The top 10 pitfalls of agile capitalization

As part of your agile transformation, have you engaged with your finance and accounting...

By Catherine Connor   February 16, 2016

Maybe you’ve read “Why Agilists Should Care About Agile Capitalization” and you’re trying to help your finance group make sense of agile.

If your organization capitalizes software development, chances are Finance and Accounting are struggling to make sense of capitalizing agile software and you may run into one of two risks: either Finance conservatively expenses all agile labor costs, short-changing
your organization from its true valuation, or
Finance obstructs agile adoption because they’re
comfortable capitalizing waterfall development
but not agile development.

In a world where business is being rewritten by
software, agile capitalization is becoming a
major financial concern for organizations
growing their agile practices, to the point of
becoming material to profitability.

Because capitalization guidelines captured in the
Accounting Standards Executive Committee
(AcSEC) Statement of Position (SOP 98-1) were
written in 1998, when waterfall development
was prevalent, these guidelines are challenging
to interpret for agile development. As a result,
financial conservatism leads many organizations
to expense all agile development work. With the
labor costs associated with agile development
growing exponentially, finance groups are
starting to ask:

**Is there a way to interpret the SOP 98-1
guidance to effectively (and defensibly)
capitalize labor costs on assets being
created using agile practices?**

Not only the answer is a resounding YES, but
the potential for effective expense management
is higher with agile practices than with waterfall
practices. Because of agile’s strong focus on
value, agile teams create more financial assets. Smart companies are realizing that agile capitalization can indeed be a big contributor to greater business value.

At its core, agile is about increasing value and delighting customers. You can delight customers without reporting on and depreciating the software assets delivered by agile (through capitalizing project labor costs), but if you expense all the agile labor costs you could be capitalizing, not only do you not gain the financial benefits of agile, your company could appear less profitable than it really is: that’s what has companies paying a lot of attention to this topic.

Over the past several months, as part of our continued research connecting agile and finances, I had the privilege to spend quality time with Pat Reed, director of the Agile Alliance Agile Accounting standard initiative. Not being a financial or accounting person, I marveled at how clearly Pat explains what many consider a complex and boring topic. Low and behold, I found myself getting excited about agile capitalization! (I know, how can anyone possibly get excited about capitalization?) Well, there’s a huge opportunity to help organizations avoid waste while promoting best agile practices. Here are the key ideas to keep in mind:

- The WHY. Remembering the original intent of the Generally Accepted Accounting
Principles (GAAP) is key to companies creating *accurate*, *consistent* and *defensible* capitalization solutions. Broader International Financial Reporting Standards add that financial reporting should be useful to investors, helpful for making decisions and helpful in improving the performance of the business.

- The WHAT NOT TO DO. As companies look at updating their capitalization policies to more accurately account for labor costs independent of the development method AND improve performance of the business they need to watch out for pitfalls that can lead to unnecessary wasteful activities (and, ironically, are contrary to the intent of the accounting standards.) Little guidance is published and the little published is creating confusion and misunderstanding. Teams I work with know my motto: “just because you can does not mean you should,” which has served me well as a product manager. How often and how strongly can you say “No” to focus on the right thing? Agile capitalization reminds me to bring back this motto: just because you can over-engineer capitalization policies by, for instance, computing cost per story point, does’t mean this is the right thing to do.

In this blog I want to share what I’ve learned from Pat so that you can benefit from her
practical and proven experiences. Let’s go over the pitfalls to avoid.

#1: Expecting technical team members to know about capitalization

The idea that engineers should know which bucket (CapEx or OpEx) their stories fall under is a form of waste as disruptive as asking them to track their time. Agile teams should be buffered from any CapEx/Opex categorization. Their role is to focus on delivering value fast, and anything that gets in the way of meeting this goal should be eliminated. ScrumMasters, however, have a very important role in implementing their company capitalization policy: building transparency and trust with Accounting. By focusing the teams on value delivery and removing impediments getting in the way of that goal, they are the “guardians” of capitalizable software assets. A good ScrumMaster helps teams focus on delivering their feature commitments AND on continuously learning and improving, so they reduce the cost of value (e.g. software “asset.”) ScrumMasters are closest to identifying the teamwork that may fall under Opex because work was aborted (for good reason). All developers should know is whether they work on feature work or fixing production defects; capitalization should not require any developer expertise beyond that.
Assuming time tracking is required

Time tracking is prevalent in the industry because for decades, common financial practices have relied on hours to derive costs. There are valid uses for time tracking—such as when time is a traceable element for paying employees, especially in unionized, specialized industries such as the movie industry and the oil and energy industry, where the type of work an employee does directly impacts how much they are paid (ex: time-and-a-half.) In software development environments with heavy contractor use, time tracking is most often used to manage the payment of those contractors. For companies already using time tracking, agile capitalization can leverage the hours that agile team members entered on timesheets to derive labor costs.

