# In Search of the Perfect Project Management Tool

**By Dmitry Stillermann**

The market for project management tools, specifically those targeted at the software development industry, is in a very peculiar state nowadays. Every available solution is either lacking some rather critical features, or is cumbersome to use, or both. Basically this means that any newcomer that addresses all of the shortcomings of the existing products can easily grab a huge market share from the competitors.

This situation is similar to how the market of integrated development environments (IDEs) looked during the late 1990s, just before the arrival of IntelliJ IDEA (http://www.jetbrains.com/idea/). When the first version of JetBrains' IDE came out, it instantly became a highly disruptive innovation that redefined both the market and the industry's perception of an IDE as a product class. The sheer scale of this event was so great that Martin Fowler, a renowned industry guru, proclaimed that the world had entered "the post-IntelliJ era" (http://martinfowler.com/bliki/PostIntelliJ.html).

So, what exactly is wrong with existing software project management tools? As it happens, there is actually no single answer to this question, because the market is so diverse.

### Gantt Charts: Your Dad's Project Management

On the high-ticket side of the spectrum, there are numerous "enterprise-y" product suites, usually marketed under the umbrella term "Application Life Cycle Management" (ALM) or "Enterprise ALM." These usually support the full range of project-related functions and activities, including product and requirements management, traditional project management, QA management, and defect tracking, configuration management, budgeting, and procurement. The typical problem of such suites is that their vendors have usually assembled them from several parts that were originally unrelated, often through a string of acquisitions, and these parts are not always integrated well enough. Also, because the "feature completeness" is a big sales driver for this kind of product, the vendors too often end up overloading the products with features and not paying attention to usability. I believe that any person who uses a typical enterprise ALM product on an everyday basis ultimately becomes highly dissatisfied.

"There are numerous 'enterprise-y' product suites, usually marketed under the umbrella term 'Application Life Cycle Management' (ALM) or 'Enterprise ALM'."

Another problem with high-end enterprise suites is that they are rooted in a more traditional school of project management. Until now, formal project management training has been generally organized in accordance with *A Guide to the Project Management Body of Knowledge (PMBOK® Guide),* an international standard (Project Management Institute, 2008). In the world of project management tools, the embodiment of this style is the ubiquitous Microsoft Project. The problem here is that this approach is too generic and industry-neutral, and the realities of any specific industry (whether information technology, construction, or pharmaceutical) don't always perfectly fit the ideal model of the *PMBOK® Guide*. MS project is organized around the representation of project plan as the Gantt chart—a hierarchical breakdown of tasks and

activities, usually with strict dependencies and fixed timing and cost for each task. However, Gantt charts capture real-world software development very imperfectly. Compared with a neat Gantt chart, software development at the "micro" level is a mess—it is nonhierarchical, dependencies are often weak or nonexistent, tasks can go on in parallel or be interrupted, timing is never accurate, and resource allocation may often change the whole plan (for instance, the famous Brooks' law that states that adding people to a late software project makes it later doesn't actually often apply to construction).

All this being said, there is no escaping the burden of detailed long-term planning if you're dealing with the stringent nature of fixed-price contracts. The traditional, top-down approach is inevitable for a complex orchestration of projects that bring together diverse teams from multiple organizations, governing their relationships by means of strictly defined contractual obligations. The overall performance of such a framework may be less than stellar, but in this case predictability (real or perceived) is often more important than speed, adaptability or, often enough, even quality. Thus, dependencies and milestones become the key focus of managers' attention and all kinds of "crystal balls" are employed for the purpose of predicting the state of things a year in advance. Gantt charts naturally support this approach exactly because they capture the world in terms of hard dependencies, predefined timing and costs.

### Backlogs: Lightweight and Cool

It should come as no surprise that the most convenient, intuitive, and usable tools have grown out of the practices adopted by the people who are "out in the trenches." For many years, the first step for a self-organizing software team towards a centralized, well-managed project bookkeeping has been using a bug tracker of choice as their project management tool. Even before the agile became fashionable, software teams have felt that a bug tracker's worldview is actually a very good representation of a small software project. A bug tracker is essentially a to-do list that can be prioritized, estimated, assigned to developers for handling, and used for all kinds of monitoring and reporting purposes via clever configuration and querying. Really good bug trackers allow the project managers to fine-tune the bug life cycle to better model the particular way to do business that has evolved within a specific team or company. So, it is only natural to begin tracking not only bugs, but also regular parts of the project scope. This trend actually led some of the bug tracking software vendors to start marketing their products as "project management" solutions. One well-known example is FogBugz from Joel Spolsky's company FogCreek (http://fogcreek.com/FogBugz/) that was born as a bug tracking system and then was rebranded as a "complete project management system designed to help software teams communicate." FogCreek, however, has some real content behind this claim: They not only changed the product's positioning, they also added some very smart features, such as Evidence-Based Scheduling (EBS), which puts FogBugz together with some of the best-of-breed members of the new project management tool generation.

The rise of the agile lifestyle and the widespread adoption of agile or semi-agile processes and practices have also had a great impact on the project management tool market. Probably the best tools on the market today are firmly rooted in the agile model. The best example is most likely Rally (http://www.rallydev.com/), which provides superb agile project management capabilities and at the same time offers a free-of-charge edition for small teams (up to 10 users). Other well-known players are CollabNet TeamForge (http://www.open.collab.net/products/sfee/) and XPlanner (http://www.xplanner.org/), an open-source tool. The best thing about these tools is that they provide the best fit for small-to-medium projects (up to a few tens of people) and also use the best practices that agile methodologies made popular. They are usually highly usable and sufficiently decrease the amount of time the project manager and the team have to spend on mundane bookkeeping tasks. The learning curve is probably more or less the same as for Gantt-centric tools, but once it has been learned it, it is very natural to follow the underlying approach.

