Overcoming Traditional Project Release Reporting with an Agile Approach Focused on Change

Hans-Peter Samios
Security, Government and Infrastructure
Intergraph Corporation
Madison, AL USA
hans.samios@intergraph.com

Abstract— **If you are an Executive in a large organization Scrum / agile helps address a number of issues that you face in releasing products – improving quality, predictability, engagement of your people, and productivity. You've got help in the form of an Agile Coach and you understand that change is required to get these benefits, not only for the Scrum teams directly affected but also in the way you think about and approach development projects. The transition is making progress, and you are seeing improvements at all levels but you cannot help feeling that you've lost an understanding of the key decisions that the organization needs to make. You still have to run the business, to make commitments, and to release products. In the past you had information from the product teams that allowed you to understand where the issues were. That information seems to have disappeared. When you ask for it, you are given a "release burn-up chart" and are told that "Scrum means we respond to change" and "we don't do big upfront planning any more." How can you get what you need and still be "Scrum", still be "agile"?**

Keywords-reporting; Scrum; agile; executive management; culture; adoption;

I. Introduction

Intergraph is a large software development company focused on developing complex products aimed at professionals. It is made up of two distinct organizations. Intergraph Process, Power & Marine (PP&M) is the leading global provider of engineering software for the design, construction, and operation of plants, ships, and offshore facilities. Intergraph Security, Government & Infrastructure (SG&I) provides geospatially powered solutions to the public safety and security, defense and intelligence, government, transportation, photogrammetry, utilities and communications industries. Intergraph Corporation is a wholly owned subsidiary of Hexagon AB. Each division has a large software development shop comprising more than 500 technical (development, QA, documentation, planning) people in more than 10 countries around the world.

Intergraph PP&M moved their development operation to Scrum starting in December 2007. This effort culminated in the formation of 75 Scrum Teams and our ability to say that everyone who should be on a Scrum team is on a Scrum team. I was responsible for the rollout of Scrum in the PP&M division. In 2011 I moved to the Intergraph SG&I division to take them through the process of change. At the time of writing we had about 45 Scrum Teams and were about ¾ of the way through the first phase of the transition. While the divisions are part of the same organization, the reality is that they have totally different customer bases, culture, and operating methods. As a result this second transformation is totally different. This paper addresses one of the issues associated with the Intergraph SG&I transformation.

For years Intergraph SG&I had been using a traditional stage-gate approach to the reporting of progress on releases based on the presumption of a serial (waterfall) process. Project management had been automated to a significant degree and was well entrenched within the organization. The standard discussion on release reporting often fails to address the legitimate concerns of executive management for this large a development group as it focuses on team rather than the organization needs. If not addressed this creates resistance to the adoption of Scrum / Agile which, given all the other cultural issues that need to be addressed could seriously distract and even jeopardize the implementation. What is required is an approach which maintains the integrity of the Scrum team approach while still providing data to the business that allows them to make decisions.

II. Legitimate Need

Executive management asks the team for "the plan". Or they tell the team that they need a commitment for something. Or they want to know how much a feature is going to cost. We are disappointed because this feels like management wants to have all the benefits of Scrum / agile but without having to change any of their approaches. They are for things that to us look like the "old" way of doing work. Management clearly doesn't "get it." As a result we are going to have to go back to the old way of doings things and we'll be agile in name only.

Sound familiar?

I've been here. But I have to say I didn't understand the issue very well when I went through this transition. The first time I went through the agile transition with an organization we (executive management, the teams, etc) worked through issues together. As we moved to the new approach we made a lot of mistakes. Reports that we used to work with (Gantt charts, resource charts, and defect arrival rates from the QA process) simply didn't make sense any more. The Scrum replacement, the "release burn-down" chart, while powerful simply did not give us the information we needed. We eventually evolved to a "monthly health review" with a list of 5 or so charts that gave us the information we needed about the status of release projects and allowed us to make the business decisions required.

Now fast forward to my next assignment. The SG&I organization had spent a lot more time and effort building the release reporting process for the organization. They had more formal processes, more formal reporting and a lot more meetings – weekly schedule meetings as well as bi-weekly approvals meetings.

