TargetProcess, Inc.

Whitepaper

Agile Tools.
The Good, the Bad and the Ugly.

"Every gun makes its own tune."

Man With No Name

Michael Dubakov
And
Peter Stevens

www.TargetProcess.com
Phone: 877-718-2617
Fax: 607-398-7927

Table of Contents

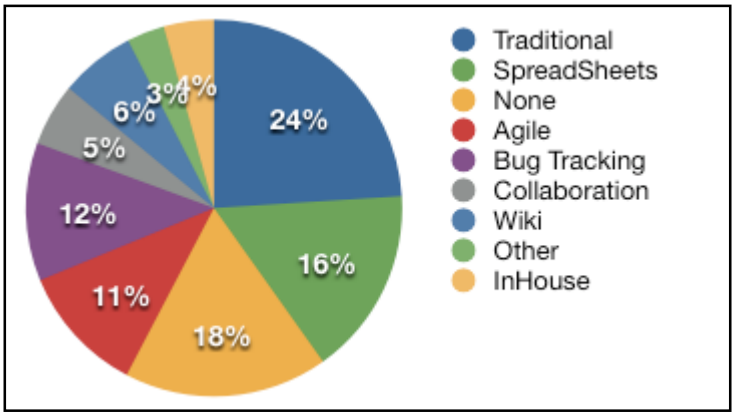
Introduction

Agile software development adoption spreads over the world. According to the latest surveys, about 70% of organizations adopt agile practices (http://www.ambysoft.com/surveys/agileFebruary2008.html) and stick with it. During the last 10 years, agile methods jumped from nowhere to the peak. Such rapid revolution demands new tools. There are many developers-focused tools appearing including JUnit, Eclipse, Cruise Control, etc. However, the project management community is less self-sufficient and unable to create new tools for the new methodology in their spare time. As a result, most companies are still using old-fashioned project management tools like MS Project or generic tools like spreadsheets for project planning and tracking.

The table below contains results of project management tools usage taken from TargetProcess leads (from May to July). People requested trial or free versions of TargetProcess. There were 371 requests with answers about tools usage.

Tool Category	Answers
Traditional	89
Spread sheets	60
None	65
Agile	41
Bug Tracking	44
Collaboration	20
Wiki	24
Other	12
In-House	16
	371

The question was in a free form and the answer was typed. Surprisingly, 18% of the respondents do not use any tool at all. Most likely, many people in the "None" category use paper and whiteboards. However, it is just an assumption.



- 24% companies use traditional project management tools (vast majority use MS Project).
- 16% use Spread Sheets (vast majority use MS Excel).
- 12% use various Bug Tracking tools (JIRA is the winner in this category).
- 11% use Agile Project Management Tools (open source XPlanner is a leader).

Tool	Answers	Category
MS Project	81	Traditional
Spread sheets	60	Spreadsheets
None	65	None
Rally	7	Agile
JIRA	17	Bug Tracking
XPlanner	12	Agile
VersionOne	9	Agile
Basecamp	9	Collaboration
Trac	16	Wiki
Mantis	10	Bug Tracking
Other Bug Tracker	17	Bug Tracking
custom/in-house	16	InHouse
dotProject	8	Traditional
Other Collaboration	11	Collaboration
Mingle	6	Agile
ScrumWorks	7	Agile
VSTS	7	Other
Wiki	8	Wiki
Paper	5	Other
	371	

In total, 52% use traditional tools, excel or bug tracking tools for project management. Are these the best choices for software development projects?

In this article, we discuss various approaches for agile project management and provide comparisons between the simplest tools (index cards and whiteboards), spreadsheets, traditional project management software and agile project management software.

Why do you need a tool?

Let's assume that we have a large and shiny nail. What is the best tool for the nail? Hopefully, the answer is obvious to most of us. Now, let's assume that we have a development team and a "shining", promising, cool new agile development process. Most likely the hammer will not help.

To tackle this problem, it is essential to have at your disposal a tool that enables requirements gathering, iteration planning, progress tracking and reporting. You can't rely on memory for requirements gathering. You can't rely on the universal perception for iteration planning and you definitely can't rely on telepathy for progress tracking and reporting. You need a tool that will do the job with minimum effort and minimum side effects.