However, organizations not currently tracking time should not assume they need to start tracking time to achieve agile capitalization. Much research has been done on ways to derive costs without requiring timesheets. In organizations where a clear delineation of engineers’ time can be defined by the activities they work on, you may use that work delineation as the basis for cost allocation. Organizations mature in their agile practices are able to dedicate agile teams for extended
financial periods of time—say, a fiscal year. As a result, all the team members are assigned to a single cost center and payroll and benefit expenses can be used to identify the cost of the team. In this way, many organizations have been able to derive labor costs from the burdened rates of dedicated teams without requiring time tracking.

**#3: Concluding that agile development cannot fit in the SOP 98-1 phases**

The irony of this pitfall is that it conflicts with the original intent of SOP 98-1. If you want to capitalize agile software, then you have to adhere to the GAAP Consistency principle, by definition of which capitalization policies must be consistent across waterfall and agile practices (throughout your organization and across industries.)

To maintain investor confidence—the SOP 98-1 “raison d’être”—and respect the GAAP Consistency principle, capitalization policies need to interpret both agile and waterfall development in a consistent way.

A common argument is that with requirements emerging continuously, and with continuous delivery, agile development cannot fit in the SOP 98-1 preliminary, development and post-implementation phases. Agile development can
absolutely fit in the three phases. These phases simply need to be interpreted for agile development.

* The *Preliminary* phase is about feasibility before deciding to invest further. Agile is not precluded from approval decisions, nor are features produced by agile teams worked on forever. Just because you adopt agile does not mean you get automatic approval to obtain all the investment funding you want. Market analysis, creating a lightweight business case and Enterprise Lean Startup MVP (before reaching the product-market fit pivot) all fit in the Preliminary phase. Think of this phase as the WHAT that’s being articulated before the investment decision can be made.

* The *Development* phase captures all activities related to creating the asset (the WHAT approved in the Preliminary phase) that represent HOW we will deliver the value expected out of the approved investment.

* The *Post-implementation* phase is where defect corrections, post-production deployment and ongoing maintenance of the asset fit in.

**#4: Forgetting why the GAAP were created**

The Generally Accepted Accounting Principles (GAAP) follow rigorous principles to ensure accuracy and consistency across industries and
maintain investor confidence. Staying true to the GAAP’s four constraint principles—Objectivity, Materiality, Conservatism and Consistency—is the key to keeping the capitalization rules simple and defensible, and avoid falling for an over-engineered solution driven by excessive fear-based controls, or fear of auditors finding skeleton in your labor costs closet. These fear-based behaviors can actually block rather than enhance business performance. A good capitalization policy (one that is “auditor-friendly”) is consistent with the company respect for Objectivity, Materiality, Conservatism and Consistency.

**#5: Forgetting who owns creating an “auditor-friendly” capitalization policy**

Technical accountants (not agilists) are on the line to defend the capitalization policy to auditors. They’re the designers (and among the final decision makers) of how and what when it comes to capitalization. They have a responsibility to defend the policy to internal auditors. This is not the agilists’ place to prescribe what falls under Capex vs. Opex. The key to an “auditor-friendly” solution is to co-create policies with technical accountants and IT leaders as equal stakeholders: explaining our respective worlds to create a policy defensible by accounting and traceable to agile work.
Everyone benefits directly or indirectly from the business outcomes linked to agile capitalization: higher profitability, better company valuation and more hiring power to increase our capacity to deliver more customer value. We all win when auditors don’t find significant findings that put our company in the news or at risk of having to restate earnings.

**#6: Treating capitalization as a technical problem**

As professional engineers we are naturally drawn to do things we can do, at the risk of forgetting to assess whether we *should* do those things. Capitalization has a bit of a technical challenge to it: is it Capex or is Opex? Can I programmatically code that decision? What we forget is that the answer is not black and white, and may not be the same from one organization to the next. That’s because each organization has a different tolerance for Objectivity, Materiality, Conservatism and Consistency. You probably can think of some companies more conservative than others.

More importantly, what makes agile capitalization not a technical problem is its complex adaptive nature. After so many years working in this domain, Pat has yet to find two customers who are solving for identical problems. Although the outcomes of each will
need to be consistent within the organization and adhere to GAAP, each organization needs to work through a unique and complex set of challenges that have evolved based on culture, practices, tools and methods. Complex adaptive problems are best solved by creating a strong collaboration between all parties—in this case technical accounting and lead agilists (and possibly financial reporting, portfolio management and audit and compliance)—to co-create a solution that leverages and honors the respective expertise.

