Inflate Gate: Mastering Overestimation for Agile Software Projects

Note: This article was first published in Computer Aid's Accelerated IT Success; (Featured Article); IT Metrics & Productivity Institute; August, 2015 and is re-created 10/7/2023 when no longer available on the original website.

The 2015 Super Bowl game was clouded by accusations of "cheating" on the part of the New England Patriots. The team was being investigated for *underinflating* the air pressure in footballs used in an earlier game. The "Wells Report" detailed psi measures, temperature, measurement devices (gauges), and statistical analysis. [0] Fan and public reactions ranged from outrage to ambivalence. The outraged were astounded that a team or its players might escape retribution for letting a little "air out of a game ball." Harmless hooey or malicious measurement? The consultant adage "it depends" is once again, appropriate.

Software projects aren't much different when it comes to variation in estimates. For decades, estimators have been accused of hyper-inflating estimates to substantiate a greater margin of error, to gain a little breathing room for the development team. "Algorithms" akin to "double the estimate and add 25 percent" were not uncommon. Nonetheless, surveys and statistics on software development performance have for decades evidenced remarkable variation between estimates and actual. Cost and schedule overruns have been the subject of some of the software industry's worst fiascos. [1, 2]

Enter agile software development. Developed in Japan's innovative manufacturing sector and later in the 1990's adopted in the US for software development [3] scrum seems to be the darling of the software development industry. Some version of scrum is used by almost 75 percent of agile teams. [4] I too have touted the benefits of agile development in "*Keep the Baby*." [5] But not everything in agile is new; iterative and incremental notions have been traced to the 1950s. [6] Disturbingly, estimating has not evolved from the practices of the past. Estimation inflation is alive and well in the world of agile development.

Note: This article was first published in Computer Aid's Accelerated IT Success; (Featured Article); IT Metrics & Productivity Institute; August, 2015 and is re-created 10/7/2023 when no longer available on the original website.

The unsuspecting eyes of the trusting product owner, the novice project management office, and the distant management disconnected by "self-managed" and "self-organizing" teams all fall victim to overestimation. Perhaps the most overlooked victim is the team itself—unable to predict a "sustainable pace" or self-disclose and steady its own vacillations via retrospectives. How do they pull it off sprint after sprint and project after project? Let's look at SIX of the techniques I've observed first hand.

1. Unclearly defined and often deceptive "productivity." This is not the "elusive to measure productivity" of the team; instead, it's the personal productivity for team members. Personal productivity is the time team members are actually working on project work. It may include team meetings. It certainly includes time on tasks as identified on the "task board." It excludes time for non-team meetings, unrelated e-mails, etc. Personal productivity scores often range from 80 to 50 percent. I begin to get nervous when team members dip or claim to dip below 65 percent; at 50 percent I'm not sure that I need that team member. Often the team agrees to use an "average" that they agree to; lower averages should be challenged and in turn, defended. Over a "person year" a 10 percent inflate rate "re-allocates" 230 hours from the work of the team.
2. A close relative of "personal productivity" is "*double counting*." While *double booking* may ensure seats are filled on commercial aircraft *double counting* is another "leak" in project time. A common practice in determining "productive time" is to claim that team members spend 10 or 15 percent of their time "doing" e-mail. But is it non-productive time when those e-mails are related to the project? What if the response is commenting on a feature or implementation approach? If that time is also allocated on the task board, double counting is present. Somewhat more insidiously, assume a team has defined peer and team reviews as part of their non-productive "personal productivity." After all they argue, team members are unable to work "their tasks" and so, they are unproductive in working on their own deliverables. Later the team begins to include "reviews" as part of their "work flow" and review tasks begin to appear on the task board. Without an adjustment to "personal productivity" the time spent in reviews is reflected in both "non-productive time" and as part of their "available work time."
3. Refactoring. "Yes" refactoring is explicitly included in eXtreme Programming (XP) and mentioned in general in other approaches. (refactoring). Refactoring is the ongoing "grooming" of code to keep it clean and lean. But the fact that refactoring is practiced at all could be described as waste—not because it doesn't do something useful, but because it needs to be touched again, revisited, updated, often in the absence of premeditated design and architecture. The notions of iterative and incremental are core to agile just as rework is defined as waste in the lean world. In a sense, the inclusion of time for refactoring is the inclusion of time for waste. The advocacy of minimal planning—certainly anything much beyond the current release—in the agile world discounts the value of architectural and design considerations early in the project. Recent studies have demonstrated the structural quality of some "waterfall" practices adoption early in agile projects. [7]