> *"I think that people and how they interact on a project are the most important thing, and I think that they need to create a way of working -- a process -- that works best for them. Because their interactions are critical to project success, I suggest that teams begin the work with an approach that will bring them together as people, not one that will let them remain apart, communicating electronically".*
>
> ***— Ron Jeffries***

The Simplest Tools

If you ever tried Extreme Programming or Scrum, you are familiar with the simplest tools commonly used for the fine-grained short-term project plans creation and progress tracking: Index Cards and Big Visible Charts (http://xp123.com/xplor/room-gallery/index.shtml).

The beauty of cards is threefold:

1) They are flexible.
2) They are easy to use when working in a group. Used to manage the sprint.
3) They are part of an extremely easy to understand overview of the state of the project.

So whether you are brainstorming user stories, planning tasks for the current sprint, or reviewing the sprint in a retrospective, cards are easy. They let people work in parallel, and are very effective for communicating and prioritizing information.

Cards have one more benefit. **Cards are tangible**. People love to use something tangible like paper, post-it notes, markers, etc. They provide a feeling of control and real feedback. For some reason, they stimulate the imagination and good ideas pop up more often.

The drawbacks of cards are also threefold:

1) Reusing data.
2) Backing up data.
3) Having remote access to the data.

So if you use cards to create your product backlog, that's fine, but now yourCEO wants a report on which stories have been completed and which are forecasted to be completed by which sprint. This is easy to do if you have a spreadsheet, but first you have to enter all the data by hand, a job which usually falls on one person. Related to reuse is backup (and possibly other security issues): what happens if you lose the cards?
Things are different in a distributed team. Everybody needs to see the task board. In a meeting, everybody needs to not just read, but also to post the cards on the wall and it is required to send the sprint contract to the customer. There are teams using web cams and Skype connections to broadcast the task board to remote sites, but even that seems like a weak alternative. The remote sites have read-only access to the cards, if they are even legible.

Moreover, using cards to manage the Scrum data (e.g. backlogs, taskboards, and burndown) does not address how to manage other documentation (e.g. design documents, and program documentation).

And finally, how well does the approach scale? When you have hundreds or thousands of stories?

You may use cards for the Sprint Retrospective and for the initial product brainstorming. For the first sprints, this is a good place to start, but most teams will quickly reach the natural limits of this approach.

Pros	Cons
• Easy to learn and easy to use • Flexible, may be adopted for team • Inexpensive	• Don't work for distributed teams • Don't work for large teams • Lack of reporting • Manual remaining time update, burn down update etc.

General Software Tools

Wiki is a very flexible tool. It can be used for almost any content management. There are teams that use wiki for scrum development management. For example, you may setup Scrum area on the wiki with structure like this:

- Sprint 1
- Sprint 2
- Sprint n
    - Sprint Contract
        - Product Backlog at start of sprint (xls and pdf, as sent to the customer)
        - Status Overview from the Planning Meeting (current burndown chart, estimated resources for next sprint, definition of done, time/location of next demo meeting, etc.)
        - Sprint Backlog at start of Sprint (pdf - this is the actual contract)
        - Daily Scrum Spreadsheet (xls - updated daily)
        - Status Review from the Demo (stories/points accomplished, costs etc)
    - Sprint Stories
        - Story 103 Some story -> all project documents related to that story
        - Story 105 Some other story

Wiki is not appropriate for backlog management, however a simple spreadsheet looks like a good tool for that purpose. It is possible to consolidate all of the wish lists into one spreadsheet which becomes the product backlog. It may have as little as 5 columns: Name, Effort, Priority and Estimate and "reference number" column to make it easier to find individual stories.

