Lean and agile financial planning

Rami Sirkiä and Maarit Laanti

Abstract

Enterprises that approach uncertainty and risk in software development based on lean and agile methods often do experience financial control as a restriction. Traditional budgeting and cost reporting is a system based on rigid frames, and it – along with the process of project cost accounting – burdens the lean and agile enterprise with unnecessary and counterproductive overhead and friction. The traditional system provides, at best, a false sense of cost control to any enterprise. In this paper we share insights related to how project-based cost accounting can be transformed to a structure based on agile and lean finance and control. Our insights and examples are intended to remove a key impediment to transforming the finance function of your enterprise to something measurably more lean and agile. This work is based both on an actual case completed with our employer and on our knowledge of financial processes at major Finnish companies.

Keywords: agile, agile software development, beyond budgeting, project costing, budgeting, traditional budgeting, cost management, management accounting, cost center accounting, cost center, lean, lean enterprise, lean finance, SAFe, Scaled Agile Framework, rolling forecast.

Introduction

Agile SW (software) development and traditional cost accounting don’t match. Firstly, traditional cost center accounting expects a long horizon with detailed cost estimates that must be frequently updated. Simply put, agile SW development avoids this detailed and long horizon planning. Secondly, traditional cost accounting values planning accuracy and executes variance analysis against the original estimate. Agile development instead allows for uncertainty: the final content delivery and total cost may be impacted by the empirical feedback received. Thirdly, traditional budgeting draws attention to budget overruns. In agile development, if the initial feedback proves to be positive we want to allow – and even encourage – further investments. In larger enterprises the division of investment can be controlled via the hierarchy of backlogs – the level of investment is dictated by the changing backlog priorities, which means the teams allocate tasks flexibly. Fourthly, traditional project cost accounting requires a re-approval for any delays which increase the project budget. Rigid approval limits pose an unnecessary control, and thus, in agile development, delays that would traditionally be caused by the budgetary approval process means teams can now take their order of development flexibly from the prioritized backlogs.

We had an opportunity to witness our organization’s transition to this lean and agile way of working. What we soon noticed was that the financial function would need to change in response to it, in order to benefit from this adaptive approach. Rami’s role was in financial planning, and he therefore needed to comply with the new principles. Maarit was the designated lean and agile coach within the same organization. Ultimately we together created the required principles, logic and visualizations of what became the new planning solution that supported Agile SW development. The following changes were implemented in our R&D unit, resulting in demonstrably leaner and more agile finance and control processes:

- Cost center planning was simplified as part of the company wide initiative and we additionally removed project level cost control altogether.
- Instead of planning cost centers we created enhanced visibility for deliverables (projects).
- Instead of focusing on OPEX limitations, we focused on competence limitations.
- We mandated that strategically important but longer lead-time investments would have a separate allocation.
- We encouraged people to be transparent with regard to the planning uncertainties.
- We defined clear roles and responsibilities for financial planning, content planning and resourcing.

Our unit’s finance and control function now has the ability to more accurately observe and forecast such things as the OPEX run rate, where the money is spent, and whether our resources are performing the tasks effectively. Resources tied to finance and controlling are now much less than they earlier were, in adherence with the lean principle of value versus waste. Our R&D has the freedom to manage content according to the pace of execution velocity and latest priorities without excessive reporting or re-approval processes required. The entire solution is powered by tool automation.

Our findings can be generalized and are valid for any organization that is doing agile software development. The agile concept provides good tools to control SW portfolio and resource allocation in a modern, more rational way. Traditional finance and control methods and tools designed for the industrial age should not be used, as these are largely in contradiction with the lean and agile paradigm. The new tools and methods are based on a modern understanding of lean, agile and 'beyond budgeting' principles[1].

The problem

During the transition to lean and agile in our R&D unit there emerged issues with our organization's structure based planning. While we had organized a prioritized backlog on the portfolio and team level, we still had difficulties getting resources allocated promptly. Each product owner on the functional unit or team level was focusing on his or her own deliverables, even though the key driver in portfolio prioritization was to maximize customer value. If one of the key deliverables was being delayed, there was no automatic resource reallocation method available. The possible delay in deliveries was clearly indicated in our metrics dashboards. The rigid organizational structure, narrow competence profiles and detailed level of projects with dedicated resources prevented prompt and flexible resource reallocation. The idea of allowing more flexible organization structure with empowered teams was born and supported by our R&D management team roughly at the same time when the idea of more flexible investment planning and control was born.

