Plans are Useless, but Planning is Indispensable

Lisamarie Babik PMP
Software Factory Floor Manager, Menlo Innovations
lbabik@menloinnovations.com

Abstract

Project planning at Menlo is a series of ongoing activities, rather than a single event at the beginning of the project. In addition to a project baseline that is established at the start of each project, the project plan is regularly revisited and actively modified throughout the duration of the project. This iterative process actively acknowledges that new data comes in from the team and the sponsor, in the form of re-estimated work packages, new priorities and changes in direction from the project sponsor. This process is meant to stimulate rather than replace the important conversations that must happen during the life of the project, something that a static requirements document sitting unread on a shelf does not produce.

Introduction

"In preparing for battle, I have always found that plans are useless, but planning is indispensable."
–General Dwight D. Eisenhower

When an organization suffers a significant project failure, they often look to improve project execution by adopting more formal project management practices. As a first step they often begin to construct Gantt charts, Activity Sequence diagrams, simple To Do lists, or some other form of task and resource tracking. These documents are often posted on the wall or stored in a notebook, but seldom referred to when attempting to answer questions about the project.

Any static plan that is reviewed, approved, bound and then filed on a bookshelf or in a cabinet is useless when unaccompanied by an on-going process of revisiting and updating the plan. This truth is even harsher if "the plan" is the work of a lone expert, because the true value of the planning process occurs in the discussions and alternatives explored by multiple stakeholders and the shared vision that these discussions help to form. The value of a plan lies not in the paper on which it is recorded, but in the collective and on-going actions that comprise the planning process itself.

I have worked in several software development organizations that used many different styles of planning, from a large steel factory that slowly moved forward for three years on a project where a chief developer marshaled out tasks from a plan that existed only in his head, to a small statistical process control company where all project "planning" was conducted by the sales team with the goal of closing the next deal. The former produced a single point of failure where all decisions were weakly driven by a single individual, while the latter suffered from diffused decision-making authority – a sales team intent on closing "the next big deal." This type of project "planning" left open to interpretation issues such as quality, cost, and scope to the person performing the work. For example, in both cases, the development team assumed that if it was scheduled, it must be completed regardless of cost and/or delays. This was, of course, not always the case.

I now work at a company where planning is not a special event held only at the beginning of the project. Instead, it is a formal process that occurs each and every week, and is a forum in which multiple stakeholders with different points of view openly collaborate about the tradeoffs that must be made in order to make the project successful.

In this paper I will outline the steps in this weekly process, the participants, the activities, and the significant benefits that are the result of a regularly scheduled planning process. By formally planning more often, I have learned to improve project communications management by having fewer meetings, planning frequent milestones as small early deliverables, communicating requirements clearly but tersely, keeping executives informed and enthused with

Originally published as a part of 2005 PMI Global Congress Proceedings – Panama City, Panama.

simple and approachable status tools, and using a tight feedback loop to control expectations and improve overall planning skills in the entire team.

The Basic Timeline

In our process, the Planning-Executing-Controlling cycle occurs formally over the course of a short time period that we call an iteration. Each iteration in the project lasts anywhere from one day to two weeks, depending on the project sponsor's preference and the duration of the project. Our projects typically run in 1 week iterations, enabling us to establish a simple repeating pattern consisting of updating requirements/deliverables definitions (i.e. define work packages), estimating, prioritizing, authorizing team members to work on specific work packages, doing the assigned work, providing tracking details through the iteration, demonstrating the results of the iteration to the stakeholders and team members, and then starting again. It's the project management equivalent of "lather, rinse, and repeat." Bob Martin, one of the key figures in the agile software process community, even goes so far as to describe agile software developers as "planning addicts" who "plan, and plan, and then plan again," (Martin, 2005) with each revision of the plan being improved with the knowledge gained during the most recent execution effort.

Figure 1 shows how these iterative processes fit within the five PMBOK process groups. Note the repetition in the Planning-Executing-Controlling loop.

