Our Estimates are Terrible!

HANS SAMIOS, Intergraph Corporation

The practice of planning and estimating has a long history. Traditionally a plan that does not complete is often seen as a call for more planning and improved estimation. While we were planning the "agile way" we still fell into the trap of over-commitment when the new idea (points) did not line up with the old approach (hours). We'll work through how this happened, the discovery process, and the process of distributing the learning to the whole organization.

1. INTRODUCTION

Any organization that has been doing software development for a long time has a lot of change to get through to get to a true Agile implementation. There is a lot of resistance to change at all levels of the organization. Some of the resistance is caused by the replacement of practices and thinking. Even when this is overcome, there is a second, more insidious tier of thinking that pushes people back to their traditional thinking instead of going forward with the new approach.

The practices associated with planning and estimation is an example where the second level operates. After working with release plans based on velocity and story points, teams felt they were not producing accurate enough information. I was constantly asked "what can we do to improve our estimates?"

This is one aspect of the insidious creeping of traditional thinking. Sure there was good conversation that estimates were not meant to be highly accurate, that we'd change our plan based on new information and so on. But there was still this feeling that the estimates we were generating were not good enough and that we need to work on this to improve our planning. It is easy to fall into the trap that if we did not meet our plan, then there was something wrong with the estimates in the first place. This lead to the next step, that we must therefore spend more time on estimating which very quickly moved us back to doing more and more planning work upfront.

This phenomenon is well recognized. What surprised us when we ran a retrospective on estimating is that there is a Sprint-by-Sprint confirmation of this thinking happening at the team planning level which reinforces the traditional thinking. Every sprint the velocity was used to help guide how much the team should take on during Sprint Planning 1. During Sprint Planning 2, when detailed work was broken down into hours there was never enough hours generated for the team to feel like they would be always busy and so the team went back to get more work, thus over-committing.

2. CONTEXT

2.1 Who We Are

Intergraph is a product development shop with a 30 year tradition. We have two major areas of expertise:

- PP&M: supply engineering applications to process plant (factory plants that produce chemical, electrical and petroleum products at large scale) and marine (large ships such as crude oil carriers) organizations. PP&M started the transition to Agile (Scrum) in 2008 to address business needs of improved visibility into development projects, improved quality, and better engagement of our people. Today there are over Agile 100 teams operating.
- SG&I: supply map-based applications to utility, dispatch and mapping organizations. SG&I started its transition to Scrum in 2011 to address similar needs. Today there are over 60 teams operating in 8 countries around the world. All product development is done using Scrum.

2.2 From Traditional to Scrum

This story comes from our SG&I division. Before moving to Scrum, we had a very traditional approach to the management of software release projects:

- We used tools such as Microsoft Project to help us manage the projects
- Estimates were done by a small group of experts
- Deviations from the plan were rigorously controlled
- People (resources) were managed as "full time equivalents (FTEs)" and allocated to work in smaller and smaller slices based on changing perception of where effort should be applied.
- Planning became an increasing percentage of the release project as we increasingly tried to plan more in order to make the project more successful.
- Certification was handled at the end of the release cycle, again as a large percentage of the release plan.

The realization that our customers and field organizations were increasingly dissatisfied with our work led us to Scrum. The transition (first phase) took about 2 years:

- We moved everyone onto Scrum Teams, creating teams that included skills in development, QA and documentation, and set up dedicated Product Owners and Scrum Masters.
- We trained everyone in the new way of thinking and working, with 3 days of training for new Scrum Teams and supporting training for Product Owners, Scrum Masters and Management
- After initial resistance everyone understood, adopted and accepted ideas such as:
    - Servant leadership from management and team leaders
    - Product Owner determines order of work
    - Team decides how much they can take on
    - Reporting is based on points and velocities
    - Feature-based plans using user stories
    - Ceremonies – daily scrum, sprint planning 1 & 2, sprint review and sprint retrospective
    - Artifacts – product backlog, sprint backlog, definition of done
    - Etc

Today we have 60 teams and our improvement efforts are related to issues associated with scaling of the practices and increasing use of Agile (XP) engineering practices beyond continuous integration and unit testing.

