Product-focused organizations create opportunities to

CAPITALIZE A HIGHER AMOUNT OF COSTS FOR INTERNAL-USE SOFTWARE

LEADINGAGILE

PREMISE

High performing software development teams that are transformed to persistent teams focused on continuously building products within an agile environment can have a higher portion of their costs capitalized as internal-use software compared to legacy waterfall environments because they are spending more time truly developing software vs. planning and reorganizing teams for each project.

KEY CHALLENGES

- As companies transform their IT organizations to become more agile there will be an opportunity to make a dramatic shift within the application organization - moving from project-based delivery to product-based management or even focused on individual product capabilities. These changes in organizational design, development approach, team funding, operations, etc. afford an opportunity to reevaluate a company's accounting for internal-use software and related capitalization policy.
- Within an agile environment where cross functional teams operate efficiently, the velocity of software development is significantly more rapid than historical waterfall environments so more costs can be capitalized because there are less administrative tasks and distractions that the development team are burdened with.
- Accounting for internal-use software under ASC 350-40 is based on waterfall methodologies but while adoption of agile has certainly increased, uncertainty continues to exist on how to account for the capitalization of software within an agile methodology.

RECOMMENDATIONS

- Need to shift both team and organization structures as well as cost tracking from an individual project focus to a product (application)-based delivery structure where persistent teams are formed around products or solutions.
- Establish a Bimodal IT governance model which will ensure that the entire IT department gains the appropriate efficiencies and improvements from this product-focused change in delivery structure.
- Look at the entire software development team and their related costs on a holistic basis so certain investments can be made for a product or capability that otherwise would be impossible under an individual project-based view.
- Clearly define business outcomes at the capability level so investments can be directly tied to actual improvements made. Capabilities are not the work people do ("We send a customer an invoice requesting on-time payment") or how they actually do it ("We check the order against our invoice. Then we call the customer to ensure invoice has been received."), but instead the focus is on the crucial purpose or outcome that is desired ("bill customer" or "collect customer payment"). This will provide direct tangible evidence of the returns generated from the investments made.
- Educate your internal stakeholders that software development costs for internal-use software can be capitalized under an agile environment. While the original accounting guidance was developed under waterfall development methodologies as long as the requirements outlined in the accounting standard are followed, one can capitalize the appropriate application development related costs.

INTRODUCTION

Product or capability based investing is critical within an agile software development environment as it allows an IT Department to move away from individual project based funding which is often done in isolation. By broadening the landscape to capture the necessary capability or product for an IT Department, the appropriate investments and related returns on those investments can be made. To move from a historic project-based mindset to one focused on capabilities or products, an agile methodology and bimodal governance model are necessary.

An additional benefit of having a well-functioning agile software development persistent team that rapidly deploys working tested software is the amount of capitalization of expenses for internal-use software can be substantially higher than under a waterfall environment. Because the impediments and administration burden have been substantially reduced within an agile environment, the transformed persistent development team can focus their efforts on what they were hired to do – building innovative software that solves a company's business problems.

BIMODAL IT GOVERNANCE

To move from a project-oriented focus to a product or capability one, an organization will need to undergo a business transformation. This transformation process should develop a governance structure that encompasses both Mode 1 and Mode 2 styles of work. Being bimodal requires that both work modes are united and share a common vision and strategy - they cannot operate independently. Instead, organizations need to approach solving a company's business problems by bridging the Mode 1 predictable evolution of products and technologies with the Mode 2 exploration and speed in solving the unknown.

While Mode 2 work is generally viewed as being agile and where experimenting to solve new problems would occur, companies must innovate and build capabilities within both Modes. Otherwise, certain aspects of the company's infrastructure and operations will become impediments to not only Mode 1 but also Mode 2 activities.