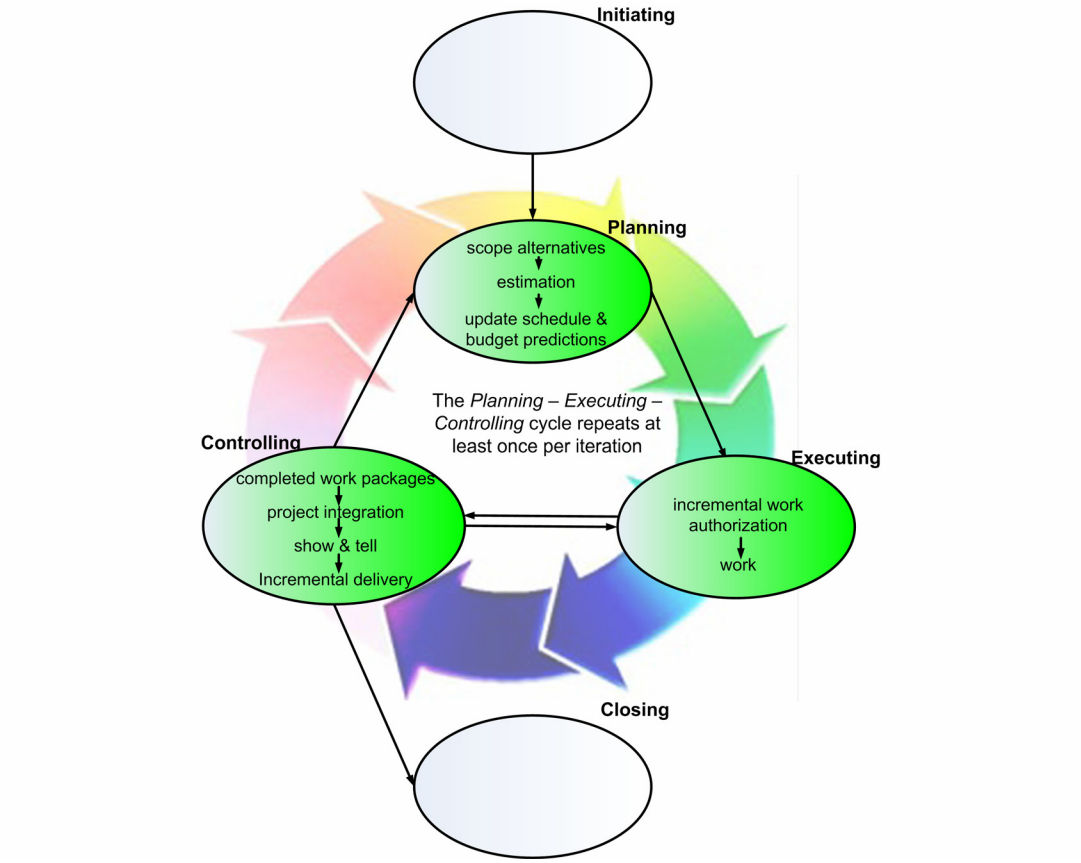

Figure 1: Iterating Through the Process Groups

Originally published as a part of 2005 PMI Global Congress Proceedings – Panama City, Panama.

Effective Work Package Definitions are Key

When project planning is taught formally, there is an emphasis placed on the Work Breakdown Structure and Work Package definitions. But in practice many project schedules are built around a list of activities instead of deliverable work packages. This is often because the deliverables are not well defined at the beginning of the project. As each planning cycle begins all of the work package definitions are reviewed, updated, and revised as necessary. These improved work package definitions lead to improved estimates and improved outcomes. Thus, the definition of the work of the project, and indeed of the project itself, improves as the team gains information and experience.

In our environment each individual work package is captured on an index card that describes something that the end user will be able to see, do, or otherwise appreciate when it is completed. We call these index carded work packages "story cards" as they are short stories describing stakeholder value. Story cards are often associated with pictures of the desired outcome, or frequently in our case with screen mockups of the desired software functionality.