In particular, we have completely transitioned to a point / velocity based approach to understand where we are up to relative to our release plan (Samios).

3. THE SITUATION

This all sounds great, right? And it is. But there are always problems and we know there are things we need to do better.

What surprised me was the level of traditional thinking that we still had in our organization, even after 3 years into our Scrum implementation. To understand how this manifested itself, let's examine how we do our planning for a Sprint.

3.1 Sprint Planning

Sprint Planning is done from the Product Backlog. The Product Owner provides the stories. The Team provides the estimates either through planning poker or triangulation approaches.

During Sprint Planning 1, the Product Owner and the Team get together and determine what might be built in the next Sprint. The Team selects how much they think they can take on based on the previous velocities (often the average) that they have.

During Sprint Planning 2, the Team gets together to determine how they will deliver the stories. In our case it is standard for teams to break down stories into (sub-)tasks and estimate each of the tasks in terms of how long they think it will take (in hours). A running total of the hours are maintained, and this number is compared to the (pre-calculated) total of hours the team has available in the Sprint. Teams are taught that if they find they don't have enough hours to complete the work identified in Sprint Planning 1, then they should tell the Product Owner, and tell her that they cannot "commit" to the lowest priority items in this Sprint. Similarly, if there are hours left over they should work with the Product Owner to take on more work.

What is the problem with this approach? With the benefit of hindsight it seems pretty obvious although it definitely was not clear at the time. Here's what happened when the teams applied the Sprint Planning approach described above.

3.2 Scrum Practices Meet Traditional Thinking

The team would start with Sprint Planning 1 where stories were selected to meet expected velocity. During Sprint Planning 2, tasks were generated and estimated to fill the hourly capacity of the team. And this is where the problem hit. Frequently the number of "task hours" generated during Sprint 2 planning was below the capacity of the team. Although some of the stories selected during Sprint 1 planning were risky or complex, the team did not know how these risks and complexities would manifest themselves. Therefore, when doing the detailed task-level planning they could only estimate what was more or less known. This resulted in fewer hours being estimated for the story than the points might indicate. This in turn resulted in the team thinking that they have excess capacity and so they taking on more stories, eventually filling up the hourly capacity bucket, but also leading to an over-commitment on the Sprint.

3.3 How Do We Improve Our Estimates?

As might be expected many teams were not meeting their commitments, including the SG&I team featured in this story. Discussions invariably turned to the topic of how to improve the estimating process. Again, this is a left over from the traditional planning where, if there was a problem with the plan, the general approach was to spend more time planning and estimating in the hope that the plan would be better the next time.

This is not the first time this discussion came up, but only now do I recognize it as a "smell". In the past we provided lots of ideas on how to improve estimation practices, from explaining about cognitive biases, to providing assistance on the ways to run planning poker sessions, and to providing specific recommendations for practices Teams could try (keystone story, work classification, comparing estimate to reality, etc).

A number of teams talked about mapping points to hours. These discussions were usually caused by a desire to improve the estimation process, since there were problems with the plan. Many teams started trying to figure out how to map hours into points, started discussions about "on average 5 hours means 1 point". Sadly this approach did not improve the quality of the plan.

Since points factor in "risk" and "complexity" it is not a surprise that there is no clean mapping. But the thinking process continues despite that. I suspect some of this discussion is also caused by the degree of comfort associated with hours – they are concrete, are the way we've done things for years, etc. And it is hard to understand on this basis why there is no clear mapping.

4. WORKING THROUGH THE PROBLEM

At the end of 2013 I received an email from one of our Scrum Masters with the subject "Our Estimates are Terrible". The Team had requested outside help to help them improve their estimates.

We decided to set up a focused Retrospective to walk through the issue of estimation. As part of my retrospective planning effort, I examined and analyzed team data on estimation accuracy.

4.1 Data

Every team at SG&I collects metrics to help understand how they are doing and whether they are improving. At a minimum, teams collect "Committed" and "Actual" Velocity Sprint-by-Sprint. More recently all teams have been collecting additional metrics (Sutherland).