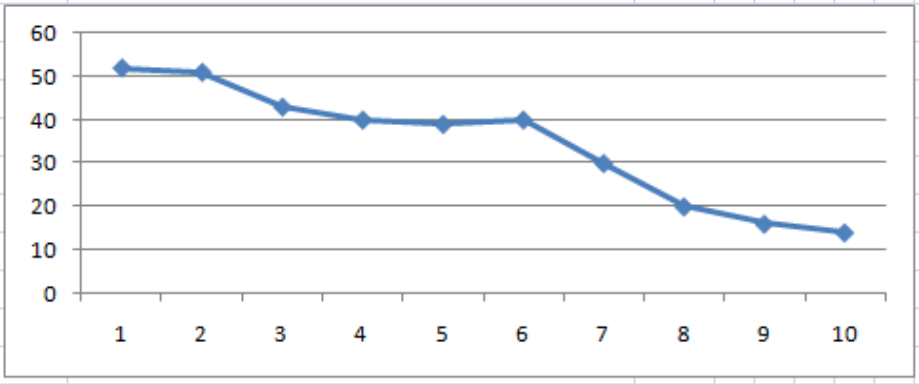
Product Backlog in Excel

	A	B	C	D	E
1	ID	Name	Effort	Initial Estimate	Priority
2	6071	Report Editor Tuning	3	2	Must Have
3	6381	Drag & Drop Tags area	8	5	Must Have
4	6384	Import custom fields values	5	5	Great
5	6302	Bind Emails/Requests based on Key in Subject	3	3	Great
6	5584	Help Desk Tickets	13	20	Great
7					

There are many possible ways to track sprint progress using spreadsheets (http://www.odd-e.com/home_page/html_files/bl_templates.html). The simplest daily scrum spreadsheet contains a list of all tasks and remaining effort for each day. The spreadsheet should be updated every day and uploaded into the wiki to make it available for all parties.

Iteration Burndown Chart in Excel

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	ID	Story/Task	Effort	Sprint Days									
2				1	2	3	4	5	6	7	8	9	10
3	6071	Report Editor Tuning	3										
4	5905	Create Test Cases	4	4	4	4	4	4	4	4	4	4	4
5	5901	Implementation	12	12	12	12	12	6	6	6	6	6	6
6	5907	Testing	4	4	4	4	4	4	4	4	4	4	4
7	6302	Bind Emails/Requests based on Key in Subject	3										
8	6303	Implementation	4	4	4	4	4	9	14	10	4	0	0
9	6304	Create Test Cases	2	2	2	2	0	0	0	0	0	0	0
10	6305	Testing	2	2	2	2	2	2	2	2	2	2	0
11	5584	Help Desk Tickets	13										
12	5615	Spec	1	1	1	0	0	0	0	0	0	0	0
13	5616	Implementation	12	15	14	7	6	6	6	0	0	0	0
14	5617	Create Test Cases	4	4	4	4	4	4	0	0	0	0	0
15	5618	Testing	4	4	4	4	4	4	4	4	0	0	0
16													
17		Effort Left		52	51	43	40	39	40	30	20	16	14



The advantage of this approach is self evident: many companies have these tools and know how to use them. A spreadsheet is very flexible, so you can slice and dice the data as you see fit. Data reuse is not a problem per se. Backup is handled by whatever mechanisms you have in place for your PC.

The wiki requires a lot of discipline so that you can find things. The basic Scrum flow was relatively easy, but data which did not obviously belong to a single sprint was always somehow a special case and finding it later was hard. A good search tool is a prerequisite for using the Wiki.

The problem with Wiki and Excel is quite common. These tools are simple, but **general**. They do not have business logic behind them, but provide frameworks to resolve simple data manipulation problems.

Most likely the product backlog management will be inefficient with the Wiki and spreadsheet. The Wiki is too inflexible as a data structure and the spreadsheet is too flexible. The customer, when he had the data, could too easily change the structure of the spreadsheet. Assuring the quality of the input was a problem and we had "file locking" problems: Either the customer could update or you could update, but you could not see changes made by the other until you got the spreadsheet back, nor could you update unless you had agreed with the customer that you had the writable copy.

It is good, but what are the benefits? Wiki+Excel will not work in distributed environment and will not work for large collocated teams well. Concurrent access, real-time reporting, integration with other dev. process tools – all these functions are iust

Pros	Cons
• Easy to learn and easy to use • Cheap	• Doesn't work for distributed teams • Doesn't work for large teams • Limited reporting • Limited visibility • Manual remaining time update, manual burn down update

Old-school project management tools

Most project managers are familiar with traditional tools like MS Project and it is natural for them to plan iterations using well known tools. However, this desire may hurt agile adoption.