There were also issues detected on the financial planning side. The tradition had been that R&D planners had been asked to provide estimates for the next 18 months. Although estimating such a long period had always been problematic, the argument was that strategic investment allocation was needed – and thus some kind of data had always been provided. Yet now that the R&D became familiar with agile and lean estimating and planning this argumentation was becoming significantly more difficult. Referring to agile principles the R&D planners took the position that they could plan only a few months ahead.

Traditionally, there was also a requirement to state how many person-months were to be spent on which projects, and when. In order to provide this information, a detailed resource profile at the individual level was necessary. That resource profile defined in which project each person would be allocated for next 18 months. Yet an agile R&D planner would refuse to create such a profile, because approval of the idea of content flexibility automatically means that one cannot plan a large-scale implementation plan ahead in such detail, as the content will almost surely change – even on the detailed SW feature[2] level. Thus it is impossible to provide a detailed investment profile per higher level[3]. The same principle is true for any agile project – thus cost planning being one of the inhibitors in many companies with regard to the adoption of agile methods.

The investment data had been updated on a monthly basis in order to improve the data quality, follow trends more closely and to get quick feedback on any investment reallocations taking place. In the new agile and lean setting these updates were seen as significant and non-value adding overhead.

Naturally, the initial data was changed as a result of these monthly investment data updates. However, only the content experts, i.e. the original contributors, were able to explain why the data had changed. Most typical explanations were that the team had learned more about the topic and the required features had been clarified, or that the architecture was now better understood, or the execution velocity was lower than expected. When more forward-looking information was needed for performance

[1] Beyond Budgeting refers to a concept described in www.bbrt.org (Beyond Budgeting Round Table, 2013)

[2] Here we refer 'epic – feature – story' as a hierarchical structure of the SW work items. (Leffingwell, 2013)

[3] (Leffingwell, 2013)

management – to set up the traditional performance targets – we discovered that efficiency was not improved by the use of targets or with the efficiency metrics. People just didn't know that much about the future – and performance improvement required more than just an ambitious target.

A bit more successful approach was when a catalog of investment profiles was created in order to support future investment planning. This catalog consisted of different kinds of development efforts that were typical in our environment. For example, the productization of a feature always required a specific combination of expertise. Most features had only a few differentiating parameters: the epic, the environment or development platform, and the complexity factor of the content or expected velocity of the development team. Estimating and calculating such a large variety of profiles was quite demanding, and luckily the accuracy of the system never got to a level that we would have used this investment profiling tool as a real control mechanism. If this had been the case, a possible budget over-run would have resulted in extra bureaucracy and overhead, as the re-approval would have resulted in a re-estimation of the complex combination of planned features and their parameters across the organization.

After we took the Scaled Agile Framework model into use with its hierarchy of backlogs – and the decision was made to allow these backlogs to dictate and guide more flexible resourcing, we soon realized that there was no way traditional cost center control practices could cope with the rapid content changes. The investment planning tool was still based on profiling the features in the deliverables, but the content of the deliverables was changing more radically and more frequently! We realized that any cost center control or deliverable-based way to approve, track and reapprove resourcing was not adding any value. Instead, we accepted that it is normal to move resources from one project to another and to spend an extra month or two on the project.

This is often case in other companies that have a portfolio of agile projects as well. While agile development has largely been adopted around the globe, traditional management methods need to be changed in response. This is true also for industries in which software is only part of the solution or the product. For the most part, the value of the service or product is dictated by how usable that service or product is for the customers – a factor that we can only measure via customer feedback. The real value of agile portfolio management is to take in this feedback in order to improve the existing product – and then identify the characteristics that actually result in a good product.

This kind of adaptability in how the product is – and how it is built – requires a new type of management that understands the underlying lean and agile principles. Ignoring either the customer feedback or these principles impacts negatively the competitiveness of the product, at least regarding the software side of it. Finance and controlling is a function that is partially driven by rules and regulations (financial accounting), and the part of it that has more flexibility (management accounting) is struggling to see the need for the change.