Once an organization has developed their bimodal IT governance, it is important that improving a company's long-term agility be a key stalwart in their strategy. Being agile and able to innovate in rapid fashion allows teams to focus on business issues and not be bogged down with administrative requirements. In turn, the various bottlenecks that existed are also minimized so the core work being completed is solving business problems. Within the software development teams that are transformed to persistent teams, they will be focused on developing working tested software and given the majority of their time will be spent on those activities, a company will have the ability to capitalize more of these costs as internal-use software.

PRODUCT AND CAPABILITY-FOCUSED MANAGEMENT

The historical approach to developing software within an IT software project framework is at a point of inflection. This is due to numerous factors including the formal annual funding of projects, how projects have created duplicate efforts within organizations and, as described above, more organizations are moving into bimodal environments where more work will be done within Mode 2 structures.

There has been a growing level of dissatisfaction with project-driven funding because the annual budgeting process and eventual completion of that project takes too long. It can take years to finally solve the needed business issue within this funding approach, yet in this highly competitive business environment where rapid changes are needed to survive, this is unacceptable and alternatives such as product oriented pool funding must be evaluated.

Another overhang for project-driven approach companies where large organizations have been built up through silos (i.e. customer segments, regions, departments, etc.) or from mergers & acquisitions is that they have created several different software applications that support a portion of the same business process, if not in some cases the entire application. This redundancy in software applications has created unnecessary technical debt burdening an IT organization where it cannot invest in new endeavors. A capability based funding model creates the appropriate solution to solve this problem.

Traditional waterfall and related incremental methods are primarily project delivery methods in which a project is a collection of business functionalities – it is not usually a cohesive, complete solution. Projects have starts and endings and generally when it is time for the next-gen of that project, the project team starts anew on the entire project by obtaining business requirements, defining the end solution, etc. all over again. As a result, projects are transient in nature or viewed as one-off events.

This is in contrast to the view of a product or application focus. Products are viewed as long-lasting sets of business capabilities where they are constantly fed and nurtured with continual updates. To support this product lifecycle, iterative and agile methods are designed for product delivery methods where a certain product is released into production on a regularly scheduled basis. Upon moving to a product-based delivery structure, a substantial amount of organizational change will be required, not only to the applications team but the way the software is viewed and used by the business. This will need to happen on several levels – in the tracking of costs, the timing of deployment for working tested software and building capability roadmaps.

Moving from focusing on projects to products will allow the business to move away from individual project costing to a higher level of analysis – either at the product level or overall department. By expanding the relevant cost pool and those involved in the corresponding processes will afford a company the opportunity to capitalize more operating costs as long as they fall within the categories of capitalization under the appropriate accounting guidance.

With project-based funding, a company may not be able to make the necessary big investment decisions an organization needs to substantially change their business operations. Instead with the myopic view of a project, the funding and expected returns are laser-focused on that specific project. Taking a broader vantage point with a product-based approach, which may expand over 18 months because of the life cycle mentality, a company can make the appropriate investment decision and fund what is necessary to create that game changer the organization desperately needs to remain competitive in the marketplace.

The timing of deployment for software within a product-based delivery system it is important to establish a demand management agile team that can package and prioritize work at the Epic level. Epics are the source of funding and costs are tracked against their outcomes. Epics and the further broken down stories are the main drivers of planning within a product-based delivery model. Because agile projects are estimated more and may have a greater range in the final cost than waterfall projects, having the appropriate tracking of these costs is imperative. It should be noted that one can capitalize software development costs at either the sprint level or upon completion of each Epic. Sprints are either two to four weeks in length while an Epic may span four to six months.

> EXPANDING THE RELEVANT COST POOL CREATES OPPORTUNITIES TO CAPITALIZE MORE OPERATING COSTS

A business capability roadmap is not a traditional list of features and dates, but rather a sequence of business outcomes

to be achieved. This roadmap will change frequently as the business uses the software and more is learned about the product needs. It is necessary to tie this to the product-based delivery system where business benefits are deployed on a regular basis based on necessary business capabilities and outcomes. The business outcomes for each capability of developed software must provide a clear line of sight to tie investments to type of improvements made.