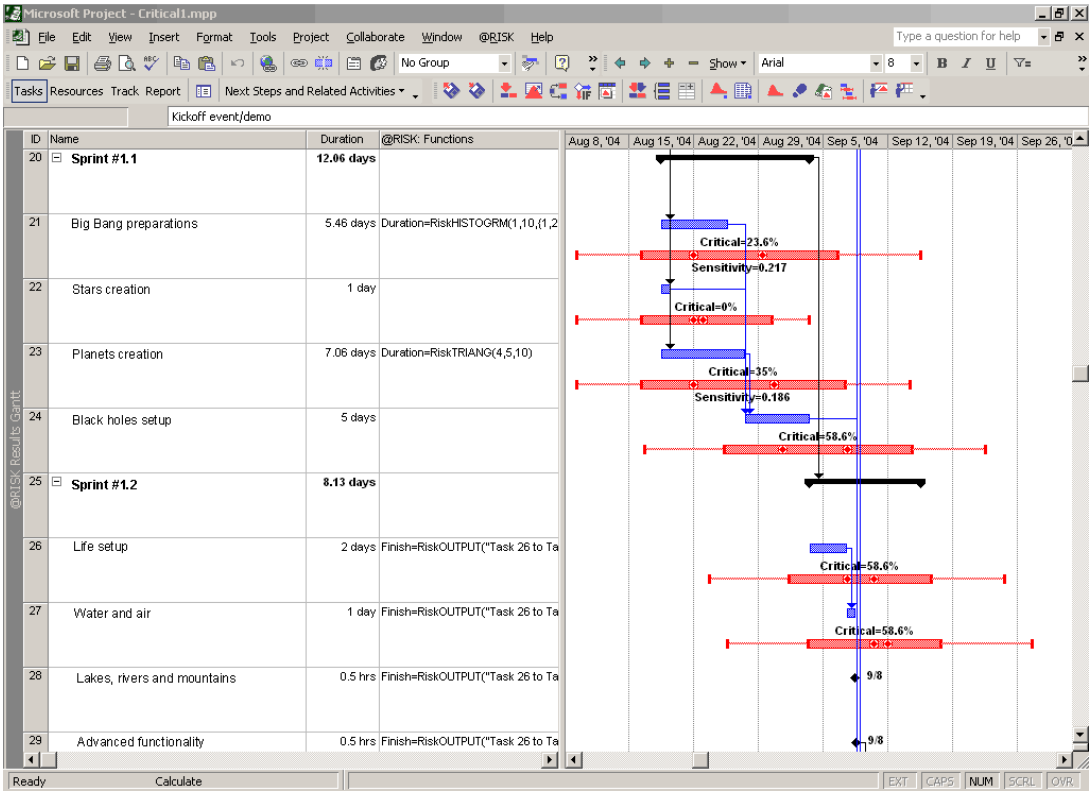
What does classical PM software do and what should agile project management software do?

Classical PM Software	**Agile PM Software**
• Schedule a series of events • Manage dependencies between events • Schedule people and resources • Deal with uncertainties in the estimates of the duration of each task • Arrange tasks to meet various deadlines • Calculate critical paths	• Product/Release backlog maintenance • Release and Iteration plan maintenance • Burn down reports • Iteration velocity concept • Task Board

Let's evaluate how classical PM software functions correspond to the needs of an agile team.

- **Schedule a series of events.** The schedule of events is largely determined by the priorities of the Product Owner and is negotiated from sprint to sprint with the developers. It is not required to set dates for tasks and user stories.
- **Manage dependencies between events.** Dependencies are not required most of the time. Software development teams may handle dependencies inside without any additional tool since they are pretty obvious.
- **Schedule people and resources.** That may be helpful for large projects with several development teams.
- **Deal with uncertainties in the estimates of the duration of each task.** The estimates are handled in two levels of detail - 1) seat of the pants 'story points' for rough sizing and scheduling at the release level, and 2) very detailed task estimates for monitoring progress through a sprint. There is no need for additional uncertainties analysis.
- **Arrange tasks to meet various deadlines.** The only deadlines are sprint demos and they are fixed by the sprint rhythm.
- **Calculate critical paths.** Critical paths are not important for single iteration. The development team "feels" it during the iteration planning and daily scrum meetings since the iteration is usually short. High level critical paths may be helpful for large projects with several development teams, but still it differs from the usual critical paths in traditional PM tools. It may include just releases and iterations, not tasks.