This pitfall is accentuated when friction or animosity has built up over years between Finance/Accounting and agilists. The first step is to help expose and work through that friction by empathizing for the respective roles, as we simply cannot, as technologists, be the ones prescribing the capitalization solution to our company (or pretending to have all the answers) when accounting is on the line to defend the policy. The agilist’s role is to help Finance and Accounting understand how agile development works and collaboratively determine what needs to be reported to provide the transparency that builds trust.

**#7: Over-engineering the solution**

Over-engineered solutions—many around leveraging user story points—are often rooted in
fear and outdated beliefs that *more* controls are better. Over-engineered solutions focus on handling any and all impairments, which creates unnecessary waste. To remove that fear and limit the waste, we must build a trusted relationship with those actually dealing with the auditors. Technical accountants have a working relationship with internal auditors to create “audit-friendly” solutions.

The KISS (“Keep It Simple, Stupid”) principle reminds us that a simple solution is actually more defensible than an over-engineered one. Capitalization is amazingly not that complicated in nature (at least considering what agilists need to understand.) With “software eating the world,” software is becoming a major financial asset. Finance simply needs to understand which assets are being released. Do we release stories? No! We implement stories but we release features. Features (or bundles of released features) are the financial assets. Feature releases are deployed to customers, creating natural, capitalizable bundles. Finance does not like “stories” figuratively and figuratively : ) Finance does not like estimates, either, and story points are an estimate of effort.

A defensible solution requires actuals, not estimates. Very few agile teams really track their *actualized* story points, which would be considered a form of waste since that activity
does not contribute to delivering value. Capitalization solutions relying on calculating cost per story point are not only over-engineered and wasteful, they run the risk of creating capitalization rules that are too complex to manage, as they force accounting into putting financial controls at too low a level of granularity. Remember the Consistency principle at the heart of the solution? Many agile teams don’t even use story points (for example, Kanban teams,) so the Consistency audit test would likely fail.

Over-engineered capitalization policies often have their roots in poor agile practices. Good agile practices lead to features predictably delivered in a release, where all release-related labor costs can be considered for Capex. Occasionally stories not completed for a feature release are identified and labor costs associated with their efforts should be reported as Opex, but good agile practices would make that an exception (and an opportunity for learning and improving.) Ideally the effort would be recognized early enough to limit labor costs to a point of non-materaility.

Important to agilists is this: the stronger your agile practices, the simpler your capitalization policy can be. Strong agile practices focus on delivering valuable, high-quality features that translate into financial assets with limited production fixes and minimal overhead
“expense.” Past the feasibility (investment approval) phase, most of the agile work can be considered for Capex. Exceptions, such as stories not completed for a release, can simply be reported by ScrumMasters to technical Accounting for a materiality assessment before deciding on Opex. Favoring conversations over tools and processes, you can trust ScrumMasters to have the conversation instead of wastefully automating story-level tracking.

Bottom line: a simple and defensible capitalization policy leverages strong agile practices and has provisions to handle the exceptions.

**#8: Bending agile practices to fit in waterfall-based phases**

Organizations familiar with waterfall software capitalization and new to agile have a tendency to fall into this pitfall. Instead of fitting good agile development practices into the (designed-for-waterfall) SOP 98-1 phases, they bend their agile practices to make the mapping of agile labor costs similar to waterfall phases. For instance, they may create Design stories, Implementation stories and Test stories, then apply the labor cost of each story to the respective GAAP buckets, expensing Design and Test stories and capitalizing the Implementation stories. As described in the previous pitfall,
when adopting good agile practices you should not have to track Capex/Opex at the story level. Once the feasibility phase is achieved, all development activities that can reasonably be traced to delivering features—the capitalizable financial assets—fall into the Capex bucket, including ScrumMaster efforts to remove impediments and as agile ceremonies like standups and release planning.

#9: Assuming one size fits all

Wouldn’t this be easier? As we mentioned in #6, capitalization is a complex, adaptive problem. A successful solution in one company would not pass Finance’s muster in another. Every company needs to create its own capitalization policy. That’s because every company has its own culture and strategy around adherence to GAAP principles: different levels of comfort regarding Conservatism and a different threshold for Materiality, as well as asset management and depreciation. Copying another organization’s policy won’t create a solution for your organization. The key to creating a simple and defensible solution that works for your organization is to involve the right stakeholders from the get-go—Finance, technical accountants and agile leaders—and have them co-create a no-waste, defensible solution that both sides (Finance and agilists) understand, can
implement, and that will leverage agility to
enhance the performance of the business.

