Update

Agile Product & Project Management Advisory Service
Executive Update Vol. 10, No. 3

Thoughts on a Project-Volatility Metric:

Part I — Definitions and Assumptions

by Vince Kellen, Senior Consultant, Cutter Consortium

Like the stock market, IT projects can be volatile: requirements can change; scope can creep; unknown dependencies can appear; teams can get mired down in myriad ways; technology can fail; executive sponsorship can evaporate; schedules can jitter; and dates can slip.

Like a rodeo rider, anyone who has managed difficult projects knows that some projects are just meaner and more volatile than others. However, in all my years of managing and sponsoring projects, I have rarely, if ever, seen robust measures of project volatility routinely collected and evaluated. Sometimes, anecdotal evidence may surface, but this is usually after milestones have been missed. Normally, overoptimism reigns, and project team members believe they are just around the corner in providing deliverables on time, when in actuality they are not.

The project management discipline does include reviewing estimate and project variance; however, these two analysis mechanisms don't adequately capture the important notion of project volatility. Standard task estimating calls for establishing an optimistic estimate, a pessimistic estimate, and a most likely estimate of completion time. Estimation variance analysis involves identifying tasks that have large differences between the optimistic and pessimistic estimates. All this does is capture the uncertainty in the estimator's mind, not the variance in the team's ability to deliver on the estimate. Project-variance reporting (which looks at overall, to-date budget and schedule performance) calculates aggregate metrics, which by themselves don't pinpoint the source of volatility and may detect trends too late in the project. Moreover, conventional project-estimation and variance-analysis techniques can often burden a handful of estimators and project managers to the point where the cost of getting low-level and detailed volatility metrics may exceed the perceived benefits.

Obtaining automatic and systematic insight into project volatility can help trigger management intervention earlier when it can make a difference. If volatility can be detected very early in the project lifecycle and assigned to the right potential causes, managers can alter the allocation of resources, architects can rethink the solution design, and users can revisit their requirements without negatively impacting the business benefit needed from the project. But how can managers get a handle on project volatility early in the project's lifecycle?

Here in Part I of this two-part *Executive Update* series on project volatility, I examine different notions of project volatility and explore the assumptions that underlie my own project planning approach.

DIFFERENT DEFINITIONS OF PROJECT VOLATILITY

As I am considering it, project volatility is different than common notions of task-estimate variance. Task-estimate variance, which is the difference between optimistic and pessimistic completion times, does not capture the variance in the actual effort the project team expended to complete the task. For the purposes of this *Update*, I am defining a key project-volatility metric as the variance for the *difference between the task estimates and the actual effort expended to complete the task*. Project volatility, as I am conceiving of it, measures the difference between what the task estimators anticipated would be the effort expended and how much effort the project team actually expended.

Defined this way, volatility then is a measure of estimator accuracy.

Unless objectively measured, volatility is often hard to detect. IT staff members often believe sincerely that the project milestones will be met when they won't. Senior IT managers will often aggressively enforce delivery dates, which can create an environment where staff members capitulate and incorrectly state delivery schedules to avoid confrontation. More experienced IT staff will "sand bag" an estimate, making the task appear longer than it actually is, which creates bias. Experienced project managers certainly can develop an intuitive feel for which projects are volatile and which ones aren't, but junior project managers are more easily fooled. Worse still, senior IT managers who lack sufficient project management expertise can jump to incorrect conclusions about a project's volatility or realize much too late that a project is more volatile than the team thought it was.

Over the years, IT researchers and practitioners have come up with several different ways of measuring project volatility. One way is to count the number of changes in requirements — be they additions, changes, or deletions; changes in prioritization; or changes in the delivery sequence.[1, 2] Another method is to keep track of low and high estimates for tasks (also called "work packages") and rank those tasks with a high range as being more volatile.[3] A third way is to classify projects according to some size and duration estimates. For example, a recent study classified projects likely to not succeed (which we can consider as volatile) as those with long duration (18 months or greater), large team size (>20 FTE), or large effort (>380,000 hours or 2,400 person-months).[4]

All of these approaches for controlling volatility suffer from limitations. For projects that do not follow more linear (waterfall) paths to completion, requirements may be ever-changing and evolving. Just monitoring requirement changes, especially for highly flexible and agile approaches, may not give project managers enough insight to monitor ongoing volatility. On the surface, because of the frequent requirement revisions, these projects may look highly volatile even when they are consistently delivered on time and on budget. The level of requirement changes may not be a good predictor of agile project volatility. In addition, teams and individuals vary in their ability to accommodate project requirement changes.