Iteration Plan in MS Project



*"The trouble with Microsoft Project is that it assumes that you want to spend a lot of time worrying about dependencies... I've found that with software, the dependencies are so obvious that it's just not worth the effort to formally keep track of them." — **Joel Spolsky***

Let's see how traditional PM tools support required agile practices:

- **Product/Release backlog**. Just do not exist in traditional PM tools. You will have to store backlog in Excel or somewhere else and copy/paste features when required.
- **Release and Iteration plan** maintenance. May be done in traditional PM tools without problems.
- **Burn down reports**. Just do not exist in traditional PM tools.
- **Task Board**. Just do not exist in traditional PM tools.
- **Iteration velocity concept**. If you create iteration in traditional PM tools, Effort will be Iteration Velocity in fact, but you don't have a clear velocity progress picture.

In short, the tool needed for successful agile project management should be the opposite of what classical PM tools have to offer. It should compensate for classical PM tools' weaknesses in an effort to avoid frustration and gaps in communication among ALL of the team members.

How many features in MS Project will you use for Iterative Development? Resources and Tasks — that's it. Tasks will be independent, so critical path is useless and Gantt chart is not very helpful. Resource leveling sometimes helps, but in most cases conflicts may be resolved very easily during daily meetings. Most reports are useless.

It appears, that **traditional PM tools are too complex and don't bring enough value to justify its usage in iterative planning**.

Pros	Cons
• May already exist in a company • People allocation support	• Does not support agile development concepts • Limited reporting • Limited visibility

Agile project management tools

The integration of the entire planning and development process is the major argument for a dedicated tool. Almost any development process includes activities like:

- Requirements management (product/release backlogs).
- Planning (release/iteration planning).
- Tracking (project/release/iteration progress tracking).
- Quality Assurance (testing, bugs management).
- Feedback Gathering (feedback from customers, ideas, issues).

Most teams use several tools to manage all these activities. For example, the toolset may include MS Project, Requisite Pro, Bugzilla, Test Run and Support Forum. Each tool does a part of the job, but it takes time for people to export/import data between applications and often slice and dice data manually to have the complete picture. The solution is an integrated project management software that supports all development activities and provides information in a single shot.
Modern agile project management software combines common activities and provides open API for advanced integration. It powers:

- User Stories and Epics management.
- Backlogs prioritization.
- High level release planning and low level iteration planning.
- Progress tracking via virtual burndown charts, Task Board and Daily Progress.
- Tests management via Test Cases support and integration with automated testing tools.
- Bugs management via Bug Tracking support and integration with external bug tracking tools.
- Customers' requests management via Help Desk functionality or integration with third-party tools like Salesforce.

In short, you get support for an integrated process (Lean Principle at Work: Eliminate Waste, in particular, searching for information), starting from preparation through deployment and operations.
The other goal of agile project management tools is to integrate several teams. In the real world, several teams work together on a single large project. Often the teams are separated by continents and time zones. It is really hard to integrate them without a tool that can be easily accessible by all teams all the time. Paper and whiteboards will not help; desktop applications will not help either. The only viable solution so far is the web based software.