EFFECTIVENESS OF SOFTWARE DEVELOPMENT TEAMS

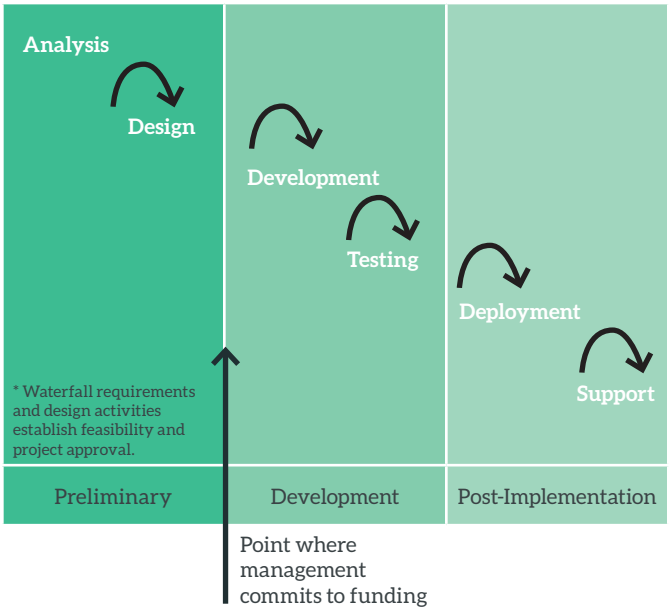
Within an agile environment where persistent cross functional teams operate efficiently, the velocity of software development is significantly more rapid than historical waterfall environments so more costs can be capitalized because there are less administrative tasks and distractions that the development teams have to spend their time on. Studies showed that some non-persistent, non-agile development teams were spending on average 50% or more of their time on administrative tasks or sitting idle because of an upstream bottleneck. Within performing agile teams, they spend most of their time on software development. The highest performing teams are spending almost 90% of their time on true software development and are not be bogged down with admin or other burdens. And by spending more time on software development and not administrative tasks or sitting idle, more of these team's costs can be capitalized as internal-use software.

If a company is able to capitalize costs from efficiently developed software there are two benefits an organization gains. One benefit is that the overall cost for that solution is substantially less so that the future amortization expense will also be substantially less. The second benefit is more projects are able to be capitalized each year because they are being completed in rapid fashion. By having a higher amount of current capitalization and lower future amortization costs, these two benefits will reduce a company's total operating costs allowing the organization to invest more in the business to solve more of their business issues.

ACCOUNTING FOR INTERNAL-USE SOFTWARE

The guidance for accounting for internal-use software in the FASB's Accounting Standards Codification (ASC) 350-40, Accounting for Internal-Use Software, outlines how companies should capitalize or expense internal-use software based on the project stage of the development effort as well as the type of actual work being performed.

The original accounting guidance for internal-use software is based on the AICPA's Statement of Position (SOP) 98-1, Accounting for the Costs of Computer Software Developed or Obtained for Internal Use, and was issued in August 1998. As one can imagine, during the time of SOP 98-1's issuance the waterfall methodology was predominantly, if not exclusively, used – the Agile Manifesto was not signed until February 2001. Because waterfall was so dominant in practice, the accounting standard followed that development model's tight structure and laid out activities that should happen within a prescribed stage as detailed below:



SOP 98-1 and now ASC 350-40's three distinct stages dictate how costs incurred by an organization should be treated – either expensed or capitalized. The three stages and their general accounting treatment are:

PRELIMINARY PROJECT STAGE

The Preliminary Project Stage is when an organization is evaluating software solutions – either off-the-shelf or internally developed. Internal and external costs shall be expensed as incurred. Notable activities include:

- Analyzing strategic decisions focused on allocating resources between alternative projects
- Determining performance and system requirements for each project
- Evaluating vendor solutions based on vendor demonstrations and other vendor provided materials
- Exploring alternative means of achieving performance requirements (i.e. buy vs. build analyses)

- Selecting vendor and/or consultant in installation or development of chosen solution

When the project feasibility has been completed and funding has been approved for development, the Preliminary Project Stage is complete.