A typical story card (**Figure 2**), in this case the card written to produce the first draft of this paper, is included below:

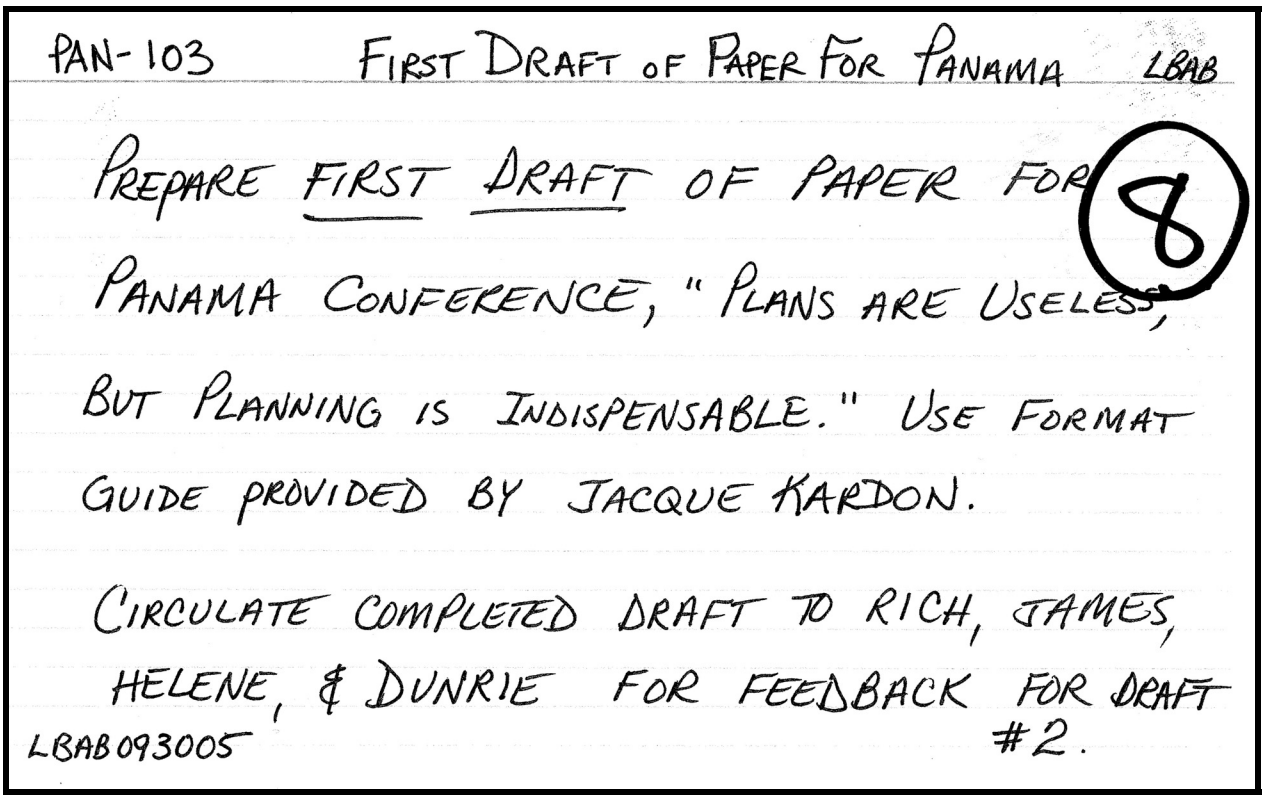

Figure 2: Sample Story Card

The fundamental elements are simple: a unique number, a title, and a description. We have historically found that handwritten story cards (i.e. work packages) tend to be better thought out. Dunrie Greiling – one of our project managers – has further suggested that handwritten cards appear negotiable in a way that printed cards do not and invite the conversations they are meant to evoke. We therefore prefer to handwrite all our story cards.

These cards may be as brief as "First draft of paper for Panama" or they may include mockups or elaborate illustrations. Ultimately, a story card is a placeholder for a conversation between the person who wrote it and the person who is going to do the work. Both should be able to consistently answer the question: "How will I know when I am finished?"

It is important to note that anyone can write a new story card at any time. Many organizations fear that this will lead to scope creep; however, articulating a work package idea is not the same as scope creep. The distinction lies in which story cards are scheduled and authorized.

For example, when technical experts write cards they are often difficult for other stakeholders to understand and discern the value of because the language used to describe the work is a barrier to communication. An example, as written by a software developer, might be "Reduce the number of SQL calls at startup." This story card is not automatically added to the schedule.