Projects with high and low estimates for tasks may be only capturing the estimator's volatility and not the actual volatility of the project in action. In order to keep a pulse on ongoing actual volatility, project managers would need to update these high and low estimates. Moreover, good project managers manage slack in the schedule and work to keep as many steps off the critical path as possible. Calculating volatility would require keeping the complex critical path logic up to date as well. Most project managers simplify their models and do not capture all the details regarding critical path changes and changes in estimate ranges.

While avoiding projects with large duration, team size, or total effort is a best practice, how can you manage project volatility for the remaining small-to-medium-sized projects? Duration, team size, and total effort are not specific enough measures and do not provide guidance for medium-sized project intervention, which some projects will still require. Team interactions, project management skill, complexities within the IT architecture, and other factors can and do contribute to excessive project volatility.

ASSUMPTIONS FOR A PROJECT-VOLATILITY METRIC

In my own work, I have developed a project planning approach that perhaps can strike a balance between the desire for project managers to have simpler project plans that require less effort to maintain and the desire for senior IT management to have measurable project performance. A benefit of this approach has been the ability to easily generate a project-volatility measure, which can provide good insight into project volatility early in a project's lifecycle.

This approach has a number of assumptions that may cause some project managers to hyperventilate as they read it. The assumptions are somewhat counterconventional project management approaches, but have their merits. If you find yourself hyperventilating, just relax. These assumptions include:

- **Work packages are independent of each other.** If one creates a project plan that removes all critical paths, what remains are tasks or activities that can proceed independently of each other. Moreover, project teams can get creative in rethinking technical and organizational dependencies and frequently do so in the middle of the project. If work packages are represented in the project management system as independent of each other, the project manager does not have to spend time modeling or monitoring dependencies. Clearly, dependencies do exist, but I have been surprised with how little of the dependencies need to be captured in the project management system with project management control not suffering.
- **Work packages are small.** The optimal size of work packages varies. Small projects of a few hundred or thousand hours can easily afford work packages (individual tasks) that can be eight to 20 hours each, or smaller. Large projects in the hundreds of thousands of hours or more might not be able to afford such small work packages. Conventional thought is that work packages are about 80 hours in size and two to four weeks in duration (or between 0.5 and 2.5% of the total project size).[5] For the purposes of this project-volatility metric, I advise an average work package or task size of 20-40 hours for projects up to about 10,000 hours and then increasing as needed but not exceeding 0.5% of project size. Many projects I have managed in the 10,000-50,000-hour size have average task sizes of less than 0.1% of the project size. For example, a 20,000-hour project with a task-size average of 0.2% of the project has 500 tasks averaging 40 hours total. Projects in the hundreds of thousands of hours will most likely have larger tasks sizes.
- **Projects may have large numbers of estimators.** In order to accommodate a small work package size, estimation needs to be distributed across the various teams. In extreme cases, each independent technician can estimate work packages assigned to them. Those who will actually carry out the work can estimate the effort required. This has a side effect of increasing buy-in to the overall estimate, since the individual estimates are generated bottom up across the project team. For example, a large two-year project of one million hours, 300 people, and an 80-hour average task size will have 12,500 tasks, with each task on average at 0.008% of the total project size. Clearly, these are very small task sizes and a large number of tasks. If one uses a decentralized, bottom-up approach to managing estimates and each estimator can handle about 500 tasks (which my experiences indicates is reasonable), this project will require 25 estimators, again reasonable given the 300-person project team.
- **Work packages are estimated without conscious bias.** Project estimators should develop a work package estimate that reflects the belief that the teams have just as much of a chance of coming under as they do going over the estimate. Sometimes this is called a 50/50 estimate. Project estimators do not need to worry about high or low ranges for the estimate.
- **Everybody enters time on a daily basis.** In order to capture the difference between the estimated effort and the actual effort expended, project team members need to enter time against specific work packages. In order to provide timely and accurate reporting, team members need to log their time daily. In prior work, I have measured the effort required to enter time and have found when time entry is facilitated with a simple and easy software package, less than 2% of the total project time is spent on entering, reviewing, and altering time records and adjusting the project plan. As work packages are completed, they are closed immediately. In this approach, the key insight is the difference between the estimated effort of the work package and the actual effort expended. As work packages are completed, IT managers, project managers, and project team members can begin assessing project volatility. If work packages are small, many work packages will be closed out weekly and enough will be closed out early in the project, allowing managers to more accurately infer volatility.
- **Work breakdown structures can vary.** The mechanism for this project volatility metric lies in the variance between effort estimates and actual effort spent at the work-package level. How project managers choose to break down the work into the packages can vary and, from the perspective of project volatility, are merely convenient ways of aggregating and analyzing volatility at levels between the work package and the project overall or across multiple projects. The project-volatility metric doesn't care about how work is hierarchically broken down. In fact, project managers can apply multiple work breakdown structures (WBSs) to the same work packages. In this regard, a WBS is an example of a taxonomy. A project that contains a set of tasks (work packages) can be assigned multiple work breakdown taxonomies.