   But waste is not the inflator; it's the outcome. The inclusion of refactoring as part of many stories (or features development) is the inflator. I've literally witnessed "refactor" as a task during sprint planning after virtually every other task. Unfortunately, I've seen agile coaches and scrum masters leave this unchallenged. Product owners are rarely in a position to recognize refactoring as rework or ask for justification. Historically, rework accounts for about 30 percent of project team's effort. [8]

4. Insignificant tasks. Teams often use tasks as a way to define the work associated with stories and features. Business partners (product owners) are often ill-equipped to recognize spurious tasks. Some of these are related to refactoring but others may be related to reviews that don't occur or are cut short. Others are related

Note: This article was first published in Computer Aid's Accelerated IT Success; (Featured Article); IT Metrics & Productivity Institute; August, 2015 and is re-created 10/7/2023 when no longer available on the original website.

to meetings with stakeholders or status reporting. All of these are legitimate activities, when performed. Not all team leaders and scrum masters may be familiar enough with the team's development work flow to challenge bogus tasks. A great deal of trust is granted to agile teams as part of their self-managed, self-organized structure. Yet the temptation to cushion estimates may be too difficult to resist. Insignificant tasks may contribute to the "*double counting*" in 2 above.

Teams may decompose tasks to a degree that affords the opportunity to overstate the actual work. While a common rule of thumb is to keep tasks to less than eight hours, a guideline to limit tasks to activities that require an hour or two might be worth considering.

5. All tasks are subject to exaggerated task times. Adding a buffer to each of the task times provides an additional measure of discretion for the development team. Padding the estimates almost seems natural. But specious refactoring and insignificant tasks that are also embellished, add insult to injury.

Consider the compounding effect of these sources of estimation error. One could argue that they are offset by the "optimism" that has been observed and documented by teams developing estimates. [9] If each of the above five sources of variation were present and represented a conservative 10 percent impact, teams could easily be adding 50 percent to their schedule times. Even in time-boxed agile, a team would require half-again as many sprints or iterations to complete all the work. Conceivably, the impact could be much higher than 10 percent and it would be less plausible for all of these sources of variation to be present simultaneously and go unnoticed for an extended period of time—at least we would hope!

Not all teams are going to resort to these tactics or go unchallenged; but many teams will avail themselves to some of these sources of variations some of the time. Observing the co-existence of each of these practices is what inspired this article.

Unfortunately, the inability to compare agile team productivity is an ongoing criticism. Teams "self-measure" their delivery using velocity which is often based on story points. Story points represent the "degree of difficulty" associated with a story. As stories are "accepted" by the customer (product owner in scrum) the team is "credited" with the story points associated with that story. The sum of the stories points completed for a sprint constitute the team's velocity.

Teams are generally expected to sustain their velocity once it stabilizes after a few iterations. Better teaming, efficiency of process, and improvements from retrospectives can all accelerate velocity. Of course there's one other way to improve velocity without delivering any additional "potentially shippable product."

6. Inflate the story points. As the team becomes experienced with the work being performed they should become better estimators. That's good. If the team is pressured to increase its velocity (and implied business value) they may "adjust" the story points to ensure sustained velocity or even an increase in velocity. That's bad. Velocity increases but delivery remains the same. Since story points are not compared across agile projects, who can argue otherwise? Teams can compare analogous stories to each other as a "self-check" but don't expect that to happen without pressure from outside the team. Self-managed teams may reject those comparisons as well. An "independent" source would likely upset the trust relationship between an agile team and its business partner; but this may be prudent in the long term. Ronald Reagan's "trust but verify" seems appropriate here as well. Under all circumstances, caution is urged.

Note: This article was first published in Computer Aid's Accelerated IT Success; (Featured Article); IT Metrics & Productivity Institute; August, 2015 and is re-created 10/7/2023 when no longer available on the original website.