As each story was done in the Sprint, the team would ask themselves "Now that we have completed this work, and factoring in risk and complexity compared to our keystone story, what do we think the estimate should have been for this story?" The data was collected as "actual story points" and we were able to compare it to the original estimated "story points". As a result we had over a year's worth of data with 200 separate points comparing the original estimate to the actual. (Note: for the purposes of reporting of velocity etc. the original estimates are used, not the actuals).

Since this information was available we decided to use it to see how bad this team's estimates really were. Based on work on estimation (Helms) I started by preparing a simple frequency chart of the estimates and the resultant actuals. The team's data looked like:

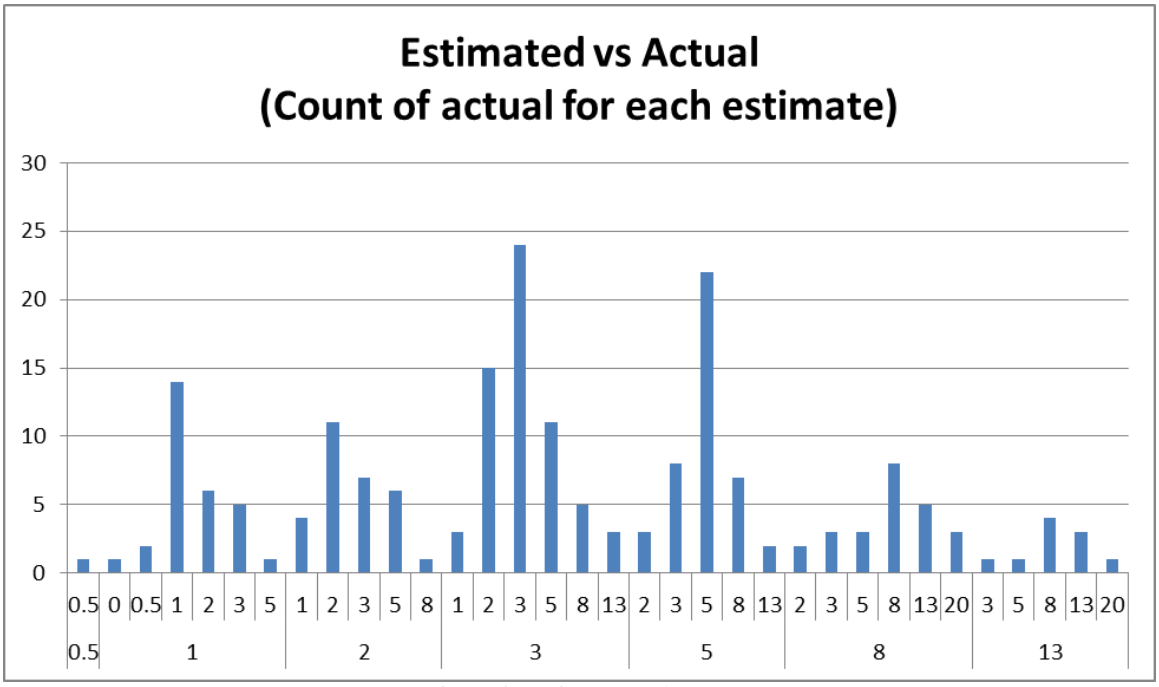

Figure 1: Estimates vs Actuals

The chart shows for each of estimates the team made (bottom set of numbers – 0.5, 1, 2, etc) the number of times the actual result matched the estimate. So for the estimate of "3" there were 24 times that the actual was also 3 (tallest spike on chart), 15 times it was a 2 (to the left of the tallest spike), 11 times it was an 8 (to the right of the tallest spike) and so on.

Looking at the chart you can see that most of the time when the team estimated a story as being a certain number of points, the actual number of points matched the estimate. Further you can see that when the actual did not match the estimate exactly, the deviation was typically not that great. This is good news. We cannot expect that when we estimate something we get it 100% right every time. Sometimes the actual work with be higher, sometimes lower. An estimate is not a commitment, after all. But you can see the data shows that the estimates are pretty good, close, overall.