We started down the transformation process. When people asked about reporting on release status, I thought I had the answer based on the work that we'd done in the other division. Eventually we got to the stage where the executives needed to see something and so I started to talk about the kinds of things we could do. However I kept on hearing that the executives wanted something else and have to say, based on the language used, that this sounded suspiciously like they wanted the things they were used to getting in the past. When teams heard demands for information, they started to produce the data required, whether it made sense or not. The teams also complained that "they (management) clearly did not get it" and I have to say I started to think this as well.

It was becoming a problem.

What we were all missing was something pretty fundamental. The business as represented by the executives had a legitimate need for more information than what we were providing. They were asking for it in terms they used for this information before the transition, based on years and years of doing this work in a particular way. To the Scrum teams this came across as "we want the old traditional, command and control way of working." What management was really saying was "we need more information to help us run the business."

There is a legitimate need. However, we were two groups separated by a common language and we both did not get it.

III. Understanding Needs

How did we figure this out? We needed to start with the basics – we needed to change the assumptions we were forming based on the language being used. Most people come to work every day wanting to do the right thing, what is best for our customers, our people and our business. If an issue is constantly coming back then it means that we are probably misunderstanding something important. We decided to spend some time with the executives to understand what they wanted and what we thought we could provide.

The first step was to introduce an approach to reporting based on the Scrum / agile world. We pulled together a meeting of the executives with the following agenda:

Purpose:

- To understand release reporting based on Scrum / agile principles.
- To determine additional needs.

Agenda:

- Base Concepts:
    - Base release status reporting on "done" software.
    - Focus on delivery of value rather than utilization of people.
    - Use story points rather than hours as this focuses on delivery of value.
    - Reporting is a natural by-product of the work done by teams.
- Key Release Questions:
    - Are we going to meet the date?
    - What scope are we going to have in relationship to the initial plan?
    - What changes are happening?
    - Where are the issues we need to address?
    - Customer's view of quality?
- Requirements:
    - What other questions do we need to answer?

The first part of the agenda was used to explain the base principles behind the reporting. The key part of the meeting was the last part. This is where we started to understand the overall requirements. Starting with the basic "release burn-up" chart sourced from team data (story point estimates and velocity), we developed and sold internally the notion that five simple reports would give us everything we need to run the business. These requirements were formed as User Stories to ensure we were all on the same page.

1. "I, as a Scrum Product Owner who is trying to optimize the delivery of a product release in terms of date, scope, and cost need a way to show what the current scope is in the release and the progress we are making toward it so that we can make well-informed trade-off decisions and commitments based on the reality of our company's capabilities." This is the basic release burn-up chart showing total scope and progress against the scope, with trend-lines.

2. "I, as a Scrum Product Owner who wants to communicate with the stakeholders need a way to show how the work for a release has changed from the original baseline to the current status so that everyone is fully informed about the current status of the plan." We called this the "Scope change report."

3. "I, as a Scrum Product Owner who wants to understand the status of the release need a way to show how work is progressing against the major epics of the release so that I can make adjustments to the plan based on completion of these epics." We called this the "Epic progress report."

4. "I, as a Scrum Product Owner who wants to make good investment decisions need a way to show how work is split in terms of the basic investment categories against so that I can make plans based on history and make adjustments during execution of a release when investment category assumptions do not work out." We called this the "Investment Allocation report."

5. "I, as a Scrum Product Owner who wants to make good investment decisions need a way to show how work is split in terms of the next release and existing fielded releases so that I can make predictions for future plans based on history and make adjustments during execution of a release

when these assumptions do not work out." We called this the "Project Allocation report."

The first three of these reports I suspect would be expected by all people interested in release progress reporting. Expressing them as user stories helped to make sure that everyone knew the purpose of the report in addition to simply having the report.

The last two reports came out of requirements the executive team had to their Board. It turned out that the way the Board thought about investment decisions was based on the kind of money they would spend – investment categories. They had specific categories that they tracked to – allocating money spent on new features (discretionary), maintenance, as a result of contractual obligations and as a result of changes in platform that we needed to support.

The project allocation report allowed the executives to understand where we are putting the maintenance spends. Once our software is in place there is a reluctance to upgrade (you don't just upgrade a system being used for 911 calls). The result is that we have to support a lot of fielded versions of the product. This report makes the spend on fielded version more visible.

The high level planning was done in terms of investment and project allocation but most of the development shop was not aware, and did not track, this information. This meant that the high level planning conversation was not the same as the lower level conversation. Making these reports possible meant that everyone was looking at and making decisions using the same data. In addition it went a long way to making the overall approach creditable.