#10: Approaching the problem in a non-agile way

Agile and Lean are mindsets. A practical
interpretation of an agile vs. fixed mindset
includes the realization that estimates are
increasingly flawed in conditions of extreme
change and uncertainty. Applying agile
techniques to craft your agile capitalization
policy is a great example of taking agile beyond
engineering. The objective of Lean accounting is
to eliminate waste, free up capacity, speed up
the process, eliminate errors and defects and
make the process clear and understandable.
Instead of creating a tomb to document your
capitalization rules, or creating heavy processes
to track every corner for an audit-sniffer, start
with a Minimal Viable Process approach (and
include a “test” for your capitalization practices
and policy.) Bring Finance and IT stakeholders
to the table to validate and iterate on the MVP.
Favor individuals and interactions over
processes and tools. Establish trust and
collaboration. Provide transparency and
traceability. Focus on continuously learning and
improving to the benefit of effective expense
management and optimized business
performance. You’ll be amazed at what satisfies
Finance stakeholders once they understand the difference between agile development and waterfall development. They may become your biggest agile supporter!

So where do you go from here? As a start, reach across the aisle and take your technical accountant friends to lunch to initiate the partnership:

- Understand their viewpoints on Objectivity, Materiality, Conservatism and Consistency.
- Educate them on how agile focuses on value delivery, to create more fiscal responsibility in expense management than waterfall development.
- Share these excellent resources with them:
  
  Agile Capitalization for Greater Business Value
  
  Project Labor Cost Accounting for Agile Projects
  
  Accounting for capitalization of agile labor costs

Most importantly, as you read the growing number of published articles and guidance on agile capitalization, remember to use your pitfall detector! This year we expect to see more and more companies looking for ways to accurately and defensibly capitalize agile software development, and consequently there’ll be more and more agilist advice on how to implement agile capitalization without the accounting considerations discussed in this blog. There’s urgency to ensure companies striving to increase their business agility don’t create waste
updating their capitalization policies with over-engineered solutions, just because they can.

I sincerely hope this will prevent your organization from wasting time falling into any of the 10 pitfalls. If, after reading this blog, you need help kickstarting your agile capitalization initiative to a co-create simple and defensible capitalization policy, please contact us. We can help you establish the Finance/Accounting mental models to capitalize on the financial benefits agile delivers.

Catherine Connor
Catherine is an Agile Solutions Specialist at CA Technologies where she focuses on bringing the...

See all posts >
We’re shutting down – time to celebrate!

2 comments • 5 months ago

Kavitha Tumuluri — Jin! thanks for sharing the great experience. The experience that you and your team have gained out of this

Mainframe: platform of choice for machine learning and ops intel

1 comment • 3 months ago

Jim Mercer — Makes good sense. PTTA Predictive Time to Avoidance is the key. By the time, you are working on a repair (MTTR),

Does your monitoring lack Contextual Intelligence?

1 comment • 2 months ago

Sham — Log analysis? Event Mgmt(AEC) component in CA NSM.

Breaking the hardware refresh cycle

3 comments • 2 months ago

Pete Harteveld — David - In light of CA's inability to name the State CIO, we can certainly exclude the CIO for State of Pennsylvania.
Data centric security: Stop treating the mainframe separately
By Ashok Reddy  May 16, 2017

Learning about the magic behind customer experience
By Dayton Semerjian  May 15, 2017

SD-WAN: Just the ticket for greater flexibility and lowered costs

Cybersecurity executive order targets two common attack vectors
By Mordecai Rosen  May 15, 2017

The boon and bane of technology advancements for...
By Guest Contributor  May 10, 2017

The missing pieces in the European Digital Single Market puzzle
By Marco Comastri  May 10, 2017

Utilizing machine learning to secure financial networks
By Ayman Sayed  May 9, 2017

Three ways to crack the test design code with model-based testing
By Gedeon Hombrebueno  May 8, 2017

Are you drowning in data, but starving for insights?
By Viki Paige  May 5, 2017
General

What does Continuous Delivery have to do with assembling bikes?
By Aruna Ravichandran  May 1, 2017

DevOps

Has the buzz around Docker hidden its complexities?
By Mark Emeis  April 28, 2017

Application Performance Management

APM monitoring governance: The Jurassic Park...
By Cary Andrews  April 27, 2017

MORE STORIES

rewrite

Insights from the app driven world

Subscribe Now>

RECOMMENDED

Why Do We Keep Clicking Malicious Links?>

The Evolution of Serverless Computing>

Introducing the Augmented Customer>
The latest thoughts and concepts from some of the great minds in Enterprise IT.

Read more

Get the latest Highlight in your inbox.

Start here >