Our idea (What)

Our idea was to comply with lean and agile principles and practices. We didn't believe that any organization structure, or any finance or planning process should hinder, impede or add an overhead to how software is actually made. We understood that agile and lean principles provide tools and processes to manage content as well as to support performance and efficiency. We trusted that teams would be as efficient as possible. We acknowledged that when you follow the agile way of working, the content must be kept flexible. We knew that total cost is mostly driven by headcount and thus is pretty stable across all of organizations. We accepted that SW developers didn't know what they were going to do in the mid/

long term future. So we removed the control and relied on trust. For many this would seem to be a radical approach, but we knew that our leap of faith would be rewarded by transparency, and any potential errors would be corrected by rapid feedback. What we did is summarized here:

- We created resource pools and more generic competences instead of small cost centers and detailed profiling.
- We simplified the cost center planning by taking it to a higher level of granularity. Cost centers of 30-50 persons no longer provided rolling forecasts for the next 13 months for selected accounts. Forecasts were expected one or two levels up in the organization structure, i.e. cost center for 300-500 persons.
- We downplayed cost center planning and implemented visibility for deliverables. Cost center plans – for cost centers of 300-500 persons – were built after the deliverables had been resourced, based on the deliverable view.
- We promoted a culture of transparency regarding what we know and what we don't know. We showed unallocated resources after the four to six months. So if teams and planners did not yet know what they were going to do, they marked their resources as unallocated in the capacity planning.
- We highlighted the role of competence management as a constraint instead of OPEX. Portfolio planning focused on bottleneck resources, OPEX planning was secondary.
- We redefined the roles for the capacity management and cost center planning. Cost center planning provided visibility to OPEX run rate on a higher (cost center) level, and deliverable -based capacity management details on an effort profile for each epic.
- Task fluidity. We lifted the project level cost control: content management was done by project office and efficiency metrics was left to the teams themselves. No changes in the epic schedule or resourcing required centralized re-approval. Our project management office had the authority to reallocate tasks between teams.
- We created business plans based on minimum content, and nice-to-have's were just extra. Additionally we allocated a specific lump sum to support innovation and quality.
- Our company had already earlier separated forecasting from target-setting, which allows for more realistic forecasting and more ambitious target setting.

These changes enabled our finance and planning function to become truly agile and lean. These changes are also a true enabler of agile and lean development. They don't just remove the impediments that traditional finance and planning impose on an agile and lean organization, but, together with a flexible organization, also enable deferred decision making when planning the investments. Deferred decision making or real options here means that based on a feedback, an agile and lean organization may balance their investment portfolio based on the field feedback simply by reprioritizing the backlog priorities. A rising priority indicates that more work must be done to the prioritized epic or feature. A company with multiple agile projects may thus flexibly balance the resources and the investments across the projects based on received feedback and needs, as well as estimated payback functions. This is true option thinking.

The Details (why and how)

Shorter planning horizon and room for uncertainty

A requirement to collect long-term estimates for epics comes from a waterfall approach to planning. Traditionally, it is expected that we would be able to accurately estimate the resource needs, the content (value added to user / consumer) and the completion date for each requirement. The same was now expected for epics, although on the financial planning side.

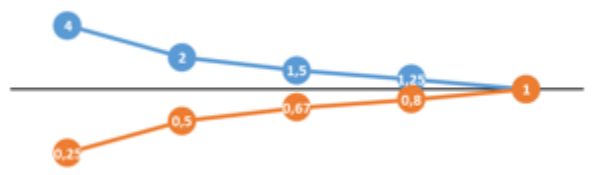

Figure 1 Cone of uncertainty (Boehm, 1981)

The Cone of Uncertainty states that the further we try to estimate into the future, the more uncertainty there is in these estimates. The cone of uncertainty states that there are only two ways that the estimates can be more accurate – that we advance in time and gain more information on the project or that we estimate over shorter periods of time. Originally, the cone of uncertainty was used to explain why all given estimates needed to be multiplied by four.

The agile approach suggests that since change is evitable, we should try to make estimates across shorter time periods. Some agile teams are thus refusing to give any long-term estimates, claiming this is simply wasted effort. Scaled Agile Framework points toward a two-level lean way of planning: plan roughly for the longer time horizon, and more accurately for the shorter time period.

An additional, complexity-based explanation to planning is also available: the smaller the time-window we use for estimation, the less internal and external coherence is happening. Software development is an art for solving extremely difficult problems – and the solutions themselves cause additional problems, with the system under development thus constantly interacting with itself.