Samples of these charts are shown at the end of this paper. The only adjustments required to produce the reports were some simple attribution on the user stories. For example we added an attribute called "Investment Allocation" with possible values of "Discretionary", "Maintenance" and so on to track the information required for the Investment Allocation report. Product Owners were glad to fill in the additional attribution since it gave them the information that they could use when discussing the release with their business stakeholders in the language of the stakeholder.

IV. Principles of Reporting

While we are on the subject of reporting, there are a number of principles that we used to help us drive the overall thinking behind the approach we took.

A. Interview "C" Levels in the Organization to Understand Their Criteria

OK, obvious, but worth repeating. Make sure that you interview "C" levels in your organization to understand their areas of interest in the reporting so that, again, everyone is looking at the same data and making their decisions by the same criteria.

B. Report the Changes

Scrum and agile is all about embracing the fact that plans change. What we've found is that people do not deal well with release reporting if everything is assumed to be new in each and every Sprint. For release reporting we establish a release plan and then report on changes to that release plan. People find it easier to deal with "what has changed" rather than "the current version of the plan", especially at the beginning of the transition to Scrum.

C. Make Everything Available To Everyone

I really mean this - make everything available to everyone in the organization. While not strictly an issue about the acceptance of the release reporting approach, we have found this idea helps with acceptance in general. Traditionally our product groups were reluctant to open things up for all. The feeling was that people outside the development shop would use this kind of information against them. By changing the approach so that anyone could get to the information, we helped reduce the distrust external people had of the development shop and made a cleaner conversation for all stakeholders. When executives see the same reports being used by the teams as what they see, it also helps build confidence and offers the opportunity for increased trust.

D. Make Sure the Backlog Includes Everything

We make sure that our release plan includes user stories for everything that is going to be needed for the release. Like most agile implementations, we use User Stories to track requirements. These are supposed to be "requests of the team to produce value." In other words features. However if we only put these items in our backlog, we wouldn't have a real understanding of the release status as some of our work is handled, for example, in release sprints (workflow and regression tests that we have not automated as yet). Yes, release sprints are bad and reflect things that we have not really done. But to really understand the status of the software we need to track this work just like new feature work. We therefore create user stories that represent this work, estimate it like everything else (points) and track the work in the release plan. This has the additional benefit of making this kind of work visible and encourages us to reduce this kind of work incrementally (get closer to "done" release after to release).

I realize that this may not be pure "agile" but it has helped us understand the real status of the release and provides increased visibility for issues which would otherwise remain hidden.

E. Make It Simple

Make sure that reporting can be done simply, with no additional work required by the teams, and using simple approaches to categorize data.

F. Use the Same Reports for All Levels of the Organization

Ensure that reports are the same at all levels (team, product release, portfolio), based on the same information (team based estimates and velocity) and producing the same results so that everyone is looking at their view of the same data. Ensure there is a continuum of reporting not only helps with business but more importantly helps the teams so that the data required by the business are a natural by-product of normal team work.

V. What Mistakes Did We Make?

The biggest mistake I made was underestimating the amount of inertia associated with the traditional reporting model. What I should have done was simply tackle this issue much earlier. If I had done this I suspect that there would have been less middle managerial resistance to the adoption of agile. A second mistake I made was not supplying better templates to produce the information required. This led to confusion when teams wanted to do release reports since they virtually had to learn how to do it themselves.

VI. What Challenges Remain?

The main challenge that remains is moving this work to a tool so that we can automate the process. We have sold the concepts, are producing the reports manually but it is taking time when it should be easy to automate. This is in progress.

VII. What Makes This Such a Hard Issue?

One of the things that surprised me as we went through the process of change was how hard it was to get the basic idea that these simple, powerful charts were more than sufficient to do what we needed and that the information provided will actually allow better decision making. With the benefit of hindsight, I think there are two things that make this such a difficult issue to deal with.

Firstly, Scrum / agile reporting are based on delivery of value. Traditional reporting is based on effort expended. For people that have been involved in Scrum and agile for a while this is obvious and we understand that reporting based on value makes a lot of sense. We feel that any business must want to have this view as well. However all the traditional reporting mechanisms focus on effort, on cost. There are a lot of reasons for this but there is a whole culture that has been developed around this type of measurement. Even worse, the financial parts of the organization also focus in this area which means that there is a perception that to manage the business you need to manage the cost.