While it may be intuitive that agile projects provide higher quality products, a faster pace, and timely completions, all of those assertions have been refuted by some agile practitioners and observers. Some of those benefits can likely be ascribed to most agile endeavors. Nonetheless, assertion without objective data can be troubling.

So how do we get teams to reduce or eliminate the estimation inflation? Knowledgeable scrum masters that challenge potentially invalid team estimating practices and experienced product owners are poor substitutes for team integrity. Given the choice between remaining "in the dark" or being aware of inflation-focused estimation practices, which would you prefer? I've tried to raise the awareness and offer a couple of antidotes. As agile approaches mature, reliable and consistent estimates and measures must advance. Otherwise, objective evidence of the impact of agile techniques will rightfully reside under a cloud of suspicion. The quantitative data associated with Deflate Gate removes some of the subjectivity from that discussion. Perhaps we will someday be able to echo the same sentiments regarding agile productivity.

[0] Investigative Report Concerning Footballs Used During the AFC Championship Game on January 18, 2015; Paul, Weiss, Rifkind, Wharton, & Garrison, LLP; Theodore V. Wells, Jr., et al; May 6, 2015

[1] News from the National Academies; July 20, 2006; nationalacademies.org

[2] Dynamic Markets Limited; August 2007

[3] *The New New Product Development Game*; Hirotaka Takeuchi and Ikujiro Nonaka; Harvard Business Review; 1986.

[4] 8th Annual State of Agile; VERSIONONE; 2014

[5] *Keep the Baby* (Examining Agile); Schofield; MetricViews; Winter, 2014 / 2015

[6] Gerald M. Weinberg, as quoted in Larman, Craig; Basili, Victor R. (June 2003). "Iterative and Incremental Development: A Brief History". *Computer* **36** (6): 47–56.

[7] The CRASH Report 2014-2015; pg.3; Dr. Bill Curtis, et al

[8] *The Requirements Payoff*; Karl Wiegers; informationweek; July 12, 2010; pg 39

[9] Underestimation in the "When It Gets Worse Before it Gets Better" Phenomenon in Process Improvement; Advanced Concurrent Engineering, 2011, Part 1, 3-10; Ricardo Valerdi and Braulio Fernandes

About the author . . .

Joe Schofield a sought international speaker, an active agile transition coach, and is widely published in an array of media. He is President Emeritus of the International Function Point Users Group. Joe retired from Sandia National Laboratories as a Distinguished Member of the Technical Staff after a 31-year career. During twelve of those years he served as the SEPG Chair for an organization of about 400 personnel which was awarded a SW-CMM® Level 3 in 2005. He continued as the migration lead to CMMI® Level 4 until his departure. Joe has facilitated over 100 teams in the areas of software specification, team building and organizational planning as a lean six sigma black belt while also employing business process reengineering. He is a Certified Agile Expert and Scrum Master, an active CMMI Institute-certified Instructor for the Introduction to the

Note: This article was first published in Computer Aid's Accelerated IT Success; (Featured Article); IT Metrics & Productivity Institute; August, 2015 and is re-created 10/7/2023 when no longer available on the original website.

CMMI®, a Certified Software Quality Analyst, a Certified Function Point Specialist, and a Certified Software Measurement Specialist. Joe is a frequent presenter in software measurement forums, including the Software Best Practices Webinar Series. Joe taught over 100 college courses since 1990, almost all of these at the graduate level. He was a licensed girl's mid-school basketball coach for 21 seasons--the last five undefeated, over a span of 50 games. He has over 80 published books, papers, conference presentations and keynotes—including contributions to the books *The IFPUG Guide to IT and Software Measurement (2012), IT Measurement, Certified Function Point Specialist Exam Guide, and The Economics of Software Quality.* Joe completed his Master's degree in MIS at the University of Arizona in 1980. By "others" he is known as a husband, father, and grandfather.

Note: This article was first published in Computer Aid's Accelerated IT Success; (Featured Article); IT Metrics & Productivity Institute; August, 2015 and is re-created 10/7/2023 when no longer available on the original website.