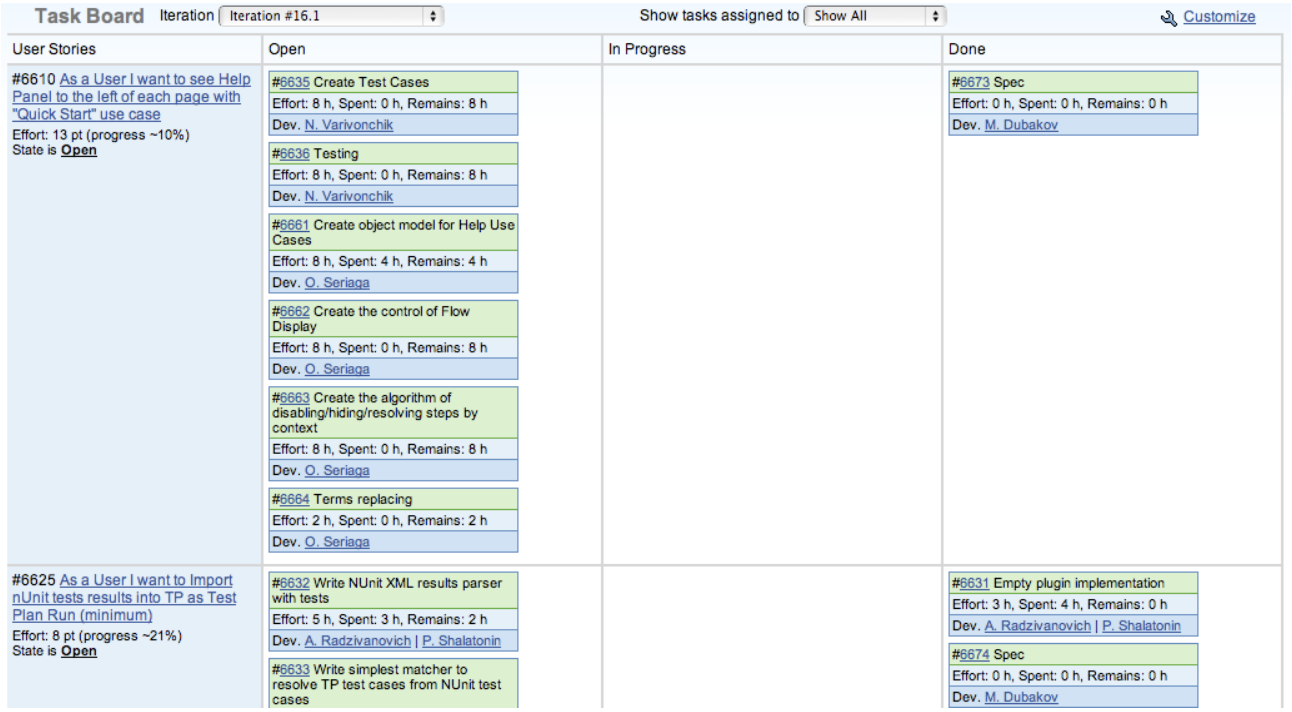
There are various opinions about distributed teams, including extreme:

> *"Distributed teams are not teams; they are at best a collection of people who communicate regularly. But communication is not collaboration. A distributed team cannot create the kind of energy that comes from human eye contact, from shared spontaneous laughter, from physical touch. True collaboration requires all five senses. Distributed teams require managers, and thus can never be truly self-organizing.*

*Time differences and delayed response times inevitably slow down conversation, hold up decisions and ultimately cripple agility." — **Tobias Mayer***

Without a doubt, is it better to have collocated teams. However many companies are distributed geographically for business reasons and software development projects will be executed by distributed teams. There are ways to mitigate the problem. For example, collocate teams for the first release, exchange team members on a regular basis, use web cams and interactive software for meetings. Web based agile project management tool is one more way to reduce the negative effect of distributed teams.

Task Board in TargetProcess



What is the downside of an online tool? Mostly reduced flexibility compared to cards. A physical taskboard is a much more effective Information Radiator than a web page and cards are extremely productive for the kind of group work that occurs in Scrum — brainstorming stories, sprint planning and the retrospectives. A program structures information the way it was designed to. If your needs are different or simply not envisioned by the tool, then you have to figure out a work around.

Pros	Cons
• Works for distributed teams • Works for large teams • Real-time reporting • Integrated solution, provide connectors to source control, but tracking, etc.	• Limited visibility • Sometimes hard to adopt for existing development process • Significant learning curve • Maybe quite expensive

When is whiteboard better than everything else?

Let's try to compare the simplest tools with the web-based tools for the collocated team. Some areas are definitely more important than others. Obviously, communication is more important than fancy plan updating or automatic time tracking — therefore we'll use weights to assign relative values. There are three weights (1-3) and four scores:

1 - Poor
2 - Average
3 - Goods
4 - Great

The formula is quite simple: Category score = Weight * Score.

Total score is just a sum of all categories' scores. In the end we expect to have some numbers that we will use in our analysis.

Category	Weight	Simplest Tools	Web-based Software	Spreadsheets
Planning process	3	4 - Tangible and exciting	3 – simple, but less exciting and visible	2 - doable
Plan visibility	2	2 – good for the team, poor for execs	2– good for execs, poor for the team	1 – poor for all
Plan update	1	3 – re-stick some notes	4 – several clicks, from anywhere	4 – move some rows or mark them for release
Velocity tracking, Time tracking	2	1 – manual, asking each person	4 – automatic	2 – manual, asking each person
Burn Down Update and other charts update	1	1 – manual	4 – automatic	4 – automatic
Communication	3	4 – just great	2 – exists	1 – no
Reporting	3	1 – poor reports since all data offline	4 – almost endless reporting capabilities	3 - good reporting capabilities
People involvement	3	4 – everyone involved	1 – may become a problem	1 – may become a problem
Cost	2	4 – almost free	1 – may be quite expensive	4 – almost free
Total:		**57**	**52**	**43**

We have 57 for tangible tools, 52 for web-based software tools and as little as 43 for spreadsheets. Are you still using spreadsheets? Although the above scores are somewhat subjective, it is still clear there is a better way than spreadsheets or other tools not designed for Agile.