Rather a quick series of questions jumps to mind. *Why should I choose to schedule this story card? Are a small number of SQL calls relevant to "good" or "bad" code? If nothing changes, what's the impact?* When asked why we would do this, the developer said simply, "Oh, it'll reduce startup time from 7 seconds to 2 seconds." Therein lies the value for the sponsor, that's why they would choose to schedule the story card. The value to the stakeholder lies not in the number of SQL calls, but in the decreased startup time.

As initially written, without a quantification or qualification of the number of SQL calls to remove, "reduce the number of SQL calls" cannot be estimated reliably. One team may interpret it as "decrease by 1 call," and give it a small estimate; another team may interpret it as "decrease the number of calls by half," and give it a larger estimate. Until it can be understood by all, it simply cannot be estimated in a meaningful way. Accounting for this new information, the new story card might be rewritten like this: "Decrease application startup time from 7 seconds by reducing the number of SQL calls. It is anticipated to reduce startup time to 2 seconds" – something that anyone, regardless of technical expertise can prioritize.

Estimating

The final element that a story card requires before being scheduled is an estimate. The process of estimating is surprisingly simple: gather story cards (i.e. work package definitions), bring the team together (**Figure 3**), read the cards aloud, collect estimates. The amount of time required for this will be affected by the number of story cards to estimate and the number of times the cards have been previously estimated.

As with all things, estimating gets easier with practice, but let's be clear: you are never going to get accurate numbers. At best, you will get "good enough numbers." (Beck, 2001) A good enough number will enable you to prioritize stories, reduce the scope as required, and identify when there is simply too much to do (versus "not enough time").

When we estimate, we are not so much interested in the exact amount of time it will take to complete a task, but rather in its order of magnitude of time. For example, we may use an estimating scale such as the following: 1 hour, 2 hour, 4 hours, 8 hours, 16 hours, and 32 hours. This is a "doubling scale," though the scale used in a particular project may be measured in hours or weeks, depending on the iteration length and the tasks on the project.

Figure 3: The whole team comes together to estimate. The project manager reads the story cards aloud and the team quickly moves through estimating. When a pair stops for too long talking through a problem, the first question asked is, "Does it affect your estimate?" If not, the pair is encouraged to choose and estimate and then the team moves on.

The project manager provides the team with estimation worksheets. Then, each pair within the team circles their estimate and moves on to the next card. To blunt the effect of pair-to-pair variation, we use a "consensus estimate" for planning. This value is selected so that in scheduling, the maximum number of teams would be capable of completing the card. For example, if I had the following estimates from five teams for the story card PAN-103:

Table 1: Estimates for PAN-103 from 5 different teams.

Team A	**1 hour**	2 hours	4 hours	8 hours	16 hours	32 hours
Team B	1 hour	2 hours	4 hours	**8 hours**	16 hours	32 hours
Team C	1 hour	**2 hours**	4 hours	8 hours	16 hours	32 hours
Team D	1 hour	2 hours	**4 hours**	8 hours	16 hours	32 hours
Team E	1 hour	**2 hours**	4 hours	8 hours	16 hours	32 hours

I would select a consensus value of 2 hours for planning with the sponsor. By choosing "2 hours" as my consensus estimate, I have ensured that I have at least three teams who can complete the story card in that amount of time. The interest lies not necessarily in the specific estimate of any one team, but rather in the group's estimate.

Though consensus estimates are used for planning, when story cards are actually assigned, each team is assigned the amount of time to complete the card that they estimated it would take. So, if one of the teams that had estimated "4 hours" is assigned the card that had a 2 hour consensus estimate, they are allocated 4 hours to complete the card. This sometimes makes for interesting math challenges when constructing the weekly project plan. However, it maintains the trust and buy-in of the team, as they are not assigned tasks with estimates obtained from another pair, or those that simply fit the budget. And, it is pragmatic: groundless numbers are not used to manufacture an unrealistic budget for an iteration or a team that no one accepts or believes.

Planning Game: Forcing Tough Choices

In most projects, planning is something that occurs a single time: at the beginning, when the project manager is trying to determine when the project will be completed and the level of resources (people and equipment) that will be required. In fortunate situations, the team is allowed to estimate their own tasks. In unfortunate situations, a project manager (or other "expert") estimates the effort for each task and team members have to live with those estimations.