There is no doubt that management of cost is important. But it is only one part of the equation. To understand whether you are going get a return on the work (your return on investment) you have to also understand that the production of value is critical. However value has not traditionally been tracked. When executives come to the realization that measurement of value is part of what Scrum and agile can provide they certainly want this. Since we do not present this in a clear way (alright, I did not present this in a clear way) with a solid discussion about the consequences, executives can only have traditional thinking approaches in mind. It is up to us to help them understand the additional possibilities of the Scrum / agile approach.

Another reason this issue is so hard is that there is a fundamental difference in the reporting of a Scrum / agile project and a project based on a more serial workflow. The difference is that at the end of each Sprint, the software is "done". We are developing features to completion. The result is that the reporting is a lot simpler and also more accurately reflects the true status of the release project. It also generates data earlier in the release cycle so that people can make business decisions sooner. Even if you don't get completely "done" (if you have a couple of release or hardening Sprints in the release plan) the information is much more accurate and useful and, once understood, really can be used to run the business.

When Scrum and agile is introduced to the organization people don't immediately understand the huge impact that "done" software has on the release plan. Sure they understand that they are seeing done software when they see the demonstration in the Sprint Review. Another leap in understanding is required to really grok what it means to the overall schedule. Firstly scope really does become a variable that you can work with. If we do the most important things first, then by definition the scope at the back end of the plan is less important. Pareto (80% of value delivered with 20% of the work) style thinking indicates that we should be able to deliver early in a lot of cases and still have supplied most of the value.

Secondly, we can report on progress by these completed features. Since they are done, we don't have any risk associated with the features that have been completed any more (with the possible exception is regression issues in the future). Progress against our "top 5 marketing list" represents real, completed progress. This means that business people really can have input into the release plan after the first couple of sprints as the data that we have on the status of what is complete and what remains to be done is solid.

This base level of understanding is required by all in order to be comfortable with Scrum / agile style release reporting. There is a business need to address planning and when this basis is not understood, when there is not a common way to talk about the change, you end up creating an environment where there is no progress.

VIII. Concluding Remarks

The most important thing that I learned from this process is awareness of the trap I suspect we all fall into on a regular basis - don't let your assumptions about others lead you to thinking that you understand everything. As an agile coach you can easily fall into the trap of hubris.

Having got past the initial misunderstanding the application of an overall philosophy of release reporting helped the business get better by establishing a common approach for all levels of the organization. In addition, by focusing on the issue we were able to remove an impediment to the rollout of Scrum and agile while keeping true to the overall agile approach.

There is no doubt that this is not the final solution to the problem. A full Scrum / agile implementation would have a release plan that simply rolls quarter by quarter, for example. This is a battle for another day.

IX. Samples

These samples show what we used to present the ideas to the executives to get buy-in. In other words, it shows slightly idealized versions of the charts with mocked-up data.

Eventually we had a number of Excel spreadsheets which worked as templates and which we offered to teams to help them start the reporting process. More recently we have been working to generate these reports in Jira / Greenhopper.

A. *Release Burn-up Chart*

Figure 1 shows the release burn-up chart we use. The chart is slightly different to the traditional release burn-down or burn-up chart in that it shows (on a per sprint basis) the total scope as well as the progress toward that scope. Trend lines indicate likely completion. Our Product Owners were the ones who settled on this modification. They wanted to understand the relationship between changing scope and changing velocity over time. Product Owners felt that the standard chart hid this kind of information.

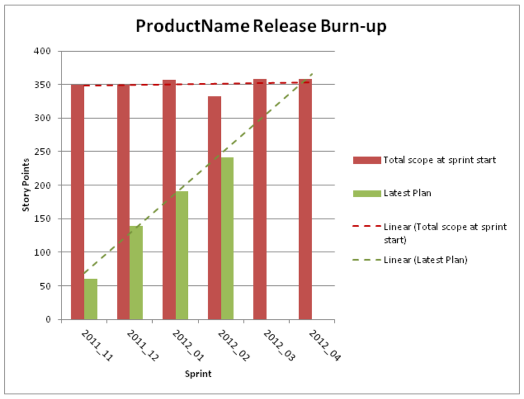


Figure 1. Release Burn-up Chart

B. *Epic Progress Report*

We use epics to capture the main themes of a release. The idea is that these epics represent what our business and sales people talk about when someone asks "what is coming in the next release?" We are interested in progress (to 100%) of the user stories associated with each epic. The data labels show the total number of points for this epic in this release.