For me this was a surprise. I'd never really questioned the premise that the team's failure to meet its commitments was due to "poor estimation". And it led me to think about the retrospective. If the problem is not poor estimation, then what is the real problem?

4.2 The Retrospective

We started with a discussion on the reason I was invited to facilitate – the email stating that their estimates were terrible. The Team worked through a brainstorming session on what they thought their problem was. Example responses include:

- I am not sure how to point
- We have different ideas about what a 5 is versus what an 8 is
- We are feeling pressure to meet plans
- Our leadership is frustrated by stories that were not complete during sprint
- We are taking a long time to do estimation

Without making it sound too dramatic, I then showed the result of the analysis of their data. After a quick explanation of what they were looking at the team quickly came to the conclusion that their estimates weren't too bad. This was a surprise to the team as well. Sure they could look to make the variance between estimated and actual point values a little narrower but in fact there was not even consensus that was necessary, let alone possible.

We then went back and discussed some of the literature on estimation and in particular the impact of time on estimation accuracy (Cohn):

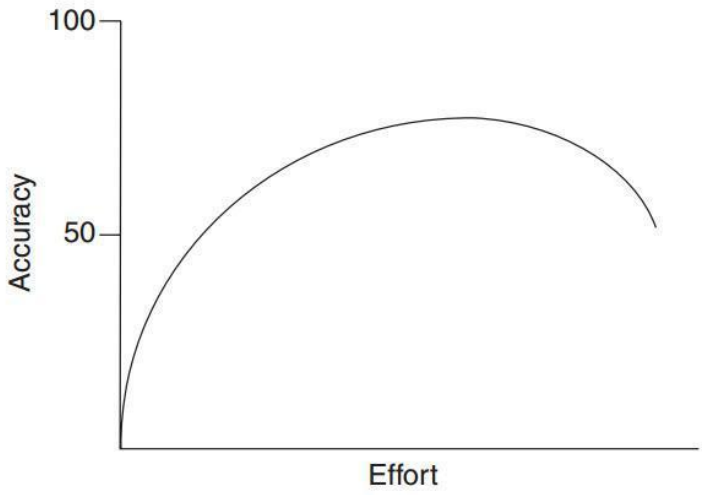

Figure 6.1 Additional estimation effort yields very little value beyond a certain point.

Since the issue was no longer seen as "estimation accuracy" the Team went on to identify what the real issues were. Identified issues included:

- Pressure from team leads to improve estimation
- Pressure from management to meet plans
- Overly optimistic plans with no buffer to address problems

The reason they were over-committing was that they did not trust their point estimates. When they couldn't fully rationalize the point estimates to hours upfront, they second guessed themselves.

With this as a background the teams decided their next steps were:

- No more second guessing team estimates
- Increase focus on previous velocity to determine what can be done in Sprint so that Story Points really became the measure of capacity
- Increasing focus on use of buffers etc to avoid over-commit for release (Goldratt). This meant allocating a couple of Sprints in the release that were empty, where we didn't know what we were going to do. This was done at the top level release plan, not at the team level. In other words we told teams to work as normal (with the improved understanding of their true capacity). The Product Owner maintained the buffer at the release level and would then aggressively manage the influx of expanding work (whether by discovery by the team or as a result of improved understanding of the requirements) and ensure this buffer was not used too early in the release plan.

5. WHAT DID THIS LEAD US TO?

5.1 Feedback from the Team

When asked, the Team said they got the following from the retrospective:

- Estimation is not just about how long something is. We need to make sure complexity (as represented in our story points scale) is part of the estimation process.
- We need to understand that we will never be 100% accurate in estimates
- That estimation really is a guessing game
- That the team had a problem over-committing
- That the team needed to pay attention more to the points being taken on then the hours
- That it is expected that there will be discrepancies between points and hours and be OK with it.
- We can trust our estimates

The Team said the retrospective was "a positive shot in the arm."