APPLICATION DEVELOPMENT STAGE

The Application Development Stage is when an organization is developing and installing the approved software solution. Internal and external costs incurred are generally capitalized; however, certain tasks are expensed.

- Capitalizable tasks include design of software configuration and interfaces, actual coding, installation of purchased software and testing of software.
- Tasks that should be expensed include training, data conversion (unless you need a tool to assist in the data conversion process the costs for that tool can be capitalized) and administrative & overhead.
- The Application Development Stage is completed after final user acceptance testing in production and responsibility has transitioned to the Operations team.

POST IMPLEMENTATION/OPERATION STAGE

The Post Implementation/Operation Stage is when an organization is providing support for deployed software, training on software and maintaining the software. Internal and external costs should be expensed as incurred. Activities include:

- Capitalizable Completing documentation on procedures
- Conducting formal training
- Certifying operational systems
- Preparing post-implementation reviews including analysis of project
- Completing routine maintenance activities such as bug fixes, security reviews and refactoring code

Preliminary Project Stage	Application Development Stage	Post-Implementation / Operation Stage
Conceptual formulation of alternatives	Design of chosen path, including software configuration and software interfaces	Training
Evaluation of alternatives	Coding	Application maintenance
Determination of existence of needed technology	Installation of hardware	
Final selection of alternatives	Testing, including parallel processingy phase	

Delving deeper into the details, ASC 350-40-55-4 provides further clarification regarding the stages and the type of activities that generally happen in each stage. The standard is quite specific in stating that "the development of internal-use computer software may not follow the order shown in the preceding list. For example, coding and testing are often performed simultaneously. Regardless, for costs incurred subsequent to completion of the Preliminary Project Stage, the guidance shall be applied based on the nature of the costs incurred, not the timing of their incurrence."

Based on this explicit guidance from the FASB there are several items for consideration. First, there is no requirement that there is a set stage where everything has to follow a certain specific step, as is required within the waterfall methodology. This means that performing software development within an agile methodology which is iterative and incremental in nature, where coding and testing are happening concurrently, is allowed for capitalizing software development costs versus expensing them immediately. Following waterfall's prescribed structure is not a requirement – there is latitude within the accounting guidance.

The second item for consideration is the clear delineation of the completion of the Preliminary Project Stage. This is a key milestone that has to be completed before any expenses can be capitalized – it is not optional and must occur for every single development effort. ASC 350-40-25-12 states that "Capitalization of costs shall begin when both of the following occur:

a) Preliminary Project Stage is completed; and

b) Management, with the relevant authority, implicitly or explicitly authorizes and commits to funding a computer software project and it is probable that the project will be completed and the software will be used to perform the function intended."

There is no ambiguity with either of these requirements. There is a certain level of diligence for a software development effort that has to be completed to eventually capitalize future development costs – this would include formulating and evaluating each of the alternative solutions and ensuring that a technological solution exists for the business problem at hand. If one cannot prove that this work has been completed, they'll remain in the Preliminary Project Stage until this work has been done. After diligence has been completed, the final step is obtaining the funding and appropriate authorizations from management. Ensuring these approvals are obtained and documenting them in advance of capitalizing costs will be important for a company's internal and external auditors.

CAPITALIZATION OF COSTS

Now that the Preliminary Project Stage has been completed and the necessary authorizations obtained, capitalizing software development costs can begin. There are several ways to track costs for capitalization. With most organizations having implemented time cards for their waterfall methodologies, using these already deployed time tracking systems will work within an agile format. However, there are considerations in using time cards:

- Existing time tracking systems should reduce additional training on how to use the system.
- Should be relatively intuitive to development team members.
- May force unneeded tasks to be broken down into smaller increments of time management.
- Productivity could be impacted if done accurately each day which is counter to the agile and scrum philosophy.
- Inaccuracy occurs if productivity is not impacted because team members are estimating their time at the end of each week.
- May create higher level of overhead in administration and tooling and not gain much benefit in return.