In an agile environment, this is quite different. Planning and estimating occur at the end of each iteration – be it a day long or two weeks long. (See **Figure 4**.) Common arguments about why this process cannot work are that a project is either too big or too small, or too long or too short to be replanned so frequently. At Menlo we can offer direct evidence to challenge these assumptions: we frequently run teams of 4-16 people on projects lasting days to weeks to months to years, estimating and replanning after each weeklong iteration. Once we even ran a day long project with two developers that estimated and planned every hour.

In the three years I have been working with Menlo, it has been my experience that generalized "too big" and "too small" arguments fail to hold up to scrutiny. More typically, I believe that such protests are the result of project managers who have not been able to run the experiment, or who have been lulled by a false sense of security into avoiding difficult choices until the very end of the project, when it is too late to have many choices at all.

Figure 4: At the end of each iteration, the project manager and project sponsor meet to review all the story cards and to plan for the next iteration. It is not unusual for the project to be completely replanned at the end of each iteration.

Using agile planning techniques does not wholly insulate the project manager from these problems. Rather, as estimates change throughout the life of the project, they will be faced with an all too familiar problem: more work remaining than time. The difference is that rather than being surprised by it very late in the project, the project

manager works with the sponsor throughout the life of the project to schedule and prioritize stories, deciding which stories are most important to the stated project goals and explicitly choosing which stories will *not* be completed. If a new story card must be scheduled, it is handled in one of two ways: 1) some other piece of work is removed to make room for the required task, or 2) additional resources are temporarily added to the project to complete the work.

Show & Tell

At the end of each iteration, the whole team assembles for a ceremony we call "Show & Tell." This is an opportunity for the team to demonstrate the work they've completed during the iteration. This serves many purposes: 1) reinforcing the planning by demonstrating actual work completed, 2) building a shared vision of the product being built, 3) providing an open forum for tough discussions on functionality and changing needs, and 4) reinforcing the building and delivering of business value each week to the client.

Conclusion

The project management process described is not unique to software development. Consider any profession where planning is involved before execution, surgeons for example. Before conducting surgery they plan, trying to think of every possible detail, they imagine everything they will need for the surgery – but they do not stick to the plan to the detriment of the patient. No one is excited about perfect plan execution if the patient dies.

The same is true in software project management. If you hit the project management trifecta – on time, on spec, and on budget – but the end product doesn't actually do what is required, is your sponsor excited? They never are. But the key challenge is that most project management practices only ask once. The solution is to ask repeatedly: *What are the estimates? What are the priorities?* It is the process of planning, repeated throughout the life of the project that ultimately leads to a successful outcome.

For this to work, project managers must be willing to abandon the rule of the rod when it comes to estimates. When someone misses an estimate, this is not good or bad – it is purely information that, when shared as early as possible, can be used to adjust scope and schedule to ensure that the most important work is being completed by the team. Continually making these adjustments is what will enable you to deliver projects successfully.

References

A Guide to the Project Management Body of Knowledge (PMBOK® Guide) Third Edition. © 2004 Project Management Institute, Four Campus Boulevard, Newtown Square, PA 19073-3299 USA.

Bob Martin. *Conformance to Plan,* from ButUncleBob.com. Accessed online 9/15/2005 at http://butunclebob.com/ArticleS.UncleBob.ConformanceToPlan.

Kent Beck and Cynthia Andres. 2005. *Extreme Programming Explained: Embrace Change.*

Kent Beck and Martin Fowler. 2001. *Planning Extreme Programming.*

Malcolm Gladwell. Keynote address at SXSW Interactive 2005. Recorded March 15, 2005. From IT Conversations #478, available at http://www.itconversations.com/audio/download/ITConversations-478.mp3.

Malcolm Gladwell. *blink: The Power of Thinking without Thinking.* 2005. Little Brown & Company.

Pascal Van Cauwenberghe. *Agile Fixed Price Projects part 2: "Do You Want Agility with That?"* Accessed online July 28, 2005 at *http://www.nayima.be/download/fixedpriceprojects.pdf.*