Agile software development is an attempt to solve the problems encountered in traditional development. While traditionally projects fail to deliver on-time or on-budget, and fail to fulfill customer needs for quality within the holistic project timeframe, agile introduces adaptive, self-corrective control mechanisms that are based both on internal and external feedback and on measurements of the intermediate results. In this way the quality promise can be kept.

We decided to abandon the cost center or deliverable-based approach for budget approvals or controlling. We did collect the long-term epic estimates, but centralized this effort to a few experienced planners. We were also very cautious with regard to how this piece of data was used: we acknowledged that the estimates were not accurate, and were not intended to be treated as an agreement on resource availability, consumption, or as a promise of the content.

The estimates from experienced planners were used to calculate the OPEX distribution of the deliverables. This deliverable-based view was valuable to both the R&D management team as well for frequent ad hoc analysis. After all, epics and HW (hardware) deliverables were exactly the content most cared about and talked about, and having an investment number next to a list of items gave insight to investment distribution.

Traditionally, cost center accounting as well as project accounting forms a baseline against which the actual numbers and latest estimates are compared to. The problem with this kind of reactive tracking is that it only reveals problems when the gap between the originally planned baseline and the actuals grow big enough. It's only then that you need someone to explain whether the changes are acceptable or not. For a traditional company this approach may seem feasible – the more software you develop, and the more you want your content to evolve to satisfy the customer needs, the less you want to prematurely define your investments and opportunities for the future. The more software-driven a company is, the more it wants to adapt to emerging needs, and the more it would like to preserve room for changes. Another way to put this is that the more uncertainty there is, the more real option thinking needs to be allowed to play a role in the future planning.

The agile way of working thus changes how uncertainty is approached. Instead of multiplying estimates by four and trying to banish uncertainty by planning in small details, we use more accurate short term planning and only rough long-term planning at the project level. Why it is then so hard to estimate accurately? One reason is that the old way of planning with 100% utilization rate leaves no room for unexpected changes. Whenever an error happens, there is no mechanism to make the corrections. Secondly, our portfolio has often consisted of multiple parallel projects, and changing between projects creates extra overhead, and delays all of the ongoing projects. The better way is to sequence the projects one after the other, and to use pull-based scheduling. This way one can optimize the portfolio of projects based on the cost of delay. Thirdly, when a new project is started, it is never started from an "empty table" – traditionally, we have not given any attention to the projects already started. In agile portfolio management all new projects are planned in smaller batches – from a single project we create 1-3 epics – and all epics are prioritized based on existing work and existing priorities. A project may also have to prove it is executable. If a project does not show promising progress, it will drop in our priorities hierarchy and in resourcing.

The pull-based scheduling works naturally with queuing principles. Instead of 100% utilization rate we plan for 80 or 90% utilization. This allows for better tolerance for errors and small changes or surprises along the way. The slack is also an enabler for all innovations, as Tom DeMarco describes in his book of the same name. (DeMarco, 2001)

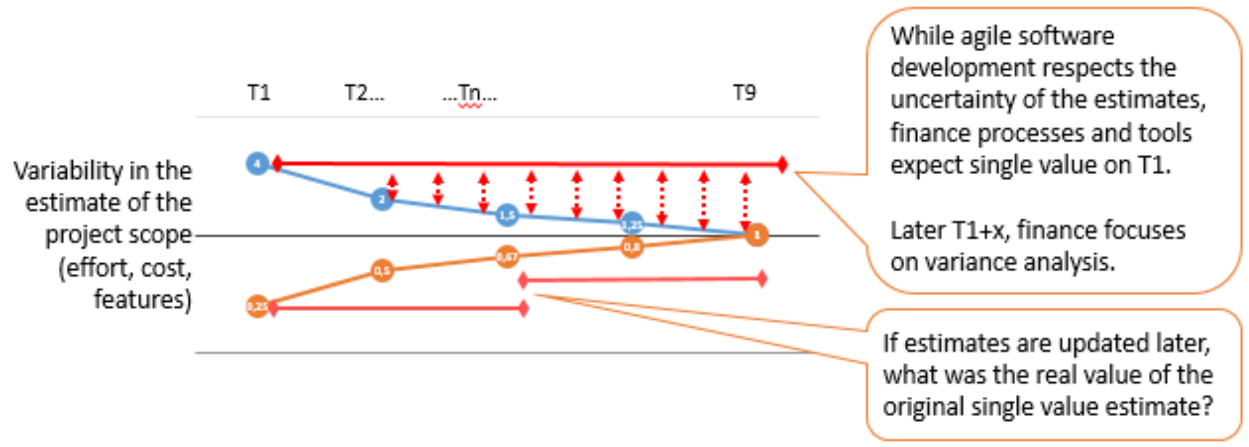
T1
T2...
...Tn...
T9
Variability in the estimate of the project scope (effort, cost, features)
4
2
1,5
1,25
1
0,25
0,5
0,67
0,8
While agile software development respects the uncertainty of the estimates, finance processes and tools expect single value on T1.
Later T1+x, finance focuses on variance analysis.
If estimates are updated later, what was the real value of the original single value estimate?