An alternative way to tracking costs for capitalization is using story points and assigning a dollar value for each story point completed during a sprint. Considerations include:

- There is potentially a higher correlation with a team's effort such that with the higher amount of features completed the higher level of capitalization.
- Because story points are already used in a scrum environment, this is a natural way to track activities which in turn tracks costs.
- Story points are subjective for each team and individual team members. Aligning consistency across every development team may prove challenging.
- The use of story points is based on an estimate of costs which certain auditors may not gain comfort with. Ensuring their approval of this process is obtained before implementation will be very important. Trying to recast history under a different cost tracking structure may prove challenging.
- Our experience in capitalizing software development costs is a good portion of companies that have transformed their development teams are capitalizing 70% of these costs. The 70% level is based on a cohesive, persistent team remaining together inclusive of holidays and other administrative tasks. This differs than a project-based environment where teams are continually formed and broken down and reformed to work on each specific project which adds to idle time or waste. Certain customers have become so agile and focused on producing working tested software that during the actual software development process they are capitalizing 90% of their development costs. While this is more of an extreme it proves that high performing software development teams can make direct impacts to a company's operating income.

Capitalization of these development costs "shall cease no later than the point at which a computer software project is substantially complete and ready for its intended use, that is, after all substantial testing is completed (ASC 350-40-25-14)." Within an agile environment after the last story is written and deployed, the capitalization effort should cease.

AMORTIZATION

Once software development costs have been capitalized and the asset is placed in service for its intended use, these costs will need to be amortized over the useful life of the software. ASC 350-40-35-6 provides guidance for the appropriate amortization requirements.

The capitalized software development costs will need to be amortized over the useful life of the software. Due to the rapid changes that happen within technology, these useful lives should be short in nature – generally 3 to 5 years is most common. However, certain deployments like a company's ERP system could be amortized over a longer time horizon because to change out such a system is very expensive and would impact the entire company. So a period of 7 or 8 years may be appropriate. All of these are estimates and are based on management's judgment.

Within a waterfall environment, amortization would begin when the entire project is completed. This is not the case for agile methods. Some agile companies opt to amortize their costs each month with the release of each sprint because with each sprint they have produced working tested software. So if a software solution is deemed to have a 48-month useful life, the first release of that solution will be amortized over 48 months. The second release will be amortized over 47 months and the third release will be over 46 months. Other agile companies will capitalize costs over each individual Epic and then amortize those costs over the appropriate useful life of that individual Epic. Generally, each Epic will take 4 to 6 months to complete. If there are 4 different Epics as part of the total software solution, each Epic will have 48 months of amortization expense.

One should keep a watchful eye on the amount of costs being capitalized so they don't create too much technical debt for an organization. There is a rule of thumb that a company's combined IT depreciation and amortization expenses should not exceed 20% of a company's total IT budget. If too much in costs are capitalized in today's environment that will later need to be amortized, the future will have to bear these costs which may limit funding of needed projects at that time. So this has to be a constant balancing act that the IT and Finance team members will need to work through together.

ACCOUNTING POLICIES & PROCEDURES

Creating the appropriate accounting policies and procedures around all of these requirements will be necessary for your capitalization efforts. It is important that these policies and procedures be simple to deploy and scalable to grow with an organization. Using the mantra of simplicity will provide great benefit. If it becomes overly complicated in the tracking of costs and becomes too burdensome on the organization as a whole, any cost savings from capitalizing certain costs will be lost in adverse impacts to productivity, decline in associate morale or even associate turn-over as they look for alternative work environments that are not as cumbersome to work in. The final documented and auditable policies and procedures need to be consistent to work within both Mode 1 and Mode 2 work environments.