We also asked the team what other things they think should be stressed in discussing this retrospective with others:

- People are unduly hard on themselves. The team really did think there were terrible at estimating, and this turned out not to be the case.
- Don't assume your instincts are wrong. The team really did feel like they were taking on too much work each time, but were unable to change the behavior.
- Just because you cannot quantify risk in hours doesn't mean there is no risk.
- Estimates help the team understand capacity. It is not just a management tool.
- There is value in capturing data. But only if you use it.

5.2 Why Wasn't the Team Able to Solve the Problem?

The Team was asked why they were not able to solve the problem themselves:

- Inexperience with "agile" (actually points and expectations)
- Feeling that points were "just a management tool"
- Lack of understanding of how to use the data that they were collecting

I think there was something more basic than this. The team did not take the time to step back from the problem and really determine what was happening. The references cited here had been talked about in training sessions, had been documented on internal blogs and so on, and so there was awareness about the

general ideas. But time is still required to formulate the discussion into something that will help the team and this time was not taken.

5.3 Getting the Message Out

It is pretty clear that at SG&I we needed to conduct this discussion with just about every team. The "smells" were pretty prevalent. In particular (and something that gave us a lot of concern) we had survey information that showed chronic over-commitment across all teams (by about 141% on average).

Because the survey was recent, we had a reason to publicize the results of the retrospective. We have a regular distribution at SG&I called "Scrum Distributions" that goes to all Product Owners, Scrum Masters, Management and anyone who has registered an interest. The contents of the distributions include both "formal" information (such as changes in guidelines) and more general information (such as learning, articles that people have found useful, and so on).

I spoke with the Team about using their retrospective and the data as a case study, and the Team happily agreed. We then crafted a Scrum Distribution called "Our Estimates are Terrible!" This was blog post that provided background on what was done, provided the spreadsheet we used for the analysis, and what we learned as a result.

Of course, it is not enough to simply say this kind of message once, and hope that you get traction. The materials were also incorporated in an "advanced materials" training session we provided for all Product Owners, Scrum Masters and management. We also included a discussion in this about the concept of "relative error" so people can do additional analysis on their "errors" (Helms).

Additional materials were added to our corporate Wiki tool so that in the future people will be able to find the information and what was learned. This makes it easy to begin the discussion with the next person that says "our estimates are terrible."

And finally the result has been socialized through numerous one-on-one conversations with all levels of the organization. In general we have found that management has accepted the idea, although they still may not like what it means when some of the plans do not go as expected.

6. CONCLUSION

I think the biggest surprise that came out of this discussion was the sway the traditional hour-based thinking had over everyone and how long it took for us to recognize the problem we were facing. In some ways the fact that we did use points and velocity for so much (e.g. - progress reporting on the release, planning) obscured the fact that hours and the associated estimating process was alive and well in many parts of our organization.

I will say there is no magic here. We are not suddenly "better". In reality we have an increasing awareness of the issue, and we have seen some improvement in our over-commitment problem (last survey indicated that average over-commit had gone down by about 7%.) This means that we still have a long way to go and, I suspect, we have not identified all the issues yet.

7. ACKNOWLEDGEMENTS

A special thank-you goes to Jen Jackson and the Bootleggers team not only for the experience of the retrospective but also for allowing me to use the data to help the rest of the organization.

I am always appreciative of Intergraph and its management team. They have been very supportive of the both our on-going efforts at improvement and allowing me to publish information that at first blush may not present the company in its best light. After all, we are talking about problems we had!

I'd also like to thank Nanette Brown who mentored me through the process and made sure I actually got on with the commitment I made. There is no doubt that to and fro with her helped me clarify the message and everyone benefited as a result.

REFERENCES

Hans Samios "Overcoming Traditional Project Release Reporting with an Agile Approach Focused on Change", Agile 2012 Conference
Jeff Sutherland / Scott Downey "Scrum Metrics for Hyper-Productive Teams: How They Fly like Fighter Aircraft", Agile 2010 Conference
John Helm "Lessons from Space Shuttle Disasters for Avoiding IT Project Disasters", Agile 2013 Conference
Mike Cohn, *Agile Estimation and Planning*, Chapter 6
Eli Goldratt, *Critical Chain*