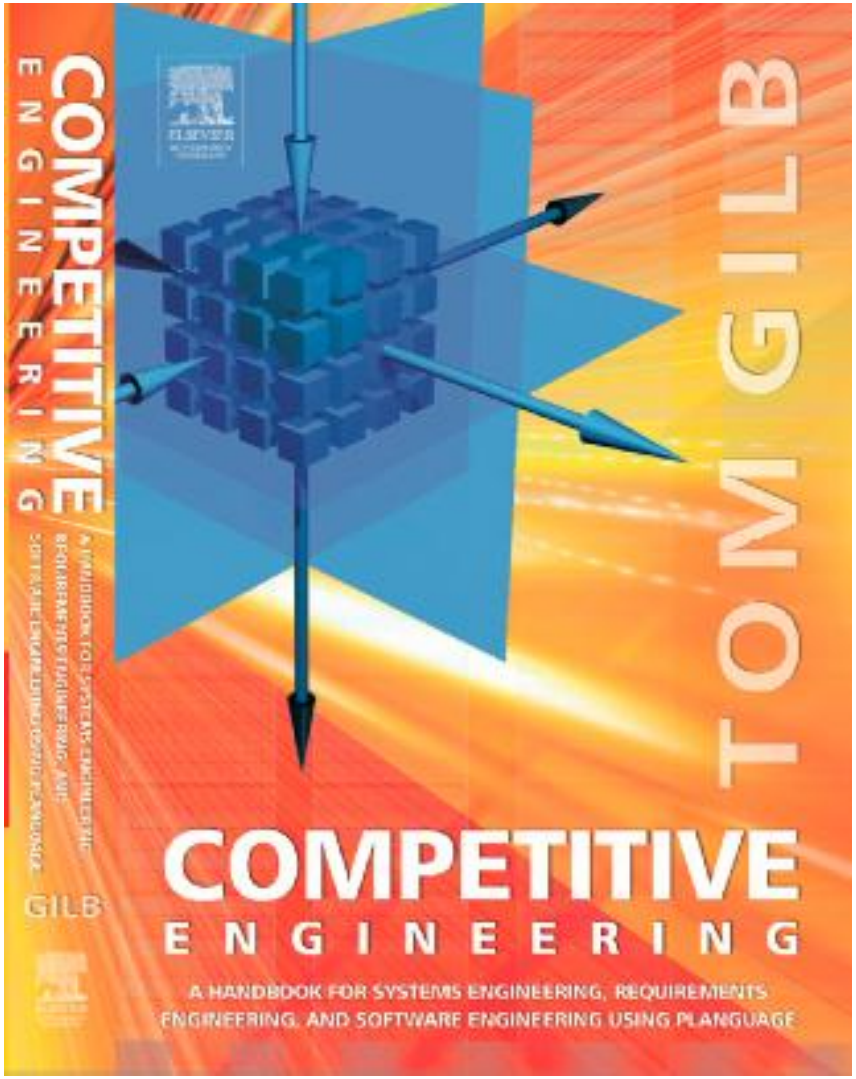
So, the prioritization argument for user stories **is logically unreasonable.**

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

- The arguments above, particularly the prioritization argument, say *no, everybody does not understand why.*
- › They may *feel* they understand,
- › but since the user story is incomplete and ambiguous,
- › they cannot *really* understand anything;
- › for example anything about value, stakeholders, design, costs, and risks.
- › There may be an *illusion* of understanding,
- › but there is no rationally defined understanding.

- However, there may be social comfort if teams misunderstand it together,
- › but in non-transparently different interpretations.
- › That does not lead to value or system success,
- › even for those who thought they understood the consequences of the user story choice.

