Agile Record

The Magazine for Agile Developers and Agile Testers

Requirements in Agile Projects

August 2012

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

issue 11

The Top 10 Critical Requirements are the Most Agile Way to Run Agile Projects

by Tom and Kai Gilb

There is a dangerous assumption amongst agile professionals. It is that 'requirements' are mainly in 'use cases' (1), and that *progress* is measured by burn-down charts. We would like to strongly dispute this dangerous idea, and offer a more powerful, and 'more agile' requirements concept.

Our idea, and it is one we have *practiced* for *decades (2)*, is simple, like agile should be. It reduces bureaucratic and up-front documentation – like agile should do, and it is focussed on delivery of value to our stakeholders – like agile says it *wants* to.

Our idea is (3,4) that in 1 day, on 1 page, 1 team will draft 1 set of maximum 10 critical project requirements. These will be agreed as the most critical reasons for funding the project. If these requirements are met, then the project will be a complete success.

Everything else, including use cases, functions, user stories, designs, architecture, is regarded as the necessary 'means' (*design*, not requirement) for meeting the top 10 requirements.

It is our experience that when you put the right questions ("what are the primary expectations for funding this project?"), to the right people (the project funders and sponsors, *not* the users), you will always get, and get agreed, a limited set of answers, your 'top 10'. In fact, the top 10 are usually *already* hiding in the project documentation and the management project slides. And our current projects totally ignore them! We are so busy laying stones and walls that we forget the cathedral we are supposed to be building (5).

Our experience, when asking responsible managers to tell us what their top 10 requirements are, is this:

1. They can identify the top few immediately.
2. They have already documented them *somewhere* (but the project has ignored them).
3. They can *quantify* the degree of improvement they expect the project to produce (if guided, they can develop a defined scale of measure, and a numeric goal on that scale).
4. They can do this in a morning session and edit it to be 'good enough' for project use in one day of work (3).
5. It is not perfect. It can and will be continuously improved. But it is also remarkably stable.

Here is what our student, Richard Smith at Citigroup, reports (6):

"You may be interested to know that I wrote a detailed business requirements spec (no design, just requirements) adopting many of your ideas shortly after the course, including key quantified requirements. This spec ended up staying largely stable for a year as we did an Evo–like (Ref. 4) development process, at the end of which we successfully went live with a brand-new FX order management system in a global big-bang release to 800 Citi users in 20 locations." (2009 e-mail to us).

This is evidence that there is less 'requirements churn' if there are fewer but critical requirements.

However, something has to be learned as we release increments of the system!

Richard Smith follows up by saying (6)

"but the detailed designs (of the GUI, business logic, performance characteristics) changed many, many times, guided by lessons learned and feedback gained by delivering a succession of early deliveries to real users."

- This sounds like the essence of *real* agile to me! Agile should be about learning which designs satisfy the critical requirements; not about implementing so-called functions,

features and use cases, which are usually sub-optimal amateur 'design' with another name.

We have also discovered that much of what people call 'requirements' are really design. The test is simple. Ask why!

If the answer is clearly a real requirement, then what you were *calling* a requirement is probably really a *design*. Example: Why 'password'? Answer: "Security!". Ah, so security is your requirement, password is a design. And possibly not the smartest design. It depends on the unstated security 'requirement'. (7)

Here is our definition of a 'requirement' (8):

"**A Stakeholder-valued future state,** under defined conditions."

And here is our definition of a 'design':

"**A design is a concept that is *intended* to satisfy some requirements,** to some degree."

The top 10 most critical requirements are mostly 'quality' (defined as how well the system functions) requirements, together with some work capacity and cost reduction requirements. Everybody can quantify, and thus clarify, work capacity and cost reduction. But almost everybody has a big problem with the '-ilities'.

How *do* you quantify things like security, usability, maintainability, and adaptability?

There are 3 methods we teach (9, 10), and we have found no 'unquantifiable' qualities.

1. Look it up in a book (9).
2. Use common sense and domain knowledge to work out scales (9, process).
3. Google it: for example 'usability metrics'. Lots of good answers on 1 page.