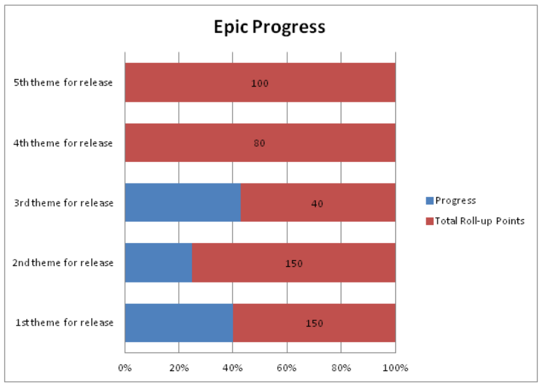
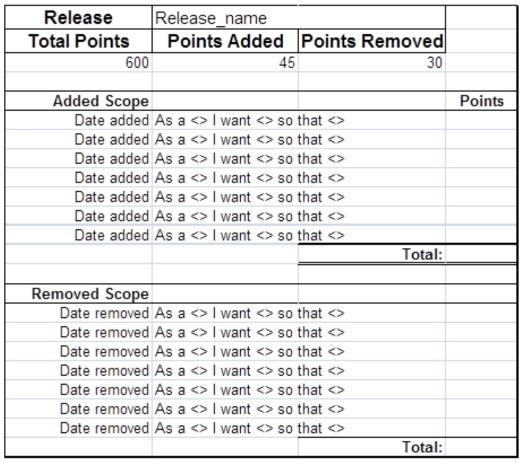


Figure 2. Epic Progress

C. *Scope Change Report*

We save the scope when we complete the release planning, calling this the "baseline scope". We then report on changes to that scope, showing summary information such as a total number of points for the release, what has been added and removed. This is followed by a listing of the user stories that have been added and removed so that everyone understands the current status of the plan.

Conceptually this looks like:



Release	Release_name		
Total Points	Points Added	Points Removed	
600	45	30	
Added Scope			Points
Date added	As a <> I want <> so that <>		
Date added	As a <> I want <> so that <>		
Date added	As a <> I want <> so that <>		
Date added	As a <> I want <> so that <>		
Date added	As a <> I want <> so that <>		
Date added	As a <> I want <> so that <>		
Date added	As a <> I want <> so that <>		
		Total:	
Removed Scope			
Date removed	As a <> I want <> so that <>		
Date removed	As a <> I want <> so that <>		
Date removed	As a <> I want <> so that <>		
Date removed	As a <> I want <> so that <>		
Date removed	As a <> I want <> so that <>		
Date removed	As a <> I want <> so that <>		
Date removed	As a <> I want <> so that <>		
		Total:	

Figure 3. Scope Change Report

D. *Investment Allocation*

This is one of those reports that came directly as a result of meetings with the Executive. The chart shows how much we are spending on the various investment allocations for the release in comparison to the planned allocation. The Actual Allocation is calculated by summing the story points for completed stories in the release in each of the categories.

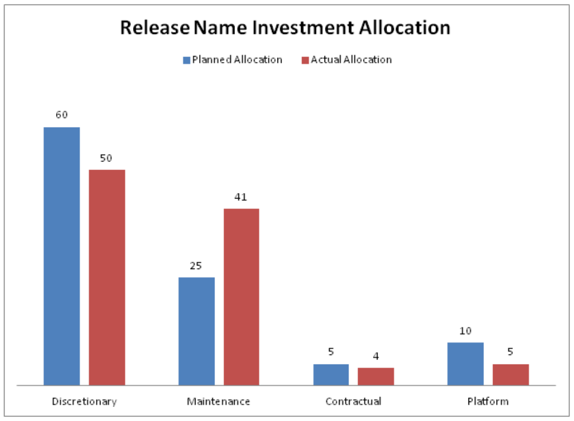


Figure 4. Investment Allocation

Note that the Project Allocation works the same as the Investment Allocation, except it reports on work associated with fielded versions of the products.