Tangible tools are more preferable for agile processes — use them if you can! But remember, that it remains the case for collocated teams only! It is impossible to use usual tangible tools for distributed teams. Distributed teams hardly can share whiteboards and task boards. They need more formal processes and more formal tools. Agile offshore development (http://www.martinfowler.com/articles/agileOffshore.html) is there and definitely is better than the traditional waterfall process. More and more distributed teams use agile development processes successfully and that is the fact of life. What should you use in this case? Obviously, web based software is a great tool for sharing knowledge, project state and other project information. It coordinates distributed teams nicely.

In fact, there are some guidelines that you may think about when reviewing the comparison table.

You should prefer White Boards, Cards and Markers if:

- You are **trying Extreme Programming or Scrum for the first time** (that's important).
- Team is collocated.
- You tried simplest tools, achieved great results and don't feel a need to change anything.

You should prefer a web-based software tool for project management in agile projects if:

- You have a distributed team (offshore development).
- You have a large collocated team (20+ people).
- Your executives need status reports and without them would not accept the agile process ("OK, if I can see real-time project status somewhere, you may try XP/Scrum/Lean").
- You tried simplest tools and for whatever reason they did not work in your environment. (In that case, you should try to define exact reasons for the failure. Perhaps, there are communication problems or something else that can be fixed).

Another important question is how can we alleviate potential problems that may arise with web-based tools for project management in agile environment? Some thoughts:

Focus on communication and collaboration. Do not think that email is OK and enough. It is just not true. Use Skype, use web cams, whenever it is possible — travel and meet in person. Try to break the borders in all possible ways.

Plan iterations as usual — using cards! Enter data into the system at the completion of the planning game.

Integrate information radiators with software. For example, you may bind Ambient Orb to project status. It will be green if everything is OK and red if iteration is in trouble. You may print out iteration plan on A3 paper and stick it to the wall.

We may combine results in a small matrix.

	No Status reporting required	Status reporting important
Small Collocated Team	Tangible tools	Mix of tangible tools and web based agile tools
Large Collocated Team	Mix of tangible tools and web based agile tools	Web based agile tools
Distributed Team	Web based agile tools	Web based agile tools

Conclusion and Predictions

> *"A trend that will continue to influence software tools is ever-tightening release cycles. Where releases once took years, an increasing number of software products will release new functionality to production monthly, week, daily, or even more frequently. […] The trend towards support more frequent transitions between activities will continue. More activities will be supported without large changes of context."*
>
> ***—Kent Beck***

Agile project management tools have a short history. It is obvious that current tools are just a first try and they will evolve in the future. Currently three main trends may be mentioned.

Tangible – intangible linkage

Obviously, teams like tangible tools and agile software tools will use something tangible to provide better user experience. Large sensor displays like Microsoft surface (http://www.microsoft.com/surface/index.html) will be used for iteration planning, daily meetings, and other interactive meetings.

Complete integrated development life-cycle solutions

Agile project management software is evolving into complete life-cycle solutions. Development teams need a platform that combines and exposes information about all aspects of software development, from initial requirements to unit tests results and source code commits.

Distributed collaboration tools

More and more companies will have development teams worldwide working on the same projects. Distributed teams are a reality in a software development world and this trend will be progressing over time. Agile tools will focus on distributed teams support better, providing integration with communication tools like Skype and WebEx.

Author Biographies

Peter Stevens

My mission is to help you realize complex projects effectively. I provide coaching, training and project management to help you get started with Scrum, save projects in crisis, and make your IT operations leaner and more effective. I studied Computer Science at Colgate University and started my professional life at Microsoft. I am a Certified Scrum Master.

Michael Dubakov

I am a founder of TargetProcess ("best agile project management software in the world"). My Mission is to provide solutions to real problems in agile projects. I wrote several books about web development and many articles related to almost all aspects of software development.