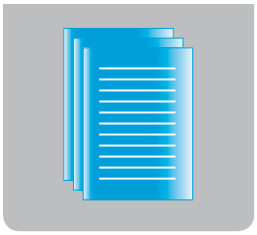
Agile Record

The Magazine for Agile Developers and Agile Testers

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

July 2011

issue 7



Real Value Delivery, and The Unity Method for Decomposition into Iterations

by Tom and Kai Gilb

The Myth we want to discuss this time is about decomposing projects to fit into iterations. We want to first look at some of the poor ideas in the Agile Manifesto related to decomposition to deliver value.

"Our highest priority is to satisfy the customer through early and continuous delivery of valuable software." (Agile Manifesto Principles)

This is an excellent sentiment. However, it has the following flaws, which make the entire Agile culture a weak one. A culture that cannot really live up to the excellent intentions. First, the "**customer**" concept is a very narrow perspective, and it is not carefully defined.

A more useful term would have been "**stakeholder**". Stakeholders are all people, organizations and things that have values which we need to respect, in order to avoid failure and to achieve success. Stakeholders are a lot more comprehensive than customers, no matter how you define customers. Normal projects have at least 40 interesting stakeholders (Miller).

So Agile projects are at high risk of failing to identify most stakeholders and most values they should deliver, simply by adopting the too narrow "customer" scope.

The second problem is the "**valuable software**" paradigm.

"**Software**": This reflects the code-centric world of the Manifesto writers. They should rather have specifically said they wanted to deliver value, even if this could not be attained by code. They should ideally have taken a systems perspective. This means that **any** useful way to transmit value to stakeholders is a good thing, for example through training, motivation, hardware, databases. But no. If we can't do it with code, you cannot have it. Sounds either selfish or ignorant to me. Who needs such misers running our projects?

"**Valuable**": This is not defined. Sounds good. Who could be against it? In practice, however, as we now know, this translates into user stories that are deemed valuable, probably by a Product Owner. Sounds OK, if you don't think too deeply. However, what they fail to explicitly say is that most stakeholder values are variables, and they are also *multi-dimensional*. Values are of course very tailored to the business and the times, and in particular to the *stakeholder* values. They do not look at all like user stories. They look like quality attributes, performance attributes and cost attributes. These need to be defined on a Scale of Measure; and different requirements need to be specified with corresponding ideas of value for reaching that requirement level. This is nowhere near what user stories do in practice or theory. Of course, these user stories could be enhanced. Conventional Agile culture, however, hardly discusses, let alone teaches and practices any such value improvement to user stories.

The format for value specification looks something like this:

Value Idea Template:

***Headline**: Sharp reduction in real and total cost of making a trade.*

***Type**: Stakeholder value requirement*

***Stakeholders**: Marketing, Customer Operations, Financial Authorities.*

Scale: average of the total, consequential, cost of every trade, of a given type, made by a given trader

Status [Type = Simple, Frequent, Trader = Amateur] €1.00

Goal [Type = Simple, Frequent, Trader = Amateur] €0.10

Value: €50 billion annual saving to our market

Here is another example:

User-Friendliness.Learn.Contacts

***Type**: Product value requirement*

***Stakeholders**: Users, sales*

Scale: average time in minutes, to learn how to program contact names and telephone numbers into the memory of the phone.

Past [July 2011] **35 min.**

Goal [July 2012] **5 min.**

The top 10 or so critical objectives for a project should be specified this way. Then we could track the measurable progress towards the stakeholder values, next week and every week. We fail to see how any user stories can define value so directly, and allow us to fulfil the Aagile movement's intent of delivering such value early and continuously.

"Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale." (Agile Manifesto Principles)

We have to interpret "working software" as code, almost bug free,. Aas user stories that "work". Unfortunately, the connection between the working code and the idea of stakeholder value is at best weak, and normally totally disconnected.

We did a project (Bring.com) for the Norwegian Postal System. A professional specialist Scrum team developed the system. They did everything right, from the Scrum point of view. Working code was delivered. One detail. The sales of postal services went down drastically and immediately when the Sscrum developed system was installed. It was only by analyzsing the business values (make sales) and the stakeholder values (the speed at which potential customers found the services they needed) that we found the root cause of the negative value delivered by the Scrum run project. Fair to say that the fault was not with codes and Scrum teams, but was in bad analysis of customer values! Scrum delivered, user stories, burn down charts and working software, it did not deliver value to stakeholders.

"Working software is the primary measure of progress." (Agile Manifesto Principles)

Silly narrow view of the real world. Only a coder's mother could love this.
The primary measure of progress must be real measurable incremental value, defined in the requirements (oh, must not have those nasty requirements...) by the stakeholders (who?).

The whole concept of planning in terms of uUser sStories, and related burn down charts for progress is one that, might work for some classes of development (small systems, non-t quality-critical systems). But However, we do not think it is an intelligent idea, not even for those smaller less critical projects.

Decompose Projects by Value.
Our conclusion is that we must decompose projects and prioritize our actions by **value for money**.
Decomposing by sStories is simply **not** value for money. There is no estimate or measurement of value or money in stories.

We have a variety of teachable, free, methods for decomposing by value. There are 19 principles published in our book and paper (Decomposition Principles). You can even become a Decomposition Master in 2 dDays for free.

Here is a sketch of our **Unity Method**, or the 111111 method which subdivides project deliveries by value, using the following simplified concepts:

Divide, so that you focus on

1 **Function**, at least
1 % percent increase in **value**, towards gGoal, at least
1 **stakeholder**, at least
1 week delivery **cycle**
1 **design** applied, to deliver the value, for
1 defined scalar **value**
See **Unity** reference for more detail and a practical case.

There are other principles, anything that works is good here. The *important point* is to *end up with high value*, that can be delivered within an incremental delivery cycle. The important idea is that *real stakeholders* can experience *real value* in a fully operational and integrated (but of course not 'completed') system. Notice we did not mention code or stories. Irrelevant!

If the Agile community does not get behind real value delivery, then someone else will. And coders will be told by them what to code.

Let's make up our minds. Do we want to deliver real value, or do we want to code, perhaps with little or no value perceived?

Who're ya gonna call? The Myth Busters!
Tom Gilb & Kai Gilb,
Tom@Gilb.com Kai@Gilb.com
www.Gilb.com
www.KickAssProject.com

References

Miller: Roxanne Miller, CBAP
"The Quest for Software Requirements".
Maven Mark Books, May 2009, ISBN 1595980679
This book has excellent and deep material on the stakeholder concept.
http://www.requirementsquest.com

Principles: 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

Values: 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

US: User Stories: A skeptical View. Agile Record, previous issue.
www.agilerecord.com/agilerecord_06.pdf

Bring: Case Study slides by Kai Gilb, of the Bring system.
http://www.gilb.com/tiki-download_file.php?fileId=277

Decomposition Principles:

Evo chapter of CE Book.
Chapter 10: Evolutionary Project Management:
http://www.gilb.com//tiki-download_file.php?fileId=77

Detailed discussion of these decomposition principles:
Decomposition of Projects: How to Design Small Incremental Steps INCOSE 2008
http://www.gilb.com/tiki-download_file.php?fileId=41

and the (**Unity**) 111111 Method
http://www.gilb.com/tiki-download_file.php?fileId=451
at the Smidig (Agile) conference Oslo 2010

> 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.