- **Infer project volatility statistically, not causally.** This assumption follows from the first. If work packages are modeled without dependencies, project managers won't be able to describe slippages in dates based on the logic of dependencies. While this may seem to be a fatal flaw, as capturing dependencies in the project management system is considered primary in predicting project completion, it does not have to be. Another way to think about this is to consider the following question: could an observer totally unfamiliar with the causal logic in the project dependencies infer the project completion date from looking at task-volatility and completion rates alone? In many projects, because of short business opportunity windows and limited operational time frames to accommodate IT change, the project completion date is fixed. Given the plasticity of IT and myriad alternate approaches to the same problem, project teams can often radically alter the critical path to reduce it or remove it. Project managers can then consider critical path manipulations as *additional (or reduced) resources* within the above set of assumptions rather than as a recalculation of how those resources depend on each other. Project volatility then can be used as a key metric to infer what resources need to be added or what parts of the project plan need to be removed or deferred to achieve the promised delivery date.
- **Ideal projects have low volatility and high predictability.** Conventional wisdom says that one should underpromise and overdeliver, and good projects are ones that are ahead of schedule and undercost. In the approach advocated here, good projects have low volatility and are ones that are delivered as close to the estimated date and estimated cost as possible. Expert management control is about getting the results that are intended with minimal variation. Teams can improve their control over projects only when they are motivated to reduce error, not increase it with underpromised project plans. Experienced IT managers and business executives find out who consistently underpromises and factor this deviation into their expectations. In the approach discussed in this *Update*, underpromising can be easily detected and corrected by analyzing project volatility, which will clearly indicate the degree to which underpromising is occurring.

In Part II, I will define project volatility in more detail and show how a project-volatility metric can be put into practice.

ACKNOWLEDGMENT

The author would like to thank John Stewart for his suggestions on volatility trading and project-management volatility.

ENDNOTES

[1]Loconsole, Annabella, and Jurgen Borstler. "Are Size Measures Better than Expert Judgment? An Industrial Case Study on Requirements Volatility." *Proceedings of the 14th Asia-Pacific Software Engineering Conference*, IEEE Computer Society, December 2007, pp. 238-245.

[2]Lam, W., and V. Shankararaman. "Requirements Change: A Dissection of Management Issues." *Proceedings of the 25th EUROMICRO Workshop on Software Process and Product Improvement*, IEEE Computer Society, 1999, pp. 244-251.

[3]Kerzner, Harold. *Project Management: A Systems Approach to Planning, Scheduling, and Controlling*. 9th Edition. Wiley, 2006.

[4]Sauer, Chris, Andrew Gemino, and Blaize Horner Reich. "The Impact of Size and Volatility on IT Project Performance." *Communications of the ACM*, Vol. 50, No. 11, 2007, pp. 79-84.

[5]Kerzner. See 3.

ABOUT THE AUTHOR

Vince Kellen is a Senior Consultant with Cutter's Business-IT Strategies and Business Intelligence practices. Mr. Kellen's 25-year experience involves a rare combination of IT operations management, strategic consulting, and entrepreneurialism. He is currently CIO at the University of Kentucky, one of the top public research institutions and academic medical centers in the US.

Mr. Kellen previously served as VP for Information Services (CIO) at DePaul University, where he won *CIO* magazine's coveted Top 100 award in 2007. He also served as a partner with strategy consulting firms, where he helped *Fortune* 500 and midsized companies with business and IT strategies, IT organizational development, customer experience management, customer relationship management (CRM), and data warehousing and analytics.

A national and international speaker on business and IT strategy issues, Mr. Kellen has authored four books on database technology and more than 120 articles and presentations on IT and business strategy topics. He holds a master's degree from DePaul's College of Computing and Digital Media and is currently completing his PhD in computer science at DePaul. Mr. Kellen was also an adjunct faculty member at DePaul for 10 years, where he helped launch the graduate program in e-commerce — one of the nation's first graduate programs concentrating on e-commerce — and designed and taught graduate courses in enterprise architecture, CRM technologies, and portals. He can be reached at vkellen@cutter.com.