So then, what's the downside of sophisticated bug trackers and agile-based project management tools? Unfortunately, it's that they don't capture the full complexity of the real world. Being perfectly suited to small-to-medium teams and relatively simply organized product life cycles, these tools often fail the project managers when they need to work in a broader organizational context or with large cross-organizational teams governed by intricate contractual frameworks, or with more complex product design workflows. The agile tools are great at the "micro" level, but they are not always adequate at the "macro" level, where there are hard deadlines and dependencies, elaborately defined acceptance criteria, complex sign-off workflows, or even so dull a task as cost management done properly. Once we add to the picture the need to integrate with product management and marketing, where the concept of "feature" is often very

different from the "user story" or "work package" at the development level, we begin dealing with complexities that the agile project management tools simply cannot support. In this case, project managers resort to tried and trusted MS Excel, MS Project, and all kinds of home-grown tools that flourish in each mature organization.

It is also well known that a pure agile model is not very well compatible with a fixed-price, fixed-scope type of contract, which makes project managers' lives even harder—they actually have to do their planning and monitoring work in two, three, or more environments, depending on whether they look at the project at the "micro" or "macro" level, and on which particular aspect of the project they are working at the moment. This not only hinders productivity and is very inconvenient, it is also very error-prone and potentially dangerous for the projects and companies that run them.

## How To Deal With It All

The world of project management tools is obviously far from perfect. Where does this leave project management practitioners until a revolutionary new tool redefines the area? And what advice can be given to them?

The recommendations below may sound a bit mundane and boring, but they have been learned and proved by years and years of several managers' real experience.

- **Be aware of your project's context and priorities.** Establish a clear understanding of what is truly important for your customers, your company, and your team, what commitments have been made, and what everyone's expectations are. It makes little sense to focus on technical and functional excellence if time to market or overall budget is strictly constrained. Conversely, an overhead of detailed planning is hardly justified if a rapidly shifting business demand must be continuously satisfied. Identify the variables that need to be optimized and base your thinking on them.
- **Decide what data you need to track.** A proper information system design process should take into account what kinds of data the users will need and how they will use them. As a project manager, you design your project's information system by deciding what data you will track and what questions you're going to ask about the data. This is clearly related to project priorities: You may need very different data to be able to say when the project is going to finish, depending on your acceptance criteria. Status reporting requirements, either stipulated in a contract or in a project communication plan, or defined in your corporate policies, are another big source of data tracking requirements.
- **Select the right mix of tools for the job.** Based on your context and needs, make a conscious selection of tools that will support you, the team, and the stakeholders. Don't instinctively fire up your favorite tool just because it worked well on your previous project, or because everyone around you uses it. Don't constrain yourself to a single tool or even a single "toolbox"—it may be more optimal to perform some things using only those tools best suited for them. Have every tool in place by the moment you need start collecting and tracking the respective type of data.
- **Designate one tool as primary.** You need one definitive source of project information, referenced by all other tools and systems. Otherwise, you will either lose track, or you will need to regularly spend extra effort on synchronization, or both. It is possible to use two independent systems for a limited period of time, but you certainly want to ultimately move to a single source situation. Often, if your customer strongly insists on using their own project management system, it is better to organize your process around it instead of setting up a complex synchronization with your favorite tool, even if your tool is clearly superior. Any centralized registries, such as issue tracking databases or agile project management tools, are often ideal candidates for the primary project data storage: Besides allowing you to have all of your project scope in one place, they give you a convenient identification scheme for all of your tasks, which then can be referred to by number or ID.
- **Beware of unnecessary overhead.** Your job is challenging enough—don't add on an additional paperwork burden, especially if no one is going to need it. Confine yourself to what actually helps you run the project or has been demanded by the contract. Usually, one of the most time-consuming activities is preparing detailed status reports according to the peculiar needs of many powerful stakeholders. Try to do as much as possible using the built-in reporting features of your primary project management tool. What can't be supported directly, is almost always possible to do using raw data extraction and some Excel magic; if you're not an Excel/VBA guru yet, a little self-education in this area would be your best investment. One special area where most popular tools are relatively weak is project metrics' trends over time. In this case, sometimes the best solution is to preserve the

time series in a separate Excel spreadsheet and establish a daily routine to update them.

- **Watch the changes in the project context and react accordingly**. Especially interesting to you as a project manager are changes that affect your process and your information needs. These often lead to changes in data tracking routines and may require new tools being introduced or the tools already in place being applied in new ways. One very typical example is moving from active development into a user acceptance testing and stabilization phase; despite their similarities, bugs and user stories are treated and measured differently. The progress in the development phase is usually measured by techniques like Earned Value Management or burndown charts. When you move into UAT and stabilization (if you're in that kind of project), everyone starts to be concerned about bug counts and the corresponding trends.

> "Project managers can determine the best mix of tools and processes for their particular projects by staying alert and aware of what the project context requires."

## Conclusion

The future of software project management tools, it seems, is going to be defined by a future product that will bring the same level of usability and conceptual fitness to the "macro" level of software project that the agile project management tools have brought to the "micro" level. The software project managers of the world are waiting for their own IntelliJ IDEA, but, unfortunately, none can be seen on the horizon. Meanwhile, however, project managers can determine the best mix of tools and processes for their particular projects by staying alert and aware of what the project context requires.

## About the Author

Dmitry Stillermann is vice president of Enterprise Project Management at DataArt Solutions, Inc. He supervises application development projects spanning multiple organizations, functions, locations and time zones, and, as a member of PMO, promotes best practices for project delivery.