An excellent example of doing this with the Evo-agile method is our client Confirmit. (11)

Impact Estimation Table: Reportal codename "Hyggen"

Current Status	Improvements		Reportal - E-SAT features		
Units	Units	%	Past	Tolerable	Goal
			Usability.Intuitivness (%)		
75,0	25,0	62,5	50	75	90
			Usability.Consistency.Visual (Elements)		
14,0	14,0	100,0	0	11	14
			Usability.Consistency.Interaction (Components)		
15,0	15,0	107,1	0	11	14
			Usability.Productivity (minutes)		
5,0	75,0	96,2	80	5	2
5,0	45,0	95,7	50	5	1
			Usability.Flexibility.OfflineReport.ExportFormats		
3,0	2,0	66,7	1	3	4
			Usability.Robustness (errors)		
1,0	22,0	95,7	7	1	0
			Usability.Replacability (nr of features)		
4,0	5,0	100,0	8	5	3
			Usability.ResponseTime.ExportReport (minutes		
1,0	12,0	150,0	13	13	5
			Usability.ResponseTime.ViewReport (seconds)		
1,0	14,0	100,0	15	3	1
			Development resources		
203,0			0		191

Current Status	Improvements		Survey Engine .NET		
Units	Units	%	Past	Tolerable	Goal
			Backwards.Compatibility (%)		
83,0	48,0	80,0	40	85	95
0,0	67,0	100,0	67	0	0
			Generate.WI.Time (small/medium/large seconds)		
4,0	59,0	100,0	63	8	4
10,0	397,0	100,0	407	100	10
94,0	2290,0	103,9	2384	500	180
			Testability (%)		
10,0	10,0	13,3	0	100	100
			Usability.Speed (seconds/user rating 1-10)		
774,0	507,0	51,7	1281	600	300
5,0	3,0	60,0	2	5	7
			Runtime.ResourceUsage.Memory		
0,0	0,0	0,0		?	?
			Runtime.ResourceUsage.CPU		
3,0	35,0	97,2	38	3	2
			Runtime.ResourceUsage.MemoryLeak		
0,0	800,0	100,0	800	0	0
			Runtime.Concurrency (number of users)		
1350,0	1100,0	146,7	150	500	1000
			Development resources		
64,0			0		84

Current Status	Improvements		Reportal - MR Features		
Units	Units	%	Past	Tolerable	Goal
			Usability.Replacability (feature count)		
1,0	1,0	50,0	14	13	12
			Usability.Productivity (minutes)		
20,0	45,0	112,5	65	35	25
			Usability.ClientAcceptance (features count)		
4,4	4,4	36,7	0	4	12
			Development resources		
101,0			0		86

Current Status	Improvements		XML Web Services		
Units	Units	%	Past	Tolerable	Goal
			TransferDefinition.Usability.Efficiency		
7,0	9,0	81,8	16	10	5
17,0	8,0	53,3	25	15	10
			TransferDefinition.Usability.Response		
943,0	-186,0	######	170	60	30
			TransferDefinition.Usability.Intuitiveness		
5,0	10,0	95,2	15	7,5	4,5
			Development resources		
2,0			0		48

Fig.1: The 25 quantified quality and work capacity requirements

The 25 quantified quality and work capacity requirements (Fig.1) were developed in a week by Confirmit. These were then used in twelve one-week cycles of quality delivery, with release to world market after every 12 weeks. The illustration above shows the feedback at cycle 9 of 12, and the improvements % shows the % of the way to target levels accomplished by the 4 parallel teams.

One of the authors (Kai) analyzed a project (Bring, Oslo, 12) which had used Scrum in the conventional way, correctly. However, on delivery to the market, the sales dropped dramatically. Kai's analysis was that there was no quality requirement for *the speed with which customers could find the correct service*. The result was intolerable. Customers gave up and went to competitors. However, when the system was redesigned to meet such a quality requirement, all was well. Scrum alone is not enough! Quality requirements are necessary to manage the Scrum process!

One of our Bank clients in London has instituted our Evo process as a framework to manage agile processes like Scrum (4), so that they are more specifically responsive to stakeholder needs at the top quality levels. Scrum alone is not enough.

Ref. 1. Gilb's Mythodology Column, 1, Agile Record 6, User Stories: A Skeptical View

Ref. 2. Quantified top level project objectives, as documented in 1988, *Principles of Software Engineering Management (20th printing)*. The book that many agilistas (like Kent Beck) credit with their early inspiration. Especially about incremental and evolutionary delivery, the core of agile today.

