# 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 best way to achieve the ideals. We are going to argue that we need to improve user stories for serious and large projects. It is possible for trivial projects that user stories are sufficient tools.

**Myth 1**: User stories and the conversations provoked by them comprise *verbal communication*, which is clearer than written communication.
There may be occasions where good, conversational communication can help clear up bad written communication.
In fact we see a lot of really bad written 'user needs' communication; where we have measured the density of unintelligible words at 30% to 90% and more. [3]

It should be possible to reduce defective written requirements defects by two orders of magnitude, as our clients have done. [4]
A good written specification of any requirement type should be so clear and comprehensive that it is not necessary, as it is assumed with user stories, to have an oral conversation to clarify it.
The useful power of the well-written specification increases with the frequency of referring to it, and the number of people that need to interpret it.

Try to have a 'conversation' about the following example of a story:
*"We want the most intuitive system possible"*

Now compare your conversation with a specification like [5]:

**Intuitiveness:**
*Type: Quality Requirement*
*Stakeholders: Product Marketing, end users, trainers*
*Ambition Level: To make the intuitive and immediate application 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]*
*Goal [Deadline = 1st Release, Users = Novice, Tasks = Most Complex, Help = {No Training, No Written References} ] 10 seconds ± 5 seconds <- Product Marketing Manager.*
*Correctly Complete: defined as: the result would not ever need to be corrected as an error or as sub-optimal.*

If there are any questions about this spec, then the answer needs to be written down in the spec, for reference by all future users of the specification. Not just 'discussed' orally, and forgotten in practice.

**Myth 2**: "User stories represent a common language. They are intelligible to both users and developers."
User stories are not necessarily intelligible to all users, all developers, or any of them.
In fact it is very easy to prove that user stories are normally NOT intelligible.
We use the **ambiguity test** to measure intelligibility, and ask any available set of people to write down their personal interpretation of the words in the spec. Try, for example, the following statement:
*"We want the most intuitive system possible"*
*How many words are potentially ambiguous? All.*
*We collect the interpretations, and you will find everybody has quite different interpretations, none are identical.*

An alternative way to prove unintelligibility is counting defects in relation to the following standard using the Spec QC review method. [3]
Rule 1: The specification will be **clear enough to test**. Not later, but in *itself! Now!*
Rule 2: The specification will be unambiguous to all intended

readers, anywhere, anytime (including lawyers, and expert witnesses in your lawsuit).
Now using the spec
*"We want the most intuitive system possible"*
*How many of the words potentially violate those rules?*
*My personal answer is 7, but even 1 disqualifies the spec as useful.*

**Myth 3**: "User stories are the *right size* for planning and prioritizing."
We have no idea what this 'right size' assertion means. 'Right size' is highly ambiguous to all readers, and cannot, as it stands, be clear enough to test.
Let me give you my definition of 'right size', but before you read it, write down your own, and compare them.

**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.

*Hint: if that is a page of 60 lines in a clear and complete spec, for a single critical requirement, to do that, then that is the right size. If a one liner does the trick, fine. There is no point in oversimplifying the requirement just to have the project fail after a year, is there?*

**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*.[6]

**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. [7]
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! [8, Estimation paper in SQP March 2011]
In addition, *until you know the specific design*, you *cannot understand the risk of deviation* from your objectives and costs [9], 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. [10, Decision Rationale].

**Summary:**
If you think the user stories culture might be a problem for your project domain, you may be right. Leading Agile leaders believe we need something more relevant for the more demanding project environments. User stories are useful at one level, but 'too simple', as a primary or sole tool, for many software and IT environments. However, if they do work for you, there is no reason to upgrade to more powerful tools, so don't panic yet!

Who're ya gonna call? The Myth Busters!
Tom and Kai Gilb, www.gilb.com


1. http://www.stevedenning.com/Business-Narrative/user-stories-applied.aspx
   Attributes of User Stories by Denning and Cohn.
2. http://en.wikipedia.org/wiki/User_story
3. Gilb, Agile Specification Quality Control:
   http://www.gilb.com/tiki-download_file.php?fileId=264
   in Testing Experience March 2009
   Agile SQC Paper Gilb, in Testing Experience
4. DAC and Boeing experiences with aircraft design quality, using Gilbs Inspection method (now called Spec QC [5] .
   Case study.
   http://www.gilb.com/tiki-download_file.php?fileId=254
5. Gilb, Tom (2005) Competitive Engineering: A Handbook for Systems Engineering, Requirements Engineering, and Software Engineering, Using Planguage, Elsevier Butterworth-Heinemann. ISBN 0750665076.
   Download of Chapter 10, Evolutionary Project Management is available from http://www.gilb.com/tiki-download_file.php?fileId=77
   [Gilb 2005 b] Chapter 5 Scales of measure free sample

chapter
http://www.gilb.com/tiki-download_file.php?fileId=26
[Both Chapters Accessed 3 January 2011].

6. Gilb, Tom (2010c) Value-Driven Development Principles and Values – Agility is the Tool, Not the Master. Agile Record, July 2010, 3. Also available from: http://www.gilb.com/tiki-download_file.php?fileId=431

   See also part 2 of the paper at Part 2 "Values for Value" http://www.gilb.com/tiki-download_file.php?fileId=448 Agile Record 2010, www.agilerecord.com, October 2010, Issue 4
7. Gilb and Maier, Managing Priorities. http://www.gilb.com/tiki-download_file.php?fileId=60
8. Gilb Estimation or Control http://homepage.mac.com/tomgilb/filechute/Estimation%20or%20Control%202010%20MASTER.pdf edit note, a major revision of this paper will be published March 2011 in Software Quality Progress and we need an updated reference to it. I can supply our edit of this on request to readers. March 9 2011 tom gilb
9. Gilb, Risk Management : A practical toolkit for identifying, analyzing and coping with project risk. http://www.gilb.com/tiki-download_file.php?fileId=20
10. Gilb, Decision Rationale: A Quantified Decision-Making Basis Using Planguage, 2006 http://www.gilb.com/tiki-download_file.php?fileId=43

## > About the authors

*Tom Gilb and Kai Gilb have, together with many professional friends and clients, personally developed the methods they teach. The methods have been developed over decades of practice all over the world in both small companies and projects, as well as in the largest companies and projects.*

**Tom Gilb**

*Tom is the author of nine books, and hundreds of papers on these and related subjects. His latest book 'Competitive Engineering' is a substantial definition of requirements ideas. His ideas on requirements are the acknowledged basis for CMMI level 4 (quantification, as initially developed at IBM from 1980). 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**

*has partnered with Tom in developing these ideas, holding courses and practicing them with clients since 1992. He coach 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 the common methods somewhat differently. Consequently the students benefit from two different styles.*

*There are very many organizations and individuals who use some or all of their methods. IBM and HP were two early corporate adopters. Recently over 6,000 (and growing) engineers at Intel have adopted the Planguage requirements methods. Ericsson, Nokia and lately Symbian and A Major Mulitnational Finance Group use parts of their methods extensively. Many smaller companies also use the methods.*