Figure 2 Traditional financial planning cannot address uncertainty

No CEO in any company will forecast exactly what the company share price will be after next year. Is natural to include the uncertainly, and communicate the estimated share price within a range of probability. It’s a bit odd then that as we go down the organization ladder, this uncertainty vanishes and project completion schedules are routinely given with accurate dates.

Our solution was to allow ‘delivery windows’ in the product roadmap. Naturally, marketing, hardware deliverables and manufacturing need a target date – and typically they prefer a fixed, committed date. However, if this launch date has been fixed too early – given the inherent uncertainty of SW and HW development – lots of resources are wasted if the launch date is missed, such as already booked marketing campaigns or manufacturing capacity.

Traditionally, major bonuses were tied to a given launch date in order to secure the date. However, pushing too hard toward a launch date can result in cutting corners in order to receive the bonuses, which may result in diminished quality in the first releases, and disappointment of early, loyal promoters. In cases in which the launch date was missed, people lost their bonuses, which subsequently resulted in second-guessing the next launch date.

Now with the ‘delivery window’ all stakeholders are able to see the level of accuracy in the planning: how long the window was, when did it start and end, and also the documented rational for the related uncertainty. Revenue planning was based on the end date of the window as a safe and conservative estimate. The rest of the portfolio was kept open and only when the maturity increased was the schedule fixed and resources promised for next project in the prioritized backlog.

Another part of our solution was to enable ’unallocated’ classification in our capacity plans. This new classification was explained to planners so that they would not have reason to hide the uncertainty but rather keep transparency throughout the planning and reporting phases, and align the plans with the reality. Our plans reflected exactly the uncertainty that the management was talking about in their meetings – the range of risks, delays and consequences.

Abandon detailed project cost management

Our third and fourth issues were about attention to budget overrun and resourcing re-approval upon an unavoidable change in the epic projects. We define epic project here as a group of PSIs[4] or trains, targeting a single delivery and launch. In traditional projects, the project cost, scope and/or schedule usually fail at some point of time. Sandbagging (i.e. the multiplication of the estimates with concealed numbers) is sometimes used so that problems might be hidden, but sooner or later – usually after a few rounds of cat-and-mouse regarding the target setting – people must come out with a fixed number and approve a budget where no surprises are allowed. That means that when further changes happen, the whole budget needs to be re-done and re-approved.

A company with a portfolio with multiple agile projects may need to keep a few “firemen” in order to be able to rescue a project that has run into a trouble. Our organization allowed the teams to self-organize when needed in order to be able to ship the deliverables that were requested via the prioritized backlogs. We had a hierarchy of product owners to help prioritize the backlogs, and who took part in agile release train (ART)[5] planning to ensure that everyone worked towards the shared vision.

[4] PSI or potentially shippable increment is the larger development time box that uses cadence and synchronization to facilitate planning, provide for aggregation of newsworthy value, and provide a quantum unit of thinking for portfolio level considerations and roadmapping. (Leffingwell, 2013)

[5] See (Leffingwell, 2013)

The theory of constraints tells us that any system is capable of delivering at the throughput rate of its weakest link. Hence, the key point is to understand the limiting factor at each business/ level/ function in order to execute proper resource allocation. Any effort to understand details in a non-bottleneck part of the project should be challenged – as the decision would be made based on the bottleneck resources anyway. This approach is adapted from Theory of Constraints by Goldratt, in which one should first identify system's constraints and then make efforts to maximize the output of that limited resource. (Cox & Goldratt, 1984)

R&D costs come mainly from two sources – the headcount driven resource costs (internal or external) and investments on development environment (indirect) or prototypes (direct). The investments are known, and can be planned at the same time when backlog priorities are decided and epic projects started. In our R&D the total cost comes mainly from headcount, and the overall annual cost remains quite steady from period to period.