References

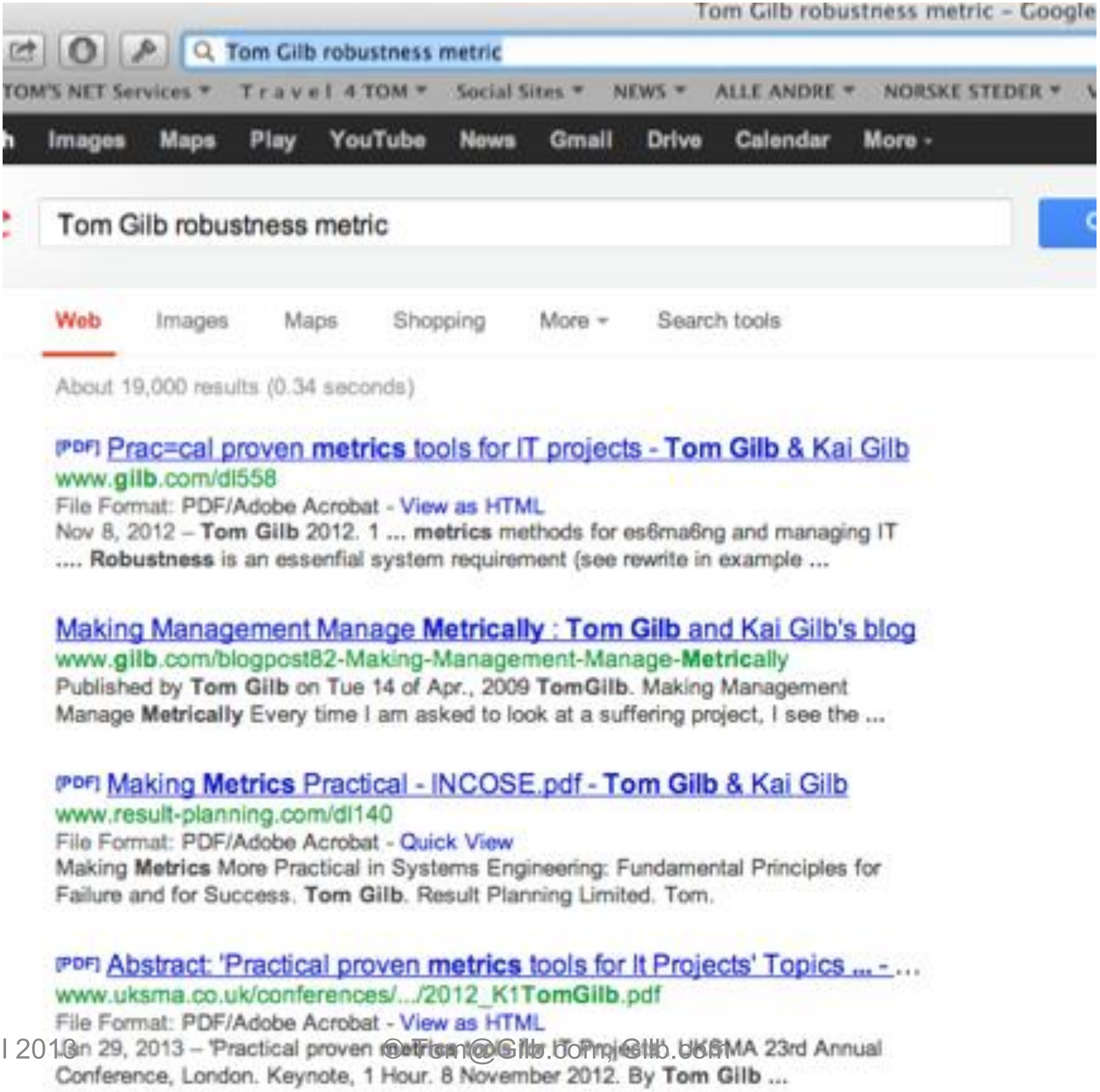

TOM GILB
COMPETITIVE ENGINEERING
A HANDBOOK FOR SYSTEMS ENGINEERING, REQUIREMENTS ENGINEERING, AND SOFTWARE ENGINEERING USING PLANGUAGE
ELSEVIER
COMPETITIVE ENGINEERING
GILB

- Ask me for free digital copy
  - Tom@Gilb.com
  - Subject : ‘Book’

- Download Related Papers and Slides and 2 Chapters at
- www.Gilb.com
- (Downloads tab)

The End

Even Tom Gilb ☺ for 'Software'



Tom Gilb robustness metric

About 19,000 results (0.34 seconds)

[PDF] Prac=cal proven metrics tools for IT projects - Tom Gilb & Kai Gilb
www.gilb.com/dl558
File Format: PDF/Adobe Acrobat - View as HTML
Nov 8, 2012 – Tom Gilb 2012. 1 ... metrics methods for es6ma6ng and managing IT .... Robustness is an essenfial system requirement (see rewrite in example ...

Making Management Manage Metrically : Tom Gilb and Kai Gilb's blog
www.gilb.com/blogpost82-Making-Management-Manage-Metrically
Published by Tom Gilb on Tue 14 of Apr., 2009 TomGilb. Making Management Manage Metrically Every time I am asked to look at a suffering project, I see the ...

[PDF] Making Metrics Practical - INCOSE.pdf - Tom Gilb & Kai Gilb
www.result-planning.com/dl140
File Format: PDF/Adobe Acrobat - Quick View
Making Metrics More Practical in Systems Engineering: Fundamental Principles for Failure and for Success. Tom Gilb. Result Planning Limited. Tom.

[PDF] Abstract: 'Practical proven metrics tools for It Projects' Topics ... - ...
www.uksma.co.uk/conferences/.../2012_K1TomGilb.pdf
File Format: PDF/Adobe Acrobat - View as HTML
Jan 29, 2013 – 'Practical proven metrics tools for IT Projects' Tom Gilb. UKSMA 23rd Annual Conference, London. Keynote, 1 Hour. 8 November 2012. By Tom Gilb ...