Ref. 3. The outline of our agile Evo startup week process
http://www.gilb.com/dl521
Day 1: **Project Objectives:** The top few critical objectives quantified.
Objective: Determine, clarify, agree critical few project objectives – results – end states
Process:
Analyze current documentation and slides for expressed or implied objectives (often implied by designs or lower-level objectives)
Develop list of **Stakeholders** and their needs and values
Brainstorm 'top ten' critical objectives names list. Agree they are top critical few.
Detail definition in Planguage – meaning quantify and define clearly, unambiguously and in detail (one page)
Quality Control Objectives for Clarity: Major defect measurement. Exit if less than 1.0 majors per page
Quality Control Objectives for Relevance: Review against higher-level objectives than project for alignment.
Define Constraints: resources, traditions, policies, corporate IT architecture, hidden assumptions.
Define Issues – yet unresolved
Note we might well choose to do several things in *parallel*.
Output: A solid set of the top few critical *objectives* in quantified and measurable language. *Stakeholder* data specified.
Participants: anybody who is concerned with the business results, the higher the management level, the better.

Ref. 4. The Evo standard
http://www.gilb.com/dl487

The 'Evo' (**Evo**lutionary) Method for Project Management.

Process Description

1. Gather from all the key stakeholders the top few (5 to 20) most critical goals that the project needs to deliver.
   Give each goal a reference name (a tag).

2. For each goal, define a scale of measure and a 'final' goal level.
   For example: *Reliable: Scale: Mean Time Before Failure, Goal: 1 month.*

3. Define approximately 4 budgets for your most limited resources
   (e.g., time, people, money, and equipment).

4. Write up these plans for the goals and budgets
   (*Try to ensure this is kept to only one page*).

5. Negotiate with the key stakeholders to formally agree the goals and budgets.

6. Plan to deliver some benefit
   (that is, progress towards the goals)
   in *weekly* (or shorter) increments (Evo steps).

7. Implement the project in Evo steps.
   Report to project sponsors after each Evo step (weekly, or shorter) with your best available estimates or measures, for each performance goal, and each resource budget.
   On a single page, summarize the *progress to date* towards achieving the goals and the costs incurred.

8. When all goals are reached: 'Claim success and move on'.

a. Free remaining resources for more profitable ventures

Ref. 5 "Three stonemasons were building a cathedral".
http://www.thehighcalling.org/audio/work/stonemasons
Three stonemasons were building a cathedral when a stranger wandered by. The first stonemason was toting rocks to a pile, near a wall. "What are you doing?" said the stranger.
"Can't you see that I'm carrying rocks?"
The stranger asked the second laborer, "What are you doing?"
"I'm building a wall" he replied.
A few steps away, the stranger came upon a third mason. "What are you doing?" he asked. This worker smiled. "I'm building a cathedral to the glory of God!" Same jobs, different missions.

Ref. 6. Richard Smith: Citigroup experience with Evo
http://rsbatechnology.co.uk/blog:8

Ref. 7. Gilb: Quantifying Security: How to specify security requirements in a quantified way.
http://www.gilb.com/dl40

Ref. 8. Full Planguage Glossary of Concepts (655+) updated June 13 2012
http://www.gilb.com/dl46, see particularly Requirement and Design idea.

Ref. 9 CE book chapter 5, Scales of Measure, free download.
http://www.gilb.com/tiki-download_file.php?fileId=26

Ref. 10 How to Tackle Quantification of the Critical Quality Aspects for Projects for Both Requirements and Designs
http://www.gilb.com/tiki-download_file.php?fileId=486 Slides
http://www.gilb.com/tiki-download_file.php?fileId=124 Paper

11. Confirmit Case (of Evo and quantified qualities as main drivers)
http://www.gilb.com/dl152 slides including Confirmit case (Firm)
'WHAT'S WRONG WITH AGILE METHODS? SOME PRINCIPLES AND VALUES TO ENCOURAGE QUANTIFICATION' with Confirmit Case. http://www.gilb.com/dl50

12. Bring Case. The Inmates are running the asylum, Construx Summit talk Oct 25 2011 Seattle, contains considerable Bring Case slides
www.gilb.com/tiki-download_file.php?fileId=488

> 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 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 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.