We abandoned the project-level budgets that needed continuous updating and re-approvals. Thus there were no project overruns, and no variance analysis against to the original plans. We realized that even if a project's execution velocity would be slower than expected or the project's scope would be changed, i.e. the project's cost would increase significantly; we could keep the project priority high, which would "automatically" allocate more resources onto it. This would need no re-approvals from any controllers. The total cost of the whole organization would not change, only the least prioritized projects would not get any resources and thus would not get done – at least, not now.

The priorities between Epic projects impact how much total value is created on each 'Potential Shippable Increment' (PSI e.g. 8 week period which targets delivering something shippable). It does not add any real value if the cost of each project is followed separately and re-stamped by management. If a project looked promising, but needed a bit more resourcing, the priority was kept high and resourcing was secured with that priority decision. If the value added from extended work was less than with other projects, priority was dropped and thus other projects were resourced. Release train backlog provides a good control mechanism – having traditional project cost control on top of that does not add any value. (Figure 3)

Even though it is hard to estimate the total cost per project beforehand, it is easy to track the cost once the PSI gets done. You simply need to see from the team backlogs how many days each individual spent on which feature. For us, there typically exists one-to-one mapping between features and projects. But we didn't have mandatory individual level update requirement, thus we used earlier described resource data collection to create visibility for epic level costs.

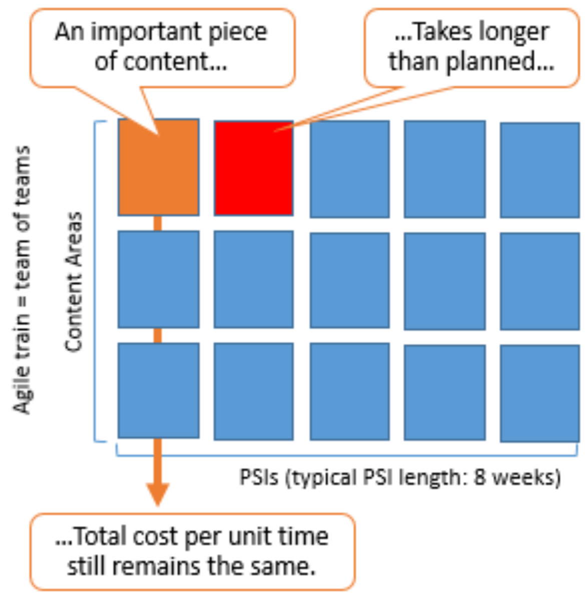
An important piece of content...
...Takes longer than planned...
Agile train = team of teams
Content Areas
PSIs (typical PSI length: 8 weeks)
...Total cost per unit time still remains the same.

Figure 3 From cost control to content management

In agile development we typically want to flex the content, not the resources or schedule. So in this classic triangle (resources – schedule – content) you have the two corners fixed, and the third one is floating. (Figure 4) The resources in our organization were fixed, as we were constrained by competences, and thus unable to increase resources on demand. The schedule is also fixed because we have timeboxed the deliveries into sprints and PSIs. Quality is not scarified, but all Sprint and PSI releases ship with good quality.

The third corner: scope, content, customer value – is floating. The controllers may also label that corner as revenue. This turns around the entire traditional project revenue-cost planning: traditionally, we had fixed content and thus fixed revenue, but upon delay we added resources and thus created additional costs to a project. Traditionally, the revenue estimate has not been changed.

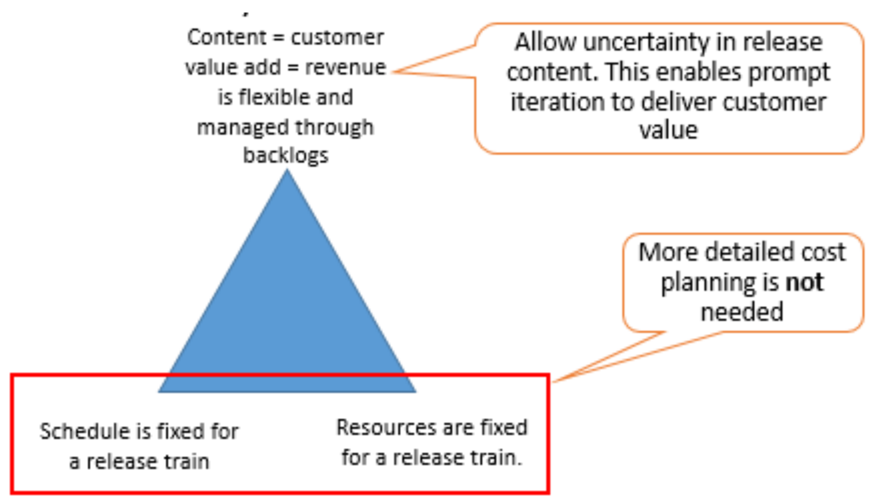
Content = customer value add = revenue is flexible and managed through backlogs
Allow uncertainty in release content. This enables prompt iteration to deliver customer value
More detailed cost planning is **not** needed
Schedule is fixed for a release train
Resources are fixed for a release train.

Figure 4: agile transfers variability from cost and schedule to revenue

Flexible revenue – like flexible output from the outsourced project – is a challenging concept to any business-oriented person. But in real life the revenue typically depends on not just what we do but also on what the overall market situation is. The flexible revenue model raised the question of whether the teams were working as hard as possible if they had not committed to a fixed scope.

We made several attempts to measure the effectiveness of the software development teams. But at the same time there seemed to be so many major architectural and way-of-working changes that we never reached a stable environment for any velocity or cycle-time metrics.

In an environment where velocity is going up all the time the best way to measure completeness of a PSI is to use cumulative flow diagrams on feature / week basis. The shape of the CFD enabled us to estimate when a project would be complete. Alternatively, if the CFD curve was too flat – as was the case in some of our projects – it enabled the management to take the necessary actions: to raise the project priority or to cut the content or the feature inflow.

We now have a set-up where agility replaces accuracy and trust replaces control. It is far more important to get visibility via information radiators and focus on removing bottlenecks, rather than explaining variances to original effort estimates. Backlogs enable the decentralization of decisions so that decisions happen in timely manner rather than creating situations in which teams are left waiting for permission to proceed.

Related Work

During our journey to lean and agile we learned about the Beyond Budgeting concept. I found the principles matched perfectly with agile software development principles in many ways. We also met smart and kind people both from Handelsbanken and from Statoil, and their insights into modern management accounting were useful and enlightening.

We brainstormed all the ideas with our unit Finance & Control directors, and I received key guidance and support from our boss. We got the approval and go ahead from the R&D unit head. Under the corporate policy these persons will remain unnamed.

We would also like to give a big thank to whoever in our large organization initiated Simplified Planning. This approach enabled lots of documented changes.

Certainly all start-up companies without strong controller resources – or which are otherwise insightful – have already solved the challenges described in this paper. Hopefully you can share your knowledge with us.

Conclusion and further work

Our finance and control process is not an impediment for the journey to agile and lean. Instead, finance and control as a function is pretty lean itself, as it focuses on total R&D level cost and leaves content management to agile ways of working.

We are observing that many other organizations are becoming aware of the impact of the finance and control on agile software development. The awareness is increasing amongst SW related personnel – yet not with the finance function. We would be thrilled to learn of just how many other companies are facing similar issues, and then see role of finance changed – naturally in Finland first.

Additionally, further work would need to happen in the area of system support. For example having the SAP R/3 finance and control module update to support ranges instead of single value points only – and likewise updating reward and bonus systems to align with the agile way of working.

Bibliography

Beyond Budgeting Round Table. (2013, 12 31). Retrieved from http://bbrt.org/

Boehm, B. (1981). Software Engineering Economics.

Brooks, F. P. (1975). The Mythical Man-Month: Essays on Software Engineering.

Cox, J., & Goldratt, E. M. (1984). The Goal.

DeMarco, T. (2001). *Slack: Getting Past Burnout, Busywork and the Myth of Total Efficiency.*

Leffingwell, D. (2013). *SAFe Glossary*. Retrieved from Scaled Agile Framework: http://scaledagileframework.com/glossary/